Dissertation

Pre-emptive Modeling of Concurrent and Distributed Systems

by

Gerald Lüttgen

submitted to

Fakultät für Mathematik und Informatik
Universität Passau

November 1997
(revised in May 1998)

Abstract

Process algebras constitute a widely studied framework for modeling and verifying *concurrent systems*. Such theories typically consist of a simple language with a well-defined operational semantics given in terms of *labeled transition systems*; a *behavioral equivalence* is then used to relate implementations and specifications, which are both given as terms in the language. In order to facilitate *compositional reasoning*, in which systems are verified on the basis of the behavior of their components, researchers have devoted great attention to the definition of *behavioral congruences*, which permit the substitution of "equals for equals" inside larger systems.

Traditional process algebras focus on modeling the potential *nondeterminism* that concurrent processes may exhibit. They abstract away the details such as *time*, *priority* and *data* from implementation. However, many of these abstractions lose too much information and, thus, complicate the accurate modeling of important system behavior. Consequently, approaches have been suggested for re-introducing sensitivity to some of the mentioned aspects of system behavior, especially to priority and time. Both of these phenomena deal with *pre-emption* since some transitions may have precedence over others. In temporal process algebras this is due to the *maximal progress assumption* which prevents clock ticks whenever the system under consideration can engage in internal computation. Most of these extensions, however, have been devoted to modeling centralized, as opposed to distributed systems; the real-time work, in particular, has focused on systems with a single clock. The centralized character of these approaches is reflected by the choice of a *global* notion of pre-emption in the underlying semantic theories. Whereas this is adequate for modeling concurrent systems in uni-processor environments, it is too restrictive for achieving an intuitive semantics for distributed systems.

The contribution of this dissertation is to investigate the deep connections between priority and time, as embodied in the notion of pre-emption. The first part considers and improves on classical approaches to priority and time in process algebras based on global pre-emption. Two medium-scale case studies dealing with a safety-critical railway signaling system and the SCSI-2 bus-protocol testify to the importance of incorporating these aspects in process algebras for system modeling and verification. Moreover, it is formally shown that time can conceptually be understood as a *dynamic* variant of priority. In the second part of the thesis the shortcomings for reasoning about existing theories with respect to *distributed* systems are pointed out and, subsequently, are eliminated without sacrificing the simplicity and the elegance that have led to their success. Therefore, process algebras with priority and time are developed incorporating a *localized* notion of pre-emption which allows one to keep the abstract view of process algebras as far as possible but has far-reaching consequences for the semantic theories. Technically, the considered and developed calculi extend the well-studied process algebra CCS by priority and time. The presented semantic theories are based on the notion of bisimulation and include abstractness results as well as axiomatic and logical characterizations of the behavioral relations. Generic examples illustrate the practical utility of the approaches to system modeling and verification.

Acknowledgments

First of all, I would like to thank my supervisor Prof. Bernhard Steffen for introducing me to the broad area of formal methods and for showing me the beauty of algebra in Computer Science. Numerous inspiring discussions with him convinced me to leave the Technical University of Aachen after my masters and to follow him to the University of Passau, located at a lovely place in Lower Bavaria, for pursuing a Ph.D. During the following three years it was a pleasure to work with Prof. Bernhard Steffen; I profited a lot from his scientific experience and criticism. His generous financial assistance made it possible for me to attend several international workshops and conferences, including the summer school in Marktoberdorf.

Thanks to the support from Prof. Bernhard Steffen, Prof. Rance Cleaveland, Prof. Christian Lengauer, Prof. Klaus Indermark, and the German Academic Exchange Service (DAAD; Doktorandenstipendium HSP II/AUFE) I had the opportunity to visit North Carolina State University (NCSU) for a full year. My stay in North Carolina proved to be very fruitful for my academic career. This was largely due to the patient guidance, enlightening discussions, and friendship of my host Prof. Rance Cleaveland. I am also indebted to his students, especially Dr. V. Natarajan, Dr. Girish Bhat, and Dr. Steve Sims, with whom I shared many research interests. I acknowledge the Department of Computer Science at NCSU for providing a pleasant working environment. Special thanks to Dr. Girish Bhat, Zeynep Dayar, Dan DuVarney, Jatin Hansoty, Murali Narasimha, Dr. V. Natarajan, and Pranav Tiwari for hosting me during my following visits to NCSU.

I am grateful to Dr. Michael Mendler for guiding me through research after my return to Passau and also for critically reading drafts of this thesis. Many thanks to the Fakultät für Mathematik und Informatik at the University of Passau, especially to Prof. Winfried Hahn, whose support made it possible for me to finish this dissertation as planned. I also benefited from many discussions with Can Albayrak, Dr. Michael von der Beeck, Volker Braun, Dr. Andreas Claßen, Dr. Alfons Geser, Prof. Purushothaman Iyer, Dr. Peter Kelb, Marco Kick, Dr. Tiziana Margaria, Dr. Markus Müller-Olm, Dr. Thomas Noll, and Dr. Carsten Weise. Dr. Thomas Noll deserves special mention for carefully proof reading many parts of this dissertation.

As a Ph.D. student one spends (too) many hours in office. Fortunately, this turned out to be great fun thanks to my office mates Dr. Girish Bhat, Neerja Bhatt, Dr. Ufuk Çelikkan, Dr. Andreas Claßen, Zeynep Dayar, Dan DuVarney, Dr. Peter Kelb, Dr. Jens Knoop, Dr. Michael Mendler, Brad Mott, Murali Narasimha, and Pranav Tiwari. Last but not least I want to thank my parents, family, and friends for their ongoing encouragement and support, and for reminding me that there is more to life than Computer Science.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Our every-day life is heavily influenced by many kinds of computer systems ranging from simple systems, e.g. for modern kitchen aids and TV-sets, to more complicated systems, such as phone services, to complex systems as used for air-traffic or nuclear power-plant control. Well-trained software engineers usually prove the quality of their products by extensive testing. However, the high complexity of today's computer systems and the fast changing markets prevent this testing from being exhaustive. Thus, a certain risk of failures of these systems remains. Often a failure occurs in unforeseen situations which happen very rarely and, therefore, are usually not detected during the testing phase. Although failures often are well-calculated, sometimes unreliability cannot be tolerated. This is especially true for critical applications such as in medicine or aviation, where a failure of a system may cause the loss of human lives. Unfortunately, too many disasters have happened in history and the recent past which are due to malfunctioning computer systems. Hence, testing itself is not enough to ensure the *correctness* of systems within reasonable bounds.

A precise analysis and verification is needed in order to answer the question if a system meets its specification. This requires a solid mathematical framework for modeling and reasoning, which has to be rich enough to reflect the behavior of real-world systems accurately. *Formal semantics* and *formal verification* are the areas of Computer Science which are concerned with the mathematical foundations for modeling and reasoning about aspects of system behavior [7, 115, 170].

In the past, the semantics of *sequential programs* has been extensively studied. Usually, it can be expressed by a partial function which maps input values to output values. Thus, a sequential program, which is desired to terminate, can be viewed as a transformer. The semantics of *concurrent* and *distributed programs* is another area of intensive research. The importance of distributed systems is steadily growing. As one of the examples, one may think of networks like the Internet which allow us to share information across the world. Another example are transputers where a key motivation for distribution is the efficiency-gain due to parallelization of programs [113]. Many concurrent programs like operating systems or communication protocols cannot be adequately given a semantics by their input/output behavior since they are not designed to terminate. Instead, their

semantic behavior can be characterized by their ongoing interaction with the environment, which is why these systems are called *reactive systems* [142].

1.1 Traditional Process Algebras

Process algebras, e.g. CCS [125], CSP [94], ACP [16, 17], and SCCS [123], provide an *algebraic* approach for modeling and reasoning about reactive systems [8, 52, 64, 65]. Their abstractness makes them well-suited for analyzing semantic phenomena such as *nondeterminism*. Process algebra can be understood as a mathematical base for defining programming languages. They intentionally ignore important programming language concepts like data structures, but concentrate on modeling the communication behavior of reactive systems. A process algebraic theory typically consists of a simple language with a well-defined semantics given as *labeled transition systems* that are defined via structured operational rules in the sense of Plotkin [141] and provide a basis for many existing analysis and verification methods. In order to verify that an implementation satisfies its specification, often *behavioral equivalences* are used. Intuitively, an implementation and a specification are related by the equivalence under consideration whenever they share the same properties. Usually, a specification can be expressed by a simple "sequential" process whereas an implementation consists of a much more complex concurrent or distributed system involving auxiliary synchronization actions. Therefore, the behavioral equivalence has to abstract from internal computation and concentrate on the external behavior of systems.

A variety of such behavioral equivalences have been proposed along the following design criteria: an appropriate equivalence should be (1) finer than *trace equivalence*, (2) *deadlock* sensitive, and (3) *compositional*. The absence of deadlocks is a desired property of reactive systems and is of the same importance as "termination" for sequential programs. Thus, a behavioral relation should never relate a deadlock-free specification with a deadlock-prone implementation. Compositionality is of utmost importance for any behavioral relation since it enables a modular verification of systems. This means that the behavior of a system can be analyzed based on the behavior of its components, i.e. the principle "substituting equals for equals" is valid. A coarsest behavioral equivalence satisfying these criteria is *testing equivalence* [72] developed by DeNicola and Hennessy. Unfortunately, testing equivalence is P-SPACE hard to compute [55]. An efficiently computable [103, 138] but very fine equivalence satisfying the above criteria is *bisimulation equivalence* which has originally been introduced by Park [139] and subsequently been used by Milner for defining a behavioral equivalence for his process algebra CCS [125] (*Calculus of Communicating Systems*). It has been proved useful for implementing verification tools [64, 65, 156, 159] and for system verification [76, 125, 155].

Hennessy and Milner have done pioneering work analyzing the concept of bisimulation and characterizing this behavioral relation algebraically and logically within the process-algebraic framework of CCS [125]. The *algebraic characterization* is done by means of an axiom system which consists of equations that are sound and complete with respect to

bisimulation. Then, two terms in the algebra are bisimular if and only if one term can be rewritten to the other by applying the equations. Some researchers doubt the value of such an axiomatization because proving the correctness of systems by manipulating equations by hand is not practical in view of the complexity of real-world systems. Moreover, there are theoretical boundaries preventing the automation of this job. For us, the primary reason for providing axiomatizations of behavioral relations is to testify to the mathematical tractability of the presented semantic theory. Regarding a *logical characterization* of bisimulation Hennessy and Milner have developed a temporal logic with the following property: two processes are bisimular if and only if they satisfy exactly the same formulas [125]. Enlarging the expressiveness of their logic by introducing maximal and minimal fixed point constructs yields the well-known modal μ-calculus [109, 160]. Efficient algorithms, called *model checkers*, have been developed in order to check if a system, encoded as a labeled transition system, satisfies a given formula [77, 155]. This approach to program verification is by now the most successful one since it can be handled by engineers. In practice, the formalization of desired properties of the system under consideration is relatively easy in several modal logics, and the model checking itself can be done in a fully automatic fashion for finite-state systems [25, 26, 49, 50, 56, 66] and classes of infinite-state systems [40, 41, 42].

Consequently, the contribution of process algebras for formally reasoning about concurrent and distributed systems stems from their utility to conveniently modeling systems whose properties are of interest. The semantics of this description, a labeled transition system, can directly be taken as a representation on which verification methods work. This makes process algebras such a distinguished semantic framework for modeling and verifying important aspects of system behavior.

1.2 Shortcomings of Traditional Approaches

Although many case studies [8, 52, 64, 65] prove the practical utility of the process algebraic approach to system modeling and verification, many systems in practice cannot be modeled accurately within this framework. One reason is that traditional process algebras focus on modeling the potential nondeterminism that the interplay of concurrent processes may exhibit. They do not provide any mechanisms, except some simple but insufficient form of *synchronization* via communication, that allow one to *restrict nondeterminism*. However, most concurrent and distributed systems are composed of sequential processes running in parallel, which themselves behave deterministically. This deterministic behavior is often due to *real-time* and *qualitative timing constraints* which are imposed on systems. As a particular instance of qualitative timing constraints one can think of *interrupts*: "Whenever an interrupt request is raised, the responsible interrupt handler should react on it *immediately*." Another example for qualitative timing constraints concerns forced *synchronizations* in which all or some processes of a system have to take part.

It is worth emphasizing that this thesis considers aspects of *qualitative* time as well as of *quantitative* time, often called *real-time*. A quantitative view of time is suitable for rea-

soning about timing issues of low-level implementations of soft- and hardware, such as *hard deadlines* and *response times*. It requires a system designer to give precise and complete timing information. In real-time programming, however, timing information often is only introduced in order to substitute poorly understood dependencies caused by phenomena like interrupts and other causality constraints [163]. Because of this observation Wirth has suggested to eliminate quantitatively specified delays in high-level programming languages by introducing explicit synchronizations via signals [172]. In this thesis, we employ Wirth's philosophy of time, which is referred to as qualitative time in the sequel, for abstract system specifications.

The above mentioned forms of timing constraints for reducing nondeterminism are simply abstracted away when using traditional process algebras in which only nondeterministic choices can be expressed. As a consequence, the resulting model of the system under consideration is often not faithful since it contains paths that cannot be traversed by the real-world system itself. In particular, this implies that verification results obtained for the model, especially concerning *liveness* properties, may not be valid for the real-world system.

Recently, approaches have been suggested introducing sensitivity to aspects of system behavior which enables one to restrict potential nondeterminism. In this light, the most relevant semantic aspects are *priority* [10, 13, 24, 37, 38, 43, 54, 60, 79, 102, 110, 116, 134, 133, 154] and *time* [5, 6, 92, 45, 128, 136, 135, 173]. Priorities allow one to give some transitions in an underlying semantic model precedence over others, thereby reducing nondeterminism. A typical example are interrupt requests where "standard" transitions at a state are preempted, i.e. "cut off," whenever an interrupt is raised. Another example for the utility of priorities is modeling certain programming language constructs, which impose an order on choices, e.g. the `PRIALT` construct in `occam`. Also synchronization constraints, which can often be expressed via the concept of clocks or time, restrict the potential of nondeterminism. For instance, in communication protocols one may only switch from one protocol phase to another if all (or some particular) components of the system agree. This eliminates the nondeterminism due to the pure *interleaving* of different phases which may arise when no synchronization constraints are imposed. However, most approaches to temporal process algebras have devoted their attention to quantitative time measured by a single, global clock. On the one hand, this enables reasoning about the timing behavior of a modeled system. On the other hand, system designers must provide absolute timing information which may not be available at early design stages but only at low-level implementation stages. Nevertheless, system behavior is often constrained by required synchronizations of processes which can be conveniently expressed by abstract, qualitative timing constraints as illustrated above. For studying qualitative timing constraints, real-time process algebras are too complicated whereas untimed process algebras are simply inadequate.

Up to now, most extensions of traditional process algebras have been devoted to modeling *centralized*, as opposed to *distributed* systems. In approaches to priority the precedence of some transitions over others is implemented by introducing a *global* notion of

pre-emption: whenever a process may engage in a high prioritized communication, low prioritized transitions must not be executed. This semantic notion of pre-empting transitions is well motivated for centralized, uni-processor systems. However, when considering distributed systems, the high prioritized transition may be internal to one processor whereas the pre-empted low prioritized transition might be scheduled on a different processor. This is an undesired and unintended side effect because of the following reasons. First, it requires the existence of some global convention of the meaning of priority values which contradicts the philosophy of a *modular* development of distributed systems. Second, even worse, one process scheduled on a particular processor has to realize if some other process, possibly scheduled on a different processor, engages in a high prioritized communication. This communication may be internal to the second process and, therefore, unobservable for the first one. Hence, a process algebra with a global view of pre-emption can hardly be implemented within a distributed environment. A similar observation is true for temporal process algebras. Here, researchers have implicitly or explicitly focused on systems with a *single* clock which is suitable if the underlying measure of time is absolute. However, when thinking of computer networks where each computer possesses its own, local clock, it is crucial to allow multiple clocks for modeling. The notion of pre-emption exists in temporal process algebras also since processes which wait for some communication partner are assumed to stop waiting and engage in the communication immediately when the communication partner becomes ready. This assumption is referred to as *maximal progress assumption* [92, 173] or *synchrony hypothesis* [21]. In a distributed system with multiple clocks this global view of maximal progress, and consequently of pre-emption, is no longer adequate since a process, that is connected to a local clock, cannot observe if another process, attached to its own local clock, engages in a communication at the very same moment. The reason is that local clocks are hidden to processes in the environment. Hence, a global version of the maximal progress assumption is not implementable within a distributed environment, either.

Summarizing, finding the right level of modeling abstraction in process algebras is a difficult task. On the one extreme, the lack of capabilities to express priorities and timing constraints makes models of real-world systems too abstract and, thereby, inaccurate for system analysis and verification. On the other extreme, traditional extensions of process algebras by global priority and real-time restrict distributed system behavior too much.

1.3 Contributions and Organization of this Dissertation

The aim of this dissertation is to shed light on the semantic concept of pre-emption and to remedy the above mentioned shortcomings of existing theories without sacrificing the simplicity and the elegance that have led to their success. Thereby, we achieve our main objective, i.e. making process algebras more attractive to system designers by enhancing their domain of applicability. Regarding the technical parts of this dissertation the reader is assumed to possess basic knowledge of process algebras, especially of Milner's Calculus of Communicating Systems [125] (CCS).

Contributions

As argued so far, lower-level behavioral concepts such as priority and time need to be incorporated in process algebras in order to model the behavior of real-world systems accurately which is of utmost importance for verification and implementation purposes. Therefore, notions of priority and time have been introduced to several process algebras, including CCS. Herewith, the concept of pre-emption has become part of process-algebraic semantics. The various aspects of pre-emption together with their semantic implications are in the center of our interest.

The most popular process-algebraic approach to priority has been developed by Cleaveland and Hennessy [54]. However, their work is restricted to a two-level priority-scheme and incorporates *prioritization* and *deprioritization operators*. On the one hand, these restrictions are convenient for establishing elegant algebraic results regarding the underlying pre-emptive semantics, while on the other hand, they are not helpful in practice since the implied semantic theory based on Milner's notion of weak bisimulation [125] is too restrictive for reasoning about concurrent systems. We improve Cleaveland and Hennessy's algebra by by adopting a multi-level priority-scheme and also by leaving out the mentioned prioritization and deprioritization operators. Subsequently, we re-investigate the bisimulation-based semantic theory for the new algebra, referred to as $\mathsf{CCS}^{\mathsf{ch}}$, by introducing an adequate notion of prioritized weak bisimulation and prioritized observational congruence. The technical results obtained for $\mathsf{CCS}^{\mathsf{ch}}$ include abstractness results – a kind of result which is rarely verified in other work – as well as operational and logical characterizations of prioritized weak bisimulation which allow its efficient computation and enable one to adapt temporal logics for $\mathsf{CCS}^{\mathsf{ch}}$, respectively. The utility of our approach to priority is demonstrated by a real-world example dealing with the design of a safety-critical network that is part of a railway signaling system [69]. The case study shows that priorities in process algebras support an intuitive modeling of concurrent systems since undesired interleavings can be suppressed due to the pre-emptive semantics. This fact also leads to a substantial reduction of the sizes of models. We have implemented $\mathsf{CCS}^{\mathsf{ch}}$ as a new front-end for the *Concurrency Workbench of North Carolina* [65, 153], `CWB-NC`, and we use model checking for verifying properties of the signaling system.

Intuitively, priority and time are related aspects of system behavior since priorities often play an essential role for implementing scheduling mechanisms. The basic idea for relating priority and time is that priority values may be considered as *time-stamps*. In the present thesis we give a precise formalization of this idea of time-stamping which relates process calculi with priority and time. We show that in contrast to existing approaches to priority, e.g. $\mathsf{CCS}^{\mathsf{ch}}$ which adopts a *static* notion of priority that is suitable for expressing interrupts and certain programming language constructs, a *dynamic* notion of priority, where priority values may change as the system under consideration evolves, is needed. In fact, scheduling mechanisms can be modeled conveniently in traditional real-time algebras due to their *dynamic* view of priority and time-stamps reflecting that the priority of a process which waits for accessing some system resource is continuously increased the longer it is waiting.

To this end, we introduce – up to our knowledge for the first time in the literature – a process algebra with *dynamic*-priority, called $\mathsf{CCS}^{\mathsf{dp}}$ (CCS with dynamic-priority), and establish a one-to-one semantic relationship to a version of Moller and Toft's *Temporal CCS* [128], referred to as $\mathsf{CCS}^{\mathsf{rt}}$ (CCS with real-time). This relationship, identifying maximal progress in $\mathsf{CCS}^{\mathsf{rt}}$ with pre-emption in $\mathsf{CCS}^{\mathsf{dp}}$, is made precise in terms of bisimulation semantics and temporal logics [118]. The advantage of using dynamic-priority for modeling real-time is that it drastically reduces the state space sizes of the systems in question while preserving properties of their functional behavior. This is demonstrated by formally modeling and verifying aspects of the widely-used *SCSI-2 bus-protocol*.

Traditional approaches to priority and time, including $\mathsf{CCS}^{\mathsf{ch}}$ and $\mathsf{CCS}^{\mathsf{rt}}$, use a *global* notion of pre-emption in order to give some transitions precedence over others. Moreover, time is measured with respect to a *single, global* clock. However, these design decisions are sometimes not adequate. Providing generic examples, we demonstrate that temporal process algebras using a single, global clock are too restrictive when modeling distributed systems and that a notion of global pre-emption is unsuitable when dealing with distributed priority or when considering timed settings with multiple clocks. Instead, a local view of pre-emption is more natural and leads to process algebras which are implementable as abstract programming languages within distributed environments. In this vein, we develop two particular process algebras, called $\mathsf{CCS}^{\mathsf{prio}}$ and CSA, respectively, together with their semantic theories which are again defined in terms of bisimulation [125, 139] and obey the principle of *localized pre-emption*. Whereas $\mathsf{CCS}^{\mathsf{prio}}$ extends the traditional process algebra CCS by distributed priority, CSA extends CCS by a notion of distributed time using multiple clocks. The algebra $\mathsf{CCS}^{\mathsf{prio}}$, which adapts the priority-scheme of [54], is distinguished by the design decision that it only allows transitions to pre-empt others at the same "location" and, therefore, captures a notion of localized precedence. An example testifies to the utility of our approach to distributed priority. Moreover, we extend $\mathsf{CCS}^{\mathsf{prio}}$ along two complementary directions resulting in a new process algebra, $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$. First, we show how the two-level priority-scheme of $\mathsf{CCS}^{\mathsf{prio}}$ can be generalized to multiple levels. Second, we remove the design decision of $\mathsf{CCS}^{\mathsf{prio}}$ that processes may only communicate on complementary ports having the *same* priority. These extensions allow us to formally compare our approach with work of Camilleri and Winskel [43]. The second process algebra mentioned above, the temporal process algebra CSA, is aimed at reasoning about *distributed* systems that involve *qualitative* timing constraints. It is a conservative extension of Milner's CCS [125] that combines the idea of *multiple clocks* with the assumption of *maximal progress*. CSA includes operators for binding processes to different clocks capturing local time synchronizations. Moreover, a *local* version of the maximal progress assumption is integrated which respects the scoping of clocks. Using a typical class of examples drawn from hardware design, we motivate why these features are useful for modeling and verifying distributed systems. Since our process algebras are extensions of CCS and adapt a semantic theory based on the notion of bisimulation, our technical developments extend constructions and proof methods presented in [125]. The main al-

gebraic results for $\mathsf{CCS}^{\mathsf{prio}}$ and CSA include the following: (1) abstractness results of our strong bisimulations and observational congruences with respect to the naive adaptations of strong and weak bisimulation from [125], i.e. we show that our behavioral relations are the *largest congruences* contained in the naive relations, (2) axiomatic characterizations of the strong bisimulations, i.e. we provide *sound* and *complete equational theories* or *axiomatizations* with respect to our strong bisimulations, (3) logical characterizations of the behavioral relations in terms of Hennessy-Milner logic [125], which yields a foundation for adopting model-checking techniques for system verification, and (4) operational characterizations of our behavioral relations as standard bisimulations on enriched transition systems which enable one to apply existing partition-refinement algorithms [103, 138] for their computation.

Finally, we discuss the common semantic grounds of our process algebras with priority and time based on the notion of pre-emption, as well as their differences. Moreover, we relate our work to existing approaches in specification and programming languages that are popular among engineers. This survey enables the interested reader to get first ideas for adapting our results to various settings in practice.

Organization

The body of this dissertation consists of three parts. The first part, including Chapters 2–5, re-investigates and advances existing semantic theories for priority and time in process algebras and shows their utility for system modeling and verification. Chapter 2 develops the process algebra $\mathsf{CCS}^{\mathsf{ch}}$ together with its bisimulation-based semantic framework. The following chapter deals with the case study of the railway signaling system which has appeared in [63] and a preliminary version of it in [61]. Chapter 4 describes an approach for modeling real-time systems using *dynamic priorities*. It presents the traditional-style real-time process algebra $\mathsf{CCS}^{\mathsf{rt}}$ and introduces the process algebra $\mathsf{CCS}^{\mathsf{dp}}$ with dynamic-priority. Both frameworks are formally related in terms of bisimulation semantics and in terms of modal logics. The utility of $\mathsf{CCS}^{\mathsf{dp}}$ for encoding real-time is demonstrated in Chapter 5 by modeling and verifying several aspects of the `SCSI-2` bus-protocol. An extended abstract of Chapters 4 and 5 has been presented in [27].

The second part of the dissertation includes Chapters 6 and 7 dealing with distributed priority and distributed time, respectively. In Chapter 6 the process algebra $\mathsf{CCS}^{\mathsf{prio}}$ and its bisimulation-based semantic theory are developed. Generic examples point to the flavor of the new algebra, which has been published in [58, 59]. Chapter 7, an abstract of which can be found in [57], presents the process algebra CSA, thereby applying our insights into local pre-emption gained for $\mathsf{CCS}^{\mathsf{prio}}$ to distributed time. The algebraic theory for CSA follows the lines of Chapter 6 and an example motivated from typical hardware designs emphasizes the utility of our theory.

To close the dissertation, Chapter 8 discusses differences and similarities of the semantic phenomena of global and local pre-emption regarding our approaches to priority and time. Moreover, it touches the question how our semantic insights of pre-emption can be

re-used when developing other well-founded semantic frameworks, and how they can be applied in practice. In this vein, Chapter 8 also presents concepts of priority and time in existing programming and specification languages, and investigates their relation to our process-algebraic approaches. Finally, we draw our conclusions and give directions for future research in Chapter 9.

It is worth mentioning that the key chapters of this thesis, i.e. Chapters 2–7 can be read independently, except that the contents of Chapter 3 relies on the algebra developed in Chapter 2 and that Chapter 5 uses material from Chapter 4. For notational convenience, sometimes symbols for domains and relations are overloaded in this dissertation. However, no confusion should arise since every symbol has a fixed meaning within each chapter.

Chapter 2

A Traditional Process Algebra with Priority

2.1 Introduction

Traditional process algebras focus on modeling the potential nondeterminism that concurrent processes may exhibit; approaches have also been suggested for introducing sensitivity to other aspects of system behavior, including *priority* [10, 24, 43, 54, 79, 83, 100, 102, 133, 134, 154]. The concept of user-defined priorities enables the modeling of systems in which some system transitions may take precedence over others, e.g. for modeling *interrupts* or certain programming language constructs, such as *prioritized choice* [43], which impose an order on choices. In this chapter, we investigate a variant of a process algebra with priority which originates in a paper by Cleaveland and Hennessy [54] where Milner's CCS [125] is extended by a priority mechanism. In contrast to [54] our calculus, henceforth referred to as $\mathsf{CCS}^{\mathsf{ch}}$ (CCS with priority based on Cleaveland and Hennessy), includes a *multi-level* priority-scheme but excludes the *prioritization* and *deprioritization* operators. For $\mathsf{CCS}^{\mathsf{ch}}$ we develop a semantic theory based on Park and Milner's notion of *bisimulation* [125, 139]. Whereas the adaptation of *strong bisimulation* together with its axiomatization from Cleaveland and Hennessy's approach is straightforward, the theory for *prioritized weak bisimulation* and *prioritized observational congruence*, as investigated in [133, 134], cannot be carried over one-to-one. This is due to the absence of the prioritization and deprioritization operators in $\mathsf{CCS}^{\mathsf{ch}}$, which provides additional freedom for abstracting away internal computation.

As a simple example motivating why priorities and this additional abstraction are useful, consider the system depicted in Figure 2.1. It consists of two processes, A that flips back and forth between two states and B that checks if A runs properly. Whenever B receives a `check` message it requests status information from A via the interrupt port $\underline{i}$ which in turn responds by action $\overline{\mathtt{ok}}$. Without being able to specify that a communication on $\underline{i}$ is more important than one on `back` and `forth`, the process A is able to ignore a possible communication on `check` forever. Hence, the process algebra CCS is unsuitable

for reasoning about this simple system involving priorities.

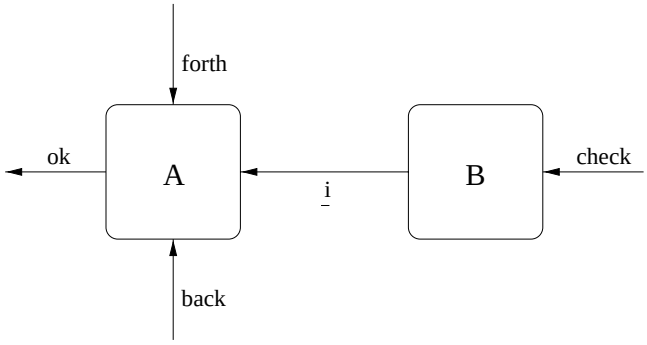

Figure 2.1: A simple example system

In contrast to CCS, which is not designed for handling priorities, one expects from the process algebra with priority considered in [54] and the corresponding bisimulation-based semantic theory [133, 134] that it is adequate for reasoning about our example system. Especially, from a notion of *prioritized observational congruence* one intuitively demands that our system is equivalent to one which abstracts from all internal computation. Unfortunately, it turns out that the congruence presented in [133, 134] is too fine in order to fulfill this expectation since it does not sufficiently abstract from internal computation. However, the version of prioritized observational congruence, which we obtain in this chapter for the modified calculus $\mathsf{CCS}^{\mathsf{ch}}$, does. Algebraically, we show that this prioritized observational congruence is the largest congruence contained in a naive adaptation of weak bisimulation [125]. We also characterize our notion of prioritized weak bisimulation as standard bisimulation on an alternative prioritized transition relation such that well-known partition-refinement algorithms [103, 138] for its computation become applicable.

The remainder of the chapter is structured as follows. The next section formally introduces the syntax and semantics of the process algebra $\mathsf{CCS}^{\mathsf{ch}}$. Section 2.3 focuses on a semantic theory based on strong bisimulation. In Section 2.4 prioritized weak bisimulation and prioritized observational congruence are developed, their algebraic properties are investigated, and it is shown how prioritized weak bisimulation can be computed. Our theory is applied to an illustrating example in Section 2.5, whereas the following section discusses the advantages of our setting compared to the one introduced in [54]. Finally, Section 2.7 summarizes this chapter.

2.2 Syntax and Semantics of $\mathsf{CCS}^{\mathsf{ch}}$

The process algebra $\mathsf{CCS}^{\mathsf{ch}}$ with priority, which we consider in this chapter, is based on the calculus proposed by Cleaveland and Hennessy in [54], where CCS [125] has been extended for capturing aspects of priority. More precisely, $\mathsf{CCS}^{\mathsf{ch}}$ modifies Cleaveland and Hennessy's calculus by introducing a *multi-level* priority-scheme and a *disabling* operator,

but disallowing the *prioritization* and *deprioritization operators*. Therefore, our process algebra is basically CCS [125] where priorities, modeled by natural numbers, are part of actions. We use the convention that smaller numbers mean higher priorities; so 0 is the highest priority value. Intuitively, visible actions represent potential communications that a process may be willing to engage in with its environment. Given a choice between a communication on a high priority and one on a low priority, a process should choose the former.

Formally, let $\{\Lambda_k \mid k \in \mathbb{N}\}$ denote a family of pairwise-disjoint, countably infinite sets of *labels* or *ports*. One may think of Λ_k containing the ports with priority k that processes may synchronize over. Then the set of *actions* $\mathcal{A}_k$ with priority k may be defined by $\mathcal{A}_k =_{\mathrm{df}} \Lambda_k \cup \overline{\Lambda}_k \cup \{\tau_k\}$, where $\overline{\Lambda}_k =_{\mathrm{df}} \{\overline{\lambda} \mid \lambda \in \Lambda_k\}$ and $\tau_k \notin \Lambda_k \cup \overline{\Lambda}_k$. The set of all ports Λ and the set of all actions $\mathcal{A}$ are defined by $\bigcup\{\Lambda_k \mid k \in \mathbb{N}\}$ and $\bigcup\{\mathcal{A}_k \mid k \in \mathbb{N}\}$, respectively. For better readability we write $\lambda{:}k$ if $\lambda \in \Lambda_k$ and $\tau{:}k$ for τ_k. An action $\lambda{:}k \in \Lambda_k$ may be thought of as representing the receipt of an input on port λ that has priority k, while $\overline{\lambda}{:}k \in \overline{\Lambda}_k$ constitutes the deposit of an output on λ. The invisible actions $\tau{:}k$ represent internal computation steps with priority k. In what follows, we use $\alpha{:}k, \beta{:}k, \ldots$ to range over $\mathcal{A}$ and $a{:}k, b{:}k, \ldots$ to range over $\Lambda \cup \overline{\Lambda}$. We also extend $\bar{}$ to all *visible* actions by defining $\overline{\overline{a}}{:}k =_{\mathrm{df}} a{:}k$. Finally, if $L \subseteq \mathcal{A} \setminus \{\tau{:}k \mid k \in \mathbb{N}\}$ then $\overline{L} =_{\mathrm{df}} \{\overline{a}{:}k \mid a{:}k \in L\}$. For the sake of simplicity, we also write $\tau \in M$, where $M \subseteq \mathcal{A}$, if $\tau{:}k \in M$ for some $k \in \mathbb{N}$. Note that our framework allows an infinite number of priority levels, although there is a highest priority. The *syntax* of CCS^{ch} is defined by the following BNF.

$$\begin{array}{lllllllllll} P & ::= & \mathbf{0} & \mid & x & \mid & \alpha{:}k.P & \mid & P + P & \mid & P \mid P \\ & & P[f] & \mid & P \setminus L & \mid & P \rangle\!\rangle P & \mid & \mu x.P & & \end{array}$$

Here f is a *finite relabeling*, i.e. a mapping on $\mathcal{A}$ which satisfies $f(\tau{:}k) = \tau{:}k$ for all $k \in \mathbb{N}$, $\overline{f(\overline{a}{:}k)} = f(a{:}k)$ for all $a{:}k \in \mathcal{A} \setminus \{\tau{:}k \mid k \in \mathbb{N}\}$, and $|\{\alpha{:}k \mid f(\alpha{:}k) \neq \alpha{:}k\}| < \infty$. Moreover, a relabeling preserves priority values, i.e. for all $a{:}k \in \mathcal{A} \setminus \{\tau{:}k \mid k \in \mathbb{N}\}$ we have $f(a{:}k) = b{:}k$ for some $b{:}k \in \Lambda_k \cup \overline{\Lambda}_k$. If $f(\alpha_i{:}k_i) = \beta_i{:}k_i$ for $1 \leq i \leq n$ and $n \in \mathbb{N}$, and $f(\alpha{:}k) = \alpha{:}k$ for all $\alpha{:}k \neq \alpha_i{:}k_i$, where $1 \leq i \leq n$, we sometimes write $[\beta_1{:}k_1/\alpha_1{:}k_1, \beta_2{:}k_2/\alpha_2{:}k_2, \ldots, \beta_n{:}k_n/\alpha_n{:}k_n]$ for f. Further, $L \subseteq \mathcal{A} \setminus \{\tau{:}k \mid k \in \mathbb{N}\}$ is a *restriction set*, and x is a variable taken from a countable set $\mathcal{V}$. Sometimes it is convenient to write $C \stackrel{\mathrm{def}}{=} P$ for $\mu C.P$ where the identifier C is interpreted as variable. In addition to the CCS operators, we include the *disabling* operator $\rangle\!\rangle$ in our language which is closely related to the corresponding operator in LOTOS [30]. We adopt the standard definitions for *sort* of a term, *free* and *bound variables*, *open* and *closed terms*, *guarded recursion*, and *contexts* (cf. [125]). The syntactic substitution of all free occurrences of variable x by Q in term P is written as $P[Q/x]$. We refer to closed and guarded terms as *processes* and use $P, Q, R, \ldots$ to range over the set $\mathcal{P}$ of processes. Finally, the binary relation $\equiv$ denotes syntactic equality.

The *semantics* of a process $P \in \mathcal{P}$ is given by a labeled transition system $\langle \mathcal{P}, \mathcal{A}, \longrightarrow, P \rangle$, where $\mathcal{P}$ is the set of *states*, $\mathcal{A}$ is the *alphabet*, $\longrightarrow \subseteq \mathcal{P} \times \mathcal{A} \times \mathcal{P}$ is the transition relation

Table 2.1: Potential initial action sets for $\mathsf{CCS}^{\mathsf{ch}}$

$\mathrm{I}^k(\alpha{:}l.P) =_{\mathrm{df}} \{\alpha{:}l \mid l = k\}$	$\mathrm{I}^k(\mu x.P) =_{\mathrm{df}} \mathrm{I}^k(P[\mu x.P/x])$
$\mathrm{I}^k(P + Q) =_{\mathrm{df}} \mathrm{I}^k(P) \cup \mathrm{I}^k(Q)$	$\mathrm{I}^k(P \rangle\!\rangle Q) =_{\mathrm{df}} \mathrm{I}^k(P) \cup \mathrm{I}^k(Q)$
$\mathrm{I}^k(P \mid Q) =_{\mathrm{df}} \mathrm{I}^k(P) \cup \mathrm{I}^k(Q) \cup \{\tau{:}k \mid \mathrm{I}^k(P) \cap \overline{\mathrm{I}^k(Q)} \neq \emptyset\}$	
$\mathrm{I}^k(P[f]) =_{\mathrm{df}} \{f(\alpha{:}l) \mid \alpha{:}l \in \mathrm{I}^k(P)\}$	$\mathrm{I}^k(P \setminus L) =_{\mathrm{df}} \mathrm{I}^k(P) \setminus (L \cup \overline{L})$

formally defined to be the least relation satisfying the operational rules presented in Table 2.2 in Plotkin-style notation [141], and P is the *start state*. We write $P \xrightarrow{\alpha:k} P'$ instead of $\langle P, \alpha{:}k, P' \rangle \in \longrightarrow$ and say that P *may engage in action* α *with priority* k *and thereafter behave like process* P'. Moreover, we let $P \xrightarrow{\alpha:k}$ stand for $\exists P' \in \mathcal{P}.\, P \xrightarrow{\alpha:k} P'$. The presentation of the operational rules requires *potential initial action sets* which are defined similar to the corresponding definition for CCS as the smallest sets satisfying the equations in Table 2.1. It is important that the potential initial action sets are independently defined from the transition relation $\longrightarrow$, so the transition relation is well-defined. Intuitively, $\mathrm{I}^k(P)$ denotes the set of all actions with priority k in which P can initially engage. This set is called *potential* initial action set since, in general, only the inclusion $\{\alpha{:}k \mid P \xrightarrow{\alpha:k}\} \subseteq \mathrm{I}^k(P)$ holds; but not conversely, e.g. $\{\alpha{:}1 \mid (a{:}1.\mathbf{0} + \tau{:}0.\mathbf{0}) \xrightarrow{\alpha:1}\} = \emptyset \subset \{a{:}1\} = \mathrm{I}^1(a{:}1.\mathbf{0} + \tau{:}0.\mathbf{0})$ since we have not taken account of *pre-emption*, yet. For convenience, we abbreviate $\bigcup\{\mathrm{I}^l(P) \mid l < k\}$ by $\mathrm{I}^{<k}(P)$ and $\bigcup\{\mathrm{I}^l(P) \mid l \in \mathbb{N}\}$ by $\mathrm{I}(P)$. Finally, we write $\alpha \in \mathrm{I}^{<k}(P)$ if $\alpha{:}l \in \mathrm{I}^{<k}(P)$ for some $l < k$. Observe that $\mathrm{I}^{<0}(P) = \emptyset$ for all processes P.

The rules in Table 2.2 capture the following operational behavior. The process $\alpha{:}k.P$ may engage in action α with priority k. The *summation operator* $+$ denotes *nondeterministic choice*. The process $P + Q$ may behave like process P (Q) if Q (P) does not pre-empt it by performing a higher prioritized internal transition. The *restriction operator* $\setminus L$ prohibits the execution of actions in $L \cup \overline{L}$ and, thus, permits the *scoping* of actions. $P[f]$ behaves exactly as process P with the actions renamed with respect to f. The process $P \mid Q$ stands for the *parallel composition* of P and Q according to an *interleaving semantics* with *synchronized communication* on complementary actions on some priority level k resulting in the internal action $\tau{:}k$. However, if Q (P) is capable of engaging in a higher prioritized internal action or in a synchronization, then lower prioritized actions of P (Q) are pre-empted. The process $P \rangle\!\rangle Q$ behaves like P and, additionally, is capable of *disabling* P by engaging in Q. The side conditions ensure that its pre-emptive behavior is consistent with that of the summation and the parallel composition operators. In practice, Q is often an *in-*

Table 2.2: Operational semantics for CCS^{ch}

Act	$\dfrac{-}{\alpha{:}k.P \xrightarrow{\alpha:k} P}$	Rec	$\dfrac{P[\mu x.P/x] \xrightarrow{\alpha:k} P'}{\mu x.P \xrightarrow{\alpha:k} P'}$
Sum1	$\dfrac{P \xrightarrow{\alpha:k} P'}{P+Q \xrightarrow{\alpha:k} P'} \tau \notin \mathrm{I}^{<k}(Q)$	Sum2	$\dfrac{Q \xrightarrow{\alpha:k} Q'}{P+Q \xrightarrow{\alpha:k} Q'} \tau \notin \mathrm{I}^{<k}(P)$
Com1	$\dfrac{P \xrightarrow{\alpha:k} P'}{P \mid Q \xrightarrow{\alpha:k} P' \mid Q} \tau \notin \mathrm{I}^{<k}(P \mid Q)$	Com2	$\dfrac{Q \xrightarrow{\alpha:k} Q'}{P \mid Q \xrightarrow{\alpha:k} P \mid Q'} \tau \notin \mathrm{I}^{<k}(P \mid Q)$
Com3	$\dfrac{P \xrightarrow{a:k} P' \quad Q \xrightarrow{\overline{a}:k} Q'}{P \mid Q \xrightarrow{\tau:k} P' \mid Q'} \tau \notin \mathrm{I}^{<k}(P \mid Q)$		
Dis1	$\dfrac{P \xrightarrow{\alpha:k} P'}{P \rangle\!\rangle Q \xrightarrow{\alpha:k} P' \rangle\!\rangle Q} \tau \notin \mathrm{I}^{<k}(Q)$	Dis2	$\dfrac{Q \xrightarrow{\alpha:k} Q'}{P \rangle\!\rangle Q \xrightarrow{\alpha:k} Q'} \tau \notin \mathrm{I}^{<k}(P)$
Rel	$\dfrac{P \xrightarrow{\alpha:k} P'}{P[f] \xrightarrow{f(\alpha:k)} P'[f]}$	Res	$\dfrac{P \xrightarrow{\alpha:k} P'}{P \setminus L \xrightarrow{\alpha:k} P' \setminus L} \alpha{:}k \notin L \cup \overline{L}$

terrupt handler. Finally, $\mu x.P$ denotes a *recursively defined* process that is a distinguished solution to the equation $x = P$. The side conditions of the operational rules guarantee that high-priority internal actions have pre-emptive power over low-priority actions. The reason that high-priority visible actions do *not* have priority over low-priority actions is that visible actions only indicate the potential of a synchronization, i.e. the potential of progress, whereas internal actions describe complete synchronizations, i.e. *real* progress, in our model.

When defining a semantic theory for CCS^{ch} we also need the (real) initial action set $\mathcal{I}^k(P)$ of a process P which includes all actions with priority k in which P can indeed initially engage in, i.e. those potential initial actions which are not pre-empted. We formally define $\mathcal{I}^k(P)$ by $\{\alpha{:}k \in \mathrm{I}^k(P) \mid \tau \notin \mathrm{I}^{<k}(P)\}$. For notational convenience we also introduce sets $\mathcal{I}^{<k}(P)$ analogously to $\mathrm{I}^{<k}(P)$ and write $\alpha \in \mathcal{I}^{<k}(P)$ if $\alpha{:}l \in \mathcal{I}^{<k}(P)$ for some $l < k$. Finally, we define initial action sets ignoring internal actions by $\mathcal{II}^k(P) =_{\mathrm{df}} \mathcal{I}^k(P) \setminus \{\tau{:}k\}$ and $\mathcal{II}^{<k}(P) =_{\mathrm{df}} \mathcal{I}^{<k}(P) \setminus \{\tau{:}l \mid l < k\}$. The following proposition states that these definitions reflect our intuition about initial actions and the notion of pre-emption. Especially,

lower prioritized transitions are pre-empted exactly when the considered process can engage in a higher prioritized internal transition.

Proposition 2.2.1 *For all $P \in \mathcal{P}$ and $\alpha{:}k \in \mathcal{A}$ we have:*

1. *$\alpha{:}k \in \mathcal{I}(P)$ if and only if $P \xrightarrow{\alpha:k}$.*

2. *$\tau \notin \mathrm{I}^{<k}(P)$ if and only if $\nexists l < k.\, P \xrightarrow{\tau:l}$.*

Proof: The second statement is an immediate consequence of the first one for $\alpha \equiv \tau$ and the fact that $\tau \notin \mathcal{I}^{<k}(P)$ if and only if $\tau \notin \mathrm{I}^{<k}(P)$, which follows from the definition of $\mathcal{I}^{<k}(P)$. Therefore, we concentrate on the proof of the first statement which is done by induction on the structure of P. For convenience we introduce the abbreviation $\mathcal{T}^k(P)$ for the set $\{\alpha{:}k \mid P \xrightarrow{\alpha:k}\}$.

- $P \equiv \mathbf{0}$: According to the definitions of $\mathcal{I}^k(\mathbf{0})$, $\mathrm{I}^k(\mathbf{0})$, and $\mathcal{T}^k(\mathbf{0})$ these sets are all empty.

- $P \equiv \alpha{:}l.Q$: Here, we obtain $\mathcal{I}^k(P) = \mathrm{I}^k(P) = \mathcal{T}^k(P) = \{\alpha{:}l\}$, if $l = k$, and $\mathcal{I}^k(P) = \mathrm{I}^k(P) = \mathcal{T}^k(P) = \emptyset$, otherwise.

- $P \equiv Q_1 + Q_2$:

$$
\begin{array}{lcl}
 & & \alpha{:}k \in \mathcal{I}^k(P) \\
(\text{def. of } \mathcal{I}^{\cdot}(\cdot)) & \iff & \alpha{:}k \in \mathrm{I}^k(P) \text{ and } \tau \notin \mathrm{I}^{<k}(P) \\
(\text{def. of } \mathrm{I}^{\cdot}(\cdot)) & \iff & (\alpha{:}k \in \mathrm{I}^k(Q_1) \text{ or } \alpha{:}k \in \mathrm{I}^k(Q_2)) \text{ and} \\
 & & (\tau \notin \mathrm{I}^{<k}(Q_1) \text{ and } \tau \notin \mathrm{I}^{<k}(Q_2)) \\
 & \iff & (\alpha{:}k \in \mathrm{I}^k(Q_1) \text{ and } \tau \notin \mathrm{I}^{<k}(Q_1) \text{ and } \tau \notin \mathrm{I}^{<k}(Q_2)) \text{ or} \\
 & & (\alpha{:}k \in \mathrm{I}^k(Q_2) \text{ and } \tau \notin \mathrm{I}^{<k}(Q_1) \text{ and } \tau \notin \mathrm{I}^{<k}(Q_2)) \\
(\text{def. of } \mathcal{I}^{\cdot}(\cdot)) & \iff & (\alpha{:}k \in \mathcal{I}^k(Q_1) \text{ and } \tau \notin \mathrm{I}^{<k}(Q_2)) \text{ or} \\
 & & (\alpha{:}k \in \mathcal{I}^k(Q_2) \text{ and } \tau \notin \mathrm{I}^{<k}(Q_1)) \\
(\text{ind. hyp.}) & \iff & (\alpha{:}k \in \mathcal{T}^k(Q_1) \text{ and } \tau \notin \mathrm{I}^{<k}(Q_2)) \text{ or} \\
 & & (\alpha{:}k \in \mathcal{T}^k(Q_2) \text{ and } \tau \notin \mathrm{I}^{<k}(Q_1)) \\
(\text{oper. semantics}) & \iff & \alpha{:}k \in \mathcal{T}^k(P)
\end{array}
$$

- $P \equiv Q_1 \,|\, Q_2$:

$$
\begin{aligned}
& & & \alpha{:}k \in \mathcal{I}^k(P) \\
& \text{(def. of } \mathcal{I}^{\cdot}(\cdot)) & \iff\ & \alpha{:}k \in \mathrm{I}^k(P) \text{ and } \tau \notin \mathrm{I}^{<k}(P) \\
& \text{(def. of } \mathrm{I}^{\cdot}(\cdot)) & \iff\ & (\alpha{:}k \in \mathrm{I}^k(Q_1) \text{ or } \alpha{:}k \in \mathrm{I}^k(Q_2) \text{ or} \\
& & & \ \alpha{:}k \in \{\tau{:}k \,|\, \mathrm{I}^k(Q_1) \cap \overline{\mathrm{I}^k(Q_2)} \neq \emptyset\}) \text{ and} \\
& & & (\tau \notin \mathrm{I}^{<k}(Q_1) \text{ and } \tau \notin \mathrm{I}^{<k}(Q_2) \text{ and} \\
& & & \ \mathrm{I}^{<k}(Q_1) \cap \overline{\mathrm{I}^{<k}(Q_2)} = \emptyset) \\
& & \iff\ & (\alpha{:}k \in \mathrm{I}^k(Q_1) \text{ and } \tau \notin \mathrm{I}^{<k}(Q_1) \text{ and } \tau \notin \mathrm{I}^{<k}(Q_2) \\
& & & \ \text{and } \mathrm{I}^{<k}(Q_1) \cap \overline{\mathrm{I}^{<k}(Q_2)} = \emptyset) \text{ or} \\
& & & (\alpha{:}k \in \mathrm{I}^k(Q_2) \text{ and } \tau \notin \mathrm{I}^{<k}(Q_1) \text{ and } \tau \notin \mathrm{I}^{<k}(Q_2) \\
& & & \ \text{and } \mathrm{I}^{<k}(Q_1) \cap \overline{\mathrm{I}^{<k}(Q_2)} = \emptyset) \text{ or} \\
& & & (\alpha{:}k \in \{\tau{:}k \,|\, \mathrm{I}^k(Q_1) \cap \overline{\mathrm{I}^k(Q_2)} \neq \emptyset\} \text{ and } \tau \notin \mathrm{I}^{<k}(Q_1) \\
& & & \ \text{and } \tau \notin \mathrm{I}^{<k}(Q_2) \text{ and } \mathrm{I}^{<k}(Q_1) \cap \overline{\mathrm{I}^{<k}(Q_2)} = \emptyset) \\
& \text{(def. of } \mathcal{I}^{\cdot}(\cdot) \text{ and } \mathrm{I}^{\cdot}(\cdot)) & \iff\ & (\alpha{:}k \in \mathcal{I}^k(Q_1) \text{ and } \tau \notin \mathrm{I}^{<k}(Q_1 \,|\, Q_2)) \text{ or} \\
& & & (\alpha{:}k \in \mathcal{I}^k(Q_2) \text{ and } \tau \notin \mathrm{I}^{<k}(Q_1 \,|\, Q_2)) \text{ or} \\
& & & (\alpha{:}k \in \{\tau{:}k \,|\, \mathcal{I}^k(Q_1) \cap \overline{\mathcal{I}^k(Q_2)} \neq \emptyset\} \text{ and} \\
& & & \ \tau \notin \mathrm{I}^{<k}(Q_1 \,|\, Q_2)) \\
& \text{(ind. hyp.)} & \iff\ & (\alpha{:}k \in \mathcal{T}^k(Q_1) \text{ and } \tau \notin \mathrm{I}^{<k}(Q_1 \,|\, Q_2)) \text{ or} \\
& & & (\alpha{:}k \in \mathcal{T}^k(Q_2) \text{ and } \tau \notin \mathrm{I}^{<k}(Q_1 \,|\, Q_2)) \text{ or} \\
& & & (\alpha{:}k \in \{\tau{:}k \,|\, \mathcal{T}^k(Q_1) \cap \overline{\mathcal{T}}^k(Q_2) \neq \emptyset\} \text{ and} \\
& & & \ \tau \notin \mathrm{I}^{<k}(Q_1 \,|\, Q_2)) \\
& \text{(oper. semantics)} & \iff\ & \alpha{:}k \in \mathcal{T}^k(P)
\end{aligned}
$$

The case $P \equiv Q_1 \,]\!]\, Q_2$ is similar to the one for $P \equiv Q_1 + Q_2$. The remaining cases are easier to prove and, therefore, are omitted. □

Summarizing, the framework of CCS^{ch} can briefly be outlined by the "formula"

$$\text{CCS}^{ch} = \text{CCS} + \text{disabling} + \text{priority} + \text{pre-emption}.$$

Intuitively, a process P can engage in an $\alpha{:}k$-transition to P' if it can do so according to CCS semantics (including disabling) and, additionally, if P cannot engage in a higher prioritized internal transition, i.e. the global constraint $\tau \notin \mathrm{I}^{<k}(P)$ regarding pre-emption is satisfied.

As a very simple example, which illustrates the utility of considering priority in process algebras, we take a look at the system already introduced in Section 2.1, an alteration of

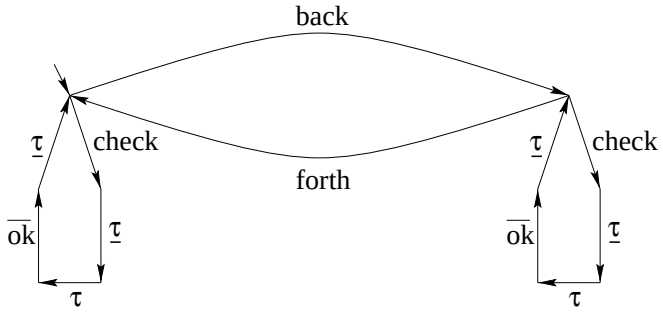

Figure 2.2: Semantics of `Sys`

an example presented in [54]. Since a two-level priority-scheme suffices for its definition, we write $\underline{\alpha}$ for the "*prioritized*" action $\alpha{:}0$ and α for the "*unprioritized*" action $\alpha{:}1$. A more complex example is presented in Section 2.5 and yet another one, which makes use of both the multi-level priority-scheme and the additional disabling operator, can be found in the next chapter. The system depicted in Figure 2.1 can be formalized in $\mathsf{CCS}^{\mathsf{ch}}$ as follows: $\mathtt{Sys} \stackrel{\text{def}}{=} (A \mid B) \setminus \{\underline{i}\}$ where $A \stackrel{\text{def}}{=} \mathtt{back}.A' + \underline{i}.\tau.\overline{\mathtt{ok}}.\overline{\underline{i}}.A$, $A' \stackrel{\text{def}}{=} \mathtt{forth}.A + \underline{i}.\tau.\overline{\mathtt{ok}}.\overline{\underline{i}}.A'$, and $B \stackrel{\text{def}}{=} \mathtt{check}.\overline{\underline{i}}.\underline{i}.B$. Intuitively, $\underline{i}$ is an internal *interrupt*, and thus prioritized and restricted (via $\setminus\{\underline{i}\}$), which is invoked whenever `check` is executed. Hence, in such a state the process A cannot engage in a transition labeled by `back` or `forth` according to our pre-emptive operational semantics, but must accept the communication on the prioritized port $\underline{i}$. In pure CCS, this is not the case, i.e. `Sys` can ignore to respond to `check` by $\overline{\mathtt{ok}}$ forever. The difference of our example to a similar one in [54] arises from the additional unprioritized internal action τ preceding the actions $\overline{\mathtt{ok}}$. One can think of the τ's as representing some internal activities determining the current status of the system. The semantics of `Sys` is shown in Figure 2.2. In the sequel, we present a semantic theory for $\mathsf{CCS}^{\mathsf{ch}}$ allowing us to prove that `Sys` meets its intuitive specification `Spec` which can be formalized as follows: $\mathtt{Spec} \stackrel{\text{def}}{=} \mathtt{back}.\mathtt{Spec}' + \mathtt{check}.\overline{\mathtt{ok}}.\mathtt{Spec}$ where $\mathtt{Spec}' \stackrel{\text{def}}{=} \mathtt{forth}.\mathtt{Spec} + \mathtt{check}.\overline{\mathtt{ok}}.\mathtt{Spec}'$. Unfortunately, the semantic theory presented in [54] is inadequate for relating `Sys` and `Spec` since it does not abstract away from any unprioritized internal transitions.

2.3 A Semantic Theory based on Strong Bisimulation

The semantic theory for $\mathsf{CCS}^{\mathsf{ch}}$, which we intend to develop, is based on Park and Milner's notion of *bisimulation* [125, 139]. First, we adapt *strong bisimulation* [125] from CCS to our setting as follows; we refer to this relation as *prioritized strong bisimulation*.

Definition 2.3.1 (Prioritized Strong Bisimulation)
A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is called a prioritized strong bisimulation *if for every $\langle P, Q\rangle \in \mathcal{R}$ and $\alpha{:}k \in \mathcal{A}$ the following condition holds.*

$$P \xrightarrow{\alpha:k} P' \text{ implies } \exists Q'.\, Q \xrightarrow{\alpha:k} Q' \text{ and } \langle P', Q'\rangle \in \mathcal{R} \ .$$

We write $P \simeq Q$ if there exists a prioritized strong bisimulation $\mathcal{R}$ such that $\langle P, Q\rangle \in \mathcal{R}$.

It is easy to see that $\simeq$ is an equivalence and that it is the *largest* prioritized strong bisimulation. The following result, which enables compositional reasoning, can be proved straightforwardly using standard techniques [54, 167].

Theorem 2.3.2 *Prioritized strong bisimulation $\simeq$ is a congruence.*

An *axiomatization* of $\simeq$ for *finite* processes, i.e. guarded and closed $\mathsf{CCS}^{\mathsf{ch}}$ terms that do not contain recursion, can be developed closely along the lines of [54]. Since axiomatizations give rise to equational reasoning by manipulating terms, we also call processes *process terms*, ranged over by $t, u, \ldots$, in this context. We write $\vdash t = u$ if process term t can be rewritten to u using the axioms in Table 2.3. Regarding the axioms presented in [54] we have modified Axioms (E), (Res2), (Res3), (Rel3) and (P) to cover our new action structure for multi-level priority-schemes. In the *Expansion Axiom* (E) the symbol $\sum$ stands for the indexed version of $+$, where the empty sum denotes the inaction process $\mathbf{0}$. We have also added Axioms (D1), (D2), and (D3) dealing with the disabling operator. Axiom (D1) allows us to push the disabling operator inside its first operand while inserting an outer nondeterministic branch for the second operand. Axiom (D2) identifies a situation in which the disabling operator may be dropped. Finally, Axiom (D3) presents a *distributive law* which enables us to push the disabling operator over nondeterministic choices. The next theorem states that our equations characterize prioritized strong bisimulation for finite $\mathsf{CCS}^{\mathsf{ch}}$ processes.

Theorem 2.3.3 (Soundness & Completeness)
For finite $\mathsf{CCS}^{\mathsf{ch}}$ process terms t and u we have: $t \simeq u$ if and only if $\vdash t = u$.

The proof of this theorem is similar to the corresponding one presented in [54] and, therefore, is omitted. The completeness part is based on the fact that every finite process term can be rewritten into *normal form*, i.e. into a term of the form $\sum_i \alpha_i{:}k_i.t_i$, where $\alpha_i \equiv \tau$ implies $k_j \leq k_i$ for all j, and the terms t_i are in normal form, too. It is easy to see that the additional Axioms (D1)–(D3) are sufficient to obtain a normal form for processes containing disabling operators. The above axiomatization and Theorem 2.3.3 can be extended to *regular* process terms, which describe a class of finite-state processes, using the technique introduced in [124].

Prioritized strong bisimulation can also be characterized logically. In fact, the Hennessy-Milner logic presented in [125] can be carried over one-to-one. Moreover, prioritized strong bisimulation can be related to modal logics by adapting the idea of *characteristic formulas* [157].

For the back-and-forth example considered in the previous section it can be shown by equational reasoning that $\mathtt{Sys} \simeq \mathtt{Spec}^*$, where $\mathtt{Spec}^*$ is defined as $\mathtt{Spec}$ but with a $\underline{\tau}$- and a τ-action preceding each action $\overline{\mathtt{ok}}$ and a $\underline{\tau}$-action trailing every $\overline{\mathtt{ok}}$. The reason for inserting these internal actions is that two prioritized strong bisimular terms have to match each

Table 2.3: Axiomatization of $\simeq$

(A1)	$t + u = u + t$		(A2)	$t + (u + v) = (t + u) + v$
(A3)	$t + t = t$		(A4)	$t + \mathbf{0} = t$
(E)	Let $t = \sum_i \alpha_i{:}k_i.t_i$ and $u = \sum_j \beta_j{:}l_j.u_j$. Then $t \mid u = \sum_i \alpha_i{:}k_i.(t_i \mid u) + \sum_j \beta_j{:}l_j.(t \mid u_j) + \sum_{\alpha_i:k_i=\overline{\beta_j}:l_j} \tau{:}k_i.(t_i \mid u_j)$			
(Res1)	$\mathbf{0} \setminus L = \mathbf{0}$		(Rel1)	$\mathbf{0}[f] = \mathbf{0}$
(Res2)	$(\alpha{:}k.t) \setminus L = \mathbf{0}$	$(\alpha{:}k \in L \cup \overline{L})$	(Rel2)	$(\alpha{:}k.t)[f] = f(\alpha{:}k).(t[f])$
(Res3)	$(\alpha{:}k.t) \setminus L = \alpha{:}k.(t \setminus L)$	$(\alpha{:}k \notin L \cup \overline{L})$	(Rel3)	$(t + u)[f] = t[f] + u[f]$
(Res4)	$(t + u) \setminus L = (t \setminus L) + (u \setminus L)$			
(P)	$\tau{:}k.t + \alpha{:}l.u = \tau{:}k.t$	$(k < l)$		
(D1)	$(\alpha{:}k.t) \between u = \alpha{:}k.(t \between u) + u$		(D2)	$\mathbf{0} \between t = t$
(D3)	$(t + u) \between v = (t \between v) + (u \between v)$			

other's transitions exactly, even those labeled by internal actions. Therefore, `Sys` and `Spec` fail to be equivalent with respect to prioritized strong bisimulation.

2.4 A Semantic Theory based on Weak Bisimulation

As illustrated by the back-and-forth example, prioritized strong bisimulation is too fine for reasoning about systems in practice. In this section we remedy the above mentioned shortcomings by developing a semantic congruence that abstracts from internal transitions along the lines of [125, 134]. We start off with the definition of a *naive* prioritized weak bisimulation which is an adaptation of *observational equivalence* [125].

Definition 2.4.1 (Naive Prioritized Weak Transition Relation)

1. $\widehat{\tau{:}k} =_{df} \epsilon$ *and* $\widehat{a{:}k} =_{df} a{:}k$
2. $\stackrel{\epsilon}{\Longrightarrow}_{\times} =_{df} (\{\stackrel{\tau:k}{\longrightarrow} \mid k \in \mathbb{N}\})^*$
3. $\stackrel{\alpha:k}{\Longrightarrow}_{\times} =_{df} \stackrel{\epsilon}{\Longrightarrow}_{\times} \circ \stackrel{\alpha:k}{\longrightarrow} \circ \stackrel{\epsilon}{\Longrightarrow}_{\times}$

Observe that this naive approach abstracts from priority levels when defining $\widehat{\tau:k}$ and $\stackrel{\epsilon}{\Longrightarrow}_{\times}$. As one might expect this will cause some problems for the resulting notion of *naive prioritized weak bisimulation* which is defined next.

Definition 2.4.2 (Naive Prioritized Weak Bisimulation)
A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a naive prioritized weak bisimulation *if for every $\langle P, Q\rangle \in \mathcal{R}$, and $\alpha{:}k \in \mathcal{A}$ the following condition holds.*

$$P \stackrel{\alpha:k}{\longrightarrow} P' \text{ implies } \exists Q'.\, Q \stackrel{\widehat{\alpha:k}}{\Longrightarrow}_{\times} Q' \text{ and } \langle P', Q'\rangle \in \mathcal{R} \ .$$

We write $P \approx_{\times} Q$ if $\langle P, Q\rangle \in \mathcal{R}$ for some naive prioritized weak bisimulation $\mathcal{R}$.

Naive prioritized weak bisimulation can easily be shown to be an equivalence. Unfortunately, $\approx_{\times}$ is not a congruence for $\mathsf{CCS}^{\mathsf{ch}}$ with respect to parallel composition, disabling, summation, and recursion. Whereas the compositionality defect for summation is similar to the one for CCS [125] and the one with respect to recursion is due to the others, the defect of disabling can partly be reduced to the one for parallel composition and the one for summation. As an example concerned with the problem regarding parallel composition consider the processes $P \stackrel{\text{def}}{=} a{:}1.\mathbf{0} + b{:}0.\mathbf{0}$ and $Q \stackrel{\text{def}}{=} a{:}1.\mathbf{0} + \tau{:}1.(a{:}1.\mathbf{0} + b{:}0.\mathbf{0})$. It is easy to see that $P \approx_{\times} Q$. However, when composing these processes in parallel with the process $\overline{b}{:}0.\mathbf{0}$ then $Q \,|\, \overline{b}{:}0.\mathbf{0} \stackrel{a:1}{\longrightarrow} \mathbf{0} \,|\, \overline{b}{:}0.\mathbf{0}$ whereas $P \,|\, \overline{b}{:}0.\mathbf{0} \stackrel{a:1}{\not\Longrightarrow}_{\times}$, i.e. $P \,|\, \overline{b}{:}0.\mathbf{0} \not\approx_{\times} Q \,|\, \overline{b}{:}0.\mathbf{0}$.

2.4.1 Prioritized Weak Bisimulation

Despite the lack of compositionality, the above definition of $\approx_{\times}$ reflects an intuitive approach to abstracting from internal computation. For attacking the congruence problem it is important to consider the following fact from universal algebra.

Proposition 2.4.3 (Largest Congruence)
Let $\mathcal{R}$ be an equivalence over an algebra $\Re$. Then the largest congruence $\mathcal{R}^+$ in $\mathcal{R}$ exists and $\mathcal{R}^+ = \{\langle P, Q\rangle \mid \forall\, \Re\text{-contexts } C[X].\, \langle C[P], C[Q]\rangle \in \mathcal{R}\}$, where a $\Re$-context $C[X]$ is a term in $\Re$ with one free occurrence of the variable X.

Consequently, we know that $\approx_{\times}$ contains a largest congruence $\approx_{\times}^{+}$ for $\mathsf{CCS}^{\mathsf{ch}}$, and we devote the rest of this section to characterizing this congruence. We first define a new weak transition relation which takes pre-emption into account.

Definition 2.4.4 (Prioritized Weak Transition Relation)
For $L \subseteq \mathcal{A} \setminus \{\tau{:}k \mid k \in \mathbb{N}\}$ we define the following.

1. $\widehat{\tau:k} =_{df} \hat{\tau}{:}k =_{df} \epsilon{:}k$ *and* $\widehat{a:k} =_{df} \hat{a}{:}k =_{df} a{:}k$
2. $P \stackrel{\alpha:k}{\underset{L}{\longrightarrow}} P'$ *if* $P \stackrel{\alpha:k}{\longrightarrow} P'$ *and* $\mathit{I\!I}^{<k}(P) \subseteq L$

3. $\stackrel{\epsilon:0}{\Longrightarrow} =_{df} (\stackrel{\tau:0}{\longrightarrow})^*$

4. $\stackrel{\epsilon:k}{\underset{L}{\Longrightarrow}} =_{df} (\{ \stackrel{\tau:l}{\underset{L}{\longrightarrow}} \mid l \leq k\})^*$

5. $\stackrel{\alpha:k}{\underset{L}{\Longrightarrow}} =_{df} \stackrel{\epsilon:k}{\underset{L}{\Longrightarrow}} \circ \stackrel{\alpha:k}{\underset{L}{\longrightarrow}} \circ \stackrel{\epsilon:0}{\Longrightarrow}$

Intuitively, we have made the transition relation sensitive to pre-emption by introducing conditions involving initial action sets and by preserving priority values of internal actions. In the remainder, we show that initial action sets are an adequate mean for measuring pre-emption potentials. In this light, $P \stackrel{\alpha:k}{\underset{L}{\longrightarrow}} P'$ states that P can evolve to P' by performing action α with priority k if the pre-emptive power of P is at most L, i.e. if the environment does not offer any communication on ports in L. Moreover, we abbreviate $P \stackrel{a:0}{\underset{L}{\Longrightarrow}} P'$ by $P \stackrel{a:0}{\Longrightarrow} P'$ since the inclusion $\mathcal{I}\!\!\mathcal{I}^{<0}(P) \subseteq L$ trivially holds for all L. For convenience we also define $P \stackrel{\epsilon}{\longrightarrow} P$ and $P \stackrel{\epsilon}{\underset{L}{\longrightarrow}} P$ for all $P \in \mathcal{P}$ and $L \subseteq \mathcal{A} \setminus \{\tau{:}k \mid k \in \mathbb{N}\}$.

Definition 2.4.5 (Prioritized Weak Bisimulation)
A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a prioritized weak bisimulation *if for every $\langle P, Q\rangle \in \mathcal{R}$, $k \in \mathbb{N}$, and $\alpha{:}k \in \mathcal{A}$ the following conditions hold.*

1. *$\tau \notin \mathcal{I}^{<k}(P)$ implies $\exists Q'.\, Q \stackrel{\epsilon:k}{\underset{L}{\Longrightarrow}} Q'$, $\mathcal{I}\!\!\mathcal{I}^{<k}(Q') \subseteq L$ where $L = \mathcal{I}\!\!\mathcal{I}^{<k}(P)$, $\tau \notin \mathcal{I}^{<k}(Q')$, and $\langle P, Q'\rangle \in \mathcal{R}$.*

2. *$P \stackrel{\alpha:k}{\longrightarrow} P'$ implies $\exists Q'.\, Q \stackrel{\hat{\alpha}:k}{\underset{L}{\Longrightarrow}} Q'$, $L = \mathcal{I}\!\!\mathcal{I}^{<k}(P)$, and $\langle P', Q'\rangle \in \mathcal{R}$.*

We write $P \underline{\approx} Q$ if $\langle P, Q\rangle \in \mathcal{R}$ for some prioritized weak bisimulation $\mathcal{R}$.

From this definition we may conclude that $\underline{\approx}$ is the *largest* prioritized weak bisimulation and that $\underline{\approx}$ is an equivalence, which can be extended to open terms in the usual way. More important, we obtain the following result.

Proposition 2.4.6 *The equivalence $\underline{\approx}$ is a congruence with respect to prefixing, parallel composition, relabeling, and restriction. Moreover, $\underline{\approx}$ is characterized as the* largest congruence *contained in $\approx_{\times}$, in the sub-algebra of CCS^{ch} induced by these operators and recursion.*

Note that still a compositionality problem with respect to summation exists, which is dealt with later. One may wonder why the definition of prioritized weak bisimulation cannot be simplified in order to obtain a weak behavioral relation that is compositional for parallel composition. First, the absence of Condition (1) in Definition 2.4.5 would cause compositionality defects as is best illustrated by the following example. The processes $P \stackrel{\text{def}}{=} \tau{:}1.a{:}0.\mathbf{0}$ and $Q \stackrel{\text{def}}{=} a{:}0.\mathbf{0}$ would be considered as equivalent if Condition (1) would be absent. However, the context $C[X] \stackrel{\text{def}}{=} X \mid (\overline{a}{:}0.\mathbf{0} + b{:}1.\mathbf{0})$ is able to distinguish them.

Second, the $a{:}k$-step of a weak $a{:}k$-transition may only be followed by a sequence of $\tau{:}0$-transitions. Why may one not define $\stackrel{\alpha:k}{\underset{L}{\Longrightarrow}} =_{\mathrm{df}} \stackrel{\epsilon:k}{\underset{L}{\Longrightarrow}} \circ \stackrel{\alpha:k}{\underset{L}{\longrightarrow}} \circ \stackrel{\epsilon:k}{\underset{L}{\Longrightarrow}}$? In order to see this, consider the processes $P \stackrel{\mathrm{def}}{=} a{:}1.P' + b{:}0.\mathbf{0}$, where $P' \stackrel{\mathrm{def}}{=} \tau{:}1.P'' + b{:}0.\mathbf{0}$ and $P'' \stackrel{\mathrm{def}}{=} c{:}0.\mathbf{0} + b{:}0.\mathbf{0}$, and $Q \stackrel{\mathrm{def}}{=} a{:}1.P'' + P$. According to the above proposed weak transition relation P and Q would be deemed equivalent since the $a{:}1$-transition of Q to P'' can be matched by the $a{:}1$-transition followed by the $\tau{:}1$-transition of P. However, the context $C[X] \stackrel{\mathrm{def}}{=} (X \mid R) \setminus \{a{:}1, b{:}0\}$, where $R \stackrel{\mathrm{def}}{=} \overline{a}{:}1.\overline{b}{:}0.\mathbf{0}$, distinguishes P and Q. More precisely, the process $C[P]$ never reaches a state where it can engage in the $c{:}0$-transition, whereas $C[Q]$ does. The former is due to the fact that the $\tau{:}1$-transition of P gets pre-empted in the context $C[X]$.

In the remainder of this section we prove the first part of Proposition 2.4.6 which states that $\underline{\approx}$ is compositional for prefixing, parallel composition, relabeling, and restriction. Many cases are standard. The only interesting non-standard case is the compositionality of $\underline{\approx}$ with respect to parallel composition which we give in detail. The second part of the proposition, which claims that $\underline{\approx}$ is characterized as the *largest congruence* contained in $\approx_{\times}$ in the sub-algebra of $\mathsf{CCS}^{\mathsf{ch}}$ induced by the above mentioned operators and recursion, follows by inspection of the proof of Proposition 2.4.19 presented below. First, let us establish a convenient lemma.

Lemma 2.4.7 *Let $P, P', Q, Q', R \in \mathcal{P}$ and $k \in \mathbb{N}$. Then*

1. *$P \stackrel{\epsilon:0}{\Longrightarrow} P'$ implies $P \mid R \stackrel{\epsilon:0}{\Longrightarrow} P' \mid R$, and*
2. *$\tau \notin \mathcal{I}^{<k}(P \mid R)$ and $Q \stackrel{\epsilon:k}{\underset{L}{\Longrightarrow}} Q'$, where $L = \mathcal{U}^{<k}(P)$, implies for $L' = \mathcal{U}^{<k}(P \mid R)$: $Q \mid R \stackrel{\epsilon:k}{\underset{L'}{\Longrightarrow}} Q' \mid R$.*

Proof: The proof of Part (1) can be done straightforwardly by induction on the length of the weak transition $P \stackrel{\epsilon:0}{\Longrightarrow} P'$. Part (2) is proved by induction on the length i of the prioritized weak transition $Q \stackrel{\epsilon:k}{\underset{L}{\Longrightarrow}} Q'$. For $i = 0$ the validity of the statement is trivial. Assume $i > 0$, i.e. $Q \stackrel{\tau:l}{\underset{L}{\longrightarrow}} Q'' \stackrel{\epsilon:k}{\underset{L}{\Longrightarrow}} Q'$ for some $Q'' \in \mathcal{P}$ and $l \leq k$. This implies $\mathcal{U}^{<l}(Q) \subseteq L = \mathcal{U}^{<k}(P)$ and $\tau \notin \mathcal{I}^{<l}(Q)$. Because of $\tau \notin \mathcal{I}^{<k}(P \mid R)$ we also have $\mathcal{U}^{<k}(P) \cap \overline{\mathcal{U}^{<k}(R)} = \emptyset$ and $\tau \notin \mathcal{I}^{<k}(R)$. Thus and because of $l \leq k$, $\mathcal{U}^{<l}(Q) \cap \overline{\mathcal{U}^{<l}(R)} \subseteq \mathcal{U}^{<k}(P) \cap \overline{\mathcal{U}^{<k}(R)} = \emptyset$ and $\tau \notin \mathcal{I}^{<l}(R)$, i.e. $\tau \notin \mathcal{I}^{<l}(Q \mid R)$. Therefore, $Q \mid R \stackrel{\tau:l}{\underset{L'}{\longrightarrow}} Q'' \mid R$ for $L' = \mathcal{U}^{<k}(P \mid R)$ by the operational semantics and since $\mathcal{U}^{<l}(Q \mid R) = \mathcal{U}^{<l}(Q) \cup \mathcal{U}^{<l}(R) \subseteq \mathcal{U}^{<k}(P) \cup \mathcal{U}^{<k}(R) = \mathcal{U}^{<k}(P \mid R)$ due to $l \leq k$, as desired. Additionally, we may use the induction hypothesis to conclude $Q'' \mid R \stackrel{\epsilon:k}{\underset{L'}{\Longrightarrow}} Q' \mid R$. Hence, $Q \mid R \stackrel{\epsilon:k}{\underset{L'}{\Longrightarrow}} Q' \mid R$ by the definition of the prioritized weak transition relation. □

Now, we are able to prove the compositionality of $\underline{\approx}$ with respect to parallel composition.

Proposition 2.4.8 *$P \underline{\approx} Q$ implies $P \mid R \underline{\approx} Q \mid R$ for all processes R.*

Proof: According to Definition 2.4.5 it is sufficient to prove that the symmetric relation

$$\mathcal{R} =_{\mathrm{df}} \{\langle P \,|\, R, Q \,|\, R\rangle \,|\, P \underline{\approx} Q\}$$

is a prioritized weak bisimulation. Consider some arbitrary pair $\langle P \,|\, R, Q \,|\, R\rangle \in \mathcal{R}$, i.e. $P \underline{\approx} Q$.

1. Let $\tau \notin \mathcal{I}^{<k}(P \,|\, R)$ for some $k \in \mathbb{N}$. This implies $\tau \notin \mathcal{I}^{<k}(P)$, $\tau \notin \mathcal{I}^{<k}(R)$, and $\mathcal{U}^{<k}(P) \cap \overline{\mathcal{U}^{<k}(R)} = \emptyset$. Because of $P \underline{\approx} Q$ we know of the existence of some $Q' \in \mathcal{P}$ such that $Q \overset{\epsilon:k}{\underset{L}{\Longrightarrow}} Q'$, $\mathcal{U}^{<k}(Q') \subseteq L$ where $L = \mathcal{U}^{<k}(P)$, $\tau \notin \mathcal{I}^{<k}(Q')$, and $P \underline{\approx} Q'$. We conclude $Q \,|\, R \overset{\epsilon:k}{\underset{L'}{\Longrightarrow}} Q' \,|\, R$ for $L' = \mathcal{U}^{<k}(P \,|\, R)$ by Lemma 2.4.7(2), and $\tau \notin \mathcal{I}^{<k}(Q' \,|\, R)$ since $\tau \notin \mathcal{I}^{<k}(Q')$, $\tau \notin \mathcal{I}^{<k}(R)$, and $\mathcal{U}^{<k}(Q') \cap \overline{\mathcal{U}^{<k}(R)} = \emptyset$ due to $\mathcal{U}^{<k}(Q') \subseteq \mathcal{U}^{<k}(P)$. Moreover, $\langle P \,|\, R, Q' \,|\, R\rangle \in \mathcal{R}$ by the definition of $\mathcal{R}$. Hence, Condition (1) of Definition 2.4.5 is satisfied.

2. Let $P \,|\, R \xrightarrow{\alpha:k} V$ for some $V \in \mathcal{P}$ and $\alpha{:}k \in \mathcal{A}$, i.e. $\tau \notin \mathcal{I}^{<k}(P \,|\, R)$ which means $\tau \notin \mathcal{I}^{<k}(P)$, $\tau \notin \mathcal{I}^{<k}(R)$, and $\mathcal{U}^{<k}(P) \cap \overline{\mathcal{U}^{<k}(R)} = \emptyset$. We have to show the existence of some $W \in \mathcal{P}$ such that $Q \,|\, R \overset{\hat{\alpha}:k}{\underset{L}{\Longrightarrow}} W$ where $L = \mathcal{U}^{<k}(P \,|\, R)$ and $\langle V, W\rangle \in \mathcal{R}$. Consider the following case distinction according to the operational rules for parallel composition.

   (a) $P \xrightarrow{\alpha:k} P'$ and $V \equiv P' \,|\, R$.

   Since $P \underline{\approx} Q$ we know of the existence of some $Q' \in \mathcal{P}$ such that $Q \overset{\hat{\alpha}:k}{\underset{L}{\Longrightarrow}} Q'$ and $P' \underline{\approx} Q'$ where $L = \mathcal{U}^{<k}(P)$. Thus, $Q \overset{\epsilon:k}{\underset{L}{\Longrightarrow}} Q'' \overset{\hat{\alpha}:k}{\underset{L}{\longrightarrow}} Q''' \overset{\epsilon:0}{\Longrightarrow} Q'$ for some $Q'', Q''' \in \mathcal{P}$. Especially, $\tau \notin \mathcal{I}^{<k}(Q'')$ if $Q'' \not\equiv Q'''$. Hence for $L' = \mathcal{U}^{<k}(P \,|\, R)$, (i) $Q \,|\, R \overset{\epsilon:k}{\underset{L'}{\Longrightarrow}} Q'' \,|\, R$ by Lemma 2.4.7(2), (ii) $Q'' \,|\, R \overset{\hat{\alpha}:k}{\underset{L'}{\longrightarrow}} Q''' \,|\, R$ which is trivial for $\alpha \equiv \tau$ and $Q'' \equiv Q'''$ and, otherwise, follows from $\mathcal{U}^{<k}(Q'') \subseteq L = \mathcal{U}^{<k}(P)$ implies $\mathcal{U}^{<k}(Q'') \cap \overline{\mathcal{U}^{<k}(R)} = \emptyset$, i.e. $\tau \notin \mathcal{I}^{<k}(Q'' \,|\, R)$, and since $\mathcal{U}^{<k}(Q'' \,|\, R) = \mathcal{U}^{<k}(Q'') \cup \mathcal{U}^{<k}(R) \subseteq \mathcal{U}^{<k}(P) \cup \mathcal{U}^{<k}(R) = \mathcal{U}^{<k}(P \,|\, R)$, and (iii) $Q''' \,|\, R \overset{\epsilon:0}{\Longrightarrow} Q' \,|\, R$ by Lemma 2.4.7(1). Thus, $Q \,|\, R \overset{\hat{\alpha}:k}{\underset{L'}{\Longrightarrow}} Q' \,|\, R$ by the definition of the prioritized weak transition relation and $\langle P' \,|\, R, Q' \,|\, R\rangle \in \mathcal{R}$ by the definition of $\mathcal{R}$.

   (b) $R \xrightarrow{\alpha:k} R'$ and $V \equiv P \,|\, R'$.

   Since $P \underline{\approx} Q$ and $\tau \notin \mathcal{I}^{<k}(P)$ there exists some $Q' \in \mathcal{P}$ such that $Q \overset{\epsilon:k}{\underset{L}{\Longrightarrow}} Q'$, $\mathcal{U}^{<k}(Q') \subseteq L$ where $L = \mathcal{U}^{<k}(P)$, $\tau \notin \mathcal{I}^{<k}(Q')$, and $P \underline{\approx} Q'$. We conclude $Q \,|\, R \overset{\epsilon:k}{\underset{L'}{\Longrightarrow}} Q' \,|\, R$ for $L' = \mathcal{U}^{<k}(P \,|\, R)$ by Lemma 2.4.7(2) and $\tau \notin \mathcal{I}^{<k}(Q' \,|\, R)$. Hence, $Q' \,|\, R \xrightarrow{\alpha:k} Q' \,|\, R'$ by $\mathsf{CCS}^{\mathsf{ch}}$ semantics and $\mathcal{U}^{<k}(Q' \,|\, R) = \mathcal{U}^{<k}(Q') \cup \mathcal{U}^{<k}(R) \subseteq \mathcal{U}^{<k}(P) \cup \mathcal{U}^{<k}(R) = \mathcal{U}^{<k}(P \,|\, R)$. Consequently, $Q \,|\, R \overset{\hat{\alpha}:k}{\underset{L'}{\Longrightarrow}} Q' \,|\, R'$ by the definition of the prioritized weak transition relation and $\langle P \,|\, R', Q' \,|\, R'\rangle \in \mathcal{R}$ by the definition of $\mathcal{R}$.

(c) $P \xrightarrow{a:k} P'$, $R \xrightarrow{\overline{a}:k} R'$, and $V \equiv P' \mid R'$.

Because of $P \underline{\approx} Q$ we know of the existence of some $Q' \in \mathcal{P}$ satisfying $Q \overset{a:k}{\underset{L}{\Longrightarrow}} Q'$ and $P' \underline{\approx} Q'$ where $L = \mathcal{U}^{<k}(P)$. According to the definition of the prioritized weak transition relation $Q \overset{\epsilon:k}{\underset{L}{\Longrightarrow}} Q'' \xrightarrow[L]{a:k} Q''' \overset{\epsilon:0}{\Longrightarrow} Q'$ for some $Q'', Q''' \in \mathcal{P}$. Note that this implies $\tau \notin \mathcal{I}^{<k}(Q'')$. Thus, we have for $L' = \mathcal{U}^{<k}(P \mid R)$, (i) $Q \mid R \overset{\epsilon:k}{\underset{L'}{\Longrightarrow}} Q'' \mid R$ by Lemma 2.4.7(2), (ii) $Q'' \mid R \xrightarrow[L']{a:k} Q''' \mid R'$ since $\mathcal{U}^{<k}(Q'') \subseteq L = \mathcal{U}^{<k}(P)$ implies $\mathcal{U}^{<k}(Q'') \cap \overline{\mathcal{U}^{<k}(R)} = \emptyset$, i.e. $\tau \notin \mathcal{I}^{<k}(Q'' \mid R)$, and since $\mathcal{U}^{<k}(Q'' \mid R) = \mathcal{U}^{<k}(Q'') \cup \mathcal{U}^{<k}(R) \subseteq \mathcal{U}^{<k}(P) \cup \mathcal{U}^{<k}(R) = \mathcal{U}^{<k}(P \mid R)$, and (iii) $Q''' \mid R' \overset{\epsilon:0}{\Longrightarrow} Q' \mid R'$ according to Lemma 2.4.7(1). Hence, $Q \mid R \overset{\epsilon:k}{\underset{L'}{\Longrightarrow}} Q' \mid R'$ by the definition of the prioritized weak transition relation and $\langle P' \mid R', Q' \mid R' \rangle \in \mathcal{R}$ by the definition of $\mathcal{R}$.

Therefore, also Condition (2) of Definition 2.4.5 is established, and the proof is done.

□

In order to show that $\underline{\approx}$ is compositional with respect to recursion in the considered subalgebra of $\mathsf{CCS}^{\mathsf{ch}}$, we need to define a notion of *prioritized weak bisimulation up to* $\underline{\approx}$ (cf. [149]).

Definition 2.4.9 (Prioritized Weak Bisimulation up to $\underline{\approx}$)
A relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a prioritized weak bisimulation up to $\underline{\approx}$ *if for every $\langle P, Q \rangle \in \mathcal{R}$, $k \in \mathbb{N}$, and $\alpha{:}k \in \mathcal{A}$ the following conditions and their symmetric counterparts hold.*

1. *$\tau \notin \mathcal{I}^{<k}(P)$ implies $\exists Q'.\, Q \overset{\epsilon:k}{\underset{L}{\Longrightarrow}} Q'$, $\mathcal{U}^{<k}(Q') \subseteq L$ where $L = \mathcal{U}^{<k}(P)$, $\tau \notin \mathcal{I}^{<k}(Q')$, and $P \,\mathcal{R} \underline{\approx}\, Q'$.*

2. *$P \xrightarrow{\alpha:k} P'$ implies $\exists Q'.\, Q \overset{\hat{\alpha}:k}{\underset{L}{\Longrightarrow}} Q'$, $L = \mathcal{U}^{<k}(P)$, and $P' \,\mathcal{R} \underline{\approx}\, Q'$.*

This notion satisfies the property that, if $\mathcal{R}$ is a *prioritized weak bisimulation up to* $\underline{\approx}$, then $\mathcal{R} \subseteq \underline{\approx}$. Using the definition of prioritized weak bisimulation up to $\underline{\approx}$ the compositionality proof of $\underline{\approx}$ with respect to recursion follows the standard lines [125].

We now return to our back-and-forth example as introduced in Section 2.2 and show that the system `Sys` meets its specification `Spec` by proving $\texttt{Sys} \underline{\approx} \texttt{Spec}$. This can be verified using Table 2.4 which presents a relation whose symmetric closure is a prioritized weak bisimulation containing $\langle \texttt{Sys}, \texttt{Spec} \rangle$. Note that the validation of Condition (1) of Definition 2.4.5 for this relation is drastically simplified by the fact that the semantics of both processes, `Sys` and `Spec`, do not contain any visible prioritized actions.

Table 2.4: A relation whose symmetric closure is a prioritized weak bisimulation

$\{\ \langle$ Sys	,	Spec $\rangle$,
$\langle\ (A' \mid B) \setminus \{\underline{i}\}$	,	Spec$'$ $\rangle$,
$\langle\ (A \mid \overline{\underline{i}}.\underline{i}.B) \setminus \{\underline{i}\}$	,	$\overline{\mathtt{ok}}$.Spec $\rangle$,
$\langle\ (\tau.\overline{ok}.\overline{\underline{i}}.A \mid \underline{i}.B) \setminus \{\underline{i}\}$	,	$\overline{\mathtt{ok}}$.Spec $\rangle$,
$\langle\ (\overline{ok}.\overline{\underline{i}}.A \mid \underline{i}.B) \setminus \{\underline{i}\}$	,	$\overline{\mathtt{ok}}$.Spec $\rangle$,
$\langle\ (\overline{\underline{i}}.A \mid \underline{i}.B) \setminus \{\underline{i}\}$	,	Spec $\rangle$,
$\langle\ (A' \mid \overline{\underline{i}}.\underline{i}.B) \setminus \{\underline{i}\}$	,	$\overline{\mathtt{ok}}$.Spec$'$ $\rangle$,
$\langle\ (\tau.\overline{ok}.\overline{\underline{i}}.A' \mid \underline{i}.B) \setminus \{\underline{i}\}$	,	$\overline{\mathtt{ok}}$.Spec $\rangle$,
$\langle\ (\overline{ok}.\overline{\underline{i}}.A' \mid \underline{i}.B) \setminus \{\underline{i}\}$	,	$\overline{\mathtt{ok}}$.Spec$'$ $\rangle$,
$\langle\ (\overline{\underline{i}}.A' \mid \underline{i}.B) \setminus \{\underline{i}\}$	,	Spec$'$ $\rangle$ $\}$

2.4.2 Prioritized Observational Congruence

In contrast to [134], the summation fix presented in [125] is not sufficient in order to achieve a congruence based on prioritized weak bisimulation. To see why, let $D \stackrel{\text{def}}{=} \tau{:}0.E$ and $E \stackrel{\text{def}}{=} \tau{:}1.D$. Now define $P \stackrel{\text{def}}{=} \tau{:}1.D$ and $Q \stackrel{\text{def}}{=} \tau{:}0.E$. By Definition 2.4.5 we may observe $P \underline{\approx} Q$, but $P + a{:}1.\mathbf{0} \not\underline{\approx} Q + a{:}1.\mathbf{0}$ since the former can perform an $a{:}1$-action whereas the latter cannot. It turns out that we have to require that observationally congruent processes must possess the same initial action sets; a requirement which is stronger than Condition (1) of Definition 2.4.5.

Definition 2.4.10 (Prioritized Observational Congruence)
We define $P \underline{\approx}^{\mathrm{l}} Q$ if for all $\alpha{:}k \in \mathcal{A}$ the following conditions and their symmetric counterparts hold.

1. $\mathcal{I}(P) \supseteq \mathcal{I}(Q)$
2. $P \xrightarrow{\alpha:k} P'$ *implies* $\exists Q'.\, Q \stackrel{\alpha:k}{\underset{L}{\Longrightarrow}} Q'$, *where* $L = \mathcal{I}^{<k}(P)$, *and* $P' \underline{\approx} Q'$.

The following theorem states the main result of this section.

Theorem 2.4.11 *The relation $\underline{\approx}^{\mathrm{l}}$ is the largest congruence contained in $\approx_{\times}$, i.e. $\underline{\approx}^{\mathrm{l}} = \approx_{\times}^{+}$.*

The congruence property of $\underline{\approx}^{\mathrm{l}}$ can be established by standard techniques [125]. The proof of the compositionality of $\underline{\approx}^{\mathrm{l}}$ with respect to parallel composition is omitted here since it follows by inspection of the proof of Proposition 2.4.8. The only difference is that an internal transition is always matched by at least one internal computation step. The

much stricter initial action set condition in the definition of $\underline{\approx}^{\mathrm{l}}$ follows easily by using the definition of initial action sets. In the following, we present the compositionality result of $\underline{\approx}^{\mathrm{l}}$ with respect to summation and disabling.

Proposition 2.4.12 *$P \underline{\approx}^{\mathrm{l}} Q$ implies $P + R \underline{\approx}^{\mathrm{l}} Q + R$ for all processes R.*

Proof: Let $P, Q, R \in \mathcal{P}$ such that $P \underline{\approx}^{\mathrm{l}} Q$. Hence, $\mathcal{I}(P) = \mathcal{I}(Q)$ from which one may easily conclude $\mathcal{I}(P + R) = \mathcal{I}(Q + R)$. Moreover, let $P + R \xrightarrow{\alpha:k} V$ for some $V \in \mathcal{P}$ and $\alpha{:}k \in \mathcal{A}$. Now, consider the following case distinction according to the operational rules for summation.

1. $P \xrightarrow{\alpha:k} P'$, $V \equiv P'$, and $\tau \notin \mathcal{I}^{<k}(R)$. Especially, we know that $\tau \notin \mathcal{I}^{<k}(P)$ which implies $\tau \notin \mathcal{I}^{<k}(Q)$ since $\mathcal{I}(P) = \mathcal{I}(Q)$.

   Because of $P \underline{\approx}^{\mathrm{l}} Q$ we know of the existence of some $Q' \in \mathcal{P}$ such that $Q \overset{\alpha:k}{\underset{L}{\Longrightarrow}} Q'$ and $P' \underline{\approx} Q'$ where $L = \mathcal{I}\!\!\mathcal{I}^{<k}(P)$. We distinguish the following cases: (i) $Q \xrightarrow{\alpha:k} Q'' \overset{\epsilon:0}{\Longrightarrow} Q'$ for some $Q'' \in \mathcal{P}$ and $\mathcal{I}\!\!\mathcal{I}^{<k}(Q) \subseteq L$, and (ii) $Q \xrightarrow{\tau:l} Q'' \overset{\alpha:k}{\underset{L}{\Longrightarrow}} Q'$ for some $Q'' \in \mathcal{P}$, $l \leq k$, and $\mathcal{I}\!\!\mathcal{I}^{<l}(Q) \subseteq L$. In the former case, we obtain $Q + R \xrightarrow{\alpha:k} Q'' \overset{\epsilon:0}{\Longrightarrow} Q'$ since $\tau \notin \mathcal{I}^{<k}(R)$. Hence, $Q+R \overset{\alpha:k}{\underset{L'}{\Longrightarrow}} Q'$ for $L' = \mathcal{I}\!\!\mathcal{I}^{<k}(P + R)$ because of the definition of the prioritized weak transition relation and the fact that $\mathcal{I}(P + R) = \mathcal{I}(Q + R)$. In the latter case, we conclude $Q + R \overset{\tau:l}{\underset{L'}{\longrightarrow}} Q''$ since $\tau \notin \mathcal{I}^{<k}(R)$ and $l \leq k$ implies $\tau \notin \mathcal{I}^{<l}(R)$. Further, $Q'' \overset{\alpha:k}{\underset{L'}{\Longrightarrow}} Q'$ because $L \subseteq L'$. Hence, $Q + R \overset{\alpha:k}{\underset{L'}{\Longrightarrow}} Q'$ by the definition of the prioritized weak transition relation. Summarizing, in both cases we obtain $Q + R \overset{\alpha:k}{\underset{L'}{\Longrightarrow}} Q'$ for $L' = \mathcal{I}\!\!\mathcal{I}^{<k}(P + R)$ and $P' \underline{\approx} Q'$, as desired.

2. $R \xrightarrow{\alpha:k} R'$, $V \equiv R'$, and $\tau \notin \mathcal{I}^{<k}(P)$.

   Because of $\mathcal{I}(P) = \mathcal{I}(Q)$ we have $\tau \notin \mathcal{I}^{<k}(Q)$, too. Therefore, $Q+R \xrightarrow{\alpha:k} R'$. Hence, $Q + R \overset{\alpha:k}{\underset{L'}{\Longrightarrow}} R'$ for $L' = \mathcal{I}\!\!\mathcal{I}^{<k}(P + R)$ since $\mathcal{I}(P + R) = \mathcal{I}(Q + R)$. Finally, also $R' \underline{\approx} R'$ holds because $\underline{\approx}$ is reflexive.

Due to symmetry the proof is done. □

For proving the compositionality of $\underline{\approx}^{\mathrm{l}}$ with respect to the disabling operator we need some auxiliary properties analogous to those presented in Lemma 2.4.7 for parallel composition.

Lemma 2.4.13 *Let $P, P', Q, Q', R \in \mathcal{P}$. Moreover,*

1. *let $P \overset{\epsilon:0}{\Longrightarrow} P'$. Then $P \rangle\!\rangle R \overset{\epsilon:0}{\Longrightarrow} P' \rangle\!\rangle R$ holds.*

2. *let* $k \in \mathbb{N}$, $\tau \notin \mathcal{I}^{<k}(P \between R)$, $R \underline{\approx}^{\mathrm{l}} S$, *and* $Q \stackrel{\epsilon:k}{\Longrightarrow}_{L} Q'$ *for* $L = \mathcal{U}^{<k}(P)$. *Then* $Q \between S \stackrel{\epsilon:k}{\Longrightarrow}_{L'} Q' \between S$ *holds for* $L' = \mathcal{U}^{<k}(P \between R)$.

The proof of Part (1) is similar to the proof of Lemma 2.4.7(1), and the proof of Part (2) follows the lines of the proof of Lemma 2.4.7(2) under the consideration that $\mathcal{I}(R) = \mathcal{I}(S)$ since $R \underline{\approx}^{\mathrm{l}} S$.

Proposition 2.4.14 *Let* $R \underline{\approx}^{\mathrm{l}} S$. *Then*

1. $P \underline{\approx} Q$ *implies* $P \between R \;\underline{\approx}\; Q \between S$, *and*
2. $P \underline{\approx}^{\mathrm{l}} Q$ *implies* $P \between R \;\underline{\approx}^{\mathrm{l}}\; Q \between S$.

Proof: In order to establish the first part it is sufficient to prove that the symmetric relation

$$\mathcal{R} =_{\mathrm{df}} \{\langle P \between R, Q \between S\rangle \mid P \underline{\approx} Q,\ R \underline{\approx}^{\mathrm{l}} S\} \cup \underline{\approx}$$

is a prioritized weak bisimulation. Therefore, let $\langle P \between R, Q \between S\rangle \in \mathcal{R}$ be arbitrary, i.e. $P \underline{\approx} Q$ and $R \underline{\approx}^{\mathrm{l}} S$. Observe that $\mathcal{I}^{<k}(P \between R) = \mathcal{I}^{<k}(P) \cup \mathcal{I}^{<k}(R) = \mathcal{I}^{<k}(P) \cup \mathcal{I}^{<k}(S)$. Condition (1) of Definition 2.4.5 can be established along the lines of the corresponding proof part of Proposition 2.4.8 by using Lemma 2.4.13(2) instead of Lemma 2.4.7(2). Now, let $P \between R \stackrel{\alpha:k}{\longrightarrow} V$ for some $V \in \mathcal{P}$ and $\alpha{:}k \in \mathcal{A}$. We distinguish the following two cases according to the operational semantics for disabling.

1. $P \stackrel{\alpha:k}{\longrightarrow} P'$ for some $P' \in \mathcal{P}$, $V \equiv P' \between R$, and $\tau \notin \mathcal{I}^{<k}(R)$.

   The argumentation in this case is analogous to the one for establishing Condition (2) of Definition 2.4.5 in the proof of Proposition 2.4.8, if one replaces the use of Lemma 2.4.7(1) by Lemma 2.4.13(1) and the use of Lemma 2.4.7(2) by Lemma 2.4.13(2).

2. $R \stackrel{\alpha:k}{\longrightarrow} R'$ for some $R' \in \mathcal{P}$, $V \equiv R'$, and $\tau \notin \mathcal{I}^{<k}(P)$.

   Since $P \underline{\approx} Q$ there exists some $Q' \in \mathcal{P}$ such that $Q \stackrel{\epsilon:k}{\Longrightarrow}_{L} Q'$, $\mathcal{U}^{<k}(Q') \subseteq L$ where $L = \mathcal{U}^{<k}(P)$, $\tau \notin \mathcal{I}^{<k}(Q')$, and $P \underline{\approx} Q'$. Applying Lemma 2.4.13(2) we obtain $Q \between S \stackrel{\epsilon:k}{\Longrightarrow}_{L'} Q' \between S$ where $L' = \mathcal{U}^{<k}(P \between R)$. Moreover, since $R \underline{\approx}^{\mathrm{l}} S$ we know of the existence of some $S' \in \mathcal{P}$ such that $S \stackrel{\alpha:k}{\Longrightarrow}_{\overline{L}} S'$ for $\overline{L} = \mathcal{U}^{<k}(R)$ and $R' \underline{\approx} S'$. Since $\tau \notin \mathcal{I}^{<k}(Q')$ we obtain $Q' \between S \stackrel{\alpha:k}{\Longrightarrow}_{L'} S'$ similar to the reasoning in the proof of Proposition 2.4.12 since $\overline{L} \subseteq L'$ according to the definition of the initial action sets. Thus, $Q \between S \stackrel{\alpha:k}{\Longrightarrow}_{L'} S'$ by the definition of the prioritized weak transition relation and $\langle R', S'\rangle \in \mathcal{R}$ by the definition of $\mathcal{R}$.

Hence, Condition (2) of Definition 2.4.5 holds, and the proof of the first part of Proposition 2.4.14 is done. The second part follows pretty much the reasoning above and, therefore, is omitted. □

In order to show that $\underline{\approx}^{\mathrm{l}}$ is compositional with respect to recursion, we need to define a notion of *prioritized observational congruence up to* $\underline{\approx}^{\mathrm{l}}$ in order to adapt the standard proof technique [125, 149].

Definition 2.4.15 (Prioritized Observational Congruence up to $\underline{\approx}^{\mathrm{l}}$)
A relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a prioritized observational congruence up to $\underline{\approx}^{\mathrm{l}}$ *if for every $\langle P, Q\rangle \in \mathcal{R}$ and $\alpha{:}k \in \mathcal{A}$ the following conditions hold.*

1. $\mathcal{I}(P) = \mathcal{I}(Q)$
2. $P \xrightarrow{\alpha:k} P'$ *implies* $\exists Q'.\, Q \overset{\alpha:k}{\underset{L}{\Longrightarrow}} Q'$, $L = \mathcal{U}^{<k}(P)$, *and* $P' \,\mathcal{R}\, \underline{\approx}\, Q'$.
3. $Q \xrightarrow{\alpha:k} Q'$ *implies* $\exists P'.\, P \overset{\alpha:k}{\underset{L}{\Longrightarrow}} P'$, $L = \mathcal{U}^{<k}(Q)$, *and* $P' \,\underline{\approx}\, \mathcal{R}\, Q'$.

As desired, a *prioritized observational congruence up to* $\underline{\approx}^{\mathrm{l}}$ is a sub-relation of prioritized observational congruence.

The remainder of this section is concerned with the proof of the "largest" part of the above theorem. Our proof is an instance of the following result from universal algebra which can easily be established by using Proposition 2.4.3 (cf. [133]).

Proposition 2.4.16 *Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be equivalences over an algebra $\Re$ such that $\mathcal{R}_1^+ \subseteq \mathcal{R}_2 \subseteq \mathcal{R}_1$. Then $\mathcal{R}_1^+ = \mathcal{R}_2^+$ holds.*

For our purposes, we choose $\mathcal{R}_1 = \approx_\times$ and $\mathcal{R}_2 = \underline{\approx}$. First, we establish that $\mathcal{R}_2^+ = \underline{\approx}^{\mathrm{l}}$ along the lines of a similar proof in [59]. In the sequel we use $\mathcal{S}(P)$ to denote the *sort* of the process P, i.e. the set of all visible actions occurring as labels of the transition system corresponding to P. The following property of the sort of $\mathsf{CCS}^{\mathsf{ch}}$ processes is of importance.

Lemma 2.4.17 (Finite Sorts) *For $P \in \mathcal{P}$ its sort $\mathcal{S}(P)$ is finite.*

The validity of the lemma is a consequence of the fact that the summation operator is binary and that relabelings f satisfy the property $|\{\alpha{:}k \mid f(\alpha{:}k) \not\equiv \alpha{:}k\}| < \infty$. The lemma is used in the proof of the following theorem.

Theorem 2.4.18 *The relation $\underline{\approx}^{\mathrm{l}}$ is the* largest *congruence contained in $\underline{\approx}$.*

Proof: Since the relation $\underline{\approx}^{\mathrm{l}}$ is a congruence, which is contained in $\underline{\approx}$ because $\underline{\approx}^{\mathrm{l}}$ can be shown to be a prioritized weak bisimulation, we know that $\underline{\approx}^{\mathrm{l}} \subseteq \underline{\approx}^{+}$ holds. For the other inclusion, consider the equivalence $\underline{\approx}^{\mathrm{l}}_{\mathrm{a}} =_{\mathrm{df}} \{\langle P, Q\rangle \mid C_{PQ}[P] \underline{\approx} C_{PQ}[Q]\}$ where $C_{PQ}[X] \overset{\mathrm{def}}{=} X + \sum_{k \leq \mathtt{min}} c{:}k.\mathbf{0}$. Here, `min` stands for the maximal priority value, i.e. the

minimal priority, of actions in $\mathcal{S}(P) \cup \mathcal{S}(Q)$. The actions $c{:}k$ are supposed to be "fresh" ones, i.e. they are not in the sort of the processes P and Q under consideration. Note that those actions exist by Lemma 2.4.17. Obviously, $\underline{\approx}^+ \subseteq \underline{\approx}^{\mathrm{l}}_{\mathrm{a}}$ according to Proposition 2.4.3. For the proof of the remaining inclusion $\underline{\approx}^{\mathrm{l}}_{\mathrm{a}} \subseteq \underline{\approx}^{\mathrm{l}}$, let $P, Q \in \mathcal{P}$ such that $P \,\underline{\approx}^{\mathrm{l}}_{\mathrm{a}}\, Q$. We have to show that P and Q are prioritized observational congruent.

1. Assume $\tau{:}k \notin \mathcal{I}^{<k}(P)$ for $k \leq \mathtt{min}$. Hence, $C_{PQ}[P] \xrightarrow{c:k} \mathbf{0}$. Since $C_{PQ}[P] \,\underline{\approx}\, C_{PQ}[Q]$ and $c{:}k \notin \mathcal{S}(Q)$ necessarily $C_{PQ}[Q] \stackrel{c:k}{\underset{L}{\Longrightarrow}} \mathbf{0}$ holds, where $L = \mathit{\Pi}^{<k}(P) \cup \{c{:}l \mid l < k\}$. This requires $\tau \notin \mathcal{I}^{<k}(Q)$ and $\mathit{\Pi}^{<k}(Q) \subseteq \mathit{\Pi}^{<k}(P)$. Hence, $\mathcal{I}^{<k}(Q) \subseteq \mathcal{I}^{<k}(P)$.

2. Let $P \xrightarrow{\alpha:k} P'$ for some $P' \in \mathcal{P}$ and some $\alpha{:}k \in \mathcal{A}$, i.e. $C_{PQ}[P] \xrightarrow{\alpha:k} P'$. Since $C_{PQ}[P] \,\underline{\approx}\, C_{PQ}[Q]$ there exists some $Q' \in \mathcal{P}$ satisfying $C_{PQ}[Q] \stackrel{\hat{\alpha}:k}{\underset{L'}{\Longrightarrow}} Q'$, $L' = \mathit{\Pi}^{<k}(P) \cup \{c{:}l \mid l < k\}$ and $P' \,\underline{\approx}\, Q'$. We know that $Q' \not\equiv C_{PQ}[Q]$ because $P' \,\underline{\approx}\, Q'$ and P' is *not* capable of performing a prioritized weak $c{:}0$-transition. Therefore, the matching step is necessary, even if $\alpha \equiv \tau$. Thus, $Q \stackrel{\alpha:k}{\underset{L}{\Longrightarrow}} Q'$ and $P' \,\underline{\approx}\, Q'$ for $L = \mathit{\Pi}^{<k}(P)$.

Since also the symmetric properties hold, all conditions of Definition 2.4.10 are satisfied, and we obtain $P \,\underline{\approx}^{\mathrm{l}}\, Q$, as desired. □

In order to apply Proposition 2.4.16, we further have to show that $\mathcal{R}_1^+ \subseteq \mathcal{R}_2 \subseteq \mathcal{R}_1$, i.e. $\approx_\times^+ \subseteq \underline{\approx} \subseteq \approx_\times$. The inclusion $\underline{\approx} \subseteq \approx_\times$ follows immediately from the definition of the naive prioritized weak and the prioritized weak transition relation. In order to apply Proposition 2.4.16, we have to establish $\approx_\times^+ \subseteq \underline{\approx}$. This inclusion turns out to be difficult to show directly. Therefore, we define an auxiliary equivalence

$$\underline{\approx}_{\mathrm{a}} =_{\mathrm{df}} \{\langle P, Q\rangle \mid C_{PQ}[P] \approx_\times C_{PQ}[Q]\}$$

which lies in between. Using the abbreviation $S =_{\mathrm{df}} \mathcal{S}(P) \cup \mathcal{S}(Q)$, let $C_{PQ}[X] \stackrel{\mathrm{def}}{=} X \mid H_{PQ}$ and

$$H_{PQ} \stackrel{\mathrm{def}}{=} c{:}0.\mathbf{0} + \sum_{\substack{L \subseteq \overline{S},\\ b:k \,\in\, S \cup \{\tau:k \mid k \leq \mathtt{min}\}}} \tau{:}0. \left(\begin{array}{c} \tau{:}k.\mathbf{0} + \\ d_{L,b:k}{:}0.H_{PQ} + \\ D_L + e{:}k.H_{PQ} + \\ \overline{b}{:}k.H_{PQ} \end{array}\right)$$

where $\mathtt{min}$ is defined as in the proof above. For notational convenience we define $\overline{b}{:}k =_{\mathrm{df}} \tau{:}k$ if $b{:}k \equiv \tau{:}k$. Note that H_{PQ} is well-defined by Lemma 2.4.17. Moreover, D_L is defined as $\sum_{\alpha:k \in L} \alpha{:}k.\mathbf{0}$, and the actions $c{:}0, d_{L,b:k}{:}0, e{:}k$ for all $L \subseteq \overline{S}$ and $b{:}k \in S$, and their complements, are supposed to be "fresh" actions (cf. Lemma 2.4.17). The context $C_{PQ}[X]$ is inspired by one in [133]. By Proposition 2.4.3 we may conclude $\approx_\times^+ \subseteq \underline{\approx}_{\mathrm{a}}$. The other necessary inclusion is established by the following proposition whose proof can be found in Appendix A due to its length.

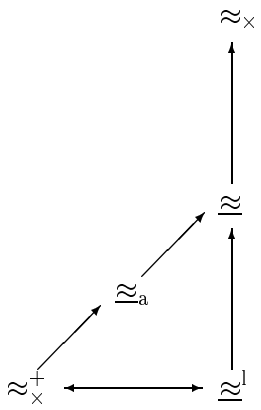

Figure 2.3: Situation in the proof of Theorem 2.4.11

Proposition 2.4.19 *The inclusion* $\underline{\approx}_a \subseteq \underline{\approx}$ *holds.*

The above proposition completes the proof of Theorem 2.4.11 as illustrated by Figure 2.3. Here, an arrow from relation R_1 to relation R_2 means that $R_1 \subseteq R_2$. More precisely, we have already established all premises of Proposition 2.4.16. Thus, $\mathcal{R}_1^+ = \mathcal{R}_2^+$, i.e. $\approx_\times^+ = \underline{\approx}^+$. Moreover, we have shown in Theorem 2.4.18 that $\underline{\approx}^+ = \underline{\approx}^l$. Hence, $\underline{\approx}^l = \approx_\times^+$, as desired.

We close this section by returning to our illustrating back-and-forth example. Because of the facts that $\texttt{Sys} \underline{\approx} \texttt{Spec}$, that both processes only possess visible initial actions, and that their initial action sets are identical, we can conclude $\texttt{Sys} \underline{\approx}^l \texttt{Spec}$ from Definition 2.4.10, as intuitively expected. In contrast, the prioritized observational congruence developed in [133, 134] does not relate our system and its specification due to the presence of the unprioritized internal action τ in `Sys`. It is this additional freedom of abstracting from internal transitions which makes our calculus and its semantic theory more useful in practice than the one reported in [54, 133, 134].

2.4.3 Operational Characterization

The aim of this section is to show how prioritized weak bisimulation can be efficiently computed by adapting partition-refinement algorithms [103, 138] developed with respect to standard bisimulation [125]. Therefore, we provide an operational characterization of prioritized weak bisimulation as standard bisimulation by introducing an *alternative prioritized weak transition relation*. It uses the notation $\stackrel{\epsilon:<k}{\Longrightarrow}$ for the relation $(\{\stackrel{\tau:l}{\longrightarrow} \mid l < k\})^*$ whenever $k > 0$.

Definition 2.4.20 (Alternative Prioritized Weak Transition Relation)
For $P, P' \in \mathcal{P}$ *and* $\alpha{:}0, \alpha{:}k \in \mathcal{A}$*, where* $k > 0$*, we define*

1. $\stackrel{\hat{\alpha}:0}{\Longrightarrow}_* =_{df} \stackrel{\hat{\alpha}:0}{\Longrightarrow}$ *and*

2. $P \stackrel{\hat{\alpha}:k}{\Longrightarrow}_* P'$ *if* $\exists P'' \in \mathcal{P}.\ \tau \notin \mathcal{I}^{<k}(P'')$, $P \stackrel{\epsilon:<k}{\Longrightarrow} P''$, *and* $P'' \stackrel{\hat{\alpha}:k}{\underset{L}{\Longrightarrow}} P'$ *where* $L =_{df} \mathcal{U}^{<k}(P'')$.

Observe that the alternative prioritized weak transition relation is not any longer parameterized by initial action sets. The importance of the new weak transition relation results from the coincidence of $\underline{\approx}$ with the relation $\underline{\approx}_*$ which is defined as standard bisimulation over the alternative prioritized weak transition relation.

Definition 2.4.21 (Alternative Prioritized Weak Bisimulation)
A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ *is called an* alternative prioritized weak bisimulation *if for all* $\langle P, Q\rangle \in \mathcal{R}$ *and* $\alpha{:}k \in \mathcal{A}$ *the following condition holds:*

$$P \stackrel{\hat{\alpha}:k}{\Longrightarrow}_* P' \text{ implies } \exists Q'.\, Q \stackrel{\hat{\alpha}:k}{\Longrightarrow}_* Q' \text{ and } \langle P', Q'\rangle \in \mathcal{R}\ .$$

We write $P \underline{\approx}_* Q$ *if* $\langle P, Q\rangle \in R$ *for an alternative prioritized weak bisimulation* $\mathcal{R}$.

Note that in the above definition, Condition (1) of Definition 2.4.5 is already encoded in the requirement for $P \stackrel{\epsilon:k}{\Longrightarrow}_* P$. As desired, we obtain the following result.

Theorem 2.4.22 (Operational Characterization)
Prioritized and alternative prioritized weak bisimulation coincide, i.e. $\underline{\approx} = \underline{\approx}_*$.

Using the alternative prioritized weak transition relation and Theorem 2.4.22 we may apply standard algorithms for partition refinement [103, 138] in order to compute $\underline{\approx}$. The computation of the alternative prioritized weak transition relation can be done efficiently using *dynamic programming* techniques. The following proposition is central for the proof of Theorem 2.4.22.

Proposition 2.4.23 *Let* $P, Q, Q' \in \mathcal{P}$ *and* $k > 0$ *satisfy* $\tau \notin \mathcal{I}^{<k}(P)$ *and* $\tau \notin \mathcal{I}^{<k}(Q')$, $P \underline{\approx} Q$, *and* $Q \stackrel{\epsilon:<k}{\Longrightarrow} Q'$. *Then we have* $P \underline{\approx} Q'$ *and* $\mathcal{U}^{<k}(P) = \mathcal{U}^{<k}(Q')$. *Moreover,* $Q \stackrel{\epsilon:k}{\underset{L}{\Longrightarrow}} Q'$ *where* $L = \mathcal{U}^{<k}(P)$. *Finally, the same statements hold if* $\underline{\approx}$ *is replaced by* $\underline{\approx}_*$.

Proof: We prove the proposition by induction on the length i of the weak transition $Q \stackrel{\epsilon:<k}{\Longrightarrow} Q'$. For the induction base let $i = 0$, i.e. $Q \equiv Q'$. Hence, $P \underline{\approx} Q'$ since $P \underline{\approx} Q$. Moreover, let $a{:}l \in \mathcal{U}^{<k}(P)$ which implies $l < k$ and $P \stackrel{a:l}{\longrightarrow} P'$ for some $P' \in \mathcal{P}$ by the definition of initial action sets and by Proposition 2.2.1(1). Because of $P \underline{\approx} Q$ we know of the existence of some $Q'' \in \mathcal{P}$ such that $Q \stackrel{a:l}{\underset{L}{\Longrightarrow}} Q''$ where $L = \mathcal{U}^{<l}(P)$ and $P' \underline{\approx} Q''$. This implies $Q \stackrel{a:l}{\underset{L}{\longrightarrow}} \stackrel{\epsilon:0}{\Longrightarrow} Q''$ by the definition of the prioritized weak transition relation since $\tau \notin \mathcal{I}^{<l}(Q)$. Hence, $a{:}l \in \mathcal{U}^{<k}(Q)$ because of $l < k$. Due to symmetry we may also conclude $\mathcal{U}^{<k}(Q) \subseteq \mathcal{U}^{<k}(P)$. Thus, $\mathcal{U}^{<k}(P) = \mathcal{U}^{<k}(Q)$, as desired.

For the induction step let $Q \stackrel{\tau:l}{\longrightarrow} Q'' \stackrel{\epsilon:<k}{\Longrightarrow} Q'$ for some $Q'' \in \mathcal{P}$ and $l < k$. Since $P \underline{\approx} Q$ we have $P \stackrel{\epsilon:l}{\underset{L}{\Longrightarrow}} P'$ for some $P' \underline{\approx} Q''$ where $L = \mathcal{U}^{<l}(Q)$. However, $\tau \notin \mathcal{I}^{<k}(P)$, i.e. the weak

transition is trivial. Hence, $P \equiv P'$ and, therefore, $P \underline{\approx} Q''$. Now, we have established the premises in order to apply the induction hypothesis which allows us to finish the induction step straightforwardly.

The statement $Q \stackrel{\epsilon:k}{\underset{L}{\Longrightarrow}} Q'$ where $L = \mathit{\Pi}^{<k}(P)$ is a consequence of the already established properties since $\tau \notin \mathcal{I}^{<k}(P)$ implies $\tau \notin \mathcal{I}^{<l}(P)$, and $\mathit{\Pi}^{<l}(P) \subseteq \mathit{\Pi}^{<k}(P)$ for all $l < k$. Finally, the above statements can be proved in the same fashion if one replaces $\underline{\approx}$ by $\underline{\approx}_*$, due to the definition of the alternative prioritized weak transition relation. □

The next lemma turns out to be useful for the proof of Theorem 2.4.22.

Lemma 2.4.24 *Let $P, P', Q \in \mathcal{P}$ such that $P \underline{\approx} Q$ and $P \stackrel{\epsilon:0}{\Longrightarrow} P'$. Then there exists some $Q' \in \mathcal{P}$ satisfying $Q \stackrel{\epsilon:0}{\Longrightarrow} Q'$ and $P' \underline{\approx} Q'$.*

The proof of this lemma can be done straightforwardly by induction on the length of the weak transition $P \stackrel{\epsilon:0}{\Longrightarrow} P'$. Now, we turn to the proof of Theorem 2.4.22.

Proof: For proving $\underline{\approx}_* = \underline{\approx}$ we establish both inclusions.

- For the inclusion "$\subseteq$" it is sufficient to establish that the symmetric relation $\underline{\approx}_*$ is a prioritized weak bisimulation. Let $P, Q \in \mathcal{P}$ such that $P \underline{\approx}_* Q$.

  1. Moreover, let $\tau \notin \mathcal{I}^{<k}(P)$ for some $k \in \mathbb{N}$ satisfying $k > 0$. Then $P \stackrel{\epsilon:k}{\Longrightarrow}_* P$ according to the definition of the alternative prioritized weak transition relation. Since $P \underline{\approx}_* Q$ we know of the existence of some $Q' \in \mathcal{P}$ satisfying $Q \stackrel{\epsilon:k}{\Longrightarrow}_* Q'$ and $P' \underline{\approx}_* Q'$. Hence, $Q \stackrel{\epsilon:<k}{\Longrightarrow} Q'' \stackrel{\epsilon:k}{\underset{L}{\Longrightarrow}} Q'$ for some $Q'' \in \mathcal{P}$ where $L = \mathit{\Pi}^{<k}(Q'')$ and $\tau \notin \mathcal{I}^{<k}(Q'')$. By Proposition 2.4.23 for $\underline{\approx}_*$ we obtain $Q \stackrel{\epsilon:k}{\underset{L}{\Longrightarrow}} Q''$, $L = \mathit{\Pi}^{<k}(Q'') = \mathit{\Pi}^{<k}(P)$, and $P \underline{\approx}_* Q''$. Thus, Condition (1) of Definition 2.4.5 is established. Note that this condition trivially holds for $k = 0$.
  2. Now, let $P \stackrel{\alpha:k}{\longrightarrow} P'$ for some $P' \in \mathcal{P}$ and $\alpha{:}k \in \mathcal{A}$ such that $k > 0$. Hence, $\tau \notin \mathcal{I}^{<k}(P)$ and $P \stackrel{\hat{\alpha}:k}{\underset{L'}{\Longrightarrow}} P'$ where $L' = \mathit{\Pi}^{<k}(P)$. Thus, $P \stackrel{\hat{\alpha}:k}{\Longrightarrow}_* P'$ according to the definition of the alternative prioritized weak transition relation. Since $P \underline{\approx}_* Q$ we know of the existence of some $Q' \in \mathcal{P}$ such that $Q \stackrel{\hat{\alpha}:k}{\Longrightarrow}_* Q'$ and $P' \underline{\approx}_* Q'$. Therefore, $Q \stackrel{\epsilon:<k}{\Longrightarrow} Q'' \stackrel{\hat{\alpha}:k}{\underset{L}{\Longrightarrow}} Q'$ for some $Q'' \in \mathcal{P}$ satisfying $L = \mathit{\Pi}^{<k}(Q'')$ and $\tau \notin \mathcal{I}^{<k}(Q'')$. Using Proposition 2.4.23 for $\underline{\approx}_*$ we obtain $P \underline{\approx}_* Q''$, $L = \mathit{\Pi}^{<k}(Q'') = \mathit{\Pi}^{<k}(P)$, and $Q \stackrel{\epsilon:k}{\underset{L}{\Longrightarrow}} Q''$. Consequently, Condition (2) of Definition 2.4.5 is established. The case $k = 0$ follows analogous to the corresponding proofs within the framework of CCS [125] since the considered definitions of the weak transition relations coincide.

  Hence, $\underline{\approx}_* \subseteq \underline{\approx}$ by Definition 2.4.5.

- For the reverse inclusion "$\supseteq$" it is, according to Definition 2.4.21, sufficient to prove that the symmetric relation $\underline{\approx}$ is an alternative prioritized weak bisimulation. Therefore, let $P, Q \in \mathcal{P}$ satisfying $P \underline{\approx} Q$ and $P \stackrel{\hat{\alpha}:k}{\Longrightarrow}_* P'$ for some $P' \in \mathcal{P}$ and $\alpha{:}k \in \mathcal{A}$ such that $k > 0$. Hence, $P \stackrel{\epsilon:<k}{\Longrightarrow} P'' \stackrel{\hat{\alpha}:k}{\underset{L}{\Longrightarrow}} P'$ for some $P'' \in \mathcal{P}$ where $\tau \notin \mathcal{I}^{<k}(P'')$ and $L = \mathcal{U}^{<k}(P'')$. Thus, there exist $r, s \in \mathbb{N}$, $P_i, P_{r+j} \in \mathcal{P}$, $l_i < k$, and $l_{r+j} \leq k$ for $0 \leq i < r$ and $0 \leq j \leq s$ such that $P_0 \equiv P$, $P_r \equiv P''$, $P_i \stackrel{\tau:l_i}{\underset{L}{\longrightarrow}} P_{i+1}$ for all $0 \leq i < r+s$, and $P_{r+s} \stackrel{\hat{\alpha}:k}{\underset{L}{\longrightarrow}} \stackrel{\epsilon:0}{\Longrightarrow} P'$. Note that $l_r = k$ because $\tau \notin \mathcal{I}^{<k}(P_r)$ and $l_r \leq k$.

  Since $P \underline{\approx} Q$ we may successively conclude the existence of $Q', Q_i \in \mathcal{P}$ and $L_i \subseteq \mathcal{A} \setminus \{\tau{:}k \mid k \in \mathbb{N}\}$ such that $Q_i \stackrel{\epsilon:l_i}{\underset{L_i}{\Longrightarrow}} Q_{i+1}$, where $L_i = \mathcal{U}^{<l_i}(P_i)$, and $P_i \underline{\approx} Q_i$ for all $0 \leq i < r+s$, as well as $Q_{r+s} \stackrel{\hat{\alpha}:k}{\underset{L}{\Longrightarrow}} \stackrel{\epsilon:0}{\Longrightarrow} Q'$ and $P' \underline{\approx} Q'$ by Lemma 2.4.24. Because of the premise $\mathcal{U}^{<l_i}(P_i) \subseteq L$ we obtain $L_i \subseteq L$.

  Further, let $\widehat{Q}$ denote the first process on the path of internal transitions starting from Q_r whose outgoing internal transition has priority k, if such a process exist. Thus, $Q \stackrel{\epsilon:<k}{\Longrightarrow} \widehat{Q} \stackrel{\hat{\alpha}:k}{\underset{L}{\Longrightarrow}} Q'$ and $\tau \notin \mathcal{I}^{<k}(\widehat{Q})$ by the choice of $\widehat{Q}$ and since $L_i \subseteq L = \mathcal{U}^{<k}(P'')$.

  Consider the case that no such $\widehat{Q}$ exists. Hence, $\alpha \equiv \tau$ and $P'' \equiv P'$ since priority k is attached to α and $l_r = k$. Because of $P'' \equiv P_r \underline{\approx} Q_r$ we may conclude the existence of another $Q' \in \mathcal{P}$ such that $Q_r \stackrel{\epsilon:k}{\underset{L}{\Longrightarrow}} Q'$, $\mathcal{U}^{<k}(Q') \subseteq L$ where $L = \mathcal{U}^{<k}(P'')$ as above, $\tau \notin \mathcal{I}^{<k}(Q')$, and $P' \equiv P'' \underline{\approx} Q'$. Now, one can find a process $\widehat{Q}$ on the path of internal transitions from Q_r to Q' with the desired properties as mentioned in the previous paragraph; note that Q' itself is a potential candidate for $\widehat{Q}$.

  Summarizing, we have found processes $\widehat{Q}$ and Q' such that $Q \stackrel{\epsilon:<k}{\Longrightarrow} Q_r \stackrel{\epsilon:<k}{\Longrightarrow} \widehat{Q} \stackrel{\hat{\alpha}:k}{\underset{L}{\Longrightarrow}} Q'$ where $L = \mathcal{U}^{<k}(P'')$, $\tau \notin \mathcal{I}^{<k}(\widehat{Q})$, $P'' \equiv P_r \underline{\approx} Q_r$, and $P' \underline{\approx} Q'$. By Proposition 2.4.23 for $\underline{\approx}$ we obtain $\mathcal{U}^{<k}(P'') = \mathcal{U}^{<k}(\widehat{Q})$. Hence, $Q \stackrel{\hat{\alpha}:k}{\Longrightarrow}_* Q'$ and $P' \underline{\approx} Q'$, as desired.

  For $k = 0$ the proof is analogous to the corresponding one within the framework of CCS [125] and uses Lemma 2.4.24.

By Definition 2.4.21 we obtain $\underline{\approx} \subseteq \underline{\approx}_*$, which finishes the proof of Theorem 2.4.22. □

As a consequence of the operational characterization, the Hennessy-Milner logic presented in [125] for CCS can straightforwardly be adapted for logically characterizing $\underline{\approx}$ by replacing the usual transition relation by the alternative prioritized weak transition relation in the definition of the logic's semantics.

2.5 Example

In this section we provide a more complex example showing the utility of our theory for prioritized observational congruence. We apply this congruence to formally reason about the relationship between a specification and an implementation of a system modeling a

4-*counter*. The example is based on one presented in [134] and shows the benefits of our semantic theory compared to the theory developed in [134]. For our purposes it is sufficient to consider a two-level priority-scheme. Hence, we write again $\underline{\alpha}$ for the *prioritized action* $\alpha{:}0$ and α for the *unprioritized action* $\alpha{:}1$.

The 4-counter is designed to output $\overline{\texttt{timeout}}$ immediately after four consecutive inputs of the kind `tick`. Its specification is formally given by the process

$$\texttt{Spec} \stackrel{\text{def}}{=} \texttt{tick.tick.tick.tick.}\overline{\texttt{timeout}}\texttt{.Spec} \ .$$

Suppose we are asked to implement `Spec` using a 2-*count alarm-clock*

$$\texttt{Alarm-Clock} \stackrel{\text{def}}{=} \texttt{tick.tick.}\overline{\underline{\texttt{alarm}}}\texttt{.Alarm-Clock}$$

and a 2-*time repeater*

$$\texttt{Repeater} \stackrel{\text{def}}{=} \underline{b}.\underline{b}.\overline{\texttt{event}}\texttt{.Repeater}$$

which cyclically awaits two interrupts on $\underline{b}$ before signaling $\overline{\texttt{event}}$. A natural way to implement the specification is to compose the 2-count alarm-clock and the 2-time repeater by connecting the $\underline{\texttt{alarm}}$ port of the alarm-clock with the $\underline{b}$ port of the repeater. Moreover, we relabel the interrupt signal `event` of the repeater to `timeout`. Hence, the resulting system `Sys` is defined by $\texttt{Sys} \stackrel{\text{def}}{=} (A \mid B) \setminus \{\underline{b}\}$ where $A \stackrel{\text{def}}{=} \texttt{Alarm-Clock}[\underline{b}/\underline{\texttt{alarm}}]$ and $B \stackrel{\text{def}}{=} \texttt{Repeater}[\texttt{timeout}/\texttt{event}]$.

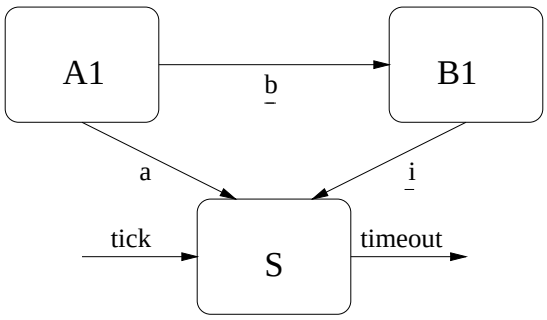

Figure 2.4: Architecture of the 4-counter

Unfortunately, the implementation `Sys` does not satisfy the specification `Spec` with respect to any reasonable behavioral relation since `Sys` can engage in a sequence of five consecutive `tick` transitions whereas `Spec` obviously cannot. In other words, `Sys` and `Spec` are not even *trace equivalent*. The same argument also holds for frameworks which do not allow one to model priority at all, as e.g. for CCS [125]. The reason for this inequivalence is that the availability of the action `timeout` in process B does not preclude process A to perform a `tick`. However, prioritizing the action $\overline{\texttt{timeout}}$ in the above definitions does not help since visible prioritized transitions do not pre-empt unprioritized transitions. One possible solution to this problem is to introduce a *server* process S through

which interactions between the environment and the system are passed; for the sake of uniformity not only `timeout` but also `tick` actions. Hence, we consider the following modified system $\texttt{Sys1} \stackrel{\text{def}}{=} (A1 \,|\, B1 \,|\, S) \setminus \{a, \underline{b}, \underline{i}\}$ where $A1 \stackrel{\text{def}}{=} a.a.\overline{\underline{b}}.A1$, $B1 \stackrel{\text{def}}{=} \underline{b}.\underline{b}.\overline{\underline{i}}.B1$, and $S \stackrel{\text{def}}{=} \texttt{tick}.\overline{a}.S + \underline{i}.\overline{\texttt{timeout}}.S$. Its architecture is illustrated in Figure 2.4.

Table 2.5: A relation whose symmetric closure is a prioritized weak bisimulation

$\{\ \langle$ `Sys1`	,	`Spec` $\rangle$,
$\langle\ (A1 \,\|\, B1 \,\|\, \overline{a}.S) \setminus \{a, \underline{b}, \underline{i}\}$	,	`tick.tick.tick.`$\overline{\texttt{timeout}}$`.Spec` $\rangle$,
$\langle\ (a.\overline{\underline{b}}.A1 \,\|\, B1 \,\|\, S) \setminus \{a, \underline{b}, \underline{i}\}$	,	`tick.tick.tick.`$\overline{\texttt{timeout}}$`.Spec` $\rangle$,
$\langle\ (a.\overline{\underline{b}}.A1 \,\|\, B1 \,\|\, \overline{a}.S) \setminus \{a, \underline{b}, \underline{i}\}$	,	`tick.tick.`$\overline{\texttt{timeout}}$`.Spec` $\rangle$,
$\langle\ (\overline{\underline{b}}.A1 \,\|\, B1 \,\|\, S) \setminus \{a, \underline{b}, \underline{i}\}$	,	`tick.tick.`$\overline{\texttt{timeout}}$`.Spec` $\rangle$,
$\langle\ (A1 \,\|\, \underline{b}.\overline{\underline{i}}.B1 \,\|\, S) \setminus \{a, \underline{b}, \underline{i}\}$	,	`tick.tick.`$\overline{\texttt{timeout}}$`.Spec` $\rangle$,
$\langle\ (A1 \,\|\, \underline{b}.\overline{\underline{i}}.B1 \,\|\, \overline{a}.S) \setminus \{a, \underline{b}, \underline{i}\}$	,	`tick.`$\overline{\texttt{timeout}}$`.Spec` $\rangle$,
$\langle\ (a.\overline{\underline{b}}.A1 \,\|\, \underline{b}.\overline{\underline{i}}.B1 \,\|\, S) \setminus \{a, \underline{b}, \underline{i}\}$	,	`tick.`$\overline{\texttt{timeout}}$`.Spec` $\rangle$,
$\langle\ (a.\overline{\underline{b}}.A1 \,\|\, \underline{b}.\overline{\underline{i}}.B1 \,\|\, \overline{a}.S) \setminus \{a, \underline{b}, \underline{i}\}$	,	$\overline{\texttt{timeout}}$`.Spec` $\rangle$,
$\langle\ (\overline{\underline{b}}.A1 \,\|\, \underline{b}.\overline{\underline{i}}.B1 \,\|\, S) \setminus \{a, \underline{b}, \underline{i}\}$	,	$\overline{\texttt{timeout}}$`.Spec` $\rangle$,
$\langle\ (A1 \,\|\, \overline{\underline{i}}.B1 \,\|\, S) \setminus \{a, \underline{b}, \underline{i}\}$	,	$\overline{\texttt{timeout}}$`.Spec` $\rangle$,
$\langle\ (A1 \,\|\, \overline{\underline{i}}.B1 \,\|\, \overline{\texttt{timeout}}.S) \setminus \{a, \underline{b}, \underline{i}\}$	,	$\overline{\texttt{timeout}}$`.Spec` $\rangle\ \}$

Now, we are able to show that `Sys1` and `Spec` are related with respect to prioritized weak bisimulation. This can be verified using Table 2.5 which presents a relation whose symmetric closure is a prioritized weak bisimulation containing $\langle$`Sys1`, `Spec`$\rangle$. Note that the validation of Condition (1) of Definition 2.4.5 is easy since the semantics of both processes, `Sys1` and `Spec`, do not contain any visible prioritized actions. Finally, `Sys1` and `Spec` do possess the same initial actions, namely the action `tick`, such that we may conclude $\texttt{Sys1} \underline{\approx}^{\mathrm{l}} \texttt{Spec}$ by Definition 2.4.10, too.

2.6 Discussion and Related Work

In this section we compare the advantages of our algebraic framework by considering definitions of prioritized weak bisimulation and of prioritized observational congruence proposed in [134]. Moreover, we discuss related work on approaches to priority which are based on global pre-emption.

2.6.1 Discussion

In this chapter we have shown that our notions of prioritized weak equivalence and prioritized observational congruence are suitable from an algebraic point of view, especially since $\underline{\approx}^{l}$ is the largest congruence contained in $\approx_{\times}$. However, there may also be other interesting congruences which, on the one hand, do not abstract that much from internal transitions but, on the other hand, can be computed more efficiently. One particularly interesting candidate for such a behavioral relation is a slight variant of the prioritized observational congruence developed for Cleaveland and Hennessy's calculus with priority [54, 134]. This calculus is mainly distinguished from $\mathsf{CCS}^{\mathsf{ch}}$ by its fixed two-level priority-scheme and the presence of *prioritization* and *deprioritization* operators which allow one to change priority values of actions. In particular, [134] introduces a finer weak transition relation than ours since a weak unprioritized α-transition consists of an α-transition that is preceded and trailed by *prioritized* internal transitions corresponding to $\tau:0$-transitions in our calculus. Hence, the approach in [134] just abstracts from internal actions with priority 0. The reason for this restriction is that, otherwise, prioritized weak bisimulation would not be compositional with respect to the prioritization and deprioritization operators. However, this finer transition relation can obviously be computed more efficiently than our alternative prioritized weak transition relation since initial action set containments do not play such a dominant role. In contrast, our prioritized weak transition relation allows an $\alpha:k$-transition to be preceded by any sequence of $\tau:l$-transitions (satisfying a condition on initial action sets), where $l \leq k$, and only to be trailed by $\tau:0$-transitions.

From our point of view, operators changing priority values of actions are not desirable when dealing with priority in order to describe *interrupts* or certain programming language constructs such as *prioritized choice*. Those priorities are chosen by the programmer and are fixed once and for all, i.e. they are static. Contradicting this, the prioritization and deprioritization operators allow one to change priority values *dynamically* within contexts. As seen above, leaving out these operators results in a coarser observational congruence that is more practicable since it abstracts from more internal computation. As an example confirming this statement re-consider the 4-counter. In fact, `Sys` and `Spec` are related by the prioritized observational congruence developed in this chapter, as seen in the previous section, but *not* by the observational congruence suggested in [134]. The reason for the latter is that the processes $\texttt{Sys}' \stackrel{\text{def}}{=} (A1 \,|\, B1 \,|\, \overline{a}.S) \setminus \{a, \underline{b}, \underline{i}\}$ and $\texttt{Spec}' \stackrel{\text{def}}{=} \texttt{tick.tick.tick.}\overline{\texttt{timeout}}\texttt{.Spec}$ must necessarily be prioritized weak bisimular. However, $\texttt{Sys}'$ cannot match the `tick`-transition of $\texttt{Spec}'$ since the former process can neither initially engage in a `tick`-transition nor in a $\underline{\tau}$-transition, but only in an unprioritized τ-transition which leads to a `tick`-transition.

A very important insight obtained in this chapter is that the generalization of the calculus presented in [54] to a multi-level priority-scheme has no consequences on the development of a suitable notion of prioritized observational congruence. The additional freedom gained in our calculus for abstracting away internal transitions is solely due to the absence of prioritization and deprioritization operators.

2.6.2 Other Related Work

Baeten, Bergstra, and Klop were the first researchers who introduced priorities into process algebras [10] by conservatively extending the *Algebra of Communicating Processes* (ACP) [17]; a process algebra which is equipped with an axiomatic semantics. Their work is inspired by the insight that it is essential to incorporate an interrupt mechanism in process-algebraic frameworks in order to enhance their expressive power as specification and verification formalisms for concurrent systems. Therefore, a piece of syntax together with semantics defining equations is introduced. Based on a given partial order $<$ on actions a unary operator θ is defined. Intuitively, $\theta(P)$ is the context of P in which action a has precedence over action b whenever $b < a$, i.e. non-deterministic choices between actions a and b are resolved within $\theta(P)$. Technically, the axiomatic semantics of the new language, given in terms of a rewrite system, is shown to possess nice algebraic properties such as confluence and termination. The utility of the theory is demonstrated by simple examples dealing with interrupts, timeouts, and aspects of real-time behavior. The approach in [10] differs from the work presented in this thesis in that the partial order expressing priorities is fixed with respect to the system under consideration, i.e. the same priority relation holds at all states of the system. For example, if $a < b$ at some state of the system, then $a > b$ cannot be valid at another state. Moreover it should be mentioned that the version of ACP used in [10] does not include a designated internal action, cf. action τ in CCS; a fact which simplifies the development of an algebraic theory.

Gerber and Lee have developed a real-time process algebra, called *Calculus of Communicating Shared Resources* (CCSR) [82], that explicitly takes the availability of system resources into account. Semantically, synchronizations between processes are modeled in an interleaving fashion using instant transitions, whereas the access of resources is truly concurrent and consumes time. In CCSR a priority structure may be defined over resources which indicates their urgency. It can be used in order to ensure that deadlines are met. The underlying concept of priority is taken from $\mathsf{CCS}^{\mathsf{ch}}$, i.e. priorities are static and pre-emption is global. In [83] a resource-based prioritized (strong) bisimulation for CCSR together with a congruence result and axiomatizations with respect to several classes of processes [36] have been given.

Prasad has also extended his *Calculus of Broadcasting Systems* (CBS) [144] for dealing with a notion of static priority [145]. He refers to the priority calculus as PCBS. For PCBS nice semantic theories based on Milner's strong and weak bisimulation [125] have been developed along with congruence proofs. Remarkably, these theories do not suffer from the technical subtleties which have been encountered for $\mathsf{CCS}^{\mathsf{ch}}$, although the concept of pre-emption is basically the same. The reason is that PCBS is concerned with a very different but much simpler model for communication. In PCBS communication is based on the principle of *broadcasting* such as used for public address systems. Moreover, priority values are only attached to output actions which cannot be restricted or hidden like in traditional process algebras. Finally, it should be mentioned that PCBS contains an operator, called *translate*, which allows for prioritization and deprioritization of actions.

2.7 Summary

We have presented a CCS-based calculus with priority, called $\mathsf{CCS}^{\mathsf{ch}}$, which is distinguished from Cleaveland and Hennessy's calculus presented in [54, 134] by allowing a *multi-level* priority-scheme as well as the disabling operator known from LOTOS and disallowing the prioritization and deprioritization operators. We have developed a semantic theory for $\mathsf{CCS}^{\mathsf{ch}}$ based on Park and Milner's notion of bisimulation. Whereas the usual definition of strong bisimulation is a congruence for $\mathsf{CCS}^{\mathsf{ch}}$, the naive adaptation of observational congruence is not. Consequently, we have devoted our attention to characterizing the largest congruence, referred to as prioritized observational congruence, contained in the naive weak bisimulation. We have compared prioritized observational congruence with a similar congruence as inspired by [134]. It turns out that the latter behavioral relation can be computed more efficiently but does not abstract as much from internal transitions as prioritized observational congruence does and as much as is necessary for dealing with realistic applications.

Chapter 3

Case Study: A Railway Signaling System

3.1 Introduction

This chapter presents a real-world case study which shows the benefits of priority for modeling and verifying concurrent systems. Our example is based on a case study by Glenn Bruns [39] that deals with the design of a safety-critical part of a network used in *British Rail's Solid State Interlocking* (SSI) [69], a system which controls railway signals and points. Bruns has modeled and verified a high-level design of the system that has abstracted from low-level implementation details. He has used plain CCS [125] for modeling the system, a temporal logic [109, 118] for specifying properties of the system, and the *Edinburgh Concurrency Workbench* [64] for verifying that these properties hold for the model.

We investigate an elaboration of Bruns' case study using the process algebra $\mathsf{CCS}^{\mathsf{ch}}$ with priority developed in the previous chapter, which allows a more intuitive modeling of the signaling system. We also augment Bruns' model in two ways based on key concepts of the SSI system described in the original design document [69]. First we add an *error-recovery scheme* that is invoked when a link fails, and second we insert a *backup line* in order to make the system fault-tolerant. Since in both cases *interrupt mechanisms* come into play, the use of a process algebra with priority is mandatory and, moreover, leads to elegant models. We also show that, by eliminating invalid interleavings, priorities can dramatically cut the number of states and transitions in our systems. This is particularly significant since the large complexity of practical problems often prevents their automatic analysis. We verify our models by showing that several safety properties hold using the *Concurrency Workbench of North-Carolina* [65, 153], CWB-NC. This verification tool is a re-implementation of the Edinburgh Concurrency Workbench that offers similar functionality, but is faster and more memory-efficient and gives diagnostic information when a verification routine returns false.

The remainder of this chapter is structured as follows. Section 3.2 gives an introduction to the railway signaling system in question. In the following section we adapt Bruns' model

of the considered network used in a safety-critical railway signaling system. Further, we extend the model by an error-recovery scheme in Section 3.4 and a fault-tolerant network link in Section 3.5. Section 3.6 discusses our verification results whereas the last section contains a summary of this chapter.

3.2 A Railway Signaling System

Our example is embedded in *British Rail's Solid State Interlocking* (SSI) system [39, 69], which adjusts and controls *signals* and *points* along rail routes. Its aim is to prevent situations that may lead to a collision or derailment of trains. Therefore, a formal verification of the design of the SSI and its environment is of particular importance.

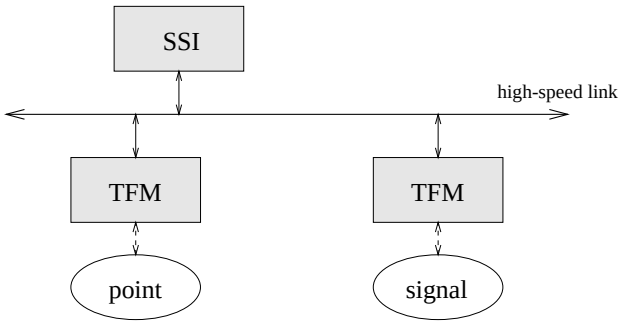

Figure 3.1: The SSI environment – overview

Figure 3.1 shows the basic design of the interlocking system. It consists of three different components: the SSI, several *trackside functional modules* (TFM), and a *high-speed link* which connects the TFMs with the SSI. The SSI is the main logical unit of the system. It is connected to a control panel to which a signal operator can input commands. The SSI checks the validity of those commands and sends them to the TFMs along the track via the high-speed link. A TFM connects a signal or a point to the network. Its task is to listen to the network in order to receive messages for adjusting its signal or point and to send status information about the signal or point to the SSI.

The pattern of communication between the SSI and the TFMs is as follows. The SSI cyclically sends a message to each TFM. The message includes the TFM's address and the status for the corresponding signal or point (e.g. signal on/off). After sending a message the SSI waits a short time for the addressed TFM to respond with the current state of its signal or point. This *polling* scheme reflects the safety-critical design of the system since it leads to a quick detection of failures. For example, if the addressed TFM does not respond then either the TFM or the connection between the SSI and the TFM is down. Moreover, if the corresponding signal or point has autonomously changed its state, it is forced to return to its proper state. The disadvantage of this polling scheme is its communication overhead, which necessitates an expensive high-speed network. This expense is even worse if the distance between some TFMs and the SSI is very large. Therefore, the question arises as to

whether distant high-speed links can be connected via a low-grade link without violating safety properties. Our case study concentrates on this aspect of the SSI system since the use of a high-speed link is known to satisfy the requirements on the error-free delivery of commands and timely detection of failures [39].

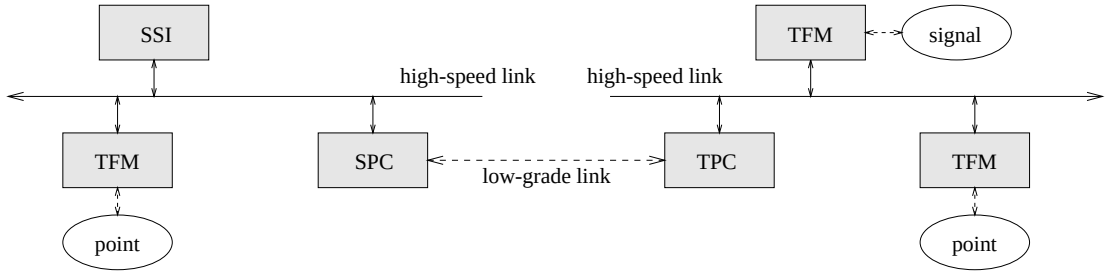

Figure 3.2: The slow-scan system – overview

The integration of a low-grade link (LGL) is illustrated in Figure 3.2. It is connected to the SSI-side high-speed link via a *SSI-side protocol converter* (SPC) and to the TFM-side high-speed link via a *TFM-side protocol converter* (TPC). Intuitively, the SPC is expected to behave like the TFMs on the other side of the LGL, i.e. to accept commands for those TFMs and to respond to the SSI with their current states, but the SPC occasionally sends these commands along the LGL and receives new status information about those TFMs. On the other side, the TPC should mimic a SSI. We refer to the part of the system which consists of SPC, LGL, and TPC as the *slow-scan system*. In order not to violate safety conditions of the overall system, the slow-scan system is expected to satisfy properties of the following kind. If the low-grade link fails, then the TPC (SPC) will detect the problem and stop sending messages to the TPC-side TFMs (SSI). The TFMs are also expected to change signals to red and to lock points in their current setting if they stop receiving messages.

3.3 The Slow-Scan Model

In the following, we formally model the slow-scan system in three steps. First, we present the system as in [39] and discuss the advantages of priorities for modeling. In the second step, we augment our model with an *error-recovery scheme* and re-model the low-grade link in a *full-duplex* fashion. Finally, we show how the required fault-tolerance of the system [69] can be reflected in our design.

Figure 3.3 shows the channels between the three parallel components of the slow-scan system; it also includes an additional clock. Since the correct behavior of our system depends on timing constraints, which cannot be modeled in our process algebra directly, we use an explicit clock in our model to signal the progression of time to the SPC and TPC via the channels mcs and mct, respectively. Tables 3.1 and 3.2 contain the model of the slow-scan system as it is accepted by the CWB-NC, where the symbol $*$ introduces comments. The $\mathsf{CCS}^{\mathsf{ch}}$ front-end for the CWB-NC has been generated by the *Process Algebra*

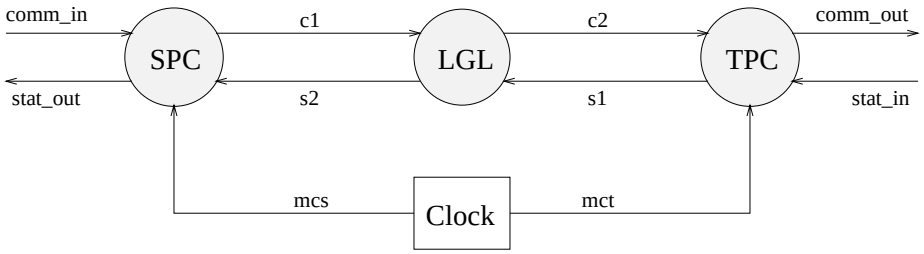

Figure 3.3: The `LGL` model

Table 3.1: The slow-scan model (Part I)

```
* priority value 0: fail_overfull, det, mcs, mct
* priority value 1: ---
* priority value 2: out (c2, s2), fail_wire, comm_in, comm_out, stat_in, stat_out
* priority value 3: in (c1, s1)
* priority value 4: outu (c2u, s2u), tick

* Slow-scan system

proc SS   = (SPC | LGL | TPC | Clock)\{c1:3,c2:2,c2u:4,s1:3,s2:2,s2u:4,
                                       mcs:0,mct:0}

* SSI-side protocol converter (SPC)

proc SPC  = SPC0
proc SPC0 = comm_in:2.'stat_out:2.SPC0 + 'c1:3.SPC0 + s2:2.SPC0 + s2u:4.SPC0 +
            mcs:0.'c1:3.SPC1
proc SPC1 = comm_in:2.'stat_out:2.SPC1 + 'c1:3.SPC1 + s2:2.SPC0 + s2u:4.SPC1 +
            mcs:0.'c1:3.SPC2
proc SPC2 = comm_in:2.'stat_out:2.SPC2 + 'c1:3.SPC2 + s2:2.SPC0 + s2u:4.SPC2 +
            mcs:0.#'det:0.SPCF
proc SPCF = comm_in:2.SPCF + s2:2.SPCF + s2u:4.SPCF + mcs:0.SPCF
```

Compiler [62], `PAC`, which is a generic tool for integrating new interfaces as front-ends of the `CWB-NC`, and uses the following syntactical notations for expressions: `proc x = P` for the process algebra term $\mu x.P$, and `'a:k` for the action $\overline{a}{:}k$. Details about the implementation in `PAC` as well as the `CWB-NC` can be found in [153]. In the following, we sometimes wish to have visible (i.e. non-τ) actions pre-empt actions of lower priority. This happens when the pre-emptive action is signaling some information about events that have just occurred; in this case, the high-priority action plays the role of an "atomic proposition." To give such an action pre-emptive power within the calculus $\mathsf{CCS}^{\mathsf{ch}}$, we insert a $\tau{:}k$ loop at the

Table 3.2: The slow-scan model (Part II)

```
* Track-side protocol converter (TPC)

proc TPC   = TPC0
proc TPC0  = 'comm_out:2.stat_in:2.TPC0 + 's1:3.TPC0 + c2:2.TPC0 + c2u:4.TPC0 +
             mct:0.'s1:3.TPC1
proc TPC1  = 'comm_out:2.stat_in:2.TPC1 + 's1:3.TPC1 + c2:2.TPC0 + c2u:4.TPC1 +
             mct:0.'s1:3.TPC2
proc TPC2  = 'comm_out:2.stat_in:2.TPC2 + 's1:3.TPC2 + c2:2.TPC0 + c2u:4.TPC2 +
             mct:0.#'det:0.TPCF
proc TPCF  = stat_in:2.TPCF + c2:2.TPCF + c2u:4.TPCF + mct:0.TPCF

* Low grade link (LGL)

proc LGL   = Comm[c1:3/in:3,c2:2/out:2,c2u:4/outu:4] |
             Comm[s1:3/in:3,s2:2/out:2,s2u:4/outu:4]

proc Comm  = Comm0
proc Comm0 = in:3.Comm1 + 'outu:4.Comm0 + 'fail_wire:2.CommF
proc Comm1 = in:3.#'fail_overfull:0.CommF + 'out:2.Comm0 + 'fail_wire:2.CommF
proc CommF = in:3.CommF + 'outu:4.CommF

* Clock

proc Clock = 'tick:4.'mcs:0.'mct:0.Clock
```

source states of transitions labeled by $a{:}k$. These loops have the effect of pre-empting all actions of priority lower than k. However, they do not interfere with the verification of the slow-scan model in Section 3.6 since our properties of interest are not sensitive to *divergence*. For convenience, we use the notation $\#a{:}k.P$ as short hand for the process $\mu x.(a{:}k.P + \tau{:}k.x)$.

The low-grade link is modeled by two parallel unidirectional links. Since we are concerned with the design of a system we choose a poor capacity (or bandwidth) link, capacity one for each direction, and we abstract from message headers and contents. Moreover, the `SPC` and `TPC` should be able to deliver a message to and get a message from the medium at any time. If no capacity in a link is left, a new message overwrites a message which is already in the medium, and an overfull error occurs. Therefore, in each direction a link behaves as an input-enabled one-place buffer. Additionally, it offers the action `'outu` to its environment if the buffer is empty. With respect to its reliability, we assume that a link can fail because of a broken wire (action `'fail_wire`) or if its buffering capacity is

exceeded (action 'fail_overfull). If an error has occurred, the medium enters the error state CommF in which it only accepts but never delivers any messages.

The states of the SPC are parameterized by a time mark. In each state the SPC can accept a message from the SSI (action comm_in) and respond with the appropriate status information of the requested TFM (action 'stat_out). At least once every clock cycle (action 'mcs) the SPC sends a message over the LGL to the TPC (action 'c1) and increases its internal time-counter by changing its state from SPC0 to SPC1 or from SPC1 to SPC2. If the SPC receives a message (action s2) from the TPC within two time units, it resets its internal time-counter to 0 by changing its state to SPC0. Otherwise, the SPC times out (action 'det) and enters the failure state SPCF. In this state the SPC never sends messages to the SSI or TPC again, but it remains input-enabled.

Up to now, we have not discussed how priorities can be used in modeling the system. However, one has probably already noticed that various parts of the model would be counterintuitive without priorities. For example, if a link has no capacity for an additional message, it should favor outputting a message over accepting a new one, as the latter immediately leads to the failure of the link. Similarly, a link should favor accepting a new message instead of signaling that the buffer is empty. If the clock gives a new time pulse by performing the action 'tick, it should immediately inform the SPC and TPC by performing the (interrupt) actions 'mcs and 'mct. In other words, no action should interfere between the actions 'tick and 'mcs and the actions 'mcs and 'mct. Moreover, the actions 'det and 'fail_overfull are signaling failures which have already occurred, i.e. they should not be delayed. Finally, the failure of the LGL is always possible, i.e. no action of the LGL should be able to pre-empt the action 'fail_wire.

Based on these observations, we give the actions 'fail_overfull and 'det – which can be viewed as *atomic propositions* – overall pre-emptive power in our model, i.e. they are translated to #'fail_overfull:0 and to #'det:0, respectively. The actions 'mcs and 'mct are also assigned the highest priority. Thus, they cannot be prevented by any action in the system, and the atomicity of the above mentioned action sequences is guaranteed. Moreover, in the LGL, 'out should have a higher priority than in, and in a higher one than 'outu. The action 'tick is assigned to the lowest priority value, reflecting our design decision to adopt the *maximal progress assumption* of real-time process algebra (cf. Chapters 4 and 7). This assumption states that time may only proceed if the system cannot engage in a communication. Finally, 'fail_wire is assigned the highest priority value with respect to the actions of the LGL. These observations lead to the priority-scheme of actions for the slow-scan model presented at the top of Table 3.1.

3.4 The Recovery Model

The slow-scan model represents a substantial abstraction from reality since it is not capable of recovering from a failure. Therefore, we augment the slow-scan model by an error-recovery scheme and change the design of the medium to a more realistic *full-duplex* version.

Table 3.3: The recovery model (Part I)

```
* Low grade link (LGL)

proc LGL    = Comm00[c1:3/in:3,s1:3/in':3,c2:2/out:2,s2:2/out':2,
                     c2u:4/outu:4,s2u:4/outu':4,ok:1/online:1]

proc Comm00 = in:3.Comm10 + 'outu:4.Comm00 + in':3.Comm01 + 'outu':4.Comm00 +
              'fail_wire:2.CommF + 'online:1.Comm00
proc Comm10 = in:3.#'fail_overfull:0.CommF + 'out:2.Comm00 + in':3.Comm11 +
              'outu':4.Comm10 + 'fail_wire:2.CommF + 'online:1.Comm10
proc Comm01 = in:3.Comm11 + 'outu:4.Comm01 + in':3.#'fail_overfull:0.CommF +
              'out':2.Comm00 + 'fail_wire:2.CommF + 'online:1.Comm01
proc Comm11 = in:3.#'fail_overfull:0.CommF + 'out:2.Comm01 +
              in':3.#'fail_overfull:0.CommF + 'out':2.Comm10 +
              'fail_wire:2.CommF + 'online:1.Comm11
proc CommF  = in:3.CommF + 'outu:4.CommF + in':3.CommF + 'outu':4.CommF +
              'repaired:2.Comm00
```

Full-duplex media have the property that if one direction fails then the other should also be considered as unreliable. In the remainder of this section, the action names of both directions of the link will only differ by a trailing prime. As long as the full-duplex medium which is modeled in Table 3.3 provides service, i.e. it is in one of the states `Comm00`, `Comm10`, `Comm01`, or `Comm11`, an `'online` (`'ok`) is signaled to the environment. In contrast to the slow-scan model, a broken medium can be repaired in the recovery model. This is modeled by the action `'repaired`, which is enabled in the failure state `CommF` and allows the `LGL` to reset to its initial state `Comm00`. The recovery of the system as modeled in Table 3.4 works as follows. If the `SPC` (`TPC`) is in its failure state `SPCF` (`TPCF`) and detects that the medium has been repaired by receiving the action `ok` (`online`), it sends one interrupt (action `'reset`) to the clock and another (action `'init`) to the `TPC` (`SPC`). The invoked interrupt handler of the clock resets the clock. The handler of the `TPC` (`SPC`) agrees to that request by sending an acknowledgment (action `'ack_init`) back to the `SPC` (`TPC`), signaling its re-initialization (action `'recovered`), and resetting itself to its initial state. Since we are dealing with abstract models, we leave it open to an implementation as to how to send interrupt signals between `SPC`, `TPC`, and the clock; e.g. one could use the already-repaired line.

If a link has been repaired, the system should reset itself immediately. Therefore, all actions involving the recovery scheme are interrupt actions. However, they should not be able to interfere with the atomicity of the clock signals (actions `'mcs` and `'mct`). Therefore, the actions `ok`, `online`, `init` and `ack_init` are assigned to priority value one.

Table 3.4: The recovery model (Part II)

```
* priority value 0: fail_overfull, det, recovered, mcs, mct
* priority value 1: online (ok), init, ack_init, reset
* priority value 2: out (c2, s2), fail_wire, repaired,
*                   comm_in, comm_out, stat_in, stat_out
* priority value 3: in (c1, s1)
* priority value 4: outu (c2u, s2u), tick

* Slow-scan system

proc SS     = (SPC | LGL | TPC | Clock)\{c1:3,c2:2,c2u:4,s1:3,s2:2,s2u:4,mcs:0,
                                         mct:0,ok:1,init:1,ack_init:1,reset:1}

* SSI-side (SPC) and track-side (TPC) protocol converters

proc SPC    = SPC0 [> (init:1.'ack_init:1.#'recovered:0.SPC + ack_init:1.SPC)
...
proc SPCF   = comm_in:2.SPCF + s2:2.SPCF + s2u:4.SPCF + mcs:0.SPCF +
              ok:1.'reset:1.'init:1.nil

proc TPC    = TPC0 [> (init:1.'ack_init:1.#'recovered:0.TPC + ack_init:1.TPC)
...
proc TPCF   = stat_in:2.TPCF + c2:2.TPCF + c2u:4.TPCF + mct:0.TPCF +
              ok:1.'reset:1.'init:1.nil

* Clock

proc Clock  =  Clock0 [> reset:1.Clock
proc Clock0 = 'tick:4.'mcs:0.'mct:0.Clock0
```

The action `'repaired` should never be pre-empted by any communication in which the buffer is involved, so it gets priority value two. Finally, the action `'recovered` is handled as "atomic proposition."

3.5 The Fault-Tolerant Model

We now turn our attention to modeling *fault-tolerance*, which is an essential requirement of the SSI [69]. We have already modeled an error-recovery scheme for the medium, which ensures fault-tolerance on a *software level*. In practice, the *hardware* of the system is also replicated in order to guarantee better safety-critical behavior [69].

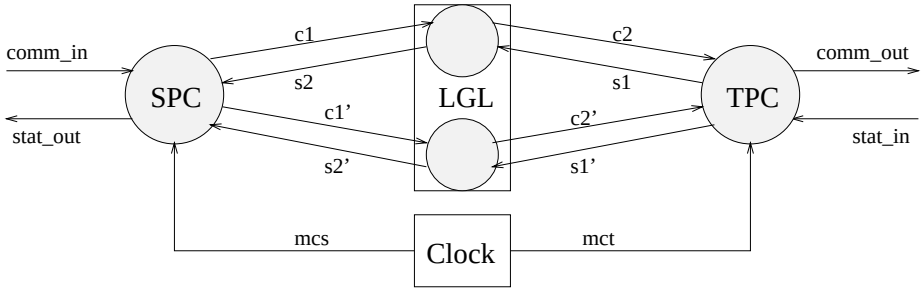

Figure 3.4: The fault-tolerant model

Table 3.5: The fault-tolerant model (Part I)

```
* priority value 0: fail_overfull, det, recovered, mcs, mct
* priority value 1: online (ok, ok'), init, ack_init, switch, ack_switch, reset
* priority value 2: out (c2, s2), out' (c2', s2'), fail_wire, repaired,
*                   comm_in, comm_out, stat_in, stat_out
* priority value 3: in (c1, s1), in' (c1', s1')
* priority value 4: outu (c2u, s2u), outu' (c2u', s2u'), tick

* Slow-scan system

proc SS = (SPC | LGL | TPC | Clock)\{c1:3,c2:2,c2u:4,s1:3,s2:2,s2u:4,c1':3,c2':2,
                                    c2u':4,s1':3,s2':2,s2u':4,mcs:0,mct:0,ok:1,
                                    ok':1,init:1,ack_init:1,reset:1,switch:1,
                                    ack_switch:1}
```

Therefore, we explicitly duplicate the data path in our design. The new situation is depicted in Figure 3.4, where the `LGL` now contains a spare link whose corresponding actions are annotated by a prime. Our fault-tolerant model, where the "prime" states of the `SPC` and `TPC` indicate that the system works on the second line, behaves as follows. If a failure of the currently used line is detected by the `SPC` (`TPC`), i.e. it is in its failure state, and the other line is "up," then the `SPC` (`TPC`) signals its wish to switch the line to the `TPC` (`SPC`) by performing the action `'switch`. The interrupt handlers react to that request (action `'ack_switch`) in the same fashion as in the recovery model. Tables 3.5 and 3.6 summarize the necessary changes to our model.

Ideally, we assume the switch to be atomic in the *design* of the slow-scan system, i.e. `SPC` and `TPC` switch to the new link at the same time. Using priority, this can be modeled by giving the actions `switch` and `ack_switch` the same priority as the interrupt actions `init`, `ack_init`, and `ok`.

Table 3.6: The fault-tolerant model (Part II)

```
* SSI-side protocol converter (SPC) ... (changes to TPC analogous)

proc SPC   = SPC0 [> (init:1.'ack_init:1.#'recovered:0.SPC +
                      ack_init:1.SPC +
                      switch:1.'ack_switch:1.#'recovered:0.SPC' +
                      ack_switch:1.SPC')
...
proc SPCF  = comm_in:2.SPCF + s2:2.SPCF + s2u:4.SPCF + mcs:0.SPCF +
             ok:1.'reset:0.'init:1.nil + ok':1.'reset:0.'switch:1.nil

proc SPC'  = SPC0' [> (init:1.'ack_init:1.#'recovered:0.SPC' +
                       ack_init:1.SPC' +
                       switch:1.'ack_switch:1.#'recovered:0.SPC +
                       ack_switch:1.SPC)

proc SPC0' = comm_in:2.'stat_out:2.SPC0' + 'c1':3.SPC0' + s2':2.SPC0' +
             s2u':4.SPC0' + mcs:0.'c1':3.SPC1'
...
proc SPCF' = comm_in:2.SPCF' + s2':2.SPCF' + s2u':4.SPCF' + mcs:0.SPCF' +
             ok':1.'reset:0.'init:1.nil + ok:1.'reset:0.'switch:1.nil

* Low grade link (LGL)

proc LGL   = Comm00[c1:3/in:3,s1:3/in':3,c2:2/out:2,s2:2/out':2,
                    c2u:4/outu:4,s2u:4/outu':4,ok:1/online:1] |
             Comm00[c1':3/in:3,s1':3/in':3,c2':2/out:2,s2':2/out':2,
                    c2u':4/outu:4,s2u':4/outu':4,ok':1/online:1]
```

3.6 Verifying the Railway Signaling System

In this section, we specify and verify requirements of the slow-scan, recovery, and fault-tolerant models. We use the well-known modal μ-calculus [109] as our specification language and determine the validity of our properties by model checking [25, 26]. For the verification, we use the `CWB-NC` on a SUN SPARC 20 workstation with 512 MByte of main memory.

3.6.1 State Spaces of the Models

We have run the `CWB-NC` on a SUN SPARC 20 workstation to construct the state spaces of our models. We refer to the slow-scan model as `bruns.ccsch`, to the recovery model as `recovery.ccsch`, and to the fault-tolerant model as `ftolerant.ccsch`. In addition, we call the slow-scan model where the buffer has been replaced by the full-duplex version

`basic.ccsch`. The CCS models corresponding to `bruns.ccsch` and `basic.ccsch`, which are obtained by leaving out all priority values, are called `bruns.ccs` and `basic.ccs`, respectively. For each model, Table 3.7 provides the number of states and transitions of the corresponding transition systems which have been automatically constructed by the `CWB-NC` in less than a minute in average. Whereas the number of states decreases by over 70% when using the calculus with priority, the reduction of transitions by approximately 85% is even more impressive. These large reductions result from the fact that we are not able to model the *atomicity of action sequences* and *interrupts* in plain CCS [125], thereby demonstrating the utility of priority for the verification of concurrent systems.

Table 3.7: Transition system sizes

Model Name	states	trans.
`bruns.ccs`	3527	17122
`bruns.ccsch`	899	2567
`basic.ccs`	1114	4721
`basic.ccsch`	312	801
`recovery.ccsch`	1100	2801
`ftolerant.ccsch`	11905	33760

3.6.2 Properties of Interest

Since the slow-scan system is embedded in a safety-critical system, we want to verify that our designs satisfy the following required properties.

- After a low-grade link fails, either the slow-scan system will eventually detect the error or the link is repaired. Moreover, this property holds after every re-initialization of the system.
- The slow-scan system is always capable of continuing to tick. If this property holds, then the system is *deadlock-free*, too.
- A failure of the slow-scan system is possible. This property should also be valid after every re-initialization of the system.
- A failure is detected only if a failure has actually occurred. Also this property should hold after every re-initialization of the system.
- After a low-grade link fails, the slow-scan system will eventually stop responding to the `SSI` and `TFM`s if the low-grade link does not recover from the error.

- If a failure is detected and the broken line is repaired, then the system will be re-initialized.

All properties – except for the last one – are adapted from [39]. However, since the recovery and the fault-tolerant model are able to recover from an error, the properties of [39] should also hold after every re-initialization of these models.

3.6.3 Specifying the Properties

For specifying our requirements we use a temporal logic, the *modal μ-calculus* [109]. Its syntax is defined by the following BNF, which uses a set of variables $\mathcal{V}$ with $X \in \mathcal{V}$.

$$\Phi \quad ::= \quad tt \quad | \quad X \quad | \quad \neg\Phi \quad | \quad \Phi \wedge \Phi \quad | \quad \langle \alpha\!:\!k \rangle \Phi \quad | \quad \mu X.\Phi$$

Formulas are also required to satisfy the following additional constraint: in $\mu X.\Phi$, every occurrence of X in Φ must be inside an even number of negations. We also define the following dual operators: $ff =_{\mathrm{df}} \neg tt$, $\Phi_1 \vee \Phi_2 =_{\mathrm{df}} \neg(\neg\Phi_1 \wedge \neg\Phi_2)$, $[\alpha\!:\!k]\Phi =_{\mathrm{df}} \neg\langle \alpha\!:\!k \rangle(\neg\Phi)$, and $\nu X.\Phi =_{\mathrm{df}} \neg\mu X.(\neg\Phi[\neg X/X])$, where $[\neg X/X]$ denotes the substitution of all free occurrences of X by $\neg X$. Moreover, we introduce the following abbreviations, where $L \subseteq \mathcal{A}$: $\langle L \rangle \Phi =_{\mathrm{df}} \bigvee\{\langle \alpha\!:\!k \rangle \Phi \mid \alpha\!:\!k \in L\}$, $\langle - \rangle \Phi =_{\mathrm{df}} \langle \mathcal{A} \rangle \Phi$, $\langle -L \rangle \Phi =_{\mathrm{df}} \langle \mathcal{A} \setminus L \rangle \Phi$, $[L]^{\infty}\Phi =_{\mathrm{df}} \nu X.(\Phi \wedge [L]X)$, and $\langle L \rangle^{*}\Phi =_{\mathrm{df}} \mu X.(\Phi \vee \langle L \rangle X)$. Finally, we let $\mathcal{F}$ denote the set of all μ-calculus formulas.

The semantics $\{\!|\Phi|\!\}$ of a μ-calculus formula Φ is defined with respect to an environment $\sigma : \mathcal{V} \longrightarrow 2^{\mathcal{P}}$, that maps variables to sets of processes. Intuitively, $\{\!|\Phi|\!\}(\sigma)$ is the set of all processes that satisfy Φ under the environment σ. Formally, the semantic mapping $\{\!|\cdot|\!\} : (\mathcal{F} \times \mathcal{E}) \longrightarrow 2^{S}$, where $\mathcal{E}$ denotes the set of all environments, is inductively defined over the structure of formulas as shown in Table 3.8. If Φ is a closed formula, its semantics is independent of the environment. In this case, we simply write $\{\!|\Phi|\!\}$ instead of $\{\!|\Phi|\!\}(\sigma)$. We say that the process P satisfies property Φ if $P \in \{\!|\Phi|\!\}$.

Intuitively, the formula tt is satisfied by every process. The Boolean operators are interpreted as usual. The formula $\langle \alpha\!:\!k \rangle \Phi$ is satisfied by those processes that have an $\alpha\!:\!k$–successor for which Φ holds. Finally, $\mu X.\Phi$ stands for the least solution of the equation $X = \Phi$. On the basis of these intuitions, one can deduce that a process $P \in \mathcal{P}$ satisfies $[\alpha\!:\!k]\Phi$ if all its $\alpha\!:\!k$-derivatives satisfy Φ, and it satisfies $[L]^{\infty}\Phi$ if along every process reachable from P via a sequence of transitions labeled with actions in L, the formula Φ holds. Similarly, $\langle L \rangle^{*}\Phi$ holds for a process if some sequence of transitions with labels drawn from L leads to a process satisfying Φ.

We now formally specify the requirements of the slow-scan system as presented above in the modal μ-calculus. We take particular care in implementing *eventuality* since we want to consider only execution paths in which the clock continues to tick. Indeed, our models contain execution paths where the clock stops to tick (cf. *livelocks*). Those paths are called *unfair* since they are artificial and do not occur in practice [81]. Therefore, we filter them out using a notion of *fair eventuality* [39]:

$$even(\Phi) =_{\mathrm{df}} \mu X.\,(\nu Y.\,(\Phi \vee ([\overline{\texttt{tick}}]X \wedge [-\overline{\texttt{tick}}]Y))) \ .$$

Table 3.8: Semantics of the modal μ-calculus

$\{\!\![tt]\!\!\}(\sigma)$	$=_{\mathrm{df}}$	$\mathcal{P}$
$\{\!\![\neg\Phi]\!\!\}(\sigma)$	$=_{\mathrm{df}}$	$\mathcal{P} \setminus \{\!\![\Phi]\!\!\}(\sigma)$
$\{\!\![\Phi_1 \wedge \Phi_2]\!\!\}(\sigma)$	$=_{\mathrm{df}}$	$\{\!\![\Phi_1]\!\!\}(\sigma) \cap \{\!\![\Phi_2]\!\!\}(\sigma)$
$\{\!\![\langle\alpha{:}k\rangle\Phi]\!\!\}(\sigma)$	$=_{\mathrm{df}}$	$\{P \in \mathcal{P} \mid \exists P' \in \mathcal{P}.\, P \xrightarrow{\alpha:k} P' \text{ and } P' \in \{\!\![\Phi]\!\!\}(\sigma)\}$
$\{\!\![\mu X.\Phi]\!\!\}(\sigma)$	$=_{\mathrm{df}}$	$\bigcap\{\mathcal{P}' \subseteq \mathcal{P} \mid \{\!\![\Phi]\!\!\}(\sigma[\mathcal{P}'/X]) \subseteq \mathcal{P}'\}$

Note that the variables X and Y are mutually nested in alternating fixed points. Therefore, the *alternation depth* of any formula $even(\Phi)$ is two if Φ is a closed formula with alternation depth not larger than two. Moreover, we need a meta-formula which expresses that the argument formula holds again if the low-grade system has recovered:

$$again(\Phi) =_{\mathrm{df}} [-]^{\infty}[\overline{\texttt{recovered}}]\Phi \ .$$

Now, we can formalize the desired properties of the slow-scan model and the fault-tolerant model, where $\texttt{Fail} =_{\mathrm{df}} \{\overline{\texttt{fail\_wire}}, \overline{\texttt{fail\_overfull}}\}$.

- After the low-grade link fails (for the first time), the slow-scan system will eventually detect the error or the link is repaired:

  $$\textit{failures-responded} =_{\mathrm{df}} [-\texttt{Fail}]^{\infty}[\texttt{Fail}]even(\langle\overline{\texttt{det}}\rangle tt \vee \langle\overline{\texttt{repaired}}\rangle tt) \ .$$

  The formula *failures-responded* holds after every re-initialization of the system again:

  $$\textit{failures-responded-again} =_{\mathrm{df}} again(\textit{failures-responded}) \ .$$

  Since the formula *failures-responded* is trivially true if the underlying model cannot perform the action `'tick` or if it cannot fail, we are also interested in the next two properties.

- The slow-scan model is always capable of continuing to tick:

  $$\textit{can-tick} =_{\mathrm{df}} [-]^{\infty}\langle-\rangle^{*}\langle\overline{\text{tick}}\rangle tt \ .$$

- A failure of the slow-scan system is possible:

$$\mathit{failures\text{-}possible} =_{\mathrm{df}} \langle -\overline{\texttt{recovered}}\rangle^{*}\langle \texttt{Fail}\rangle tt \ .$$

  The formula *failures-possible* holds after every re-initialization of the system again:

$$\mathit{failures\text{-}possible\text{-}again} =_{\mathrm{df}} \mathit{again}(\mathit{failures\text{-}possible}) \ .$$

- A failure is detected only if it has actually occurred since the last re-initialization of the system:

$$\mathit{no\text{-}false\text{-}alarms} =_{\mathrm{df}} [-\texttt{Fail}, \overline{\texttt{recovered}}]^{\infty}([\overline{\texttt{det}}]ff \vee \langle \overline{\texttt{fail\_overfull}}\rangle tt) \ .$$

  The body of the formula reflects that $\overline{\texttt{fail\_overfull}}$ indicates the occurrence of a failure, i.e. it may be enabled at the same time as $\overline{\texttt{det}}$. Moreover, the formula *no-false-alarms* should hold after every re-initialization of the system:

$$\mathit{no\text{-}false\text{-}alarms\text{-}again} =_{\mathrm{df}} \mathit{again}(\mathit{no\text{-}false\text{-}alarms}) \ .$$

- The auxiliary property "the system never responds," which is used below, can be encoded as follows:

$$\mathit{silent} =_{\mathrm{df}} [-]^{\infty}[\overline{\texttt{comm\_out}}, \overline{\texttt{stat\_out}}]ff \ .$$

  After a low-grade link fails, the slow-scan system will eventually be silent if the low-grade link does not recover from the error:

$$\mathit{eventually\text{-}silent} =_{\mathrm{df}} [-]^{\infty}[\overline{\texttt{det}}]\mathit{even}(\texttt{silent} \vee \langle \overline{\texttt{recovered}}\rangle tt) \ .$$

- If a failure is detected and the broken line is repaired, then the system will be re-initialized:

$$\mathit{react\text{-}on\text{-}repair} =_{\mathrm{df}} [-]^{\infty}[\overline{\texttt{det}}]([-\overline{\texttt{recovered}}]^{\infty}[\overline{\texttt{repaired}}]\mathit{even}(\langle \overline{\texttt{recovered}}\rangle tt)) \ .$$

3.6.4 Verification Results

The model checker implemented in the `CWB-NC` is based on [25, 26]; it is a *local* model checker for a fragment of the modal μ-calculus. The formulas we intend to verify can be rewritten into semantical equivalent ones which satisfy the syntactic restriction required in [26]. The time and space complexity of the model checker is linear in the size of the formula, in its alternation depth, and in the size of the considered transition system.

In contrast to [39], we were able to verify all properties automatically and without using any abstractions by hand. Each formula took no longer than ten minutes to check.

Using a *local* model checker often gains no verification speed-up. This is because most of the formulas are valid safety properties, and the local model checker has to investigate all states of the models anyway. However, our model checker has quickly detected invalid formulas.

In the prioritized models all properties hold as expected. The formula *no-false-alarms* does not hold for the models in plain CCS. This is due to the fact that the atomicity of actions cannot be expressed without priorities. Indeed, there exist interleavings in the CCS models where one observes a `'det` before a failure has occurred. Surprisingly, we have found the formula *failures-responded* invalid in the model `bruns.ccs` whereas in [39] it is reported to hold. The reason for this is that we have left out the actions `c1'` (`s1'`) which occur directly before a `'det` in Bruns' model. Although that reflects our intuition that a `'det` should be signaled as soon as an error is detected, Bruns' modeling does not allow both `SPC` and `TPC` to detect the overfull-failure of the medium before the action `'fail_overfull` has occurred.

3.7 Summary

We have demonstrated the importance of priority for modeling and verifying concurrent systems by means of a practically relevant case study of the slow-scan part of a railway signaling system. Priorities allow us to favor one communication over another and to make action sequences atomic. While the former helps to model systems more realistically, the latter drastically cuts the number of states and transitions. Our models explicitly reflect safety-critical parts of the slow-scan system, namely an error-recovery scheme and a fault-tolerant medium, which are required in the design document [69]. Finally, we have used the `CWB-NC` for checking several required properties of our design.

Chapter 4

Dynamic-Priority for Modeling Real-Time

4.1 Introduction

This chapter addresses the problem of modeling and verifying concurrent systems where *real-time* plays an important role for their functional behavior. On the one hand, real-time is often used to implement abstract *synchronization constraints* in distributed environments. As an example of a synchronization constraint, consider a communication protocol where the next protocol phase may be entered only if some or all components agree. On the other hand, electric phenomena in digital circuits, e.g. *signal glitches*, that may lead to malfunction, can be avoided using *deskew delays*. Thus, for accurately modeling such systems in process algebras it is necessary to capture their real-time aspects, thereby motivating the need for implementing real-time process algebras [128, 173] efficiently.

Traditional implementations of real-time process algebras typically cause state spaces to explode, the reason for this being that time is considered as part of the state, i.e. a new state is generated for every clock tick. We tackle this problem by using *dynamic-priority* to model real-time. We introduce a new process algebra, called $\mathsf{CCS}^{\mathsf{dp}}$ (CCS with dynamic-priority), which essentially extends CCS [125] by assigning priorities to actions. Unlike the process algebra $\mathsf{CCS}^{\mathsf{ch}}$ presented in Chapter 2 and – up to our knowledge – all other existing process-algebraic approaches to priority, actions in our algebra do not have fixed or *static* priorities; but priority values may change as systems evolve. It is in this sense that we refer to $\mathsf{CCS}^{\mathsf{dp}}$ as a process algebra with *dynamic*-priority. In contrast to traditional real-time algebras, e.g. a variation of Temporal CCS [128] incorporating the well-known *maximal progress assumption*, which we call $\mathsf{CCS}^{\mathsf{rt}}$ (CCS with real-time), the $\mathsf{CCS}^{\mathsf{dp}}$ semantics interprets delays preceding actions as priority values attached to these actions. In other words, the longer the delay preceding an action, the lower is its priority. The semantics of $\mathsf{CCS}^{\mathsf{dp}}$ avoids the unfolding of delay values into sequences of elementary steps, each consuming one time unit, thereby providing a formal foundation for *efficiently* modeling real-time. The soundness and completeness of our approach is proved by establishing a one-to-one

correspondence between the $\mathsf{CCS}^{\mathsf{dp}}$ and the $\mathsf{CCS}^{\mathsf{rt}}$ semantics in terms of *bisimulation* [125, 139] and *temporal logics* [109, 160]. Hereby, we relate the concept of maximal progress in $\mathsf{CCS}^{\mathsf{rt}}$ to the concept of *pre-emption* in $\mathsf{CCS}^{\mathsf{dp}}$, the latter of which is adopted from $\mathsf{CCS}^{\mathsf{ch}}$. It is important to note that our approach does not abstract away any real-time information. Thus, all quantitative timing constraints explicit in $\mathsf{CCS}^{\mathsf{rt}}$ semantics can still be analyzed within $\mathsf{CCS}^{\mathsf{dp}}$ semantics. In Chapter 5 the utility of our technique is shown by a case study, modeling and verifying several aspects of the `SCSI-2` bus-protocol.

The remainder of this chapter is organized as follows. The next section presents our process-algebraic framework for developing the temporal process algebra $\mathsf{CCS}^{\mathsf{rt}}$ and the process algebra $\mathsf{CCS}^{\mathsf{dp}}$ with dynamic-priority. Section 4.3 formally defines the semantics of $\mathsf{CCS}^{\mathsf{rt}}$ whereas Section 4.4 describes the semantics of $\mathsf{CCS}^{\mathsf{dp}}$. A one-to-one relationship between $\mathsf{CCS}^{\mathsf{dp}}$ and $\mathsf{CCS}^{\mathsf{rt}}$ semantics is established in Section 4.5. The following section discusses the utility of our approach for system modeling and verification and compares it to related work. Finally, Section 4.7 contains a summary of this chapter.

4.2 Process-Algebraic Framework

In this section we introduce the process-algebraic framework for $\mathsf{CCS}^{\mathsf{rt}}$ and $\mathsf{CCS}^{\mathsf{dp}}$, both having the same syntax but different semantics. Whereas $\mathsf{CCS}^{\mathsf{rt}}$ is an extension of CCS [125] in order to capture *discrete, quantitative timing aspects* with respect to a single, global clock, $\mathsf{CCS}^{\mathsf{dp}}$ extends CCS by a concept of *dynamic-priority*. The syntax of $\mathsf{CCS}^{\mathsf{rt}}$ and $\mathsf{CCS}^{\mathsf{dp}}$ differs from CCS by associating delay and priority values with actions, respectively, and by including the *disabling* operator known from LOTOS [30]. Formally, let Λ be a countably infinite set of *actions* or *ports*, not containing the so-called *silent* or *internal* action τ. With every $a \in \Lambda$ we associate a *complementary action* $\overline{a}$. Intuitively, an action $a \in \Lambda$ may be thought of as representing the receipt of an input on port a, while $\overline{a}$ constitutes the deposit of an output on a. We define $\overline{\Lambda} =_{\mathrm{df}} \{\overline{a} \mid a \in \Lambda\}$ and take $\mathcal{A}$ to denote the set of all actions $\Lambda \cup \overline{\Lambda} \cup \{\tau\}$. In what follows, we let $a, b, \ldots$ range over $\Lambda \cup \overline{\Lambda}$ and $\alpha, \beta, \ldots$ over $\mathcal{A}$. Complementation is lifted to actions in $\Lambda \cup \overline{\Lambda}$, also called *visible actions*, by defining $\overline{\overline{a}} =_{\mathrm{df}} a$.

In our syntax actions are associated with *delay values*, or *priority values*, taken from the natural numbers. More precisely, the notation $\alpha{:}k$, where $\alpha \in \mathcal{A}$ and $k \in \mathbb{N}$, specifies that action α is ready for execution after a minimum delay of k time units or, respectively, that action α possesses (at most) priority k. In the priority interpretation, smaller numbers encode higher priority values. In contrast to $\mathsf{CCS}^{\mathsf{ch}}$, action labels and priority values are separated in $\mathsf{CCS}^{\mathsf{dp}}$, i.e. priority is not part of an action. The syntax of our language is defined by the following BNF.

$$
\begin{array}{lllllllllllll}
P & ::= & \mathbf{0} & \mid & x & \mid & \alpha{:}k.P & \mid & P + P & \mid & P \between P & \mid \\
 & & P \mid P & \mid & P[f] & \mid & P \setminus L & \mid & \mu x.P & & & &
\end{array}
$$

where $k \in \mathbb{N}$, the mapping $f : \mathcal{A} \to \mathcal{A}$ is a *relabeling*, $L \subseteq \mathcal{A} \setminus \{\tau\}$ is a *restriction set*, and x is a *variable* taken from some countable domain $\mathcal{V}$. A relabeling f satisfies the properties

$f(\tau) = \tau$ and $f(\overline{a}) = \overline{f(a)}$. If $f(\alpha_i) = \beta_i$ for $1 \leq i \leq n$ and $n \in \mathbb{N}$, and $f(\alpha) = \alpha$ for all $\alpha \neq \alpha_i$, where $1 \leq i \leq n$, we also write $[\beta_1/\alpha_1, \beta_2/\alpha_2, \ldots, \beta_n/\alpha_n]$ for f. Moreover, we adopt the usual definitions for *free* and *bound* variables, *open* and *closed* terms, and *guarded* recursion, and refer to the closed guarded *terms* as *processes*. The syntactic substitution of all free occurrences of variable x by Q in term P is symbolized by $P[Q/x]$. Finally, we let $\mathcal{P}$, ranged over by $P, Q, R, \ldots$, denote the set of all processes.

4.3 Real-Time Semantics

We first introduce a real-time semantics for our language, referred to as $\mathsf{CCS}^{\mathsf{rt}}$ semantics, which explicitly represents timing behavior. The semantics of a process is defined by a *labeled transition system* which contains explicit *clock transitions* – each representing a delay of one time unit – as well as *action transitions*. With respect to clock transitions, the operational semantics is set up such that processes willing to communicate with some process running in parallel are able to wait until the communication partner is ready. However, as soon as it is available the communication has to take place, i.e. further idling is prohibited. This assumption is usually referred to as *maximal progress assumption* [173] or *synchrony hypothesis* [21].

Formally, the labeled transition system corresponding to a process $P \in \mathcal{P}$ is a fourtuple $\langle \mathcal{P}, \mathcal{A} \cup \{1\}, \longmapsto, P \rangle$ where $\mathcal{P}$ is the set of states, $\mathcal{A} \cup \{1\}$ the alphabet satisfying $1 \notin \mathcal{A}$, $\longmapsto$ the transition relation, and P represents the start state. The transition relation $\longmapsto \subseteq \mathcal{P} \times (\mathcal{A} \cup \{1\}) \times \mathcal{P}$ is defined in Tables 4.1 and 4.2 using operational rules in Plotkin-style notation [141]. For the sake of simplicity, let us use γ as representative of $\mathcal{A} \cup \{1\}$, and write $P \stackrel{\gamma}{\longmapsto} P'$ instead of $\langle P, \gamma, P' \rangle \in \longmapsto$. We say that P *may engage in transition* γ *and thereafter behave like process* P'. If $\gamma \in \mathcal{A}$ we speak of an *action transition*, otherwise of a *clock transition*. Sometimes it is convenient to write $P \stackrel{\gamma}{\longmapsto}$ for $\exists P' \in \mathcal{P}.\, P \stackrel{\gamma}{\longmapsto} P'$. In order to ensure maximal progress our operational semantics is set up in a way such that $P \stackrel{1}{\not\longmapsto}$ whenever $P \stackrel{\tau}{\longmapsto}$, i.e. clock transitions are prevented as long as P can engage in internal computation.

Intuitively, the process $\alpha{:}k.P$, where $k > 0$ may engage in a clock transition and then behave like $\alpha{:}(k-1).P$. The process $\alpha{:}0.P$ performs an α transition to become process P. Moreover, if $\alpha \neq \tau$, it may also idle by executing a clock transition to itself. The summation operator $+$ denotes *nondeterministic* choice, i.e. the process $P + Q$ may either behave like P or Q with respect to action transitions. Time has to proceed equally on both sides of summation, i.e. $P+Q$ can engage in a clock transition and, thus, delay the nondeterministic choice if and only if both P and Q can engage in a clock transition. The process $P \rangle\!\rangle Q$, involving the *disabling* operator $\rangle\!\rangle$, has the same semantics for clock transitions. For action transitions it behaves as P and, additionally, it is capable of disabling P by engaging in Q. The *restriction* operator $\backslash L$ prohibits the execution of actions in $L \cup \overline{L}$ and thus permits the scoping of actions. $P[f]$ behaves exactly like P where actions are renamed by the *relabeling* f. The process $P|Q$ stands for the *parallel composition* of P and Q according

Table 4.1: Operational semantics for $\mathsf{CCS}^{\mathsf{rt}}$ (Part I)

Act	$\dfrac{-}{\alpha{:}0.P \stackrel{\alpha}{\longmapsto} P}$	tNil	$\dfrac{-}{\mathbf{0} \stackrel{1}{\longmapsto} \mathbf{0}}$
Sum1	$\dfrac{P \stackrel{\alpha}{\longmapsto} P'}{P+Q \stackrel{\alpha}{\longmapsto} P'}$	tAct1	$\dfrac{-}{\alpha{:}k.P \stackrel{1}{\longmapsto} \alpha{:}(k-1).P}\ k>0$
Sum2	$\dfrac{Q \stackrel{\alpha}{\longmapsto} Q'}{P+Q \stackrel{\alpha}{\longmapsto} Q'}$	tAct2	$\dfrac{-}{a{:}0.P \stackrel{1}{\longmapsto} a{:}0.P}$
Dis1	$\dfrac{P \stackrel{\alpha}{\longmapsto} P'}{P \rangle\!\rangle Q \stackrel{\alpha}{\longmapsto} P' \rangle\!\rangle Q}$	tSum	$\dfrac{P \stackrel{1}{\longmapsto} P' \quad Q \stackrel{1}{\longmapsto} Q'}{P+Q \stackrel{1}{\longmapsto} P'+Q'}$
Dis2	$\dfrac{Q \stackrel{\alpha}{\longmapsto} Q'}{P \rangle\!\rangle Q \stackrel{\alpha}{\longmapsto} Q'}$	tDis	$\dfrac{P \stackrel{1}{\longmapsto} P' \quad Q \stackrel{1}{\longmapsto} Q'}{P \rangle\!\rangle Q \stackrel{1}{\longmapsto} P' \rangle\!\rangle Q'}$
Rel	$\dfrac{P \stackrel{\alpha}{\longmapsto} P'}{P[f] \stackrel{f(\alpha)}{\longmapsto} P'[f]}$	tRel	$\dfrac{P \stackrel{1}{\longmapsto} P'}{P[f] \stackrel{1}{\longmapsto} P'[f]}$
Res	$\dfrac{P \stackrel{\alpha}{\longmapsto} P'}{P \setminus L \stackrel{\alpha}{\longmapsto} P' \setminus L}\ \alpha \notin L \cup \overline{L}$	tRes	$\dfrac{P \stackrel{1}{\longmapsto} P'}{P \setminus L \stackrel{1}{\longmapsto} P' \setminus L}$
Rec	$\dfrac{P[\mu x.P/x] \stackrel{\alpha}{\longmapsto} P'}{\mu x.P \stackrel{\alpha}{\longmapsto} P'}$	tRec	$\dfrac{P[\mu x.P/x] \stackrel{1}{\longmapsto} P'}{\mu x.P \stackrel{1}{\longmapsto} P'}$

Table 4.2: Operational semantics for $\mathsf{CCS}^{\mathsf{rt}}$ (Part II)

Com1	$\dfrac{P \stackrel{\alpha}{\longmapsto} P'}{P \mid Q \stackrel{\alpha}{\longmapsto} P' \mid Q}$	Com3	$\dfrac{P \stackrel{a}{\longmapsto} P' \quad Q \stackrel{\overline{a}}{\longmapsto} Q'}{P \mid Q \stackrel{\tau}{\longmapsto} P' \mid Q'}$
Com2	$\dfrac{Q \stackrel{\alpha}{\longmapsto} Q'}{P \mid Q \stackrel{\alpha}{\longmapsto} P \mid Q'}$	tCom	$\dfrac{P \stackrel{1}{\longmapsto} P' \quad Q \stackrel{1}{\longmapsto} Q'}{P \mid Q \stackrel{1}{\longmapsto} P' \mid Q'}\ P \mid Q \stackrel{\tau}{\not\longmapsto}$

to an interleaving semantics with synchronized communication on complementary actions resulting in the internal action τ. Similar to summation and disabling, P and Q have to synchronize on clock transitions according to Rule (tCom). Its side condition implements maximal progress by ensuring that there is no pending communication between P and Q. Although this condition is negative, our semantics is still well-defined [29]. Finally, $\mu x.P$ denotes a *recursively defined* process that is a distinguished solution of the equation $x = P$. The semantics satisfies the following properties [128], where $\equiv$ denotes syntactic equality.

Proposition 4.3.1 *For all $P, P', P'' \in \mathcal{P}$ the following holds.*

1. *$P \stackrel{\tau}{\not\longmapsto}$ implies $P \stackrel{1}{\longmapsto}$ (*idling*).*
2. *$P \stackrel{\tau}{\longmapsto}$ implies $P \stackrel{1}{\not\longmapsto}$ (*maximal progress*).*
3. *$P \stackrel{1}{\longmapsto} P'$ and $P \stackrel{1}{\longmapsto} P''$ implies $P' \equiv P''$ (*time determinacy*).*

The validity of Part (1) is a consequence of the idling capability of the processes $\mathbf{0}$ and $\alpha{:}k.P$ for $k > 0$ or $\alpha \not\equiv \tau$. Properties (2) and (3) can easily be checked by induction on the structure of P and induction on the maximum of the depths of the derivation trees of $P \stackrel{1}{\longmapsto} P'$ and $P \stackrel{1}{\longmapsto} P''$, respectively. For $\mathsf{CCS}^{\mathsf{rt}}$ a semantic theory based on the notion of *bisimulation* [125] has been developed [128]. For the purposes of this chapter, we restrict ourselves to *strong* temporal bisimulation which is defined as follows.

Definition 4.3.2 (Temporal Bisimulation)
A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is called temporal bisimulation *if for every $\langle P, Q\rangle \in \mathcal{R}$ and $\gamma \in \mathcal{A} \cup \{1\}$ the following holds: $P \stackrel{\gamma}{\longmapsto} P'$ implies $\exists Q'.\, Q \stackrel{\gamma}{\longmapsto} Q'$ and $\langle P', Q'\rangle \in \mathcal{R}$. We write $P \sim_{rt} Q$ if there exists a temporal bisimulation $\mathcal{R}$ such that $\langle P, Q\rangle \in \mathcal{R}$.*

The behavioral relation $\sim_{\mathtt{rt}}$, which can be shown to be the largest temporal bisimulation and an equivalence relation, enjoys several pleasant properties, the most important being the *congruence* property, which gives rise to *compositional reasoning* (cf. [76]). Note that temporal bisimulation requires equivalent processes to match exactly each others behavior, including their timing behavior. In the literature preorders have been developed which relax this requirement by introducing "*faster-than*" relations [129].

Unfortunately, $\mathsf{CCS}^{\mathsf{rt}}$ semantics unfolds every delay value into a sequence of elementary time units, thereby creating many additional states. For example, the process $a{:}k.\mathbf{0}$ has $k + 2$ states, namely $\mathbf{0}$ and $a{:}l.\mathbf{0}$ where $0 \leq l \leq k$ (see also Figure 4.1 on Page 65). It would be much more efficient if we could represent $a{:}k.\mathbf{0}$ by a single transition labeled by $a{:}k$ leading to state $\mathbf{0}$. This idea of compacting the state space of real-time systems can be implemented by viewing k as a *priority value* assigned to action a. In other words, one may consider the delay value k as the *time-stamp* of action a. The following sections elaborate on this idea.

4.4 Dynamic-Priority Semantics

In order to formalize our intuition, we present a new semantics for our language that uses a notion of priority taken from [54], generalized to a *multi-level* priority-scheme. We refer to our process algebra as $\mathsf{CCS}^{\mathsf{dp}}$ when interpreted with respect to the new semantics which, in contrast to the priority approach mentioned above, *dynamically* adjusts priorities along transitions. However, the notion of pre-emption incorporated in $\mathsf{CCS}^{\mathsf{dp}}$ follows the lines in Chapter 2. Note that this notion of pre-emption naturally mimics the maximal progress assumption employed in $\mathsf{CCS}^{\mathsf{rt}}$ semantics which is made precise in the next section. Formally, the $\mathsf{CCS}^{\mathsf{dp}}$ semantics of a process $P \in \mathcal{P}$ is given by a labeled transition system $\langle \mathcal{P}, \mathcal{A} \times \mathbb{N}, \longrightarrow, P \rangle$. The presentation of the operational rules for the transition relation $\longrightarrow$ requires two auxiliary definitions.

First, we introduce *potential initial action sets*, taking account of the actions and their priority values in which a given process can initially engage in. These sets are defined similar as for $\mathsf{CCS}^{\mathsf{ch}}$ (cf. Table 2.1) but where the equation concerning relabeling is replaced by $\mathrm{I}^k(P[f]) =_{\mathrm{df}} \{f(\alpha){:}l \mid \alpha{:}l \in \mathrm{I}^k(P)\}$, since relabelings in $\mathsf{CCS}^{\mathsf{dp}}$ ignore priority values. Finally, we adopt the definitions of (real) initial action sets $\mathcal{I}^k(P)$ of a process P and the notations $\mathrm{I}^{<k}(P)$, $\mathcal{I}^{<k}(P)$, $\alpha \in \mathrm{I}^{<k}(P)$, $\alpha \in \mathcal{I}^{<k}(P)$, $\mathcal{I\!I}^k(P)$, and $\mathcal{I\!I}^{<k}(P)$. Hence, Proposition 2.2.1 and all remarks made in Chapter 2 concerning initial action sets are also valid within the $\mathsf{CCS}^{\mathsf{dp}}$ framework.

Table 4.3: Priority adjustment function

$[\mathbf{0}]^k =_{\mathrm{df}} \mathbf{0}$	$[x]^k =_{\mathrm{df}} x$
$[\alpha{:}l.P]^k =_{\mathrm{df}} \alpha{:}(l-k).P$ if $l > k$	$[\alpha{:}l.P]^k =_{\mathrm{df}} \alpha{:}0.P$ if $l \leq k$
$[P + Q]^k =_{\mathrm{df}} [P]^k + [Q]^k$	$[P \rangle\!\rangle Q]^k =_{\mathrm{df}} [P]^k \rangle\!\rangle [Q]^k$
$[P \mid Q]^k =_{\mathrm{df}} [P]^k \mid [Q]^k$	$[\mu x.P]^k =_{\mathrm{df}} [P[\mu x.P/x]]^k$
$[P[f]]^k =_{\mathrm{df}} [P]^k[f]$	$[P \setminus L]^k =_{\mathrm{df}} [P]^k \setminus L$

As second auxiliary definition for presenting the transition relation, we introduce a *priority adjustment function* as shown in Table 4.3. Intuitively, our semantics is set up in a way such that if one parallel component of a process engages in a transition with priority k, then the priority values of all initial actions at every other parallel component have to be decreased by k, i.e. those actions become equally "more important." Thus, the semantics of

parallel composition deploys a kind of *fairness assumption*, and priorities have a *dynamic* character. More precisely, the priority adjustment function applied to a process $P \in \mathcal{P}$ and a natural number $k \in \mathbb{N}$, denoted as $[P]^k$, returns a process term which is "identical" to P except that the priority values of the initial, top-level actions are decreased by k. Note that a priority value cannot become less than 0. The phrase "identical" does not mean syntactic equality but syntactic equality *up to unfolding* of recursion. Formally, let $\doteq$ stand for the smallest congruence which contains $\equiv$ and satisfies the axiom $\mu x.P \doteq P[\mu x.P/x]$. Our semantics is compatible with $\doteq$ in the sense that $P \doteq Q$ and $P \xrightarrow{\alpha:k} P'$ implies $Q \xrightarrow{\alpha:k} Q'$ for some $Q' \in \mathcal{P}$ satisfying $P' \doteq Q'$. This compatibility is used in the remainder of this chapter without explicit mentioning. Sometimes we also write $P \xrightarrow{\alpha:k} P'$ if $Q \xrightarrow{\alpha:k} Q'$ for some $Q \doteq P$ and $Q' \doteq P'$.

Table 4.4: Operational semantics for $\mathsf{CCS}^{\mathsf{dp}}$

Act1	$\dfrac{-}{a{:}k.P \xrightarrow{a:l} P}\ l \geq k$	Act2	$\dfrac{-}{\tau{:}k.P \xrightarrow{\tau:k} P}$
Sum1	$\dfrac{P \xrightarrow{\alpha:k} P'}{P+Q \xrightarrow{\alpha:k} P'}\ \tau \notin \mathrm{I}^{<k}(Q)$	Sum2	$\dfrac{Q \xrightarrow{\alpha:k} Q'}{P+Q \xrightarrow{\alpha:k} Q'}\ \tau \notin \mathrm{I}^{<k}(P)$
Dis1	$\dfrac{P \xrightarrow{\alpha:k} P'}{P \between Q \xrightarrow{\alpha:k} P' \between [Q]^k}\ \tau \notin \mathrm{I}^{<k}(Q)$	Dis2	$\dfrac{Q \xrightarrow{\alpha:k} Q'}{P \between Q \xrightarrow{\alpha:k} Q'}\ \tau \notin \mathrm{I}^{<k}(P)$
Com1	$\dfrac{P \xrightarrow{\alpha:k} P'}{P\|Q \xrightarrow{\alpha:k} P'\|[Q]^k}\ \tau \notin \mathrm{I}^{<k}(P\|Q)$	Rel	$\dfrac{P \xrightarrow{\alpha:k} P'}{P[f] \xrightarrow{f(\alpha):k} P'[f]}$
Com2	$\dfrac{Q \xrightarrow{\alpha:k} Q'}{P\|Q \xrightarrow{\alpha:k} [P]^k\|Q'}\ \tau \notin \mathrm{I}^{<k}(P\|Q)$	Res	$\dfrac{P \xrightarrow{\alpha:k} P'}{P \setminus L \xrightarrow{\alpha:k} P' \setminus L}\ \alpha \notin L \cup \overline{L}$
Com3	$\dfrac{P \xrightarrow{a:k} P' \quad Q \xrightarrow{\overline{a}:k} Q'}{P\|Q \xrightarrow{\tau:k} P'\|Q'}\ \tau \notin \mathrm{I}^{<k}(P\|Q)$	Rec	$\dfrac{P[\mu x.P/x] \xrightarrow{\alpha:k} P'}{\mu x.P \xrightarrow{\alpha:k} P'}$

The operational rules in Table 4.4 capture the following intuition. The process $a{:}k.P$ may engage in action a with priority value $l \geq k$ yielding process P. The side condition $l \geq k$ reflects that k does not specify an exact priority but the *maximal* priority of the initial transition of $a{:}k.P$. It may also be interpreted as *lower-bound* "timing constraint." Due to the notion of pre-emption incorporated in $\mathsf{CCS}^{\mathsf{dp}}$, $\tau{:}k.P$ may not perform the initial

τ-transition with a lower priority than k. The process $P+Q$ may behave like P (Q) if Q (P) does not pre-empt the considered transition by being able to engage in a higher prioritized internal transition. Thus, our notion of pre-emption reflects implicit *upper-bound* "timing constraints." The process $P \mid Q$ denotes the *parallel composition* of P and Q according to an interleaving semantics with synchronized communication on complementary actions of P and Q having the same priority value k which results in the internal action τ attached with priority value k (cf. Rule (Com3)). The interleaving Rules (Com1) and (Com2) incorporate the dynamic behavior of priority values as explained in the previous paragraph. Their side conditions implement pre-emption. The operational semantics for *disabling*, *restriction*, *relabeling*, and *recursion* is straightforward and, thus, does not require extra explanation. The following proposition shows that the notion of pre-emption encoded in the side conditions of the operational rules coincides with the one of classic approaches to priority as presented in Chapter 2. It can be proved by induction on the structure of processes.

Proposition 4.4.1 *Let $P \in \mathcal{P}$, $\alpha \in \mathcal{A}$, and $k \in \mathbb{N}$ such that $P \xrightarrow{\alpha:k}$. Then $\tau \notin \mathrm{I}^{<k}(P)$.*

As for $\mathsf{CCS}^{\mathsf{rt}}$, we may adapt a notion of strong bisimulation, called *prioritized bisimulation.*

Definition 4.4.2 (Prioritized Bisimulation)
A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is called prioritized bisimulation *if for every $\langle P, Q\rangle \in \mathcal{R}$, $\alpha \in \mathcal{A}$, and $k \in \mathbb{N}$ the following holds: $P \xrightarrow{\alpha:k} P'$ implies $\exists Q'.\, Q \xrightarrow{\alpha:k} Q'$ and $\langle P', Q'\rangle \in \mathcal{R}$. We write $P \sim_{dp} Q$ if there exists a prioritized bisimulation $\mathcal{R}$ such that $\langle P, Q\rangle \in \mathcal{R}$.*

It is easy to see that $\sim_{\mathsf{dp}}$ is the largest prioritized bisimulation and that it contains $\doteq$; a fact that we use silently.

We close this section with some remarks on the integration of our language in automated verification tools. For both process algebras, $\mathsf{CCS}^{\mathsf{dp}}$ and $\mathsf{CCS}^{\mathsf{rt}}$, front-ends for the `CWB-NC` [65, 153] have been created by using `PAC` [62]. Whereas the implementation of $\mathsf{CCS}^{\mathsf{rt}}$ has been straightforward, we have needed some more effort regarding $\mathsf{CCS}^{\mathsf{dp}}$. The reason is that Rule (Act1) of $\mathsf{CCS}^{\mathsf{dp}}$ semantics gives rise to potentially infinite-branching transition systems since priority value l in its side condition ranges over all natural numbers greater or equal than k. However, for practical purposes this problem can be eliminated. One possibility is to provide an upper bound `upper` reflecting the maximal priority value of any action occurring in the considered process. The validity of this solution stems from the fact that a higher priority value than `upper` has no effect on the semantics since priority values cannot be adjusted to a value below zero. This idea can be refined in the following way. Instead of choosing a value `upper` with respect to the *overall* process, we determine this value with respect to the *particular* system state in which the system under consideration is currently in. Our implementation of the $\mathsf{CCS}^{\mathsf{dp}}$ semantics follows the latter idea. Thus, the number of transitions of a process according to $\mathsf{CCS}^{\mathsf{dp}}$ semantics is always less than or equal to the number of transitions with respect to $\mathsf{CCS}^{\mathsf{rt}}$ semantics. Finally, we want to point

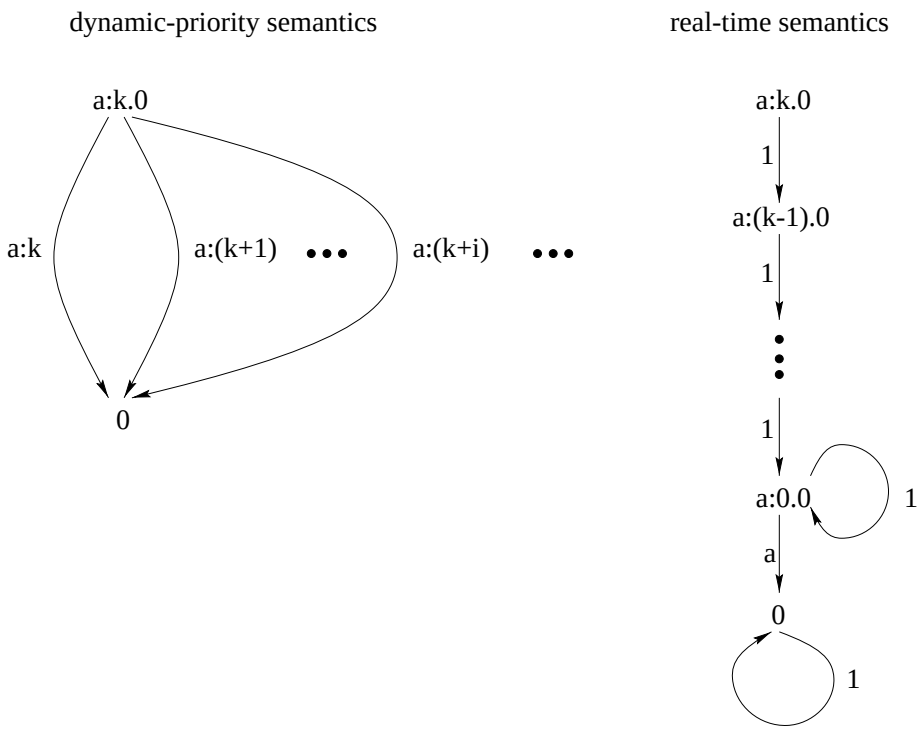

Figure 4.1: Relating CCS^{dp} and CCS^{rt} semantics

out that these solutions affect the compositionality of the implemented CCS^{dp} semantics in the following sense. If a system is combined with another one having a greater upper priority value, additional system behavior is possible. However, already computed parts of the semantics need not to be re-computed.

4.5 Relating CCS^{dp} and CCS^{rt} Semantics

In this section we show that CCS^{dp} and CCS^{rt} semantics are closely related. The underlying intuition is best illustrated by a simple example dealing with the prefixing operator. Figure 4.1 depicts the dynamic-priority and real-time semantics of the process $a{:}k.\mathbf{0}$. Both transition systems intuitively reflect that the process $a{:}k.\mathbf{0}$ must at least delay k time units before it may engage in an a-transition. According to CCS^{rt} semantics this process consecutively engages in k clock transitions passing the states $a{:}(k-i).\mathbf{0}$, for $0 \leq i \leq k$, before it may either continue idling in state $a{:}0.\mathbf{0}$ or perform an a-transition to the inaction process $\mathbf{0}$. Thus, time is explicitly part of states and made visible by clock transitions each representing a step consuming one time unit. In contrast, the dynamic-priority semantics encodes the delay of at least k time units in the transitions rather than in the states. Hence, it possesses only the two states $a{:}k.\mathbf{0}$ and $\mathbf{0}$ connected via transitions labeled by $a{:}l$ for $l \geq k$. Although at first sight it seems that the price for saving intermediate states is to be forced to deal with infinite-branching, an upper bound of l can be given as discussed in

the previous section. In our example this upper bound is k itself, since a delay by more than k time units only results in idling and does not enable new or disable existing system behavior. Therefore, the dynamic-priority transition system of $a{:}k.\mathbf{0}$ would just consist of the two states $a{:}k.\mathbf{0}$ and $\mathbf{0}$ and a *symbolic* transition labeled by $a{:}k$, whereas the real-time transition system has $k+2$ states and $k+2$ transitions. This simple example already suggests that $\mathsf{CCS}^{\mathsf{dp}}$ semantics yields much more compact models than $\mathsf{CCS}^{\mathsf{rt}}$ semantics. The remainder of this section aims at proving a one-to-one correspondence between the two semantics such that $\mathsf{CCS}^{\mathsf{dp}}$ semantics can indeed be understood as an efficient encoding of $\mathsf{CCS}^{\mathsf{rt}}$ semantics. To this end, one also needs to make sure that the notion of pre-emption employed in $\mathsf{CCS}^{\mathsf{dp}}$ corresponds to the notion of maximal progress adopted in $\mathsf{CCS}^{\mathsf{rt}}$.

Before making the relationship between both semantics precise, we first state an important lemma whose last part presents the connection between clock transitions and the priority adjustment function.

Lemma 4.5.1 *For all $P, P' \in \mathcal{P}$ and all $k, l \in \mathbb{N}$ the following holds.*

1. $[P]^0 \doteq P$ *and* $[[P]^l]^k \doteq [P]^{k+l}$.
2. $\mathrm{I}^k([P]^l) = \mathrm{I}^{k+l}(P)$.
3. $P \stackrel{1}{\longmapsto}{}^k P'$ *if and only if* $P' \doteq [P]^k$ *and* $\tau \notin \mathrm{I}^{<k}(P)$.

Proof: Let $P, P' \in \mathcal{P}$ and $k, l \in \mathbb{N}$.

- Part (1) follows immediately from the definitions of the adjustment function and of the congruence $\doteq$.
- Part (2) is proved by induction on the structure of P.
  1. $P \equiv \mathbf{0}$:

     $\mathrm{I}^{<k}([\mathbf{0}]^l) = \mathrm{I}^{<k}(\mathbf{0}) = \emptyset = \mathrm{I}^{<k+l}(\mathbf{0})$ by the definitions of the adjustment function and of potential initial action sets.
  2. $P \equiv \alpha{:}m.Q$:

$$\mathrm{I}^{<k}([\alpha{:}m.Q]^l)$$

$$(\text{def. of } [\cdot]^{\cdot}) \quad = \quad \begin{cases} \mathrm{I}^{<k}(\alpha{:}(m-l).Q) & \text{if } m > l \\ \mathrm{I}^{<k}(\alpha{:}0.Q) & \text{otherwise} \end{cases}$$

$$(\text{def. of } \mathrm{I}^{\cdot}(\cdot)) \quad = \quad \begin{cases} \{\alpha\} & \text{if } k > (m-l) \text{ or } (m \leq l \text{ and } k > 0) \\ \emptyset & \text{otherwise} \end{cases}$$

$$= \begin{cases} \{\alpha\} & \text{if } (k+l) > m \\ \emptyset & \text{otherwise} \end{cases}$$

$$(\text{def. of } \mathrm{I}^{\cdot}(\cdot)) \quad = \quad \mathrm{I}^{<k+l}(\alpha{:}m.Q)$$

3. $P \equiv Q_1|Q_2$*:*

$$\mathrm{I}^{<k}([Q_1|Q_2]^l)$$

$$(\text{def. of } [\cdot]^{\cdot}) \quad = \quad \mathrm{I}^{<k}([Q_1]^l|[Q_2]^l)$$

$$(\text{def. of } \mathrm{I}^{\cdot}(\cdot)) \quad = \quad \mathrm{I}^{<k}([Q_1]^l) \cup \mathrm{I}^{<k}([Q_2]^l) \cup \{\tau \mid \mathrm{I}^{<k}([Q_1]^l) \cap \overline{\mathrm{I}^{<k}([Q_2]^l)} \neq \emptyset\}$$

$$(\text{ind. hyp.}) \quad = \quad \mathrm{I}^{<k+l}(Q_1) \cup \mathrm{I}^{<k+l}(Q_2) \cup \{\tau \mid \mathrm{I}^{<k+l}(Q_1) \cap \overline{\mathrm{I}^{<k+l}(Q_2)} \neq \emptyset\}$$

$$(\text{def. of } \mathrm{I}^{\cdot}(\cdot)) \quad = \quad \mathrm{I}^{<k+l}(Q_1|Q_2)$$

The other cases are easier to establish than the ones above and, therefore, are omitted.

- We prove Part (3) by induction on k. The case $k = 0$ is trivial. Therefore, we directly consider the statement for $k = 1$ which can be proved as follows.

  For the *"only if"–direction* one may observe that $P \stackrel{1}{\longmapsto} P'$ implies $P \stackrel{\tau}{\not\longmapsto}$ by Proposition 4.3.1(2), i.e. $\tau \notin \mathrm{I}^{<1}(P)$, by Proposition 2.2.1(2). Thus, it remains to establish that $P' \doteq [P]^1$, for which we use structural induction on P.

  1. $P \equiv \alpha{:}k.Q$*:*
     $\alpha{:}k.Q \stackrel{1}{\longmapsto} P'$ implies $k > 0$ or ($k = 0$ and $\alpha \not\equiv \tau$) by $\mathsf{CCS}^{\mathsf{rt}}$ semantics. In the former case we have $P' \equiv \alpha{:}(k-1).Q \doteq [\alpha{:}k.Q]^1$ by the definition of the adjustment function. In the latter case we obtain $P' \equiv \alpha{:}0.Q \doteq [\alpha{:}k.Q]^1$, as desired.
  2. $P \equiv Q_1|Q_2$*:*
     $Q_1|Q_2 \stackrel{1}{\longmapsto} P'$ implies $Q_1 \stackrel{1}{\longmapsto} Q_1'$, $Q_2 \stackrel{1}{\longmapsto} Q_2'$, and $P' \equiv Q_1'|Q_2'$ for some $Q_1', Q_2' \in \mathcal{P}$. By induction hypothesis we may conclude $Q_1' \doteq [Q_1]^1$ and $Q_2' \doteq [Q_2]^1$. Hence, $P' \equiv Q_1'|Q_2' \doteq [Q_1]^1|[Q_2]^1 \equiv [Q_1|Q_2]^1$ by the definition of the adjustment function.

  The other cases follow by similar reasoning. For the *"if"–direction* let $\tau \notin \mathrm{I}^{<1}(P)$, i.e. $P \stackrel{\tau}{\not\longmapsto}$ by Proposition 2.2.1(2). Hence, $P \stackrel{1}{\longmapsto} P'$ for some $P' \in \mathcal{P}$ according to Proposition 4.3.1(1). Moreover, we have $P' \doteq [P]^1$ by the "only if"–direction of this proof part.

For the induction step let $k > 1$. Then, we have $P \stackrel{1}{\longmapsto}^{k+1} P'$ if and only if $P \stackrel{1}{\longmapsto} P'' \stackrel{1}{\longmapsto}^{k} P'$ for some $P'' \in \mathcal{P}$. By induction hypothesis and Part (3) for $k = 1$ this is exactly the case if and only if $P' \doteq [P'']^k$, $\tau \notin \mathrm{I}^{<k}(P'')$, $P'' \doteq [P]^1$, and $\tau \notin \mathrm{I}^{<1}(P)$, i.e. $P' \doteq [P]^{k+1}$ and $\tau \notin \mathrm{I}^{<k+1}(P)$ by Lemmata 4.5.1(1) and 4.5.1(2), respectively.

□

Now, we are able to state and prove our main results relating $\mathsf{CCS}^{\mathsf{dp}}$ and $\mathsf{CCS}^{\mathsf{rt}}$ semantics. Given an arbitrary process in our language there exists a one-to-one semantic correspondence between the associated transition systems according to $\mathsf{CCS}^{\mathsf{dp}}$ and to $\mathsf{CCS}^{\mathsf{rt}}$ semantics in the following sense.

Proposition 4.5.2 (One-to-one Correspondence)
Let $P, P' \in \mathcal{P}$, $\alpha \in \mathcal{A}$, and $k \in \mathbb{N}$. Then the following one-to-one correspondence holds: $P \stackrel{\alpha:k}{\longrightarrow} P'$ if and only if $\exists P'' \in \mathcal{P}.\, P \stackrel{1}{\longmapsto}^{k} P'' \stackrel{\alpha}{\longmapsto} P'$.

Proof: Let $P, P' \in \mathcal{P}$ and $k \in \mathbb{N}$. According to Lemma 4.5.1(3) it is sufficient to show that $[P]^k \stackrel{\alpha}{\longmapsto} P'$ and $\tau \notin \mathrm{I}^{<k}(P)$ if and only if $P \stackrel{\alpha:k}{\longrightarrow} P'$. The proof is done by induction on the structure of P.

1. $P \equiv \mathbf{0}$:

   Here, our statement trivially holds by the definition of potential initial action sets and since $\mathbf{0}$ cannot engage in any transition.

2. $P \equiv \alpha{:}l.P'$:

   According to $\mathsf{CCS}^{\mathsf{rt}}$ semantics $[\alpha{:}l.P']^k \stackrel{\alpha}{\longmapsto} P'$ is valid if $[\alpha{:}l.P']^k \doteq \alpha{:}0.P'$, which is exactly the case if $k \geq l$. Since $\tau \notin \mathrm{I}^{<k}(P)$ we know $k = l$ if $\alpha \equiv \tau$. Hence, $\alpha{:}l.P' \stackrel{\alpha:k}{\longrightarrow} P'$ by $\mathsf{CCS}^{\mathsf{dp}}$ semantics.

   Reversely, $\alpha{:}l.P' \stackrel{\alpha:k}{\longrightarrow} P'$ implies $k \geq l$, if $\alpha \not\equiv \tau$, and $k = l$, otherwise, according to $\mathsf{CCS}^{\mathsf{dp}}$ semantics. Thus, $[\alpha{:}l.P']^k \equiv \alpha{:}0.P'$ and $\alpha{:}0.P' \stackrel{\alpha}{\longmapsto} P'$ by the definitions of the adjustment function and of $\mathsf{CCS}^{\mathsf{rt}}$ semantics. Finally, $\tau \notin \mathrm{I}^{<k}(\alpha{:}l.P')$ since $\alpha \equiv \tau$ implies $k = l$.

3. $P \equiv Q_1 + Q_2$:

   Due to $\mathsf{CCS}^{\mathsf{rt}}$ semantics, the definitions of the adjustment function and of potential initial action sets, and Proposition 4.4.1, we obtain $[Q_1 + Q_2]^k \equiv [Q_1]^k + [Q_2]^k \stackrel{\alpha}{\longmapsto} P'$ and $\tau \notin \mathrm{I}^{<k}(Q_1 + Q_2)$ if and only if $([Q_1]^k \stackrel{\alpha}{\longmapsto} P'$ or $[Q_2]^k \stackrel{\alpha}{\longmapsto} P')$ and $\tau \notin \mathrm{I}^{<k}(Q_1) \cup \mathrm{I}^{<k}(Q_2)$. By induction hypothesis and Proposition 4.4.1 this is exactly the case if $(Q_1 \stackrel{\alpha:k}{\longrightarrow} P'$ and $\tau \notin \mathrm{I}^{<k}(Q_2))$ or $(Q_2 \stackrel{\alpha:k}{\longrightarrow} P'$ and $\tau \notin \mathrm{I}^{<k}(Q_1))$, which holds if and only if $Q_1 + Q_2 \stackrel{\alpha:k}{\longrightarrow} P'$ according to $\mathsf{CCS}^{\mathsf{dp}}$ semantics.

4. $P \equiv Q_1|Q_2$:

   Let $[Q_1|Q_2]^k \equiv [Q_1]^k|[Q_2]^k \stackrel{\alpha}{\longmapsto} P'$ (already exploiting the definition of the adjustment function) and $\tau \notin \mathrm{I}^{<k}(Q_1|Q_2)$. According to the semantics for parallel composition we may split this case into the following three subcases.

   (a) $[Q_1]^k \stackrel{\alpha}{\longmapsto} Q'$ for some $Q' \in \mathcal{P}$ and $P' \equiv Q'|[Q_2]^k$. Since $\tau \notin \mathrm{I}^{<k}(Q_1|Q_2)$ implies $\tau \notin \mathrm{I}^{<k}(Q_1)$ by the definition of potential initial action sets we may apply the induction hypothesis to conclude $Q_1 \stackrel{\alpha:k}{\longrightarrow} Q'$. This is exactly the case if $Q_1|Q_2 \stackrel{\alpha:k}{\longrightarrow} Q'|[Q_2]^k \equiv P'$ according to $\mathsf{CCS}^{\mathsf{dp}}$ semantics and the fact that $\tau \notin \mathrm{I}^{<k}(Q_1|Q_2)$.

   (b) $[Q_2]^k \stackrel{\alpha}{\longmapsto} Q'$ for some $Q' \in \mathcal{P}$ and $P' \equiv [Q_1]^k|Q'$. This case can be shown in a symmetric fashion to the previous one.

   (c) $\alpha \equiv \tau$, $[Q_1]^k \stackrel{a}{\longmapsto} Q_1'$, and $[Q_2]^k \stackrel{\overline{a}}{\longmapsto} Q_2'$ for some $a \in \mathcal{A} \setminus \{\tau\}$ and $Q_1', Q_2' \in \mathcal{P}$ such that $P' \equiv Q_1'|Q_2'$. Because of the premise $\tau \notin \mathrm{I}^{<k}(Q_1|Q_2)$ we also know $\tau \notin \mathrm{I}^{<k}(Q_1)$ and $\tau \notin \mathrm{I}^{<k}(Q_2)$. Thus, the induction hypothesis implies $Q_1 \stackrel{a:k}{\longrightarrow} Q_1'$ and $Q_2 \stackrel{\overline{a}:k}{\longrightarrow} Q_2'$, and also $\tau \notin \mathrm{I}^{<k}(Q_1|Q_2)$. According to $\mathsf{CCS}^{\mathsf{dp}}$ semantics this is equivalent to $Q_1|Q_2 \stackrel{\tau:k}{\longrightarrow} Q_1'|Q_2' \equiv P'$, as desired.

The remaining cases can be established easier and, therefore, are omitted. □

This proposition explicitly reflects our intuition about the meaning of a natural number attached to an action in both calculi. Whereas in $\mathsf{CCS}^{\mathsf{rt}}$ we interpret $\alpha{:}k$ as the action α which is enabled after a delay of (at least) k time units, the value k indicates the urgency of α in $\mathsf{CCS}^{\mathsf{dp}}$.

4.5.1 Bisimulation Correspondence

Proposition 4.5.2 is the key to prove the next theorem, which states that the two behavioral relations, prioritized and temporal bisimulation, coincide.

Theorem 4.5.3 (Bisimulation Correspondence)
Let $P, Q \in \mathcal{P}$. Then $P \sim_{dp} Q$ if and only if $P \sim_{rt} Q$.

Proof: We first prove the *"if"-direction* by showing that $\sim_{\mathtt{rt}}$ is a prioritized bisimulation. Let $P, Q \in \mathcal{P}$, $\alpha \in \mathcal{A}$, and $k \in \mathbb{N}$ satisfying $P \stackrel{\alpha:k}{\longrightarrow} P'$. By Proposition 4.5.2 we may conclude the existence of some $P'' \in \mathcal{P}$ such that $P \stackrel{1}{\longmapsto} {}^k P'' \stackrel{\alpha}{\longmapsto} P'$. Since $P \sim_{\mathtt{rt}} Q$ there exist some $Q', Q'' \in \mathcal{P}$ satisfying $Q \stackrel{1}{\longmapsto} {}^k Q'' \stackrel{\alpha}{\longmapsto} Q'$, $P'' \sim_{\mathtt{rt}} Q''$, and $P' \sim_{\mathtt{rt}} Q'$, which can formally be derived by a straightforward induction on k and the definition of $\sim_{\mathtt{rt}}$. Proposition 4.5.2 now implies $Q \stackrel{\alpha:k}{\longrightarrow} Q'$, as desired. For the *"only if"-direction* it is sufficient to show that

$$\mathcal{R}_t =_{\mathrm{df}} \{\langle [P]^k, [Q]^k\rangle \mid P \sim_{\mathtt{dp}} Q,\ \tau \notin \mathrm{I}^{<k}(P),\ \tau \notin \mathrm{I}^{<k}(Q),\ \text{and } k \in \mathbb{N}\}$$

is a temporal bisimulation. Note that $\langle P, Q\rangle \in \mathcal{R}_t$ by choosing $k = 0$ (cf. Lemma 4.5.1(1) and the fact that $\mathrm{I}^{<0}(\cdot) = \emptyset$). Let $\langle [P]^k, [Q]^k\rangle \in \mathcal{R}_t$ for some arbitrary $k \in \mathbb{N}$, i.e. $P \sim_{\mathtt{dp}} Q$, $\tau \notin \mathrm{I}^{<k}(P)$, and $\tau \notin \mathrm{I}^{<k}(Q)$.

First, consider $[P]^k \stackrel{\alpha}{\longmapsto} P'$ for some $P' \in \mathcal{P}$. Because of $\tau \notin \mathrm{I}^{<k}(P)$ we conclude $P \stackrel{1}{\longmapsto}{}^k [P]^k \stackrel{\alpha}{\longmapsto} P'$ by Lemma 4.5.1(3). Hence, $P \stackrel{\alpha:k}{\longrightarrow} P'$ according to Proposition 4.5.2. Since $P \sim_{\mathtt{dp}} Q$ we know of the existence of some $Q' \in \mathcal{P}$ such that $Q \stackrel{\alpha:k}{\longrightarrow} Q'$ and $P' \sim_{\mathtt{dp}} Q'$. Now, we use Proposition 4.5.2 and Lemma 4.5.1(3) again to obtain $[Q]^k \stackrel{\alpha}{\longmapsto} Q'$. Moreover, $\langle [P']^0, [Q']^0\rangle \in \mathcal{R}_t$ can be derived from $P' \sim_{\mathtt{dp}} Q'$, as desired.

Second, let $[P]^k \stackrel{1}{\longmapsto} P'$ for some $P' \in \mathcal{P}$. Hence, $[P]^k \stackrel{\tau}{\not\longmapsto}$ by Proposition 4.3.1(2), i.e. $\tau \notin \mathrm{I}^{<1}([P]^k) = \mathrm{I}^{<k+1}(P)$ by Proposition 2.2.1(2) and Lemma 4.5.1(3), and $P' \doteq [P]^{k+1}$ by Lemmata 4.5.1(1) and 4.5.1(3). Due to the first case we know $[Q]^k \stackrel{\tau}{\not\longmapsto}$, i.e. $\tau \notin \mathrm{I}^{<1}([Q]^k) = \mathrm{I}^{<k+1}(Q)$ according to Proposition 2.2.1(2) and Lemma 4.5.1(3). Now, Lemma 4.5.1(3) is applicable, and $[Q]^k \stackrel{1}{\longmapsto} [[Q]^k]^1 \doteq [Q]^{k+1}$ holds by Lemma 4.5.1(1). Moreover, $\langle [P]^{k+1}, [Q]^{k+1}\rangle \in \mathcal{R}_t$ by the definition of $\mathcal{R}_t$, which finishes the proof. □

As a consequence of this result, prioritized and temporal bisimulation possess the same properties. Especially, we may conclude that prioritized bisimulation is a congruence for our algebra.

4.5.2 Logical Correspondence

$\mathsf{CCS}^{\mathsf{dp}}$ and $\mathsf{CCS}^{\mathsf{rt}}$ semantics can be logically related, too. We formally establish this correspondence using a variant of the *modal μ-calculus* [109] as *temporal logic*. More precisely, we adapt the variant presented in Section 3.6.3 for $\mathsf{CCS}^{\mathsf{rt}}$ and $\mathsf{CCS}^{\mathsf{dp}}$ by replacing the semantics of the operators $\langle \alpha:k\rangle$. For convenience, we write $\{\!|\Phi|\!\}_{\mathtt{dp}}(\sigma)$ for the $\mathsf{CCS}^{\mathsf{dp}}$ version of the μ-calculus semantics and $\{\!|\Phi|\!\}_{\mathtt{rt}}(\sigma)$ for the $\mathsf{CCS}^{\mathsf{rt}}$ version of the μ-calculus semantics of the formula Φ with respect to the environment σ. We define

$$\{\!|\langle \alpha:k\rangle\Phi|\!\}_{\mathtt{dp}}(\sigma) =_{\mathrm{df}} \{P \in \mathcal{P} \mid \exists P' \in \mathcal{P}.\, P \stackrel{\alpha:k}{\longrightarrow} P' \text{ and } P' \in \{\!|\Phi|\!\}_{\mathtt{dp}}(\sigma)\}$$

as well as

$$\{\!|\langle \alpha:k\rangle\Phi|\!\}_{\mathtt{rt}}(\sigma) =_{\mathrm{df}} \{P \in \mathcal{P} \mid \exists P', P'' \in \mathcal{P}.\, P \stackrel{1}{\longmapsto}{}^k P'' \stackrel{\alpha}{\longmapsto} P' \text{ and } P' \in \{\!|\Phi|\!\}_{\mathtt{rt}}(\sigma)\}\ .$$

Intuitively, the formula $\langle \alpha:k\rangle\Phi$ is satisfied by those processes that may engage in an α–transition with delay/priority value k to some process for which Φ holds. The next theorem shows that processes satisfy the same formulas, independently if those are interpreted with respect to $\mathsf{CCS}^{\mathsf{dp}}$ or $\mathsf{CCS}^{\mathsf{rt}}$ semantics.

Theorem 4.5.4 (Logical Coincidence)
Let $P \in \mathcal{P}$, $\Phi \in \mathcal{F}$, and let σ be an arbitrary environment. Then $\{\!|\Phi|\!\}_{dp}(\sigma) = \{\!|\Phi|\!\}_{rt}(\sigma)$.

Proof: The proof is done by induction on the structure of the formula Φ.

- $\Phi \equiv \mathit{tt}$**:** This case is trivial since $\{\!|\mathit{tt}|\!\}_{\mathtt{dp}}(\sigma) = \mathcal{P} = \{\!|\mathit{tt}|\!\}_{\mathtt{rt}}(\sigma)$.

- $\Phi \equiv \neg\Psi$**:**

$$\begin{array}{lcl} & & \{\!|\neg\Psi|\!\}_{\mathtt{dp}}(\sigma) \\ (\text{def. of } \{\!|\cdot|\!\}_{\mathtt{dp}}) & = & \mathcal{P} \setminus \{\!|\Psi|\!\}_{\mathtt{dp}}(\sigma) \\ (\text{ind. hyp.}) & = & \mathcal{P} \setminus \{\!|\Psi|\!\}_{\mathtt{rt}}(\sigma) \\ (\text{def. of } \{\!|\cdot|\!\}_{\mathtt{rt}}) & = & \{\!|\neg\Psi|\!\}_{\mathtt{rt}}(\sigma) \end{array}$$

- $\Phi \equiv \Psi_1 \wedge \Psi_2$**:**

$$\begin{array}{lcl} & & \{\!|\Psi_1 \wedge \Psi_2|\!\}_{\mathtt{dp}}(\sigma) \\ (\text{def. of } \{\!|\cdot|\!\}_{\mathtt{dp}}) & = & \{\!|\Psi_1|\!\}_{\mathtt{dp}}(\sigma) \cap \{\!|\Psi_2|\!\}_{\mathtt{dp}}(\sigma) \\ (\text{ind. hyp.}) & = & \{\!|\Psi_1|\!\}_{\mathtt{rt}}(\sigma) \cap \{\!|\Psi_2|\!\}_{\mathtt{rt}}(\sigma) \\ (\text{def. of } \{\!|\cdot|\!\}_{\mathtt{rt}}) & = & \{\!|\Psi_1 \wedge \Psi_2|\!\}_{\mathtt{rt}}(\sigma) \end{array}$$

- $\Phi \equiv \langle\alpha{:}k\rangle\Psi$**:**

$$\begin{array}{lcl} & & \{\!|\langle\alpha{:}k\rangle\Psi|\!\}_{\mathtt{dp}}(\sigma) \\ (\text{def. of } \{\!|\cdot|\!\}_{\mathtt{dp}}) & = & \{P \in \mathcal{P} \mid \exists P' \in \mathcal{P}, k \in \mathbb{N}.\, P \xrightarrow{\alpha:k} P' \text{ and } P' \in \{\!|\Psi|\!\}_{\mathtt{dp}}(\sigma)\} \\ (\text{ind. hyp.}) & = & \{P \in \mathcal{P} \mid \exists P' \in \mathcal{P}, k \in \mathbb{N}.\, P \xrightarrow{\alpha:k} P' \text{ and } P' \in \{\!|\Psi|\!\}_{\mathtt{rt}}(\sigma)\} \\ (\text{Prop. 4.5.2}) & = & \{P \in \mathcal{P} \mid \exists P', P'' \in \mathcal{P}.\, P \stackrel{1}{\longmapsto}{}^{k} P'' \stackrel{\alpha}{\longmapsto} P' \text{ and } P' \in \{\!|\Psi|\!\}_{\mathtt{rt}}(\sigma)\} \\ (\text{def. of } \{\!|\cdot|\!\}_{\mathtt{rt}}) & = & \{\!|\langle\alpha{:}k\rangle\Psi|\!\}_{\mathtt{rt}}(\sigma) \end{array}$$

- $\Phi \equiv \mu X.\Psi$**:**

$$\begin{array}{lcl} & & \{\!|\mu X.\Psi|\!\}_{\mathtt{dp}}(\sigma) \\ (\text{def. of } \{\!|\cdot|\!\}_{\mathtt{dp}}) & = & \bigcap\{\mathcal{P}' \subseteq \mathcal{P} \mid \{\!|\Psi|\!\}_{\mathtt{dp}}(\sigma[\mathcal{P}'/X]) \subseteq \mathcal{P}'\} \\ (\text{ind. hyp.}) & = & \bigcap\{\mathcal{P}' \subseteq \mathcal{P} \mid \{\!|\Psi|\!\}_{\mathtt{rt}}(\sigma[\mathcal{P}'/X]) \subseteq \mathcal{P}'\} \\ (\text{def. of } \{\!|\cdot|\!\}_{\mathtt{rt}}) & = & \{\!|\mu X.\Psi|\!\}_{\mathtt{rt}}(\sigma) \end{array}$$

□

Hence, properties of processes interpreted with respect to $\mathsf{CCS}^{\mathtt{dp}}$ semantics hold also in the $\mathsf{CCS}^{\mathtt{rt}}$ interpretation, and vice versa. It is worth noting that by leaving out the fixed point operators μX we obtain versions of the so called *Hennessy-Milner logic* [125]. As for CCS and strong bisimulation, a logical characterization of temporal bisimulation can be provided along the lines of [125]. Unfortunately, the new logic for our dynamic-priority approach cannot be shown to characterize prioritized bisimulation by using standard techniques. The problem is caused by the potential *infinite-branching* transition systems (cf. [125]) which may arise according to Rule (Act1). However, in practice finite-branching can be enforced, as discussed in Section 4.4, such that the mentioned problem vanishes. Since the logical characterizations of our bisimulations are not of importance here, we do not consider them further.

4.6 Discussion and Related Work

One may wonder why the $\mathsf{CCS}^{\mathsf{dp}}$ semantics does not only consider actions with minimal delay or priority values as labels of transition systems. In particular, we could have avoided the side condition of Rule (Act1) yielding potentially infinite-branching transition systems by allowing communication on different priority levels. The reason that we have not followed this approach is that it imposes an unsound abstraction [68, 104] with respect to $\mathsf{CCS}^{\mathsf{rt}}$ semantics. For instance, consider the process $P \stackrel{\text{def}}{=} (a{:}1.b{:}0.\mathbf{0} \mid (\overline{b}{:}1.\mathbf{0} + c{:}2.\mathbf{0})) \setminus \{b\}$. According to the modified $\mathsf{CCS}^{\mathsf{dp}}$ semantics, P can engage in an a-transition with priority value 1 to the process $(b{:}0.\mathbf{0} \mid (\overline{b}{:}0.\mathbf{0} + c{:}1.\mathbf{0})) \setminus \{b\}$. Hence, after an a-transition, a c-transition is always pre-empted since a communication on b with priority value 0 is possible. According to the original $\mathsf{CCS}^{\mathsf{dp}}$ semantics however, P may also engage into an a-transition with priority value 2 yielding $(b{:}0.\mathbf{0} \mid (b{:}0.\mathbf{0} + c{:}0.\mathbf{0})) \setminus \{b\}$. Thus, there exists a path starting with an a-transition, after which the action c may be observed. Hence, cutting off this additional path changes the behavior of P and, therefore, the modified $\mathsf{CCS}^{\mathsf{dp}}$ semantics is incorrect with respect to original $\mathsf{CCS}^{\mathsf{dp}}$ and $\mathsf{CCS}^{\mathsf{rt}}$ semantics (cf. Theorem 4.5.3).

Regarding related work, the closest related approach has been made by Jeffrey [101] who has established a formal relationship between a quantitative real-time process algebra and a process algebra with *static*-priority which is very similar to our algebra $\mathsf{CCS}^{\mathsf{ch}}$ presented in Chapter 2. Jeffrey also translates real-time to priority based on the idea of time-stamping. In contrast to $\mathsf{CCS}^{\mathsf{rt}}$ semantics, a process modeled in Jeffrey's framework may either *immediately* engage in an action transition or idle forever. However, this semantics does not reflect our intuition about the behavior of *reactive* systems, i.e. a process should wait until a desired communication partner becomes available, instead of engaging in a "livelock." It is only because of this counter-intuitive assumption, that Jeffrey does not need to choose a *dynamic*-priority framework.

In [37] a variant of CCSR [38], referred to as CCSR92, has been introduced which does not only allow for modeling static-priority but also for dynamic-priority. Since CCSR focuses on specifying and verifying real-time concurrent system, the ability to capture scheduling behavior is central. Thus, a notion of dynamic-priority, such as in priority-inheritance and earliest-deadline-first scheduling algorithms, is crucial. Dynamic priorities are given as a function of the history of the system under consideration, in a way that priority values can also be calculated from the system's history. Accordingly, the operational semantics of CCSR92 is re-defined to include the history context. The authors show that dynamic priorities do in general not lead to a compositional semantics and give a sufficient condition that ensures compositionality.

Finally, we want to note that the concept of priority is not only related to real-time but also to probability [46, 90, 152, 166], which has been made precise in [154], and to stochastic process algebras [18, 161]. However, the former relationship requires a static-priority approach rather than a dynamic one.

4.7 Summary

We have introduced the process algebra $\mathsf{CCS}^{\mathsf{dp}}$ with dynamic-priority whose semantics corresponds one-to-one with the discrete, quantitative real-time semantics of $\mathsf{CCS}^{\mathsf{rt}}$. Thus, a semantic theory of $\mathsf{CCS}^{\mathsf{dp}}$ based on the notion of bisimulation, including axiomatic and logical characterizations, can be adapted from $\mathsf{CCS}^{\mathsf{rt}}$. However, the $\mathsf{CCS}^{\mathsf{dp}}$ semantics yields significantly more compact models. The state space of models is usually by several factors smaller, depending on the delay values specified in the system under consideration, and the number of transitions is at least not worse for $\mathsf{CCS}^{\mathsf{dp}}$ semantics than for $\mathsf{CCS}^{\mathsf{rt}}$ semantics. Hence, $\mathsf{CCS}^{\mathsf{dp}}$ semantics provides a mean for efficiently implementing traditional real-time process algebras. Moreover, our approach does not abstract away any aspects of real-time, i.e. all quantitative timing constraints can still be verified within $\mathsf{CCS}^{\mathsf{dp}}$ semantics. The compactness of models can be drastically improved further if one is not interested in compositionality and in verifying properties involving quantitative time. In this case a $\mathsf{CCS}^{\mathsf{dp}}$ model can be collapsed by abstracting away from all priority values of the labels and, subsequently, by minimizing it according to standard bisimulation [125]. Finally, it should be mentioned that $\mathsf{CCS}^{\mathsf{rt}}$ and $\mathsf{CCS}^{\mathsf{dp}}$ have been implemented as front-ends of the `CWB-NC` [65].

Chapter 5

Case Study: The SCSI-2 Bus-Protocol

5.1 Introduction

This chapter demonstrates the utility of the process algebra $\mathsf{CCS}^{\mathsf{dp}}$ with dynamic-priority, developed in Chapter 4, for system verification by a case study dealing with the bus-protocol of the *Small Computer System Interface*, or SCSI for short, a protocol used in many of today's computers. The protocol's model is derived from the official standard [3] as issued by the *American National Standard Institute* (ANSI), where real-time delays are recommended for implementing synchronization constraints as well as for ensuring correct behavior in the presence of signal glitches. Accurate modeling of the SCSI-2 bus-protocol thus requires considering real-time. To this end, we model our protocol in the syntax common to both $\mathsf{CCS}^{\mathsf{rt}}$ and $\mathsf{CCS}^{\mathsf{dp}}$. We then generate the state spaces according to both semantics. We show that the size of our model is almost an order of magnitude smaller in the $\mathsf{CCS}^{\mathsf{dp}}$ semantics than in the $\mathsf{CCS}^{\mathsf{rt}}$ semantics. The modeling of the protocol has been carried out in the CWB-NC [65, 153], a tool for analyzing and verifying concurrent systems. In order to verify and to prove the accuracy of our model, we extract several mandatory properties from the ANSI document and validate them for our model. We use the version of the *modal μ-calculus* defined in the previous chapter as our specification language, and automatically check the formalized properties by using the *local model checker* [25, 26] integrated in the CWB-NC.

The body of this chapter is organized as follows. A general introduction to the SCSI bus and its protocol is given in the next section. Section 5.3 presents central design decisions as well as modeling assumptions. Sections 5.4–5.6 contain the models of the bus signals, the data bus, and the bus phases for connection establishment and information transfer. In the following section several desired properties of the bus-protocol are stated, specified in the modal μ-calculus, and verified by using model checking. Finally, Section 5.8 summarizes this chapter.

5.2 The SCSI-2 Bus-Protocol

The *SCSI bus* is designed to provide an efficient peer-to-peer I/O connection for peripheral devices such as disks, tapes, printers, processors, etc. It usually connects several of these devices with one *host adapter* which often resides on a computer's motherboard. In contrast to the host adapter, peripherals are not attached directly to the bus but via *controllers*, also called *logical units* (LUNs). Thus, LUNs provide the physical and logical interface between the bus and the peripherals. Conceptually, up to seven LUNs can be connected to one bus, and one LUN can support up to seven peripherals. However, in practice most peripherals contain their own SCSI controller (cf. Figure 5.1).

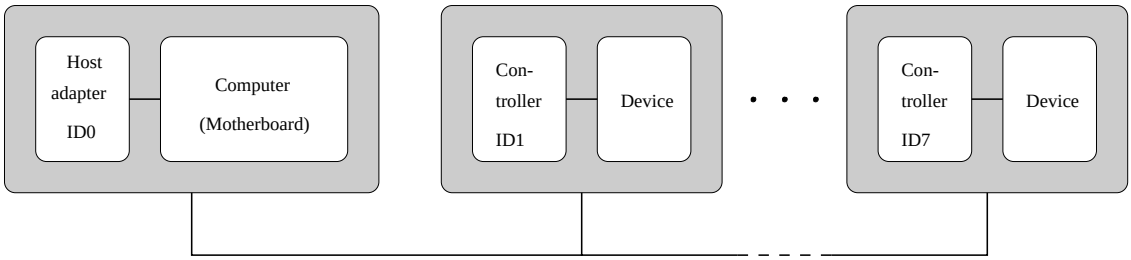

Figure 5.1: Typical SCSI configuration

The *SCSI-2 bus-protocol* implements the logical mechanism regulating how peripherals and the host adapter communicate with each other on the bus. Communication on the SCSI bus is point-to-point, i.e. at any time either none or exactly two LUNs may communicate with each other. In order to allow easy addressing each LUN is assigned a fixed SCSI id in form of a number ranging from one to seven. Id 0 is reserved for the host adapter which is also, conceptually, a LUN. Communication on the bus is organized by the use of eight signal lines whereas the actual information, like *messages*, *commands*, *data*, and *status information*, are transferred over a data bus.

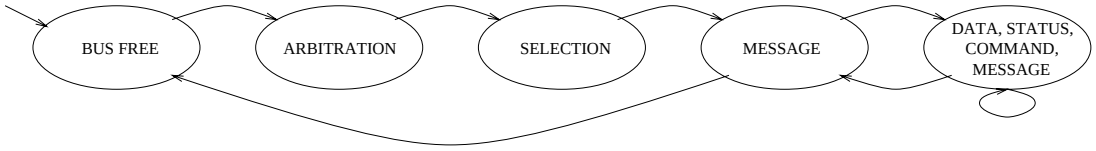

Figure 5.2: Usual progression of the SCSI-2 bus-phases

The SCSI-2 bus-protocol is organized in eight distinct phases: Bus Free, Arbitration, Selection, Reselection, Command, Data, Status, and Message phase. At any given time, the SCSI bus is exactly in one phase. The usual progression of phases is shown in Figure 5.2. During the Bus Free phase no device is in possession of the bus, i.e. LUNs may request access. If more than one device competes for the bus in order to initiate a communication, the one with the highest SCSI id is granted access. In the Arbitration phase, every LUN that has posed a request determines if it has won the competition. All LUNs which lose

may compete for the bus again later, whereas the winner, also referred to as *initiator*, proceeds to the `Selection` phase. In this phase the initiator tries to connect to the desired destination, called *target*. When the link between initiator and target has been established, the so-called *information transfer phases*, including the `Command`, the `Data`, the `Status`, and the `Message` phases are entered. In the `Command` phase the target may request a command from the initiator. During a `Message` phase information is exchanged between the initiator and the target concerning the bus-protocol itself. Finally, the `Status` phase is used to transfer status information to the initiator upon completion of a command executed by the target. The key idea for accelerating communication on the bus, which has significantly contributed to the success of `SCSI`, is that the target can free the bus whenever it receives a time-intensive command from the initiator. As soon as the execution of such a command is finished, the target competes for the bus in order to transmit the result to the former initiator. As a simple example, one may think of the initiator as the host adapter, of the target as a hard disk, and of the command as the request to read a certain block from that hard disk. Since accessing hard disks takes some time, the bus can be used for other purposes until the requested block is found and its data is ready for transmission to the host adapter.

5.3 Modeling the SCSI-2 Bus-Protocol

In this section we model the `SCSI-2` bus-protocol in our language. The syntax we use here is the one implemented in the `CWB-NC` in which $\mathsf{CCS}^{\mathsf{dp}}$ and $\mathsf{CCS}^{\mathsf{rt}}$ have been integrated as front-ends. It slightly departs from the syntax introduced in Section 4.2 by using the conventions already presented in Section 3.3. Moreover, we use the notation $\alpha(\mathtt{obs}):k$ which, for the purposes of this section, may be interpreted as $\alpha:k$. Actions `obs` come into play in the next section where they serve as "probes" for verification purposes.

Before we present the actual modeling of the `SCSI-2` bus-protocol, we comment on some assumptions we imposed. First, we restrict ourselves to modeling two `LUN`s, called `LUN0` and `LUN1`, having id 0 and id 1, respectively. This is sufficient for dealing with the aspects of the `SCSI-2` bus-protocol we are interested in. Note that even in the situation of two `LUN`s there exists competition for the bus. Moreover, we abstract from timeout procedures and from the contents of most messages, commands, and data. These abstractions are justified since they do not affect the conceptual parts of the bus-protocol's behavior. For example, the sole purpose of a timeout is to determine if a target is alive or not. The contents of information sent over the bus, except from messages presenting the completion of some transmission, are only relevant for the device-specific part of `LUN`s but not for the bus-protocol itself. Additionally, the bus signals `BSY` (*busy*) and `SEL` (*select*) are *wired-or* signals in reality. However, we need not model this "or"-behavior, since our model only deals with two `LUN`s, and just one `LUN` at a time can assert the `BSY` or `SEL` signal. Finally, all quantitative timing information occurring in the model is measured relative to a time unit of 5 ns, including *arbitration delays* (480 time units), *bus clear delays* (160 time units), *bus*

settle delays (80 time units), *deskew delays* (9 time units), and *cable skew delays* (9 time units).

The underlying structure of the bus-protocol is explicitly reflected in our model. Each `LUN` connected to the bus is modeled as a separate parallel component containing models of the different bus phases as discussed in the previous section. The logical behavior of the bus control is implemented by bus signals. Each signal physically consists of a wire which we model as a separate process similar to a global Boolean variable. Note that signal delays are not modeled in the wires but in the operations used for transmitting information over the `SCSI` bus. Since we abstract away the content of most information, we do not need to model each bit of the data bus. Hence, arbitration is modeled via a global variable which stores the highest id of all `LUN`s requesting access to the bus. Accordingly, the structure of our model, called `SCSIBus`, consists of the parallel composition of both `LUN`s, and the `BusSignals`, including the regular signals and the data path. Formally,

```
proc SCSIBus = (LUN0 | BusSignals | LUN1) \ Restriction
```

where `Restriction` contains all actions that are internal to the protocol, i.e. those concerned with setting/releasing signals, requesting signal status, and placing/reading messages, commands, and data on/from the data bus.

5.4 Modeling the Bus Signals and the Data Bus

Conceptually, each bus signal is modeled as a Boolean variable which is either true (signal on) or false (signal off). Thus, the processes representing the signals `BSY` (*busy*), `SEL` (*select*), `C/D` (*command/data*), `I/O` (*input/output*), `MSG` (*message*), `ATN` (*attention*), `REQ` (*request*), and `ACK` (*acknowledgment*) are generically created by relabeling the actions of the process `Off` (see Table 5.1). Using the ports `set` and `rel` one can set or release the signal, i.e. switch the state to `On` or `Off`, respectively. Actions `'off` (`'on`) indicate that the signal is currently in state `Off` (`On`). Note that the atomicity of actions in process algebras guarantees that conflicts, arising by setting several signals simultaneously, are avoided.

In the following, we abstract away the contents of most messages. Only the distinguished messages `disconnect` and `complete` are explicitly considered since they require to exit the information transfer phases and to switch to the initial state of the `LUN`. Accordingly, we may model the data bus, as seen in Table 5.1, as a variable which can store and read out information (actions `placeXXX` and `readXXX`, respectively). The labels `obsXXX` are used to record the events of placing and reading messages on the bus.

For modeling arbitration we introduce the process `Arbitrator` which models a variable that stores the value of the highest id of all `LUN`s which compete for the bus. The situation in which no `LUN` wants to access the bus is captured by a special "undefined" state. Accordingly, the process `Arbitrator` possesses three states as shown in Table 5.1, called `Undef`, `Id0` and `Id1`, respectively. One may set the variable to state `Id`k via port `setid`k whenever the current state of `Arbitrator` is either `Undef` or `Id`j for $j \leq k$. In other words,

Table 5.1: Model of the bus signals, the data bus, and the arbitration variable

```
proc BusSignals =   DataBus
                  | Arbitrator
                  | Off[setBSY/set,relBSY/rel,isBSY/on,noBSY/off]
                  | Off[setSEL/set,relSEL/rel,isSEL/on,noSEL/off]
                  | ...

proc Off  =   'off:0.Off + set:0.On + rel:0.Off
proc On   =   'on:0.On   + set:0.On + rel:0.Off

proc DataBus  = DataBus' [> release(obsrelease):0.DataBus
proc DataBus' =   placemsgIn(obsplace):0.'readmsgIn(obsread):0.DataBus'
                + placemsgOut(obsplace):0.'readmsgOut(obsread):0.DataBus'
                + placefinished(obsplace):0.'readfinished(obsread):0.DataBus'
                + placedata(obsplace):0.'readdata(obsread):0.DataBus'
                + placecmd(obsplace):0.'readcmd(obsread):0.DataBus'
                + placestatus(obsplace):0.'readstatus(obsread):0.DataBus'
                + sentdisconnect(obssentdisconnect):0.
                  'readdisconnect(obsreaddisconnect):0.DataBus'
                + sentcomplete(obssentcomplete):0.
                  'readcomplete(obsreadcomplete):0.DataBus'
                + writetarget0(obswritet0):0.'readtarget0(obsreadt0):0.DataBus'
                + writetarget1(obswritet1):0.'readtarget1(obsreadt1):0.DataBus'

proc Arbitrator =  Undef [> clear:0.Arbitrator

proc Undef = setid0:0.Id0 + setid1:0.Id1 + 'noid0:0.Undef + 'noid1:0.Undef
proc Id0   = setid0:0.Id0 + setid1:0.Id0 + 'isid0:0.Id0   + 'noid1:0.Id0
proc Id1   = setid0:0.Id1 + setid1:0.Id1 + 'noid0:0.Id1   + 'isid1:0.Id1
```

the variable always maintains its maximum value. However, it may be reset to its initial state `Undef` via port `clear`. In reality, the `LUN`s that want to compete for access put their id on the data bus. Before acquiring the bus the `LUN` has to check if a higher id than its own is asserted. Modeling this technique one-to-one implies to implement the n bit wide data bus, where n is the number of `LUN`s attached to the bus. This induces a semantic complexity of 2^n states, compared to $n+1$ states used by our technique.

5.5 Modeling the Bus Phases for Connection Establishment

In this section we focus on modeling the logical characteristics of the `SCSI-2` bus-protocol (see Section 6 of [3]) for the initial bus phases handling connection establishment.

Table 5.2: `Bus Free`, `Arbitration`, and `Selection` phase

```
proc LUN0  =   t(start0):9.'relIO:0.(BusFree0 + GetSelected0)
             + t(start0):9.'setIO(obs_setIO):0.(BusFree0 + GetSelected0)
             + t:9.LUN0
             + GetSelected0

proc BusFree0 =   t(busfree):80.'setBSY(obs_setBSY):80.'setid0:0.Arbitrate0
                + isSEL(obs_isSEL):0.LUN0
                + isBSY(obs_isBSY):0.LUN0

proc Arbitrate0 =   noid1(obs_winner_id0):480.'setSEL(obs_setSEL):0.Selection0
                  + isid1(obs_winner_id1):480.LUN0


proc Selection0 = 'writetarget1:240.'setATN:9.'relBSY(obs_relBSY):18.
                  isBSY:80.'relSEL(obs_relSEL):9.t(begin_ITP):0.
                  (noIO:0.Initiator0 + isIO:0.Target0)


proc GetSelected0 = isATN:0.( isSEL:0.noBSY:0.readtarget0:0.'setBSY(obs_setBSY):0.
                              'release:0.'clear:0.noSEL:0.
                              (noIO:0.Target0 + isIO:0.Initiator0)
                            + noSEL:0.LUN0
                            )

proc Initiator0  =  H0 [> noBSY(obs_noBSY):0.'relATN:0.LUN0

proc H0 =   t:9.'setATN(obs_setATN):9.H0
          + isREQ(obs_isREQ):9.
            ( noMSG:0.( noCD:0.(noIO:0.DataOutI0 + isIO:0.DataInI0)
                      + isCD:0.(noIO:0.CommandI0 + isIO:0.StatusI0)
                      )
            + isMSG:0.isCD:0.(noIO:0.MsgOutI0 + isIO:0.MsgInI0)
            )

proc Target0 = (noIO:0.MsgOutT0 + isIO:0.'relATN:0.MsgInT0) [>
               noBSY:0.'relATN:0.LUN0
```

In the `Bus Free` phase, no device is in possession of the bus; hence it is available for arbitration. The `SCSI` bus is defined to be in the `Bus Free` phase as soon as the signals `SEL` and `BSY` have been off for at least a bus settle delay. Accordingly, the process `BusFree0` of `LUN0` detects the `Bus Free` phase when the actions `isBSY` and `isSEL` are absent for 80 time

units (cf. Table 5.2). If one of the actions isBSY or isSEL is observed, the bus is occupied and LUN0 returns to the start state. If the bus is free, the logical unit asserts the BSY signal (action 'setBSY) and sets the arbitration variable accordingly (action 'setid0) before it performs an arbitration delay and switches to the Arbitration phase.

In the *Arbitration phase* a LUN, which competes for access to the bus, looks up if it has won the arbitration by checking if no device with a higher id has asserted its id on the bus. Before the winner proceeds to the Selection phase, it asserts the SEL signal. All LUNs that have lost arbitration return to their initial states. The models of the Arbitration phase as well as of the Selection phase are presented in Table 5.2.

The *Selection phase* is distinguished from the *Reselection phase* by the de-asserted I/O signal. In the Selection phase the winning LUN, the initiator, tries to connect to the desired destination, the target, which is in case of Selection01 the logical unit LUN1. Therefore, it writes the id of the target on the data bus (action 'writetarget) and asserts the ATN signal to force each device to check if it is the desired target. The initiator then waits for some deskew delays and releases the BSY signal. After a short delay it looks for a response from the target. If the BSY signal is asserted, the target has responded and taken over control of the bus-protocol. In this case the initiator releases the SEL signal (action 'relSEL) and then behaves as Initiator0, or as Target0 in case of a Reselection phase.

If the ATN signal is asserted, each device looks up if the bus-protocol is in the Selection or Reselection phase (cf. process GetSelected0). Therefore, it checks if the SEL signal is asserted (action isSEL) and waits until the initiator releases the BSY signal (action 'relBSY). Then it asserts the BSY signal (action 'setBSY), releases the data bus (action 'release), and re-initializes the arbitration variable (action 'clear) before behaving as Target0, or Initiator0 in case of the Reselection phase.

After the Arbitration and (Re)Selection phases the target – being the master of the bus-protocol – proceeds to the MessageOut or MessageIn phase depending on if it has been selected as target or if it wants to re-connect to a former initiator, as indicated by the status of the IO signal (cf. Table 5.2). The initiator – the slave of the bus–protocol – continuously checks the status of the signals MSG, C/D, and I/O in order to determine the next phase selected by the target. Moreover, it may indicate its wish to proceed to the MessageOut phase by asserting the ATN signal. Finally, upon detection of the de-assertion of the BSY signal, caused by the target's expected or unexpected release of the SCSI bus, the initiator de-asserts the ATN signal and returns to its initial state.

5.6 Modeling the Information Transfer Phases

The processes Target0 and Initiator0 initiate the Information Transfer Phases (ITP) which include the Command, Data, Status, and Message phases. In those phases, information is exchanged between the initiator and the target. The Data and the Message phases are further divided in DataIn, DataOut, MessageIn, and MessageOut phases according to the direction of information flow. The "In" phases are concerned with transferring in-

Table 5.3: `Command` phase

```
proc CommandI0  = isREQ:0.( 'placecmd:0.'setACK:9.noREQ:0.'release:0.
                            'relACK:0.CommandI0
                          + 'placefinished:0.'setACK:9.noREQ:0.'release:0.
                            'relACK:0.Initiator0'
                          )

proc CommandT0  = 'relMSG:0.'setCD:0.'relIO(begin_Command):0.t(begin_Phase):0.
                  CommandT0'

proc CommandT0' = 'setREQ:0.isACK(obs_isACK):0.
                  ( readcmd:0.'relREQ(obs_relREQ):0.noACK:0.CommandT0'
                  + readfinished:0.'relREQ(obs_relREQ):0.noACK:0.t(end_Phase):0.
                    (MsgOutT0 + MsgInT0 + DataOutT0 + DataInT0 + StatusT0)
                  )
```

formation from the target to the initiator whereas the "`Out`" phases are concerned with transferring information in the other direction. The information transfer takes place using a byte-wise *handshaking mechanism*. In the following, we explain the ***`Command`*** *phase* and its modeling in detail (cf. Table 5.3).

The `Command` phase is entered if the target, the master of the bus-protocol, intends to request a command from the initiator. The target indicates the `Command` phase by de-asserting the `MSG` and `I/O` signals and asserting the `C/D` signal. After waiting for a deskew delay the target requests a command from the initiator by setting the `REQ` signal (action `'setREQ`). In the meantime, the initiator detects that the target has switched to the `Command` phase by observing the status of the `MSG`, `C/D`, and `I/O` signals (cf. process `HO` in Table 5.2). Upon detection of the asserted `REQ` signal (action `isREQ`) the initiator places the first byte of the command on the data bus (action `'placecmd`), waits for a deskew delay, and asserts the `ACK` signal (action `'setACK`). After the target detects the asserted `ACK` signal (action `isACK`) it reads the command from the data bus (action `readcmd`) and releases the `REQ` signal (action `'relREQ`). At this point the handshake procedure for receiving (the first byte of) the command is completed. Now, the initiator may release the data bus (action `'release`) and the `ACK` signal (action `'relACK`). Alternatively, since a command may consist of more than one byte, the bus may remain in the `Command` phase, and the handshake mechanism may be repeated, until the message *finished* (action `readfinished`) has been transferred. Note that in practice the length of a command can always be determined from its first byte.

The ***`Data`*** *phases* are intended for transferring data between the target and the initiator. The ***`Status`*** *phase* is usually entered in order to transmit status information to the initiator,

Table 5.4: `Data` and `Status` phases

```
proc DataInI0  = isREQ:0.( readdata:0.'setACK:0.noREQ:0.'relACK:0.DataInI0
                         + readfinished:0.'setACK:0.noREQ:0.'relACK:0.HO
                         )

proc DataInT0  = 'relMSG:0.'relCD:0.'setIO(begin_DataIn):0.t(begin_Phase):0.
                 DataInT0'

proc DataInT0' =   'placedata:0.'setREQ:9.isACK(obs_isACK):0.'release:0.
                   'relREQ(obs_relREQ):0.noACK:0.
                   DataInT0'
                 + 'placefinished:0.'setREQ:9.isACK(obs_isACK):0.'release:0.
                   'relREQ(obs_relREQ):0.noACK(end_Phase):0.
                   (MsgOutT0 + MsgInT0 + StatusT0)


proc DataOutI0  = isREQ:0.
                  ( 'placedata:0.'setACK:9.noREQ:0.'release:0.'relACK:0.DataOutI0
                  + 'placefinished:0.'setACK:9.noREQ:0.'release:0.'relACK:0.HO
                  )

proc DataOutT0  = 'relMSG:0.'relCD:0.'relIO(begin_DataOut):0.t(begin_Phase):0.
                  DataOutT0'

proc DataOutT0' = 'setREQ:0.isACK(obs_isACK):0.
                  ( readdata:0.'relREQ(obs_relREQ):0.noACK:0.
                    DataOutT0'
                  + readfinished:0.'relREQ(obs_relREQ):0.noACK(end_Phase):0.
                    (MsgOutT0 + MsgInT0 + StatusT0)
                  )

proc StatusI0 = readstatus:0.'setACK:0.noREQ:0.'relACK:0.HO

proc StatusT0 = 'relMSG:0.'setCD:0.'setIO(begin_Status):0.t(begin_Phase):0.
                'placestatus:0.'setREQ:9.isACK(obs_isACK):0.'release:0.
                'relREQ(obs_relREQ):0.noACK(end_Phase):0.(MsgOutT0 + MsgInT0)
```

which is always one byte long, after the target has finished executing a command. These phases are modeled as shown in Table 5.4.

During a ***`MessageIn`*** or ***`MessageOut`*** *phase* (see Table 5.5) information is exchanged between initiator and target which concern the bus-protocol itself. It is important that forcing a `MessageOut` phase is the only way for the initiator to attract the attention of

Table 5.5: Message phases

```
proc MsgInI0  = isREQ:0.( readmsgIn:0.'setACK:0.noREQ:0.'relACK:0.MsgInI0
                        + readfinished:0.'setACK:0.noREQ:0.'relACK:0.H0
                        + readcomplete:0.'setACK:0.noREQ:0.'relACK:0.nil
                        + readdisconnect:0.'setACK:0.noREQ:0.'relACK:0.nil
                        )

proc MsgInT0  = 'setMSG:0.'setCD:0.'setIO(begin_MsgIn):0.t(begin_Phase):0.
                MsgInT0'

proc MsgInT0' =   'placemsgIn:0.'setREQ(obs_setREQ):9.isACK(obs_isACK):0.
                  'release:0.'relREQ(obs_relREQ):0.noACK:0.
                  MsgInT0'
                + 'placefinished:0.'setREQ(obs_setREQ):9.isACK(obs_isACK):0.
                  'release:0.'relREQ(obs_relREQ):0.noACK(end_Phase):0.
                  (MsgOutT0 + DataOutT0 + DataInT0 + CommandT0 + StatusT0)
                + 'sentcomplete:0.'setREQ:9.isACK(obs_isACK):0.'release:0.
                  'relREQ(obs_relREQ):0.noACK(end_Phase):0.t(end_ITP):0.
                  'relBSY(obs_relBSY):0.nil
                + 'sentdisconnect:0.'setREQ:9.isACK(obs_isACK):0.'release:0.
                  'relREQ(obs_relREQ):0.noACK(end_Phase):0.t(end_ITP):0.
                  'relBSY(obs_relBSY):0.nil

proc MsgOutI0  = isREQ:0.
                 ( 'placemsgOut:0.'setACK:9.noREQ:0.'release:0.
                   'relACK:0.MsgOutI0
                 + 'placefinished:0.'relATN:9.'setACK:0.noREQ:0.'release:0.
                   'relACK:0.H0
                 )

proc MsgOutT0  = isATN:0.'setMSG:0.'setCD:0.'relIO(begin_MsgOut):0.
                 t(begin_Phase):0.MsgOutT0'

proc MsgOutT0' = 'setREQ:0.isACK(obs_isACK):0.
                 ( readmsgOut:0.'relREQ(obs_relREQ):0.noACK(obs_noACK):0.MsgOutT0'
                 + readfinished:0.'relREQ(obs_relREQ):0.noACK:0.
                   ( t(end_Phase):0.
                     (MsgInT0 + DataOutT0 + DataInT0 + CommandT0 + StatusT0)
                   + t:0.MsgOutT0'
                   )
                 )
```

the target which is in charge of the bus-protocol. This may be the case if the initiator wants to send another command to the target. In order to do so, the initiator sets the ATN signal by performing the action 'setATN. After exiting the current phase the target detects the assertion of the ATN signal (cf. the maximal progress property of $\mathsf{CCS}^{\mathsf{rt}}$ and the concept of pre-emption in $\mathsf{CCS}^{\mathsf{dp}}$) and switches to the MessageOut phase by asserting the MSG and C/D signals and de-asserting the I/O signal. The target then requests the message by setting the REQ signal such that the usual handshake procedure takes place (cf. Command phase). It is worth mentioning that we consider two messages, complete and disconnect, explicitly since those messages require a special reaction from the initiator, namely to exit the information transfer phases.

The validity of each information byte received over the bus is checked with the help of a *parity bit*. If some corrupted information is received, a special message is put on the bus that requests the re-transmission of the information. This "software" solution is replaced by a "hardware" solution in the case of information corrupted during critical MessageOut phases. More precisely, if corrupted information has been received by the target, it does not exit the MsgOut phase, thereby indicating to the initiator to re-sent the last information. A parity error may always occur which we model by a nondeterministic choice right after receiving the message finished. In case of a parity error we switch to state MsgOutT0'. Otherwise, we leave the MessageOut phase, which is observed by action end_Phase.

5.7 Verifying the SCSI-2 Bus-Protocol

In this section we construct the state space of the model of the SCSI-2 bus-protocol, specify several safety and liveness properties which our model is expected to satisfy in a variant of the *modal μ-calculus* [109], and verify them by employing the *local model-checker* [25, 26] integrated in the CWB-NC. The one-to-one correspondence between $\mathsf{CCS}^{\mathsf{dp}}$ and $\mathsf{CCS}^{\mathsf{rt}}$ semantics ensures that the properties, verified for the $\mathsf{CCS}^{\mathsf{dp}}$ model, hold with respect to the $\mathsf{CCS}^{\mathsf{rt}}$ model, too.

5.7.1 State Spaces of the Model

We have run the CWB-NC on a SUN SPARC 20 workstation to construct the state spaces of our model. Whereas the model has 62 400 states and 65 624 transitions according to $\mathsf{CCS}^{\mathsf{rt}}$ semantics, it possesses only 8 391 states and 14 356 transitions with respect to $\mathsf{CCS}^{\mathsf{dp}}$ semantics. This drastic saving in state space emphasizes the utility of using dynamic priorities for modeling discrete, quantitative real-time.

5.7.2 Properties of Interest

The following desired requirements of the SCSI-2 bus-protocol have been extracted from the official ANSI document [3].

- *Property 1:* All bus phases are always reachable. This implies that the model is free of deadlocks.
- *Property 2:* Whenever a bus phase is entered, it is eventually exited.
- *Property 3:* The signals REQ and ACK do not change between two information transfer phases.
- *Property 4:* The signal BSY is on and the signal SEL off during information transfer phases.
- *Property 5:* Whenever a device sends a message on the bus, the message is eventually received by the intended LUN.
- *Property 6:* Whenever the initiator sets the ATN signal, the bus eventually enters the MessageOut phase.

Note that none of these properties contains real-time constraints. In fact, the properties describe the functional behavior of the SCSI-2 bus-protocol rather than real-time issues concerned with *hard deadlines* or *response times*.

5.7.3 Specifying the Properties

Since our properties of interest concerning the bus-protocol do not involve timing constraints, we abstract from delay/priority values in the semantics of μ-calculus formulas. In this vein, we replace the modal operators $\langle \alpha{:}k \rangle$ introduced in Section 4.5.2 by $\langle \alpha \rangle$, and define $\{\![\langle \alpha \rangle \Phi]\!\}_{\mathtt{rt}}(\sigma) =_{\mathrm{df}} \{P \in \mathcal{P} \mid \exists P', P'' \in \mathcal{P}.\, P \stackrel{1}{\longmapsto}{}^{*} P'' \stackrel{\alpha}{\longmapsto} P' \text{ and } P' \in \{\![\Phi]\!\}_{\mathtt{rt}}(\sigma)\}$ as well as $\{\![\langle \alpha \rangle \Phi]\!\}_{\mathtt{dp}}(\sigma) =_{\mathrm{df}} \{P \in \mathcal{P} \mid \exists P' \in \mathcal{P}, k \in \mathbb{N}.\, P \stackrel{\alpha:k}{\longrightarrow} P' \text{ and } P' \in \{\![\Phi]\!\}_{\mathtt{dp}}(\sigma)\}$. An adaptation of Theorem 4.5.4 can easily be shown to hold for the modified temporal logics, too. Therefore, we can verify our properties of the SCSI-2 bus-protocol within the more compact $\mathsf{CCS}^{\mathsf{dp}}$ model and conclude that those are also valid for the $\mathsf{CCS}^{\mathsf{rt}}$ model.

For convenience, we use analogous notational conventions for our logic as the ones presented in Section 3.6.3. Moreover, we introduce the following meta-formulas, where $\alpha, \beta \in \mathcal{A}$, $L \subseteq \mathcal{A}$, and Φ is a μ-calculus formula.

$$\mathit{between}(\alpha, \beta, \Phi) =_{\mathrm{df}} \nu X.[\alpha](\nu Y.(\Phi \wedge [\beta]X \wedge [-\beta]Y)) \wedge [-\alpha]X$$

$$\mathit{fair\text{-}follows}(\alpha, \beta, L, \Phi) =_{\mathrm{df}} \nu X.[\alpha](\nu Y.\mu Z.(\Phi \wedge [\beta]X \wedge [L]Y \wedge [-(\{\beta\} \cup L)]Z)) \wedge [-\alpha]X$$

The meta-formula $\mathit{between}(\alpha, \beta, \Phi)$ can be interpreted as follows. On every path it is always the case that after α, the formula Φ is true at every state until the action β is seen. Note that action β need not occur after α since β only *releases* the requirement that Φ be true at every state. The meta-formula $\mathit{fair\text{-}follows}(\alpha, \beta, L, \Phi)$ encodes that on every path it is always the case that after action α is seen, either Φ is always true until β is seen or Φ is

always true, and an action from L occurs infinitely often on the path. Note that on unfair paths, i.e. paths on which actions from L do occur infinitely often, action β has to appear eventually. Without this notion of fairness [81], which we use to encode e.g. that messages transferred over the SCSI bus have finite length, some properties cannot be validated.

Unfortunately, our process algebras $\mathsf{CCS}^{\mathsf{dp}}$ and $\mathsf{CCS}^{\mathsf{rt}}$ turn any visible action a or $\overline{a}$ into the internal action τ when communicating on port a. However, in order to prove any interesting property except deadlock, we have to observe certain actions of the system, e.g. those modeling the assertion and de-assertion of bus signals. Therefore, we attach to some actions a (either the input or the output action belonging to channel a) and the internal action τ a visible action or *probe* o, thus leading to a complex action $a(\mathtt{o})$, $\overline{a}(\mathtt{o})$, or $\tau(\mathtt{o})$, respectively. Whenever a transition labeled by $a(\mathtt{o})$ ($\overline{a}(\mathtt{o})$) synchronizes with a transition labeled by $\overline{a}$ (a), the resulting τ is annotated by o, i.e. $\tau(\mathtt{o})$ is produced. Hence, a communication on port a is immediately observed by probe o, as intended. Our model includes the probes begin_Phase and end_Phase marking the beginning and end of each information transfer phase, respectively, the probes begin_ph signaling the beginning of some particular phase ph, the probes obs_place and obs_read observing the writing and reading of information on/from the data bus, respectively, and the probes obs_setSIG and obs_relSIG indicating the assertion and de-assertion of some signal SIG, respectively. Now we can formalize the desired properties in the modal μ-calculus as follows.

- *Property 1:* This property ensures that the model does not possess undesired livelocks. For each bus phase ph we have to consider the following formula.

$$[-]^{\infty}\langle -\rangle^{*}(\langle \mathtt{begin\_ph}\rangle tt) \ .$$

- *Property 2:* We have to check for every path that probe begin_Phase is eventually followed by probe end_Phase before another begin_Phase is observed.

$$\mathit{fair\text{-}follows}(\mathtt{begin\_Phase}, \mathtt{end\_Phase}, \{\mathtt{obs\_setATN}\}, \langle -\rangle tt) \ .$$

  The implicit fairness constraint ensures that the initiator does not forever ignore the target's wish to enter a new phase by continuously asserting the ATN signal.

- *Property 3:* We have to encode that on all paths the probes obs_setREQ, obs_relREQ, obs_setACK, and obs_relACK do not occur in between end_Phase and begin_Phase by the formula

$$\begin{aligned} \mathit{between}(\ & \mathtt{end\_Phase}, \mathtt{begin\_Phase}, \\ & [\mathtt{obs\_setREQ}, \mathtt{obs\_relREQ}, \mathtt{obs\_setACK}, \mathtt{obs\_relACK}]\mathit{ff} \ ) \ . \end{aligned}$$

- *Property 4:* This formalization can be done along the lines of the one of Property 3.

$$\begin{aligned} \mathit{between}(\ & \mathtt{begin\_Phase}, \mathtt{end\_Phase}, \\ & [\mathtt{obs\_setBSY}, \mathtt{obs\_relBSY}, \mathtt{obs\_setSEL}, \mathtt{obs\_relSEL}]\mathit{ff} \ ) \ . \end{aligned}$$

- *Property 5:* Here, one has to encode that `obs_place` is always followed by `obs_read` in our model. The incorporated fairness constraint corresponds to the one in Property 2.

$$\mathit{fair\_follows}(\texttt{obs\_place}, \texttt{obs\_read}, \{\texttt{obs\_setATN}\}, [\texttt{obs\_place}]\mathit{ff}) \ .$$

- *Property 6:* We have to formalize that every probe `obs_setATN` is always eventually followed by a probe `begin_MsgOut`.

$$\mathit{fair\_follows}(\texttt{obs\_setATN}, \texttt{begin\_MsgOut}, \emptyset, [\texttt{obs\_setATN}]\mathit{ff}) \ .$$

  Note that this property does not require any fairness assumptions.

5.7.4 Verification Results

We have been able to validate each property in our model within at most two minutes when running the `CWB-NC` on a SUN SPARC 20 workstation. The model checker we have used is a local model checker for a fragment of the modal μ-calculus [26]. In Sections 3.6.3 and 3.6.4 we have already commented on this particular model checker and its advantages. Especially, applying a *local* model checker in contrast to a *global* one remarkably speeds-up the task of verification during the initial modeling attempts. In fact, the modeling of the `SCSI-2` bus-protocol has been done in several stages, after each of which the above mentioned properties have been checked. At early modeling stages the model checker has invalidated most properties immediately. The encountered errors have ranged from missed fairness constraints to wrong timing information. However, the diagnostic information in form of failure traces provided by the model checker simplifies the task of finding bugs in models.

During the process of verification, we have also realized that the timing constraints of the bus-protocol are not only imposed for avoiding wire glitches but also in order to implement necessary synchronization constraints during the initial bus-phases. Without these synchronization constraints, two `LUN`s may gain access to the bus for arbitration, which leads to a deadlock. This emphasizes the necessity of dealing with real-time constraints in reactive systems.

5.8 Summary

In this chapter we have used the process algebras $\mathsf{CCS}^{\mathsf{dp}}$ and $\mathsf{CCS}^{\mathsf{rt}}$ to formally model and reason about the `SCSI-2` bus-protocol. The size of our model is almost an order of magnitude smaller when constructed with $\mathsf{CCS}^{\mathsf{dp}}$ instead of $\mathsf{CCS}^{\mathsf{rt}}$ semantics and could be handled easily by the `CWB-NC`, in which both process algebras have been implemented as front-ends. This emphasizes the utility of our dynamic-priority approach for implementing real-time and justifies the effort we have made in Chapter 4. In addition, we have specified several desired properties of the bus-protocol in the modal μ-calculus and validated them by using model checking.

Chapter 6

A Process Algebra with Distributed Priority

6.1 Introduction

In this chapter we develop an algebraic theory of action priority which is suitable for modeling and reasoning about *distributed* systems. As for the process algebra $\mathsf{CCS}^{\mathsf{ch}}$ presented in Chapter 2 and other existing approaches to priority, our aim is to model systems in which some transitions may have precedence over others. Our point of departure is that the priority-scheme should be localized within individual sites in the system; high prioritized internal transitions should only be able to pre-empt lower prioritized transitions being performed at the "same location." This constraint reflects an essential intuition about distributed systems, namely, that the execution of a process on one processor should not affect the behavior of a process on another processor unless the designer explicitly builds in an interaction (e.g. synchronization) between them. Technically, we begin with a theory of priority that includes a notion of global precedence, similar to the one introduced in Chapter 2, and show how its semantics may be altered to localize capabilities for pre-emption by using locations [131]. We then define a semantic theory for this language, referred to as $\mathsf{CCS}^{\mathsf{prio}}$, based on Milner and Park's notion of bisimulation [125, 139]: we present a strong congruence, axiomatize it for finite processes, and derive an observational congruence along the lines of [125]. Moreover, we discuss how some design decisions adopted from Chapter 2 may be lifted, thus leading to more "general" versions of $\mathsf{CCS}^{\mathsf{prio}}$, and show that Camilleri and Winskel's process algebra with priority [43] is a sub-algebra in a specific "generalization" of $\mathsf{CCS}^{\mathsf{prio}}$.

Our semantic framework for $\mathsf{CCS}^{\mathsf{prio}}$ is based on traditional CCS [125]; in particular, we "reduce" concurrency to nondeterminism using *interleaving*. Other semantic theories of concurrency [171] treat parallelism as a primitive notion; such "truly concurrent" frameworks include *causal approaches* — e.g. *partial order semantics* [73, 74, 168], *event structures* [137], *proved transitions* [32], and *Mazurkiewicz traces* [122] — and *location semantics* [33, 80, 131, 132]. As these theories make concurrency explicit, they may be

seen as natural bases for modeling distributed systems. However, in making concurrency generally observable, the frameworks become technically more complex than traditional interleaving-based ones. Our aim in this chapter is to examine the extent to which one aspect of distribution — namely, priority localization — may be formalized within an interleaved theory of concurrency. Consequently, we start with an interleaving semantics and introduce only as much sensitivity to "distribution" as our local view of pre-emption requires.

The remainder of the chapter is organized as follows. In the next section we present a generic example illustrating the need for local pre-emption in modeling distributed systems. Sections 6.3–6.5 present our calculus and derive the technical results discussed above, while Section 6.6 contains an example showing the application of our theory. The following section discusses some alternatives to our formulation, one of which is formally related to a calculus with priority developed by Camilleri and Winskel [43]. Section 6.8 focuses on related work, and the last section presents our summary. Appendix C contains the proof of an important technical result, which has been left out in this chapter due to its length.

6.2 Motivating Example

The example depicted in Figure 6.1 motivates the need for considering a local notion of pre-emption when dealing with priority in distributed systems. The system consists of two sites (computers), `Site1` and `Site2`, that are connected via the network `Network`. Each site runs an application, `Application1` and `Application2`, respectively, which may send or receive information from the application at the other site via its (interrupt-)handler, `Handler1` or `Handler2`. A handler delivers the message to the network or receives a message for its site from the network and notifies the application by sending an interrupt.

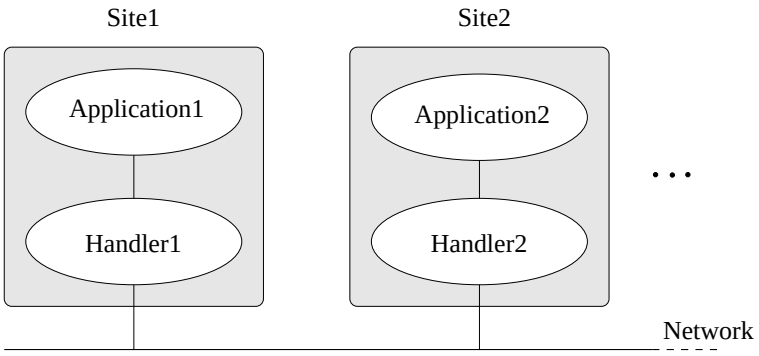

Figure 6.1: A distributed system

We have the following intuitive requirements that the semantics of a design language should satisfy in order to reflect the behavior of the system correctly. First, an interrupt of a handler should pre-empt the normal work of the application at its site, i.e. the application should immediately respond to an interrupt request. Second, both sites should be

able to perform internal computation that are local to their site without interference from the other site. In particular, internal activities of `Handler1` should not pre-empt those of `Handler2`, and vice versa. While traditional process-algebraic treatments of priority, e.g. the approaches considered so far and those reported in [54, 134], satisfy the first requirement, they typically violate the second, since they allow an action of `Application1` to pre-empt an action of `Application2` if the former has higher priority, even though they are performed on different sites. In general, one would expect priorities at different sites to be incomparable. The semantics given in Chapter 2 and in [54, 134], however, do not permit this distinction to be made; the net effect is that some computations that one would expect to find in a distributed system are improperly suppressed. We propose to remedy this shortcoming regarding distributed systems by introducing a notion of *local pre-emption*.

6.3 Syntax and Semantics of CCS^{prio}

In this section we define the syntax and semantics of our calculus CCS^{prio}, which is based on CCS [125].

6.3.1 Syntax of CCS^{prio}

The syntax of CCS^{prio} differs from CCS in that the action set exhibits a priority-scheme, i.e. priorities are assigned to actions. It is important to note that an action may have different priorities in different states of the system under consideration. This property of priorities is called *globally dynamic* in [154]. For the sake of simplicity, we restrict ourselves to a two-level priority-scheme. In Section 6.7, we discuss how our results presented in this chapter can be adapted to multi-level priority-schemes. Intuitively, actions represent potential synchronizations that a process may be willing to engage in with its environment. Given a choice between a synchronization on a high priority action and one on a low priority action, a process should choose the former.

Formally, let Λ be a countably infinite set of action labels, not including the *internal* or *silent* action τ. For every *input action* $a \in \Lambda$, there exists a *complementary action* $\overline{a}$, the corresponding *output action*. Let $\overline{\Lambda} =_{\mathrm{df}} \{\overline{a} \mid a \in \Lambda\}$, and let $A = \Lambda \cup \overline{\Lambda} \cup \{\tau\}$, where $\tau \notin \Lambda \cup \overline{\Lambda}$, denote the set of all unprioritized actions. Intuitively, an action constitutes a willingness to perform a synchronization on the *port* associated with the action name. Thus a process that wishes to perform action a is willing to receive a message on port a, whereas a process that wishes to engage in $\overline{a}$ may send a message via port a. The action τ represents either an internal action of a process or the synchronization of two processes on some port in order to communicate with each other. We use $a, b, \ldots$ to range over $\Lambda \cup \overline{\Lambda}$ and $\alpha, \beta, \ldots$ to range over A.

In order to define *prioritized actions*, let $\underline{\Lambda}$ be a countably infinite set of prioritized action labels disjoint from Λ. Then $\underline{A} =_{\mathrm{df}} \underline{\Lambda} \cup \underline{\overline{\Lambda}} \cup \{\underline{\tau}\}$ is the set of *prioritized actions*, where $\underline{\tau} \notin A \cup \underline{\Lambda} \cup \underline{\overline{\Lambda}}$ is the prioritized *internal* or *silent* action. We use $\mathcal{A} =_{\mathrm{df}} A \cup \underline{A}$ to denote the set of all actions. Intuitively, prioritized actions represent communication potentials

over "important" channels. Therefore, communications involving prioritized actions should be preferred over communications involving unprioritized actions. In the remainder of the chapter, we let $\underline{a}, \underline{b}, \ldots$ range over $\underline{\Lambda} \cup \overline{\underline{\Lambda}}$, $\underline{\alpha}, \underline{\beta}, \ldots$ over $\underline{A}$, and γ over $\mathcal{A}$. Additionally, we extend $\bar{}$ by defining $\overline{\overline{\gamma}} =_{\mathrm{df}} \gamma$, and if $L \subseteq \mathcal{A} \setminus \{\tau, \underline{\tau}\}$ then $\overline{L} =_{\mathrm{df}} \{\overline{\gamma} \mid \gamma \in L\}$. A mapping f on $\mathcal{A}$ is a finite *relabeling* if f preserves priorities (i.e. $f(\Lambda) \subseteq \Lambda$ and $f(\underline{\Lambda}) \subseteq \underline{\Lambda}$), is such that the set $\{\gamma \mid f(\gamma) \neq \gamma\}$ is finite, and satisfies the following: $f(\overline{a}) = \overline{f(a)}$, $f(\overline{\underline{a}}) = \overline{f(\underline{a})}$, $f(\tau) = \tau$, and $f(\underline{\tau}) = \underline{\tau}$. The syntax of our calculus may now be defined by the BNF

$$P \quad ::= \quad \mathbf{0} \quad | \quad x \quad | \quad \gamma.P \quad | \quad P + P \quad | \quad P|P \quad | \quad P[f] \quad | \quad P \setminus L \quad | \quad \mu x.P$$

where f is a finite relabeling, $L \subseteq \mathcal{A} \setminus \{\tau, \underline{\tau}\}$ is a *restriction set*, and x is a process variable taken from a countable set $\mathcal{V}$ of variables. We use the standard definitions for *sort* of a process, *free* and *bound variables*, *open* and *closed terms*, *guarded recursion*, and *contexts*. In the remainder, the syntactic substitution of x by Q in term P is written as $P[Q/x]$. We refer to closed and guarded terms as *processes* and denote syntactic equality by $\equiv$. Moreover, we let $P, Q, R, \ldots$ range over the set $\mathcal{P}$ of processes. Finally, sometimes it is convenient to write $C \stackrel{\mathrm{def}}{=} P$ for the process $\mu C.P$, where the identifier C is interpreted as variable.

6.3.2 Locations

We now introduce the notion of *location*, which will be used in the next section in the operational semantics for $\mathsf{CCS}^{\mathsf{prio}}$ as a basis for deciding when one transition pre-empts another. Intuitively, a location represents the "address(es)" of subterm(s) inside a larger term; when a system performs an action, our semantics will also note the location of the subterm(s) that "generate(s)" this action. Observe that because of the potential for synchronization more than one subterm may be involved in an action. Our account of locations closely follows that of [131].

Formally, let $\mathcal{A}_{\mathrm{addr}} =_{\mathrm{df}} \{L, R, l, r\}$ be the *address alphabet*, and let $\bullet$ be a special symbol not in $\mathcal{A}_{\mathrm{addr}}$. Then $\mathcal{A}ddr =_{\mathrm{df}} \{\bullet s \mid s \in \mathcal{A}^*_{\mathrm{addr}}\}$ represents the set of (process) *addresses* ranged over by v, w. We abuse notation by omitting $\bullet$ from addresses on occasion. If $\bullet s_1$ and $\bullet s_2$ are addresses then we write $\bullet s_1 \cdot \bullet s_2 = \bullet s_1 s_2$ to represent address concatenation (where $s_1 s_2$ represents the usual concatenation of elements in $\mathcal{A}^*_{\mathrm{addr}}$). Further, if $V \subseteq \mathcal{A}ddr$ and $\zeta \in \mathcal{A}_{\mathrm{addr}}$ then we write $V \cdot \zeta$ for $\{v \cdot \zeta \mid v \in V\}$. Intuitively, an element of $\mathcal{A}ddr$ represents the address of a subterm, with $\bullet$ denoting the current term, l (r) representing the left (right) subterm of $+$, and L (R) the left (right) subterm of $|$. For example, in the process $(a.\mathbf{0} \mid b.\mathbf{0}) + c.\mathbf{0}$ the address of $a.\mathbf{0}$ is $\bullet Ll$, of $b.\mathbf{0}$ is $\bullet Rl$, and of $c.\mathbf{0}$ is $\bullet r$.

As mentioned in the introduction, we want to adopt the view that processes on different sides of the parallel composition operator are logically – not necessarily physically – executed on different processors. Thus, priorities on different sides of the parallel composition operator are distributed and, therefore, should be incomparable. However, priorities on different sides of the summation operator, which models nondeterministic choice, should be comparable since argument processes of summation are logically scheduled on the same

processor. We formalize this intuition in the following *comparability relation* on addresses which is adapted from [89].

Definition 6.3.1 (Comparability Relation)
The comparability relation $\bowtie$ on addresses is the smallest reflexive and symmetric subset of $\mathcal{A}ddr \times \mathcal{A}ddr$ such that for all $v, w \in \mathcal{A}ddr$:

1. $\langle v \cdot l, w \cdot r \rangle \in \bowtie$, *and*
2. $\langle v, w \rangle \in \bowtie$ *implies* $\langle v \cdot \zeta, w \cdot \zeta \rangle \in \bowtie$ *for* $\zeta \in \mathcal{A}_{addr}$.

In the sequel we write $v \bowtie w$ instead of $\langle v, w \rangle \in \bowtie$. If $v \in \mathcal{A}ddr$ then we use $[v]$ to denote the set $\{w \in \mathcal{A}ddr \mid v \bowtie w\}$. Note that the comparability relation is not transitive, e.g. we have $Ll \bowtie r$ and $r \bowtie Rl$ but $Ll \not\bowtie Rl$, since $L \not\bowtie R$. Considering our example $(a.\mathbf{0} \mid b.\mathbf{0}) + c.\mathbf{0}$ above, the addresses of $a.\mathbf{0}$ and $c.\mathbf{0}$, and the addresses of $b.\mathbf{0}$ and $c.\mathbf{0}$, are comparable since they are on different sides of the summation operator. In contrast, the addresses of $a.\mathbf{0}$ and $b.\mathbf{0}$ are incomparable since they are on different sides of the parallel composition operator.

We may now define the set of (transition) *locations*, $\mathcal{L}oc$, as $\mathcal{L}oc =_{\mathrm{df}} \mathcal{A}ddr \cup (\mathcal{A}ddr \times \mathcal{A}ddr)$. Intuitively, a transition location records the addresses of the components in a term that participate in the execution of a given action. In our algebra, transitions are performed by single processes or pairs of processes (in the case of a synchronization). We define $\langle v, w \rangle \cdot \zeta =_{\mathrm{df}} \langle v \cdot \zeta, w \cdot \zeta \rangle$ and $[\langle v, w \rangle] =_{\mathrm{df}} [v] \cup [w]$ where $v, w \in \mathcal{A}ddr$ and $\zeta \in \mathcal{A}_{\mathrm{addr}}$. We use $m, n, o, \ldots$ to range over $\mathcal{L}oc$ in what follows.

6.3.3 Semantics of CCS^{prio}

The (*operational*) *semantics* of a CCS^{prio} process $P \in \mathcal{P}$ is given by a labeled transition system $\langle \mathcal{P}, \mathcal{L}oc \times \mathcal{A}, \longrightarrow, P \rangle$. The transition relation $\longrightarrow \subseteq \mathcal{P} \times (\mathcal{L}oc \times \mathcal{A}) \times \mathcal{P}$ is defined in Tables 6.2 and 6.3 using Plotkin-style operational rules [141]. We write $P \xrightarrow{m,\gamma} P'$ instead of $\langle P, \langle m, \gamma \rangle, P' \rangle \in \longrightarrow$ and say that P *may engage in action γ offered from location m and thereafter behave like process P'*. We also write $P \xrightarrow{\gamma} P'$ if there exists a location $m \in \mathcal{L}oc$ such that $P \xrightarrow{m,\gamma} P'$.

The presentation of the operational rules requires *distributed prioritized initial action sets*, which are defined as the least sets satisfying the rules in Table 6.1. Intuitively, $\underline{\mathcal{I}}_m(P)$ denotes the set of all distributed prioritized initial actions of P from location m. Note that these sets are either empty or contain exactly one initial action. $\underline{\mathcal{I}}_m(P) = \emptyset$ means that either m is not a location of P or P is incapable of performing a prioritized action at location m. Additionally, let us denote the set $\bigcup\{\underline{\mathcal{I}}_m(P) \mid m \in M\}$ of all distributed prioritized initial actions of process P from locations $M \subseteq \mathcal{L}oc$ by $\underline{\mathcal{I}}_M(P)$ and the set $\underline{\mathcal{I}}_{\mathcal{L}oc}(P)$ of all distributed prioritized initial actions of process P by $\underline{\mathcal{I}}(P)$. We also define analogous sets restricted to visible actions: $\underline{\mathcal{I}\!\!\mathcal{I}}_M(P) =_{\mathrm{df}} \underline{\mathcal{I}}_M(P) \setminus \{\underline{\tau}\}$ and $\underline{\mathcal{I}\!\!\mathcal{I}}(P) =_{\mathrm{df}} \underline{\mathcal{I}}(P) \setminus \{\underline{\tau}\}$, respectively. The initial action sets satisfy the following properties with respect to the summation operator, which will be useful in the remainder.

Table 6.1: Distributed prioritized initial action sets for $\mathsf{CCS}^{\mathsf{prio}}$

$\underline{\mathcal{I}}_m(\mu x.P) =_{\mathrm{df}} \underline{\mathcal{I}}_m(P[\mu x.P/x])$	$\underline{\mathcal{I}}_\bullet(\underline{\alpha}.P) =_{\mathrm{df}} \{\underline{\alpha}\}$
$\underline{\mathcal{I}}_{m\cdot l}(P+Q) =_{\mathrm{df}} \underline{\mathcal{I}}_m(P)$	$\underline{\mathcal{I}}_{n\cdot r}(P+Q) =_{\mathrm{df}} \underline{\mathcal{I}}_n(Q)$
$\underline{\mathcal{I}}_m(P[f]) =_{\mathrm{df}} \{f(\underline{\alpha}) \mid \underline{\alpha} \in \underline{\mathcal{I}}_m(P)\}$	$\underline{\mathcal{I}}_m(P \setminus L) =_{\mathrm{df}} \underline{\mathcal{I}}_m(P) \setminus (L \cup \overline{L})$
$\underline{\mathcal{I}}_{m\cdot L}(P\|Q) =_{\mathrm{df}} \underline{\mathcal{I}}_m(P)$	$\underline{\mathcal{I}}_{n\cdot R}(P\|Q) =_{\mathrm{df}} \underline{\mathcal{I}}_n(Q)$
$\underline{\mathcal{I}}_{\langle m\cdot L, n\cdot R\rangle}(P\|Q) =_{\mathrm{df}} \{\underline{\tau} \mid \underline{\mathcal{I}}_m(P) \cap \overline{\underline{\mathcal{I}}_n(Q)} \neq \emptyset\}$	

Lemma 6.3.2 *Let $P, Q \in \mathcal{P}$ and $m, n \in \mathcal{L}oc$. Then the following properties hold for all $R \in \mathcal{P}$ and $o \in \mathcal{L}oc$.*

1. $\underline{\Pi}_{[n]}(Q) \subseteq \underline{\Pi}_{[m]}(P)$ *implies* $\underline{\Pi}_{[n\cdot l]}(Q+R) \subseteq \underline{\Pi}_{[m\cdot l]}(P+R)$
2. $\underline{\Pi}(Q) \subseteq \underline{\Pi}(P)$ *implies* $\underline{\Pi}_{[o\cdot r]}(Q+R) \subseteq \underline{\Pi}_{[o\cdot r]}(P+R)$
3. $\underline{\Pi}(Q) \subseteq \underline{\Pi}(P)$ *implies* $\underline{\Pi}(Q+R) \subseteq \underline{\Pi}(P+R)$.

The proof of this lemma is straightforward by using the definition of $\underline{\Pi}_{[\cdot]}(\cdot)$ together with Definition 6.3.1, and the definition of $\underline{\Pi}(\cdot)$. With respect to parallel composition we obtain an analogous lemma.

Lemma 6.3.3 *Let $P, Q, R \in \mathcal{P}$ and $m, n, o \in \mathcal{L}oc$. Then the following properties hold.*

1. $\underline{\Pi}_{[n]}(Q) \subseteq \underline{\Pi}_{[m]}(P)$ *implies* $\underline{\Pi}_{[n\cdot L]}(Q \mid R) \subseteq \underline{\Pi}_{[m\cdot L]}(P \mid R)$
2. $\underline{\Pi}_{[o\cdot R]}(Q \mid R) = \underline{\Pi}_{[o\cdot R]}(P \mid R)$
3. $\underline{\Pi}_{[n]}(Q) \subseteq \underline{\Pi}_{[m]}(P)$ *implies* $\underline{\Pi}_{[\langle n\cdot L, o\cdot R\rangle]}(Q \mid R) \subseteq \underline{\Pi}_{[\langle m\cdot L, o\cdot R\rangle]}(P \mid R)$
4. $\underline{\Pi}(Q) \subseteq \underline{\Pi}(P)$ *implies* $\underline{\Pi}(Q \mid R) \subseteq \underline{\Pi}(P \mid R)$

The proof can be done in a similar fashion to the proof of Lemma 6.3.2. For the semantics of $\mathsf{CCS}^{\mathsf{prio}}$ observe that the initial action sets are defined independently from the transition relation $\longrightarrow$. Therefore, $\longrightarrow$ is well-defined (cf. [167]). The side conditions of the operational semantic rules guarantee that a process does not perform an unprioritized action if

Table 6.2: Operational semantics for CCS^{prio} (Part I)

Act	$\dfrac{-}{\underline{\alpha}.P \xrightarrow{\bullet,\underline{\alpha}} P}$	Act	$\dfrac{-}{\alpha.P \xrightarrow{\bullet,\alpha} P}$
Sum1	$\dfrac{P \xrightarrow{m,\underline{\alpha}} P'}{P+Q \xrightarrow{m\cdot l,\underline{\alpha}} P'}$	Sum1	$\dfrac{P \xrightarrow{m,\alpha} P'}{P+Q \xrightarrow{m\cdot l,\alpha} P'}\ \underline{\tau} \notin \underline{\mathcal{I}}(Q)$
Sum2	$\dfrac{Q \xrightarrow{n,\underline{\alpha}} Q'}{P+Q \xrightarrow{n\cdot r,\underline{\alpha}} Q'}$	Sum2	$\dfrac{Q \xrightarrow{n,\alpha} Q'}{P+Q \xrightarrow{n\cdot r,\alpha} Q'}\ \underline{\tau} \notin \underline{\mathcal{I}}(P)$
Rel	$\dfrac{P \xrightarrow{m,\underline{\alpha}} P'}{P[f] \xrightarrow{m,f(\underline{\alpha})} P'[f]}$	Rel	$\dfrac{P \xrightarrow{m,\alpha} P'}{P[f] \xrightarrow{m,f(\alpha)} P'[f]}$
Res	$\dfrac{P \xrightarrow{m,\underline{\alpha}} P'}{P \setminus L \xrightarrow{m,\underline{\alpha}} P' \setminus L}\ \underline{\alpha} \notin L \cup \overline{L}$	Res	$\dfrac{P \xrightarrow{m,\alpha} P'}{P \setminus L \xrightarrow{m,\alpha} P' \setminus L}\ \alpha \notin L \cup \overline{L}$
Rec	$\dfrac{P[\mu x.P/x] \xrightarrow{m,\underline{\alpha}} P'}{\mu x.P \xrightarrow{m,\underline{\alpha}} P'}$	Rec	$\dfrac{P[\mu x.P/x] \xrightarrow{m,\alpha} P'}{\mu x.P \xrightarrow{m,\alpha} P'}$

Table 6.3: Operational semantics for CCS^{prio} (Part II)

Com1	$\dfrac{P \xrightarrow{m,\underline{\alpha}} P'}{P\|Q \xrightarrow{m\cdot L,\underline{\alpha}} P'\|Q}$	Com1	$\dfrac{P \xrightarrow{m,\alpha} P'}{P\|Q \xrightarrow{m\cdot L,\alpha} P'\|Q}\ \underline{\mathcal{I}}_{[m]}(P) \cap \overline{\underline{\mathcal{I}}(Q)} = \emptyset$
Com2	$\dfrac{Q \xrightarrow{n,\underline{\alpha}} Q'}{P\|Q \xrightarrow{n\cdot R,\underline{\alpha}} P\|Q'}$	Com2	$\dfrac{Q \xrightarrow{n,\alpha} Q'}{P\|Q \xrightarrow{n\cdot R,\alpha} P\|Q'}\ \underline{\mathcal{I}}_{[n]}(Q) \cap \overline{\underline{\mathcal{I}}(P)} = \emptyset$
Com3	$\dfrac{P \xrightarrow{m,\underline{a}} P' \quad Q \xrightarrow{n,\underline{\overline{a}}} Q'}{P\|Q \xrightarrow{\langle m\cdot L, n\cdot R\rangle,\underline{\tau}} P'\|Q'}$	Com3	$\dfrac{P \xrightarrow{m,a} P' \quad Q \xrightarrow{n,\overline{a}} Q'}{P\|Q \xrightarrow{\langle m\cdot L, n\cdot R\rangle,\tau} P'\|Q'}\ \begin{array}{l}\underline{\mathcal{I}}_{[m]}(P) \cap \overline{\underline{\mathcal{I}}(Q)} = \emptyset \wedge \\ \underline{\mathcal{I}}_{[n]}(Q) \cap \overline{\underline{\mathcal{I}}(P)} = \emptyset\end{array}$

it can engage in a prioritized synchronization or internal computation, i.e. a $\underline{\tau}$-transition, from a comparable location. In contrast to the global notion of pre-emption defined in Chapter 2 and in [54, 133], the local notion here is much weaker since $\underline{\mathcal{I}}_{[m]}(P) \subseteq \underline{\mathcal{I}}(P)$ holds for all $m \in \mathcal{L}oc$ and $P \in \mathcal{P}$. In other words, local pre-emption does not pre-empt as many unprioritized transitions as global pre-emption does.

The semantics of $\mathsf{CCS}^{\mathsf{prio}}$ for prioritized transitions is the same as the usual CCS semantics. The difference arises by the side conditions of the rules for summation and parallel composition with respect to unprioritized transitions. The process $\gamma.P$ may engage in action γ and then behaves like P. The *summation operator* $+$ denotes *nondeterministic choice*. The process $P + Q$ may behave like process P (Q) if Q (P) does not pre-empt unprioritized actions by being able to perform a $\underline{\tau}$-transition. Note that priorities arising from different sides of the summation operator are comparable. The *restriction operator* $\backslash L$ prohibits the execution of actions in $L \cup \overline{L}$. Thus, it permits the scoping of actions. $P[f]$ behaves exactly as the process P with actions renamed according to the *relabeling* f. The process $P \,|\, Q$ stands for the *interleaved parallel composition* of P and Q with synchronized communication on complementary actions resulting in the internal action τ or $\underline{\tau}$. Since locations on different sides of a parallel operator are incomparable, $\underline{\tau}$'s arising from a location of P (Q) cannot pre-empt the execution of a transition, even an unprioritized one, of Q (P). Only if P (Q) engages in a prioritized synchronization with Q (P) can unprioritized actions from a comparable location of P (Q) be pre-empted. For example, the initial a-transition gets pre-empted in the process $(a.\mathbf{0} + \underline{b}.\mathbf{0}) \,|\, \overline{\underline{b}}.\mathbf{0}$ but not in the process $(a.\mathbf{0} \,|\, \underline{b}.\mathbf{0}) \,|\, \overline{\underline{b}}.\mathbf{0}$. Also observe that, as in the algebra $\mathsf{CCS}^{\mathsf{ch}}$, actions a and $\overline{\underline{a}}$ cannot synchronize. Thus, one may view the sets of prioritized and unprioritized action labels as being disjoint. Finally, $\mu x.P$ denotes a *recursively defined* process that behaves as a distinguished solution of the equation $x = P$.

6.4 A Semantic Theory based on Strong Bisimulation

In this section we present an equivalence relation for $\mathsf{CCS}^{\mathsf{prio}}$ processes that is based on bisimulation [139]. Our aim is to characterize the largest congruence contained in the "naive" adaptation of strong bisimulation [125] to our framework obtained by ignoring location information.

Definition 6.4.1 (Naive Distributed Prioritized Strong Bisimulation)
A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is called naive distributed prioritized strong bisimulation *if for every $\langle P, Q\rangle \in \mathcal{R}$ and $\gamma \in \mathcal{A}$ the following condition holds.*

$$P \stackrel{\gamma}{\rightarrow} P' \text{ implies } \exists Q'.\, Q \stackrel{\gamma}{\rightarrow} Q' \text{ and } \langle P', Q'\rangle \in \mathcal{R} \ .$$

We write $P \simeq Q$ if there exists a naive distributed prioritized strong bisimulation $\mathcal{R}$ such that $\langle P, Q\rangle \in \mathcal{R}$.

It is straightforward to establish that $\simeq$ is the *largest* naive distributed prioritized strong bisimulation and that $\simeq$ is an equivalence. Unfortunately, $\simeq$ is *not* a congruence, which is a necessary requirement for an equivalence to be suitable for compositional reasoning. The lack of compositionality is demonstrated by the following example, which embodies the traditional view that "parallelism = nondeterminism." We have $a.\underline{b}.\mathbf{0} + \underline{b}.a.\mathbf{0} \simeq a.\mathbf{0} \,|\, \underline{b}.\mathbf{0}$ but $(a.\underline{b}.\mathbf{0} + \underline{b}.a.\mathbf{0}) \,|\, \overline{\underline{b}}.\mathbf{0} \not\simeq (a.\mathbf{0} \,|\, \underline{b}.\mathbf{0}) \,|\, \overline{\underline{b}}.\mathbf{0}$, since the latter process can perform an a-transition

while the corresponding a-transition of the former is pre-empted because the right process in the summation can engage in a prioritized communication. The above observation is not surprising since the distribution of processes influences the pre-emption of transitions and, consequently, the bisimulation. However, we know by Proposition 2.4.3 that $\simeq$ includes a largest congruence $\simeq^+$ for $\mathsf{CCS}^{\mathsf{prio}}$.

6.4.1 Distributed Prioritized Strong Bisimulation

In the remainder, we develop a characterization of $\simeq^+$. To do so we need to take the *local* pre-emption of processes into account.

Definition 6.4.2 (Distributed Prioritized Strong Bisimulation)
A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a distributed prioritized strong bisimulation *if for every $\langle P, Q\rangle \in \mathcal{R}$, $\alpha \in A$, $\underline{\alpha} \in \underline{A}$, and $m \in \mathcal{L}oc$ the following conditions hold.*

1. *$P \stackrel{\underline{\alpha}}{\Rightarrow} P'$ implies $\exists Q'.\, Q \stackrel{\underline{\alpha}}{\Rightarrow} Q'$ and $\langle P', Q'\rangle \in \mathcal{R}$.*

2. *$P \stackrel{m,\alpha}{\longrightarrow} P'$ implies $\exists Q', n.\, Q \stackrel{n,\alpha}{\longrightarrow} Q', \underline{\mathcal{I}}_{[n]}(Q) \subseteq \underline{\mathcal{I}}_{[m]}(P)$, and $\langle P', Q'\rangle \in \mathcal{R}$.*

We write $P \simeq^{\mathrm{l}} Q$ if there exists a distributed prioritized strong bisimulation $\mathcal{R}$ such that $\langle P, Q\rangle \in \mathcal{R}$.

The difference between this definition and that of $\simeq$ is the additional requirement concerning the initial action sets, parameterized with the appropriate locations, in the condition for unprioritized transitions. Intuitively, the distributed prioritized initial action set of a process with respect to some location is a measure of the pre-emptive power of the process relative to that location. Thus, the second condition of Definition 6.4.2 states that an unprioritized action α from some location m of the process P must be matched by the same action from some location n of Q and that the pre-emptive power of Q relative to n is at most as strong as the pre-emptive power of P relative to m. The following straightforward, technical lemma is useful in the remainder of this chapter.

Lemma 6.4.3 *For $P, Q \in \mathcal{P}$ such that $P \simeq^{\mathrm{l}} Q$ we have $\underline{\mathcal{I}}(P) = \underline{\mathcal{I}}(Q)$ and $\underline{\mathcal{II}}(P) = \underline{\mathcal{II}}(Q)$.*

Now we can prove the desired congruence result.

Proposition 6.4.4 *The relation $\simeq^{\mathrm{l}}$ is a congruence, i.e. for all $\mathsf{CCS}^{\mathsf{prio}}$ contexts $C[X]$ we have: $P \simeq^{\mathrm{l}} Q$ implies $C[P] \simeq^{\mathrm{l}} C[Q]$.*

Proof: Most cases of the proof are straightforward (cf. [125]). Therefore, we restrict ourselves to the more interesting cases, namely the compositionality of $\simeq^{\mathrm{l}}$ with respect to summation and parallel composition. Therefore, let $P, Q, R \in \mathcal{P}$ such that $P \simeq^{\mathrm{l}} Q$.

1. We first show $P + R \simeq^{\mathrm{l}} Q + R$. According to Definition 6.4.2 it is sufficient to prove that

$$\mathcal{R} =_{\mathrm{df}} \{\langle P + R, Q + R\rangle \mid P \simeq^{\mathrm{l}} Q\} \cup \simeq^{\mathrm{l}}$$

is a distributed prioritized strong bisimulation. Let $P + R \xrightarrow{p,\alpha} V$ for some $V \in \mathcal{P}$ and $p \in \mathcal{L}oc$. We have to show the existence of some $W \in \mathcal{P}$ and $q \in \mathcal{L}oc$ such that $Q + R \xrightarrow{q,\alpha} W$ and $\langle V, W\rangle \in \mathcal{R}$. Therefore, consider the following case distinction according to the operational rules for summation.

- **Case 1:** Let $V \equiv P'$ and $p \equiv m \cdot l$. Here, we conclude

		$P + R \xrightarrow{m\cdot l,\alpha} P'$
(Rule (Sum1))	implies	$P \xrightarrow{m,\alpha} P'$ and $\underline{\tau} \notin \underline{\mathcal{I}}(R)$
$(P \simeq^{\mathrm{l}} Q)$	implies	$\exists Q', n.\, Q \xrightarrow{n,\alpha} Q'$, $P' \simeq^{\mathrm{l}} Q'$, $\underline{\mathcal{I}\!\mathcal{I}}_{[n]}(Q) \subseteq \underline{\mathcal{I}\!\mathcal{I}}_{[m]}(P)$, and $\underline{\tau} \notin \underline{\mathcal{I}}(R)$
(Rule (Sum1), def. of $\mathcal{R}$, Lemma 6.3.2(1))	implies	$\exists Q', n.\, Q + R \xrightarrow{n\cdot l,\alpha} Q'$, $\langle P', Q'\rangle \in \mathcal{R}$, and $\underline{\mathcal{I}\!\mathcal{I}}_{[n\cdot l]}(Q + R) \subseteq \underline{\mathcal{I}\!\mathcal{I}}_{[m\cdot l]}(P + R)$.

  Now, we finish this proof part by choosing $W \equiv Q'$ and $q \equiv n \cdot l$.

- **Case 2:** Let $V \equiv R'$ and $p \equiv o \cdot r$. In this case, we observe

		$P + R \xrightarrow{o\cdot r,\alpha} R'$
(Rule (Sum2))	implies	$R \xrightarrow{o,\alpha} R'$ and $\underline{\tau} \notin \underline{\mathcal{I}}(P)$
(Lemma 6.4.3)	implies	$R \xrightarrow{o,\alpha} R'$, $\underline{\tau} \notin \underline{\mathcal{I}}(Q)$, and $R' \simeq^{\mathrm{l}} R'$
(Rule (Sum2), def. of $\mathcal{R}$, Lemmata 6.3.2(1) and 6.4.3)	implies	$Q + R \xrightarrow{o\cdot r,\alpha} R'$, $\langle R', R'\rangle \in \mathcal{R}$, and $\underline{\mathcal{I}\!\mathcal{I}}_{[o\cdot r]}(Q + R) = \underline{\mathcal{I}\!\mathcal{I}}_{[o\cdot r]}(P + R)$.

  By choosing $W \equiv R'$ and $q \equiv o \cdot r$, the proof of this case is done.

The above argument is sufficient to prove the second condition of Definition 6.4.2. The proof of the first condition is standard since the semantics of the summation operator coincides with the one of CCS for prioritized actions.

2. We have to establish $P \mid R \simeq^{\mathrm{l}} Q \mid R$. According to Definition 6.4.2 it is sufficient to prove that

$$\mathcal{R} =_{\mathrm{df}} \{\langle P \mid R, Q \mid R\rangle \mid P \simeq^{\mathrm{l}} Q\}$$

is a distributed prioritized strong bisimulation. Therefore, let $P \mid R \xrightarrow{p,\alpha} V$ for some $V \in \mathcal{P}$ and $p \in \mathcal{L}oc$. We show the existence of some $W \in \mathcal{P}$ and $q \in \mathcal{L}oc$ such that $Q \mid R \xrightarrow{q,\alpha} W$, $\underline{\mathcal{I}\!\mathcal{I}}_{[q]}(Q \mid R) \subseteq \underline{\mathcal{I}\!\mathcal{I}}_{[p]}(P \mid R)$, and $\langle V, W\rangle \in \mathcal{R}$. Consider the following case distinction according to the operational rules for parallel composition.

- **Case 1:** Let $V \equiv P' \mid R$ and $p \equiv m \cdot L$. We conclude

		$P \mid R \xrightarrow{m \cdot L, \alpha} P' \mid R$
(Rule (Com1))	implies	$P \xrightarrow{m,\alpha} P'$ and $\underline{\mathit{fl}}_{[m]}(P) \cap \overline{\underline{\mathit{fl}}(R)} = \emptyset$
$(P \simeq^{\mathrm{l}} Q)$	implies	$\exists Q', n.\, Q \xrightarrow{n,\alpha} Q', P' \simeq^{\mathrm{l}} Q',$
		$\underline{\mathit{fl}}_{[n]}(Q) \subseteq \underline{\mathit{fl}}_{[m]}(P),$ and
		$\underline{\mathit{fl}}_{[m]}(P) \cap \overline{\underline{\mathit{fl}}(R)} = \emptyset$
	implies	$\exists Q', n.\, Q \xrightarrow{n,\alpha} Q', P' \simeq^{\mathrm{l}} Q',$
		$\underline{\mathit{fl}}_{[n]}(Q) \subseteq \underline{\mathit{fl}}_{[m]}(P),$ and
		$\underline{\mathit{fl}}_{[n]}(Q) \cap \overline{\underline{\mathit{fl}}(R)} = \emptyset$
(Rule (Com1),	implies	$\exists Q' \mid R, n.\, Q \mid R \xrightarrow{n \cdot L, \alpha} Q' \mid R,$
def. of $\mathcal{R}$,		$\langle P' \mid R, Q' \mid R \rangle \in \mathcal{R},$ and
Lemma 6.3.3(1))		$\underline{\mathit{fl}}_{[n \cdot L]}(Q \mid R) \subseteq \underline{\mathit{fl}}_{[m \cdot L]}(P \mid R)$.

  By choosing $W \equiv Q' \mid R$ and $q \equiv n \cdot L$, we are done.

- **Case 2:** Let $V \equiv P \mid R'$ and $p \equiv o \cdot R$. Here, we observe

		$P \mid R \xrightarrow{o \cdot R, \alpha} P \mid R'$
(Rule (Com2))	implies	$R \xrightarrow{o,\alpha} R'$ and $\underline{\mathit{fl}}_{[o]}(R) \cap \overline{\underline{\mathit{fl}}(P)} = \emptyset$
(Lemma 6.4.3)	implies	$R \xrightarrow{o,\alpha} R'$ and $\underline{\mathit{fl}}_{[o]}(R) \cap \overline{\underline{\mathit{fl}}(Q)} = \emptyset$
(Rule (Com2),	implies	$Q \mid R \xrightarrow{o \cdot R, \alpha} Q \mid R',$
def. of $\mathcal{R}$,		$\langle P \mid R', Q \mid R' \rangle \in \mathcal{R},$ and
Lemma 6.3.3(2))		$\underline{\mathit{fl}}_{[o \cdot R]}(P \mid R) = \underline{\mathit{fl}}_{[o \cdot R]}(Q \mid R)$.

  Our claim follows by choosing $W \equiv Q \mid R'$ and $q \equiv o \cdot R$.

- **Case 3:** Let $V \equiv P' \mid R'$ and $p \equiv \langle m \cdot L, o \cdot R \rangle$. In this case, we establish

		$P \mid R \xrightarrow{\langle m \cdot L, o \cdot R \rangle, \tau} P' \mid R'$
(Rule (Com3))	implies	$\exists a \in \Lambda \cup \overline{\Lambda}.\, P \xrightarrow{m,a} P', \underline{\mathit{fl}}_{[m]}(P) \cap \overline{\underline{\mathit{fl}}(R)} = \emptyset,$
		$R \xrightarrow{o,\overline{a}} R',$ and $\underline{\mathit{fl}}_{[o]}(R) \cap \overline{\underline{\mathit{fl}}(P)} = \emptyset$
$(P \simeq^{\mathrm{l}} Q)$	implies	$\exists Q', n, a.\, Q \xrightarrow{n,a} Q', P' \simeq^{\mathrm{l}} Q',$
		$\underline{\mathit{fl}}_{[n]}(Q) \subseteq \underline{\mathit{fl}}_{[m]}(P),$
		$\underline{\mathit{fl}}_{[m]}(P) \cap \overline{\underline{\mathit{fl}}(R)} = \emptyset, R \xrightarrow{o,\overline{a}} R',$ and
		$\underline{\mathit{fl}}_{[o]}(R) \cap \overline{\underline{\mathit{fl}}(P)} = \emptyset$
(Lemma 6.4.3)	implies	$\exists Q', n, a.\, Q \xrightarrow{n,a} Q', P' \simeq^{\mathrm{l}} Q',$
		$\underline{\mathit{fl}}_{[n]}(Q) \subseteq \underline{\mathit{fl}}_{[m]}(P),$
		$\underline{\mathit{fl}}_{[n]}(Q) \cap \overline{\underline{\mathit{fl}}(R)} = \emptyset, R \xrightarrow{o,\overline{a}} R',$ and
		$\underline{\mathit{fl}}_{[o]}(R) \cap \overline{\underline{\mathit{fl}}(Q)} = \emptyset$
(Rule (Com3),	implies	$\exists Q' \mid R', n.\, Q \mid R \xrightarrow{\langle n \cdot L, o \cdot R \rangle, \tau} Q' \mid R',$
def. of $\mathcal{R}$,		$\langle P' \mid R', Q' \mid R' \rangle \in \mathcal{R},$ and
Lemma 6.3.3(3))		$\underline{\mathit{fl}}_{[\langle n \cdot L, o \cdot R \rangle]}(Q \mid R) \subseteq \underline{\mathit{fl}}_{[\langle m \cdot L, o \cdot R \rangle]}(P \mid R)$.

  By defining $W \equiv Q' \mid R'$ and $q \equiv \langle n \cdot L, o \cdot R \rangle$, we have finished the proof according to the definition of $\mathcal{R}$.

The first condition of Definition 6.4.2 can be proved as usual since the semantic rules for parallel composition coincide with those of CCS with respect to prioritized actions.

In order to prove the compositionality of $\simeq^{\mathrm{l}}$ with respect to recursion one has to adapt a definition of *distributed prioritized strong bisimulation up to* from [125]. Then the proof follows the lines of the corresponding proof in [125]. □

The following theorem is the main theorem of this section.

Theorem 6.4.5 *Distributed prioritized strong bisimulation $\simeq^{\mathrm{l}}$ is the* largest *congruence contained in naive distributed prioritized strong bisimulation $\simeq$, i.e. $\simeq^{+} = \simeq^{\mathrm{l}}$.*

In the proof of this theorem we use $\mathcal{S}(P)$ to denote the unprioritized sort of P and $\underline{\mathcal{S}(P)}$ for its prioritized sort, i.e. the set of all unprioritized and prioritized visible actions occurring as labels in the transition system corresponding to P, respectively. The next property of $\mathsf{CCS}^{\mathsf{prio}}$ processes is important for the proof of Theorem 6.4.5 as well as for the proofs of other important theorems presented in the sequel.

Lemma 6.4.6 (Finite Sorts)
For all $P \in \mathcal{P}$ the sorts $\mathcal{S}(P) \subseteq \Lambda \cup \overline{\Lambda}$ and $\underline{\mathcal{S}(P)} \subseteq \underline{\Lambda} \cup \overline{\underline{\Lambda}}$ are finite.

The validity of this lemma is an immediate consequence of the facts that the summation operator is binary and that relabelings f satisfy the condition $|\{\gamma \,|\, f(\gamma) \not\equiv \gamma\}| < \infty$. Now, we turn to the proof of Theorem 6.4.5.

Proof: By Proposition 2.4.3 the largest congruence $\simeq^{+}$ in $\simeq$ exists and is characterized by $P \simeq^{+} Q$ if and only if $\forall \mathsf{CCS}^{\mathsf{prio}}$ contexts $C[X]$. $C[P] \simeq C[Q]$.

It is straightforward to show that $\simeq^{\mathrm{l}} \subseteq \simeq$; one need only prove that $\simeq^{\mathrm{l}}$ is a naive distributed prioritized strong bisimulation. The proof is standard and is omitted. Also, since $\simeq^{\mathrm{l}}$ is a congruence we know that $\simeq^{\mathrm{l}} \subseteq \simeq^{+}$. In order to prove the inclusion $\simeq^{+} \subseteq \simeq^{\mathrm{l}}$ it suffices to show that $\mathcal{R} =_{\mathrm{df}} \{\langle P, Q\rangle \,|\, C_{PQ}[P] \simeq C_{PQ}[Q]\}$ is a distributed prioritized strong bisimulation for some $\mathsf{CCS}^{\mathsf{prio}}$ context $C_{PQ}[X]$. For our purposes we define $C_{PQ}[X] \stackrel{\mathrm{def}}{=} X \,|\, H_{PQ}$ for $P, Q \in \mathcal{P}$ where

$$H_{PQ} \stackrel{\mathrm{def}}{=} \sum_{L \subseteq \overline{\underline{\mathcal{S}(P)}} \cup \overline{\underline{\mathcal{S}(Q)}}} \underline{\tau}.(\underline{d}_L.H_{PQ} + \underline{D}_L)$$

and $\underline{D}_L =_{\mathrm{df}} \sum_{\underline{c} \in L} \underline{c}.\mathbf{0}$. Note that $\sum$ is the extension of the associative binary operator $+$ to finitely many operands and that H_{PQ} is well-defined by Lemma 6.4.6. We assume that $\underline{d}_L, \overline{\underline{d}}_L \notin \underline{\mathcal{S}(P)} \cup \underline{\mathcal{S}(Q)}$. Such $\underline{d}_L$'s exist because the prioritized sort of a process is finite according to Lemma 6.4.6. Note that the context $C_{PQ}[X]$ is inspired by the one presented in Section 2.4.2.

Now, let $P, Q \in \mathcal{P}$ satisfying $C_{PQ}[P] \simeq C_{PQ}[Q]$ and $P \stackrel{m,\alpha}{\longrightarrow} P'$. Therefore, $C_{PQ}[P]$ can engage in the transitions illustrated in the left hand side of Figure 6.2 where $L = \{\overline{\underline{c}} \,|\, \underline{c} \in$

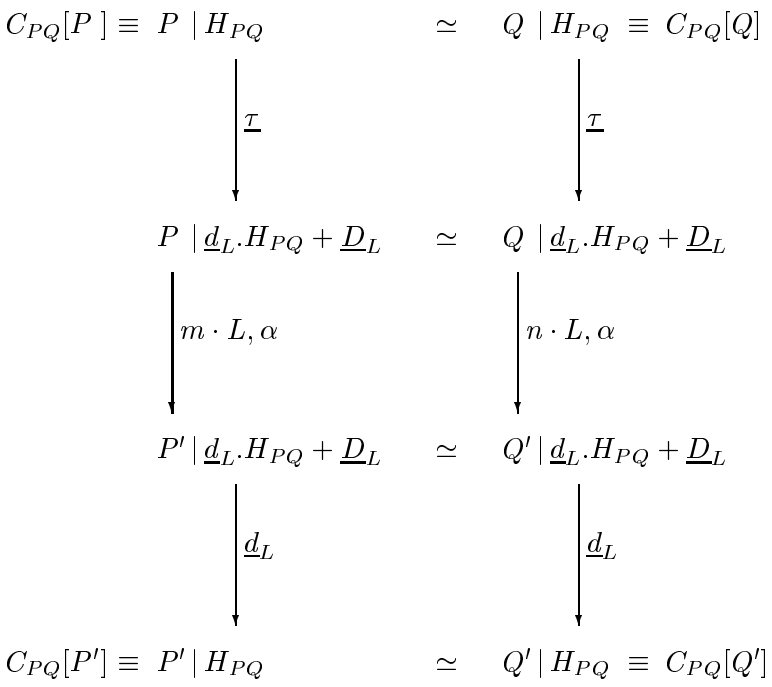

Figure 6.2: Largest congruence proof – illustration

$(\underline{\mathcal{S}(P)} \cup \underline{\mathcal{S}(Q)}) \setminus \underline{\mathit{\Pi}}_{[m]}(P)\}$. Since $C_{PQ}[P] \simeq C_{PQ}[Q]$, the process $C_{PQ}[Q]$ has to match each step.

In order to be able to match the first step, $C_{PQ}[Q]$ has to choose exactly the same branch of H_{PQ} yielding to the process $\underline{d}_L.H_{PQ} + \underline{D}_L$, because only this process is able to execute the distinguished action $\underline{d}_L$. For matching the second step, the process Q must be able to perform an α-transition from some location $n \in \mathcal{L}oc$. According to our semantics for parallel composition, the condition $\underline{\mathit{\Pi}}_{[n]}(Q) \cap \overline{\underline{\mathit{\Pi}}(\underline{d}_L.H_{PQ} + \underline{D}_L)} = \emptyset$ has to be satisfied. Because of the choice of L, this implies $\underline{\mathit{\Pi}}_{[n]}(Q) \subseteq \underline{\mathit{\Pi}}_{[m]}(P)$. The match of the third step, observing action $\underline{d}_L$, is straightforward. Thus, Figure 6.2 shows the existence of some $Q' \in \mathcal{P}$ satisfying $C_{PQ}[P'] \simeq C_{PQ}[Q']$. Since $\underline{\mathcal{S}(P')} \subseteq \underline{\mathcal{S}(P)}$ and $\underline{\mathcal{S}(Q')} \subseteq \underline{\mathcal{S}(Q)}$ it follows that $C_{P'Q'}[P'] \simeq C_{P'Q'}[Q']$ by the construction of our context, as desired.

The case where P performs a prioritized transition needs no special attention since the condition of Definition 6.4.1 and Condition (1) of Definition 6.4.2 are identical in this case. Summarizing, we have shown that all conditions of Definition 6.4.2 are satisfied, and we may conclude that $\mathcal{R}$ is a distributed prioritized strong bisimulation. Hence, $P \simeq^{\mathrm{l}} Q$, which completes the proof. □

Table 6.4: Axiomatization of $\simeq^{\mathrm{l}}$ (Axioms E)

(A1)	$t + u = u + t$	(iA1)	$t \oplus u = u \oplus t$
(A2)	$t + (u + v) = (t + u) + v$	(iA2)	$t \oplus (u \oplus v) = (t \oplus u) \oplus v$
(A3)	$t + t = t$	(iA3)	$t \oplus t = t$
(A4)	$t + \mathbf{0} = t$	(iA4)	$t \oplus \mathbf{0} = t$

(P) $\quad \underline{\tau}.t + \alpha.u = \underline{\tau}.t$

(E) $\quad t \equiv \bigoplus_i \sum_j \gamma_{ij}.t_{ij}$ and $u \equiv \bigoplus_k \sum_l \delta_{kl}.u_{kl}$ implies $t \,|\, u =$

$$\begin{aligned}
&\bigoplus_i \textstyle\sum_j (\gamma_{ij}.(t_{ij} \,|\, u) + \sum_k \sum_l \{\tau.(t_{ij} \,|\, u_{kl}) \,|\, \gamma_{ij} \equiv \overline{\delta}_{kl},\ \gamma_{ij}, \delta_{kl} \in A\} \\
&\qquad + \textstyle\sum_k \sum_l \{\underline{\tau}.(t_{ij} \,|\, u_{kl}) \,|\, \gamma_{ij} \equiv \overline{\delta}_{kl},\ \gamma_{ij}, \delta_{kl} \in \underline{A}\}) \oplus \\
&\bigoplus_k \textstyle\sum_l (\delta_{kl}.(t \,|\, u_{kl}) + \sum_i \sum_j \{\tau.(t_{ij} \,|\, u_{kl}) \,|\, \gamma_{ij} \equiv \overline{\delta}_{kl},\ \gamma_{ij}, \delta_{kl} \in A\} \\
&\qquad + \textstyle\sum_i \sum_j \{\underline{\tau}.(t_{ij} \,|\, u_{kl}) \,|\, \gamma_{ij} \equiv \overline{\delta}_{kl},\ \gamma_{ij}, \delta_{kl} \in \underline{A}\})
\end{aligned}$$

(Res1)	$\mathbf{0} \setminus L = \mathbf{0}$	
(Res2)	$(\gamma.t) \setminus L = \mathbf{0}$	$(\gamma \in L \cup \overline{L})$
(Res3)	$(\gamma.t) \setminus L = \gamma.(t \setminus L)$	$(\gamma \notin L \cup \overline{L})$
(Res4)	$(t + u) \setminus L = (t \setminus L) + (u \setminus L)$	
(iRes4)	$(t \oplus u) \setminus L = (t \setminus L) \oplus (u \setminus L)$	

(Rel1)	$\mathbf{0}[f] = \mathbf{0}$	(Rel3)	$(t + u)[f] = t[f] + u[f]$
(Rel2)	$(\gamma.t)[f] = f(\gamma).(t[f])$	(iRel3)	$(t \oplus u)[f] = t[f] \oplus u[f]$

6.4.2 Axiomatic Characterization

In this section we give an axiomatization of $\simeq^{\mathrm{l}}$ for *finite* processes, i.e. processes that do not contain recursion. Since axioms provide a means for rewriting terms into semantically equivalent ones, we prefer to speak of processes as *process terms*, ranged over by $t, u, v, \ldots$, in this section. In order to develop the axiomatization, we add a new binary summation operator $\oplus$ to the process algebra $\mathsf{CCS}^{\mathsf{prio}}$. This operator is called *distributed summation* and needed for giving an *Expansion Axiom* (cf. Axiom (E) in Table 6.4). Its operational semantics is the following.

$$\underline{\mathsf{dSum1}}\ \frac{t \xrightarrow{m,\underline{\alpha}} t'}{t \oplus u \xrightarrow{m \cdot L,\underline{\alpha}} t'} \qquad \mathsf{dSum1}\ \frac{t \xrightarrow{m,\alpha} t'}{t \oplus u \xrightarrow{m \cdot L,\alpha} t'}$$

$$\underline{\mathsf{dSum2}}\ \frac{u \xrightarrow{n,\underline{\alpha}} u'}{t \oplus u \xrightarrow{n \cdot R,\underline{\alpha}} u'} \qquad \mathsf{dSum2}\ \frac{u \xrightarrow{n,\alpha} u'}{t \oplus u \xrightarrow{n \cdot R,\alpha} u'}$$

It can easily be checked that distributed prioritized strong bisimulation is also compositional with respect to $\oplus$.

Table 6.5: Axioms E (continued)

(D1)	$(t \oplus t') + (u \oplus u') = ((t \oplus t') + u') \oplus ((u \oplus u') + t')$	
	$(\vdash_I t \sqsubseteq_{\mathsf{i}} t' \,,\ \vdash_I u \sqsubseteq_{\mathsf{i}} u')$	
(D2)	$(t \oplus u) + \underline{\alpha}.v = (t + \underline{\alpha}.v) \oplus (u + \underline{\alpha}.v)$	
(Ic1)	$t \oplus \underline{\alpha}.u = t + \underline{\alpha}.u$	$(\natural t)$
(Ic2)	$(\underline{\alpha}.t + u) = (\underline{\alpha}.t + u) \oplus \underline{\alpha}.t$	
(S1)	$(t + \underline{\alpha}.u) \oplus (t' + \underline{\alpha}.u') = (t + \underline{\alpha}.u + \underline{\alpha}.u') \oplus (t' + \underline{\alpha}.u')$	
(S2)	$(t + \alpha.v) \oplus (u + \alpha.v) = (t + \alpha.v) \oplus u$	$(\vdash_I t \sqsubseteq_{\mathsf{i}} u)$
(S3)	$t \oplus u = t + u$	$(\vdash_I t =_{\mathsf{i}} u)$

Table 6.6: Axiomatization of $\sqsubseteq_{\mathsf{i}}$ (Axioms I)

(iC1)	$\underline{\alpha}.t \sqsubseteq_{\mathsf{i}} \underline{\alpha}.u$	(iC2)	$\mathbf{0} \sqsubseteq_{\mathsf{i}} \nu.t$	$\nu \in \mathcal{A} \setminus \{\underline{\tau}\}$
(iC3)	$\alpha.t \sqsubseteq_{\mathsf{i}} \mathbf{0}$			

Now, we turn to the axiom system for distributed prioritized strong bisimulation. We write $\vdash_E t = u$ if term t can be rewritten to u using the axioms in Tables 6.4 and 6.5. Axioms (Ic1), (D1), (S2), and (S3) involve side conditions. Regarding Axiom (Ic1), we introduce the unary predicate $\natural$ over process terms of the form $\sum_{j \in J} \gamma_j.t_j$ for some nonempty index set J together with the following proof rules: (i) $\natural\underline{\alpha}.t$ and (ii) $\natural t$ and $\natural u$ implies $\natural(t + u)$. Intuitively, $\natural(\sum_{j \in J} \gamma_j.t_j)$ if and only if $\gamma_j \in \underline{A}$ for all $j \in J$. The relation $\sqsubseteq_{\mathsf{i}}$ is the pre-congruence on finite process terms generated from the axioms presented in Table 6.6 using the laws of inequational reasoning. Its meaning is precised by Lemma 6.4.7 below. We write $\vdash_I t \sqsubseteq_{\mathsf{i}} u$ if t can be related to u by Axioms (iC1), (iC2), and (iC3), and notate $\vdash_I t =_{\mathsf{i}} u$ if $\vdash_I t \sqsubseteq_{\mathsf{i}} u$ and $\vdash_I u \sqsubseteq_{\mathsf{i}} t$. The axioms in Table 6.4 are basically those given in [54] and augmented with the corresponding axioms for the distributed summation operator. Moreover, the Expansion Axiom has been adapted for our algebra (cf. Axiom (E) where $\sum$ is the indexed version of $+$, and $\bigoplus$ is the indexed version of $\oplus$). The axioms in Table 6.5

are new and show how we may "restructure" locations. They deal with the *distributivity* of the summation operators (Axioms (D1) and (D2)), the *interchangeability* of the summation operators (Axioms (Ic1) and (Ic2)), and the *saturation* of locations (Axioms (S1), (S2), and (S3)), respectively.

The following lemma presents a semantic interpretation of $\vdash_I t \sqsubseteq_{\mathrm{i}} u$ that is essential for the soundness and completeness proof of our axiomatization. It uses the notation $\vdash_A t = u$ meaning that t can be rewritten to u by using Axioms (A1)–(A4) only. Hence, $\vdash_A t = u$ implies $\vdash_E t = u$.

Lemma 6.4.7 (Semantic Interpretation of $\sqsubseteq_{\mathrm{i}}$)

1. *Let $\vdash_I t \sqsubseteq_{\mathrm{i}} u$. Then, $\underline{\Pi}(t) \subseteq \underline{\Pi}(u)$, and $\underline{\tau} \in \underline{\mathcal{I}}(t)$ if and only if $\underline{\tau} \in \underline{\mathcal{I}}(u)$.*

2. *Let $t \equiv \sum_{i=1}^{m} \gamma_i.t_i$ and $u \equiv \sum_{j=1}^{n} \delta_j.u_j$ be finite process terms such that $\underline{\Pi}(t) \subseteq \underline{\Pi}(u)$, and $\underline{\tau} \in \underline{\mathcal{I}}(t)$ if and only if $\underline{\tau} \in \underline{\mathcal{I}}(u)$. Then there exist process terms t' and u' such that $\vdash_A t' = t$, $\vdash_A u' = u$, and $\vdash_I t' \sqsubseteq_{\mathrm{i}} u'$. The same holds if we replace "$\subseteq$" by "$=$" and "$\sqsubseteq_{\mathrm{i}}$" by "$=_{\mathrm{i}}$."*

Proof: The proof of Part (1) is done in a straightforward manner by inducting upon the length of the proof $\vdash_I t \sqsubseteq_{\mathrm{i}} u$.

We prove Part (2) by induction on the cardinality of $\underline{\mathcal{I}}(t)$. As the base case suppose $\underline{\mathcal{I}}(t)$ is empty. Then it is easy to show the existence of process terms t' and u' such that $\vdash_A t' = t$, $\vdash_A u' = u$, and $\vdash_I t' \sqsubseteq_{\mathrm{i}} u'$ by just using Axioms (iC2) and (iC3) for establishing the latter. For the induction step, we have $|\underline{\mathcal{I}}(t)| > 0$, and let $\underline{\alpha} \in \underline{\mathcal{I}}(t)$ as well. Since $\underline{\Pi}(t) \subseteq \underline{\Pi}(u)$, and $\underline{\tau} \in \underline{\mathcal{I}}(t)$ if and only if $\underline{\tau} \in \underline{\mathcal{I}}(u)$, we have $\underline{\alpha} \in \underline{\mathcal{I}}(u)$. Let l_t and l_u be the number of summands in t and u that are prefixed by $\underline{\alpha}$. Formally, if $L_t =_{\mathrm{df}} \{j \mid \delta_j \equiv \underline{\alpha}\}$ and $L_u =_{\mathrm{df}} \{i \mid \gamma_i \equiv \underline{\alpha}\}$, then $l_t =_{\mathrm{df}} |L_t| \geq 1$ and $l_u =_{\mathrm{df}} |L_u| \geq 1$. Let t^1 and u^1 be the process terms obtained by re-ordering the summands of the process terms t and u such that the summands prefixed by $\underline{\alpha}$ precede those that do not. Formally, let $t^1 \equiv t_1^1 + t_2^1$ where $t_1^1 \equiv \sum_{i \in L_t} \gamma_i.t_i$ and $t_2^1 \equiv \sum_{i \notin L_t} \gamma_i.t_i$. Similarly, let $u^1 \equiv u_1^1 + u_2^1$ where $u_1^1 \equiv \sum_{j \in L_u} \delta_j.u_j$ and $u_2^1 \equiv \sum_{j \notin L_u} \delta_j.u_j$. We may apply the induction hypothesis to t_2^1 and u_2^1 to obtain process terms t_2' and u_2' such that $\vdash_A t_2' = t_2^1$, $\vdash_A u_2' = u_2^1$, and $\vdash_I t_2' \sqsubseteq_{\mathrm{i}} u_2'$.

We define process terms t_1' and u_1' such that they have an equal number of summands and such that they are derivable from t_1^1 and u_1^1, respectively, just by using Axioms (A1)–(A3). Their structures depend on the relative values of l_t and l_u.

- If $l_t = l_u$ then take $t_1' \equiv t_1^1$ and $u_1' \equiv u_1^1$.
- If $l_t > l_u$ then take $t_1' \equiv t_1^1$ and $u_1' \equiv u_1^1 + (\sum_{j=1}^{l_t - l_u} \delta_k.u_k)$, where $k \in L_u$.
- If $l_t < l_u$ then take $u_1' \equiv u_1^1$ and $t_1' \equiv t_1^1 + (\sum_{i=1}^{l_u - l_t} \gamma_k.t_k)$, where $k \in L_t$.

It is easy to show that $\vdash_I t_1' \sqsubseteq_{\mathrm{i}} u_1'$. Thus if we take $t' \equiv t_1' + t_2'$ and $u' \equiv u_1' + u_2'$, then $\vdash_A t = t'$, $\vdash_A u = u'$, and $\vdash_I t' \sqsubseteq_{\mathrm{i}} u'$. This concludes the proof of the induction step. By

inspection of the above proof it is clear that the same statement holds if we replace "$\subseteq$" by "$=$" and "$\sqsubseteq_{\mathrm{i}}$" by "$=_{\mathrm{i}}$" which completes the proof of the lemma. □

The following theorem states the soundness of our axiom system for arbitrary, not only for finite, process terms.

Theorem 6.4.8 (Soundness) *For process terms t, u satisfying $\vdash_E t = u$ we have $t \simeq^{\downarrow} u$.*

Proof: Formally, the soundness of the axioms can be shown by constructing a distributed prioritized strong bisimulation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ for each axiom $t = u$ such that $\langle t, u \rangle \in \mathcal{R}$. This is obvious for all axioms beside Axioms (D1), (Ic1), (S2), and (S3). Those axioms are only true if the appropriate side condition is satisfied. However, their soundness can easily be established by using the semantic interpretation of the syntactic side conditions given in Lemma 6.4.7(1) and the meaning of $\natural$, respectively. □

In order to prove our axiomatization complete, we introduce a notion of *normal form* of process terms that is based on the following definition.

Definition 6.4.9 (Summation Form)
A process term t is in summation form *if it has the form $t \equiv \bigoplus_{i=1}^{m} \sum_{j=1}^{n_i} \gamma_{ij}.t_{ij}$ where $m, n_i \in \mathbb{N}$ and the process terms t_{ij} are in summation form, too. Per definition, $\mathbf{0}$ is in summation form.*

Intuitively, t is distributed throughout m incomparable locations which themselves consist of n_i comparable locations, $1 \leq i \leq m$. The following proposition states that every finite process term can be rewritten into summation form.

Proposition 6.4.10 *For every finite process term t there exists a process term v in summation form such that $\vdash_E v = t$.*

The proof of Proposition 6.4.10 uses the following lemma.

Lemma 6.4.11 *Let $t \equiv \bigoplus_{i=1}^{m} \sum_{j=1}^{n_i} \gamma_{ij}.t_{ij}$. Then $\vdash_E t = t \oplus \underline{t}$ and $\vdash_I t \sqsubseteq_{\mathrm{i}} \underline{t}$ for $\underline{t} \equiv \bigoplus_{i=1}^{m} \sum_{j=1}^{n_i} \underline{t}_{ij}$, and $\underline{t}_{ij} \equiv \gamma_{ij}.t_{ij}$ if $\gamma_{ij} \in \underline{A}$, and $\underline{t}_{ij} \equiv \mathbf{0}$ if $\gamma_{ij} \in A$.*

In the following we often write $(\mathrm{A})^*$ to denote that we apply the axiom named A multiple times.

Proof: Let $t \equiv \bigoplus_{i=1}^{m} \sum_{j=1}^{n_i} \gamma_{ij}.t_{ij}$. The proof is done by induction on m. For the induction base let $m = 0$. Then $t \equiv \mathbf{0}$ and $\underline{t} \equiv \mathbf{0}$. Thus, $\vdash_E t = t \oplus \underline{t}$ by Axiom (iA3), and $\vdash_I t \sqsubseteq_{\mathrm{i}} \underline{t}$ since reflexivity is part of inequational reasoning. For the induction step let $t \equiv t_1 \oplus t_2$

where $t_1 \equiv \sum_{j=1}^{n_1} \gamma_{1j}.t_{1j}$ and $t_2 \equiv \bigoplus_{i=2}^{m+1} \sum_{j=1}^{n_i} \gamma_{ij}.t_{ij}$. In the following lines we establish $\vdash_E t_1 = t_1 \oplus \underline{t}_1$.

$$
\begin{array}{lcl}
 & \vdash_E & t_1 \\
\text{(def. } t_1\text{)} & = & \sum_{j=1}^{n_1} \gamma_{1j}.t_{1j} \\
(\mathsf{lc2})^*, (\mathsf{iA1})^*, (\mathsf{iA2})^* & = & (\sum_{j=1}^{n_1} \gamma_{1j}.t_{1j}) \oplus \\
 & & (\bigoplus\{\gamma_{1j}.t_{1j} \mid 1 \leq j \leq n_1 \wedge \gamma_{1j} \in \underline{A}\}) \\
(\mathsf{lc1})^*\text{, def. } t_1) & = & t_1 \oplus (\sum\{\gamma_{1j}.t_{1j} \mid 1 \leq j \leq n_1 \wedge \gamma_{1j} \in \underline{A}\}) \\
(\mathsf{A4})^* & = & t_1 \oplus (\sum\{\gamma_{1j}.t_{1j} \mid 1 \leq j \leq n_1 \wedge \gamma_{1j} \in \underline{A}\} + \\
 & & \quad \sum\{\mathbf{0} \mid 1 \leq j \leq n_1 \wedge \gamma_{1j} \in A\}) \\
\text{(def. } \underline{t}_1\text{, } (\mathsf{A1})^*, (\mathsf{A2})^*) & = & t_1 \oplus \underline{t}_1
\end{array}
$$

Moreover, it is easy to verify that $\vdash_I t_1 \sqsubseteq_{\mathsf{i}} \underline{t}_1$ by Axioms (iC1)–(iC3).

By induction hypothesis $\vdash_E t_2 = t_2 \oplus \underline{t}_2$ and $\vdash_I t_2 \sqsubseteq_{\mathsf{i}} \underline{t}_2$. Now, we complete the proof of the induction step by combining the result of the previous paragraph and the induction hypothesis using equational and inequational reasoning, respectively: $\vdash_E t = t_1 \oplus t_2 = (t_1 \oplus \underline{t}_1) \oplus (t_2 \oplus \underline{t}_2) = (t_1 \oplus t_2) \oplus (\underline{t_1} \oplus \underline{t_2}) = t \oplus \underline{t}$ and $\vdash_I t = t_1 \oplus t_2 \sqsubseteq_{\mathsf{i}} \underline{t}_1 \oplus \underline{t_2} = \underline{t}$. □

In the remainder we use the notion of the *depth* of a finite process term in summation form, which is defined to be the maximum number of nested prefixes in the considered process term, as usual. Now, we are able to prove Proposition 6.4.10.

Proof: The proof is done by induction on the structure of the finite process term t.

1. $\mathbf{t \equiv 0}$**:** Per definition, $\mathbf{0}$ is in summation form.

2. $\mathbf{t \equiv \gamma.t'}$**:** By induction hypothesis there exists $v' \in \mathcal{P}$ in summation form such that $\vdash_E v' = t'$. Thus, we conclude $\vdash_E t = \gamma.v'$ where $\gamma.v'$ is in summation form according to Definition 6.4.9.

3. $\mathbf{t \equiv t' + t''}$**:** Applying the induction hypothesis twice, we know of the existence of process terms $v' \equiv \bigoplus_{i=1}^{m} \sum_{j=1}^{n_i} \gamma_{ij}.v'_{ij}$ and $v'' \equiv \bigoplus_{k=1}^{r} \sum_{l=1}^{s_k} \delta_{kl}.v''_{kl}$ in summation form such that $\vdash_E v' = t'$ and $\vdash_E v'' = t''$. By using Lemma 6.4.11 and by the transitivity of equational reasoning we obtain $\vdash_E t = (v' \oplus \underline{v}') + (v'' \oplus \underline{v}'')$ where $\vdash_I v' \sqsubseteq_{\mathsf{i}} \underline{v}'$ and $\vdash_I v'' \sqsubseteq_{\mathsf{i}} \underline{v}''$. Therefore, we apply Axiom (D1) to obtain $\vdash_E t = ((v' \oplus \underline{v}') + \underline{v}'') \oplus ((v'' \oplus \underline{v}'') + \underline{v}')$ and Lemma 6.4.11 again to conclude $\vdash_E t = (v' + \underline{v}'') \oplus (v'' + \underline{v}')$. Now, we use Axiom (lc1) and replace all $\oplus$-operators in $\underline{v}''$ and $\underline{v}'$ by $+$-operators since all initial actions in v'' and v' are prioritized ones. We denote the so obtained process terms from $\underline{v}''$ and $\underline{v}'$ by $\hat{v}''$ and $\hat{v}'$, respectively. Finally, we apply Axiom (D2) (and Axioms (iA1) and (iA2)) multiple times to push $\hat{v}'$ in every distributed summand of v' and $\hat{v}''$ in every distributed summand of v''. The resulting process term is in summation form.

4. $\mathbf{t \equiv t' \oplus t''}$**:** By induction hypothesis there exist process terms v' and v'' in summation form such that $\vdash_E v' = t'$ and $\vdash_E v'' = t''$. Thus, we conclude $\vdash_E v' \oplus v'' = t' \oplus t'' = t$ where $v' \oplus v''$ is obviously in summation form, as desired.

5. $\mathbf{t \equiv t' \,|\, t''}$**:** By induction hypothesis there exist process terms $v' \equiv \bigoplus_{i=1}^{m} \sum_{j=1}^{n_i} \gamma_{ij}.v'_{ij}$ and $v'' \equiv \bigoplus_{k=1}^{r} \sum_{l=1}^{s_k} \delta_{kl}.v''_{kl}$ in summation form such that $\vdash_E v' = t'$ and $\vdash_E v'' = t''$. We prove this case by induction on the sum of the depths of v' and v''. If the sum of the depths is 0, then both process terms v' and v'' can be rewritten to $\mathbf{0}$ by applying Axiom (iA3). Thus, $\vdash_E t = t' \,|\, t'' = v' \,|\, v'' = \mathbf{0} \,|\, \mathbf{0}$. Now, we may conclude $\vdash_E \mathbf{0} \,|\, \mathbf{0} = \mathbf{0}$ by using Axiom (E) and Axioms (A3) and (iA3). Since $\mathbf{0}$ is in summation form the induction base is done. For the induction step let the sum of the depths of v' and v'' be greater than 0. Here, we obtain

$$\begin{aligned}
\vdash_E t &= t' \,|\, t'' = v' \,|\, v'' \\
&= \bigoplus_i \sum_j (\gamma_{ij}.(v'_{ij} \,|\, v'') + \sum_k \sum_l \{\tau.(v'_{ij} \,|\, v''_{kl}) \,|\, \gamma_{ij} \equiv \overline{\delta}_{kl},\ \gamma_{ij}, \delta_{kl} \in A\} \\
&\qquad + \sum_k \sum_l \{\underline{\tau}.(v'_{ij} \,|\, v''_{kl}) \,|\, \gamma_{ij} \equiv \overline{\delta}_{kl},\ \gamma_{ij}, \delta_{kl} \in \underline{A}\}) \oplus \\
&\quad \bigoplus_k \sum_l (\delta_{kl}.(v' \,|\, v''_{kl}) + \sum_i \sum_j \{\tau.(v'_{ij} \,|\, v''_{kl}) \,|\, \gamma_{ij} \equiv \overline{\delta}_{kl},\ \gamma_{ij}, \delta_{kl} \in A\} \\
&\qquad + \sum_i \sum_j \{\underline{\tau}.(v'_{ij} \,|\, v''_{kl}) \,|\, \gamma_{ij} \equiv \overline{\delta}_{kl},\ \gamma_{ij}, \delta_{kl} \in \underline{A}\})
\end{aligned}$$

by applying Axiom (E). Since the sums of the depths of v'_{ij} and v'', v'_{ij} and v''_{kl}, and v' and v''_{kl} are strictly less than the sum of the depths of v' and v'', respectively, there exist process terms w_{ij}, w^{kl}_{ij}, and w_{kl} in summation form such that $\vdash_E w_{ij} = v'_{ij} \,|\, v''$, $\vdash_E w^{kl}_{ij} = v'_{ij} \,|\, v''_{kl}$, and $\vdash_E w_{kl} = v' \,|\, v''_{kl}$. Hence,

$$\begin{aligned}
\vdash_E t &= \bigoplus_i \sum_j (\gamma_{ij}.w_{ij} + \sum_k \sum_l \{\tau.w^{kl}_{ij} \,|\, \gamma_{ij} \equiv \overline{\delta}_{kl},\ \gamma_{ij}, \delta_{kl} \in A\} \\
&\qquad + \sum_k \sum_l \{\underline{\tau}.w^{kl}_{ij} \,|\, \gamma_{ij} \equiv \overline{\delta}_{kl},\ \gamma_{ij}, \delta_{kl} \in \underline{A}\}) \oplus \\
&\quad \bigoplus_k \sum_l (\delta_{kl}.w_{kl} + \sum_i \sum_j \{\tau.w^{kl}_{ij} \,|\, \gamma_{ij} \equiv \overline{\delta}_{kl},\ \gamma_{ij}, \delta_{kl} \in A\} \\
&\qquad + \sum_i \sum_j \{\underline{\tau}.w^{kl}_{ij} \,|\, \gamma_{ij} \equiv \overline{\delta}_{kl},\ \gamma_{ij}, \delta_{kl} \in \underline{A}\})
\end{aligned}$$

where the latter process term is in summation form as desired. This completes the proof of the inner induction step and the proof of the case of parallel composition.

6. $\mathbf{t \equiv t'[f]}$**:** By induction hypothesis there exists a process term $v' \equiv \bigoplus_{i=1}^{m} \sum_{j=1}^{n_i} \gamma_{ij}.v'_{ij}$ in summation form such that $\vdash_E v' = t'$. Thus, we may conclude $\vdash_E t = v'[f]$. In order to prove that $v'[f]$ can be rewritten into summation form, we need to use induction on the depth of the process term v'. If the depth is 0 then $\vdash_E v' = \mathbf{0}$ is obtained by Axiom (iA3). Applying Axiom (Rel1) finishes the proof of the induction base. For the induction step let the depth of v' be greater than 0, i.e. $n_i > 0$ for some $1 \leq i \leq m$. We conclude as follows.

$$\begin{array}{lcl}
 & \vdash_E & t \\
\text{(see above)} & = & (\bigoplus_{i=1}^{m} \sum_{j=1}^{n_i} \gamma_{ij}.v'_{ij})[f] \\
\text{((iRel3)}^*\text{)} & = & \bigoplus_{i=1}^{m} ((\sum_{j=1}^{n_i} \gamma_{ij}.v'_{ij})[f]) \\
\text{((Rel3)}^*\text{)} & = & \bigoplus_{i=1}^{m} \sum_{j=1}^{n_i} ((\gamma_{ij}.v'_{ij})[f]) \\
\text{((Rel2)}^*\text{)} & = & \bigoplus_{i=1}^{m} \sum_{j=1}^{n_i} f(\gamma_{ij}).(v'_{ij}[f]) \\
\text{(ind. hyp.)} & = & \bigoplus_{i=1}^{m} \sum_{j=1}^{n_i} f(\gamma_{ij}).v_{ij}
\end{array}$$

where v_{ij} is in summation form for $1 \leq i \leq m$ and $1 \leq j \leq n_i$, and $\vdash_E v_{ij} = v'_{ij}[f]$. In the third step we apply Axiom (Rel1) instead of Axiom (Rel3) whenever $n_i = 0$, i.e.

$\sum_{j=1}^{n_i}((\gamma_{ij}.t_{ij})[f]) \equiv \mathbf{0}$. Since the process term $\bigoplus_{i=1}^{m}\sum_{j=1}^{n_i} f(\gamma_{ij}).v_{ij}$ is in summation form we have finished this case.

7. $\mathbf{t \equiv t' \setminus L}$**:** This case follows pretty much the same lines of the previous paragraph but uses Axioms (Res1)–(Res4) and (iRes4) instead of Axioms (Rel1)–(Rel3) and (iRel3) and, additionally, Axiom (iA4) whenever Axiom (Res2) has been applied.

□

Based on summation forms we define *normal forms* as follows.

Definition 6.4.12 (Normal Form)
Let $t \equiv \bigoplus_{i=1}^{m}\sum_{j=1}^{n_i}\gamma_{ij}.t_{ij}$ *be in summation form. We define* $\underline{\gamma}_{i*} =_{df} \{\gamma_{ij} \mid 1 \leq j \leq n_i\} \cap \underline{A}$. *The process term* t *is said to be in* normal form *if the following properties hold.*

1. $\emptyset \subseteq L \subseteq \underline{I}(t)$ *implies* $\exists i.\, \underline{\gamma}_{i*} = L$.
2. $\underline{\gamma}_{i*} = \underline{\gamma}_{k*}$ *implies* $i = k$.
3. $\gamma_{ij} \in A$ *implies* $\forall 1 \leq l \leq n_i.\ \gamma_{il} \not\equiv \underline{\tau}$.
4. $\gamma_{ij} \equiv \gamma_{kl} \equiv \underline{\alpha}$ *implies* $\exists j'.\, t_{ij'} \equiv t_{kl}$ *and* $\gamma_{ij'} \equiv \underline{\alpha}$.

Intuitively, Conditions (1) and (2) state that a term t in normal form contains exactly one distributed (or "outer") summand for each possible pre-emption potential, i.e. for each subset of the distributed prioritized initial actions of t. Condition (3) reflects our notion of pre-emption: an outer summand does not contain unprioritized initial actions when it also includes a prioritized internal action. The last condition requires outer summands to be "saturated" in a certain sense with respect to prioritized actions (cf. Axiom (S1)). The following proposition plays a key role in the completeness proof of our axiomatization for finite process terms.

Proposition 6.4.13 *If* t *is a finite process term, then there exists a normal form* w *such that* $\vdash_E w = t$.

Proof: Let t be a finite process term. By Proposition 6.4.10 we know of the existence of a process term $v \equiv \bigoplus_{i=1}^{m} v_{i*}$, where $v_{i*} \equiv \sum_{j=1}^{n_i}\gamma_{ij}.v_{ij}$, in summation form such that $\vdash_E v = t$. Therefore, it remains to establish that v can be equationally rewritten into a process term w in normal form such that $\vdash_E w = v$. This is done by induction on the depth of v. If the depth of v is 0, then we apply Axiom (iA3) in order to obtain $\vdash_E v = \mathbf{0}$ where $\mathbf{0}$ is obviously in normal form. For the induction step let the depth of v be greater than 0. Now, we proceed according to the following steps.

1. We use Lemma 6.4.11 to rewrite v to $v \oplus \underline{v}$, where $\underline{v} \equiv \bigoplus_{i=1}^{m} \sum_{j=1}^{n_i} \underline{v}_{ij}$, and $\underline{v}_{ij} \equiv \gamma_{ij}.v_{ij}$, if $\gamma_{ij} \in \underline{A}$, and $\underline{v}_{ij} \equiv \mathbf{0}$, otherwise. We switch the distributed summation operators in $\underline{v}$ into usual summation operators using Axiom (lc1), and Axioms (A2), (iA2), (A4), and (iA4), if needed. By Axioms (iA1)–(iA3) we duplicate summands of v and regroup them such that for each $\emptyset \neq L \subseteq \underline{\mathcal{I}}(v)$ there exists a distributed summand of the form $\bigoplus_{j \in J} \underline{\alpha}_j.v_j$ satisfying $L = \{\underline{\alpha}_j \mid j \in J\}$. Using Axiom (lc1) each of the above mentioned distributed summands is rewritten into a summand $\sum_{j \in J} \underline{\alpha}_j.v_j$. Finally, we apply Axiom (iA4) once obtaining the distributed summand $\mathbf{0}$ which fulfills the required condition for $L = \emptyset$. This concludes the establishment of Property (1).

2. Whenever $i \neq k$ and $\underline{\gamma}_{i*} = \underline{\gamma}_{k*}$, where $1 \leq i, k \leq m$, merge the terms v_{i*} and v_{k*} into one by applying the following steps. Use Axioms (iA1) and (iA2) to restructure the distributed summands of v such that the terms v_{i*} and v_{k*} are standing side by side. According to Lemma 6.4.7(2) there exist process terms v'_{i*} and v'_{k*} such that $\vdash_A v'_{i*} = v_{i*}$, $\vdash_A v'_{k*} = v_{k*}$, and $\vdash_E v'_{i*} =_{\mathrm{i}} v'_{k*}$. Thus, we may rewrite $v_{i*} \oplus v_{k*}$ to $v'_{i*} \oplus v'_{k*}$ and apply Axiom (S3) to substitute $v'_{i*} \oplus v'_{k*}$ by $v'_{i*} + v'_{k*}$. Repeat the above procedure as often as possible. Hence, Property (2) is established.

3. We use Axiom (P) in order to obtain Property (3).

4. As long as Property (4) is not satisfied, i.e. there exists $\gamma_{ij} \in \underline{\gamma}_{k*}$ for some $1 \leq k \leq m$ and $i \neq k$ but there is no $1 \leq l \leq n_k$ such that $\gamma_{kl}.v_{kl} \equiv \gamma_{ij}.v_{ij}$, then apply Axiom (S1), and possibly Axioms (A1), (A2), (iA1), and (iA2), to add the term $\gamma_{ij}.v_{ij}$ to v_{k*} as an additional summand. Thus, Property (4) is achieved.

This concludes the proof of the induction step and of the proposition. □

Rewriting a process term in its normal form requires restructuring its locations. After this is done, standard techniques used in CCS (cf. [125]) can be applied in order to show our axiomatization complete.

Theorem 6.4.14 (Completeness for Finite Process Terms)
For finite process terms t, u satisfying $t \simeq^{\mathrm{l}} u$ we have $\vdash_E t = u$.

Proof: Let t and u be finite process terms such that $t \simeq^{\mathrm{l}} u$. By Proposition 6.4.13 we may assume w.l.o.g. that t and u are in normal form, i.e. $t \equiv \bigoplus_{i=1}^{m} t_{i*}$ where $t_{i*} \equiv \sum_{j=1}^{n_i} \gamma_{ij}.t_{ij}$, and $u \equiv \bigoplus_{k=1}^{r} u_{k*}$ where $u_{k*} \equiv \sum_{l=1}^{s_k} \delta_{kl}.u_{kl}$, and Properties (1)–(4) of Definition 6.4.12 are satisfied. We reason by induction on the maximum of the depths of t and u.

- **Induction base:** If the maximum depth is 0 then t and u are of the form $\bigoplus_{i=1}^{m} \mathbf{0}$ and $\bigoplus_{k=1}^{r} \mathbf{0}$, respectively. Both process terms can be rewritten to $\mathbf{0}$ by applying Axiom (iA3), and since $\vdash_E \mathbf{0} = \mathbf{0}$ by equational reasoning, we are done.

- **Induction step:** Here, we use the following additional properties.

  (I) We assume w.l.o.g. that $t_{ij} \equiv u_{kl}$ for some $1 \leq i \leq m$, $1 \leq j \leq n_i$, $1 \leq k \leq r$, and $1 \leq l \leq s_k$ whenever $t_{ij} \simeq^{\mathrm{l}} u_{kl}$. The reason for being able to assume this is that the maximum of the depths of the process terms t_{ij} and u_{kl} is smaller than that of t and u. Hence the induction hypothesis is applicable, i.e. $\vdash_E t_{ij} = u_{kl}$ and we may substitute t_{ij} for u_{kl} since substitution is part of equational reasoning.

  (II) Moreover, we may assume w.l.o.g. that t satisfies the following property.

$$\gamma_{ij}.t_{ij} \equiv \gamma_{kl}.t_{kl}, \gamma_{ij} \in A, \text{ and } i \neq k \text{ implies } \underline{\gamma}_{i*} \not\subseteq \underline{\gamma}_{k*} \ .$$

  Otherwise, we use Lemma 6.4.7(2) that allows us to substitute t_{i*} and t_{k*} by process terms t'_{i*} and t'_{k*}, respectively, satisfying $\vdash_A t'_{i*} = t_{i*}$, $\vdash_A t'_{k*} = t_{k*}$, and $\vdash_I t'_{i*} \sqsubseteq_{\mathsf{i}} t'_{k*}$. Now, we apply Axioms (S2), (A1), (A2), (iA1), and (iA2) to achieve the above mentioned property. Note that this transformation does not destroy the normal forms of t and u, and Property (I). Similarly, we may assume that u fulfills Property (II).

Proof goal: In the main part of the induction step we show the existence of a bijection $\Phi : \{1, \ldots, m\} \longrightarrow \{1, \ldots, r\}$ such that $\forall 1 \leq i \leq m. \ \vdash_E t_{i*} = u_{\Phi(i)*}$. Thus, $\vdash_E t = u$ follows by possibly using Axioms (iA1) and (iA2) to re-order and re-group distributed summands.

Main part of the induction step: Since $t \simeq^{\mathrm{l}} u$ we have $\underline{\mathcal{I}}(t) = \underline{\mathcal{I}}(u)$. By Properties (1) and (2) of Definition 6.4.12 we may conclude that $m = r$. Moreover, the mapping $\Phi : \{1, \ldots, m\} \longrightarrow \{1, \ldots, r\}$ defined by $\Phi(i) =_{\mathrm{df}} k$ where $\underline{\gamma}_{i*} = \underline{\delta}_{k*}$ is a bijection. It remains to show that $\vdash_E t_{i*} = u_{\Phi(i)*}$ for some arbitrary $1 \leq i \leq m$. Let $k =_{\mathrm{df}} \Phi(i)$, i.e. $\underline{\gamma}_{i*} = \underline{\delta}_{k*}$. We show that every summand of t_{i*} is syntactically identical to a summand of u_{k*}. For every $1 \leq j \leq n_i$ we have $t_{i*} \stackrel{\gamma_{ij}}{\rightarrow} t_{ij}$ because of Property (3) of Definition 6.4.12.

- *Case 1:* Let $\gamma_{ij} \equiv \underline{\alpha} \in \underline{A}$ for some $j \in \{1, \ldots, n_i\}$. We may derive $t \stackrel{\underline{\alpha}}{\Rightarrow} t_{ij}$ according to our operational rules and the definition of t. Since $t \simeq^{\mathrm{l}} u$ there exists numbers k' and l', where $1 \leq k' \leq r$ and $1 \leq l' \leq s_{k'}$, such that $u \stackrel{\underline{\alpha}}{\Rightarrow} u_{k'l'}$, $\underline{\alpha} \equiv \delta_{k'l'}$, and $t_{ij} \simeq^{\mathrm{l}} u_{k'l'}$. By Property (I) we have $t_{ij} \equiv u_{k'l'}$ and, thus, $\gamma_{ij}.t_{ij} \equiv \delta_{k'l'}.u_{k'l'}$. Moreover, the summand $\delta_{k'l'}.u_{k'l'}$ syntactically equals with a summand $\delta_{kl}.u_{kl}$ for some $1 \leq l \leq s_k$ by Property (4) of Definition 6.4.12 since $\delta_{k'l'} \in \underline{\delta}_{k*} = \underline{\gamma}_{i*}$.
- *Case 2:* Let $\gamma_{ij} \equiv \alpha \in A$ for some $j \in \{1, \ldots, n_i\}$. By the definition of t and the operational rules we also have $t \stackrel{\alpha}{\rightarrow} t_{ij}$. Note that we do not include the location of the action $\alpha \in A$ in the label since the corresponding distributed prioritized initial action set is already determined by the index i. Because of $t \simeq^{\mathrm{l}} u$ we know of the existence of numbers k' and l, where $1 \leq k' \leq r$ and $1 \leq l \leq s_{k'}$, such that $u \stackrel{\alpha}{\rightarrow} u_{k'l}$, $\alpha \equiv \delta_{k'l}$, $\underline{\delta}_{k'*} \subseteq \underline{\gamma}_{i*}$, and $t_{ij} \simeq^{\mathrm{l}} u_{k'l}$. Observe that

the inclusion $\underline{\delta}_{k'*} \subseteq \underline{\gamma}_{i*}$ is equivalent to our condition of distributed prioritized initial action set inclusion (cf. Condition (ii) of Definition 6.4.2) by Property (3) of normal forms. Because of Property (I) we may conclude that $t_{ij} \equiv u_{k'l}$ and, thus, $\gamma_{ij}.t_{ij} \equiv \delta_{k'l}.u_{k'l}$. It remains to establish $k' = k$. Since $t \simeq^{\mathrm{l}} u$ we know of the existence of numbers i' and j', where $1 \leq i' \leq m$ and $1 \leq j' \leq n_{i'}$, such that $t \stackrel{\alpha}{\Rightarrow} t_{i'j'}$, $\alpha \equiv \gamma_{i'j'}$, $\underline{\gamma}_{i'*} \subseteq \underline{\delta}_{k'*}$, and $t_{i'j'} \simeq^{\mathrm{l}} u_{k'l}$. Hence, $t_{i'j'} \equiv u_{k'l}$ by Property (I) and, thus, $\gamma_{i'j'}.t_{i'j'} \equiv \delta_{k'l}.u_{k'l}$. Together with $\gamma_{ij}.t_{ij} \equiv \delta_{k'l}.u_{k'l}$ we conclude $\gamma_{i'j'}.t_{i'j'} \equiv \gamma_{ij}.t_{ij}$. Moreover, we have established $\underline{\gamma}_{i'*} \subseteq \underline{\delta}_{k'*} \subseteq \underline{\gamma}_{i*}$. This implies $i = i'$ by Property (II), whence $\underline{\delta}_{k'*} = \underline{\gamma}_{i*} = \underline{\delta}_{k*}$. Now, we may conclude $k = k'$ by Property (2) of normal forms, as desired.

Similarly, every summand of u_{k*} is syntactically equal to a summand of t_{i*}. Hence, $\vdash_E t_{i*} = u_{k*}$ by using Axiom (A3) to eliminate duplicate summands and Axioms (A1) and (A2) to re-order and re-group summands as necessary.

This completes the induction step and, hence, the proof of the theorem. □

6.4.3 Operational Characterization

The following definition introduces an equivalence $\simeq_*$ which characterizes $\simeq^{\mathrm{l}}$ as standard strong bisimulation. It uses the notation $P \stackrel{\alpha}{\underset{L}{\longrightarrow}} P'$ for $P, P' \in \mathcal{P}$, $\alpha \in A$, and $L \subseteq \underline{A} \setminus \{\underline{\tau}\}$ whenever $\exists m \in \mathcal{L}oc.\, P \stackrel{m,\alpha}{\longrightarrow} P'$ and $\underline{\Pi}_{[m]}(P) \subseteq L$. Note that these enriched transitions take local pre-emption potential into account, thereby avoiding the explicit annotation of transitions with locations.

Definition 6.4.15 (Alternative Distributed Prioritized Strong Bisimulation) *A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is an* alternative distributed prioritized strong bisimulation *if for every $\langle P, Q\rangle \in \mathcal{R}$, $\alpha \in A$, $\underline{\alpha} \in \underline{A}$, and $L \subseteq \underline{A} \setminus \{\underline{\tau}\}$ the following conditions hold.*

1. *$P \stackrel{\alpha}{\Rightarrow} P'$ implies $\exists Q'.\, Q \stackrel{\alpha}{\Rightarrow} Q'$ and $\langle P', Q'\rangle \in \mathcal{R}$.*
2. *$P \stackrel{\alpha}{\underset{L}{\longrightarrow}} P'$ implies $\exists Q'.\, Q \stackrel{\alpha}{\underset{L}{\longrightarrow}} Q'$ and $\langle P', Q'\rangle \in \mathcal{R}$.*

We write $P \simeq_* Q$ if there exists an alternative distributed prioritized strong bisimulation $\mathcal{R}$ such that $\langle P, Q\rangle \in \mathcal{R}$.

Now, we can state the central result of this section.

Theorem 6.4.16 (Operational Characterization) *The coincidence $\simeq^{\mathrm{l}} = \simeq_*$ holds.*

Proof: First, we prove the inclusion "$\subseteq$." Let $P, Q \in \mathcal{P}$ such that $P \simeq^{\mathrm{l}} Q$. In order to establish $P \simeq_* Q$ it is by Definition 6.4.15 sufficient to show that $\simeq^{\mathrm{l}}$ is an alternative distributed prioritized strong bisimulation.

1. Let $P \stackrel{\underline{\alpha}}{\Rightarrow} P'$ for some $P' \in \mathcal{P}$ and $\underline{\alpha} \in \underline{A}$. Because of $P \simeq^{\mathrm{l}} Q$ we know of the existence of some $Q' \in \mathcal{P}$ such that $Q \stackrel{\underline{\alpha}}{\Rightarrow} Q'$ and $P' \simeq^{\mathrm{l}} Q'$, as desired.

2. Let $P \stackrel{\alpha}{\underset{L}{\longrightarrow}} P'$ for some $P' \in \mathcal{P}$, $\alpha \in A$, and $L \subseteq \underline{A} \setminus \{\underline{\tau}\}$. By the definition of $\stackrel{\alpha}{\underset{L}{\longrightarrow}}$ we obtain $P \stackrel{m,\alpha}{\longrightarrow} P'$ and $\underline{\Pi}_{[m]}(P) \subseteq L$ for some $m \in \mathcal{L}oc$. Because of $P \simeq^{\mathrm{l}} Q$ we know of the existence of some $Q' \in \mathcal{P}$ and some $n \in \mathcal{L}oc$ such that $Q \stackrel{n,\alpha}{\longrightarrow} Q'$, $\underline{\Pi}_{[n]}(Q) \subseteq \underline{\Pi}_{[m]}(P)$, and $P' \simeq^{\mathrm{l}} Q'$. Hence, $\underline{\Pi}_{[n]}(Q) \subseteq L$ due to the transitivity of $\subseteq$ and $Q \stackrel{\alpha}{\underset{L}{\longrightarrow}} Q'$ by the definition of $\stackrel{\alpha}{\underset{L}{\longrightarrow}}$.

The other necessary inclusion "$\supseteq$" is established by proving that $\simeq_*$ is a distributed prioritized strong bisimulation (cf. Definition 6.4.2). Let $P, Q \in \mathcal{P}$ such that $P \simeq_* Q$.

1. The case $P \stackrel{\underline{\alpha}}{\Rightarrow} P'$ for some $P' \in \mathcal{P}$ and $\alpha \in \underline{A}$ is again straightforward since Part (1) of Definition 6.4.2 and Part (1) of Definition 6.4.15 coincide.

2. Let $P \stackrel{m,\alpha}{\longrightarrow} P'$ for some $P' \in \mathcal{P}$, $\alpha \in A$, and $m \in \mathcal{L}oc$. By the definition of $\stackrel{\alpha}{\underset{L}{\longrightarrow}}$ we may conclude $P \stackrel{\alpha}{\underset{L}{\longrightarrow}} P'$ for $L = \underline{\Pi}_{[m]}(P)$. Since $P \simeq_* Q$ we know of the existence of some $Q' \in \mathcal{P}$ such that $Q \stackrel{\alpha}{\underset{L}{\longrightarrow}} Q'$ and $P' \simeq_* Q'$. Thus, $Q \stackrel{n,\alpha}{\longrightarrow} Q'$ and $\underline{\Pi}_{[n]}(Q) \subseteq L$ for some $n \in \mathcal{L}oc$. Finally, we obtain $\underline{\Pi}_{[n]}(Q) \subseteq \underline{\Pi}_{[m]}(P)$ by the definition of L, as desired.

Summarizing, we have shown that both relations, $\simeq^{\mathrm{l}}$ and $\simeq_*$, coincide. □

On the basis of the above characterizations we may compute $\simeq^{\mathrm{l}}$ for finite-state processes using the following general technique. First build transition systems for the processes in question that have transitions of the form specified above; then apply a standard bisimulation algorithm. Thus, to determine if $P \simeq^{\mathrm{l}} Q$ we would build transition systems for P and Q having transitions $\stackrel{\underline{\alpha}}{\Rightarrow}$ and $\stackrel{\alpha}{\underset{L}{\longrightarrow}}$ and then use some well-known partition-refinement algorithm [103, 138] to see if P and Q wind up in the same equivalence class.

Regarding efficiency, we begin by noting that the time complexity of the most efficient algorithm by Paige and Tarjan [138] is linear in the size of the transition relations of the considered processes and logarithmic in the size of their state spaces. Also, our enriched transition relation for unprioritized actions is parameterized by a subset of the prioritized visible alphabet of interest, i.e. the union of the finite sorts of the considered processes, leading to a potential exponential blow-up in the number of transitions. This is unlikely to be an issue in practice, however, since most actions used in a system definition are internal and only a few of them remain visible for an external observer. It should be noted that the local view of pre-emption does not allow us to eliminate the prioritized action set parameter of our transition relation, as has been done in Chapter 2 with respect to a priority framework dealing with global pre-emption.

6.4.4 Logical Characterization

In this section we provide a logical characterization of $\simeq^{\mathrm{l}}$ by adapting the well-known Hennessy-Milner logic [125], which is a variant of the modal μ-calculus as presented in

Sections 3.6.3 and 4.5.2 but without fixed point operators, to the (strong) enriched transition relation presented in the previous section. The syntax of our logic is defined by the following BNF where $\underline{\alpha} \in \underline{A}$, $\alpha \in A$, and $L \subseteq \underline{A} \setminus \{\underline{\tau}\}$.

$$\Phi \quad ::= \quad tt \quad | \quad \neg\Phi \quad | \quad \Phi \wedge \Phi \quad | \quad \langle\underline{\alpha}\rangle\Phi \quad | \quad \langle\alpha, L\rangle\Phi$$

The set of all formulas, ranged over by $\Phi, \Psi, \ldots$, is denoted by $\mathcal{F}$. We define the satisfaction relation $\models \subseteq \mathcal{P} \times \mathcal{F}$ between processes and formulas inductively on the structure of formulas as depicted in Table 6.7. Intuitively, every process satisfies tt. A process P satisfies the formula $\Phi \wedge \Psi$ whenever it satisfies both formula, Φ and Ψ. Moreover, $\langle\underline{\alpha}\rangle\Phi$ is satisfied by every process which may engage in an $\underline{\alpha}$-transition to some process that satisfies Φ. Similarly, P satisfies $\langle\alpha, L\rangle\Phi$ if P possesses an α-transition with parameter L to a process satisfying Φ.

Table 6.7: Semantics of the new variant of the Hennessy-Milner logic

		$P \models tt$
$P \models \neg\Phi$	if	not $P \models \Phi$
$P \models \Phi \wedge \Psi$	if	$P \models \Phi$ and $P \models \Psi$
$P \models \langle\underline{\alpha}\rangle\Phi$	if	$\exists P' \in \mathcal{P}.\ P \stackrel{\underline{\alpha}}{\Rightarrow} P'$ and $P' \models \Phi$
$P \models \langle\alpha, L\rangle\Phi$	if	$\exists P' \in \mathcal{P}.\ P \stackrel{\alpha}{\underset{L}{\longrightarrow}} P'$ and $P' \models \Phi$

Theorem 6.4.17 (Logical Characterization of $\simeq^{\mathrm{l}}$)
Let $P, Q \in \mathcal{P}$. Then $P \simeq^{\mathrm{l}} Q$ if and only if $\{\Phi \in \mathcal{F} \mid P \models \Phi\} = \{\Phi \in \mathcal{F} \mid Q \models \Phi\}$.

Most proof parts of this theorem are similar to the corresponding ones presented in [125]. First, we define yet another characterization of distributed prioritized strong bisimulation.

Definition 6.4.18 *Let $\simeq^{\mathrm{l}}{}_0 = \mathcal{P} \times \mathcal{P}$ and $P \simeq^{\mathrm{l}}{}_{i+1} Q$ for some $i \in \mathbb{N}$ if the following properties and their symmetric counterparts hold for all $\underline{\alpha} \in \underline{A}$, $\alpha \in A$, and $L \subseteq \underline{A} \setminus \{\underline{\tau}\}$.*

1. *$P \stackrel{\underline{\alpha}}{\Rightarrow} P'$ implies $\exists Q'.\ Q \stackrel{\underline{\alpha}}{\Rightarrow} Q'$ and $P' \simeq^{\mathrm{l}}{}_i Q'$.*
2. *$P \stackrel{\alpha}{\underset{L}{\longrightarrow}} P'$ implies $\exists Q'.\ Q \stackrel{\alpha}{\underset{L}{\longrightarrow}} Q'$ and $P' \simeq^{\mathrm{l}}{}_i Q'$.*

The proof of the next proposition follows the lines in [125]. Note that for all processes in $\mathcal{P}$ their corresponding transition systems are *finite-branching* (cf. [125]) because $\mathsf{CCS}^{\mathsf{prio}}$ processes are guarded and the summation operators are binary.

Proposition 6.4.19 *Let $P, Q \in \mathcal{P}$. Then we have $P \simeq^{\mathrm{l}} Q$ if and only if $P \simeq^{\mathrm{l}}_i Q$ for all $i \in \mathbb{N}$.*

Now we are able to prove Theorem 6.4.17. By Proposition 6.4.19 it is sufficient to establish the following two lemmata.

Lemma 6.4.20 *Let $P, Q \in \mathcal{P}$, $i \in \mathbb{N}$, and $\Phi \in \mathcal{F}$ such that $P \simeq^{\mathrm{l}}_i Q$ and $P \models \Phi$. Then $Q \models \Phi$ holds.*

Proof: We prove the lemma by induction on i where the induction step is divided into several cases according to the structure of Φ. The only non-standard case is $\Phi \equiv \langle \alpha, L \rangle \Psi$ for $\alpha \in A$ and $L \subseteq \underline{A} \setminus \{\underline{\tau}\}$. By definition of $\models$ we conclude the existence of a process $P' \in \mathcal{P}$ such that $P \xrightarrow[L]{\alpha} P'$ and $P' \models \Psi$. Since $P \simeq^{\mathrm{l}}_i Q$ we also know of the existence of some $Q' \in \mathcal{P}$ such that $Q \xrightarrow[L]{\alpha} Q'$, and $P' \simeq^{\mathrm{l}}_{i-1} Q'$. By induction hypothesis, $Q' \models \Psi$ holds. Therefore, $Q \models \langle \alpha, L \rangle \Psi$, as desired. □

Lemma 6.4.21 *Let $P, Q \in \mathcal{P}$ and $i \in \mathbb{N}$ such that $P \not\simeq^{\mathrm{l}}_i Q$. Then there exists a formula $\Phi \in \mathcal{F}$ satisfying $P \models \Phi$ but $Q \not\models \Phi$.*

Proof: We prove this lemma by induction on i. The induction base is trivial since the premise $P \not\simeq^{\mathrm{l}}_0 Q$ does not hold. Now, let $i > 0$ and $P \not\simeq^{\mathrm{l}}_i Q$. We have to find a formula $\Phi \in \mathcal{P}$ such that $P \models \Phi$ and $Q \not\models \Phi$. Since $P \not\simeq^{\mathrm{l}}_i Q$ we either have $P \stackrel{\underline{\alpha}}{\Rightarrow} P'$ for some $\underline{\alpha} \in \underline{A}$ and $P' \in \mathcal{P}$, or $P \xrightarrow[L]{\alpha} P'$ for some $\alpha \in A$, $L \subseteq \underline{A} \setminus \{\underline{\tau}\}$, and $P' \in \mathcal{P}$. The necessary argumentation for completing the first case follows the standard lines. In the second case we know that whenever $Q \xrightarrow[L]{\alpha} Q'$ then $P' \not\simeq^{\mathrm{l}}_{i-1} Q'$. Let $\{Q' \mid Q \xrightarrow[L]{\alpha} Q'\} = \{Q_j \mid j \in J\}$ for some index set J. By induction hypothesis we conclude the existence of formulas Ψ_j, for $j \in J$, such that $P' \models \Psi_j$ and $Q_j \not\models \Psi_j$. Now, define $\Phi =_{\mathrm{df}} \langle \alpha, L \rangle \bigwedge_{j \in J} \Psi_j$. It is easy to see that $P \models \Phi$. Since no α-derivative of Q parameterized by L satisfies $\bigwedge_{j \in I} \Psi_j$, we also have $Q \not\models \Phi$, as desired. □

6.5 A Semantic Theory based on Weak Bisimulation

The behavioral congruence developed in the previous section is too strong for verifying systems in practice, as it requires that two equivalent terms match each other's transitions exactly, even those labeled by internal actions. In this section we remedy this problem by developing a semantic congruence that abstracts away from internal transitions. Our approach follows the lines of [125, 134]. We start off with the definition of a naive distributed prioritized weak bisimulation which is an adaptation of observational equivalence [125].

Definition 6.5.1 (Naive Distributed Prioritized Weak Transition Relation)

1. $\hat{\gamma} =_{df} \epsilon$, *if* $\gamma \in \{\underline{\tau}, \tau\}$, *and* $\hat{\gamma} =_{df} \gamma$, *otherwise*
2. $\stackrel{\epsilon}{\Longrightarrow}_{\times} =_{df} (\stackrel{\tau}{\rightarrow} \cup \bigcup\{\stackrel{m,\tau}{\longrightarrow} \mid m \in \mathcal{L}oc\})^*$
3. $\stackrel{m,\gamma}{\Longrightarrow}_{\times} =_{df} \stackrel{\epsilon}{\Longrightarrow}_{\times} \circ \stackrel{m,\gamma}{\longrightarrow} \circ \stackrel{\epsilon}{\Longrightarrow}_{\times}$

In the following we write $P \stackrel{\gamma}{\Longrightarrow}_{\times} P'$ for $\exists m \in \mathcal{L}oc.\, P \stackrel{m,\gamma}{\Longrightarrow}_{\times} P'$.

Definition 6.5.2 (Naive Distributed Prioritized Weak Bisimulation)
A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ *is a* naive distributed prioritized weak bisimulation *if for every* $\langle P, Q\rangle \in \mathcal{R}$ *and* $\gamma \in \mathcal{A}$ *the following condition holds.*

$$P \stackrel{\gamma}{\rightarrow} P' \text{ implies } \exists Q'.\, Q \stackrel{\hat{\gamma}}{\Longrightarrow}_{\times} Q' \text{ and } \langle P', Q'\rangle \in \mathcal{R} \ .$$

We write $P \approx_{\times} Q$ *if there exists a naive distributed prioritized weak bisimulation* $\mathcal{R}$ *such that* $\langle P, Q\rangle \in \mathcal{R}$.

It is fairly easy to see that $\approx_{\times}$ is not a congruence for $\mathsf{CCS}^{\mathsf{prio}}$. One compositionality defect arises with respect to parallel composition and is similar to the one mentioned for naive distributed prioritized strong bisimulation. Another defect, which is carried over from CCS, is concerned with the summation operators. Nevertheless, the naive adaptation of weak bisimulation reflects an intuitive approach to abstracting away from internal computation.

6.5.1 Distributed Prioritized Weak Bisimulation

We devote the main part of Section 6.5 to characterizing the largest congruence contained in the naive distributed prioritized weak bisimulation which exists due to Proposition 2.4.3. To do so, we first redefine the weak transition relation as follows.

Definition 6.5.3 (Distributed Prioritized Weak Transition Relation)
For $L, M \subseteq \underline{\mathcal{A}} \setminus \{\underline{\tau}\}$ *we define the following notations.*

1. $\hat{\underline{\tau}} =_{df} \underline{\epsilon}$, $\hat{\underline{a}} =_{df} \underline{a}$, $\hat{\tau} =_{df} \epsilon$, *and* $\hat{a} =_{df} a$
2. $P \stackrel{m,\alpha}{\underset{L}{\longrightarrow}} P'$ *if and only if* $P \stackrel{m,\alpha}{\longrightarrow} P'$ *and* $\underline{\Pi}_{[m]}(P) \subseteq L$
3. $\stackrel{\underline{\epsilon}}{\Longrightarrow} =_{df} (\stackrel{\tau}{\rightarrow} \cup \bigcup\{\stackrel{m,\tau}{\underset{\emptyset}{\longrightarrow}} \mid m \in \mathcal{L}oc\})^*$
4. $\stackrel{\underline{\alpha}}{\Longrightarrow} =_{df} \stackrel{\underline{\epsilon}}{\Longrightarrow} \circ \stackrel{\underline{\alpha}}{\rightarrow} \circ \stackrel{\underline{\epsilon}}{\Longrightarrow}$
5. $\stackrel{\epsilon}{\underset{L}{\Longrightarrow}} =_{df} (\stackrel{\tau}{\rightarrow} \cup \bigcup\{\stackrel{m,\tau}{\underset{L}{\longrightarrow}} \mid m \in \mathcal{L}oc\})^*$
6. $P \stackrel{m,\alpha}{\underset{L,M}{\Longrightarrow}} P'$ *if and only if* $\exists P'', P'''.\, P \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} P'' \stackrel{m,\alpha}{\underset{L}{\longrightarrow}} P''' \stackrel{\underline{\epsilon}}{\Longrightarrow} P'$ *and* $\underline{\Pi}(P'') \subseteq M$.

Intuitively, these definitions are designed to reflect constraints that a process environment must satisfy in order for the given transition to be enabled. Thus, $P \xrightarrow[L]{m,\alpha} P'$ means that P can engage in action α at location m to P' *provided that* the environment does not offer a prioritized communication involving actions in L. If the environment were to offer such a communication, the result would be a $\underline{\tau}$ at a comparable location to m in P, which would pre-empt the α. In a similar vein, $P \stackrel{\epsilon}{\Longrightarrow} P'$ holds if P can evolve to P' via a nonpre-emptable sequence of internal transitions, regardless of the environment's behavior. These internal transitions should therefore involve either $\underline{\tau}$, which can never be pre-empted, or τ, in which case no prioritized actions should be enabled at the same location. Likewise, $P \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} P'$ means that, so long as the environment does not offer to synchronize with P using the prioritized actions in L, the process P may engage in a sequence of internal computation steps and become P'. Finally, the M-parameter in $\stackrel{m,\alpha}{\underset{L,M}{\Longrightarrow}}$ provides a measure of the pre-emptive impact that a process can have on its environment. From the definition, $P \stackrel{m,\alpha}{\underset{L,M}{\Longrightarrow}} P'$ is true if P can engage in some internal computation followed by α, so long as the environment refrains from synchronizations in L, and then some nonpre-emptable internal computation to arrive at P'. In addition, the state at which α is enabled should only offer prioritized communications in M. To understand the role played by M, consider the processes $P \equiv (a.\mathbf{0} + \underline{b}.\mathbf{0}) \mid \underline{c}.\mathbf{0}$ and $Q \equiv \overline{a}.\mathbf{0} + \overline{\underline{c}}.\mathbf{0}$. If one were to define $\stackrel{m,\alpha}{\underset{L}{\Longrightarrow}}$ in the obvious manner, one would conclude that $P \stackrel{lL,a}{\underset{\{\underline{b}\}}{\Longrightarrow}} P' \equiv \mathbf{0} \mid \underline{c}.\mathbf{0}$. Since Q offers a communication $\overline{a}$ and no interaction $\overline{\underline{b}}$, one might then be tempted to infer that $P \mid Q \xrightarrow{\langle lL,lR\rangle,\tau} P' \mid Q' \equiv (\mathbf{0} \mid \underline{c}.\mathbf{0}) \mid \mathbf{0}$. However, the operational semantics disallows this; as P and Q can synchronize on $\underline{c}$, Q's $\overline{a}$-transition becomes pre-empted, even though P's a-transition is not (because its location is incomparable with P's $\underline{c}$-transition). On the other hand, $P \stackrel{lL,a}{\underset{\{\underline{b}\},\{\underline{c}\}}{\Longrightarrow}} P'$ alerts us to P's pre-emptive capability on $\underline{c}$.

Note that the definition of $P \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} P'$ corresponds with our intuition that internal actions, and therefore their locations, are unobservable. Moreover, an environment of P is not influenced by internal actions performed by P since priorities arising from different sides of the parallel composition operator are incomparable. Therefore, the parameter M is unnecessary in the definition of the relation $\stackrel{\epsilon}{\underset{L}{\Longrightarrow}}$. Finally, for notational convenience we interpret $\stackrel{m,\epsilon}{\underset{L,M}{\Longrightarrow}}$ as $\stackrel{\epsilon}{\underset{L}{\Longrightarrow}}$, and we write $P \xrightarrow{\epsilon} P$ and $P \xrightarrow[L]{\epsilon} P$ for any $P \in \mathcal{P}$, $m \in \mathcal{L}oc$, and $L, M \subseteq \underline{A} \setminus \{\underline{\tau}\}$.

Definition 6.5.4 (Distributed Prioritized Weak Bisimulation)
A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a distributed prioritized weak bisimulation *if for every $\langle P, Q\rangle \in \mathcal{R}$, $\alpha \in A$, $\underline{\alpha} \in \underline{A}$, and $m \in \mathcal{L}oc$ the following conditions hold.*

1. $\exists Q', Q''.\, Q \stackrel{\epsilon}{\Longrightarrow} Q'' \stackrel{\underline{\alpha}}{\Longrightarrow} Q',\ \underline{\Pi}(Q'') \subseteq \underline{\Pi}(P),$ *and* $\langle P, Q'\rangle \in \mathcal{R}$.
2. $P \stackrel{\underline{\alpha}}{\Rightarrow} P'$ *implies* $\exists Q'.\, Q \stackrel{\hat{\underline{\alpha}}}{\Longrightarrow} Q'$ *and* $\langle P', Q'\rangle \in \mathcal{R}$.
3. $P \xrightarrow{m,\alpha} P'$ *implies* $\exists Q', n.\, Q \stackrel{n,\hat{\alpha}}{\underset{L,M}{\Longrightarrow}} Q',\ L = \underline{\Pi}_{[m]}(P),\ M = \underline{\Pi}(P),$ *and* $\langle P', Q'\rangle \in \mathcal{R}$.

We write $P \underline{\approx} Q$ *if there exists a distributed prioritized weak bisimulation* $\mathcal{R}$ *such that* $\langle P, Q\rangle \in \mathcal{R}$.

From this definition we may directly conclude that $\underline{\approx}$ is the *largest* distributed prioritized weak bisimulation and that $\underline{\approx}$ is an equivalence relation. Condition (1) of Definition 6.5.4 guarantees that distributed prioritized weak bisimulation is compositional with respect to parallel composition. Its necessity is best illustrated by the following example. The processes $P \stackrel{\text{def}}{=} \underline{\tau}.\underline{a}.\mathbf{0}$ and $Q \stackrel{\text{def}}{=} \underline{a}.\mathbf{0}$ would be considered equivalent if Condition (1) were absent. However, the context $C[X] \stackrel{\text{def}}{=} X \,|\, (\overline{\underline{a}}.\mathbf{0} + b.\mathbf{0})$ can distinguish them. The following proposition is the $\mathsf{CCS}^{\mathsf{prio}}$ equivalent of Proposition 2.4.6 stated within the $\mathsf{CCS}^{\mathsf{ch}}$ framework.

Proposition 6.5.5 *The equivalence relation* $\underline{\approx}$ *is a congruence with respect to prefixing, parallel composition, relabeling, and restriction. Moreover,* $\underline{\approx}$ *is characterized as the* largest congruence *contained in* $\approx_{\times}$, *in the sub-algebra of* $\mathsf{CCS}^{\mathsf{prio}}$ *induced by these operators and recursion.*

In the remainder of this section we prove the first part of Proposition 6.5.5 which states that $\underline{\approx}$ is compositional for the operators prefixing, parallel composition, relabeling, and restriction. Many cases are standard. The only interesting non-standard case of the proof is the compositionality of $\underline{\approx}$ with respect to parallel composition which we give in full detail. The second part of the proposition, which claims that $\underline{\approx}$ is characterized as the *largest congruence* contained in $\approx_{\times}$, in the sub-algebra of $\mathsf{CCS}^{\mathsf{prio}}$ induced by the above mentioned operators and recursion, follows by the results of this section and an inspection of the proof of Proposition 6.5.14 presented below. We first state a lemma which helps us to reduce the notational complexity in the remainder.

Lemma 6.5.6 *Let* $P, P', Q \in \mathcal{P}$, *and* $L \subseteq \underline{A}$. *Then*

1. $P \stackrel{\epsilon}{\Longrightarrow} P'$ *implies* $P \,|\, Q \stackrel{\epsilon}{\Longrightarrow} P' \,|\, Q$,
2. $Q \stackrel{\epsilon}{\Longrightarrow} Q'$ *implies* $P \,|\, Q \stackrel{\epsilon}{\Longrightarrow} P \,|\, Q'$,
3. $P \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} P'$ *and* $L \cap \overline{\underline{\mathcal{L}}(Q)} = \emptyset$ *implies* $P \,|\, Q \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} P' \,|\, Q$, *and*
4. $Q \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} Q'$ *and* $L \cap \overline{\underline{\mathcal{L}}(P)} = \emptyset$ *implies* $P \,|\, Q \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} P \,|\, Q'$.

Proof: Let $P, P', Q \in \mathcal{P}$ and $L \subseteq \underline{A}$.

1. The proof is done by induction on the length i of the transition $P \stackrel{\epsilon}{\Longrightarrow} P'$. The induction base, i.e. $i = 0$ is trivial. Therefore, we directly consider the induction step, i.e. $i > 0$, where we distinguish the following cases according to the definition of $\stackrel{\epsilon}{\Longrightarrow}$.

- *Case 1:*

  $P \xrightarrow{\tau} P'' \stackrel{\epsilon}{\Longrightarrow} P'$ for some $P'' \in \mathcal{P}$ where the transition $P'' \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} P'$ has length $i-1$. Thus, $P \mid Q \xrightarrow{\tau} P'' \mid Q \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} P' \mid Q$ by Rule (Com1) and the induction hypothesis.

- *Case 2:*

  $P \stackrel{m,\tau}{\underset{\emptyset}{\longrightarrow}} P'' \stackrel{\epsilon}{\Longrightarrow} P'$ for some $P'' \in \mathcal{P}$ and some $m \in \mathcal{L}oc$, i.e. $P \xrightarrow{m,\tau} P''$, $\underline{\Pi}_{[m]}(P) \subseteq \emptyset$, and the transition $P'' \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} P'$ has length $i-1$. Thus, $\underline{\Pi}_{[m]}(P) \cap \overline{\underline{\Pi}(Q)} \subseteq \emptyset \cap \overline{\underline{\Pi}(Q)} = \emptyset$. By Rule (Com1) and the definition of distributed prioritized initial action sets together with the definition of the comparability relation we obtain $P \mid Q \xrightarrow{m \cdot L,\tau} P'' \mid Q$ and $\underline{\Pi}_{[m \cdot L]}(P \mid Q) = \underline{\Pi}_{[m]}(P) \subseteq \emptyset$. Applying the induction hypothesis, $P \mid Q \stackrel{m \cdot L,\tau}{\underset{\emptyset}{\longrightarrow}} P'' \mid Q \stackrel{\epsilon}{\Longrightarrow} P' \mid Q$ holds.

  Hence, $P \mid Q \stackrel{\epsilon}{\Longrightarrow} P' \mid Q$ by the definition of $\stackrel{\epsilon}{\Longrightarrow}$, as desired.

2. The proof of this statement is analogous to the previous one and uses Rules (Com2) and (Com2) instead of Rules (Com1) and (Com1), respectively.

3. Let $P \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} P'$ and $L \cap \overline{\underline{\Pi}(Q)} = \emptyset$. We proceed by induction on the length i of the transition $\stackrel{\epsilon}{\underset{L}{\Longrightarrow}}$. If $i = 0$ then $P \equiv P'$ and the statement is trivial. For the induction step let $i > 0$. We have to distinguish the following cases according to the definition of $\stackrel{\epsilon}{\underset{L}{\Longrightarrow}}$.

   - *Case 1:*

     $P \xrightarrow{\tau} P'' \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} P'$ for some $P'' \in \mathcal{P}$ where the transition $P'' \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} P'$ has length $i-1$. Hence, $P \mid Q \xrightarrow{\tau} P'' \mid Q \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} P' \mid Q$ by Rule (Com1) and the induction hypothesis.

   - *Case 2:*

     $P \stackrel{m,\tau}{\underset{L}{\longrightarrow}} P'' \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} P'$ for some $P'' \in \mathcal{P}$ and some $m \in \mathcal{L}oc$, i.e. $P \xrightarrow{m,\tau} P''$, $\underline{\Pi}_{[m]}(P) \subseteq L$, and the transition $P'' \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} P'$ has length $i-1$. Hence, $\underline{\Pi}_{[m]}(P) \cap \overline{\underline{\Pi}(Q)} \subseteq L \cap \overline{\underline{\Pi}(Q)} = \emptyset$ due to our premise. By Rule (Com1) and the definition of distributed prioritized initial action sets plus the definition of the comparability relation we obtain $P \mid Q \xrightarrow{m \cdot L,\tau} P'' \mid Q$ and $\underline{\Pi}_{[m \cdot L]}(P \mid Q) = \underline{\Pi}_{[m]}(P) \subseteq L$. Thus, $P \mid Q \stackrel{m \cdot L,\tau}{\underset{L}{\longrightarrow}} P'' \mid Q \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} P' \mid Q$ can be derived by additionally using the induction hypothesis.

   In both cases we obtain $P \mid Q \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} P' \mid Q$ by the definition of $\stackrel{\epsilon}{\underset{L}{\Longrightarrow}}$, as desired.

4. The proof can be done symmetrically to the previous one by using Rules (Com2) and (Com2) instead of Rules (Com1) and (Com1), respectively. □

Now we can prove the compositionality of distributed prioritized weak bisimulation with respect to parallel composition.

Proposition 6.5.7 *$P \underline{\approx} Q$ implies $P \mid R \underline{\approx} Q \mid R$ for all processes R.*

Proof: According to Definition 6.5.4 it is sufficient to prove that

$$\mathcal{R} =_{\mathrm{df}} \{\langle P \mid R, Q \mid R\rangle \mid P \underline{\approx} Q\}$$

is a distributed prioritized weak bisimulation. Let $P \mid R \xrightarrow{p,\alpha} V$ for some $V \in \mathcal{P}$ and $p \in \mathcal{L}oc$. We have to show the existence of some $W \in \mathcal{P}$ and $q \in \mathcal{L}oc$ such that $Q \mid R \overset{q,\hat{\alpha}}{\underset{L,M}{\Longrightarrow}} W$ and $\langle V, W\rangle \in \mathcal{R}$ where $L = \underline{\Pi}_{[p]}(P \mid R)$ and $M = \underline{\Pi}(P \mid R)$. Consider the following case distinction according to the operational rules for parallel composition.

1. Let $P \xrightarrow{m,\alpha} P'$ and $\underline{\Pi}_{[m]}(P) \cap \overline{\underline{\Pi}(R)} = \emptyset$, i.e. $V \equiv P' \mid R$ and $p \equiv m \cdot L$. Since $P \underline{\approx} Q$ there exist a process $Q' \in \mathcal{P}$ and a location $n \in \mathcal{L}oc$ satisfying $Q \overset{n,\hat{\alpha}}{\underset{L',M'}{\Longrightarrow}} Q'$, $L' = \underline{\Pi}_{[m]}(P)$, $M' = \underline{\Pi}(P)$, and $P' \underline{\approx} Q'$. According to the definition of the distributed prioritized weak transition relation we have the following for some $Q'', Q''' \in \mathcal{P}$.

   (a) $Q \overset{\epsilon}{\underset{L'}{\Longrightarrow}} Q''$,

   (b) $Q'' \overset{n,\hat{\alpha}}{\underset{L'}{\longrightarrow}} Q'''$ and $\underline{\Pi}(Q'') \subseteq M'$, i.e. $Q'' \equiv Q'''$, if $\alpha \equiv \tau$, and $Q'' \xrightarrow{n,\alpha} Q'''$ and $\underline{\Pi}_{[n]}(Q'') \subseteq L'$, otherwise, and

   (c) $Q''' \overset{\epsilon}{\Longrightarrow} Q'$.

   Now, we may derive $Q \mid R \overset{n\cdot L,\hat{\alpha}}{\underset{L,M}{\Longrightarrow}} Q' \mid R$ as follows.

   (a) $Q \mid R \overset{\epsilon}{\underset{L'}{\Longrightarrow}} Q'' \mid R$ by Lemma 6.5.6(3) since $L' \cap \overline{\underline{\Pi}(R)} = \underline{\Pi}_{[m]}(P) \cap \overline{\underline{\Pi}(R)} = \emptyset$.

   (b) $Q'' \mid R \overset{n\cdot L,\hat{\alpha}}{\underset{L}{\longrightarrow}} Q''' \mid R$ by Rule (Com1) for $L = \underline{\Pi}_{[m\cdot L]}(P \mid R)$ due to the following facts, where we assume $\alpha \not\equiv \tau$ since the other case is trivial.

      - $\underline{\Pi}_{[n]}(Q'') \cap \overline{\underline{\Pi}(R)} \subseteq \underline{\Pi}_{[m]}(P) \cap \overline{\underline{\Pi}(R)} = \emptyset$ by premise, and
      - $\underline{\Pi}_{[n\cdot L]}(Q'' \mid R) \subseteq \underline{\Pi}_{[m\cdot L]}(P \mid R)$ by Lemma 6.3.3(1).

      Moreover, $\underline{\Pi}(Q'' \mid R) \subseteq M$ by Lemma 6.3.3(4).

   (c) $Q''' \mid R \overset{\epsilon}{\Longrightarrow} Q' \mid R$ by Lemma 6.5.6(1).

   By choosing $W \equiv Q' \mid R$ and $q \equiv n \cdot L$, and since $\langle P' \mid R, Q' \mid R\rangle \in \mathcal{R}$ we are done.

2. Let $R \xrightarrow{o,\alpha} R'$ and $\underline{\Pi}_{[o]}(R) \cap \overline{\underline{\Pi}(P)} = \emptyset$, i.e. $V \equiv P \mid R'$ and $p \equiv o \cdot R$. Since $P \underline{\approx} Q$ there exist processes $Q', Q'' \in \mathcal{P}$ such that $Q \overset{\epsilon}{\Longrightarrow} Q'' \overset{\epsilon}{\Longrightarrow} Q'$, $\underline{\Pi}(Q'') \subseteq \underline{\Pi}(P)$, and $P \underline{\approx} Q'$. Hence,

   (a) $Q \mid R \overset{\epsilon}{\Longrightarrow} Q'' \mid R$ by Lemma 6.5.6(1).

(b) $Q'' \mid R \xrightarrow[L]{o\cdot R,\alpha} Q'' \mid R'$ by Rule (Com2) because of the following.

- $\underline{\mathit{\Pi}}_{[o]}(R) \cap \overline{\underline{\mathit{\Pi}}(Q'')} \subseteq \underline{\mathit{\Pi}}_{[o]}(R) \cap \overline{\underline{\mathit{\Pi}}(P)} = \emptyset$ since $\underline{\mathit{\Pi}}(Q'') \subseteq \underline{\mathit{\Pi}}(P)$, and
- $\underline{\mathit{\Pi}}_{[o\cdot R]}(Q'' \mid R) \subseteq L$ by Lemma 6.3.3(2) and the definition of L.

Also, $\underline{\mathit{\Pi}}(Q'' \mid R) \subseteq M$ according to Lemma 6.3.3(4) and the definition of M.

(c) $Q'' \mid R' \stackrel{\epsilon}{\Longrightarrow} Q' \mid R'$ by Lemma 6.5.6(1).

According to the definition of the distributed prioritized weak transition relation, we have shown that $Q \mid R \overset{o\cdot R,\hat{\alpha}}{\underset{L,M}{\Longrightarrow}} Q' \mid R'$. By choosing $W \equiv Q' \mid R'$ and $q \equiv o \cdot R$ we have finished this case, because also $\langle P \mid R', Q' \mid R' \rangle \in \mathcal{R}$ holds due to the definition of $\mathcal{R}$.

3. Let $P \xrightarrow{m,a} P'$, $R \xrightarrow{o,\overline{a}} R'$, $\underline{\mathit{\Pi}}_{[m]}(P) \cap \overline{\underline{\mathit{\Pi}}(R)} = \emptyset$, and $\underline{\mathit{\Pi}}_{[o]}(R) \cap \overline{\underline{\mathit{\Pi}}(P)} = \emptyset$ for some $a \in A \setminus \{\tau\}$, i.e. $V \equiv P' \mid R'$ and $p \equiv \langle m \cdot L, o \cdot R \rangle$. Since $P \underline{\approx} Q$ there exist $Q' \in \mathcal{P}$ and $n \in \mathcal{L}oc$ satisfying $Q \overset{n,a}{\underset{L',M'}{\Longrightarrow}} Q'$, $L' = \underline{\mathit{\Pi}}_{[m]}(P)$, $M' = \underline{\mathit{\Pi}}(P)$, and $P' \underline{\approx} Q'$. According to the definition of the distributed prioritized weak transition relation, we have the following situation for some $Q'', Q''' \in \mathcal{P}$.

(a) $Q \overset{\epsilon}{\underset{L'}{\Longrightarrow}} Q''$,

(b) $Q'' \overset{n,a}{\underset{L'}{\longrightarrow}} Q'''$, i.e. $Q'' \xrightarrow{n,a} Q'''$, $\underline{\mathit{\Pi}}_{[n]}(Q'') \subseteq L'$, and $\underline{\mathit{\Pi}}(Q'') \subseteq M'$, and

(c) $Q''' \stackrel{\epsilon}{\Longrightarrow} Q'$.

Now, we may derive $Q \mid R \overset{q,\tau}{\underset{L,M}{\Longrightarrow}} Q' \mid R'$, where $q \equiv \langle n \cdot L, o \cdot R \rangle$, as follows.

(a) $Q \mid R \overset{\epsilon}{\underset{L'}{\Longrightarrow}} Q'' \mid R$ by Lemma 6.5.6(3) since $L' \cap \overline{\underline{\mathit{\Pi}}(R)} = \underline{\mathit{\Pi}}_{[m]}(P) \cap \overline{\underline{\mathit{\Pi}}(R)} = \emptyset$.

(b) $Q'' \mid R \overset{q,\tau}{\underset{L}{\longrightarrow}} Q''' \mid R'$ by Rule (Com3) due to the following facts.

- $\underline{\mathit{\Pi}}_{[n]}(Q'') \cap \overline{\underline{\mathit{\Pi}}(R)} \subseteq \underline{\mathit{\Pi}}_{[m]}(P) \cap \overline{\underline{\mathit{\Pi}}(R)} = \emptyset$ by premise,
- $\underline{\mathit{\Pi}}_{[o]}(R) \cap \overline{\underline{\mathit{\Pi}}(Q'')} \subseteq \underline{\mathit{\Pi}}_{[o]}(R) \cap \overline{M'} = \underline{\mathit{\Pi}}_{[o]}(R) \cap \overline{\underline{\mathit{\Pi}}(P)} = \emptyset$ by premise,
- $\underline{\mathit{\Pi}}_{[\langle n\cdot L, o\cdot R\rangle]}(Q'' \mid R) \subseteq \underline{\mathit{\Pi}}_{[\langle m\cdot L, o\cdot R\rangle]}(P \mid R)$ by Lemma 6.3.3(3), and
- $\underline{\mathit{\Pi}}(Q'' \mid R) \subseteq \underline{\mathit{\Pi}}(P \mid R)$ by Lemma 6.3.3(4).

(c) $Q''' \mid R' \stackrel{\epsilon}{\Longrightarrow} Q' \mid R'$ by Lemma 6.5.6(1).

Hence, $Q \mid R \overset{\epsilon}{\underset{L}{\Longrightarrow}} Q' \mid R'$ by our notational conventions. We finish by choosing $W \equiv Q' \mid R'$ and by observing that $\langle P' \mid R', Q' \mid R' \rangle \in \mathcal{R}$ according to the definition of $\mathcal{R}$.

For establishing the first condition of Definition 6.5.4 we argue as follows. Since $P \underline{\approx} Q$ there exist $Q', Q'' \in \mathcal{P}$ such that $Q \stackrel{\epsilon}{\Longrightarrow} Q'' \stackrel{\epsilon}{\Longrightarrow} Q'$, $\underline{\mathit{\Pi}}(Q'') \subseteq \underline{\mathit{\Pi}}(P)$, and $P \underline{\approx} Q'$. Hence, $Q \mid R \stackrel{\epsilon}{\Longrightarrow} Q'' \mid R \stackrel{\epsilon}{\Longrightarrow} Q' \mid R$ by Lemma 6.5.6(1). Moreover, we have $\langle P \mid R, Q' \mid R \rangle \in \mathcal{R}$ according to the definition of $\mathcal{R}$ and $\underline{\mathit{\Pi}}(Q'' \mid R) = \underline{\mathit{\Pi}}(Q'') \cup \underline{\mathit{\Pi}}(R) \subseteq \underline{\mathit{\Pi}}(P) \cup \underline{\mathit{\Pi}}(R) = \underline{\mathit{\Pi}}(P \mid R)$.

The second condition of Definition 6.5.4 is standard since the semantics for the parallel composition operator with respect to prioritized transitions coincides with the corresponding CCS semantics. □

In order to show that $\underline{\approx}$ is compositional with respect to recursion, we need to define a notion of *distributed prioritized weak bisimulation up to* $\underline{\approx}$ (cf. [149]).

Definition 6.5.8 (Distributed Prioritized Weak Bisimulation up to $\underline{\approx}$)
A relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a distributed prioritized weak bisimulation up to $\underline{\approx}$ *if for every $\langle P, Q\rangle \in \mathcal{R}$, $\alpha \in A$, $\underline{\alpha} \in \underline{A}$, and $m \in \mathcal{L}oc$ the following conditions and their symmetric counterparts hold.*

1. $\exists P', P''.\, P \stackrel{\epsilon}{\Longrightarrow} P'' \stackrel{\underline{\epsilon}}{\Longrightarrow} P'$, $\underline{\mathit{\Pi}}(P'') \subseteq \underline{\mathit{\Pi}}(Q)$, *and* $P' \,\mathcal{R} \underline{\approx}\, Q$.
2. $P \stackrel{\alpha}{\rightarrow} P'$ *implies* $\exists Q'.\, Q \stackrel{\hat{\underline{\alpha}}}{\Longrightarrow} Q'$ *and* $P' \,\mathcal{R} \underline{\approx}\, Q'$.
3. $P \stackrel{m,\alpha}{\longrightarrow} P'$ *implies* $\exists Q', n.\, Q \stackrel{n,\hat{\underline{\alpha}}}{\underset{L,M}{\Longrightarrow}} Q'$, $L = \underline{\mathit{\Pi}}_{[m]}(P)$, $M = \underline{\mathit{\Pi}}(P)$, *and* $P' \,\mathcal{R} \underline{\approx} Q'$.

This notion satisfies the property that, if $\mathcal{R}$ is a *distributed prioritized weak bisimulation up to* $\underline{\approx}$, then $\mathcal{R} \subseteq \underline{\approx}$. Using the above definition the proof can be done in a similar fashion as in [125]. However, the following lemma turns out to be important. Its proof explains from a technical point of view why the M-parameter in the definition of the distributed prioritized weak transition relation is needed.

Lemma 6.5.9 *Let $P, P', Q, Q', R, S \in \mathcal{P}$ and $m, n \in \mathcal{L}oc$ such that $P \mid Q \stackrel{\langle m\cdot L, n\cdot R\rangle,\tau}{\longrightarrow} P' \mid Q'$, $P \underline{\approx} R$, and $Q \underline{\approx} S$. Then there exist $R', S' \in \mathcal{P}$ and $o \in \mathcal{L}oc$ such that $R \mid S \stackrel{o,\tau}{\underset{L,M}{\Longrightarrow}} R' \mid S'$ where $L = \underline{\mathit{\Pi}}_{[\langle m\cdot L, n\cdot R\rangle]}(P \mid Q)$ and $M = \underline{\mathit{\Pi}}(P \mid Q)$.*

Proof: Let $P \underline{\approx} R$ and $Q \underline{\approx} S$. Moreover, let $P \mid Q \stackrel{\langle m\cdot L, n\cdot R\rangle,\tau}{\longrightarrow} P' \mid Q'$ for some $P', Q' \in \mathcal{P}$ and $m, n \in \mathcal{L}oc$. By the only applicable Rule (Com3) we obtain $P \stackrel{m,a}{\longrightarrow} P'$, $Q \stackrel{n,\overline{a}}{\longrightarrow} Q'$ for some $a \in A \setminus \{\tau\}$, $\underline{\mathit{\Pi}}_{[m]}(P) \cap \overline{\underline{\mathit{\Pi}}(Q)} = \emptyset$, and $\underline{\mathit{\Pi}}_{[n]}(Q) \cap \overline{\underline{\mathit{\Pi}}(P)} = \emptyset$. Now, consider the following.

- Since $P \underline{\approx} R$ we know of the existence of some $\overline{R}, \overline{R}' \in \mathcal{P}$ such that $R \stackrel{\epsilon}{\Longrightarrow} \overline{R}' \stackrel{\underline{\epsilon}}{\Longrightarrow} \overline{R}$, $\underline{\mathit{\Pi}}(\overline{R}') \subseteq \underline{\mathit{\Pi}}(P)$, and $P \underline{\approx} \overline{R}$.
- Due to $P \underline{\approx} \overline{R}$ there exist some $R' \in \mathcal{P}$ and some $m' \in \mathcal{L}oc$ such that $\overline{R} \stackrel{m',a}{\underset{L',M'}{\Longrightarrow}} R'$ where $L' = \underline{\mathit{\Pi}}_{[m]}(P)$ and $M' = \underline{\mathit{\Pi}}(P)$, i.e. $\overline{R} \stackrel{\epsilon}{\underset{L'}{\Longrightarrow}} R'' \stackrel{m',a}{\underset{L'}{\longrightarrow}} R''' \stackrel{\underline{\epsilon}}{\Longrightarrow} R'$ for some $R'', R''' \in \mathcal{P}$ such that $\underline{\mathit{\Pi}}(R'') \subseteq M'$.
- Because of $Q \underline{\approx} S$ there exist $S' \in \mathcal{P}$ and $n' \in \mathcal{L}oc$ such that $S \stackrel{n',\overline{a}}{\underset{L'',M''}{\Longrightarrow}} S'$, where $L'' = \underline{\mathit{\Pi}}_{[n]}(Q)$ and $M'' = \underline{\mathit{\Pi}}(Q)$, i.e. $S \stackrel{\epsilon}{\underset{L''}{\Longrightarrow}} S'' \stackrel{n',\overline{a}}{\underset{L''}{\longrightarrow}} S''' \stackrel{\underline{\epsilon}}{\Longrightarrow} S'$ for some $S'', S''' \in \mathcal{P}$ such that $\underline{\mathit{\Pi}}(S'') \subseteq M''$.

Therefore, we obtain for $L = \underline{\Pi}_{[\langle m \cdot L, n \cdot R\rangle]}(P \mid Q)$ and $M = \underline{\Pi}(P \mid Q)$ the following; note that $L' \subseteq L$ and $L'' \subseteq L$ since $L = \underline{\Pi}_{[\langle m \cdot L, n \cdot R\rangle]}(P \mid Q) = \underline{\Pi}_{[m]}(P) \cup \underline{\Pi}_{[n]}(Q) = L' \cup L''$.

1. $R \mid S \stackrel{\epsilon}{\Longrightarrow} \overline{R}' \mid S$ by Lemma 6.5.6(1).
2. $\overline{R}' \mid S \stackrel{\epsilon}{\underset{L''}{\Longrightarrow}} \overline{R}' \mid S''$ by Lemma 6.5.6(4) since $L'' \cap \overline{\underline{\Pi}(\overline{R}')} \subseteq \underline{\Pi}_{[n]}(Q) \cap \overline{\underline{\Pi}(P)} = \emptyset$. Moreover, $\overline{R}' \mid S \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} \overline{R}' \mid S''$ since $L'' \subseteq L$.
3. $\overline{R}' \mid S'' \stackrel{\epsilon}{\Longrightarrow} \overline{R} \mid S''$ by Lemma 6.5.6(1).
4. $\overline{R} \mid S'' \stackrel{\epsilon}{\underset{L'}{\Longrightarrow}} R'' \mid S''$ by Lemma 6.5.6(3) since $L' \cap \overline{\underline{\Pi}(S'')} \subseteq \underline{\Pi}_{[m]}(P) \cap \overline{\underline{\Pi}(Q)} = \emptyset$. Because of $L' \subseteq L$ also $\overline{R} \mid S'' \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} R'' \mid S''$ holds.
5. $R'' \mid S'' \stackrel{o,\tau}{\underset{L}{\longrightarrow}} R''' \mid S'''$, where $o =_{\mathrm{df}} \langle m' \cdot L, n' \cdot R\rangle$, since
   - $\underline{\Pi}_{[m']}(R'') \cap \overline{\underline{\Pi}(S'')} \subseteq \underline{\Pi}_{[m]}(P) \cap \overline{\underline{\Pi}(Q)} = \emptyset$,
   - $\underline{\Pi}_{[n']}(S'') \cap \overline{\underline{\Pi}(R'')} \subseteq \underline{\Pi}_{[n]}(Q) \cap \overline{\underline{\Pi}(P)} = \emptyset$, and
   - $\underline{\Pi}_{[\langle m' \cdot L, n' \cdot R\rangle]}(R'' \mid S'') = \underline{\Pi}_{[m']}(R'') \cup \underline{\Pi}_{[n']}(S'') \subseteq L' \cup L'' = L$.

   Moreover, $\underline{\Pi}(R'' \mid S'') = \underline{\Pi}(R'') \cup \underline{\Pi}(S'') \subseteq M' \cup M'' = M$.
6. $R''' \mid S''' \stackrel{\epsilon}{\Longrightarrow} R' \mid S'''$ by Lemma 6.5.6(1).
7. $R' \mid S''' \stackrel{\epsilon}{\Longrightarrow} R' \mid S'$ by Lemma 6.5.6(2).

Hence, $R \mid S \stackrel{o,\tau}{\underset{L,M}{\Longrightarrow}} R' \mid S'$ according to the definition of the distributed prioritized weak transition relation. □

6.5.2 Distributed Prioritized Observational Congruence

Analogously to Chapter 2, the summation fix presented in [125] is not sufficient in order to achieve a congruence relation. For an illustrating example the reader might adapt the one given at the beginning of Section 2.4.2. As for prioritized observational congruence we have to require that distributed prioritized observationally congruent processes possess the same distributed prioritized initial actions.

Definition 6.5.10 (Distributed Prioritized Observational Congruence)
We define $P \underline{\approx}^{\downarrow} Q$ if for all $\alpha \in A$, $\underline{\alpha} \in \underline{A}$, and $m \in \mathcal{L}oc$ the following conditions and their symmetric counterparts hold.

1. $\underline{\mathcal{I}}(P) \supseteq \underline{\mathcal{I}}(Q)$
2. $P \stackrel{\underline{\alpha}}{\Rightarrow} P'$ *implies* $\exists Q'.\, Q \stackrel{\underline{\alpha}}{\Longrightarrow} Q'$ *and* $P' \underline{\approx} Q'$.

3. $P \xrightarrow{m,\alpha} P'$ *implies* $\exists Q', n.\, Q \overset{n,\alpha}{\underset{L,M}{\Longrightarrow}} Q',\ L = \underline{\mathcal{I}}_{[m]}(P),\ M = \underline{\mathcal{I}}(P),$ *and* $P' \underline{\approx} Q'$.

Now, we can state the main result of this section.

Theorem 6.5.11 *Distributed prioritized observational congruence* $\underline{\approx}^{\mathrm{l}}$ *is the largest congruence contained in naive distributed prioritized weak bisimulation* $\approx_{\times}$. *Therefore,* $\approx_{\times}^{+} = \underline{\approx}^{\mathrm{l}}$ *holds.*

The proof of the congruence property can be done by using standard techniques [125]. Most cases are straightforward. Also the compositionality proof of $\underline{\approx}^{\mathrm{l}}$ with respect to parallel composition is omitted here since it follows by inspection of the proof of Proposition 6.5.5. The only difference is that a $\underline{\tau}$- or τ-transition is always matched by a distributed prioritized weak $\underline{\tau}$- or τ-transition, respectively. The much stricter initial action set condition in the definition of $\underline{\approx}^{\mathrm{l}}$ follows easily by using the definition of initial action sets. Additionally, we want to remark that the case dealing with the distributed summation operator $\oplus$ is similar to the one for the summation operator in CCS. In the following, we present the compositionality result of $\underline{\approx}^{\mathrm{l}}$ with respect to summation.

Proposition 6.5.12 $P \underline{\approx}^{\mathrm{l}} Q$ *implies* $P + R \underline{\approx}^{\mathrm{l}} Q + R$ *for all processes* R.

Proof: Let $P, Q \in \mathcal{P}$ satisfying $P \underline{\approx}^{\mathrm{l}} Q$. According to Definition 6.5.4 it is sufficient to prove that $P + R \xrightarrow{p,\alpha} V$ implies that

$$\exists W, q.\, Q + R \overset{q,\alpha}{\underset{L,M}{\Longrightarrow}} W,\ L = \underline{\mathcal{I}}_{[p]}(P + R),\ M = \underline{\mathcal{I}}(P + R),\ \text{and}\ V \underline{\approx} W$$

holds, since Property (1) of Definition 6.5.10 follows directly by Definition 6.5.10 for $P \underline{\approx}^{\mathrm{l}} Q$ and Definition 6.3.1, and Property (2) is analogous to the one of *observational congruence* in CCS.

It remains to show the existence of some $W \in \mathcal{P}$ and $q \in \mathcal{L}oc$ such that $Q + R \overset{q,\alpha}{\underset{L,M}{\Longrightarrow}} W$ and $V \underline{\approx} W$. Therefore, consider the following case distinction according to the operational rules for summation.

1. Let $P \xrightarrow{m,\alpha} P'$ and $\underline{\tau} \notin \underline{\mathcal{I}}(R)$, i.e. $p \equiv m \cdot l$ and $V \equiv P'$. Since $P \underline{\approx}^{\mathrm{l}} Q$ we know

$$\exists Q', n.\, Q \overset{n,\alpha}{\underset{L',M'}{\Longrightarrow}} Q',\ L' = \underline{\mathcal{I}}_{[m]}(P),\ M' = \underline{\mathcal{I}}(P),\ \text{and}\ P' \underline{\approx} Q'\ .$$

   According to the definition of the distributed prioritized weak transition relation we distinguish the following cases where $Q'' \in \mathcal{P}$ and $k \in \mathcal{L}oc$.

   (a) $Q \overset{n,\alpha}{\underset{L'}{\longrightarrow}} Q'' \overset{\epsilon}{\Longrightarrow} Q'$. Therefore, we have $\underline{\mathcal{I}}_{[n]}(Q) \subseteq L'$.

   (b) $Q \overset{k,\tau}{\underset{L'}{\longrightarrow}} Q'' \overset{n,\alpha}{\underset{L',M'}{\Longrightarrow}} Q'$. Here, we have $\underline{\mathcal{I}}_{[k]}(Q) \subseteq L'$.

   (c) $Q \xrightarrow{\underline{\tau}} Q'' \overset{n,\alpha}{\underset{L',M'}{\Longrightarrow}} Q'$.

Now, let us define $L =_{\mathrm{df}} \underline{\Pi}_{[m\cdot l]}(P+R)$ and $M =_{\mathrm{df}} \underline{\Pi}(P+R)$. The first case implies that

$$Q + R \xrightarrow[L]{n\cdot l,\alpha} Q'' \stackrel{\epsilon}{\Longrightarrow} Q'$$

by Rule (Sum1), the premise $\underline{\Pi}_{[n]}(Q) \subseteq L'$, and Lemma 6.3.2(1). Also, $\underline{\Pi}(Q+R) \subseteq \underline{\Pi}(P+R)$ holds by the fact $\underline{\mathcal{I}}(P) = \underline{\mathcal{I}}(Q)$ and Lemma 6.3.3(4). Hence, $Q + R \overset{n\cdot l,\alpha}{\underset{L,M}{\Longrightarrow}} Q'$ and $P' \underline{\approx} Q'$. We conclude this case by choosing $W \equiv Q'$ and $q \equiv n \cdot l$. The second case is similar to the first one, whereas the last case implies by Rule (Sum1) that

$$Q + R \xrightarrow{\tau} Q'' \overset{n,\alpha}{\underset{L',M'}{\Longrightarrow}} Q' \ .$$

Because of $L' \subseteq L$ and $M' \subseteq M$, we obtain $Q + R \overset{n,\alpha}{\underset{L,M}{\Longrightarrow}} Q'$ and $P' \underline{\approx} Q'$. By choosing $W \equiv Q'$ and $q \equiv n$ we are done.

2. Let $R \xrightarrow{o,\alpha} R'$ and $\underline{\tau} \notin \underline{\mathcal{I}}(P)$, i.e. $V \equiv R'$ and $p \equiv o \cdot r$. Since $P \underline{\approx}^{\mathrm{l}} Q$ and, therefore, $\underline{\mathcal{I}}(P) = \underline{\mathcal{I}}(Q)$ by Definition 6.5.10, we have

   (a) $Q + R \xrightarrow{o\cdot r,\alpha} R'$ (because $\underline{\tau} \notin \underline{\mathcal{I}}(P) = \underline{\mathcal{I}}(Q)$),

   (b) $\underline{\Pi}_{[o\cdot r]}(Q+R) \subseteq \underline{\Pi}_{[o\cdot r]}(P+R)$ (by Lemma 6.3.2(2)), and

   (c) $\underline{\Pi}(Q+R) \subseteq \underline{\Pi}(P+R)$ (by Lemma 6.3.2(3)).

   Because of the symmetry of $\underline{\approx}$ and the definition of the distributed prioritized weak transition relation $R' \underline{\approx} R'$ and $Q + R \overset{o\cdot r,\alpha}{\underset{L,M}{\Longrightarrow}} R'$, where $L = \underline{\Pi}_{[o\cdot r]}(P+R)$ and $M = \underline{\Pi}(P+R)$. By choosing $W \equiv R'$ and $q \equiv o \cdot r$, the proof is finished.

□

In order to show that $\underline{\approx}^{\mathrm{l}}$ is compositional with respect to recursion, one needs to adapt a notion of *distributed prioritized observational congruence up to* $\underline{\approx}^{\mathrm{l}}$ whose definition is as expected (cf. [149]) and, thus, omitted.

The remainder of this section is concerned with the more challenging proof of the "largest" part of Theorem 6.5.11. The proof follows the same technique as the one employed for establishing Theorem 2.4.11. It is an instance of Proposition 2.4.16, a basic result from universal algebra. For the purposes of this chapter we choose $\mathcal{R}_1 = \approx_{\times}$ and $\mathcal{R}_2 = \underline{\approx}$. First, we establish $\mathcal{R}_2^+ = \underline{\approx}^{\mathrm{l}}$.

Proposition 6.5.13 *Distributed prioritized observational congruence* $\underline{\approx}^{\mathrm{l}}$ *is the* largest *congruence contained in distributed prioritized weak bisimulation* $\underline{\approx}$*, i.e.* $\underline{\approx}^{+} = \underline{\approx}^{\mathrm{l}}$.

Proof: Since the relation $\underline{\approx}^{\mathrm{l}}$ is a congruence, which is obviously contained in $\underline{\approx}$, it is sufficient to show that for all $\mathsf{CCS}^{\mathsf{prio}}$ contexts $C[X]$ and processes $P, Q \in \mathcal{P}$ satisfying $C[P] \underline{\approx} C[Q]$ the relationship $P \underline{\approx}^{\mathrm{l}} Q$ holds. Moreover, we may restrict ourselves to a subset of $\mathsf{CCS}^{\mathsf{prio}}$ contexts according to Proposition 2.4.3. For this proof, we choose the family of contexts $C_{PQ}[X] \stackrel{\mathrm{def}}{=} (\underline{c}.\mathbf{0} + d.\mathbf{0}) + X$ where $\underline{c} \notin \underline{\mathcal{S}(P) \cup \mathcal{S}(Q)}$ and $d \notin \mathcal{S}(P) \cup \mathcal{S}(Q)$. Note that such actions $\underline{c}$ and d exist by Lemma 6.4.6. Now, let $P, Q \in \mathcal{P}$ such that $C_{PQ}[P] \underline{\approx} C_{PQ}[Q]$.

- Assume $\underline{\tau} \notin \underline{\mathcal{I}}(P)$ and, therefore, $C_{PQ}[P] \xrightarrow{rl,d} \mathbf{0}$. Since $C_{PQ}[P] \underline{\approx} C_{PQ}[Q]$ and $d \notin \mathcal{S}(Q)$ we necessarily have $C_{PQ}[Q] \stackrel{rl,d}{\underset{L,M}{\Longrightarrow}} \mathbf{0}$ and $\mathbf{0} \underline{\approx} \mathbf{0}$ where $L = M = \underline{\mathcal{IL}}(P) \cup \{\underline{c}\}$. This requires $\tau \notin \underline{\mathcal{I}}(Q)$ and $\underline{\mathcal{IL}}(Q) \subseteq \underline{\mathcal{IL}}(P)$. Hence, $\underline{\mathcal{I}}(Q) \subseteq \underline{\mathcal{I}}(P)$ holds.

- Let $P \xrightarrow{m,\alpha} P'$ for some $P' \in \mathcal{P}$ and some $m \in \mathcal{L}oc$, i.e. $C_{PQ}[P] \xrightarrow{mr,\alpha} P'$. Since $C_{PQ}[P] \underline{\approx} C_{PQ}[Q]$ there exist $Q' \in \mathcal{P}$ and $o \in \mathcal{L}oc$ satisfying $C_{PQ}[Q] \stackrel{o,\hat{\alpha}}{\underset{L',M'}{\Longrightarrow}} Q'$, $L' = \underline{\mathcal{IL}}_{[m]}(P) \cup \{\underline{c}\}$, $M' = \underline{\mathcal{IL}}(P) \cup \{\underline{c}\}$, and $P' \underline{\approx} Q'$. We know that $Q' \not\equiv C_{PQ}[Q]$ because $P' \underline{\approx} Q'$ and P' is *not* capable of performing a distributed prioritized weak $\underline{c}$-transition. Therefore, the matching step is necessary, even if $\alpha \equiv \tau$. Thus, $Q \stackrel{n,\alpha}{\underset{L,M}{\Longrightarrow}} Q'$ and $P' \underline{\approx} Q'$ for some $n \in \mathcal{L}oc$ where $n \equiv o$ or $o \equiv n{\cdot}r$, $L = \underline{\mathcal{IL}}_{[m]}(P)$, and $M = \underline{\mathcal{IL}}(P)$.

- Finally, let $P \xrightarrow{\alpha} P'$ for some $P' \in \mathcal{P}$. Then $C_{PQ}[P] \xrightarrow{\alpha} P'$. A simpler argumentation than the one above leads to the existence of some $Q' \in \mathcal{P}$ such that $Q \stackrel{\hat{\alpha}}{\Longrightarrow} Q'$ and $P' \underline{\approx} Q'$.

Since also the symmetric properties hold, all conditions of Definition 6.5.10 are satisfied, and we obtain $P \underline{\approx}^{\mathrm{l}} Q$, as desired. □

Further, we have to show that $\mathcal{R}_1^+ \subseteq \mathcal{R}_2 \subseteq \mathcal{R}_1$ i.e. $\approx_\times^+ \subseteq \underline{\approx} \subseteq \approx_\times$. The inclusion $\underline{\approx} \subseteq \approx_\times$ follows immediately from the definition of the naive and the distributed prioritized weak transition relation. In order to establish $\approx_\times^+ \subseteq \underline{\approx}$ we define the auxiliary equivalence

$$\underline{\approx}_{\mathrm{a}} =_{\mathrm{df}} \{\langle P, Q\rangle \mid C_{PQ}[P] \approx_\times C_{PQ}[Q]\}$$

which lies in between. Here, using $\underline{S} =_{\mathrm{df}} \underline{\mathcal{S}(P)} \cup \underline{\mathcal{S}(Q)}$ and $S =_{\mathrm{df}} \mathcal{S}(P) \cup \mathcal{S}(Q)$, we let $C_{PQ}[X] \stackrel{\mathrm{def}}{=} X \mid H_{PQ}$ and

$$H_{PQ} \stackrel{\mathrm{def}}{=} \underline{c}.\mathbf{0} + \underline{D}_{\underline{S}} + \sum_{\substack{L,\,M \subseteq \overline{\underline{S}},\\ b \in S}} \underline{\tau}.\Big(\begin{pmatrix} \underline{d}_{L,M,b}.H_{PQ} + \\ \underline{D}_L + e.H_{PQ} + \\ \overline{b}.(\underline{f}.H_{PQ} + \underline{D}_{\underline{S}}) \end{pmatrix} \oplus \underline{D}_M \Big) \ .$$

Note that H_{PQ} is well-defined by Lemma 6.4.6. Moreover, the processes $\underline{D}_L$ and $\underline{D}_M$ are defined as in the proof of Theorem 6.4.5, and the actions $\underline{c}, \underline{d}_{L,M,b}, e, \underline{f}$ and their complements are supposed to be "fresh" actions (cf. Lemma 6.4.6). By Proposition 2.4.3 we may conclude that $\approx_\times^+ \subseteq \underline{\approx}_{\mathrm{a}}$. The other necessary inclusion is established by the following proposition. Due to its length the proof is omitted here and can be found in Appendix C.

Proposition 6.5.14 *The inclusion* $\underline{\approx}_{\mathrm{a}} \subseteq \underline{\approx}$ *holds.*

This proposition completes the establishment of the premises of Proposition 2.4.16. Thus, $\mathcal{R}_1^+ = \mathcal{R}_2^+$, i.e. $\approx_\times^+ = \underline{\approx}^+$. Also, we have shown in Proposition 6.5.13 that $\underline{\approx}^+ = \underline{\approx}^{\mathrm{l}}$. Hence, $\approx_\times^+ = \underline{\approx}^{\mathrm{l}}$, and Theorem 6.5.11 is proved.

6.5.3 Operational and Logical Characterizations

We now characterize distributed prioritized weak bisimulation as standard bisimulation over an appropriately defined transition system. Our objectives are to be able to apply the generic algorithm presented in Section 6.4.3 for computing distributed prioritized weak bisimulation and to adapt the logic of Section 6.4.4 to this relation. To begin with, we introduce a family of relations $\underset{M}{\Longrightarrow}$ on processes, where $M \subseteq \underline{A} \setminus \{\underline{\tau}\}$, by defining $P \underset{M}{\Longrightarrow} P'$ if $\exists P''.\, P \stackrel{\epsilon}{\Longrightarrow} P'' \stackrel{\epsilon}{\Longrightarrow} P'$ and $\underline{\mathcal{U}}(P'') \subseteq M$. Moreover, we make the distributed prioritized weak transition relation independent of locations by writing $P \underset{L,M}{\overset{\hat{\alpha}}{\Longrightarrow}} P'$ whenever there exists some $m \in \mathcal{L}oc$ such that $P \underset{L,M}{\overset{m,\hat{\alpha}}{\Longrightarrow}} P'$.

Definition 6.5.15 (Alternative Distributed Prioritized Weak Bisimulation) *A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is an* alternative distributed prioritized weak bisimulation *if for every $\langle P, Q\rangle \in \mathcal{R}$, $\alpha \in A$, $\underline{\alpha} \in \underline{A}$, and $L, M \subseteq \underline{A} \setminus \{\underline{\tau}\}$ the following conditions hold.*

1. *$P \underset{M}{\Longrightarrow} P'$ implies $\exists Q'.\, Q \underset{M}{\Longrightarrow} Q'$ and $\langle P', Q'\rangle \in \mathcal{R}$.*
2. *$P \stackrel{\hat{\alpha}}{\Longrightarrow} P'$ implies $\exists Q'.\, Q \stackrel{\hat{\alpha}}{\Longrightarrow} Q'$ and $\langle P', Q'\rangle \in \mathcal{R}$.*
3. *$P \underset{L,M}{\overset{\hat{\alpha}}{\Longrightarrow}} P'$ implies $\exists Q'.\, Q \underset{L,M}{\overset{\hat{\alpha}}{\Longrightarrow}} Q'$ and $\langle P', Q'\rangle \in \mathcal{R}$.*

We write $P \underline{\approx}_* Q$ if there exists an alternative distributed prioritized weak bisimulation $\mathcal{R}$ such that $\langle P, Q\rangle \in \mathcal{R}$.

As desired, we obtain the following theorem.

Theorem 6.5.16 (Operational Characterization) *The coincidence $\underline{\approx} = \underline{\approx}_*$ holds.*

In order to prove the above theorem we need the following two properties of distributed prioritized weak bisimulation.

Lemma 6.5.17 *Let $P, P', Q \in \mathcal{P}$ such that $P \underline{\approx} Q$ and $P \stackrel{\epsilon}{\Longrightarrow} P'$. Then there exists some $Q' \in \mathcal{P}$ satisfying $Q \stackrel{\epsilon}{\Longrightarrow} Q'$ and $P' \underline{\approx} Q'$.*

The next lemma is a pendant to the previous one dealing with unprioritized ϵ-transitions.

Lemma 6.5.18 *Let $P, P', Q \in \mathcal{P}$ and $L \subseteq \underline{A} \setminus \{\underline{\tau}\}$ such that $P \underline{\approx} Q$ and $P \underset{L}{\overset{\epsilon}{\Longrightarrow}} P'$. Then there exists some $Q' \in \mathcal{P}$ satisfying $Q \underset{L}{\overset{\epsilon}{\Longrightarrow}} Q'$ and $P' \underline{\approx} Q'$.*

Both lemmata can easily be established by induction on the length of the transition from process P to P'. Now, we are able to prove Theorem 6.5.16.

Proof: We first prove the inclusion $\underline{\approx} \subseteq \underline{\approx}_*$ by showing that $\underline{\approx}$ is an alternative distributed prioritized weak bisimulation. Let $P, Q \in \mathcal{P}$ be such that $P \underline{\approx} Q$.

1. Let $P \stackrel{\Rightarrow}{\scriptstyle M} P'$ for some $P' \in \mathcal{P}$ and $M \subseteq \underline{A} \setminus \{\underline{\tau}\}$, i.e. $P \stackrel{\underline{\epsilon}}{\Longrightarrow} P'' \stackrel{\underline{\epsilon}}{\Longrightarrow} P'$ and $\underline{\Pi}(P'') \subseteq M$ for some $P'' \in \mathcal{P}$. Because of $P \underline{\approx} Q$ we know by Lemma 6.5.17 of the existence of some $Q'' \in \mathcal{P}$ such that $Q \stackrel{\underline{\epsilon}}{\Longrightarrow} Q''$ and $P'' \underline{\approx} Q''$. By Condition (1) of Definition 6.5.4 we obtain processes $\overline{Q}', \overline{Q}'' \in \mathcal{P}$ such that $Q'' \stackrel{\underline{\epsilon}}{\Longrightarrow} \overline{Q}' \stackrel{\underline{\epsilon}}{\Longrightarrow} \overline{Q}''$, $\underline{\Pi}(\overline{Q}') \subseteq \underline{\Pi}(P'')$, and $P'' \underline{\approx} \overline{Q}''$. Because of the latter, there also exists some $Q' \in \mathcal{P}$ such that $\overline{Q}'' \stackrel{\underline{\epsilon}}{\Longrightarrow} Q'$ and $P' \underline{\approx} Q'$ according to Lemma 6.5.17. Summarizing, we have $Q \stackrel{\underline{\epsilon}}{\Longrightarrow} Q'' \stackrel{\underline{\epsilon}}{\Longrightarrow} \overline{Q}' \stackrel{\underline{\epsilon}}{\Longrightarrow} \overline{Q}'' \stackrel{\underline{\epsilon}}{\Longrightarrow} Q'$, $\underline{\Pi}(\overline{Q}') \subseteq \underline{\Pi}(P'') \subseteq M$, and $P' \underline{\approx} Q'$. Thus, we have established the existence of some $Q' \in \mathcal{P}$ satisfying $Q \stackrel{\Rightarrow}{\scriptstyle M} Q'$ and $P' \underline{\approx} Q'$, as desired.

2. Let $P \stackrel{\hat{\underline{\alpha}}}{\Longrightarrow} P'$ for some $P' \in \mathcal{P}$ and some $\underline{\alpha} \in \underline{A}$, i.e. there exist some $P'', P''' \in \mathcal{P}$ such that $P \stackrel{\underline{\epsilon}}{\Longrightarrow} P'' \stackrel{\hat{\underline{\alpha}}}{\Rightarrow} P''' \stackrel{\underline{\epsilon}}{\Longrightarrow} P'$ by the definition of the distributed prioritized weak transition relation. In the following, we assume $\alpha \not\equiv \tau$ since the other case follows similar lines but is easier. Because of $P \underline{\approx} Q$ and Lemma 6.5.17 we conclude the existence of some $Q'' \in \mathcal{P}$ satisfying $Q \stackrel{\underline{\epsilon}}{\Longrightarrow} Q''$ and $P'' \underline{\approx} Q''$. The latter implies $Q'' \stackrel{\hat{\underline{\alpha}}}{\Longrightarrow} Q'''$ and $P''' \underline{\approx} Q'''$ for some $Q''' \in \mathcal{P}$. Finally, $P''' \underline{\approx} Q'''$ and Lemma 6.5.17 implies the existence of some $Q' \in \mathcal{P}$ such that $Q''' \stackrel{\underline{\epsilon}}{\Longrightarrow} Q'$ and $P' \underline{\approx} Q'$. Summarizing, we conclude $Q \stackrel{\hat{\underline{\alpha}}}{\Longrightarrow} Q'$ and $P' \underline{\approx} Q'$ for some $Q' \in \mathcal{P}$.

3. Let $P \stackrel{\hat{\alpha}}{\underset{L,M}{\Longrightarrow}} P'$ for some $P' \in \mathcal{P}$, $\alpha \in A$, and $L, M \subseteq \underline{A} \setminus \{\underline{\tau}\}$. Hence by definition, $P \stackrel{m,\hat{\alpha}}{\underset{L,M}{\Longrightarrow}} P'$ for some $m \in \mathcal{L}oc$. We consider here the more challenging case, where $\alpha \not\equiv \tau$, i.e. $P \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} P'' \stackrel{m,\hat{\alpha}}{\longrightarrow} P''' \stackrel{\epsilon}{\Longrightarrow} P'$, $\underline{\Pi}_{[m]}(P'') \subseteq L$, and $\underline{\Pi}(P'') \subseteq M$ for some $P'', P''' \in \mathcal{P}$ according to the definition of the distributed prioritized weak transition relation. A similar reasoning as in the previous case, using Lemma 6.5.18 instead of Lemma 6.5.17, leads to the existence of some $Q', Q'', Q''' \in \mathcal{P}$ and $n \in \mathcal{L}oc$ such that $Q \stackrel{\epsilon}{\underset{L}{\Longrightarrow}} Q'' \stackrel{n,\hat{\alpha}}{\underset{L',M'}{\Longrightarrow}} Q''' \stackrel{\epsilon}{\Longrightarrow} Q'$, where $L' = \underline{\Pi}_{[m]}(P'') \subseteq L$, $M' = \underline{\Pi}(P'') \subseteq M$, $P'' \underline{\approx} Q''$, $P''' \underline{\approx} Q'''$, and $P' \underline{\approx} Q'$. Thus, we obtain $Q \stackrel{n,\hat{\alpha}}{\underset{L,M}{\Longrightarrow}} Q'$, i.e. $Q \stackrel{\hat{\alpha}}{\underset{L,M}{\Longrightarrow}} Q'$, and $P' \underline{\approx} Q'$ by the definition of the distributed prioritized weak transition relation, as desired.

Hence, $\underline{\approx}$ is an alternative distributed prioritized weak bisimulation, i.e. $\underline{\approx} \subseteq \underline{\approx}_*$ by Definition 6.5.15. For proving the reverse inclusion $\underline{\approx}_* \subseteq \underline{\approx}$, let $P, Q \in \mathcal{P}$ be such that $P \underline{\approx}_* Q$. In the following we show that $\underline{\approx}_*$ is a distributed prioritized weak bisimulation.

1. Because of $P \stackrel{\Rightarrow}{\scriptstyle M} P$ for $M = \underline{\Pi}(P)$ and the premise $P \underline{\approx}_* Q$ we may conclude the existence of some $Q' \in \mathcal{P}$ such that $Q \stackrel{\Rightarrow}{\scriptstyle M} Q'$ and $P \underline{\approx}_* Q'$. Thus, Condition (1) of Definition 6.5.4 holds by the definition of $\stackrel{\Rightarrow}{\scriptstyle M}$.

2. Let $P \stackrel{\underline{\alpha}}{\Rightarrow} P'$ for some $P' \in \mathcal{P}$ and $\underline{\alpha} \in \underline{A}$. By the definition of the distributed prioritized weak transition relation $P \stackrel{\hat{\underline{\alpha}}}{\Longrightarrow} P'$ also holds. Because of the premise $P \underline{\approx}_* Q$ we know of the existence of some $Q' \in \mathcal{P}$ satisfying $Q \stackrel{\hat{\underline{\alpha}}}{\Longrightarrow} Q'$ and $P' \underline{\approx}_* Q'$. Hence, Condition (2) of Definition 6.5.4 is established.

3. Let $P \xrightarrow{m,\alpha} P'$ for some $P' \in \mathcal{P}$, $\alpha \in A$, and $m \in \mathcal{L}oc$. This implies $P \overset{\hat{\alpha}}{\underset{L,M}{\Longrightarrow}} P'$, where $L = \underline{\mathcal{I}}_{[m]}(P)$ and $M = \underline{\mathcal{I}}(P)$, according to the definitions in this section. By the premise $P \underline{\approx}_* Q$ there exists some $Q' \in \mathcal{P}$ such that $Q \overset{\hat{\alpha}}{\underset{L,M}{\Longrightarrow}} Q'$ and $P' \underline{\approx}_* Q'$. Therefore, $Q \overset{n,\hat{\alpha}}{\underset{L,M}{\Longrightarrow}} Q'$ for some $n \in \mathcal{L}oc$ and $P' \underline{\approx}_* Q'$, i.e. Condition (3) of Definition 6.5.4 is satisfied, as desired.

Finally, we conclude by Definition 6.5.4 that $\underline{\approx}_*$ is a distributed prioritized weak bisimulation, i.e. $\underline{\approx}_* \subseteq \underline{\approx}$ holds. □

We finish off this section with a remark on the logical characterization of $\underline{\approx}$. Defining a suitable logic can be done by replacing the $\langle \alpha, L \rangle$ operators of the logic presented in Section 6.4.4 by new operators $\langle\langle \alpha, L, M \rangle\rangle$ for $M \subseteq \underline{A} \setminus \{\underline{\tau}\}$ where a process $P \in \mathcal{P}$ satisfies the formula $\langle\langle \alpha, L, M \rangle\rangle \Phi$ if there exists a process $P' \in \mathcal{P}$ such that $P \overset{\hat{\alpha}}{\underset{L,M}{\Longrightarrow}} P'$ and $P' \models \Phi$. The operators $\langle \underline{\alpha} \rangle$ have also to be exchanged with operators $\langle\langle \underline{\alpha} \rangle\rangle$ where $P \models \langle\langle \underline{\alpha} \rangle\rangle \Phi$ if there exists a process $P' \in \mathcal{P}$ such that $P \overset{\hat{\alpha}}{\Longrightarrow} P'$ and $P' \models \Phi$. Finally, the logic has to be extended by new unary operators $\Uparrow_M$ for $M \subseteq \underline{A} \setminus \{\underline{\tau}\}$ in order to match the first requirement of Definition 6.5.4 where $P \models \Uparrow_M \Phi$ if $\exists P'.\, P \underset{M}{\Longrightarrow} P'$ and $P' \models \Phi$. Using these definitions a characterization of $\underline{\approx}$ can be done along the lines of [125]. Note that all processes in $\mathcal{P}$ give rise to finite-branching transition systems with respect to the distributed prioritized weak transition relation since processes are guarded, and the summation operators are binary.

6.6 Example

In this section we demonstrate the utility of $\mathsf{CCS}^{\mathsf{prio}}$ for the verification of distributed systems using an example involving an architecture scheme found in many of today's computers.

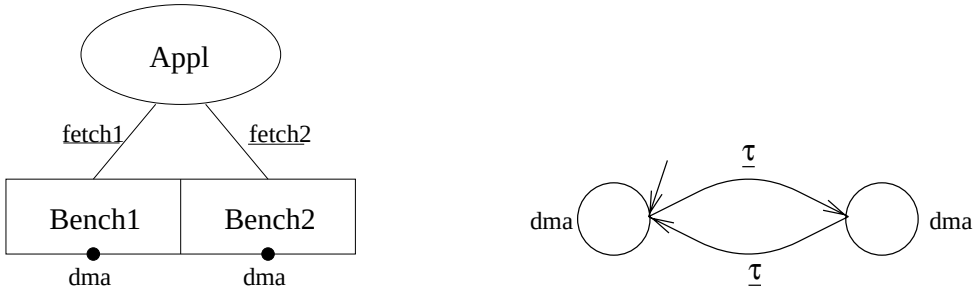

Figure 6.3: Example system and its semantics

Our example system consists of an application that manipulates data from two memory benches (cf. Figure 6.3, left-hand side). In order to improve the efficiency in the computer system each bench, `Bench1` and `Bench2`, is connected to a direct-memory-access

(DMA) controller. To overcome the low speed of most memory modules, the application Appl works alternately with each memory bench. We model Appl in CCS^{prio} by $\texttt{Appl} \stackrel{\text{def}}{=} \underline{\texttt{fetch1}}.\overline{\underline{\texttt{fetch2}}}.\texttt{Appl}$. Each memory bench Bench1 and Bench2 is continuously able to serve the application or to allow the external DMA controller to access the memory via the channel dma. However, if a memory bench has to decide between both activities, then it chooses the former since the progress of the application is considered more important. Consequently, we define $\texttt{Bench1} \stackrel{\text{def}}{=} \underline{\texttt{fetch1}}.\texttt{Bench1} + \texttt{dma}.\texttt{Bench1}$ and $\texttt{Bench2} \stackrel{\text{def}}{=} \underline{\texttt{fetch2}}.\texttt{Bench2} + \texttt{dma}.\texttt{Bench2}$. The overall system Sys is given by

$$\texttt{Sys} \stackrel{\text{def}}{=} (\texttt{Appl} \mid \texttt{Bench1} \mid \texttt{Bench2}) \setminus \{\underline{\texttt{fetch1}}, \underline{\texttt{fetch2}}\} \ .$$

Since the application uses the memory cells alternately, the DMA is expected to be allowed to access the free memory bench. Therefore, the specification simply is $\texttt{Spec} \stackrel{\text{def}}{=} \texttt{dma}.\texttt{Spec}$. The CCS^{prio} semantics of Sys is given in Figure 6.3, right hand side, where we abstract away the locations. It is easy to see that the symmetric closure of

$$\{\langle \texttt{Spec}, \texttt{Sys}\rangle, \langle \texttt{Spec}, (\overline{\underline{\texttt{fetch2}}}.\texttt{Appl} \mid \texttt{Bench1} \mid \texttt{Bench2}) \setminus \{\underline{\texttt{fetch1}}, \underline{\texttt{fetch2}}\}\rangle\}$$

is a distributed prioritized weak bisimulation. Note that Condition (1) of Definition 6.5.4 is trivially satisfied since Spec and Sys do not contain any prioritized visible actions. Therefore, we obtain $\texttt{Spec} \underline{\approx} \texttt{Sys}$ as expected. However, in the traditional approach to priority involving global pre-emption presented in Chapter 2, the dma-loops in the labeled transition system of Sys would be missing, and Sys would not be observationally equivalent to Spec.

6.7 Extensions of CCS^{prio}

Up to now we have restricted the number of priority levels in CCS^{prio} to two and communication to complementary actions having the same priority. In this section we study the implications for our theory of the removal of these restrictions which lead to a new version of CCS^{prio}, called CCS$^{prio}_{pp}$. We also compare CCS$^{prio}_{pp}$ in detail with the calculus of Camilleri and Winskel [43, 102], here referred to as CCS^{cw}. The main result of this section shows that CCS^{cw} can be embedded as a sub-calculus of CCS$^{prio}_{pp}$.

6.7.1 General Remarks

Allowing communication between unprioritized actions and complementary prioritized actions raises the question of whether the resulting internal action should be τ or $\underline{\tau}$. When dealing with local pre-emption this decision has no important consequences for sequential communicating processes, i.e. those in standard concurrent form [125]; however, it is of obvious importance for processes like $(\underline{a}.\mathbf{0} \mid \overline{a}.\mathbf{0}) + b.\mathbf{0}$ in which one has to decide if the b-transition is enabled. One reasonable view is that a communication should be pre-empted

Table 6.8: Modified operational rules

Com1	$\dfrac{P \xrightarrow{m,\alpha} P'}{P\,|\,Q \xrightarrow{m\cdot L,\alpha} P'\,|\,Q}\ \underline{\mathit{\Pi}}_{[m]}(P) \cap (\overline{\underline{\mathit{\Pi}}(Q)} \cup \overline{\mathit{\Pi}(Q)}) = \emptyset$
Com3a	$\dfrac{P \xrightarrow{m,a} P' \quad Q \xrightarrow{n,\overline{a}} Q'}{P\,|\,Q \xrightarrow{\langle m\cdot L, n\cdot R\rangle,\tau} P'\,|\,Q'}\ \begin{array}{l}\underline{\mathit{\Pi}}_{[m]}(P) \cap (\overline{\underline{\mathit{\Pi}}(Q)} \cup \overline{\mathit{\Pi}(Q)}) = \emptyset \wedge \\ \underline{\mathit{\Pi}}_{[n]}(Q) \cap (\overline{\underline{\mathit{\Pi}}(P)} \cup \overline{\mathit{\Pi}(P)}) = \emptyset\end{array}$
Com3b	$\dfrac{P \xrightarrow{m,\underline{a}} P' \quad Q \xrightarrow{n,\overline{a}} Q'}{P\,|\,Q \xrightarrow{\langle m\cdot L, n\cdot R\rangle,\tau} P'\,|\,Q'}\ \underline{\mathit{\Pi}}_{[n]}(Q) \cap (\overline{\underline{\mathit{\Pi}}(P)} \cup \overline{\mathit{\Pi}(P)}) = \emptyset$

whenever one communication partner is pre-empted, i.e. cannot engage in a communication. This implies that the *minimal* priority of the complementary actions ought to be assigned to the internal action. To reflect this in our operational semantics, we could replace Rules (Com1), (Com2), and (Com3) for parallel composition by the ones presented in Table 6.8 plus their symmetric versions. The side conditions involve sets $\mathit{\Pi}(P)$ that include all unprioritized visible actions that P can initially engage in.

It turns out that the largest congruence results concerning our behavioral relations can be carried over to the new calculus; however, the new semantics has algebraic shortcomings, since parallel composition is *not associative*, as illustrated by the following example. Consider the process $(b.\mathbf{0} + \underline{a}.\mathbf{0})\,|\,(\overline{a}.\mathbf{0} + \underline{c}.\mathbf{0})\,|\,\overline{\underline{c}}.\mathbf{0}$. When computing the semantics in a left-associative manner, the initial b-transition is pre-empted according to Rule (Com1) since $\underline{a}$ may potentially communicate with $\overline{a}$. However, when first composing the second and third parallel components, the $\overline{a}$-transition is pre-empted, and consequently the b-transition is enabled by Rule (Com1). The reason for this problem is that we pre-empt transitions because the considered process can *potentially* engage in a higher prioritized communication from a comparable location. However, this potential communication cannot take place if the communication partner is itself pre-empted. The same problem also arises when extending $\mathsf{CCS}^{\mathsf{prio}}$ to multiple priority levels, even if communication is only allowed on complementary actions of the same priority. This can be illustrated using a slight adaptation of the previous example, where priorities are given by natural numbers, with lower numbers denoting higher priority values: $(b{:}2.\mathbf{0} + a{:}1.\mathbf{0})\,|\,(\overline{a}{:}1.\mathbf{0} + c{:}0.\mathbf{0})\,|\,\overline{c}{:}0.\mathbf{0}$.

One can imagine two approaches to fixing the problems with the first (and second) alteration to our theory. One is to change the operational semantics; in particular, we could weaken the side conditions so that an unprioritized transition is only pre-empted when a prioritized action from a comparable location can *actually* engage in a communication. To do so we can adopt a conservative view of pre-emption by replacing the complementary initial action sets in the side conditions of parallel composition by ones that only include actions

having the highest priority value. Hence, only *actual* communication partners are considered. One may observe that in this setting additional transitions should be pre-empted whenever actions are restricted. For instance, in the process $P \stackrel{\text{def}}{=} (b.\mathbf{0} + \underline{a}.\mathbf{0}) \,|\, (\overline{a}.\mathbf{0} + \underline{c}.\mathbf{0})$ the b-transition is not pre-empted since $\overline{a}$ does not have the highest priority in the right parallel component. However, when plugging P in the context $C[X] \stackrel{\text{def}}{=} X \setminus \{\underline{c}, c\}$, which restricts the $\underline{c}$-transition, the communication on port a may in fact take place and, consequently, the b-transition should be pre-empted. This additional potential of pre-emption needs to be taken care of by the side conditions of the operational rules for restriction. Unfortunately, it turns out that these are hard to formalize in a compact fashion. It is also not clear to us how to establish the statements of our main theorems for the resulting semantic theory.

The second solution follows an approach developed in [43] for a different setting and involves the use of a syntax restriction on processes prohibiting output actions, i.e. actions in $\overline{\Lambda}$, from occurring as initial actions of processes that are in the scope of $+$. Hence, all potential communication partners are also actual ones, and our side conditions for parallel composition are sufficient to encode the desired notion of pre-emption. It is important to mention that the proposed syntax restriction still allows one to specify many practically relevant examples within the calculus. Indeed, a similar restriction may be found in the programming language `occam`. In the next section we follow this second solution for extending CCS^{prio}.

6.7.2 Extending CCS^{prio}

In this section we relax the mentioned design decisions of CCS^{prio} and obtain a new version of this algebra, referred to as CCS$^{prio}_{pp}$. More precisely, we allow a multi-level priority-scheme and communication between complementary actions with potentially different priorities. As seen in the previous section, both of these relaxations yield a semantics for which parallel composition is not associative. However, we have also argued that this problem vanishes if the syntax is restricted such that output actions never get pre-empted. We adapt the syntax restriction proposed by Camilleri and Winskel [43], stating that initial actions in the scope of a comparable summation operator are input actions. It is easy to see that this restriction has the desired semantic effect. Therefore, input and output actions are explicitly distinguished in CCS$^{prio}_{pp}$, where the internal action τ is also treated as input action. In the following, we let $a, b, \ldots$ range over the set Λ of input ports and $\overline{a}, \overline{b}, \ldots$ over the set $\overline{\Lambda}$ of output ports. Moreover, we let γ stand for the silent action τ or an input action and let α range over $\mathcal{A} =_{\text{df}} \Lambda \cup \overline{\Lambda} \cup \{\tau\}$. Since the priority values of output actions need never be compared with other priority values in the restricted syntax, we do not associate output actions with priority values at all. Only input actions are attached with priority values taken from a set $\mathcal{N}$, with typical elements $k, l, \ldots$, associated with a strict and total order $<$ over $\mathcal{N}$, where smaller values denote higher priorities. For the purposes of this section, one can think of $\mathcal{N}$ as the natural numbers together with the usual interpretation

of $<$. The syntax of $\mathsf{CCS}_{\mathsf{pp}}^{\mathsf{prio}}$ is formally defined by the following BNF for P.

$$\begin{array}{lcl} I & ::= & \mathbf{0} \mid x \mid \quad \gamma{:}k.I \quad \mid I+I \mid \; I \oplus I \; \mid \; I \,|\, I \; \mid I[f] \mid I \setminus L \mid \mu x.I \\ P & ::= & \mathbf{0} \mid x \mid \gamma{:}k.P \mid \overline{a}.P \mid \quad I \quad \mid P \oplus P \mid P \,|\, P \mid P[f] \mid P \setminus L \mid \mu x.P \end{array}$$

where f is an *injective*, finite relabeling, $L \subseteq \Lambda \cup \overline{\Lambda}$ is a restriction set, and x is a variable taken from a countable domain $\mathcal{V}$. A relabeling satisfies the properties $f(\Lambda) \subseteq \Lambda$, $f(\overline{\Lambda}) \subseteq \overline{\Lambda}$, $f(\tau) = \tau$, and $f(\overline{a}) = \overline{f(a)}$. Thus, additionally to the requirements of a finite relabeling in CCS, relabelings in $\mathsf{CCS}_{\mathsf{pp}}^{\mathsf{prio}}$ may only map input ports to input ports and output ports to output ports. Since actions attached with different priority values do not represent different ports, relabelings and restriction sets do not deal with priority values. Especially, the priority value of a relabeled transition remains the same, i.e. there is no implicit mechanism for prioritization or deprioritization [54]. In the remainder, we let $\mathcal{P}_{\mathsf{pp}}^{\mathsf{prio}}$ denote the set of all $\mathsf{CCS}_{\mathsf{pp}}^{\mathsf{prio}}$ processes, ranged over by $P, Q, R, \ldots$, i.e. the set of closed and guarded terms in our language.

Table 6.9: Initial output action sets for $\mathsf{CCS}_{\mathsf{pp}}^{\mathsf{prio}}$

$\overline{\mathit{I}}(\mu x.P) =_{\mathrm{df}} \overline{\mathit{I}}(P[\mu x.P/x])$	$\overline{\mathit{I}}(\overline{a}.P) =_{\mathrm{df}} \{a\}$
$\overline{\mathit{I}}(P \,\|\, Q) =_{\mathrm{df}} \overline{\mathit{I}}(P) \cup \overline{\mathit{I}}(Q)$	$\overline{\mathit{I}}(P \oplus Q) =_{\mathrm{df}} \overline{\mathit{I}}(P) \cup \overline{\mathit{I}}(Q)$
$\overline{\mathit{I}}(P[f]) =_{\mathrm{df}} \{f(a) \mid a \in \overline{\mathit{I}}(P)\}$	$\overline{\mathit{I}}(P \setminus L) =_{\mathrm{df}} \overline{\mathit{I}}(P) \setminus (L \cup \overline{L})$

The semantics of $\mathsf{CCS}_{\mathsf{pp}}^{\mathsf{prio}}$ processes are again labeled transition systems whose transition relations are specified by operational rules. As expected, the distinction of input and output actions is also reflected in the rules. Since output transitions cannot get preempted they do also not need an associated priority value, and output transitions do not need to take account of locations. We first present two auxiliary sets used when presenting the operational rules, namely (i) initial output action sets $\overline{\mathit{I}}(P)$ of a process P and (ii) initial input action sets $\mathrm{I}_m^k(P)$ of P with respect to a priority value $k \in \mathcal{N}$ and a location $m \in \mathcal{L}oc$, which are defined to be the smallest sets satisfying the equations presented in Tables 6.9 and 6.10, respectively. For technical convenience we remove the complement of output actions in the definition of $\overline{\mathit{I}}(\cdot)$, and we use the abbreviations $\mathrm{I}_M^{<k}(P) =_{\mathrm{df}} \bigcup\{\mathrm{I}_m^l(P) \mid m \in M,\ l < k\}$, $\mathrm{I\!I}_M^{<k}(P) =_{\mathrm{df}} \mathrm{I}_M^{<k}(P) \setminus \{\tau\}$, $\mathrm{I}(P) =_{\mathrm{df}} \bigcup\{\mathrm{I}_m^l(P) \mid m \in \mathcal{L}oc,\ l \in \mathcal{N}\}$, and $\mathrm{I\!I}(P) =_{\mathrm{df}} \mathrm{I}(P) \setminus \{\tau\}$. Observe that for the definition of the initial input action sets, the resulting internal action of a communication is only attached with the priority value and the location of the input transition, and not with a pair of priority values and locations as is the case for $\mathsf{CCS}^{\mathsf{prio}}$.

Table 6.10: Initial input action sets for $\mathsf{CCS}^{\text{prio}}_{\text{pp}}$

$\mathrm{I}^k_m(\mu x.P) =_{\text{df}} \mathrm{I}^k_m(P[\mu x.P/x])$	$\mathrm{I}^k_\bullet(\gamma{:}l.P) =_{\text{df}} \{\gamma \mid k = l\}$
$\mathrm{I}^k_{m\cdot l}(P + Q) =_{\text{df}} \mathrm{I}^k_m(P)$	$\mathrm{I}^k_{m\cdot L}(P \oplus Q) =_{\text{df}} \mathrm{I}^k_m(P)$
$\mathrm{I}^k_{n\cdot r}(P + Q) =_{\text{df}} \mathrm{I}^k_n(Q)$	$\mathrm{I}^k_{n\cdot R}(P \oplus Q) =_{\text{df}} \mathrm{I}^k_n(Q)$
$\mathrm{I}^k_m(P[f]) =_{\text{df}} \{f(\gamma) \mid \gamma \in \mathrm{I}^k_m(P)\}$	$\mathrm{I}^k_{m\cdot L}(P \mid Q) =_{\text{df}} \mathrm{I}^k_m(P) \cup \{\tau \mid \mathrm{I}^k_m(P) \cap \overline{\mathit{II}}(Q) \neq \emptyset\}$
$\mathrm{I}^k_m(P \setminus L) =_{\text{df}} \mathrm{I}^k_m(P) \setminus (L \cup \overline{L})$	$\mathrm{I}^k_{n\cdot R}(P \mid Q) =_{\text{df}} \mathrm{I}^k_n(Q) \cup \{\tau \mid \mathrm{I}^k_n(Q) \cap \overline{\mathit{II}}(P) \neq \emptyset\}$

Table 6.11: Operational semantics for $\mathsf{CCS}^{\text{prio}}_{\text{pp}}$ wrt. output transitions

$$\overline{\mathsf{Act}}\ \frac{-}{\overline{a}.P \xrightarrow{\overline{a}} P} \qquad \overline{\mathsf{iSum1}}\ \frac{P \xrightarrow{\overline{a}} P'}{P \oplus Q \xrightarrow{\overline{a}} P'} \qquad \overline{\mathsf{iSum2}}\ \frac{Q \xrightarrow{\overline{a}} Q'}{P \oplus Q \xrightarrow{\overline{a}} Q'}$$

$$\overline{\mathsf{Rec}}\ \frac{P[\mu x.P/x] \xrightarrow{\overline{a}} P'}{\mu x.P \xrightarrow{\overline{a}} P'} \qquad \overline{\mathsf{Rel}}\ \frac{P \xrightarrow{\overline{a}} P'}{P[f] \xrightarrow{f(\overline{a})} P'[f]} \qquad \overline{\mathsf{Res}}\ \frac{P \xrightarrow{\overline{a}} P'}{P \setminus L \xrightarrow{\overline{a}} P' \setminus L}\ \overline{a} \notin L \cup \overline{L}$$

$$\overline{\mathsf{Com1}}\ \frac{P \xrightarrow{\overline{a}} P'}{P \mid Q \xrightarrow{\overline{a}} P' \mid Q} \qquad \overline{\mathsf{Com2}}\ \frac{Q \xrightarrow{\overline{a}} Q'}{P \mid Q \xrightarrow{\overline{a}} P \mid Q'}$$

The operational rules for the $\mathsf{CCS}^{\text{prio}}_{\text{pp}}$ semantics are formally stated in Table 6.11 for output transitions and in Table 6.12 for input transitions. As expected, the rules for output transitions coincide with the ones for plain CCS [125] whereas the rules for input transitions take local pre-emption into account, thereby using location and priority value information in their side conditions. It is worth having a closer look at their side conditions of Rules (Sum1) and (Sum2) which differ in principle from the corresponding rules of $\mathsf{CCS}^{\text{prio}}$. They guarantee that an initial $\gamma{:}l$-transition of a process P is also pre-empted whenever there exists a higher prioritized initial $\gamma{:}k$-transition of P, i.e. if $k < l$. This additional kind of pre-emption reflects that output transitions can communicate with a complementary input transition regardless of its priority value, i.e. if more than one communication partner offering the matching input transition is available from comparable

Table 6.12: Operational semantics for $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ wrt. input transitions

Act	$\dfrac{-}{\gamma{:}k.P \xrightarrow{\bullet,\gamma:k} P}$	Rec	$\dfrac{P[\mu x.P/x] \xrightarrow{m,\gamma:k} P'}{\mu x.P \xrightarrow{m,\gamma:k} P'}$
Sum1	$\dfrac{P \xrightarrow{m,\gamma:k} P'}{P+Q \xrightarrow{m\cdot l,\gamma:k} P'}\ \tau,\gamma \notin \mathrm{I}^{<k}(Q)$	Com1	$\dfrac{P \xrightarrow{m,\gamma:k} P'}{P \mid Q \xrightarrow{m\cdot L,\gamma:k} P' \mid Q}\ \mathrm{I\!I}^{<k}_{[m]}(P) \cap \overline{\mathcal{I}}(Q) = \emptyset$
Sum2	$\dfrac{Q \xrightarrow{n,\gamma:k} Q'}{P+Q \xrightarrow{n\cdot r,\gamma:k} Q'}\ \tau,\gamma \notin \mathrm{I}^{<k}(P)$	Com2	$\dfrac{Q \xrightarrow{n,\gamma:k} Q'}{P \mid Q \xrightarrow{n\cdot R,\gamma:k} P \mid Q'}\ \mathrm{I\!I}^{<k}_{[n]}(Q) \cap \overline{\mathcal{I}}(P) = \emptyset$
iSum1	$\dfrac{P \xrightarrow{m,\gamma:k} P'}{P \oplus Q \xrightarrow{m\cdot L,\gamma:k} P'}$	Com3	$\dfrac{P \xrightarrow{m,a:k} P' \quad Q \xrightarrow{\overline{a}} Q'}{P \mid Q \xrightarrow{m\cdot L,\tau:k} P' \mid Q'}\ \mathrm{I\!I}^{<k}_{[m]}(P) \cap \overline{\mathcal{I}}(Q) = \emptyset$
iSum2	$\dfrac{Q \xrightarrow{n,\gamma:k} Q'}{P \oplus Q \xrightarrow{n\cdot R,\gamma:k} Q'}$	Com4	$\dfrac{P \xrightarrow{\overline{a}} P' \quad Q \xrightarrow{n,a:k} Q'}{P \mid Q \xrightarrow{n\cdot R,\tau:k} P' \mid Q'}\ \mathrm{I\!I}^{<k}_{[n]}(Q) \cap \overline{\mathcal{I}}(P) = \emptyset$
Rel	$\dfrac{P \xrightarrow{m,\gamma:k} P'}{P[f] \xrightarrow{m,f(\gamma):k} P'[f]}$	Res	$\dfrac{P \xrightarrow{m,\gamma:k} P'}{P \setminus L \xrightarrow{m,\gamma:k} P' \setminus L}\ \gamma \notin L \cup \overline{L}$

locations, the one attached with the highest priority is taken. This kind of pre-emption requires us to restrict relabelings to injective ones as has been pointed out in [43].

The behavioral relations defined for $\mathsf{CCS}^{\mathsf{prio}}$ can be adapted to $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ in a straightforward fashion, as we demonstrate by the notion of distributed prioritized strong bisimulation.

Definition 6.7.1 (Distributed Prioritized Strong Bisimulation for $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$)
A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a distributed prioritized strong bisimulation for $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ *if for every $\langle P, Q\rangle \in \mathcal{R}$, $\overline{a} \in \overline{\Lambda}$, $\gamma \in \Lambda \cup \{\tau\}$, $k \in \mathcal{N}$, and $m \in \mathcal{L}oc$, the following conditions hold.*

1. *$P \xrightarrow{\overline{a}} P'$ implies $\exists Q'.\, Q \xrightarrow{\overline{a}} Q'$ and $\langle P', Q'\rangle \in \mathcal{R}$, and*
2. *$P \xrightarrow{m,\gamma:k} P'$ implies $\exists Q', l, n.\, Q \xrightarrow{n,\gamma:l} Q'$, $\mathrm{I\!I}^{<l}_{[n]}(Q) \subseteq \mathrm{I\!I}^{<k}_{[m]}(P)$, and $\langle P', Q'\rangle \in \mathcal{R}$.*

We write $P \simeq_{pp} Q$ if $\langle P, Q\rangle \in \mathcal{R}$ for a distributed prioritized strong bisimulation $\mathcal{R}$ for CCS^{prio}_{pp}.

Regarding distributed prioritized strong bisimulation for $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ we obtain the following compositionality result.

Proposition 6.7.2 *The relation $\simeq_{pp}$ is compositional with respect to all CCS^{prio}_{pp} operators except summation.*

The proof can easily be done by using standard techniques and, therefore, is omitted here. The reason for the compositionality lack with respect to summation is illustrated by the following example: $a{:}0.\mathbf{0} \simeq_{\mathsf{pp}} a{:}1.\mathbf{0}$ holds, but $a{:}0.\mathbf{0} + \tau{:}0.\mathbf{0} \not\simeq_{\mathsf{pp}} a{:}1.\mathbf{0} + \tau{:}0.\mathbf{0}$ since the former process can engage in a transition labeled by a whereas the latter cannot. The next definition repairs this defect of $\simeq_{\mathsf{pp}}$.

Definition 6.7.3 (Distributed Prioritized Strong Congruence for $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$) *Two processes $P, Q \in \mathcal{P}^{prio}_{pp}$ are* prioritized strong congruent for $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$, *in signs $P \simeq^{+}_{pp} Q$, if for every $\overline{a} \in \overline{\Lambda}$, $\gamma \in \Lambda \cup \{\tau\}$, $k \in \mathcal{N}$, and $m \in \mathcal{L}oc$ the following conditions and their symmetric counterparts hold.*

1. *$P \stackrel{\overline{a}}{\rightarrow} P'$ implies $\exists Q'.\, Q \stackrel{\overline{a}}{\rightarrow} Q'$ and $P' \simeq_{pp} Q'$, and*
2. *$P \stackrel{m,\gamma:k}{\longrightarrow} P'$ implies $\exists Q', l, n.\, Q \stackrel{n,\gamma:l}{\longrightarrow} Q'$, $\mathbb{I}^{<l}_{[n]}(Q) \subseteq \mathbb{I}^{<k}_{[m]}(P)$, $l \leq k$, and $P' \simeq_{pp} Q'$.*

In contrast to the definition of distributed prioritized strong bisimulation for $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$, Definition 6.7.3 requires an initial γ-transition of P with priority k to be matched by an initial γ-transition of Q with some higher priority $l < k$, and vice versa.

Theorem 6.7.4 *The relation $\simeq^{+}_{pp}$ is the largest congruence contained in $\simeq_{pp}$.*

This theorem can be proved in the usual fashion. It is worth noting that $\simeq^{+}_{\mathsf{pp}}$ is also the largest congruence contained in the naive adaptation of standard strong bisimulation [125]. The proof of the "largest" part can be done by applying the same proof technique as used for $\mathsf{CCS}^{\mathsf{prio}}$ to show that distributed prioritized observational congruence is the largest congruence contained in naive distributed prioritized weak bisimulation.

6.7.3 The Camilleri–Winskel Calculus $\mathsf{CCS}^{\mathsf{cw}}$

In this section we briefly review the formal framework of Camilleri and Winskel's approach to priority [43], here referred to as $\mathsf{CCS}^{\mathsf{cw}}$. This process algebra with priority is inspired by `occam`'s priority mechanism (cf. [13]). It is also based on Milner's CCS, but in contrast to the approaches considered so far it does not assign priority values to actions. Instead, there exists a special summation operator $+\rangle$ in $\mathsf{CCS}^{\mathsf{cw}}$, called *prioritized choice*, which favors its left over its right argument. The syntax of $\mathsf{CCS}^{\mathsf{cw}}$ terms is given by the following BNF for P.

$$\begin{array}{lcl} I & ::= & \mathbf{0} \mid x \mid \gamma.I \mid I +\rangle I \mid I + I \mid I \,|\, I \mid I[f] \mid I \setminus L \mid \mu x.I \\ P & ::= & \mathbf{0} \mid x \mid \alpha.P \mid \quad I \quad \mid P + P \mid P \,|\, P \mid P[f] \mid P \setminus L \mid \mu x.P \end{array}$$

where the action γ, the injective, finite relabeling f, and the restriction set L satisfy the same restrictions as in the previous section. Again, closed and guarded terms determine

the set $\mathcal{P}^{\mathsf{cw}}$, ranged over by $P, Q, R, \ldots$, of $\mathsf{CCS}^{\mathsf{cw}}$ processes. Further, we introduce initial output and input action sets as displayed in Tables 6.13 and 6.14, respectively, and write $\mathrm{I\!I}^{\mathsf{cw}}(P)$ for $\mathrm{I}^{\mathsf{cw}}(P) \setminus \{\tau\}$. The definition of initial input action sets slightly differs from the one presented in [43] in that we do not take care of any kind of pre-emption. In [43] some pre-emption is considered in the case of prioritized choice but not in the case of parallel composition. Since for the definition of the semantics it does not matter if some potential input actions are pre-empted, our definition suffices and, in contrast to [43], is uniform.

Table 6.13: Initial output action sets for $\mathsf{CCS}^{\mathsf{cw}}$

$\overline{\mathrm{I\!I}}^{\mathsf{cw}}(\overline{a}.P) =_{\mathrm{df}} \{a\}$	$\overline{\mathrm{I\!I}}^{\mathsf{cw}}(\mu x.P) =_{\mathrm{df}} \overline{\mathrm{I\!I}}^{\mathsf{cw}}(P[\mu x.P/x])$
$\overline{\mathrm{I\!I}}^{\mathsf{cw}}(P \mid Q) =_{\mathrm{df}} \overline{\mathrm{I\!I}}^{\mathsf{cw}}(P) \cup \overline{\mathrm{I\!I}}^{\mathsf{cw}}(Q)$	$\overline{\mathrm{I\!I}}^{\mathsf{cw}}(P + Q) =_{\mathrm{df}} \overline{\mathrm{I\!I}}^{\mathsf{cw}}(P) \cup \overline{\mathrm{I\!I}}^{\mathsf{cw}}(Q)$
$\overline{\mathrm{I\!I}}^{\mathsf{cw}}(P[f]) =_{\mathrm{df}} \{f(a) \mid a \in \overline{\mathrm{I\!I}}^{\mathsf{cw}}(P)\}$	$\overline{\mathrm{I\!I}}^{\mathsf{cw}}(P \setminus L) =_{\mathrm{df}} \overline{\mathrm{I\!I}}^{\mathsf{cw}}(P) \setminus (L \cup \overline{L})$

Table 6.14: Initial input action sets for $\mathsf{CCS}^{\mathsf{cw}}$

$\mathrm{I}^{\mathsf{cw}}(\gamma.P) =_{\mathrm{df}} \{\gamma\}$	$\mathrm{I}^{\mathsf{cw}}(\mu x.P) =_{\mathrm{df}} \mathrm{I}^{\mathsf{cw}}(P[\mu x.P/x])$
$\mathrm{I}^{\mathsf{cw}}(P +\!\rangle\, Q) =_{\mathrm{df}} \mathrm{I}^{\mathsf{cw}}(P) \cup \mathrm{I}^{\mathsf{cw}}(Q)$	$\mathrm{I}^{\mathsf{cw}}(P + Q) =_{\mathrm{df}} \mathrm{I}^{\mathsf{cw}}(P) \cup \mathrm{I}^{\mathsf{cw}}(Q)$
$\mathrm{I}^{\mathsf{cw}}(P[f]) =_{\mathrm{df}} \{f(\gamma) \mid \gamma \in \mathrm{I}^{\mathsf{cw}}(P)\}$	$\mathrm{I}^{\mathsf{cw}}(P \setminus L) =_{\mathrm{df}} \mathrm{I}^{\mathsf{cw}}(P) \setminus (L \cup \overline{L})$
$\mathrm{I}^{\mathsf{cw}}(P \mid Q) =_{\mathrm{df}} \mathrm{I}^{\mathsf{cw}}(P) \cup \mathrm{I}^{\mathsf{cw}}(Q) \cup \{\tau \mid \mathrm{I}^{\mathsf{cw}}(P) \cap \overline{\mathrm{I\!I}}^{\mathsf{cw}}(Q) \neq \emptyset\}$	

The semantics of a $\mathsf{CCS}^{\mathsf{cw}}$ process is given by a labeled transition system whose transition relation gives rise to transitions of the form $\vdash^{\mathsf{cw}}_{M} P \xrightarrow{\alpha} P'$, where $M \subseteq \Lambda$. Intuitively, process P can engage in an α-transition to P' whenever the environment does not offer communications on ports in M. Despite notational differences, this is the same underlying principle as for some transition relations defined in the previous chapters which are also parameterized by initial action sets. Note that $\alpha \in \overline{\Lambda}$ implies $M = \emptyset$ as can be proved by induction on the depth of the derivation of a transition $\vdash^{\mathsf{cw}}_{M} P \xrightarrow{\alpha} P'$. The $\mathsf{CCS}^{\mathsf{cw}}$ transition relation is formally defined in Table 6.15, where $f(M)$ stands for $\{f(m) \mid m \in M\}$.

Recall that the initial actions of P in $P +\rangle Q$ are given preference over the initial actions of Q. Also in this approach a prioritized τ, i.e. an internal action in which the left argument of $+\rangle$ can initially engage in, has pre-emptive power over unprioritized actions, i.e. actions in which the right argument of $+\rangle$ can initially engage in. Thus, the prioritized choice operator $+\rangle$ of [43] corresponds to the summation operator $+$ in our framework. In [43] the operator $+$ stands for nondeterministic choice where priorities arising from the left and the right argument are incomparable. This operator is matched by the distributed summation operator $\oplus$ in CCS$^{prio}_{pp}$. We further investigate the correspondence of these operators in the next section.

Table 6.15: Operational semantics for CCS^{cw}

Act	$\dfrac{-}{\vdash^{cw}_{\emptyset} \alpha.P \xrightarrow{\alpha} P}$	Res	$\dfrac{\vdash^{cw}_{M} P \xrightarrow{\alpha} P'}{\vdash^{cw}_{M\setminus(L\cup\overline{L})} P \setminus L \xrightarrow{\alpha} P' \setminus L}\ \alpha \notin L \cup \overline{L}$
Sum1	$\dfrac{\vdash^{cw}_{M} P \xrightarrow{\alpha} P'}{\vdash^{cw}_{M} P +\rangle Q \xrightarrow{\alpha} P'}$	Sum2	$\dfrac{\vdash^{cw}_{N} Q \xrightarrow{\alpha} Q'}{\vdash^{cw}_{N\cup \mathrm{I}^{cw}(P)} P +\rangle Q \xrightarrow{\alpha} Q'}\ \tau, \alpha \notin \mathrm{I}^{cw}(P)$
iSum1	$\dfrac{\vdash^{cw}_{M} P \xrightarrow{\alpha} P'}{\vdash^{cw}_{M} P + Q \xrightarrow{\alpha} P'}$	Com1	$\dfrac{\vdash^{cw}_{M} P \xrightarrow{\alpha} P'}{\vdash^{cw}_{M} P \mid Q \xrightarrow{\alpha} P' \mid Q}\ M \cap \overline{\mathit{I}}^{cw}(Q) = \emptyset$
iSum2	$\dfrac{\vdash^{cw}_{N} Q \xrightarrow{\alpha} Q'}{\vdash^{cw}_{N} P + Q \xrightarrow{\alpha} Q'}$	Com2	$\dfrac{\vdash^{cw}_{N} Q \xrightarrow{\alpha} Q'}{\vdash^{cw}_{N} P \mid Q \xrightarrow{\alpha} P \mid Q'}\ N \cap \overline{\mathit{I}}^{cw}(P) = \emptyset$
Rel	$\dfrac{\vdash^{cw}_{M} P \xrightarrow{\alpha} P'}{\vdash^{cw}_{f(M)} P[f] \xrightarrow{f(\alpha)} P'[f]}$	Com3	$\dfrac{\vdash^{cw}_{M} P \xrightarrow{a} P' \quad \vdash^{cw}_{\emptyset} Q \xrightarrow{\overline{a}} Q'}{\vdash^{cw}_{M} P \mid Q \xrightarrow{\tau} P' \mid Q'}\ M \cap \overline{\mathit{I}}^{cw}(Q) = \emptyset$
Rec	$\dfrac{\vdash^{cw}_{M} P[\mu x.P/x] \xrightarrow{\alpha} P'}{\vdash^{cw}_{M} \mu x.P \xrightarrow{\alpha} P'}$	Com4	$\dfrac{\vdash^{cw}_{\emptyset} P \xrightarrow{\overline{a}} P' \quad \vdash^{cw}_{N} Q \xrightarrow{a} Q'}{\vdash^{cw}_{N} P \mid Q \xrightarrow{\tau} P' \mid Q'}\ N \cap \overline{\mathit{I}}^{cw}(P) = \emptyset$

Camilleri and Winskel have also developed a bisimulation-based semantic theory for CCS^{cw}. Their notion of strong bisimulation for CCS^{cw}, as defined below, is shown to be a congruence [43]. However, this relation is not algebraically compared with a naive version of standard strong bisimulation as we have done with respect to distributed prioritized strong bisimulation for CCS^{prio}.

Definition 6.7.5 (Distributed Prioritized Strong Bisimulation for $\mathsf{CCS}^{\mathsf{cw}}$)
A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a distributed prioritized strong bisimulation for $\mathsf{CCS}^{\mathsf{cw}}$ *if for every $\langle P, Q\rangle \in \mathcal{R}$, $\alpha \in \mathcal{A}$, and $M \subseteq \Lambda$ the following condition holds:*

$$\vdash^{cw}_{M} P \xrightarrow{\alpha} P' \;\; \textit{implies} \;\; \exists Q', N.\ \vdash^{cw}_{N} Q \xrightarrow{\alpha} Q',\ N \subseteq M,\ \textit{and}\ \langle P', Q'\rangle \in \mathcal{R}\ .$$

We write $P \simeq_{cw} Q$ if $\langle P, Q\rangle \in \mathcal{R}$ for a distributed prioritized strong bisimulation $\mathcal{R}$ for CCS^{cw}.

6.7.4 Embedding $\mathsf{CCS}^{\mathsf{cw}}$ in $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$

In this section we provide an embedding of $\mathsf{CCS}^{\mathsf{cw}}$ in $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ such that $\mathsf{CCS}^{\mathsf{cw}}$ can be considered as a sub-calculus in $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$. Formally, for translating a $\mathsf{CCS}^{\mathsf{cw}}$ term into a $\mathsf{CCS}^{\mathsf{prio}}$ term we choose a strict and total order $\langle \mathcal{N}, <\rangle$ representing priority values together with a "*more important*" relation. For notational convenience we overload the symbols $\mathcal{P}^{\mathsf{cw}}$ and $\mathcal{P}^{\mathsf{prio}}_{\mathsf{pp}}$ in this section, which sometimes are used for the sets of all $\mathsf{CCS}^{\mathsf{cw}}$ and $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ *terms*, and not only for the sets of all $\mathsf{CCS}^{\mathsf{cw}}$ and $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ processes, respectively. Recall that the only way of expressing priorities in $\mathsf{CCS}^{\mathsf{cw}}$ is by using the operator $+\rangle$ which gives initial actions of its left argument process priority over initial actions of the right argument process. Thus, a priority can be encoded as a bit string that points to these left and right argument processes of $+\rangle$. Hence, we define $\mathcal{N} =_{\mathrm{df}} \{0,1\}^*$, ranged over by $k, l, \ldots$, for the purposes of this section. If $k = k_1 k_2 \ldots k_m$ and $l = l_1 l_2 \ldots l_n \in \mathcal{N}$ for some $m, n \in \mathbb{N}$ then $k < l$ if (i) $\exists i.\ k_i \preceq l_i$, where $\prec$ is the order on 0 and 1 satisfying $1 \prec 0$, and (ii) $\forall j <_{\mathbb{N}} i.\ k_j = l_j$. In other words, $<$ is the lexicographic extension of $\prec$ on the domain $\{0,1\}$.

Table 6.16: Translation function

$\xi^k(\mathbf{0}) =_{\mathrm{df}} \mathbf{0}$	$\xi^k(P+Q) =_{\mathrm{df}} \xi^k(P) \oplus \xi^k(Q)$	$\xi^k(P \setminus L) =_{\mathrm{df}} \xi^k(P) \setminus L$
$\xi^k(x) =_{\mathrm{df}} x$	$\xi^k(P +\rangle Q) =_{\mathrm{df}} \xi^{k0}(P) + \xi^{k1}(Q)$	$\xi^k(P[f]) =_{\mathrm{df}} \xi^k(P)[f]$
$\xi^k(\gamma.P) =_{\mathrm{df}} \gamma{:}k.\xi^\epsilon(P)$	$\xi^k(P \mid Q) =_{\mathrm{df}} \xi^k(P) \mid \xi^k(Q)$	$\xi^k(\mu x.P) =_{\mathrm{df}} \mu x.\xi^k(P)$
$\xi^k(\overline{a}.P) =_{\mathrm{df}} \overline{a}.\xi^\epsilon(P)$		

We now introduce the *translation function* $\xi(\cdot) : \mathcal{P}^{\mathsf{cw}} \longrightarrow \mathcal{P}^{\mathsf{prio}}_{\mathsf{pp}}$ by $\xi(P) =_{\mathrm{df}} \xi^\epsilon(P)$, which maps $\mathsf{CCS}^{\mathsf{cw}}$ terms to $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ terms. The functions $\xi^k(P)$, for $k \in \mathcal{N}$, are inductively defined

over the structure of $\mathsf{CCS}^{\mathsf{cw}}$ processes as shown in Table 6.16. One might observe that the "computation" of a translation initially "starts" with the priority value ϵ as argument of the translation function. However, this initial value does not play an important role for $\simeq_{\mathsf{pp}}$, i.e. if one is not interested in compositionality with respect to summation.

Lemma 6.7.6 *Let $k \in \mathcal{N}$ and $P \in \mathcal{P}^{cw}$. Then $\xi^k(P) \simeq_{pp} \xi(P)$.*

Proof: The proof of this lemma can be done by induction on the structure of P. For the base case $P \equiv \mathbf{0}$ the statement is trivial, for $P \equiv Q_1 +\rangle\, Q_2$ it is easy to establish by using the induction hypothesis since the involved operators are dynamic, and the remaining cases follow immediately by applying the induction hypothesis and Proposition 6.7.2. □

It is also important to note that the translation function is not *surjective*, e.g. consider the process $(a{:}0.\mathbf{0} + b{:}2.\mathbf{0}) + c{:}1.\mathbf{0}$ on which no $\mathsf{CCS}^{\mathsf{cw}}$ process is mapped. This example also shows that the notion of compositionality in $\mathsf{CCS}^{\mathsf{cw}}$ is restricted with respect to $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$, since a comparable summation can only be extended by summands which have a higher or a lower priority than the already considered summands.

Before we present and proof our main result of this section which states that the translation function provides an embedding of $\mathsf{CCS}^{\mathsf{cw}}$ in $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$, we establish a lemma concerned with the initial action sets.

Lemma 6.7.7 *For all $P \in \mathcal{P}^{cw}$ and $k \in \mathcal{N}$ we have $\overline{\mathit{I}}^{cw}(P) = \overline{\mathit{I}}(\xi^k(P))$ and $\mathrm{I}^{cw}(P) = \mathrm{I}(\xi^k(P))$.*

Proof: The proof of $\overline{\mathit{I}}^{\mathsf{cw}}(P) = \overline{\mathit{I}}(\xi^k(P))$ can be done by a simple induction on the structure of P which involves the application of the definitions of the initial output actions sets for $\mathsf{CCS}^{\mathsf{cw}}$ and $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ as well as the definition of the translation function $\xi^k(\cdot)$. For proving $\mathrm{I}^{\mathsf{cw}}(P) = \mathrm{I}(\xi^k(P))$ it is sufficient to show that

$$\gamma \in \mathrm{I}^{\mathsf{cw}}(P) \text{ if and only if } \exists m \in \mathcal{L}oc, l \in \mathcal{N}.\ \gamma \in \mathrm{I}^{kl}_m(\xi^k(P))$$

which can also be done by induction on the structure of P by using the definitions of the initial input actions sets for $\mathsf{CCS}^{\mathsf{cw}}$ and $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$, and the definition of the translation function $\xi^k(\cdot)$. □

The following proposition makes the semantic relationship between a $\mathsf{CCS}^{\mathsf{cw}}$ term P and its embedding $\xi(P)$ precise.

Proposition 6.7.8 (Correspondence of Transitions)

1. *Let $P, P' \in \mathcal{P}^{cw}$, $\overline{a} \in \overline{\Lambda}$. Then $\vdash^{cw}_{\emptyset} P \xrightarrow{\overline{a}} P'$ implies $\exists \overline{P'} \in \mathcal{P}^{prio}_{pp}.\ \xi(P) \stackrel{\overline{a}}{\Rightarrow} \overline{P'}$ and $\overline{P'} \simeq_{pp} \xi(P')$,*
2. *Let $P \in \mathcal{P}^{cw}$, $\overline{P'} \in \mathcal{P}^{prio}_{pp}$, and $\overline{a} \in \overline{\Lambda}$ such that $\xi(P) \stackrel{\overline{a}}{\Rightarrow} \overline{P'}$. Then $\exists P' \in \mathcal{P}^{cw}.\ \vdash^{cw}_{\emptyset} P \xrightarrow{\overline{a}} P'$ and $\overline{P'} \simeq_{pp} \xi(P')$.*

3. *Let* $P, P' \in \mathcal{P}^{cw}$, $\gamma \in \Lambda \cup \{\tau\}$, *and* $M \subseteq \Lambda$. *Then* $\vdash_M^{cw} P \xrightarrow{\gamma} P'$ *implies* $\exists k \in \mathcal{N}, m \in \mathcal{L}oc, \overline{P}' \in \mathcal{P}_{pp}^{prio}$. $\xi(P) \xrightarrow{m,\gamma:k} \overline{P}'$, $\overline{P}' \simeq_{pp} \xi(P')$, *and* $M = \mathbb{I}_{[m]}^{<k}(\xi(P))$.

4. *Let* $P \in \mathcal{P}^{cw}$, $m \in \mathcal{L}oc$, $\gamma \in \Lambda \cup \{\tau\}$, $k \in \mathcal{N}$, *and* $\overline{P}' \in \mathcal{P}_{pp}^{prio}$ *such that* $\xi(P) \xrightarrow{m,\gamma:k} \overline{P}'$. *Then* $\exists P' \in \mathcal{P}^{cw}$. $\vdash_M^{cw} P \xrightarrow{\gamma} P'$ *and* $\overline{P}' \simeq_{pp} \xi(P')$ *where* $M = \mathbb{I}_{[m]}^{<k}(\xi(P))$.

Proof: Let $P \in \mathcal{P}^{\mathsf{cw}}$. For each statement a generalized version is proved by induction on the structure of P. The induction bases, i.e. $P \equiv \mathbf{0}$, are trivial since $\xi^l(P) \equiv \mathbf{0}$, for $l \in \mathcal{N}$, and $\mathbf{0}$ does not possess any transitions according to $\mathsf{CCS}_{\mathsf{pp}}^{\mathsf{prio}}$ and $\mathsf{CCS}^{\mathsf{cw}}$ semantics.

1. Let $P' \in \mathcal{P}^{\mathsf{cw}}$ and $\overline{a} \in \overline{\Lambda}$ such that $\vdash_{\emptyset}^{\mathsf{cw}} P \xrightarrow{\overline{a}} P'$. We show the more general statement $\xi^l(P) \xrightarrow{\overline{a}} \overline{P}'$ for some $\overline{P}' \simeq_{\mathsf{pp}} \xi(P')$ and all $l \in \mathcal{N}$.

   - $P \equiv \alpha.Q$**:** Hence, $\alpha \equiv \overline{a}$, $P' \equiv Q$, and $\xi^l(P) \equiv \overline{a}.\xi(P')$. Thus, $\xi^l(P) \xrightarrow{\overline{a}} \xi(P')$ by the operational rules for $\mathsf{CCS}_{\mathsf{pp}}^{\mathsf{prio}}$. By choosing $\overline{P}' \equiv \xi(P')$ we are done.
   - $P \equiv Q_1 +\!\!\rangle\, Q_2$**:** Because of the syntactic restriction in $\mathsf{CCS}^{\mathsf{cw}}$ this case cannot occur.
   - $P \equiv Q_1 + Q_2$**:** According to the operational rules for $\mathsf{CCS}^{\mathsf{cw}}$ we have w.l.o.g. $\vdash_{\emptyset}^{\mathsf{cw}} Q_1 \xrightarrow{\overline{a}} P'$. By induction hypothesis we know that $\xi^l(Q_1) \xrightarrow{\overline{a}} \overline{P}'$ for some $\overline{P}' \simeq_{\mathsf{pp}} \xi(P')$. Hence, $\xi^l(P) \equiv \xi^l(Q_1 + Q_2) \equiv \xi^l(Q_1) \oplus \xi^l(Q_2) \xrightarrow{\overline{a}} \overline{P}'$ according to the definition of $\xi^l(\cdot)$ and the operational rules for $\mathsf{CCS}_{\mathsf{pp}}^{\mathsf{prio}}$.
   - $P \equiv Q_1 \,|\, Q_2$**:** By the operational rules for $\mathsf{CCS}^{\mathsf{cw}}$ we may conclude w.l.o.g. $\vdash_{\emptyset}^{\mathsf{cw}} Q_1 \xrightarrow{\overline{a}} Q_1'$ and $P' \equiv Q_1' \,|\, Q_2$. By induction hypothesis we obtain $\xi^l(Q_1) \xrightarrow{\overline{a}} \overline{Q}_1'$ for some $\overline{Q}_1' \simeq_{\mathsf{pp}} \xi(Q_1')$. Thus, $\xi^l(P) \equiv \xi^l(Q_1 \,|\, Q_2) \equiv \xi^l(Q_1) \,|\, \xi^l(Q_2) \xrightarrow{\overline{a}} \overline{Q}_1' \,|\, \xi^l(Q_2)$ according to the definition of $\xi(\cdot)$ and the operational rules for $\mathsf{CCS}_{\mathsf{pp}}^{\mathsf{prio}}$. Finally, we have $\overline{Q}_1' \,|\, \xi^l(Q_2) \simeq_{\mathsf{pp}} \xi(Q_1') \,|\, \xi(Q_2) \equiv \xi(Q_1' \,|\, Q_2) \equiv \xi(P')$ by Lemma 6.7.6, Proposition 6.7.2, and the definition of the translation function.

   The remaining cases are easier to establish and, therefore, are omitted.

2. In order to prove the second statement, it needs to be generalized in the following fashion.

   > Let $P \in \mathcal{P}^{\mathsf{cw}}$, $\overline{P}' \in \mathcal{P}_{\mathsf{pp}}^{\mathsf{prio}}$, $\overline{a} \in \overline{\Lambda}$, and $l \in \mathcal{N}$ such that $\xi^l(P) \xrightarrow{\overline{a}} \overline{P}'$. Then $\exists P' \in \mathcal{P}^{\mathsf{cw}}$. $\vdash_{\emptyset}^{\mathsf{cw}} P \xrightarrow{\overline{a}} P'$ and $\overline{P}' \simeq_{\mathsf{pp}} \xi(P')$.

   The proof is not difficult and can be done using a similar argumentation as above.

3. For proving this statement by induction over the structure of P we have to generalize its conclusion as follows: $\forall l \in \mathcal{N}\ \exists k \in \mathcal{N}, m \in \mathcal{L}oc, \overline{P}' \in \mathcal{P}_{\mathsf{pp}}^{\mathsf{prio}}$.

$$\xi^l(P) \xrightarrow{m,\gamma:lk} \overline{P}', \overline{P}' \simeq_{\mathsf{pp}} \xi(P'), \text{ and } M = \mathbb{I}_{[m]}^{<lk}(\xi^l(P)) \ .$$

   Therefore, let $P' \in \mathcal{P}^{\mathsf{cw}}$, $\gamma \in \Lambda \cup \{\tau\}$, and $M \subseteq \Lambda$ such that $\vdash_M^{\mathsf{cw}} P \xrightarrow{\gamma} P'$, and let $l \in \mathcal{N}$.

- $P \equiv \alpha.Q$: Hence, $\alpha \equiv \gamma$, $P' \equiv Q$, $M = \emptyset$, and $\xi^l(P) \equiv \gamma{:}l.\xi(P')$ according to $\mathsf{CCS}^{\mathsf{cw}}$ semantics and the definition of the translation function. Thus, $\xi^l(P) \xrightarrow{\bullet,\gamma:l} \xi(P')$ by the operational rules for $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ and, moreover, $\mathbb{I}^{<l}_{[\bullet]}(\xi^l(P)) = \mathbb{I}^{<l}_{[\bullet]}(\gamma{:}l.\xi(P')) = \emptyset = M$ according to the definition of $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ initial input action sets.
- $P \equiv Q_1 +\!\rangle\, Q_2$: According to the operational semantics of $\mathsf{CCS}^{\mathsf{cw}}$ we distinguish the following two cases.
  (a) $\vdash^{\mathsf{cw}}_M Q_1 \xrightarrow{\gamma} P'$: By induction hypothesis we know of the existence of some $k \in \mathcal{N}$, $m \in \mathcal{L}oc$, and $\overline{P}' \in \mathcal{P}^{\mathsf{prio}}_{\mathsf{pp}}$ such that $\xi^{l0}(Q_1) \xrightarrow{m,\gamma:l0k} \overline{P}'$, $\overline{P}' \simeq_{\mathsf{pp}} \xi(P')$, and $M = \mathbb{I}^{<l0k}_{[m]}(\xi^{l0}(Q_1))$. Hence, $\xi^l(P) \equiv \xi^l(Q_1 +\!\rangle\, Q_2) \equiv \xi^{l0}(Q_1) + \xi^{l1}(Q_2) \xrightarrow{m\cdot l,\gamma:l0k} \overline{P}'$ and $\mathbb{I}^{<l0k}_{[m\cdot l]}(\xi^l(P)) = \mathbb{I}^{<l0k}_{[m]}(\xi^{l0}(Q_1)) \cup \mathbb{I}^{<l0k}(\xi^{l1}(Q_2)) = \mathbb{I}^{<l0k}_{[m]}(\xi^{l0}(Q_1)) = M$ according to the definition of $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ initial input action sets and the definition of the translation function.
  (b) $\vdash^{\mathsf{cw}}_N Q_2 \xrightarrow{\gamma} P'$, *where* $M = N \cup \mathbb{I}^{\mathsf{cw}}(Q_1)$, *and* $\tau, \gamma \notin \mathrm{I}^{\mathsf{cw}}(Q_1)$: By induction hypothesis we know of the existence of some $k \in \mathcal{N}$, $m \in \mathcal{L}oc$, and $\overline{P}' \in \mathcal{P}^{\mathsf{prio}}_{\mathsf{pp}}$ such that $\xi^{l1}(Q_2) \xrightarrow{m,\gamma:l1k} \overline{P}'$, $\overline{P}' \simeq_{\mathsf{pp}} \xi(P')$, and $N = \mathbb{I}^{<l1k}_{[m]}(\xi^{l1}(Q_2))$. Hence, $\xi^l(P) \equiv \xi^l(Q_1 +\!\rangle\, Q_2) \equiv \xi^{l0}(Q_1) + \xi^{l1}(Q_2) \xrightarrow{m\cdot r,\gamma:l1k} \overline{P}'$ since $\tau, \gamma \notin \mathrm{I}^{\mathsf{cw}}(Q_1) = \mathrm{I}(\xi^{l0}(Q_1))$ according to Lemma 6.7.7. Also, $\mathbb{I}^{<l1k}_{[m\cdot r]}(\xi^l(P)) = \mathbb{I}^{<l1k}(\xi^{l0}(Q_1)) \cup \mathbb{I}^{<l1k}_{[m]}(\xi^{l1}(Q_2)) = \mathbb{I}(\xi^{l0}(Q_1)) \cup N = \mathbb{I}^{\mathsf{cw}}(Q_1) \cup N = M$ by the definition of $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ initial input action sets and by Lemma 6.7.7, as desired.
- $P \equiv Q_1 + Q_2$: According to the operational rules for $\mathsf{CCS}^{\mathsf{cw}}$ we assume w.l.o.g. that $\vdash^{\mathsf{cw}}_M Q_1 \xrightarrow{\gamma} P'$. By induction hypothesis we know of the existence of some $k \in \mathcal{N}$, $m \in \mathcal{L}oc$, and $\overline{P}' \in \mathcal{P}^{\mathsf{prio}}_{\mathsf{pp}}$ such that $\xi^l(Q_1) \xrightarrow{m,\gamma:lk} \overline{P}'$, $\overline{P}' \simeq_{\mathsf{pp}} \xi(P')$, and $M = \mathbb{I}^{<lk}_{[m]}(\xi^l(Q_1))$. Hence, $\xi^l(P) \equiv \xi^l(Q_1 + Q_2) \equiv \xi^l(Q_1) \oplus \xi^l(Q_2) \xrightarrow{m\cdot L,\gamma:lk} \overline{P}'$ according to the translation function and the $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ operational rules. Moreover, $\mathbb{I}^{<lk}_{[m\cdot L]}(\xi^l(P)) = \mathbb{I}^{<lk}_{[m\cdot L]}(\xi^l(Q_1 + Q_2)) = \mathbb{I}^{<lk}_{[m\cdot L]}(\xi^l(Q_1) \oplus \xi^l(Q_2)) = \mathbb{I}^{<lk}_{[m]}(\xi^l(Q_1)) = M$ by the definition of $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ initial input action sets and the definition of the translation function.
- $P \equiv Q_1 \,|\, Q_2$: According to the definition of the operational rules for $\mathsf{CCS}^{\mathsf{cw}}$ we distinguish the following cases.
  (a) $\vdash^{\mathsf{cw}}_M Q_1 \xrightarrow{\gamma} Q_1'$, $M \cap \overline{\mathbb{I}}^{\mathsf{cw}}(Q_2) = \emptyset$, *and* $P' \equiv Q_1' \,|\, Q_2$: By induction hypothesis we know of the existence of some $k \in \mathcal{N}$, $m \in \mathcal{L}oc$, and $\overline{Q}_1' \in \mathcal{P}^{\mathsf{prio}}_{\mathsf{pp}}$ such that $\xi^l(Q_1) \xrightarrow{m,\gamma:lk} \overline{Q}_1'$, $\overline{Q}_1' \simeq_{\mathsf{pp}} \xi(Q_1')$, and $M = \mathbb{I}^{<lk}_{[m]}(\xi^l(Q_1))$. According to Lemma 6.7.7 we have $M \cap \overline{\mathbb{I}}(Q_2) = M \cap \overline{\mathbb{I}}^{\mathsf{cw}}(Q_2) = \emptyset$. Hence by the operational rules for $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ and the definition of $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ initial input action sets, $\xi^l(P) \equiv \xi^l(Q_1 \,|\, Q_2) \equiv \xi^l(Q_1) \,|\, \xi^l(Q_2) \xrightarrow{m\cdot L,\gamma:lk} \overline{Q}_1' \,|\, \xi^l(Q_2)$ and $\mathbb{I}^{<lk}_{[m\cdot L]}(\xi^l(P)) = \mathbb{I}^{<lk}_{[m\cdot L]}(\xi^l(Q_1) \,|\, \xi^l(Q_2)) = \mathbb{I}^{<lk}_{[m]}(\xi^l(Q_1)) = M$. Additionally, we have $\overline{Q}_1' \,|\, \xi^l(Q_2) \simeq_{\mathsf{pp}} \xi(Q_1') \,|\, \xi(Q_2) \equiv \xi(Q_1' \,|\, Q_2) \equiv \xi(P')$ by Lemma 6.7.6 and the definition of $\xi(\cdot)$.

(b) $\vdash_M^{cw} Q_2 \xrightarrow{\gamma} Q_2'$, $M \cap \overline{\mathbb{I}}^{cw}(Q_1) = \emptyset$, *and* $P' \equiv Q_1 \mid Q_2'$*:* This case is analogous to the previous one.

(c) $\vdash_M^{cw} Q_1 \xrightarrow{a} Q_1'$, $\vdash_\emptyset^{cw} Q_2 \xrightarrow{\overline{a}} Q_2'$ *for some* $a \in \Lambda$, $\gamma \equiv \tau$, $M \cap \overline{\mathbb{I}}^{cw}(Q_2) = \emptyset$, *and* $P' \equiv Q_1' \mid Q_2'$*:* By induction hypothesis we know of the existence of some $k \in \mathcal{N}$, $m \in \mathcal{L}oc$, and $\overline{Q}_1' \in \mathcal{P}_{\mathsf{pp}}^{\mathsf{prio}}$ such that $\xi^l(Q_1) \xrightarrow{m,a:lk} \overline{Q}_1'$, $\overline{Q}_1' \simeq_{\mathsf{pp}} \xi(Q_1')$, and $M = \mathbb{I}_{[m]}^{<lk}(\xi^l(Q_1))$. By the proof of Part (1) of this proposition we also know that $\xi^l(Q_2) \xrightarrow{\overline{a}} \overline{Q}_2'$ for some $\overline{Q}_2' \simeq_{\mathsf{pp}} \xi(Q_2')$. According to Lemma 6.7.7 we have $M \cap \overline{\mathbb{I}}(Q_2) = M \cap \overline{\mathbb{I}}^{\mathsf{cw}}(Q_2) = \emptyset$. Thus, we may conclude by the operational rules for $\mathsf{CCS}_{\mathsf{pp}}^{\mathsf{prio}}$ that $\xi^l(P) \equiv \xi^l(Q_1 \mid Q_2) \equiv \xi^l(Q_1) \mid \xi^l(Q_2) \xrightarrow{m \cdot L, \tau : lk} \overline{Q}_1' \mid \overline{Q}_2'$. We also have $\overline{Q}_1' \mid \overline{Q}_2' \simeq_{\mathsf{pp}} \xi(Q_1') \mid \xi(Q_2') \equiv \xi(Q_1' \mid Q_2') \equiv \xi(P')$ according to Proposition 6.7.2 and the definition of the translation function. Finally, $\mathbb{I}_{[m \cdot L]}^{<lk}(\xi^l(P)) = \mathbb{I}_{[m]}^{<lk}(\xi^l(Q_1)) = M$ by the definition of $\mathsf{CCS}_{\mathsf{pp}}^{\mathsf{prio}}$ initial input action sets, as desired.

(d) $\vdash_M^{cw} Q_2 \xrightarrow{a} Q_2'$, $\vdash_\emptyset^{cw} Q_1 \xrightarrow{\overline{a}} Q_1'$ *for some* $a \in \Lambda$, $\gamma \equiv \tau$, $M \cap \overline{\mathbb{I}}^{cw}(Q_1) = \emptyset$, *and* $P' \equiv Q_1' \mid Q_2'$*:* This case can be proved similar to the previous one by using the generalized statement in the proof of Part (2) instead of the one in Part (1) of this proposition.

The remaining cases are omitted since they are easier to establish than the presented ones.

4. Also the forth statement needs to be generalized.

   Let $P \in \mathcal{P}^{\mathsf{cw}}$, $m \in \mathcal{L}oc$, $\gamma \in \Lambda \cup \{\tau\}$, $k, l \in \mathcal{N}$, and $\overline{P}' \in \mathcal{P}_{\mathsf{pp}}^{\mathsf{prio}}$ such that $\xi^l(P) \xrightarrow{m,\gamma:lk} \overline{P}'$. Then $\exists P' \in \mathcal{P}^{\mathsf{cw}}$. $\vdash_M^{\mathsf{cw}} P \xrightarrow{\gamma} P'$ and $\overline{P}' \simeq_{\mathsf{pp}} \xi(P')$ where $M = \mathbb{I}_{[m]}^{<lk}(\xi^l(P))$.

   The generalized statement can be proved by similar argumentation as in the previous case and the necessary case distinctions are analogous to the ones in the proof of the second statement.

The properties of interest follow from the generalized versions by choosing $l = \epsilon$. □

The above proposition provides the foundation for our main theorem which states that $\mathsf{CCS}^{\mathsf{cw}}$ processes are distributed prioritized strong bisimular for $\mathsf{CCS}^{\mathsf{cw}}$ exactly when their translations are distributed prioritized strong bisimular for $\mathsf{CCS}_{\mathsf{pp}}^{\mathsf{prio}}$.

Theorem 6.7.9 (Semantic Correspondence)
For $P, Q \in \mathcal{P}^{cw}$ *we have:* $P \simeq_{cw} Q$ *if and only if* $\xi(P) \simeq_{pp} \xi(Q)$.

Proof: For the "if"-direction we prove in the following that the symmetric relation $\mathcal{R} =_{\mathrm{df}} \{\langle P, Q\rangle \mid \xi(P) \simeq_{\mathsf{pp}} \xi(Q)\}$ is a distributed prioritized strong bisimulation for $\mathsf{CCS}^{\mathsf{cw}}$. Therefore, let $\langle P, Q\rangle \in \mathcal{R}$ be arbitrary, i.e. $\xi(P) \simeq_{\mathsf{pp}} \xi(Q)$.

1. Let $\vdash^{\mathsf{cw}}_{\emptyset} P \xrightarrow{\overline{a}} P'$ for some $P' \in \mathcal{P}^{\mathsf{cw}}$ and some $\overline{a} \in \overline{\Lambda}$. According to Proposition 6.7.8(1) $\xi(P) \stackrel{\overline{a}}{\Rightarrow} \overline{P}'$ holds for some $\overline{P}' \in \mathcal{P}^{\mathsf{prio}}_{\mathsf{pp}}$ satisfying $\overline{P}' \simeq_{\mathsf{pp}} \xi(P')$. Since $\xi(P) \simeq_{\mathsf{pp}} \xi(Q)$ we know of the existence of some $\overline{Q}' \in \mathcal{P}^{\mathsf{cw}}$ such that $\xi(Q) \stackrel{\overline{a}}{\Rightarrow} \overline{Q}'$ and $\overline{P}' \simeq_{\mathsf{pp}} \overline{Q}'$. Applying Proposition 6.7.8(2) we obtain $\vdash^{\mathsf{cw}}_{\emptyset} Q \xrightarrow{\overline{a}} Q'$ for some $Q' \in \mathcal{P}^{\mathsf{cw}}$ satisfying $\overline{Q}' \simeq_{\mathsf{pp}} \xi(Q')$. Hence, $\langle P', Q' \rangle \in \mathcal{R}$ since $\xi(P') \simeq_{\mathsf{pp}} \overline{P}' \simeq_{\mathsf{pp}} \overline{Q}' \simeq_{\mathsf{pp}} \xi(Q')$, as desired.

2. Let $\gamma \in \Lambda \cup \{\tau\}$, $M \subseteq \Lambda$, and $P' \in \mathcal{P}^{\mathsf{cw}}$ such that $\vdash^{\mathsf{cw}}_{M} P \xrightarrow{\gamma} P'$. By Proposition 6.7.8(3) we know of the existence of $k \in \mathcal{N}$, $m \in \mathcal{L}oc$, and $\overline{P}' \in \mathcal{P}^{\mathsf{prio}}_{\mathsf{pp}}$ such that $\xi(P) \xrightarrow{m,\gamma:k} \overline{P}' \simeq_{\mathsf{pp}} \xi(P')$ and $M = \mathbb{I}^{<k}_{[m]}(\xi(P))$. Since $\xi(P) \simeq_{\mathsf{pp}} \xi(Q)$ there exist $\overline{Q}' \in \mathcal{P}^{\mathsf{prio}}_{\mathsf{pp}}$, $n \in \mathcal{L}oc$, and $l \in \mathcal{N}$ satisfying $\xi(Q) \xrightarrow{n,\gamma:l} \overline{Q}'$, $N =_{\mathrm{df}} \mathbb{I}^{<l}_{[n]}(\xi(Q)) \subseteq M$, and $\overline{P}' \simeq_{\mathsf{pp}} \overline{Q}'$. Moreover, we obtain by Proposition 6.7.8(4) some $Q' \in \mathcal{P}^{\mathsf{cw}}$ such that $\overline{Q}' \simeq_{\mathsf{pp}} \xi(Q')$ and $\vdash^{\mathsf{cw}}_{N} Q \xrightarrow{\gamma} Q'$. Finally, we have $\langle P', Q' \rangle \in \mathcal{R}$ since $\xi(P') \simeq_{\mathsf{pp}} \overline{P}' \simeq_{\mathsf{pp}} \overline{Q}' \simeq_{\mathsf{pp}} \xi(Q')$.

Hence, $\xi(P) \simeq_{\mathsf{pp}} \xi(Q)$ implies $P \simeq_{\mathsf{cw}} Q$ according to Definition 6.7.5. For the "only if"-direction we prove that the symmetric relation

$$\mathcal{R} =_{\mathrm{df}} \{\langle \overline{P}, \overline{Q} \rangle \mid \exists P', Q' \in \mathcal{P}^{\mathsf{cw}}.\ \overline{P} \simeq_{\mathsf{pp}} \xi(P'),\ \overline{Q} \simeq_{\mathsf{pp}} \xi(Q'),\ \text{and } P' \simeq_{\mathsf{cw}} Q'\}$$

is a distributed prioritized strong bisimulation for $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$. Therefore, let $\langle \overline{P}, \overline{Q} \rangle \in \mathcal{R}$ be arbitrary, i.e. $\exists P', Q' \in \mathcal{P}^{\mathsf{cw}}.\ \overline{P} \simeq_{\mathsf{pp}} \xi(P')$, $\overline{Q} \simeq_{\mathsf{pp}} \xi(Q')$, and $P' \simeq_{\mathsf{cw}} Q'$.

1. Let $\overline{P} \stackrel{\overline{a}}{\Rightarrow} \overline{P}'$ for some $\overline{P}' \in \mathcal{P}^{\mathsf{prio}}_{\mathsf{pp}}$ and some $\overline{a} \in \overline{\Lambda}$. Since $\overline{P} \simeq_{\mathsf{pp}} \xi(P')$ we know of the existence of some $\overline{P}'' \in \mathcal{P}^{\mathsf{prio}}_{\mathsf{pp}}$ such that $\xi(P') \stackrel{\overline{a}}{\Rightarrow} \overline{P}''$ and $\overline{P}' \simeq_{\mathsf{pp}} \overline{P}''$. According to Proposition 6.7.8(2) there exists some $P'' \in \mathcal{P}^{\mathsf{cw}}$ satisfying $\overline{P}'' \simeq_{\mathsf{pp}} \xi(P'')$ and $\vdash^{\mathsf{cw}}_{\emptyset} P' \xrightarrow{\overline{a}} P''$. Because of $P' \simeq_{\mathsf{cw}} Q'$ there also exists some $Q'' \in \mathcal{P}^{\mathsf{cw}}$ such that $\vdash^{\mathsf{cw}}_{\emptyset} Q' \xrightarrow{\overline{a}} Q''$ and $P'' \simeq_{\mathsf{cw}} Q''$. Hence by Proposition 6.7.8(1), $\xi(Q') \stackrel{\overline{a}}{\Rightarrow} \overline{Q}''$ for some $\overline{Q}'' \simeq_{\mathsf{pp}} \xi(Q'')$. Now, we use the fact that $\overline{Q} \simeq_{\mathsf{pp}} \xi(Q')$ to conclude the existence of some $\overline{Q}' \in \mathcal{P}^{\mathsf{prio}}_{\mathsf{pp}}$ such that $\overline{Q} \stackrel{\overline{a}}{\Rightarrow} \overline{Q}'$ and $\overline{Q}' \simeq_{\mathsf{pp}} \overline{Q}''$. Finally, $\langle \overline{P}', \overline{Q}' \rangle \in \mathcal{R}$ according to the definition of $\mathcal{R}$.

2. Let $\overline{P} \xrightarrow{m,\gamma:k} \overline{P}'$ for some $\overline{P}' \in \mathcal{P}^{\mathsf{prio}}_{\mathsf{pp}}$, $m \in \mathcal{L}oc$, $\gamma \in \Lambda \cup \{\tau\}$, and $k \in \mathcal{N}$. Since $\overline{P} \simeq_{\mathsf{pp}} \xi(P')$ we know of the existence of some $m' \in \mathcal{L}oc$, $k' \in \mathcal{N}$, and $\overline{P}'' \in \mathcal{P}^{\mathsf{prio}}_{\mathsf{pp}}$ such that $\xi(P') \xrightarrow{m',\gamma:k'} \overline{P}''$, $\mathbb{I}^{<k'}_{[m']}(\xi(P')) \subseteq \mathbb{I}^{<k}_{[m]}(\overline{P})$, and $\overline{P}' \simeq_{\mathsf{pp}} \overline{P}''$. According to Proposition 6.7.8(4) there exists some $P'' \in \mathcal{P}^{\mathsf{cw}}$ satisfying $\overline{P}'' \simeq_{\mathsf{pp}} \xi(P'')$ and $\vdash^{\mathsf{cw}}_{M} P' \xrightarrow{\gamma} P''$ where $M = \mathbb{I}^{<k'}_{[m']}(\xi(P'))$. Because of $P' \simeq_{\mathsf{cw}} Q'$ there also exists some $Q'' \in \mathcal{P}^{\mathsf{cw}}$ and $N \subseteq M$ such that $\vdash^{\mathsf{cw}}_{N} Q' \xrightarrow{\gamma} Q''$ and $P'' \simeq_{\mathsf{cw}} Q''$. Hence by Proposition 6.7.8(3), $\xi(Q') \xrightarrow{n',\gamma:l'} \overline{Q}''$ for some $n' \in \mathcal{L}oc$, $l' \in \mathcal{N}$, and $\overline{Q}'' \simeq_{\mathsf{pp}} \xi(Q'')$ such that $N = \mathbb{I}^{<l'}_{[n']}(\xi(Q'))$. Due to $\overline{Q} \simeq_{\mathsf{pp}} \xi(Q')$ we conclude the existence of some $n \in \mathcal{L}oc$, $l \in \mathcal{N}$, and $\overline{Q}' \in \mathcal{P}^{\mathsf{prio}}_{\mathsf{pp}}$ such that $\overline{Q} \xrightarrow{n,\gamma:l} \overline{Q}'$, $\mathbb{I}^{<l}_{[n]}(\overline{Q}) \subseteq \mathbb{I}^{<l'}_{[n']}(\xi(Q'))$, and

$\overline{Q}' \simeq_{\mathsf{pp}} \overline{Q}''$. Adding the small steps together, we finally obtain $\mathbb{I}^{<l}_{[n]}(\overline{Q}) \subseteq \mathbb{I}^{<k}_{[m]}(\overline{P})$, and $\langle \overline{P}', \overline{Q}' \rangle \in \mathcal{R}$ according to the definition of $\mathcal{R}$.

Thus, $P \simeq_{\mathsf{cw}} Q$ implies $\xi(P) \simeq_{\mathsf{pp}} \xi(Q)$ by Definition 6.7.1 which completes the proof. □

As a consequence, distributed prioritized strong bisimulation for $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ is also compositional with respect to summation for the sub-calculus of $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ induced by $\mathsf{CCS}^{\mathsf{cw}}$. This points again to our richer language which allows us to place a process $P \in \mathcal{P}^{\mathsf{prio}}_{\mathsf{pp}}$ in a summation context that possesses initial input actions having priority values which lie in between that range of priority values, that is limited by the priority values of the initial input actions of P.

6.8 Discussion and Related Work

Several proposals have been made for extending traditional process algebras with priority. They differ in the aspects of computation, such as interrupts [10], programming constructs like the `PRIALT` construct of `occam` [43, 102], and constructs of `Ada` [79], or real-time [83], that they aim to capture.

An extension of CCS [125] with priority, $\mathsf{CCS}^{\mathsf{ch}}$, has been proposed in Chapter 2, where priorities are also assigned to actions in a *globally dynamic* way, i.e. in one state of a system action α may have priority over action β while the situation may be reversed in another state of the system. For $\mathsf{CCS}^{\mathsf{ch}}$ a semantic theory has been developed in an analogous fashion to [125] which includes congruences based on strong and weak bisimulation. Our process algebra $\mathsf{CCS}^{\mathsf{prio}}$ is based on the approach in Chapter 2, where we adopt all design decisions except the notion of *global* pre-emption and restrict ourselves to a two-level priority-scheme. Therefore, $\mathsf{CCS}^{\mathsf{prio}}$ has the following characteristics. Only transitions labeled by complementary actions with the same priority may engage in a synchronization. As in Chapter 2 we consider actions with different priorities as *different* channels which is sufficient for most cases occurring in practice. The strong relation of $\mathsf{CCS}^{\mathsf{prio}}$ to $\mathsf{CCS}^{\mathsf{ch}}$ can be made precise by the following fact. If we leave out the distributed summation operator and globalize pre-emption in our framework by defining $[m] =_{\mathrm{df}} \mathcal{L}oc$ for all $m \in \mathcal{L}oc$, our operational semantics and our behavioral relations reduce to the corresponding notions presented in Chapter 2, when ignoring the disabling operator in $\mathsf{CCS}^{\mathsf{ch}}$.

For a comparison with other work one should note that existing approaches that assign priorities to actions are provided with a semantics dealing with *global* pre-emption. In contrast, we consider a notion of *local* pre-emption. This idea is also presented in [89], where a CSP-based calculus [94] is extended with priority. However, this process algebra suffers from a complicated semantics, especially for the hiding operator, and the authors only conjecture that their strong bisimulation is a congruence. They also do not provide an axiomatization for their equivalence and do not present a theory for observational congruence. Other approaches to introducing priority in CSP have been proposed by Fidge [79] and Lowe [116] who are both concerned with a semantic theory based on *failures*.

As shown in the previous section, having a notion of *local* pre-emption enables one to avoid some problems arising in the assignment of priority values to synchronizations involving actions at different priority levels. In traditional approaches, which assign priorities to actions [83, 89], several proposals for adjustment functions, e.g. taking the maximum, minimum, or the sum of priority values, have been made. Unfortunately, in settings involving *global* pre-emption each solution is only intuitive for certain (classes of) examples and works only for certain frameworks without violating the congruence property of the considered behavioral relation.

In the previous section we have proved that Camilleri and Winskel's calculus [43, 102], $\mathsf{CCS}^{\mathsf{cw}}$, where priorities arising from different sides of the parallel composition operator are also considered to be incomparable, is a sub-calculus of $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$. For $\mathsf{CCS}^{\mathsf{cw}}$ a prioritized parallel composition operator $\between$ is introduced in [102] which favors its left argument over its right one. It can be used in descriptions of simple scheduling algorithms. We say "simple" because most scheduling algorithms deal with priority values that may change as the system evolves (cf. Chapter 4). This *dynamic* behavior cannot be described in the framework of [102], which considers *static* priorities only. In contrast, we concentrate on modeling interrupts and prioritized choice and show that this requires the concept of *local pre-emption* when dealing with distributed systems. However, scheduling is a global task and, therefore, it is based on a notion of *global* pre-emption. We feel that describing interrupts and scheduling algorithms should be done using two different priority concepts. More precisely, each action should be assigned two priority values, the first interpreted as global priority value for scheduling purposes and the second as local priority value for modeling interrupts where the first priority value has more weight than the second one.

After stressing similarities and differences of $\mathsf{CCS}^{\mathsf{prio}}$ to the process algebra $\mathsf{CCS}^{\mathsf{cw}}$ presented in [43, 102] with respect to design decisions we focus on the algebraic results established in these frameworks. In [43, 102] the pre-emption potential is directly encoded in the transition relation. By plugging in this transition relation into the definition of standard strong bisimulation one obtains a congruence relation immediately. In contrast, we start off defining *naive* distributed prioritized strong bisimulation using the naive transition relation and consider the pre-emption potential subsequently (by introducing the distributed prioritized initial action set condition). Then we show that the so obtained congruence is the largest congruence in the naive distributed prioritized strong bisimulation. Similarly, a naive distributed prioritized weak bisimulation is defined in [102] by using the above mentioned transition relation, which already reflects some pre-emption potential. More precisely, this transition relation corresponds to our distributed prioritized weak transition relation where we drop the parameter M. Thus, our naive distributed prioritized weak transition relation is more abstract than the one in [102]. In both approaches, the cited and the presented one, the obtained distributed prioritized observational congruence is shown to be the largest congruence in the naive distributed prioritized weak bisimulation. However, our result is tighter since our naive distributed prioritized weak bisimulation is coarser.

We turn to some remarks about our notion of distributed prioritized strong and weak bisimulation. Since our semantic theory reflects local pre-emption, locations are implicitly occurring in our semantic equivalences. However, in contrast to [33, 132] we do not consider locations explicitly. Our objective is not to observe locations but to observe local pre-emption which is necessary for causal reasoning in process algebras with "distributed" priority.

Priorities have also been investigated in other concurrent frameworks, most notably in *Petri Nets* [148, 168]. In this setting priorities are either expressed explicitly by priority relations over transitions [24] or implicitly via *inhibitor arcs* [100]. In the latter work priorities are modeled via the absence of tokens, i.e. a transition can fire if some predecessor places do not contain any tokens. In contrast to Petri Nets, the focus of process algebras lies in examining properties of behavioral relations with respect to the operators included in the considered algebra. Thus, it is difficult to compare priority approaches of both areas, Petri Nets and process algebras, from a semantic point of view.

Finally, priorities can implicitly arise when studying causality for *mobile processes* (see e.g. [74]). In these approaches priorities cut off superfluous paths that only present new temporal but not causal dependencies of systems. Hence, this kind of priority is equipped with a global nature of pre-emption. In contrast, the local view of pre-emption in $\mathsf{CCS}^{\mathsf{prio}}$ is used for restricting the causal, not the temporal, behavior of distributed systems.

6.9 Summary

We have presented a process algebra, $\mathsf{CCS}^{\mathsf{prio}}$, that is capable of modeling interrupts and other prioritized behavior in distributed systems. The key idea for $\mathsf{CCS}^{\mathsf{prio}}$ is to take into account the distribution of the considered system in order to define a notion of *local* pre-emption. We have developed a bisimulation-based semantic theory for this algebra, including identifications of the largest congruence contained in naive adaptations of strong and weak bisimulation [125], respectively. Using enriched transition systems our behavioral relations have been characterized operationally, such that standard partition-refinement algorithms [103, 138] for their computation become applicable, as well as logically, based on alterations of the Hennessy-Milner logic [125]. Moreover, the algebraic utility of $\mathsf{CCS}^{\mathsf{prio}}$ has been shown by way of an example, involving a typical pattern that can be found in many distributed systems. In order to gain a better understanding of the relationship of our approach to Camilleri and Winskel's algebra $\mathsf{CCS}^{\mathsf{cw}}$ [43, 102] we have removed two design decisions of $\mathsf{CCS}^{\mathsf{prio}}$ that have been adopted from Chapter 2. The obtained "generalized" version of $\mathsf{CCS}^{\mathsf{prio}}$, called $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$, has been proved to contain $\mathsf{CCS}^{\mathsf{cw}}$ as a sub-algebra.

Chapter 7

A Process Algebra with Multiple Clocks

7.1 Introduction

In this chapter we present a temporal process algebra, called CSA (Calculus for Synchrony and Asynchrony), which is aimed at modeling *distributed*, timed systems that contain a number of independent clocks. Technically, CSA extends Hennessy and Regan's TPL [92] with operators from Andersen and Mendler's PMC [5] for managing multiple clocks. In doing so we replace the global notion of *maximal progress* found in TPL with a local one that is more appropriate for distributed systems. This combination of features yields a convenient formalism for modeling distributed timed systems; it also introduces semantic subtleties the solutions to which constitute the body of this chapter.

In CSA there is an important shift of concern underlying the use of multiple clocks [5] for modeling time in distributed systems. Traditional real-time process algebras, e.g. the algebra $\mathsf{CCS}^{\mathsf{rt}}$ presented in Chapter 4, attempt to give a precise and complete description of the *quantitative* system behavior, by measuring the distance or the duration of actions in terms of arithmetic values, for instance natural numbers (*discrete time*) or real numbers (*dense time*). In contrast, the notion of time underlying the concept of multiple clocks, in particular in connection with maximal progress, is a *qualitative* one. In this respect CSA follows the philosophy advocated by Nicollin and Sifakis [136] and others, as well as synchronous languages such as ESTEREL [20, 21] (cf. Section 8.2). Clocks capture the qualitative nature of timing constraints, in which it is not the absolute occurrence time or duration of actions that is constrained but their relative ordering and sequencing with respect to clocks. In other words, the primary purpose of clocks is to synchronize and to reduce nondeterminism, not in fact to be counted and to be arithmetized by numbers.

This chapter is organized as follows. In the next section an example motivates the utility of multiple clocks and maximal progress for modeling temporal distributed systems. Section 7.3 defines the syntax and semantics of CSA, while Section 7.4 applies our language to the motivating example. In the following section strong bisimulation [125] is adapted

to our language, and its theory, including axiomatic, operational, and logical characterizations, is developed. The corresponding observational theory is presented in Section 7.6. In Section 7.7 we use our calculus to reason about the example system while the following section is concerned with introducing a clock-hiding operator to CSA. Section 7.9 discusses our approach and related work whereas the final section summarizes this chapter.

7.2 Motivating Example

One standard hardware architecture consists of a number of cooperating synchronous systems which are distributed over different modules, e.g. chips or boards. Typically, each module possesses its own central clock to update all of its registers in a synchronous fashion. The clocks of different modules are independent, so that the modules change their states asynchronously with respect to each other. The modules communicate with the help of synchronizing registers or buffers, which store the output value of a module until it is read by another one. Such architectures are also called *globally-asynchronous, locally-synchronous* [44]. They not only arise through physical distribution, e.g. in computer networks where different sites cannot be synchronized by the same clock, but are also typical for heterogeneous real-time applications. A concrete example is the Brüel & Kjær 2145 Vehicle Signal Analyzer reported in [6], a portable real-time measurement equipment, which consists of several asynchronously clocked processors each of which performs some dedicated signal processing function.

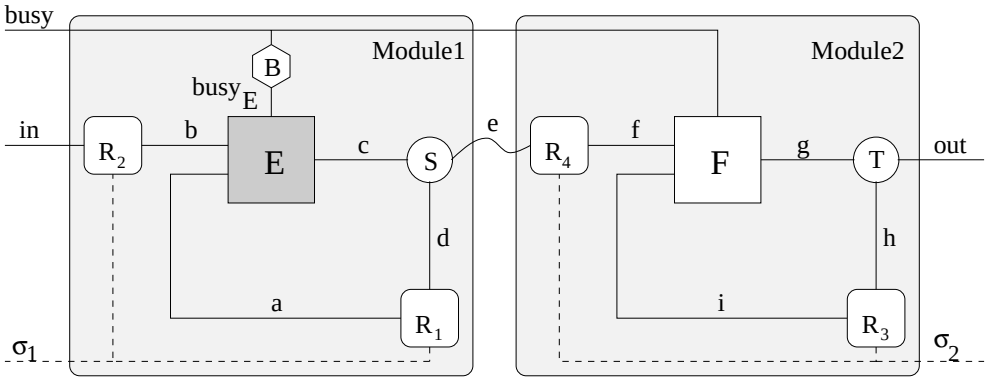

Figure 7.1: A globally-asynchronous, locally-synchronous system

A generic example for a globally-asynchronous, locally-synchronous system is depicted in Figure 7.1, where solid lines represent communication channels and dashed lines symbolize channels of clocks. Both modules, `Module1` and `Module2`, have their own local clocks σ_1 and σ_2, respectively, and their own function blocks, registers, and buffers. In every clock cycle, the function block `E` computes a new value from the current values of the state registers $\mathtt{R}_1$ and $\mathtt{R}_2$, obtained through channels a and b, and outputs it on channel c to be propagated further through `S` to register $\mathtt{R}_1$ via channel d and to the environment via chan-

nel e. External input enters the computation through channel *in*. Register $\mathtt{R}_2$ stores the most recent input value from the environment and, thus, ensures that `E` never has to wait for the environment. Component `S` and busy buffer `B` are explained later. `Module2` operates in a similar fashion, with its external input being fed by the output of `Module1`. Assuming that the clocks σ_1 and σ_2 are independent and continually tick away, this example provides a simplified but typical instance of a globally-asynchronous, locally-synchronous system.

The example clearly suggests how we can benefit from a concept of multiple clocks to model real-world distributed systems. The question of what is an adequate notion of clock leads us to the second characteristic of our process algebra: the *maximal progress assumption*. Let us take a more detailed look at the workings of `Module1`. The fundamental, in fact the defining, feature of clock σ_1 is that it must tick only after the previous clock cycle has been completed, i.e. after the function block `E` has finished its internal computation and the new value has arrived at register $\mathtt{R}_1$. Otherwise, the value stored into $\mathtt{R}_1$ upon the tick of σ_1 is undefined. If `Module1` has more than one register reading d they may all take different values, and an inconsistent state may arise. The maximal progress assumption guarantees that a clock tick is delayed until all internal computations or communications have come to an end. In the example we assume that the ongoing internal computations of `E`, which may implement a complex function or subprogram, are indicated to `Module1` by communications on channel $\mathtt{busy}_\mathtt{E}$.

To take account of distribution, the maximal progress property must be "localized" and imposed on every module independently. For instance, clock σ_1 of `Module1` must be able to tick as soon as the previous cycle of σ_1 has been completed, regardless of the state of `Module2`, which operates asynchronously with respect to `Module1`. In contrast, the traditional global version of maximal progress would imply that *all* clocks have to wait for *all* computations to complete, whence the system would be globally synchronous. This global maximal progress assumption cannot be implemented within a distributed architecture since one site cannot recognize whether internal computation local to another site has been completed. Since a process algebra ideally should be a formal basis for an implementable programming language, our local version of the maximal progress assumption is more adequate for distributed systems.

The example applies at various levels of abstraction. At the *gate level* the registers correspond to flipflops and the function blocks to combinational circuits. At the *system level* we may talk about memory and processors and the clocks become abstract synchronization signals. Further, at the *algorithm level* `Module1` and `Module2` may be software modules. We can think of function blocks as *procedures* and *statements*, and of registers as *state variables*.

The combined concept of multiple clocks and local maximal progress is quite powerful. It supports *horizontal* and *vertical* forms of synchronous decomposition that correspond to temporal abstractions with synchronized and nested scales of time. The horizontal form arises if we compose `Module1` and `Module2` in parallel with *identical* clocks $\sigma_1 = \sigma_2$. Then, maximal progress automatically ensures that both systems operate in perfect

synchrony on a single global state space defined by registers $\mathtt{R}_1$ to $\mathtt{R}_4$. The vertical form arises when we implement, say, the function `E` of `Module1` as a whole synchronous system in itself, with its own local clock ρ (see Section 7.4). Then, maximal progress ensures that the outer clock σ_1 waits for the inner computation in `E` to complete, which may involve an arbitrary number of ρ cycles. The localization of maximal progress means, on the other hand, that the inner clock ρ is independent of σ_1, so that the local computations in `E` may overlap with the local computations in function blocks that are external, i.e. parallel, to `E`. In software the vertical refinement would correspond to a procedure call, which spins off a subcomputation with its own local state space and clock, which upon completion returns control to the outer sequencing. Both kinds of decompositions are very useful when designing distributed systems, as they correspond to temporal abstractions with synchronized and nested scales of time. Examples are the *micro* and *macro* time of Statecharts [91, 143], the *zero-delay*, *δ-delay*, and *real-time* levels of VHDL [108], and the synchrony hypothesis of ESTEREL [20, 21] (see also Section 8.2).

7.3 Syntax and Semantics of CSA

In this section we define the syntax and semantics of our language CSA, which is inspired by the process algebras TPL [92] and PMC [5], that both descend from ATP [136]. The syntax of CSA is essentially the same as in PMC; it extends Milner's CCS [125] with a *timeout operator* and a clock *ignore operator*. As for $\mathsf{CCS}^{\mathsf{rt}}$ (cf. Chapter 4), the semantical framework of CSA is based on a notion of labeled transition system that involves two kinds of transitions, *action* transitions and *clock* transitions, modeling two different mechanisms of synchronization and communication in distributed systems. Action transitions, like in CCS, are local handshake communications in which two processes synchronize to take a joint state change together. A clock represents the progress of time, which manifests itself in a recurrent global synchronization event, the clock transition, in which all process components that are in the regime, or in the scope, of this clock are forced to take part. In CSA action and clock transitions are not orthogonal concepts that can be specified independently from each other, but are connected in line with the following intuitions:

- A clock records the *progress* of time, with two successive clock events marking an interval of time.
- The *passage* of time is determined by internal computation that is within the regime of the clock.

This yields the very specific semantic connection between actions and clocks, known as the *maximal progress assumption* [92, 173] or as the *synchrony hypothesis* [21]. Maximal progress usually is read as the condition that "communications must occur whenever they are possible," i.e. a process cannot be intercepted by a clock as long as it is able to perform internal computation. Thus, *progress* of time is determined by the *completion* of internal computation.

The last feature of CSA is *clock scoping*. Since we are dealing with distributed systems and multiple clocks, it is natural to localize the maximal progress assumption with respect to clocks and to limit the scope of clocks. A communication that reaches outside the scope is an external computation that must be considered asynchronous with respect to the clock. It has no fixed relation to the clock, and may finish either before or after the local clock tick. Different clocks, which represent different local views of time, may have disjoint, overlapping, or nested scopes, and amount to different abstractions of time.

Note that clocks in our setting are abstract in the sense that we do not prejudice any particular way to interpret them. We are free to think of a clock as the ticking of a global real-time watch measuring off absolute process time in constant or non-constant intervals, as the system clock of a synchronous processor, as a recurrent external interrupt, or as the completion signal of a distributed synchronization protocol. Thus, clocks can be used as a general and flexible means for bundling asynchronous behavior into sequenced intervals, and to give local meaning to the notions of "before," "after," and "state."

7.3.1 Syntax of CSA

Formally, the syntax of CSA is based on the one of PMC [5]. Let Λ be a countably infinite set of *labels* or *ports*, not including the so-called *silent* or *internal* action τ. With every $a \in \Lambda$ we associate a *complementary* action $\overline{a}$. Intuitively, an action $a \in \Lambda$ may be thought of as representing the receipt of an input on port a, while $\overline{a}$ constitutes the deposit of an output on a. We define $\overline{\Lambda} =_{\mathrm{df}} \{\overline{a} \mid a \in \Lambda\}$ and take $\mathcal{A}$ to denote the set of all actions $\Lambda \cup \overline{\Lambda} \cup \{\tau\}$, where $\tau \notin \Lambda \cup \overline{\Lambda}$. Complementation is lifted to $\overline{\Lambda}$ by defining $\overline{\overline{a}} =_{\mathrm{df}} a$. As in CCS [125] an action a communicates with its complement $\overline{a}$ to produce the internal action τ. We let $a, b, \ldots$ range over $\Lambda \cup \overline{\Lambda}$ and $\alpha, \beta, \ldots$ over $\mathcal{A}$. Besides the set $\mathcal{A}$ of actions, CSA is parameterized in a countable set $\mathcal{T} = \{\sigma, \sigma', \rho, \ldots\}$ of *clocks*. There is no further structure on this set. Also, we do not distinguish between a clock symbol, the actual clock event, and the imaginary clock mechanism that produces these events. The syntax of our language is defined by the following BNF

$$P ::= \mathbf{0} \mid x \mid \alpha.P \mid P + P \mid P \,|\, P \mid P[f] \mid P \setminus L \mid P{\uparrow}\sigma \mid \lfloor P \rfloor \sigma(P) \mid \mu x.P$$

where x is a *variable* taken from a countable set of variables $\mathcal{V}$, $f : \mathcal{A} \to \mathcal{A}$ is a *finite relabeling*, and $L \subseteq \mathcal{A} \setminus \{\tau\}$ is a *restriction set*. For notational convenience, we define $\overline{L} =_{\mathrm{df}} \{\overline{a} \mid a \in L\}$. A finite relabeling satisfies the properties $f(\tau) = \tau$, $f(\overline{a}) = \overline{f(a)}$, and $|\{\alpha \mid f(\alpha) \neq \alpha\}| < \infty$. Similar to the previous chapters we write $[\beta_1/\alpha_1, \beta_2/\alpha_2, \ldots, \beta_n/\alpha_n]$ for the relabeling f defined by $f(\alpha_i) = \beta_i$ for $1 \leq i \leq n$ and $f(\alpha) = \alpha$ for all $\alpha \not\equiv \alpha_i$, $1 \leq i \leq n$.. Moreover, ${\uparrow}\sigma$ is called the (static) *ignore operator* and $\lfloor \cdot \rfloor \sigma(\cdot)$ the *timeout operator*. Further, we use the standard definitions for the *sort* of a term P, $\mathcal{S}(P) \subseteq \Lambda \cup \overline{\Lambda}$, *static* and *dynamic* operators, *free* and *bound* variables, *open* and *closed* terms, and *contexts*. A variable is called *guarded* in a term if each occurrence of the variable is in the scope of a prefix or of the second argument of a timeout (see below). Syntactic substitution of variable x by Q in term P is written as $P[Q/x]$ in the remainder. We refer to closed and

guarded terms as *processes*. Let $\mathcal{P}$ be the set of all processes, ranged over by $P, Q, R, \ldots$, and let us denote syntactic equality by $\equiv$. In order to avoid too many parentheses we refine the binding conventions of CCS. Our operators have decreasing binding strength in the order *restriction* and *relabeling* and *ignore*, *prefixing* and *timeout*, *parallel composition*, *summation*, and *recursion*. We extend the timeout operator to sequences of clocks by $\lfloor P \rfloor =_{\mathrm{df}} P$ and $\lfloor P \rfloor \sigma_1(Q_1) \cdots \sigma_n(Q_n) =_{\mathrm{df}} \lfloor \lfloor P \rfloor \sigma_1(Q_1) \cdots \sigma_{n-1}(Q_{n-1}) \rfloor \sigma_n(Q_n)$. Moreover, we often further abbreviate sequences $\sigma_1 \ldots \sigma_n$ of clocks by $\vec{\sigma}$ and sequences $Q_1 \ldots Q_n$ of processes by $\vec{Q}$. In this vein, $\lfloor P \rfloor \vec{\sigma}(\vec{Q})$ is a shorthand for $\lfloor P \rfloor \sigma_1(Q_1) \cdots \sigma_n(Q_n)$. Finally, we abuse notation and write $\sigma \in \vec{\sigma}$ if $\sigma = \sigma_i$ for $1 \leq i \leq n$ and $\vec{\sigma} \subseteq T$ for some $T \subseteq \mathcal{T}$ if $\sigma_i \in T$ for all $1 \leq i \leq n$.

7.3.2 Semantics of CSA

The *operational semantics* of a CSA process $P \in \mathcal{P}$ is given by a labeled transition system $\langle \mathcal{P}, \mathcal{A} \cup \mathcal{T}, \longrightarrow, P \rangle$ where $\mathcal{P}$ is the set of states, $\mathcal{A} \cup \mathcal{T}$ the alphabet, $\longrightarrow \subseteq \mathcal{P} \times (\mathcal{A} \cup \mathcal{T}) \times \mathcal{P}$ the transition relation, and P the start state. We refer to transitions with labels in $\mathcal{A}$ as *action transitions*, and to those with labels in $\mathcal{T}$ as *clock transitions*. The transition relation is defined in Table 7.3 using operational rules [141]. For the sake of simplicity, let us use γ for a representative of $\mathcal{A} \cup \mathcal{T}$, and write $P \xrightarrow{\gamma} P'$ instead of $\langle P, \gamma, P' \rangle \in \longrightarrow$. We say that P *may engage in action or clock tick* γ *and thereafter behave like process* P'. Sometimes it is convenient to write $P \xrightarrow{\gamma}$ for $\exists P' \in \mathcal{P}.\, P \xrightarrow{\gamma} P'$.

Table 7.1: Initial action sets for CSA

$\mathcal{I}(\alpha.P) =_{\mathrm{df}} \{\alpha\}$	$\mathcal{I}(\mu x.P) =_{\mathrm{df}} \mathcal{I}(P[\mu x.P/x])$
$\mathcal{I}(P + Q) =_{\mathrm{df}} \mathcal{I}(P) \cup \mathcal{I}(Q)$	$\mathcal{I}(P \mid Q) =_{\mathrm{df}} \mathcal{I}(P) \cup \mathcal{I}(Q) \cup \{\tau \mid \mathcal{I}(P) \cap \overline{\mathcal{I}(Q)} \neq \emptyset\}$
$\mathcal{I}(P[f]) =_{\mathrm{df}} \{f(\alpha) \mid \alpha \in \mathcal{I}(P)\}$	$\mathcal{I}(P \setminus L) =_{\mathrm{df}} \mathcal{I}(P) \setminus (L \cup \overline{L})$
$\mathcal{I}(\lfloor P \rfloor \sigma'(Q)) =_{\mathrm{df}} \mathcal{I}(P)$	$\mathcal{I}(P \uparrow \sigma') =_{\mathrm{df}} \mathcal{I}(P)$

To ensure maximal progress the operational rules involve side conditions on *initial action sets*. Beside the usual definition of $\mathcal{I}(P)$ for the initial action set of a process P as the smallest set satisfying the equations in Table 7.1, we define the set $\mathcal{I}_\sigma(P) \subseteq \mathcal{I}(P)$ of all initial actions of P *within the scope* of the clock σ as the smallest set satisfying the equations in Table 7.2. Moreover, $\mathcal{I}_\sigma(P) = \mathcal{I}(P)$ whenever P does not contain any ignore operator. Finally, we define initial *visible* action sets by $\mathcal{I}\!\!\mathcal{I}(P) =_{\mathrm{df}} \mathcal{I}(P) \setminus \{\tau\}$ and $\mathcal{I}\!\!\mathcal{I}_\sigma(P) =_{\mathrm{df}} \mathcal{I}_\sigma(P) \setminus \{\tau\}$.

Table 7.2: Clock scoping

$\mathcal{I}_\sigma(\alpha.P) =_{\text{df}} \{\alpha\}$	$\mathcal{I}_\sigma(\mu x.P) =_{\text{df}} \mathcal{I}_\sigma(P[\mu x.P/x])$
$\mathcal{I}_\sigma(P+Q) =_{\text{df}} \mathcal{I}_\sigma(P) \cup \mathcal{I}_\sigma(Q)$	$\mathcal{I}_\sigma(P \mid Q) =_{\text{df}} \mathcal{I}_\sigma(P) \cup \mathcal{I}_\sigma(Q) \cup \{\tau \mid \mathcal{I}_\sigma(P) \cap \overline{\mathcal{I}_\sigma(Q)} \neq \emptyset\}$
$\mathcal{I}_\sigma(P[f]) =_{\text{df}} \{f(\alpha) \mid \alpha \in \mathcal{I}_\sigma(P)\}$	$\mathcal{I}_\sigma(P \setminus L) =_{\text{df}} \mathcal{I}_\sigma(P) \setminus (L \cup \overline{L})$
$\mathcal{I}_\sigma(\lfloor P \rfloor \sigma'(Q)) =_{\text{df}} \mathcal{I}_\sigma(P)$	$\mathcal{I}_\sigma(P \uparrow \sigma') =_{\text{df}} \mathcal{I}_\sigma(P)$ if $\sigma \not\equiv \sigma'$

The operational semantics for action transitions extends the one of CCS by rules dealing with the ignore and the timeout operator. More precisely, the process $\alpha.P$ may engage in action α and then behave like P. The *summation operator* $+$ denotes nondeterministic choice, i.e. the process $P+Q$ may either behave like P or Q. The *restriction operator* $\backslash L$ prohibits the execution of actions in $L \cup \overline{L}$ and thus permits the scoping of actions. $P[f]$ behaves exactly as P where ordinary actions are renamed by the *relabeling* f. The process $P|Q$ stands for the *parallel composition* of P and Q according to an interleaving semantics with synchronized communication on complementary actions resulting in the internal action τ. The processes $P \uparrow \sigma$ and $\lfloor P \rfloor \sigma(Q)$, involving the ignore and timeout operator, respectively, behave like P for action transitions. The timeout operator disappears as soon as P engages in an action transition, thereby observably changing its state. Finally, $\mu x.\, P$ denotes *recursion*, i.e. $\mu x.\, P$ is a process which behaves as a distinguished solution of the equation $x = P$.

With respect to clock transitions the operational semantics is set up such that if $\tau \in \mathcal{I}_\sigma(P)$ then clock σ is inhibited. We refer to this kind of pre-emption as *local maximal progress*. Its local nature lies in the facts that, in general, $\mathcal{I}_\sigma(P) \neq \mathcal{I}(P)$ and that the sets $\mathcal{I}_\sigma(P)$ may be different for different clocks. Accordingly, the process $\alpha.P$ may idle for each clock σ whenever $\alpha \not\equiv \tau$. Time has to proceed equally on both sides of summation, i.e. $P+Q$ can engage in a clock transition and thus delay the nondeterministic choice if and only if both P and Q can engage in the clock tick. Also both argument processes of a parallel composition have to synchronize on clock transitions according to Rule (tCom). Its side condition implements local maximal progress and can alternatively be written as $\mathcal{I\!I}_\sigma(P) \cap \overline{\mathcal{I\!I}_\sigma(Q)} = \emptyset$, i.e. there is no pending communication between P and Q on an action that lies in the scope of σ. Clocks can neither be restricted nor relabeled, so the clock transitions of $P[f]$ and $P \setminus L$ are the same as those of P. The clock transitions for a recursive process $\mu x.P$ are generated by syntactic unfolding in the same way as for action transitions. Regarding the ignore operator, the process $P \uparrow \sigma$ is capable of performing a σ-

Table 7.3: Operational semantics for CSA

Act	$\dfrac{-}{\alpha.P \xrightarrow{\alpha} P}$	tAct	$\dfrac{-}{a.P \xrightarrow{\sigma} a.P}\ \sigma \in \mathcal{T}$
Sum1	$\dfrac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'}$	tNil	$\dfrac{-}{\mathbf{0} \xrightarrow{\sigma} \mathbf{0}}\ \sigma \in \mathcal{T}$
Sum2	$\dfrac{Q \xrightarrow{\alpha} Q'}{P+Q \xrightarrow{\alpha} Q'}$	tSum	$\dfrac{P \xrightarrow{\sigma} P' \quad Q \xrightarrow{\sigma} Q'}{P+Q \xrightarrow{\sigma} P'+Q'}$
Rel	$\dfrac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f(\alpha)} P'[f]}$	tRel	$\dfrac{P \xrightarrow{\sigma} P'}{P[f] \xrightarrow{\sigma} P'[f]}$
Res	$\dfrac{P \xrightarrow{\alpha} P'}{P \setminus L \xrightarrow{\alpha} P' \setminus L}\ \alpha \notin L \cup \overline{L}$	tRes	$\dfrac{P \xrightarrow{\sigma} P'}{P \setminus L \xrightarrow{\sigma} P' \setminus L}$
Com1	$\dfrac{P \xrightarrow{\alpha} P'}{P\|Q \xrightarrow{\alpha} P'\|Q}$	tCom	$\dfrac{P \xrightarrow{\sigma} P' \quad Q \xrightarrow{\sigma} Q'}{P\|Q \xrightarrow{\sigma} P'\|Q'}\ \tau \notin \mathcal{I}_\sigma(P\|Q)$
Com2	$\dfrac{Q \xrightarrow{\alpha} Q'}{P\|Q \xrightarrow{\alpha} P\|Q'}$	tIgn1	$\dfrac{-}{P\uparrow\sigma \xrightarrow{\sigma} P\uparrow\sigma}$
Com3	$\dfrac{P \xrightarrow{a} P' \quad Q \xrightarrow{\overline{a}} Q'}{P\|Q \xrightarrow{\tau} P'\|Q'}$	tIgn2	$\dfrac{P \xrightarrow{\sigma'} P'}{P\uparrow\sigma \xrightarrow{\sigma'} P'\uparrow\sigma}\ \sigma \not\equiv \sigma'$
Ign	$\dfrac{P \xrightarrow{\alpha} P'}{P\uparrow\sigma \xrightarrow{\alpha} P'\uparrow\sigma}$	tTO1	$\dfrac{-}{\lfloor P \rfloor\sigma(Q) \xrightarrow{\sigma} Q}\ \tau \notin \mathcal{I}_\sigma(P)$
TO	$\dfrac{P \xrightarrow{\alpha} P'}{\lfloor P \rfloor\sigma(Q) \xrightarrow{\alpha} P'}$	tTO2	$\dfrac{P \xrightarrow{\sigma'} P'}{\lfloor P \rfloor\sigma(Q) \xrightarrow{\sigma'} P'}\ \sigma \not\equiv \sigma'$
Rec	$\dfrac{P[\mu x.P/x] \xrightarrow{\alpha} P'}{\mu x.P \xrightarrow{\alpha} P'}$	tRec	$\dfrac{P[\mu x.P/x] \xrightarrow{\sigma} P'}{\mu x.P \xrightarrow{\sigma} P'}$

loop, i.e. P *ignores* σ, regardless if $\tau \in \mathcal{I}_\sigma(P)$ or not. This is consistent with our definition $\mathcal{I}_\sigma(P\uparrow\sigma) =_{\mathrm{df}} \emptyset$, which means that none of the initial actions of P is in the scope of clock σ.

Thus, $\uparrow\sigma$ is actually not a *scoping* but a *co-scoping* operator, i.e. all processes are assumed to be within the scope of all clocks unless explicitly excluded using an ignore. Dealing with co-scoping instead of scoping simplifies the operational rules in a setting with multiple local clocks, since the traditional rules for summation and parallel composition with respect to clock transitions do not need to be changed. Moreover, the process $\lfloor P \rfloor \sigma(Q)$ can perform a σ-transition to Q provided that P cannot engage in an internal action which is in the scope of clock σ. Since a clock transition also represents an observable change of state, the timeout operator disappears as soon as P engages in such a transition. This intuition is the same as for the corresponding unit-delay operator in ATP [136]. For multiple clocks this leads to Rule (tTO2) in which the timeout for σ is dropped when a different σ' ticks. The idea is that the ordering of the σ and σ' ticks is observable and the first one determines the state change. Note, however, that by using recursion to insert explicit clock idling persistent versions of the timeout can be obtained. Following the usual conventions, an operator is called *static* (with respect to actions) if, at top-level, it survives all action transitions; otherwise it is *dynamic* (with respect to actions). The summation operator is treated as dynamic although it is static with respect to clock transitions.

In some proofs given below we construct contexts which include a generalization of the summation operator indexed over actions contained in sorts. Since CSA provides a binary summation operator, i.e. only finite summations can be expressed, the following lemma is important to show the well-definedness of these contexts.

Lemma 7.3.1 (Finite Sorts) *The sort $\mathcal{S}(P)$ of every process $P \in \mathcal{P}$ is finite.*

This observation is an immediate consequence of the fact that process terms are finite, and relabelings f satisfy the condition $|\{\alpha \,|\, f(\alpha) \not\equiv \alpha\}| < \infty$.

The operational semantics for CSA possesses several important properties which are presented in the next three propositions. The statement of the first proposition is essential for the congruence proofs of some of the behavioral relations presented in the next sections. It states that a process can always engage in a clock transition provided that it cannot perform an internal action which is in the scope of this clock ρ (cf. Proposition 4.3.1(1)). The validity of the proposition is a consequence of the processes $\mathbf{0}$ and $a.P$ being able to idle for all clocks.

Proposition 7.3.2 (Idling Capability)
Let $P \in \mathcal{P}$ and $\sigma \in \mathcal{T}$ satisfying $\tau \notin \mathcal{I}_\sigma(P)$. Then, $P \xrightarrow{\sigma}$ holds.

Moreover, the semantics satisfies the *local maximal progress* and the *local time determinacy* property. Both are generalizations of the well-known maximal progress and time determinacy properties (cf. Propositions 4.3.1(2) and 4.3.1(3), respectively), for global time, to a local notion of time in terms of multiple, local clocks.

Proposition 7.3.3 (Local Maximal Progress)
Let $P \in \mathcal{P}$ and $\sigma \in \mathcal{T}$. Then, $P \xrightarrow{\sigma}$ implies $\tau \notin \mathcal{I}_\sigma(P)$.

Local maximal progress is the converse of Proposition 7.3.2. Local time determinacy, which is a common feature of all real-time process algebras, states that processes react in a deterministic way to clock ticks, reflecting the intuition that the passage of time does not resolve choices.

Proposition 7.3.4 (Local Time Determinacy)
Let $P, P', P'' \in \mathcal{P}$ and $\sigma \in \mathcal{T}$ satisfying $P \xrightarrow{\sigma} P'$ and $P \xrightarrow{\sigma} P''$. Then, $P' \equiv P''$ holds.

7.4 Example (revisited)

Now let us return to our motivation and formally describe the example presented in Section 7.2 in our algebra CSA. We refer to Figure 7.1 and assume that we refine the function module `E` by a complete synchronous subsystem with its own local clock $\rho \in \mathcal{T} \supseteq \{\sigma_1, \sigma_2, \rho\}$, as depicted in Figure 7.2. At top level the structure of the overall system `System` is

$$(\texttt{Module1}\uparrow\sigma_2 \mid \texttt{Module2}\uparrow\sigma_1\uparrow\rho) \setminus \{e\} \ .$$

This captures the asynchronous parallel composition of `Module1` and `Module2`. The ignore operators $\uparrow \sigma_1$ and $\uparrow \sigma_2$ are introduced so as to make both modules ignore each other's clocks. The clock ρ is internal to `Module1`, whence `Module2` ignores it with $\uparrow \rho$. The channel e connecting both modules is internal to `System`, and thus restricted by $\setminus\{e\}$.

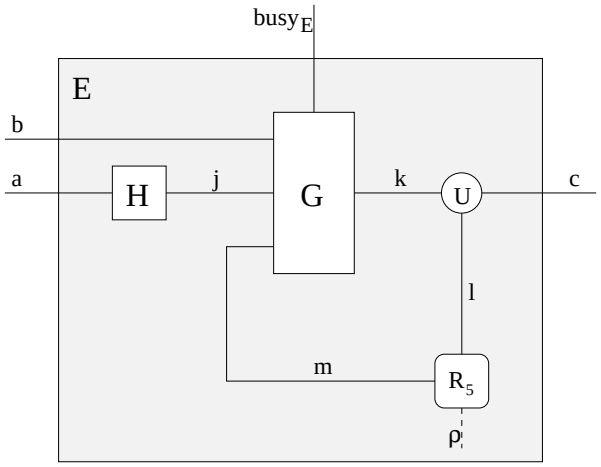

Figure 7.2: Component `E` (refined)

At the next structural level we break up the two asynchronous modules, each of which is a synchronous subsystem. Let us look at the internals of `Module1`, which is

$$(\texttt{E} \,|\, (\texttt{R}_1 \,|\, \texttt{R}_2 \,|\, \texttt{S} \,|\, \texttt{B})\uparrow\rho) \setminus \{a, b, c, d, \texttt{busy}_\texttt{E}\} \ .$$

It is a parallel composition of a function block `E`, state registers $\texttt{R}_1$ and $\texttt{R}_2$, a busy buffer `B`, and a special fork component `S`. These components communicate via the channels

$\{a, b, c, d, \mathtt{busy}_{\mathtt{E}}\}$ which are internal to `Module1` and, therefore, restricted away. The subsystem $(\mathtt{R}_1 \mid \mathtt{R}_2 \mid \mathtt{S} \mid \mathtt{B}) \upharpoonright \rho$ ignores the clock ρ, thus making ρ local to the function block `E`. This function block finally is decomposed to the synchronous system

$$\mathtt{E} \stackrel{\text{def}}{=} (\mathtt{H} \mid \mathtt{G} \mid \mathtt{R}_5 \mid \mathtt{U}) \setminus \{j, k, l, m\} \ .$$

Note that without having given the details for the "primitive" components, our description already captures the relevant synchronization structure of the example. By considering which primitive parallel component is in the scope of which ignore operator we can identify independent and nested synchronous subsystems.

Now let us turn our attention to the synchronous subsystem `E` (cf. Figure 7.2) and fill in some of the details. Block `E` should read its inputs from channels a and b, take an arbitrary number of cycles of clock ρ to compute a result that is then passed over to the environment on output channel c. The algorithm for this computation is contained in function block `G`. The register $\mathtt{R}_5$ stores the intermediate values, i.e. represents the local state on which the algorithm works. The function block `H` may be a preprocessing stage, and the component `U` is assumed to be a fork process that distributes the data on its input k to outputs l and c, i.e.

$$\mathtt{U} \stackrel{\text{def}}{=} \mu x.\, k.\overline{l}.(\mu y.\, \overline{c}.x + \tau.y) \ .$$

The τ-loop indicates waiting for external output through channel c, which will inhibit the local clock ρ as long as the output has not been delivered. In CSA the register might be specified as

$$\mathtt{R}_5 \stackrel{\text{def}}{=} \mu x.\, \lfloor l.x \rfloor \rho(\mu y.\, \overline{m}.x + l.y) \ .$$

It continuously accepts an updating input on channel l, and when the clock ρ ticks it changes its state to $\mu y.\, \overline{m}.\mathtt{R}_5 + l.y$. In this state the output action $\overline{m}$ starts the next computation cycle, while the l-loop makes sure that the register is always input-enabled. If our channels would carry real data then the new value injected into the next cycle with $\overline{m}$ would be the last value read from input l *before* the clock tick. This means that the l-loop *after* the clock must not change the registered value. From the value supplied by $\overline{m}$ after each clock tick, the function blocks compute a next state value that eventually ends up being latched into $\mathtt{R}_5$ again through l. Then the cycle is completed and ρ may tick again. To indicate the simplest case of a function block let `H` be the trivial iteration

$$\mathtt{H} \stackrel{\text{def}}{=} \mu x.\, a.\tau.\overline{j}.x \ .$$

Accordingly, `H` first reads an external input from a, then performs an internal computation represented by τ, and finally outputs on j, whereupon it returns to the initial state. For a function block with more than one input more complicated input-output patterns are possible. For instance, we want `G` to implement an algorithm that reads its inputs b and j

and initial state m and then computes its function in a number of steps, storing intermediate results in register $\mathtt{R}_5$. The following CSA process specifies such a behavior:

$$\mathtt{G} \stackrel{\text{def}}{=} \mu x.\, m.j.b.\mathtt{G}_1 \qquad \mathtt{G}_1 \stackrel{\text{def}}{=} \mu y.\, \tau.(\overline{k}.x + \overline{k}.\mathtt{G}_2) \qquad \mathtt{G}_2 \stackrel{\text{def}}{=} \mu z.\, m.y + \overline{\mathtt{busy}_{\mathtt{E}}}.z \ .$$

$\mathtt{G}$ consumes the register value on m and the result of function $\mathtt{H}$ on j, reads an input from the environment through channel b, and then passes to $\mathtt{G}_1$, which is the actual computation state. After a finite amount of internal computation, indicated by the leading τ, a result is computed that may be output with action $\overline{k}$. Now two possibilities arise: either the algorithm is completed, in which case we pass back to the initial state $\mathtt{G}$ (variable x), or we carry on with another clock cycle, in which case we move to state $\mathtt{G}_2$. This decision, of course, depends on the data, but since we do not consider values, we model this as a nondeterministic choice. In $\mathtt{G}_2$ we have reached a final state of a single ρ clock cycle, but only an intermediate state of the algorithm implemented by $\mathtt{E}$. The $\overline{\mathtt{busy}_{\mathtt{E}}}$-loop signals this to the environment of $\mathtt{E}$ in order to inhibit the outer clock σ_1. In the intermediate state $\mathtt{G}_2$ of the algorithm we do not need to read new input data, but only get the next state by m and continue with $\mathtt{G}_1$.

The above specification example shows how the ignore operator can be used to localize clocks, and the timeout operator to model the synchronized updating of registers. Maximal progress controls when a clock tick is possible and when it is delayed. The only way to stop a clock is by internal divergence, e.g. arising from a feed-back loop that does not contain any clocked register. In this case of a violated design rule, the function blocks produce divergence and the local clock of that module is never able to tick. This relationship between design error, internal divergence, and conceptual time stop is very natural for synchronous hardware, which stresses the adequacy of the maximal progress model.

7.5 A Semantic Theory based on Strong Bisimulation

The transition systems produced by the operational semantics are a rather fine-grained semantic view of processes. Usually, we are interested in more abstract behavioral aspects, and to capture those in terms of equations. For instance, for TPL [92] a testing semantics has been investigated and axiomatized. Here we follow PMC [5] and develop a semantic theory based on bisimulation [125]. Our aim in this section is to characterize the largest congruence contained in the "naive" strong bisimulation [125], where we treat clocks as actions, and to axiomatize this congruence for several classes of processes.

Definition 7.5.1 (Naive Temporal Strong Bisimulation)
A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is called naive strong bisimulation *if for every $\langle P, Q\rangle \in \mathcal{R}$, $\gamma \in \mathcal{A} \cup \mathcal{T}$, the following condition holds.*

$$P \stackrel{\gamma}{\longrightarrow} P' \text{ implies } \exists Q'.\, Q \stackrel{\gamma}{\longrightarrow} Q' \text{ and } \langle P', Q'\rangle \in \mathcal{R} \ .$$

We write $P \sim_{\mathrm{n}} Q$ if $\langle P, Q\rangle \in \mathcal{R}$ for some naive temporal strong bisimulation $\mathcal{R}$.

It is straightforward to establish that $\sim_{\mathrm{n}}$ is the *largest* naive temporal strong bisimulation and that $\sim_{\mathrm{n}}$ is an equivalence relation. Unfortunately, though not surprisingly, $\sim_{\mathrm{n}}$ is *not* a congruence. The reason is that the transition system of a process P does not contain the clock scoping information $\mathcal{I}_\sigma(P)$ needed to determine the transition system of $C[P]$ for all CSA contexts $C[X]$. For instance, $a.\mathbf{0} \sim_{\mathrm{n}} a.\mathbf{0}\!\uparrow\!\sigma$ but $a.\mathbf{0} \mid \overline{a}.\mathbf{0} \not\sim_{\mathrm{n}} (a.\mathbf{0}\!\uparrow\!\sigma) \mid \overline{a}.\mathbf{0}$ since the right-hand process can do a σ-transition while the corresponding σ-transition of the left-hand process is pre-empted due to maximal progress. In this example $a.\mathbf{0}$ and $a.\mathbf{0}\!\uparrow\!\sigma$ have identical transition systems but different clock scoping, effecting different pre-emption of clock transitions in parallel contexts. However, Proposition 2.4.3, adapted from universal algebra, shows a way to address the compositionality problem by stating the existence of the largest congruence contained in $\sim_{\mathrm{n}}$. Therefore, we devote the rest of this section in characterizing this largest congruence for which we have to take clock scoping into account.

7.5.1 Temporal Strong Bisimulation

The relation $\simeq$, which repairs the congruence defect of $\sim_{\mathrm{n}}$, is defined as follows.

Definition 7.5.2 (Temporal Strong Bisimulation)
A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a temporal strong bisimulation *if for every $\langle P, Q\rangle \in \mathcal{R}$, $\alpha \in \mathcal{A}$, and $\sigma \in \mathcal{T}$ the following conditions hold.*

1. *$P \xrightarrow{\alpha} P'$ implies $\exists Q'.\, Q \xrightarrow{\alpha} Q'$ and $\langle P', Q'\rangle \in \mathcal{R}$, and*
2. *$P \xrightarrow{\sigma} P'$ implies $\mathit{Ill}_\sigma(Q) \subseteq \mathit{Ill}_\sigma(P)$ and $\exists Q'.\, Q \xrightarrow{\sigma} Q'$ and $\langle P', Q'\rangle \in \mathcal{R}$.*

We write $P \simeq Q$ if $\langle P, Q\rangle \in \mathcal{R}$ for some temporal strong bisimulation $\mathcal{R}$.

The difference between this definition and that of $\sim_{\mathrm{n}}$ is the additional condition $\mathit{Ill}_\sigma(Q) \subseteq \mathit{Ill}_\sigma(P)$ concerning the clock scoping. The definition of $P \simeq Q$ requires not only that all clock transitions in P and Q must match each other, but also that with respect to all these clocks σ the pre-emption potential of both P and Q must be identical, i.e. $\mathit{Ill}_\sigma(P) = \mathit{Ill}_\sigma(Q)$. Note that nothing is required for those clocks σ that are stopped in P and Q. By Proposition 7.3.2 those are the ones for which $\tau \in \mathcal{I}_\sigma(P)$ and $\tau \in \mathcal{I}_\sigma(Q)$. Thus, we may replace the extra condition $\mathit{Ill}_\sigma(Q) \subseteq \mathit{Ill}_\sigma(P)$ by $\mathcal{I}_\sigma(Q) \subseteq \mathcal{I}_\sigma(P)$ and get the same relation. The equivalence $\simeq$ can be extended to open terms in the usual way. Now, we can state the following proposition.

Proposition 7.5.3 *The relation $\simeq$ is a congruence with respect to all operators, i.e. for all CSA contexts $C[X]$ we have: $P \simeq Q$ implies $C[P] \simeq C[Q]$.*

Proof: Since the semantics of the restriction and of the relabeling operators are the same as in the PMC–framework we obtain their compositionality relatively easily by adapting the results presented in [5] under the consideration of the definition of visible initial actions sets with respect to the scope of clocks. Also the compositionality of $\simeq$ for prefixing is easy to establish. Therefore, we concentrate on the more interesting cases. In the following let $P, Q, R, S \in \mathcal{P}$ be arbitrary processes such that $P \simeq Q$, $R \simeq S$, and $\sigma \in \mathcal{T}$. Then

1. $P \mid R \simeq Q \mid R$,
2. $P{\uparrow}\sigma \simeq Q{\uparrow}\sigma$, and
3. $\lfloor P \rfloor \sigma(R) \simeq \lfloor Q \rfloor \sigma(S)$.

The proofs of these statements follow the principle of bisimulation proofs as introduced in [125].

1. By the definition of $\simeq$ it is sufficient to prove that the symmetric relation

$$\mathcal{R} =_{\mathrm{df}} \{\langle P \mid R, Q \mid R\rangle \mid P \simeq Q\,, R \in \mathcal{P}\}$$

is a temporal strong bisimulation. Therefore, let $\langle P \mid R, Q \mid R\rangle \in \mathcal{R}$ be arbitrary.

- **Case 1:** The case $P \mid R \xrightarrow{\alpha} S$ for some $\alpha \in \mathcal{A}$ and $S \in \mathcal{P}$ follows the lines of the corresponding case in CCS [125] and, therefore, is omitted.
- **Case 2:** Let $P \mid R \xrightarrow{\sigma} S$ for some $\sigma \in \mathcal{T}$ and $S \in \mathcal{P}$. According to the only applicable semantic Rule (tCom) we know that $P \xrightarrow{\sigma} P'$ for some $P' \in \mathcal{P}$, $R \xrightarrow{\sigma} R'$ for some $R' \in \mathcal{P}$, $\mathit{\Pi}_\sigma(P) \cap \overline{\mathit{\Pi}_\sigma(R)} = \emptyset$, and $S \equiv P' \mid R'$. Since $P \simeq Q$ there exists a process $Q' \in \mathcal{P}$ such that $\mathit{\Pi}_\sigma(Q) \subseteq \mathit{\Pi}_\sigma(P)$, $Q \xrightarrow{\sigma} Q'$, and $P' \simeq Q'$. Therefore, we may conclude that $Q \mid R \xrightarrow{\sigma} Q' \mid R'$ by Rule (tCom) since $\mathit{\Pi}_\sigma(Q) \cap \overline{\mathit{\Pi}_\sigma(R)} = \emptyset$, and $\mathit{\Pi}_\sigma(Q \mid R) = \mathit{\Pi}_\sigma(Q) \cup \mathit{\Pi}_\sigma(R) \subseteq \mathit{\Pi}_\sigma(P) \cup \mathit{\Pi}_\sigma(R) = \mathit{\Pi}_\sigma(P \mid R)$ by the definition of $\mathit{\Pi}_\sigma(\cdot)$. Moreover, $\langle P' \mid R', Q' \mid R'\rangle \in \mathcal{R}$ holds by the definition of $\mathcal{R}$, which finishes the proof.

2. By Definition 7.5.2 it is sufficient to show that the symmetric relation

$$\mathcal{R} =_{\mathrm{df}} \{\langle P{\uparrow}\sigma, Q{\uparrow}\sigma\rangle \mid P \simeq Q\}$$

is a temporal strong bisimulation. Thus, let $\langle P{\uparrow}\sigma, Q{\uparrow}\sigma\rangle \in \mathcal{R}$.

- **Case 1:** Let $P \uparrow \sigma \xrightarrow{\alpha} P' \uparrow \sigma$ for some $P' \in \mathcal{P}$, i.e. $P \xrightarrow{\alpha} P'$ by Rule (Ign). Since $P \simeq Q$ we know of the existence of some $Q' \in \mathcal{P}$ such that $Q \xrightarrow{\alpha} Q'$ and $P' \simeq Q'$. Applying Rule (Ign) and the definition of $\mathcal{R}$ yields $Q{\uparrow}\sigma \xrightarrow{\alpha} Q'{\uparrow}\sigma$ and $\langle P'{\uparrow}\sigma, Q'{\uparrow}\sigma\rangle \in \mathcal{R}$.
- **Case 2:** Let $P \uparrow \sigma \xrightarrow{\sigma} P \uparrow \sigma$ by Rule (tIgn1). By the same rule we derive $Q \uparrow \sigma \xrightarrow{\sigma} Q \uparrow \sigma$, and by assumption we have $\langle P{\uparrow}\sigma, Q{\uparrow}\sigma\rangle \in \mathcal{R}$. Moreover, $\mathit{\Pi}_\sigma(Q{\uparrow}\sigma) = \emptyset = \mathit{\Pi}_\sigma(P{\uparrow}\sigma)$ holds, too.
- **Case 3:** Let $P \uparrow \sigma \xrightarrow{\sigma'} P' \uparrow \sigma$ for some $P' \in \mathcal{P}$ and $\sigma' \not\equiv \sigma$, i.e. $P \xrightarrow{\sigma'} P'$ by Rule (tIgn2). Since $P \simeq Q$ we know of the existence of some $Q' \in \mathcal{P}$ such that $\mathit{\Pi}_{\sigma'}(Q) \subseteq \mathit{\Pi}_{\sigma'}(P)$, $Q \xrightarrow{\sigma'} Q'$, and $P' \simeq Q'$. Applying Rule (tIgn2), the definition of $\mathcal{R}$, and the definition of $\mathit{\Pi}_{\sigma'}(\cdot)$ we may conclude $Q{\uparrow}\sigma \xrightarrow{\sigma'} Q'{\uparrow}\sigma$, $\langle P'{\uparrow}\sigma, Q'{\uparrow}\sigma\rangle \in \mathcal{R}$, and $\mathit{\Pi}_{\sigma'}(Q{\uparrow}\sigma) = \mathit{\Pi}_{\sigma'}(Q) \subseteq \mathit{\Pi}_{\sigma'}(P) = \mathit{\Pi}_{\sigma'}(P{\uparrow}\sigma)$.

3. It is sufficient to prove that the symmetric relation

$$\mathcal{R} =_{\mathrm{df}} \{\langle \lfloor P \rfloor \sigma(R), \lfloor Q \rfloor \sigma(S) \rangle \mid P \simeq Q \text{ and } R \simeq S\} \cup \simeq$$

is a temporal strong bisimulation. Consider a pair $\langle \lfloor P \rfloor \sigma(R), \lfloor Q \rfloor \sigma(S) \rangle \in \mathcal{R}$.

- **Case 1:** Let $\lfloor P \rfloor \sigma(R) \xrightarrow{\alpha} P'$ for some $P' \in \mathcal{P}$. According to the only applicable semantic Rule (TO) of CSA we know that $P \xrightarrow{\alpha} P'$. Since $P \simeq Q$ holds, the existence of some $Q' \in \mathcal{P}$ such that $Q \xrightarrow{\alpha} Q'$ and $P' \simeq Q'$ is guaranteed. By Rule (TO) we may conclude that $\lfloor Q \rfloor \sigma(S) \xrightarrow{\alpha} Q'$. Moreover, $\langle P', Q' \rangle \in \mathcal{R}$ by the definition of $\mathcal{R}$.
- **Case 2:** Let $\lfloor P \rfloor \sigma(R) \xrightarrow{\sigma} P'$ for some $P' \in \mathcal{P}$. By Rule (tTO1) we know that $P' \equiv R$ and $\tau \notin \mathcal{I}_\sigma(P)$. Moreover, Proposition 7.3.2 guarantees that $P \xrightarrow{\sigma}$. Because $P \simeq Q$ also $Q \xrightarrow{\sigma}$ is satisfied and $\mathcal{U}_\sigma(Q) \subseteq \mathcal{U}_\sigma(P)$. Thus, $\tau \notin \mathcal{I}_\sigma(Q)$ by our notion of maximal progress (cf. Proposition 7.3.3), $\lfloor Q \rfloor \sigma(S) \xrightarrow{\sigma} S$ by Rule (tTO1), and $\mathcal{U}_\sigma(\lfloor Q \rfloor \sigma(S)) = \mathcal{U}_\sigma(Q) \subseteq \mathcal{U}_\sigma(P) = \mathcal{U}_\sigma(\lfloor P \rfloor \sigma(Q))$ according to the definition of $\mathcal{U}_\sigma(\cdot)$. The observation $R \simeq S$ concludes this case.
- **Case 3:** Let $\lfloor P \rfloor \sigma(R) \xrightarrow{\sigma'} P'$ for some $P' \in \mathcal{P}$ where $\sigma' \not\equiv \sigma$. Thus, $P \xrightarrow{\sigma'} P'$ by Rule (tTO2). Since $P \simeq Q$ it follows that $\mathcal{U}_{\sigma'}(Q) \subseteq \mathcal{U}_{\sigma'}(P)$, $Q \xrightarrow{\sigma'} Q'$, and $P' \simeq Q'$ for some $Q' \in \mathcal{P}$. According to Rule (tTO2) we conclude that $\lfloor Q \rfloor \sigma(S) \xrightarrow{\sigma'} Q'$. Moreover, we have $\mathcal{U}_{\sigma'}(\lfloor Q \rfloor \sigma(S)) \subseteq \mathcal{U}_{\sigma'}(\lfloor P \rfloor \sigma(R))$ because $\mathcal{U}_{\sigma'}(\lfloor Q \rfloor \sigma(S)) = \mathcal{U}_{\sigma'}(Q)$ and $\mathcal{U}_{\sigma'}(\lfloor P \rfloor \sigma(R)) = \mathcal{U}_{\sigma'}(P)$. Since $\langle P', Q' \rangle \in \mathcal{R}$ by the definition of $\mathcal{R}$, we are done.

The proof of the compositionality of recursion requires to introduce a notion of *temporal strong bisimulation up to*. This definition and the compositionality proof itself can be done along the lines of CCS with respect to strong bisimulation (cf. [125]). □

The next theorem states the main result of this section. It borrows from ideas which have already been presented in the proof of Theorem 6.4.5.

Theorem 7.5.4 *Temporal observational congruence* $\simeq$ *is the* largest congruence *contained in naive temporal strong bisimulation* $\sim_{\mathrm{n}}$, *i.e.* $\simeq = \sim_{\mathrm{n}}^{+}$.

Proof: According to Proposition 7.5.3 the relation $\simeq$ is a congruence, which is contained in $\sim_{\mathrm{n}}$ since $\simeq$ can be shown to be a naive temporal strong bisimulation. By Proposition 2.4.3 we know that the largest congruence $\sim_{\mathrm{n}}^{+}$ in $\sim_{\mathrm{n}}$ exists and how it is characterized. Therefore, $\simeq \subseteq \sim_{\mathrm{n}}^{+}$, and it remains to show that $P \simeq Q$ for some processes $P, Q \in \mathcal{P}$ whenever $C[P] \sim_{\mathrm{n}} C[Q]$ for all CSA contexts $C[X]$. For this it suffices to consider the equivalence $\sim_{\mathrm{a}} =_{\mathrm{df}} \{\langle P, Q \rangle \mid C_{PQ}[P] \sim_{\mathrm{n}} C_{PQ}[Q]\}$. Here, $C_{PQ}[X] \stackrel{\mathrm{def}}{=} X \mid H_{PQ}$ and

$$H_{PQ} \stackrel{\mathrm{def}}{=} \mu x.(e.\mathbf{0} + \sum_{L \subseteq \overline{\mathcal{S}(P)} \cup \overline{\mathcal{S}(Q)}} \tau.(D_L + d_L.x))$$

where D_L is defined as $\sum_{d \in L} d.\mathbf{0}$. Note that H_{PQ} is well-defined because of Lemma 7.3.1. The actions e and d_L, and their complements, are supposed to be "fresh" actions. In this section we do not make use of the existence of the distinguished action e in the context but we do so when re-using this context in the proof of Theorem 7.6.8 below. In order to finish it is sufficient to prove the inclusion $\sim_{\mathrm{a}} \subseteq \simeq$ since the inclusion $\sim_{\mathrm{n}}^{+} \subseteq \sim_{\mathrm{a}}$ is obvious.

According to Definition 7.5.2 we show that $\sim_{\mathrm{a}}$ is a temporal strong bisimulation. Let $P, Q \in \mathcal{P}$ satisfying $P \sim_{\mathrm{a}} Q$, i.e. we have $C_{PQ}[P] \sim_{\mathrm{n}} C_{PQ}[Q]$ by the definition of $\sim_{\mathrm{a}}$. In the following, we consider two cases distinguishing if process P performs an action transition or a clock transition. In each case the transition of P leads to a transition of $C_{PQ}[P]$. According to the definition of $\sim_{\mathrm{n}}$ matching transitions must exist which mimic each step. From the existence of these transitions we may conclude additional conditions which are sufficient to establish $\sim_{\mathrm{a}}$ as a temporal strong bisimulation.

- **Situation 1:** Let $P \xrightarrow{\alpha} P'$ for some process $P' \in \mathcal{P}$ and some action $\alpha \in \mathcal{A}$. According to our operational semantics we have $C_{PQ}[P] \equiv P \,|\, H_{PQ} \xrightarrow{\alpha} P' \,|\, H_{PQ} \equiv C_{PQ}[P']$. This transition can only be matched by a corresponding transition of the process Q, say $Q \xrightarrow{\alpha} Q'$ for some $Q' \in \mathcal{P}$. This is even true in the case $\alpha \equiv \tau$ because the τ-successors of H_{PQ} have the distinguished actions d_L enabled. Therefore, we have $C_{PQ}[Q] \equiv Q \,|\, H_{PQ} \xrightarrow{\alpha} Q' \,|\, H_{PQ} \equiv C_{PQ}[Q']$ and $C_{PQ}[P'] \sim_{\mathrm{n}} C_{PQ}[Q']$. Because $\mathcal{S}(P') \subseteq \mathcal{S}(P)$ and $\mathcal{S}(Q') \subseteq \mathcal{S}(Q)$, also $C_{P'Q'}[P'] \sim_{\mathrm{n}} C_{P'Q'}[Q']$ holds. Thus, $P' \sim_{\mathrm{a}} Q'$.

- **Situation 2:** Let $P \xrightarrow{\sigma} P'$ for some process $P' \in \mathcal{P}$ and some clock $\sigma \in \mathcal{T}$. As illustrated in Figure 7.3, we let $C_{PQ}[P]$ perform a τ-transition to $P \,|\, H_L$, where $H_L \stackrel{\mathrm{def}}{=} D_L + d_L.H_{PQ}$ and $L =_{\mathrm{df}} \{\overline{c} \,|\, c \in (\mathcal{S}(P) \cup \mathcal{S}(Q)) \setminus \mathcal{I}\!\!\mathcal{I}_\sigma(P)\}$. Then, $P \,|\, H_L$ can perform a σ-transition to $P' \,|\, H_L$ according to Rule (tCom). Finally, we let $P' \,|\, H_L$ engage in the d_L-transition to the process $P' \,|\, H_{PQ}$.

  The process $C_{PQ}[Q]$ has to match the first step by a τ-transition to the process $Q \,|\, H_L$ since only this process has the distinguished action d_L enabled.

  Now, we take a closer look at the second step. We have to match a σ-transition. Therefore, Q has to perform a σ-transition to some process $Q' \in \mathcal{P}$ and the process H_L has to idle on σ, i.e. $Q \,|\, H_L \xrightarrow{\sigma} Q' \,|\, H_L$. According to Rule (tCom), the condition $\mathcal{I}\!\!\mathcal{I}_\sigma(Q) \cap \overline{\mathcal{I}\!\!\mathcal{I}_\sigma(H_L)} = \emptyset$ has to be satisfied. Because of the choice of L, this implies $\mathcal{I}\!\!\mathcal{I}_\sigma(Q) \subseteq \mathcal{I}\!\!\mathcal{I}_\sigma(P)$.

  Finally, the last step can only be matched by the transition $Q' \,|\, H_L \xrightarrow{d_L} Q' \,|\, H_{PQ}$. Thus, $C_{PQ}[P'] \equiv P' \,|\, H_{PQ} \sim_{\mathrm{n}} Q' \,|\, H_{PQ} \equiv C_{PQ}[Q']$.

  Since $\mathcal{S}(P') \subseteq \mathcal{S}(P)$ and $\mathcal{S}(Q') \subseteq \mathcal{S}(Q)$ it follows that $C_{P'Q'}[P'] \sim_{\mathrm{n}} C_{P'Q'}[Q']$, i.e. $P' \sim_{\mathrm{a}} Q'$.

Thus, $\sim_{\mathrm{a}}$ is a temporal strong bisimulation, i.e. $\sim_{\mathrm{a}} \subseteq \simeq$ according to Definition 7.5.2. Hence, $\sim_{\mathrm{n}}^{+} \subseteq \sim_{\mathrm{a}} \subseteq \simeq$, which – together with the inclusion $\simeq \subseteq \sim_{\mathrm{n}}^{+}$ obtained earlier – yields $\simeq = \sim_{\mathrm{n}}^{+}$, as desired. □

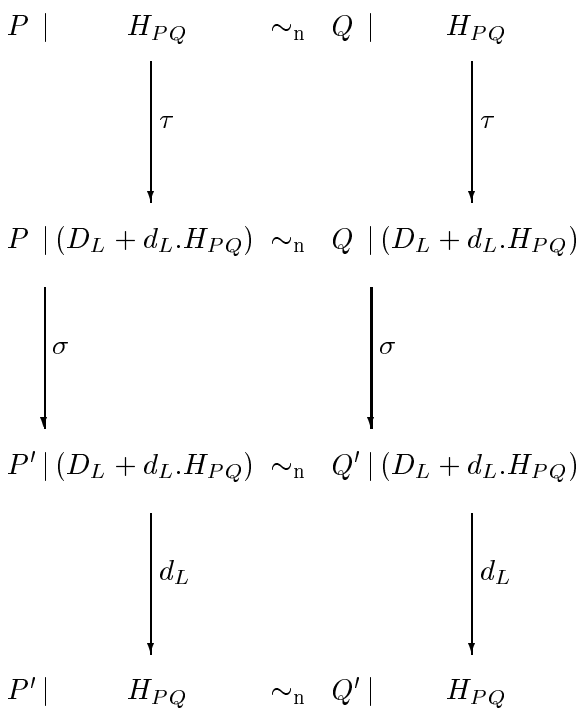

Figure 7.3: Largest congruence proof – illustration of Situation (2)

7.5.2 Axiomatic Characterization

In this section, we provide an axiomatization of $\simeq$ for *regular* processes, i.e. a class of finite-state processes that do not contain static operators inside recursion. Since axiomatization is a term-based technique we will use the letters $t, u, v, \ldots$ to range over CSA terms, possibly containing free variables. Moreover, we often call processes, i.e. closed and guarded terms, *process terms* in this section. In order to develop the axiomatization, it is convenient to add a new ignore operator $\downarrow \sigma$ to CSA, called *dynamic ignore*. The idea is, similar to axiomatizations for CCS [125], to eliminate static operators in process terms such that our completeness proof only has to deal with the dynamic operators and recursion. However, in contrast to PMC, the behavior of the ignore operator cannot be expressed by the existing dynamic operators and recursion since those cannot mimic clock co-scoping. To this end, we introduce a dynamic version of the ignore operator with the desired abilities.

Definition 7.5.5 (Dynamic Ignore)
The semantics of the dynamic ignore operator $\downarrow \sigma$ is defined by the following operational rules.

$$\mathsf{DIgn}\ \frac{t \xrightarrow{\alpha} t'}{t\downarrow\sigma \xrightarrow{\alpha} t'} \qquad \mathsf{tDIgn1}\ \frac{-}{t\downarrow\sigma \xrightarrow{\sigma} t\downarrow\sigma} \qquad \mathsf{tDIgn2}\ \frac{t \xrightarrow{\sigma'} t'}{t\downarrow\sigma \xrightarrow{\sigma'} t'\downarrow\sigma}\ \sigma \not\equiv \sigma'$$

Moreover, we extend the definition of $\mathcal{I}_\sigma(\cdot)$ *by* $\mathcal{I}_\sigma(t\downarrow\sigma') =_{df} \mathcal{I}_\sigma(t\uparrow\sigma')$.

Observe that Rules (tDIgn1) and (tDIgn2) for the dynamic ignore operator are identical to Rules (tIgn1) and (tIgn2) for the (static) ignore operator, respectively. The dynamic behavior of $\downarrow\sigma$ is apparent in Rule (DIgn) where the operator is dropped after an initial action transition. It is important to observe that $\simeq$ is compositional with respect to the dynamic ignore operator.

Proposition 7.5.6 *Let* $\sigma \in \mathcal{T}$ *and* t, u *terms such that* $t \simeq u$. *Then* $t\downarrow\sigma \simeq u\downarrow\sigma$.

The proof is similar to the corresponding one for the ignore operator and, therefore, omitted.

Definition 7.5.7 (Regular and Finite Process Terms)
A process term t *is called* regular *if it is built from nil, prefixing, summation, timeout, dynamic ignore, variables, and recursion. We say that* t *is* rs-free, *where* rs *abbreviates* recursion through static operators, *if every subterm* $\mu x.u$ *of* t *is regular. Finally, a process term* t *is* finite *if it does not contain any recursion operator.*

Table 7.4: Axiomatization of $\simeq$ (Part I)

(A1)	$t + u = u + t$	(B1)	$\lfloor\lfloor t\rfloor\sigma(u)\rfloor\sigma(v) = \lfloor t\rfloor\sigma(v)$	
(A2)	$t + (u + v) = (t + u) + v$	(B2)	$\lfloor\lfloor t\rfloor\sigma(u)\rfloor\sigma'(v) = \lfloor\lfloor t\rfloor\sigma'(v)\rfloor\sigma(u)$	$\sigma \not\equiv \sigma'$
(A3)	$t + t = t$	(B3)	$\lfloor t\rfloor\sigma(u) + \lfloor v\rfloor\sigma(w) = \lfloor t + v\rfloor\sigma(u + w)$	
(A4)	$t + \mathbf{0} = t$			
(D1)	$\mathbf{0}[f] = \mathbf{0}$	(C1)	$\mathbf{0} \setminus L = \mathbf{0}$	
(D2)	$(\alpha.t)[f] = f(\alpha).(t[f])$	(C2)	$(\alpha.t) \setminus L = \mathbf{0}$	$\alpha \in L \cup \overline{L}$
(D3)	$(t + u)[f] = t[f] + u[f]$	(C3)	$(\alpha.t) \setminus L = \alpha.(t \setminus L)$	$\alpha \notin L \cup \overline{L}$
(D4)	$(\lfloor t\rfloor\sigma(u))[f] = \lfloor t[f]\rfloor\sigma(u[f])$	(C4)	$(t + u) \setminus L = t \setminus L + u \setminus L$	
		(C5)	$(\lfloor t\rfloor\sigma(u)) \setminus L = \lfloor t \setminus L\rfloor\sigma(u \setminus L)$	

Note that the class of regular processes subsumes CSA processes in *standard concurrent form* [125]. Now, we turn to the axioms for temporal strong bisimulation. We write

- $\vdash_{rs} t = u$ if t can be rewritten to u by using the axioms in Tables 7.4, 7.5, and 7.6,

- $\vdash_r t = u$ if t can be rewritten to u by using all axioms except Axioms (D1)–(D4), Axioms (C1)–(C5), Axioms (I1)–(I8), and Axiom (E), and
- $\vdash_f t = u$ if t can be rewritten to u by using all axioms except Axioms (R0)–(R2).

Clearly, both $\vdash_f t = u$ and $\vdash_r t = u$ imply $\vdash_{rs} t = u$. Many axioms are identical to the ones presented in [5] for PMC. Axioms (L1)–(L9), and (I8) deal with the new dynamic ignore operator, where Axioms (L9) and (I8) capture the relationship between the static and the dynamic ignore operator. Moreover, the expansion axiom, Axiom (E), has been adapted to our algebra. The new semantic extension compared to PMC is reflected by Axioms (P1), (P2), (S1), and (S2). Equations (P1) and (P2) deal with the (local) pre-emptive power of τ, and Equations (S1) and (S2) make the implicit idling of clocks explicit.

Table 7.5: Axiomatization of $\simeq$ (Part II)

(I1)	$\mathbf{0}\uparrow\sigma = \mathbf{0}$	(I5)	$(t\uparrow\sigma)\uparrow\sigma = t\uparrow\sigma$
(I2)	$(t+u)\uparrow\sigma = t\uparrow\sigma + u\uparrow\sigma$	(I6)	$(t\uparrow\sigma)\uparrow\rho = (t\uparrow\rho)\uparrow\sigma$
(I3)	$(t \setminus L)\uparrow\sigma = (t\uparrow\sigma) \setminus L$	(I7)	$(\lfloor t\rfloor\rho(u))\uparrow\sigma = \lfloor t\uparrow\sigma\rfloor\rho(u\uparrow\sigma)\sigma((\lfloor t\rfloor\rho(u))\uparrow\sigma)$
(I4)	$(t[f])\uparrow\sigma = (t\uparrow\sigma)[f]$	(I8)	$(\alpha.t)\uparrow\sigma = (\alpha.(t\uparrow\sigma))\downarrow\sigma$
(L1)	$\mathbf{0}\downarrow\sigma = \mathbf{0}$	(L5)	$(t\downarrow\sigma)\downarrow\sigma = t\downarrow\sigma$
(L2)	$(t+u)\downarrow\sigma = t\downarrow\sigma + u\downarrow\sigma$	(L6)	$(t\downarrow\sigma)\downarrow\rho = (t\downarrow\rho)\downarrow\sigma$
(L3)	$(t \setminus L)\downarrow\sigma = (t\downarrow\sigma) \setminus L$	(L7)	$(\lfloor t\rfloor\rho(u))\downarrow\sigma = \lfloor t\downarrow\sigma\rfloor\rho(u\downarrow\sigma)\sigma((\lfloor t\rfloor\rho(u))\downarrow\sigma)$
(L4)	$(t[f])\downarrow\sigma = (t\downarrow\sigma)[f]$	(L8)	$(\alpha.t)\downarrow T + (\alpha.u)\downarrow T' = (\alpha.t + \alpha.u)\downarrow(T \cap T')$
		(L9)	$(t\uparrow\rho)\downarrow\sigma = (t\downarrow\sigma)\uparrow\rho$
(S1)	$\mathbf{0} = \lfloor\mathbf{0}\rfloor\sigma(\mathbf{0})$	(P1)	$(\tau.t)\downarrow T + u\downarrow\sigma = (\tau.t)\downarrow T + u \quad \sigma \notin T$
(S2)	$\alpha.t = \lfloor\alpha.t\rfloor\sigma(\alpha.t)$	(P2)	$\lfloor(\tau.t)\downarrow T + u\rfloor\sigma(v) = (\tau.t)\downarrow T + u \quad \sigma \notin T$

Axioms (L5) and (L6) allow us to introduce the notation $t\downarrow T$, where $T = \{\sigma_1, \ldots \sigma_n\}$ is a finite set of clocks, as a shorthand for $t\downarrow\sigma_1 \ldots\downarrow\sigma_n$. The same is true if we replace the dynamic ignore operator by the static one (cf. Axioms (I5) and (I6)). Thus, the simplifying notation in Axioms (L8), (P1), (P2), and (E) is justified. Observe that in Axiom (I7), and likewise in Axiom (L7), we may not simply push the ignore operator inside the operands of the timeout operator since, otherwise, the term on the left side can idle on σ but the term on the right side looses its timeout handler $u\uparrow\sigma$ when performing a σ-step. Thus, this idling is inserted explicitly in the right-side term. Axiom (L8) captures the fact that action α is not in the scope of clock σ whenever each occurrence of α is not in the scope of σ (cf. our definition of the $\mathcal{I}\!\!I_\sigma(\cdot)$–sets and the operational rules).

Table 7.6: Axiomatization of $\simeq$ (Part III)

(E)	Let $t \equiv \lfloor \sum_{i\in I}(\sum_{j\in J_i} \alpha_i.t_{ij})\downarrow T_i \rfloor \vec{\sigma}(\vec{v})\,, u \equiv \lfloor \sum_{i\in I}(\sum_{k\in K_i} \alpha_i.u_{ik})\downarrow U_i \rfloor \vec{\sigma}(\vec{w})$ and $\vec{\sigma} \equiv \sigma_1, \ldots, \sigma_n$ .Then $t \mid u = \lfloor r \rfloor \sigma_1(v_1 \mid w_1) \cdots \sigma_n(v_n \mid w_n)$ where $r \equiv \sum_{i\in I}((\sum_{j\in J_i} \alpha_i.(t_{ij} \mid u))\downarrow T_i + (\sum_{k\in K_i} \alpha_i.(t \mid u_{ik}))\downarrow U_i) + \sum_{i,i'\in I}\{\sum_{j\in J_i}\sum_{k\in K_i}(\tau.(t_{ij} \mid u_{i'k}))\downarrow (T_i \cup U_{i'}) \mid \alpha_i = \overline{\alpha}_{i'}\}$			
(R0)	$\mu x.t$	$=$	$\mu y.(t[y/x])$	y does not occur in t
(R1)	$\mu x.t$	$=$	$t[\mu x.t/x]$	
(R2)	$\vdash u = t[u/x]$	implies	$\vdash u = \mu x.t$	x guarded in t

The Expansion Axiom (E) in Table 7.6, which uses the symbol $\sum$ for the indexed version of $+$, shows how to eliminate the parallel composition operator. The timeout part of $t \mid u$ is defined pointwise for each clock. The summation part r splits up into two summands. The summand in the first line considers action transitions performed by one side alone, while the summand in the second line deals with the communication case. If we drop the dynamic ignore operators $\downarrow T_i, \downarrow U_i$, and $\downarrow T_i \cup U_{i'}$, the action part r has exactly the same structure as the Expansion Axiom in CCS [125]. The dynamic ignore operators are determined naturally by our clock-scoping semantics. Specifically, the dynamic ignore set $\downarrow T_i \cup U_{i'}$ leaves the internal action τ in the scope of a clock σ if and only if $\sigma \notin T_i$ and $\sigma \notin U_{i'}$, i.e. σ is connected to each of the communicating actions α_i and $\overline{\alpha}_{i'}$.

Theorem 7.5.8 (Soundness)
The presented axioms are sound, i.e. $\vdash_{rs} t = u$ *for terms* t *and* u *implies* $t \simeq u$.

Proof: According to the standard extension of the definition of temporal strong bisimulation to open terms it is sufficient to verify the soundness of the axioms for all closed instantiations. This is straightforward for most axioms by constructing a corresponding bisimulation. More precisely, if $s = t$ is an arbitrary axiom, except Axiom (R0), (R1), or (R2), then

$$\{\langle s[\theta], t[\theta]\rangle \mid \theta \text{ is a closed substitution}\} \cup \mathtt{Id}\,,$$

where Id is the identity relation on process terms, can be shown to be a temporal strong bisimulation by inspecting the initial action and clock transitions.

For Axiom (R0) the proof follows the lines of the corresponding one in [124]. The validity of Axiom (R1) is due to the unfolding rules, Rules (Rec) and (tRec), i.e. the process term $\mu x.t$ has exactly the same derivatives as the process term $t[\mu x.t/x]$. Finally, the soundness

of Axiom (R2) comes down to the statement that if for all closed substitutions θ the relation $u[\theta] \simeq t[u/x][\theta]$ holds, then $u[\theta'] \simeq (\mu x.t)[\theta']$ for all closed substitutions θ', provided that x is guarded in t. We may assume that x is not free in u since, otherwise, we can rename x by Axiom (R0). Hence, w.l.o.g. $t[u/x][\theta'] \equiv t[\theta'][u[\theta']/x]$ and $(\mu x.t)[\theta'] \equiv \mu x.(t[\theta'])$. Therefore, for proving the validity of Axiom (R2) it is sufficient to show that for all process terms p and $\mu x.t$ such that $p \simeq t[p/x]$ the equivalence $p \simeq \mu x.t$ holds. For establishing this, we consider the relation

$$\mathcal{R} =_{\mathrm{df}} \{\langle v[p/x], v[\mu x.t/x]\rangle \mid \text{in } v \text{ at most the variable } x \text{ is free}\}$$

and show that $\mathcal{R}$ is a temporal strong bisimulation up to $\simeq$ by induction on the height of derivations. The proof uses the assumption $p \simeq t[p/x]$ and the guardedness of x in t. $\square$

Table 7.7: Explicit clock sets

$\mathcal{C}(\mathbf{0}) =_{\mathrm{df}} \emptyset$	$\mathcal{C}(x) =_{\mathrm{df}} \emptyset$
$\mathcal{C}(\alpha.t) =_{\mathrm{df}} \mathcal{C}(t)$	$\mathcal{C}(\mu x.t) =_{\mathrm{df}} \mathcal{C}(t)$
$\mathcal{C}(t + u) =_{\mathrm{df}} \mathcal{C}(t) \cup \mathcal{C}(u)$	$\mathcal{C}(t \mid u) =_{\mathrm{df}} \mathcal{C}(t) \cup \mathcal{C}(u)$
$\mathcal{C}(t[f]) =_{\mathrm{df}} \mathcal{C}(t)$	$\mathcal{C}(t \setminus L) =_{\mathrm{df}} \mathcal{C}(t)$
$\mathcal{C}(t\uparrow\sigma) =_{\mathrm{df}} \mathcal{C}(t) \cup \{\sigma\}$	$\mathcal{C}(t\downarrow\sigma) =_{\mathrm{df}} \mathcal{C}(t) \cup \{\sigma\}$
$\mathcal{C}(\lfloor t \rfloor \sigma(u)) =_{\mathrm{df}} \mathcal{C}(t) \cup \mathcal{C}(u) \cup \{\sigma\}$	

In order to prove the completeness of our axiomatization with respect to temporal strong bisimulation we introduce a notion of *normal form* that is based on the following definition of *summation form*. Although our completeness result is only concerned with process terms, which are closed, free variables needs to be taken care of in summation forms to permit induction on the structure of terms. We also introduce the finite set $\mathcal{C}(t)$ of clocks explicitly occurring in terms t, which is inductively defined over the structure of t as shown in Table 7.7. In fact, our completeness proof only makes the clock behavior of these clocks which occur in the terms under consideration explicit and, thus, allows us to prove completeness even when the clock universe is infinite.

Definition 7.5.9 (Summation Form)
A term t with free variables in $\vec{x} \subseteq \mathcal{V}$ is called in $\langle \vec{\sigma}, \vec{x} \rangle$-summation form *if it is of the shape*

$$t \equiv t_{main} + t_{var} \equiv \lfloor \sum_{j=1}^{m} (\sum_{k=1}^{n_j} \alpha_j.t_{jk}) \downarrow T_j \rfloor \vec{\sigma}(\vec{t}) + \sum_{l=1}^{o} \lfloor x_{g(l)} \downarrow S_l \rfloor \vec{\sigma}_l(\vec{t_l}),$$

where (i) $m, n_j, o \in \mathbb{N}$ and $m \geq 1$, (ii) $\forall 1 \leq j \leq m.\ T_j \subseteq \mathcal{T}$ and $\alpha_j \in \mathcal{A}$, (iii) $\forall 1 \leq l \leq o.\ S_l \subseteq \mathcal{T}$ and $x_{g(l)} \in \vec{x}$, where g is a function on indices, (iv) $\vec{t}$ and t_{jk} are terms with free variables in $\vec{x}$ for all $1 \leq j \leq m$ and $1 \leq k \leq n_j$, and (v) for all $1 \leq l \leq o$ the components of $\vec{t_l}$ are either identical to $\mathbf{0}$ or to $x_{g(l)} \downarrow S_l$. Additionally, t must satisfy the following conditions:

1. $\alpha_1 \equiv \tau$,
2. $n_1 = 0$ *implies* $T_1 = \vec{\sigma}$,
3. $\forall 1 \leq j', j'' \leq m.\ j' \neq j''$ *implies* $\alpha_{j'} \not\equiv \alpha_{j''}$,
4. $\forall 1 \leq j \leq m.\ T_j \subseteq \vec{\sigma}$, *and*
5. $\forall 1 \leq l \leq o.\ \vec{\sigma}_l \subseteq \vec{\sigma}$ *and* $S_l \subseteq \vec{\sigma}$.

Note that the "variable"-summand t_{var} of a summation form t disappears if $\vec{x} = \epsilon$ according to Axiom (A4). Intuitively, whereas the first three conditions only have to do with the arrangement of summands in summation forms, the last two conditions state that explicit clocks appearing as arguments in the local summation operator or the timeout operators in the "variable"-summand t_{var} must be in $\vec{\sigma}$, i.e. their behavior has to be made explicit. In the following we say that an equation is in $\langle \vec{\sigma}, \vec{x} \rangle$-summation form if its right-hand side term is in $\langle \vec{\sigma}, \vec{x} \rangle$-summation form. We first show that every regular process term can be rewritten into summation form in the following sense.

Proposition 7.5.10 *For any regular term t with free unguarded variables in $\vec{x}$ and free guarded variables in $\vec{y}$ and for every $\vec{\sigma} \supseteq \mathcal{C}(t)$ there exist terms $\vec{t} = t_1 \ldots t_h$ provably satisfying h equations such that $\vdash_r t_i = u_i$ for some terms*

$$u_i \equiv \lfloor \sum_{j=1}^{m(i)} (\sum_{k=1}^{n_j(i)} \alpha_{ij}.t_{e(i,j,k)}) \downarrow T_{ij} \rfloor \vec{\sigma}(\vec{t}_{f(i)}) + \sum_{l=1}^{o(i)} \lfloor x_{g(i,l)} \downarrow S_{il} \rfloor \vec{\sigma}_{il}(\vec{t}_{il})$$

and $1 \leq i \leq h$, where e, f, g are functions on indices, $\vec{t}_{il}$ is a vector over $t_1, \ldots, t_h$, the term u_1 is in $\langle \vec{\sigma}, \vec{x} \rangle$-summation form, and u_j is in $\langle \vec{\sigma}, \vec{x}\vec{y} \rangle$-summation form for $1 < j \leq h$. Moreover, $\vdash_r t = t_1$.

Proof: Let t be a regular term with free unguarded variables in $\vec{x}$ and free guarded variables in $\vec{y}$. Moreover, let $\vec{\sigma}$ be a sequence of clocks such that $\vec{\sigma} \supseteq \mathcal{C}(t)$. The proof is done by induction on the structure of t.

- $t \equiv 0$:

  By repeated application of Axioms (S1) and (L1) we infer $\vdash_r \mathbf{0} = \lfloor \mathbf{0} \rfloor \vec{\sigma}(\vec{\mathbf{0}}) = \lfloor \mathbf{0}\downarrow\vec{\sigma} \rfloor \vec{\sigma}(\vec{\mathbf{0}})$, and by applying Axiom (A4) $\vdash_r \mathbf{0} = \mathbf{0} + \mathbf{0} = \lfloor \mathbf{0}\downarrow\vec{\sigma} \rfloor \vec{\sigma}(\vec{\mathbf{0}}) + \mathbf{0}$. Since the empty sum is identical to $\mathbf{0}$ this is an equation in $\langle \vec{\sigma}, \vec{x} \rangle$-summation form.

- $t \equiv x$:

  Especially, $x \in \vec{x}$. By the previous case we know that $\vdash_r \mathbf{0} = \lfloor \mathbf{0}\downarrow\vec{\sigma} \rfloor \vec{\sigma}(\vec{\mathbf{0}})$. Hence by Axiom (A4), by repeated application of Axiom (L1), and by notational convention, $\vdash_r x = x\downarrow\emptyset = \mathbf{0} + x\downarrow\emptyset = \lfloor \mathbf{0}\downarrow\vec{\sigma} \rfloor \vec{\sigma}(\vec{\mathbf{0}}) + x\downarrow\emptyset$ which is an equation in $\langle \vec{\sigma}, \vec{x} \rangle$-summation form.

- $t \equiv \alpha.u$:

  Here, all free variables in t are guarded, i.e. $\vec{x} = \epsilon$. However, the variables in $\vec{y}$ may occur unguarded in u. Since $\vec{\sigma} \supseteq \mathcal{C}(u) = \mathcal{C}(\alpha.u)$ we may apply the induction hypothesis for u and find terms $\vec{u} = u_1 \cdots u_h$ provably satisfying h equations where u_i is in $\langle \vec{\sigma}, \vec{y} \rangle$-summation form for $1 \leq i \leq h$. Moreover, $\vdash_r u = u_1$. By repeatedly applying Axiom (S2) we obtain $\vdash_r \alpha.u = \lfloor \alpha.u \rfloor \vec{\sigma}(\alpha.\vec{u})$, and by applying Axiom (A4) $\vdash_r \alpha.u = \alpha.u + \mathbf{0} = \lfloor \alpha.u \rfloor \vec{\sigma}(\alpha.\vec{u}) + \mathbf{0}$. Per convention, we have $\alpha.u \equiv (\alpha.u)\downarrow\emptyset$. Moreover, if $\alpha \not\equiv \tau$ we may apply Axioms (A4) and (L1) repeatedly in order to obtain $\vdash_r \alpha.u = \mathbf{0}\downarrow\vec{\sigma} + (\alpha.u)\downarrow\emptyset$. Hence, $\vdash_r \alpha.u = \lfloor (\alpha.u)\downarrow\emptyset \rfloor \vec{\sigma}(\alpha.\vec{u}) + \mathbf{0}$, if $\alpha \equiv \tau$, and $\vdash_r \alpha.u = \lfloor \mathbf{0}\downarrow\vec{\sigma} + (\alpha.u)\downarrow\emptyset \rfloor \vec{\sigma}(\alpha.\vec{u}) + \mathbf{0}$, otherwise, which are equations in $\langle \vec{\sigma}, \vec{y} \rangle$-summation form. Finally, we finish by adding the equation $\vdash_r \alpha.u_1 = \lfloor (\alpha.u_1)\downarrow\emptyset \rfloor \vec{\sigma}(\alpha.\vec{u}_1) + \mathbf{0}$, if $\alpha \equiv \tau$, and $\vdash_r \alpha.u_1 = \lfloor \mathbf{0}\downarrow\vec{\sigma} + (\alpha.u_1)\downarrow\emptyset \rfloor \vec{\sigma}(\alpha.\vec{u}_1) + \mathbf{0}$, otherwise, to the equations obtained by the induction hypothesis.

- $t \equiv u + v$:

  First, observe that $\vec{\sigma} \supseteq \mathcal{C}(u)$, $\vec{\sigma} \supseteq \mathcal{C}(v)$ since $\vec{\sigma} \supseteq \mathcal{C}(t) = \mathcal{C}(u) \cup \mathcal{C}(v)$, and u and v have free unguarded variables in $\vec{x}$ and free guarded variables in $\vec{x}\vec{y}$. By induction hypothesis there exist terms $\vec{u}$ provably satisfying equations where $\vdash_r u = u_1$, the term u_1 is in $\langle \vec{\sigma}, \vec{x} \rangle$-summation form and the other terms are in $\langle \vec{\sigma}, \vec{x}\vec{y} \rangle$-summation form. Further, there exist terms $\vec{v}$ provably satisfying equations where $\vdash_r v = v_1$, the term v_1 is in $\langle \vec{\sigma}, \vec{x} \rangle$-summation form and the other terms are in $\langle \vec{\sigma}, \vec{x}\vec{y} \rangle$-summation form. Adding together the corresponding equations for each u_i and $v_{i'}$ and applying Axiom (B3) to add the "main"-summands $u_{i,\mathrm{main}}$ and $v_{i',\mathrm{main}}$, we obtain the required summation form by rearranging the summands inside the new "main"-summand using Axioms (L2) and (L8) plus the commutativity and associativity axioms. Now, it is easy to see that the collection of equations for u

and v and the new equations $u_i + v_{i'}$ give a set of equations of the required form satisfying $\vdash_r u + v = u_1 + v_1$.

- $t \equiv \lfloor u \rfloor \sigma(v)$*:*

  As in the previous case we may apply the induction hypothesis since $\vec{\sigma} \supseteq \mathcal{C}(u)$ and $\vec{\sigma} \supseteq \mathcal{C}(v)$ according to the premise $\vec{\sigma} \supseteq \mathcal{C}(t)$ and the fact that $\mathcal{C}(t) = \mathcal{C}(u) \cup \mathcal{C}(v) \cup \{\sigma\}$. Observe that free unguarded variables in v occur guarded in t. Using the $\langle \vec{\sigma}, \vec{x} \rangle$-summation form for u_1 we get a $\langle \vec{\sigma}, \vec{x} \rangle$-summation form for $\lfloor u \rfloor \sigma(v)$ by manipulating $\lfloor u_1 \rfloor \sigma(v_1)$ as follows. By Axiom (A3) we have $\vdash_r v_1 = v_1 + \mathbf{0}$ such that we can apply Axiom (B3) to distribute $\sigma(v_1)$ to the "main"-summand $u_{1,\mathrm{main}}$ and $\sigma(\mathbf{0})$ to the "variable"-summand $u_{1,\mathrm{var}}$ of u_1, which yields again a $\langle \vec{\sigma}, \vec{x} \rangle$-summation form after repeatedly applying Axiom (A4) in the "variable"-summand t_{var}. Together with the other equations for u and v, this gives a set of equations of the required form.

- $t \equiv u{\downarrow}\sigma$*:*

  Here, we have $\vec{\sigma} \supseteq \mathcal{C}(u)$ since $\vec{\sigma} \supseteq \mathcal{C}(t)$ and $\mathcal{C}(t) = \mathcal{C}(u) \cup \{\sigma\}$. Especially, $\sigma \in \vec{\sigma}$. Moreover, u may contain free unguarded variables in $\vec{x}$ and free guarded variables in $\vec{y}$. By induction hypothesis there exist terms $\vec{u} = u_1 \cdots u_h$ provably satisfying h equations where $\vdash_r u = u_1$, the term u_1 is in $\langle \vec{\sigma}, \vec{x} \rangle$-summation form and the terms u_i, for $1 < i \leq h$, are in $\langle \vec{\sigma}, \vec{x}\vec{y} \rangle$-summation form. More precisely,

$$\begin{aligned} \vdash_r u_i &= u_{i,\mathrm{main}} + u_{i,\mathrm{var}} \\ &= \lfloor \textstyle\sum_{j=1}^{m(i)} (\sum_{k=1}^{n_j(i)} \alpha_{ij}.u_{e(i,j,k)}){\downarrow}T_{ij} \rfloor \vec{\sigma}(\vec{u}_{f(i)}) + \\ &\quad \textstyle\sum_{l=1}^{o(i)} \lfloor x_{g(i,l)}{\downarrow}S_{il} \rfloor \vec{\sigma}_{il}(\vec{u}_{il}) \end{aligned}$$

  for some index functions e, f, and g. Let $t_i \stackrel{\mathrm{def}}{=} u_i{\downarrow}\sigma$. Then

$$\begin{aligned} \vdash_r \; & t_i \\ = \; & (\lfloor \textstyle\sum_{j=1}^{m(i)} (\sum_{k=1}^{n_j(i)} \alpha_{ij}.u_{e(i,j,k)}){\downarrow}T_{ij} \rfloor \vec{\sigma}(\vec{u}_{f(i)}) + \\ & \textstyle\sum_{l=1}^{o(i)} \lfloor x_{g(i,l)}{\downarrow}S_{il} \rfloor \vec{\sigma}_{il}(\vec{u}_{il})){\downarrow}\sigma \end{aligned}$$

$$= \quad \lfloor \textstyle\sum_{j=1}^{m(i)} (\sum_{k=1}^{n_j(i)} \alpha_{ij}.u_{e(i,j,k)}) \rfloor (T_{ij} \cup \{\sigma\}) \rfloor \vec{\sigma}(\vec{u}_{f(i)} \downarrow \sigma) \sigma(u_{i,\mathrm{main}} \downarrow \sigma) +$$

$$\textstyle\sum_{l=1}^{o(i)} \lfloor x_{g(i,l)} \rfloor (S_{il} \cup \{\sigma\}) \rfloor \vec{\sigma}_{il}(\vec{u}_{il} \downarrow \sigma) \sigma(u_{i,\mathrm{var}} \downarrow \sigma)$$

by repeated application of Axiom (L2) to push $\downarrow \sigma$ inside the sums and application of Axiom (L7) – as well as Axioms (B1) and (B2) – to push it successively inside the timeouts. The expression $\vec{u}_{f(i)} \downarrow \sigma$ denotes the pointwise application of $\downarrow \sigma$.

Moreover, we associate with $u_{i,\mathrm{main}} \downarrow \sigma$ the equation

$$\vdash_r \quad u_{i,\mathrm{main}} \downarrow \sigma$$

$$= \quad \lfloor \textstyle\sum_{j=1}^{m(i)} (\sum_{k=1}^{n_j(i)} \alpha_{ij}.u_{e(i,j,k)}) \rfloor (T_{ij} \cup \{\sigma\}) \rfloor \vec{\sigma}(\vec{u}_{f(i)} \downarrow \sigma) \sigma(u_{i,\mathrm{main}} \downarrow \sigma) + \mathbf{0}$$

which is in $\langle \vec{\sigma}, \vec{x}\vec{y} \rangle$-summation form. Finally, we insert for $u_{i,\mathrm{var}} \downarrow \sigma$ the equation

$$\vdash_r u_{i,\mathrm{var}} \downarrow \sigma = \lfloor \mathbf{0} \downarrow \vec{\sigma} \rfloor \vec{\sigma}(\vec{\mathbf{0}}) + \sum_{l=1}^{o(i)} \lfloor x_{g(i,l)} \rfloor (S_{il} \cup \{\sigma\}) \rfloor \vec{\sigma}_{il}(\vec{u}_{il} \downarrow \sigma) \sigma(u_{i,\mathrm{var}} \downarrow \sigma)$$

which is also of the required form after rewriting every component of $\vec{u}_{il} \downarrow \sigma$ identical to $\mathbf{0} \downarrow \sigma$ to $\mathbf{0}$ by Axiom (L1).

- $t \equiv \mu z.u$:

  This is the most challenging case. Assume $z \notin \vec{x}\vec{y}$; otherwise, use Axiom (R0) to rename z. Since bound variables are guarded, u has free unguarded variables in $\vec{x}$ and free guarded variables in $\vec{y}z$. Since $\vec{\sigma} \supseteq \mathcal{C}(t) = \mathcal{C}(u)$ we may apply the induction hypothesis to obtain h equations $\vec{u} = u_1 \cdots u_h$ such that u_1 is in $\langle \vec{\sigma}, \vec{x} \rangle$-summation form, i.e.

$$\vdash_r u_1 = \lfloor \sum_{j=1}^{m(1)} (\sum_{k=1}^{n_j(1)} \alpha_{1j}.u_{e(1,j,k)}) \downarrow T_{1j} \rfloor \vec{\sigma}(\vec{u}_{f(1)}) + \sum_{l=1}^{o(1)} \lfloor x_{g(1,l)} \downarrow S_{1l} \rfloor \vec{\sigma}_{1l}(\vec{u}_{1l})$$

  where $x_{g(1,l)} \not\equiv z$ for $1 \leq l \leq o(1)$, and u_i, where $1 < i \leq h$, is in $\langle \vec{\sigma}, \vec{x}\vec{y}z \rangle$-summation form, i.e.

$$\vdash_r u_i = \quad \lfloor \textstyle\sum_{j=1}^{m(i)} (\sum_{k=1}^{n_j(i)} \alpha_{ij}.u_{e(i,j,k)}) \downarrow T_{ij} \rfloor \vec{\sigma}(\vec{u}_{f(i)}) +$$

$$(\textstyle\sum_{l=1}^{o(i)} \lfloor x_{g(i,l)} \downarrow S_{il} \rfloor \vec{\sigma}_{il}(\vec{u}_{il})) + \sum_{l=1}^{o'(i)} \lfloor z \downarrow S'_{il} \rfloor \vec{\sigma}'_{il}(\vec{u}'_{il})$$

where all $x_{g(i,l)} \neq z$. Also, $\vdash_r u = u_1$. Hence, $\vdash_r t = \mu z.u = \mu z.u_1$. For $1 < i \leq h$ take $t_i \stackrel{\text{def}}{=} u_i[t/z]$. Consequently,

$$\begin{aligned}
\vdash_r t_i = \; & \lfloor \textstyle\sum_{j=1}^{m(i)} (\sum_{k=1}^{n_j(i)} \alpha_{ij}.u_{e(i,j,k)}[t/z]) \downarrow T_{ij} \rfloor \vec{\sigma}(\vec{u}_{f(i)}[t/z]) + \\
& (\textstyle\sum_{l=1}^{o(i)} \lfloor (x_{g(i,l)} \downarrow S_{il})[t/z] \rfloor \vec{\sigma}_{il}(\vec{u}_{il}[t/z])) + \sum_{l=1}^{o'(i)} \lfloor t \downarrow S'_{il} \rfloor \vec{\sigma}'_{il}(\vec{u}'_{il}[t/z]) \\
= \; & \lfloor \textstyle\sum_{j=1}^{m(i)} (\sum_{k=1}^{n_j(i)} \alpha_{ij}.t_{e(i,j,k)}) \downarrow T_{ij} \rfloor \vec{\sigma}(\vec{t}_{f(i)}) + \\
& \textstyle\sum_{l=1}^{o(i)} \lfloor x_{g(i,l)} \downarrow S_{il} \rfloor \vec{\sigma}_{il}(\vec{u}_{il}) + \sum_{l=1}^{o'(i)} \lfloor t \downarrow S'_{il} \rfloor \vec{\sigma}'_{il}(\vec{t}'_{il})
\end{aligned}$$

where

$$\vdash_r t = \lfloor \sum_{j=1}^{m(1)} (\sum_{k=1}^{n_j(1)} \alpha_{1j}.t_{e(1,j,k)}) \downarrow T_{1j} \rfloor \vec{\sigma}(\vec{t}_{f(1)}) + \sum_{l=1}^{o(1)} \lfloor x_{g(1,l)} \downarrow S_{il} \rfloor \vec{\sigma}_{1l}(\vec{u}_{1l}) \;\; .$$

Moreover, every component of $\vec{t}'_{il}$ has the form $t \downarrow S'_{il}$, if the corresponding component of $\vec{u}'_{il}$ is identical to $z \downarrow S'_{il}$, and $\mathbf{0}$, otherwise.

Now, we are in a situation where the outer recursion on the variable z is eliminated. In the following, we consider t as an identifier for the equation associated with t; so we do not need to decompose t again in the remaining normalizations. Hence, we can proceed as in the cases for timeout, dynamic ignore, summation, and recursion on other variables than z in order to obtain the required summation forms for all t_i. Note that this may involve introducing new equations. However, this iterative process terminates eventually since terms contain a *finite* number of variables. Finally, $\vdash_r t = t_1$ holds since $\vdash_r t = \mu z.u_1 = u_1[\mu z.u_1/z] = u_1[t/z] = t_1$ by Axiom (R1). $\square$

Based on the notion of summation form we can define *normal forms* for process terms.

Definition 7.5.11 (Normal Form)
A process term $t \equiv \lfloor \sum_{j=1}^{m} (\sum_{k=1}^{n_j} \alpha_j.t_{jk}) \downarrow T_j \rfloor \vec{\rho}(\vec{t})$ *is in* $\vec{\sigma}$-normal form *if it is in* $\langle \vec{\rho}, \epsilon \rangle$-*summation form and if the following conditions hold:*

1. $\forall 1 \leq i, i' \leq h.\ i \neq i'$ *implies* $\rho_i \not\equiv \rho_{i'}$ *where* $\vec{\rho} = \rho_1 \cdots \rho_h$,
2. $\vec{\rho} = \vec{\sigma} \cap T_1$, *and*
3. $\forall 1 \leq j \leq m.\ T_j \subseteq T_1$.

Rewriting a process t into a normal form equation system makes explicit the semantics of t and of all its reachable states, in such a way that there is a precise correspondence between the syntactic constructs and the semantic behavior in terms of action transitions, clock transitions, and clock scoping information. The action transitions are explicit by action prefixes, the clock transitions by timeouts, and the clock scoping by the dynamic ignore operators. In particular, the normal form completely takes account of the pre-emption effect of τ-actions (cf. Conditions (2) and (3) of Definition 7.5.11). However, not all clock transitions need to be made explicit via timeouts, only those which are already explicit in term t.

Proposition 7.5.12 *For any regular process term t and for every $\vec{\sigma} \supseteq \mathcal{C}(t)$ there exist terms $\vec{t} = t_1 \ldots t_h$ provably satisfying h equations such that $\vdash_r t_i = u_i$ for some terms*

$$u_i \equiv \lfloor \sum_{j=1}^{m(i)} (\sum_{k=1}^{n_j(i)} \alpha_{ij}.t_{e(i,j,k)}) \downarrow T_{ij} \rfloor \vec{\rho}(\vec{t}_{f(i)})$$

in $\vec{\sigma}$-normal form, where $1 \leq i \leq h$. Moreover, $\vdash_r t = t_1$.

Proof: Let t be a regular process term and $\vec{\sigma} \supseteq \mathcal{C}(t)$. By Proposition 7.5.10 there exist terms $\vec{t} = t_1 \ldots t_h$ provably satisfying h equations such that $\vdash_r t_i = u_i$ for some terms u_i in $\langle \vec{\sigma}, \epsilon \rangle$-summation form, where $1 \leq i \leq h$. Additionally, $\vdash_r t = t_1$. Consider

$$u_i \equiv \lfloor \sum_{j=1}^{m(i)} (\sum_{k=1}^{n_j(i)} \alpha_{ij}.t_{e(i,j,k)}) \downarrow T_{ij} \rfloor \vec{\sigma}(\vec{t}_{f(i)}) \ .$$

Then rewrite u_i along the following steps.

1. By applying Axioms (B1) and (B2) we can remove duplicates in $\vec{\sigma}$ together with their corresponding timeout handlers in $\vec{t}_{f(i)}$ in order to satisfy Condition (1) of Definition 7.5.11.

2. By Condition (4) of Definition 7.5.9 we know that $T_{i1} \subseteq \vec{\sigma}$. For every $\sigma \in \vec{\sigma} \setminus T_{i1}$ apply the following steps: (i) Rearrange the sequence $\vec{\sigma}$ and the timeout handlers $\vec{t}_{f(i)}$ by repeated application of Axiom (B2) such that σ is placed as first component. (ii) Push $\downarrow T_{i1}$ over the first inner summand by repeatedly applying Axiom (L2). (iii) Use Axiom (P2) in order to get rid of the timeout construct involving σ. Finally, apply Step (ii) in the reverse direction. Hence, Condition (2) of Definition 7.5.11 is established.

3. For each T_{ij}, $1 < j \leq m(i)$, and for each $\sigma \in T_{ij} \setminus T_{i1}$ proceed as follows: (i) Push $\downarrow T_{i1}$ over the first inner summand by repeated application of Axiom (L2). (ii) Use Axiom (P1) for removing clock σ in T_{ij}. (iii) Undo Step (i) in reverse manner. Consequently, Condition (3) of Definition 7.5.11 is satisfied.

Following the above steps, we obtain for each u_i, $1 \leq i \leq h$, a term u_i' in $\vec{\sigma}$-normal form such that $\vdash_r u_i = u_i'$, which completes the proof since $\vdash_r t_i = u_i = u_i'$. □

The completeness proof adapts Milner's technique [124] in characterizing recursive processes uniquely by systems of equations in normal form which is founded on the following proposition.

Proposition 7.5.13 (Unique Solutions of Guarded Equation Systems)
Let $\vec{x} = x_1 \cdots x_h$ be variables and $\vec{t} = t_1 \cdots t_h$ regular process terms with free variables in $\vec{x}$ in which each x_i, $1 \leq i \leq h$, is guarded. Then there exist regular process terms $\vec{u} = u_1 \cdots u_h$ such that $\vdash_r u_i = t_i[\vec{u}/\vec{x}]$ for $1 \leq i \leq h$. Moreover, if the same holds for h process terms $\vec{v} = v_1 \cdots v_h$, i.e. $\vdash_r v_i = t_i[\vec{v}/\vec{x}]$, then $\vec{u}$ and $\vec{v}$ are element-wise provably equal, i.e. $\vdash_r u_i = v_i$ for $1 \leq i \leq h$.

Proof: The proof is similar to the corresponding one in [124] and essentially extends Rule (R2) to hold for equations systems. Its correctness is only based on the validity of Rule (R2) and not influenced by the fact that we consider a "richer" language than the one presented in [124]. □

The above results are sufficient for proving several completeness results with respect to different classes of process terms.

Theorem 7.5.14 (Completeness for Regular Process Terms)
For regular process terms t and u satisfying $t \simeq u$ we have $\vdash_r t = u$.

Proof: Let t and u be regular process terms such that $t \simeq u$ and $\vec{\rho} =_{\mathrm{df}} \mathcal{C}(t) \cup \mathcal{C}(u)$. According to Proposition 7.5.12 there exist terms $\vec{t}$ and $\vec{u}$ provably satisfying equations in $\vec{\rho}$-normal form

$$\vdash_r t_i = \lfloor \sum_{j=1}^{m(i)} (\sum_{k=1}^{n_j(i)} \alpha_{ij}.t_{e(i,j,k)}) \downarrow T_{ij} \rfloor \vec{\sigma}_i(\vec{t}_{f(i)})$$

$$\vdash_r u_{i'} = \lfloor \sum_{j'=1}^{m'(i')} (\sum_{k'=1}^{n'_{j'}(i')} \alpha'_{i'j'}.u_{e'(i',j',k')}) \downarrow T'_{i'j'} \rfloor \vec{\sigma}'_{i'}(\vec{u}_{f'(i')})$$

such that $\vdash_r t = t_1$ and $\vdash_r u = u_1$.

In the following we construct a guarded equation system for which both, t and u, provably are solutions. Consider the set $I =_{\mathrm{df}} \{\langle i, i' \rangle \mid t_i \simeq u_{i'}\}$. It is easy to see that $\langle 1, 1 \rangle \in I$ since $t_1 \simeq t \simeq u \simeq u_1$ by premise and Theorem 7.5.8. Additionally, whenever $\langle i, i' \rangle \in I$ matching action derivatives yield new pairs in I. We formalize this by a set of relations

$$J_{ii'} =_{\mathrm{df}} \{\langle \langle j, k \rangle, \langle j', k' \rangle \rangle \mid \alpha_{ij} \equiv \alpha'_{i'j'} \text{ and } \langle e(i,j,k), e'(i',j',k') \rangle \in I\}$$

for all $\langle i, i' \rangle \in I$. Since $t_i \simeq u_{i'}$, the relations $J_{ii'}$ must be total and surjective. With the help of these notions we are able to prove that $T_{ij} = T'_{i'j'}$ for $\langle\langle j, k\rangle, \langle j', k'\rangle\rangle \in J_{ii'}$ and $\vec{\sigma}_i = \vec{\sigma}'_{i'}$ (up to reordering) whenever $\langle i, i' \rangle \in I$.

- Let $\langle i, i' \rangle \in I$, i.e. $t_i \simeq u_{i'}$, and $\langle\langle j, k\rangle, \langle j', k'\rangle\rangle \in J_{ii'}$. Assume $T_{ij} \neq T'_{i'j'}$; w.l.o.g. there exists some $\sigma \in T_{ij} \backslash T'_{i'j'}$. By Condition (4) of Definition 7.5.9 we have $\sigma \in \vec{\sigma}_i$, and due to Condition (2) of Definition 7.5.11 and the operational rules we have $t_i \xrightarrow{\sigma} t_{f(i)(l)}$. Hence, $\mathcal{I}_\sigma(t_i) = \mathcal{I}_\sigma(u_{i'})$ by the definition of $\simeq$. However, because of $\sigma \in T_{ij} \setminus T'_{i'j'}$, Condition (3) of Definition 7.5.9, and $\langle\langle j, k\rangle, \langle j', k'\rangle\rangle \in J_{ii'}$ we may conclude $\alpha_{ij} \equiv \alpha'_{i'j'} \in \mathcal{I}_\sigma(u_{i'}) \setminus \mathcal{I}_\sigma(t_i)$. The latter is a contradiction to $\mathcal{I}_\sigma(t_i) = \mathcal{I}_\sigma(u_{i'})$. Hence, $T_{ij} = T'_{i'j'}$ holds.
- By Conditions (1) and (3) of Definition 7.5.9 and by the definition of $\simeq$ we have for $\langle i, i' \rangle \in I$ that $\langle\langle j, k\rangle, \langle j', k'\rangle\rangle \in J_{ii'}$ implies $j = 1$ if and only if $j' = 1$. Hence, $T_{i1} = T'_{i'1}$ by the above result and, thus, $\vec{\sigma}_i = \vec{\rho} \cap T_{i1} = \vec{\rho} \cap T'_{i'1} = \vec{\sigma}'_{i'}$ by Condition (2) of Definition 7.5.11 up to reordering (cf. Axiom (B2) and Condition (1) of Definition 7.5.11).

Again, let $\langle i, i' \rangle \in I$ and let $\sigma \equiv \vec{\sigma}_{i(l)} \equiv \vec{\sigma}'_{i'(l)}$ for some component index l. By Condition (1) of Definition 7.5.11 we conclude $t_i \xrightarrow{\sigma} t_{f(i)(l)}$ and $u_{i'} \xrightarrow{\sigma} u_{f'(i')(l)}$. Since $t_i \simeq u_{i'}$ we have $t_{f(i)(l)} \simeq u_{f'(i')(l)}$, i.e. $\langle f(i)(l), f'(i')(l)\rangle \in I$ according to our operational rules and the definition of $\simeq$. Now, we construct the above-mentioned equation system. Assume for every $\langle i, i' \rangle \in I$ a variable $x_{i,i'}$. Define for each $\langle i, i' \rangle \in I$ an equation $x_{i,i'} = v_{i,i'}$ such that

$$v_{i,i'} \stackrel{\text{def}}{=} \lfloor \sum_{\langle\langle j,k\rangle,\langle j',k'\rangle\rangle \in J_{ii'}} ((\alpha_{ij}.x_{e(i,j,k),e'(i',j',k')})\downarrow T_{ij})\rfloor \vec{\sigma}_i(\vec{x}_{f(i),f'(i')})$$

where $\vec{x}_{f(i),f'(i')} = x_{f(i)(1),f'(i')(1)} \cdots x_{f(i)(l),f'(i')(l)}$ and l is the length of the sequence $\vec{\sigma}_i = \vec{\sigma}'_{i'}$. Moreover, let Θ be the substitution that takes $x_{i,i'}$ to t_i, and Θ' the one that takes $x_{i,i'}$ to $u_{i'}$. In the following we show that $\vdash_r t_i = v_{i,i'}[\Theta]$ for all $\langle i, i' \rangle \in I$.

$$\begin{aligned}
& \vdash_r && v_{i,i'}[\Theta] \\
& = && \lfloor \textstyle\sum_{\langle\langle j,k\rangle,\langle j',k'\rangle\rangle \in J_{ii'}} ((\alpha_{ij}.t_{e(i,j,k)})\downarrow T_{ij})\rfloor \vec{\sigma}_i(\vec{t}_{f(i),f'(i')}) \\
\text{(Axiom (A3), totality of } J_{ii'}) & = && \lfloor \textstyle\sum_{j=1}^{m(i)} \sum_{k=1}^{n_j(i)} ((\alpha_{ij}.t_{e(i,j,k)})\downarrow T_{ij})\rfloor \vec{\sigma}_i(\vec{t}_{f(i),f'(i')}) \\
\text{(Axiom (L2), repeatedly)} & = && \lfloor \textstyle\sum_{j=1}^{m(i)} (\sum_{k=1}^{n_j(i)} (\alpha_{ij}.t_{e(i,j,k)})\downarrow T_{ij})\rfloor \vec{\sigma}_i(\vec{t}_{f(i),f'(i')}) \\
& = && t_i
\end{aligned}$$

Similarly, $\vdash_r u_{i'} = v_{i,i'}[\Theta']$ since $J_{ii'}$ is surjective. Note that each $x_{i,i'}$ is guarded in the constructed equation system. Hence, $\vdash_r t_i = u_{i'}$ for each $\langle i, i' \rangle \in I$ according to Proposition 7.5.13. Especially, $\vdash_r t_1 = u_1$ since $\langle 1, 1\rangle \in I$. Because of $\vdash_r t = t_1$ and $\vdash_r u = u_1$ we obtain $\vdash_r t = u$, as desired. □

Note that no complete proof system for all process terms in CSA can exist since the set of valid equivalences is not recursively enumerable whereas any finitary proof system can only yield a recursively enumerable set of provable equalities. However, the completeness result can be extended to the class of rs-free process terms by eliminating the static operators, using the Expansion Axiom (E) to get rid of parallel composition, eliminating restriction by Axioms (C1)–(C5), (L3), and (I3), and relabeling by Axioms (D1)–(D4), (L4), and (I4).

Proposition 7.5.15 *If p is a rs-free process term then there exists a regular process term q such that $\vdash_{rs} p = q$.*

Proof: The proof is done by induction on the structure of p. The base case for $p \equiv \mathbf{0}$ and the induction step for the dynamic operators as well as for recursion are straightforward according to the definition of rs-free process terms. Thus, we concentrate on the cases of the induction step concerned with the static operators.

1. $p \equiv u{\uparrow}\sigma$*:*

   By induction hypothesis we may assume that $\vdash_{rs} u = q$ where q is a regular process term. According to Proposition 7.5.10 there exist process terms $\vec{q} = q_1 \cdots q_h$ satisfying h equations in $\langle \vec{\sigma}, \epsilon \rangle$-summation form for all $\vec{\sigma} \supseteq \mathcal{C}(u)$

   $$\vdash_{rs} q_i = \lfloor \sum_{j=1}^{m(i)} (\sum_{k=1}^{n_j(i)} \alpha_{ij}.q_{e(i,j,k)}) {\downarrow} T_{ij} \rfloor \vec{\sigma}(\vec{q}_{f(i)})$$

   such that $\vdash_r q = q_1$. For $p_i \stackrel{\text{def}}{=} q_i \uparrow \sigma$ we prove that these p_i satisfy h equations in $\langle \vec{\sigma}\sigma, \epsilon \rangle$-summation form. To this end, consider the following:

   $$\begin{aligned} \vdash_{rs} \quad & p_i \\ = \quad & (\lfloor \textstyle\sum_{j=1}^{m(i)} (\sum_{k=1}^{n_j(i)} \alpha_{ij}.q_{e(i,j,k)}) {\downarrow} T_{ij} \rfloor \vec{\sigma}(\vec{q}_{f(i)})) {\uparrow} \sigma \\ = \quad & \lfloor \textstyle\sum_{j=1}^{m(i)} ((\sum_{k=1}^{n_j(i)} \alpha_{ij}.q_{e(i,j,k)}) {\downarrow} T_{ij}) {\uparrow} \sigma \rfloor \vec{\sigma}(\vec{q}_{f(i)} {\uparrow} \sigma) \sigma(p_i) \end{aligned}$$

   by application of Axiom (I7) to push $\uparrow\sigma$ inside the timeout, and repeated application of Axioms (I2), (B1), and (B2) to push it inside the outer sum. The expression $\vec{q}_{f(i)} \uparrow \sigma$ stands for the pointwise application of $\uparrow\sigma$.

$$= \quad \lfloor \textstyle\sum_{j=1}^{m(i)} (\sum_{k=1}^{n_j(i)} \alpha_{ij}.(q_{e(i,j,k)} \uparrow \sigma) \downarrow (T_{ij} \cup \{\sigma\}) \rfloor \vec{\sigma}(\vec{q}_{f(i)} \uparrow \sigma) \sigma(p_i)$$

by repeated application of Axioms (L9), (I2), and (I8) to push $\uparrow \sigma$ over the dynamic ignore operators, inside the inner sum, and inside prefixing.

$$= \quad \lfloor \textstyle\sum_{j=1}^{m(i)} (\sum_{k=1}^{n_j(i)} \alpha_{ij}.p_{e(i,j,k)}) \downarrow (T_{ij} \cup \{\sigma\}) \rfloor \vec{\sigma}(\vec{p}_{f(i)}) \sigma(p_i)$$

where the last term is in $\langle \vec{\sigma}\sigma, \epsilon \rangle$-summation form, as desired.

2. $p \equiv u[f]$:

By induction hypothesis we know $\vdash_{rs} u = q$ for some regular process term q. According to Proposition 7.5.10 there exist process terms $\vec{q} = q_1 \cdots q_h$ satisfying h equations in $\langle \vec{\sigma}, \epsilon \rangle$-summation form for all $\vec{\sigma} \supseteq \mathcal{C}(u) = \mathcal{C}(p)$

$$\vdash_{rs} q_i = \lfloor \sum_{j=1}^{m(i)} (\sum_{k=1}^{n_j(i)} \alpha_{ij}.q_{e(i,j,k)}) \downarrow T_{ij} \rfloor \vec{\sigma}(\vec{q}_{f(i)})$$

such that $\vdash_r q = q_1$. Now, we define $p_i \stackrel{\text{def}}{=} q_i[f]$ and prove that these p_i satisfy h equations in $\langle \vec{\sigma}, \epsilon \rangle$-summation form. Hence,

$$\vdash_{rs} \quad p_i$$

$$= \quad (\lfloor \textstyle\sum_{j=1}^{m(i)} (\sum_{k=1}^{n_j(i)} \alpha_{ij}.q_{e(i,j,k)}) \downarrow T_{ij} \rfloor \vec{\sigma}(\vec{q}_{f(i)}))[f]$$

$$= \quad \lfloor \textstyle\sum_{j=1}^{m(i)} ((\sum_{k=1}^{n_j(i)} \alpha_{ij}.q_{e(i,j,k)}) \downarrow T_{ij})[f] \rfloor \vec{\sigma}(\vec{q}_{f(i)}[f])$$

by application of Axiom (D4) to push $[f]$ inside the timeout, and repeated application of Axiom (D3) to push it inside the outer sum. The expression $\vec{q}_{f(i)}[f]$ represents the pointwise application of $[f]$.

$$= \quad \lfloor \textstyle\sum_{j=1}^{m(i)} (\sum_{k=1}^{n_j(i)} f(\alpha_{ij}).(q_{e(i,j,k)}[f]) \downarrow T_{ij} \rfloor \vec{\sigma}(\vec{q}_{f(i)}[f])$$

by repeated application of Axioms (L4), (D3), and (D2) to push $[f]$ over the dynamic ignore operators, inside the inner sum, and inside prefixing. However, if $n_j(1) = 0$ then Axiom (D1) needs to be applied instead of Axiom (D2).

$$= \quad \lfloor \textstyle\sum_{j=1}^{m(i)} (\sum_{k=1}^{n_j(i)} f(\alpha_{ij}).p_{e(i,j,k)}) \downarrow T_{ij} \rfloor \vec{\sigma}(\vec{p}_{f(i)})$$

which is in the required summation form up to rearranging the summands inside the timeout construct in order to achieve Conditions (1)–(3) of Definition 7.5.9. The rearrangement can easily be done by using Axioms (L2), (A1), and (A2).

3. $p \equiv u \setminus L$:

   This case can be proved more easily than the previous one by using Axioms (C1)–(C5) and (L3) instead of Axioms (D1)–(D4) and (L4).

4. $p \equiv u|v$:

   By induction hypothesis we know of the existence of regular process terms q and r such that $\vdash_{rs} p = q|r$. According to Proposition 7.5.10 there exist process terms $\vec{q}$ and $\vec{r}$ satisfying equations in $\langle \vec{\sigma}, \epsilon \rangle$-summation form for all $\vec{\sigma} \supseteq \mathcal{C}(u|v) = \mathcal{C}(u) \cup \mathcal{C}(v)$. Let us define $p_{ii'} \stackrel{\text{def}}{=} q_i|r_{i'}$. Now, one can apply Axioms (L2), (A1), and (A2) multiple times for rearranging the summands and, subsequently, use Axiom (E) for obtaining equations in $\langle \vec{\sigma}, \epsilon \rangle$-summation form.

This completes the induction step. The desired result finally follows from the existence part of Proposition 7.5.13. □

As a consequence of this proposition, the completeness result for the class of regular process terms can be lifted to the class of rs-free process terms.

Theorem 7.5.16 (Completeness for rs-free Processes)
For rs-free process terms t and u satisfying $t \simeq u$ we have $\vdash_{rs} t = u$.

The statement is an immediate consequence of the proof of Theorem 7.5.14 and the statement of Proposition 7.5.15. When proving completeness for *finite* process terms we do not need Axioms (R0)–(R2) which deal with recursion only.

Theorem 7.5.17 (Completeness for Finite Processes)
For finite process terms t and u satisfying $t \simeq u$ we have $\vdash_f t = u$.

Proof: The theorem's validity follows by inspection of the proof of Theorem 7.5.14 since (i) the proof of Proposition 7.5.10 requires Axioms (R0)–(R2) only in the case of recursion, (ii) Proposition 7.5.12 just needs the results of Step (i) but not Axioms (R0)–(R2), (iii) equation systems which arise from finite process terms trivially possess unique solutions, and (iv) the static operators can be taken care of in a separate step, analogously to the one presented in Proposition 7.5.15, which is founded on Step (iii). □

Finally, we note that the traditional proof technique for finite process terms does not work as in the case of CCS [125]. The reason is that, even if recursion operators are not contained in finite process terms, recursion still occurs *implicitly* due to clock idling.

7.5.3 Operational Characterization

In Section 7.5.1 we have introduced $\simeq$ as a stronger form of bisimulation than the standard one on naive transition systems, i.e. those generated by our operational rules. Alternatively, we can also view $\simeq$ as a naive bisimulation on a richer form of transition system. More concretely, let us define a parameterized clock transition relation

$$P \xrightarrow[L]{\sigma} P' \text{ if and only if } P \xrightarrow{\sigma} P' \text{ and } \mathcal{I}\!\mathcal{I}_\sigma(P) \subseteq L$$

where $L \subseteq \mathcal{A} \setminus \{\tau\}$. We can interpret the transition $P \xrightarrow[L]{\sigma} P'$ as a *constrained* clock transition, reading "P may engage in σ *provided* that none of the communications in L is possible in any context." Because of the condition $\mathcal{I}\!\mathcal{I}_\sigma(P) \subseteq L$ this constraint mentions at least those initial actions that are in the scope of σ. We may then reformulate Definition 7.5.2 as a naive temporal strong bisimulation in terms of these new clock transitions.

Definition 7.5.18 (Alternative Temporal Strong Bisimulation)
A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is an alternative temporal strong bisimulation *if for every $\langle P, Q\rangle \in \mathcal{R}$, $\alpha \in \mathcal{A}$, $\sigma \in \mathcal{T}$, and $L \subseteq \mathcal{A} \setminus \{\tau\}$ the following conditions hold.*

1. *$P \xrightarrow{\alpha} P'$ implies $\exists Q'.\, Q \xrightarrow{\alpha} Q'$ and $\langle P', Q'\rangle \in \mathcal{R}$, and*
2. *$P \xrightarrow[L]{\sigma} P'$ implies $\exists Q'.\, Q \xrightarrow[L]{\sigma} Q'$, and $\langle P', Q'\rangle \in \mathcal{R}$.*

We write $P \simeq_* Q$ if $\langle P, Q\rangle \in \mathcal{R}$ for some alternative temporal strong bisimulation $\mathcal{R}$.

The following theorem states the desired result that both relations, temporal strong bisimulation and alternative temporal strong bisimulation, coincide.

Theorem 7.5.19 (Operational Characterization) *The coincidence $\simeq_* \;=\; \simeq$ holds.*

Proof: First, we prove the inclusion "$\subseteq$." Let $P, Q \in \mathcal{P}$ such that $P \simeq_* Q$. In order to establish $P \simeq Q$ it is by Definition 7.5.2 sufficient to show that $\simeq_*$ is a temporal strong bisimulation.

- Let $P \xrightarrow{\alpha} P'$ for some $P' \in \mathcal{P}$ and $\alpha \in \mathcal{A}$. Because of $P \simeq_* Q$ we know of the existence of some $Q' \in \mathcal{P}$ such that $Q \xrightarrow{\alpha} Q'$ and $P' \simeq_* Q'$, as desired.
- Let $P \xrightarrow{\sigma} P'$ for some $P' \in \mathcal{P}$ and $\sigma \in \mathcal{T}$. By the definition of $\xrightarrow[L]{\sigma}$ we may conclude $P \xrightarrow[L]{\sigma} P'$ for $L = \mathcal{I}\!\mathcal{I}_\sigma(P)$. Since $P \simeq_* Q$ we know of the existence of some $Q' \in \mathcal{P}$ such that $Q \xrightarrow[L]{\sigma} Q'$ and $P' \simeq_* Q'$. Thus, $Q \xrightarrow{\sigma} Q'$ and $\mathcal{I}\!\mathcal{I}_\sigma(Q) \subseteq \mathcal{I}\!\mathcal{I}_\sigma(P) = L$, which finishes this case.

The other necessary inclusion "$\supseteq$" is established by proving that $\simeq$ is an alternative temporal strong bisimulation. Let $P, Q \in \mathcal{P}$ such that $P \simeq Q$.

- The case $P \xrightarrow{\alpha} P'$ for some $P' \in \mathcal{P}$ and $\alpha \in \mathcal{A}$ is again straightforward since Part (1) of Definition 7.5.2 and Part (1) of Definition 7.5.18 coincide.

- Let $P \xrightarrow[L]{\sigma} P'$ for some $P' \in \mathcal{P}$, $\sigma \in \mathcal{T}$, and $L \subseteq \mathcal{A} \setminus \{\tau\}$. By the definition of $\xrightarrow[L]{\sigma}$ we obtain $P \xrightarrow{\sigma} P'$ and $\mathit{\Pi}_\sigma(P) \subseteq L$. Because of $P \simeq Q$ we know $\mathit{\Pi}_\sigma(Q) \subseteq \mathit{\Pi}_\sigma(P)$ and of the existence of some $Q' \in \mathcal{P}$ such that $Q \xrightarrow{\sigma} Q'$ and $P' \simeq Q'$. Hence, $\mathit{\Pi}_\sigma(Q) \subseteq L$ by the transitivity of $\subseteq$ and $Q \xrightarrow[L]{\sigma} Q'$ by the definition of $\xrightarrow[L]{\sigma}$, as desired.

Summarizing, we have shown that both relations, $\simeq_*$ and $\simeq$, coincide. □

The characterization of temporal strong bisimulation as a standard bisimulation on enriched transition systems allows its computation for finite-state processes using standard partition-refinement algorithms [103, 138]. As already mentioned in Section 6.4.3, the complexity of the most efficient algorithm by Paige and Tarjan [138] is linear in the size of the transition relation of the considered process(es) and logarithmic in the size of the state space. However, our transition relation for clock transitions is parameterized by a subset of the visible alphabet of interest, i.e. the finite sort(s) of the considered process(es). This exponential "blow-up," however, is usually not harmful in practice since most actions used in a system definition are internal, and only a few of them remain visible for an external observer.

Another possibility for computing temporal strong bisimulation, which avoids the exponential blow-up, is to start with the partition that corresponds to the equivalence relation

$$\{\langle P, Q\rangle \in \mathcal{P} \times \mathcal{P} \mid \forall \sigma \in \mathcal{T}.\ \tau \in \mathcal{I}_\sigma(P) \text{ if and only if } \tau \in \mathcal{I}_\sigma(Q)\} \cap$$
$$\{\langle P, Q\rangle \in \mathcal{P} \times \mathcal{P} \mid \forall \sigma \in \mathcal{T}.\ \tau \notin \mathcal{I}_\sigma(P) \cup \mathcal{I}_\sigma(Q) \text{ implies } \mathit{\Pi}_\sigma(P) = \mathit{\Pi}_\sigma(Q)\}\ .$$

Then standard algorithms may be applied to finish the task. The reason that we have presented the characterization of temporal strong bisimulation via an enriched transition relation is that (i) this technique allows us to adapt a logical characterization from CCS [125] as shown in the next section and (ii) it also applies to temporal weak bisimulation, which is developed in Section 7.6, where the naive technique fails.

7.5.4 Logical Characterization

In this section we provide a logical characterization of $\simeq$ by adapting the *Hennessy-Milner logic* presented in Section 6.4.4 to CSA. Syntactically, we replace the operators $\langle\underline{\alpha}\rangle$ and $\langle\alpha, L\rangle$ by $\langle\alpha\rangle$ and $\langle\sigma, L\rangle$, respectively. Here, L is a subset of visible actions. Semantically, we define $P \models \langle\alpha\rangle\Phi$ if $\exists P' \in \mathcal{P}.\ P \xrightarrow{\alpha} P'$ and $P' \models \Phi$, as well as $P \models \langle\sigma, L\rangle\Phi$ if $\exists P' \in \mathcal{P}.\ P \xrightarrow[L]{\sigma} P'$ and $P' \models \Phi$. Intuitively, $\langle\alpha\rangle\Phi$ is satisfied by every process which possesses an α-transition to some process that satisfies Φ. Similarly, P satisfies $\langle\sigma, L\rangle\Phi$ if P possesses an enriched σ-transition, parameterized by L, to a process satisfying Φ. According to the definition of the enriched transition relation this implies that the pre-emptive power of P with respect to clock σ may be at most L. The next theorem states that our logic characterizes temporal strong bisimulation if the clock universe $\mathcal{T}$ is finite (see also Section 7.9.4).

Theorem 7.5.20 (Logical Characterization of $\simeq$)
Let $P, Q \in \mathcal{P}$ and $|\mathcal{T}| < \infty$. Then we have: $P \simeq Q$ if and only if $\{\Phi \in \mathcal{F} \mid P \models \Phi\} = \{\Phi \in \mathcal{F} \mid Q \models \Phi\}$.

The proof of this theorem is very similar to the proof of Theorem 6.4.17 and, therefore, omitted here. Note that a finite clock universe guarantees that all processes in CSA give rise to finite-branching transition systems.

7.6 A Semantic Theory based on Weak Bisimulation

Since temporal strong bisimulation treats the internal action τ as any observable action a it is too fine for verifying systems in practice. Therefore, we concentrate in this section in developing a behavioral congruence that abstracts from internal transitions.

Observational equivalence is a notion of bisimulation in which any sequence of internal τ's may be skipped. For $\gamma \in \mathcal{A} \cup \mathcal{T}$ we define $\hat{\gamma} =_{\mathrm{df}} \epsilon$ if $\gamma \equiv \tau$ and $\hat{\gamma} =_{\mathrm{df}} \gamma$, otherwise. Further, let $\stackrel{\epsilon}{\Longrightarrow} =_{\mathrm{df}} \stackrel{\tau}{\longrightarrow}^*$ and $P \stackrel{\gamma}{\Longrightarrow} Q$ if there exist processes R and S such that $P \stackrel{\epsilon}{\Longrightarrow} R \stackrel{\gamma}{\longrightarrow} S \stackrel{\epsilon}{\Longrightarrow} Q$. Carrying over Milner's weak bisimulation [125] to CSA naively, without distinguishing between actions and clocks, would suggest the following definition.

Definition 7.6.1 (Naive Temporal Weak Bisimulation)
A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a naive temporal weak bisimulation *if for every $\langle P, Q\rangle \in \mathcal{R}$, $\gamma \in \mathcal{A} \cup \mathcal{T}$, the following condition holds:*

$$P \stackrel{\gamma}{\longrightarrow} P' \text{ implies } \exists Q'.\, Q \stackrel{\hat{\gamma}}{\Longrightarrow} Q' \text{ and } \langle P', Q'\rangle \in \mathcal{R} \ .$$

We write $P \approx_{\mathrm{n}} Q$ if $\langle P, Q\rangle \in \mathcal{R}$ for some naive temporal weak bisimulation $\mathcal{R}$.

It is not surprising that $\approx_{\mathrm{n}}$ is not a congruence, for the same reason that weak bisimulation is not a congruence for CCS. In contrast to CCS, however, $\approx_{\mathrm{n}}$ is not even a congruence for parallel composition. The problem is that, again, the relation fails to account for clock scoping.

7.6.1 Temporal Weak Bisimulation

A careful analysis of the clock pre-emption caused by local maximal progress shows that the following refinement of the above definition of naive temporal weak bisimulation is needed for the *static contexts*, i.e. contexts constructed by using *static* operators only.

Definition 7.6.2 (Temporal Weak Bisimulation)
A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a temporal weak bisimulation *if for every $\langle P, Q\rangle \in \mathcal{R}$, $\alpha \in \mathcal{A}$, and $\sigma \in \mathcal{T}$ the following conditions hold.*

1. *$P \stackrel{\alpha}{\longrightarrow} P'$ implies $\exists Q'.\, Q \stackrel{\hat{\alpha}}{\Longrightarrow} Q'$ and $\langle P', Q'\rangle \in \mathcal{R}$, and*

2. $P \xrightarrow{\sigma} P'$ *implies* $\exists Q', Q'', Q'''. Q \stackrel{\epsilon}{\Longrightarrow} Q'' \xrightarrow{\sigma} Q''' \stackrel{\epsilon}{\Longrightarrow} Q'$, $\mathit{\Pi}_\sigma(Q'') \subseteq \mathit{\Pi}_\sigma(P)$, *and* $\langle P', Q' \rangle \in \mathcal{R}$.

We write $P \approx Q$ *if* $\langle P, Q \rangle \in \mathcal{R}$ *for some temporal weak bisimulation* $\mathcal{R}$.

From this definition, we may immediately conclude that $\approx$ is the *largest* temporal weak bisimulation, and that $\approx$ is an equivalence relation. Moreover, we have the following proposition similar to Propositions 2.4.6 and 6.5.5 for $\mathsf{CCS}^{\mathsf{ch}}$ and $\mathsf{CCS}^{\mathsf{prio}}$, respectively.

Proposition 7.6.3 *The relation* $\approx$ *is a congruence with respect to prefixing and the static operators. It is characterized as the largest congruence contained in* $\approx_{\mathrm{n}}$*, in the sub-algebra of* CSA *induced by these operators and recursion.*

The proof of this proposition uses the following lemma.

Lemma 7.6.4 *Let* $P, P', Q \in \mathcal{P}$ *and* $\sigma \in \mathcal{T}$ *such that* $P \stackrel{\epsilon}{\Longrightarrow} P'$. *Then*

1. $P \mid Q \stackrel{\epsilon}{\Longrightarrow} P' \mid Q$ *and* $Q \mid P \stackrel{\epsilon}{\Longrightarrow} Q \mid P'$, *and*
2. $P{\uparrow}\sigma \stackrel{\epsilon}{\Longrightarrow} P'{\uparrow}\sigma$.

This lemma can be proved straightforwardly by using induction on the length of the temporal weak transition $P \stackrel{\epsilon}{\Longrightarrow} P'$. Now, we are able to prove Proposition 7.6.3.

Proof: In the following we prove the congruence property of $\approx$ with respect to the sub-algebra of CSA under consideration. The *largest* statement of the proposition is a consequence of the proof of Theorem 7.6.8 presented below. We show that $\approx$ is compositional with respect to prefixing and the static operators of CSA, i.e. parallel composition, restriction, relabeling, ignore, and recursion. Most cases are standard and can be checked along the lines of [125]. Therefore, we restrict ourselves to the more interesting proof parts. In the following let $P, Q, R \in \mathcal{P}$ be processes such that $P \approx Q$ and let $\sigma \in \mathcal{T}$. Then

1. $P \mid R \approx Q \mid R$, and
2. $P{\uparrow}\sigma \approx Q{\uparrow}\sigma$,

which is proved as follows.

1. According to Definition 7.6.2 it is sufficient to establish that the symmetric relation

$$\mathcal{R} =_{\mathrm{df}} \{\langle P \mid R, Q \mid R \rangle \mid P \approx Q\,,\ R \in \mathcal{P}\}$$

is a temporal weak bisimulation. Let $\langle P \mid R, Q \mid R \rangle$ be an arbitrary pair in $\mathcal{R}$.

- **Case 1:** The case where $P \mid R \xrightarrow{\alpha} S$ for some $S \in \mathcal{P}$ and $\alpha \in \mathcal{A}$ is standard.

- **Case 2:** Let $P \mid R \xrightarrow{\sigma} S$ for some $S \in \mathcal{P}$ and $\sigma \in \mathcal{T}$. By the only applicable Rule (tCom) we know that $P \xrightarrow{\sigma} P'$ for some $P' \in \mathcal{P}$, $R \xrightarrow{\sigma} R'$ for some $R' \in \mathcal{P}$, $\mathcal{I}\!\!\mathcal{I}_\sigma(P) \cap \overline{\mathcal{I}\!\!\mathcal{I}_\sigma(R)} = \emptyset$, and $S \equiv P' \mid R'$. Since $P \approx Q$ there exist processes $Q', Q'', Q''' \in \mathcal{P}$ such that $Q \stackrel{\epsilon}{\Longrightarrow} Q'' \xrightarrow{\sigma} Q''' \stackrel{\epsilon}{\Longrightarrow} Q'$, $\mathcal{I}\!\!\mathcal{I}_\sigma(Q'') \subseteq \mathcal{I}\!\!\mathcal{I}_\sigma(P)$ and $P' \approx Q'$. First, observe that $\mathcal{I}\!\!\mathcal{I}_\sigma(Q'') \cap \overline{\mathcal{I}\!\!\mathcal{I}_\sigma(R)} \subseteq \mathcal{I}\!\!\mathcal{I}_\sigma(P) \cap \overline{\mathcal{I}\!\!\mathcal{I}_\sigma(R)} = \emptyset$. Applying Lemma 7.6.4(1) and Rule (tCom) again we conclude that $Q \mid R \stackrel{\epsilon}{\Longrightarrow} Q'' \mid R \xrightarrow{\sigma} Q''' \mid R' \stackrel{\epsilon}{\Longrightarrow} Q' \mid R'$. Moreover, $\mathcal{I}\!\!\mathcal{I}_\sigma(Q'' \mid R) = \mathcal{I}\!\!\mathcal{I}_\sigma(Q'') \cup \mathcal{I}\!\!\mathcal{I}_\sigma(R) \subseteq \mathcal{I}\!\!\mathcal{I}_\sigma(P) \cup \mathcal{I}\!\!\mathcal{I}_\sigma(R) = \mathcal{I}\!\!\mathcal{I}_\sigma(P \mid R)$ and $\langle P' \mid R', Q' \mid R' \rangle \in \mathcal{R}$ hold due to the definitions of $\mathcal{I}\!\!\mathcal{I}_\sigma(\cdot)$ and of $\mathcal{R}$, respectively, which completes this proof part.

2. According to Definition 7.6.2 it is sufficient to prove that the symmetric relation

$$\mathcal{R} =_{\mathrm{df}} \{\langle P\!\upharpoonright\!\sigma, Q\!\upharpoonright\!\sigma \rangle \mid P \approx Q\}$$

is a temporal weak bisimulation. Therefore, let $\langle P\!\upharpoonright\!\sigma, Q\!\upharpoonright\!\sigma \rangle \in \mathcal{R}$ be arbitrary.

- **Case 1:** Let $P\!\upharpoonright\!\sigma \xrightarrow{\alpha} P'\!\upharpoonright\!\sigma$ for some $P' \in \mathcal{P}$, i.e. $P \xrightarrow{\alpha} P'$ by Rule (Ign). Since $P \approx Q$ we know of the existence of some $Q' \in \mathcal{P}$ such that $Q \stackrel{\hat{\alpha}}{\Longrightarrow} Q'$ and $P' \approx Q'$. Applying Lemma 7.6.4(2) and the definition of $\mathcal{R}$ yields $Q\!\upharpoonright\!\sigma \stackrel{\hat{\alpha}}{\Longrightarrow} Q'\!\upharpoonright\!\sigma$ and $\langle P'\!\upharpoonright\!\sigma, Q'\!\upharpoonright\!\sigma \rangle \in \mathcal{R}$.
- **Case 2:** Let $P\!\upharpoonright\!\sigma \xrightarrow{\sigma} P\!\upharpoonright\!\sigma$ by Rule (tIgn1). By the very same rule also $Q\!\upharpoonright\!\sigma \xrightarrow{\sigma} Q\!\upharpoonright\!\sigma$ holds and, thus, $Q\!\upharpoonright\!\sigma \stackrel{\epsilon}{\Longrightarrow} Q\!\upharpoonright\!\sigma \xrightarrow{\sigma} Q\!\upharpoonright\!\sigma \stackrel{\epsilon}{\Longrightarrow} Q\!\upharpoonright\!\sigma$ according to Lemma 7.6.4(2). Moreover, $\mathcal{I}\!\!\mathcal{I}_\sigma(Q\!\upharpoonright\!\sigma) = \emptyset = \mathcal{I}\!\!\mathcal{I}_\sigma(P\!\upharpoonright\!\sigma)$. Since $\langle P\!\upharpoonright\!\sigma, Q\!\upharpoonright\!\sigma \rangle \in \mathcal{R}$ we have finished this case.
- **Case 3:** Let $P\!\upharpoonright\!\sigma \xrightarrow{\sigma'} P'\!\upharpoonright\!\sigma$ for some $P' \in \mathcal{P}$ and $\sigma' \not\equiv \sigma$, i.e. $P \xrightarrow{\sigma'} P'$ by Rule (tIgn2). Since $P \approx Q$ we know of the existence of processes $Q', Q'', Q''' \in \mathcal{P}$ such that $Q \stackrel{\epsilon}{\Longrightarrow} Q'' \xrightarrow{\sigma'} Q''' \stackrel{\epsilon}{\Longrightarrow} Q'$, $\mathcal{I}\!\!\mathcal{I}_{\sigma'}(Q'') \subseteq \mathcal{I}\!\!\mathcal{I}_{\sigma'}(P)$, and $P' \approx Q'$. By Rule (Ign), Lemma 7.6.4(2), and by the definition of $\mathcal{I}\!\!\mathcal{I}_{\sigma'}(\cdot)$ we may conclude $Q\!\upharpoonright\!\sigma \stackrel{\epsilon}{\Longrightarrow} Q''\!\upharpoonright\!\sigma \xrightarrow{\sigma'} Q'''\!\upharpoonright\!\sigma \stackrel{\epsilon}{\Longrightarrow} Q'\!\upharpoonright\!\sigma$. Moreover, $\mathcal{I}\!\!\mathcal{I}_{\sigma'}(Q''\!\upharpoonright\!\sigma) = \mathcal{I}\!\!\mathcal{I}_{\sigma'}(Q'') \subseteq \mathcal{I}\!\!\mathcal{I}_{\sigma'}(P) = \mathcal{I}\!\!\mathcal{I}_{\sigma'}(P\!\upharpoonright\!\sigma)$ by the definition of $\mathcal{I}\!\!\mathcal{I}_{\sigma'}(\cdot)$. According to the definition of $\mathcal{R}$ also $\langle P'\!\upharpoonright\!\sigma, Q'\!\upharpoonright\!\sigma \rangle \in \mathcal{R}$ holds.

The compositionality proofs of the other static operators, i.e. restriction and relabeling, follow closely the usual pattern and are omitted here. In order to show that $\approx$ is compositional with respect to recursion, we need to define a notion of *temporal weak bisimulation up to* $\approx$ (cf. [149]) which can be done in the obvious fashion. Then the proof follows the lines of [125] by using the following property in the case of parallel composition.

> Let $P, Q, R, S \in \mathcal{P}$ such that $P \approx R$ and $Q \approx S$. Moreover, let $P \mid Q \xrightarrow{\sigma} P' \mid Q'$ for some $\sigma \in \mathcal{T}$ and $P', Q' \in \mathcal{P}$. Then there exist $R', R'', R''', S', S'', S''' \in \mathcal{P}$ satisfying $R \mid S \stackrel{\epsilon}{\Longrightarrow} R'' \mid S'' \xrightarrow{\sigma} R''' \mid S''' \stackrel{\epsilon}{\Longrightarrow} R' \mid S'$, $\mathcal{I}\!\!\mathcal{I}_\sigma(R'' \mid S'') \subseteq \mathcal{I}\!\!\mathcal{I}_\sigma(P \mid Q)$, $P' \approx R'$, and $Q' \approx S'$.

The property's validity can be shown as follows. Since $P \mid Q \xrightarrow{\sigma} P' \mid Q'$ we know by Rule (tCom) that $P \xrightarrow{\sigma} P'$, $Q \xrightarrow{\sigma} Q'$, and $\mathit{\Pi}_\sigma(P) \cap \overline{\mathit{\Pi}_\sigma(Q)} = \emptyset$. Using the premise $P \approx R$ we conclude the existence of processes $R', R'', R''' \in \mathcal{P}$ such that $R \stackrel{\epsilon}{\Longrightarrow} R'' \xrightarrow{\sigma} R''' \stackrel{\epsilon}{\Longrightarrow} R'$ where $\mathit{\Pi}_\sigma(R'') \subseteq \mathit{\Pi}_\sigma(P)$ and $P' \approx R'$. Similarly, there exists processes $S', S'', S''' \in \mathcal{P}$ such that $S \stackrel{\epsilon}{\Longrightarrow} S'' \xrightarrow{\sigma} S''' \stackrel{\epsilon}{\Longrightarrow} S'$, $\mathit{\Pi}_\sigma(S'') \subseteq \mathit{\Pi}_\sigma(Q)$, and $Q' \approx S'$. Applying our semantic rules and Lemma 7.6.4(1) we obtain $R \mid S \stackrel{\epsilon}{\Longrightarrow} R'' \mid S'' \xrightarrow{\sigma} R''' \mid S''' \stackrel{\epsilon}{\Longrightarrow} R' \mid S'$, because $\mathit{\Pi}_\sigma(R'') \cap \overline{\mathit{\Pi}_\sigma(S'')} \subseteq \mathit{\Pi}_\sigma(P) \cap \overline{\mathit{\Pi}_\sigma(Q)} = \emptyset$. Moreover, $\mathit{\Pi}_\sigma(R'' \mid S'') = \mathit{\Pi}_\sigma(R'') \cup \mathit{\Pi}_\sigma(S'') \subseteq \mathit{\Pi}_\sigma(P) \cup \mathit{\Pi}_\sigma(Q) = \mathit{\Pi}_\sigma(P \mid Q)$, as desired. □

7.6.2 Temporal Observational Congruence

In order to identify the largest equivalence contained in $\approx_{\mathrm{n}}$, that is also a congruence for the other dynamic operators, the summation fix of CCS is not sufficient due to the special nature of clock transitions. Note that such an equivalence exists according to Proposition 2.4.3.

Definition 7.6.5 (Temporal Observational Congruence)
A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a temporal observational congruence *if for every $\langle P, Q\rangle \in \mathcal{R}$, $\alpha \in \mathcal{A}$, and $\sigma \in \mathcal{T}$ the following conditions hold.*

1. *$P \xrightarrow{\alpha} P'$ implies $\exists Q'.\, Q \stackrel{\alpha}{\Longrightarrow} Q'$ and $P' \approx Q'$.*
2. *$P \xrightarrow{\sigma} P'$ implies $\mathit{\Pi}_\sigma(Q) \subseteq \mathit{\Pi}_\sigma(P)$ and $\exists Q'.\, Q \xrightarrow{\sigma} Q'$ and $\langle P', Q'\rangle \in \mathcal{R}$.*

We write $P \mathrel{\underline{\approx}} Q$ if $\langle P, Q\rangle \in \mathcal{R}$ for some temporal observational congruence $\mathcal{R}$.

We first show that $\mathrel{\underline{\approx}}$ is compositional with respect to all operators in CSA.

Proposition 7.6.6 *The relation $\mathrel{\underline{\approx}}$ is a congruence.*

Proof: The compositionality of $\mathrel{\underline{\approx}}$ is easy to show for the cases of prefixing, restriction, and relabeling. In the following we deal with the remaining, more interesting cases. Let $P, Q, R, S \in \mathcal{P}$ be are arbitrary processes such that $P \mathrel{\underline{\approx}} Q$, $R \mathrel{\underline{\approx}} S$, and $\sigma \in \mathcal{T}$. Then

1. $P \mid R \mathrel{\underline{\approx}} Q \mid R$,
2. $P + R \mathrel{\underline{\approx}} Q + R$,
3. $P{\uparrow}\sigma \mathrel{\underline{\approx}} Q{\uparrow}\sigma$,
4. $\lfloor P \rfloor \sigma(R) \mathrel{\underline{\approx}} \lfloor Q \rfloor \sigma(S)$, and
5. $P{\downarrow}\sigma \mathrel{\underline{\approx}} Q{\downarrow}\sigma$,

which can be established as follows.

1. According to Definition 7.6.5 it is sufficient to prove that the symmetric relation

$$\mathcal{R} =_{\mathrm{df}} \{\langle P \mid R, Q \mid R\rangle \mid P \underline{\approx} Q\,, R \in \mathcal{P}\}$$

is a temporal observational congruence. Let $\langle P \mid R, Q \mid R\rangle \in \mathcal{R}$ be arbitrary.

- **Case 1:** The case where $P \mid R \stackrel{\alpha}{\longrightarrow} S$ for some $S \in \mathcal{P}$ and $\alpha \in \mathcal{A}$ is standard. Here, one has to use the part of Proposition 7.6.3 which deals with parallel composition.
- **Case 2:** Let $P \mid R \stackrel{\sigma}{\longrightarrow} S$ for some $S \in \mathcal{P}$ and $\sigma \in \mathcal{T}$. This case can easily be done along the lines of the corresponding case in the proof of Proposition 7.5.3.

2. By Definition 7.6.5 it is sufficient to establish that the symmetric relation

$$\mathcal{R} =_{\mathrm{df}} \{\langle P + R, Q + R\rangle \mid P \underline{\approx} Q\,, R \in \mathcal{P}\}$$

is a temporal observational congruence. Therefore, let $\langle P + R, Q + R\rangle$ be an arbitrary pair in $\mathcal{R}$.

- **Case 1:** Let $P + R \stackrel{\alpha}{\longrightarrow} V$ for some $\alpha \in \mathcal{A}$ and $V \in \mathcal{P}$. Since the operational rules for summation with respect to actions are identical to the ones in CCS, and the definition of temporal observational congruence coincides with the one of observational congruence in CCS in this particular case, the proof follows the lines of the corresponding proof in the CCS framework.
- **Case 2:** Let $P + R \stackrel{\sigma}{\longrightarrow} V$ for some $\sigma \in \mathcal{T}$ and $V \in \mathcal{P}$, i.e. $P \stackrel{\sigma}{\longrightarrow} P'$ and $R \stackrel{\sigma}{\longrightarrow} R'$ for some $P', R' \in \mathcal{P}$, and $V \equiv P' + R'$ by Rule (tSum). Since $P \underline{\approx} Q$ we know of the existence of some $Q' \in \mathcal{P}$ such that $Q \stackrel{\sigma}{\longrightarrow} Q'$, $\mathit{\Pi}_\sigma(Q) \subseteq \mathit{\Pi}_\sigma(P)$, and $P' \underline{\approx} Q'$. Therefore, we may conclude $Q + R \stackrel{\sigma}{\longrightarrow} Q' + R'$ by Rule (tSum) and $\langle P' + R', Q' + R'\rangle \in \mathcal{R}$ by the definition of $\mathcal{R}$. Moreover, we have $\mathit{\Pi}_\sigma(Q + R) = \mathit{\Pi}_\sigma(Q) \cup \mathit{\Pi}_\sigma(R) \subseteq \mathit{\Pi}_\sigma(P) \cup \mathit{\Pi}_\sigma(R) = \mathit{\Pi}_\sigma(P + R)$ by the definition of $\mathit{\Pi}_\sigma(\cdot)$, which finishes this part of the proof.

3. It is sufficient to show that the symmetric relation

$$\mathcal{R} =_{\mathrm{df}} \{\langle P{\uparrow}\sigma, Q{\uparrow}\sigma\rangle \mid P \underline{\approx} Q\}$$

is a temporal observational congruence. Thus, let $\langle P{\uparrow}\sigma, Q{\uparrow}\sigma\rangle \in \mathcal{R}$ be arbitrary.

- **Case 1:** Let $P{\uparrow}\sigma \stackrel{\alpha}{\longrightarrow} P'{\uparrow}\sigma$, i.e. $P \stackrel{\alpha}{\longrightarrow} P'$ by Rule (lgn). Since $P \underline{\approx} Q$ we know of the existence of some $Q' \in \mathcal{P}$ such that $Q \stackrel{\alpha}{\Longrightarrow} Q'$ and $P' \approx Q'$. According to Rule (lgn) and Proposition 7.6.3 we may conclude that $Q{\uparrow}\sigma \stackrel{\alpha}{\Longrightarrow} Q'{\uparrow}\sigma$ and, moreover, $P'{\uparrow}\sigma \approx Q'{\uparrow}\sigma$.
- **Case 2:** Let $P{\uparrow}\sigma \stackrel{\sigma}{\longrightarrow} P{\uparrow}\sigma$ by Rule (tlgn1). Here we can argue along the lines of the similar case in the proof of Proposition 7.5.3.

- **Case 3:** Let $P\!\uparrow\!\sigma \xrightarrow{\sigma'} P'\!\uparrow\!\sigma$ for some $\sigma' \not\equiv \sigma$. Also this case is analogous to the corresponding case in the proof of Proposition 7.5.3.

4. By Definition 7.6.5 it is sufficient to show that the symmetric relation

$$\mathcal{R} =_{\mathrm{df}} \{\langle \lfloor P \rfloor \sigma(R), \lfloor Q \rfloor \sigma(S)\rangle \mid P \underline{\approx} Q \text{ and } R \underline{\approx} S\} \cup \underline{\approx}$$

is a temporal observational congruence. The proof is similar to one which has been presented for Proposition 7.5.3.

5. Here, it is sufficient to show that the symmetric relation

$$\mathcal{R} =_{\mathrm{df}} \{\langle P\!\downarrow\!\sigma, Q\!\downarrow\!\sigma\rangle \mid P \underline{\approx} Q\}$$

is a temporal observational congruence. Therefore, let $\langle P\!\downarrow\!\sigma, Q\!\downarrow\!\sigma\rangle \in \mathcal{R}$.

- **Case 1:** Let $P\!\downarrow\!\sigma \xrightarrow{\alpha} P'$, i.e. $P \xrightarrow{\alpha} P'$ by Rule (DIgn). Since $P \underline{\approx} Q$ we know of the existence of some $Q' \in \mathcal{P}$ such that $Q \stackrel{\alpha}{\Longrightarrow} Q'$ and $P' \approx Q'$. According to Rule (DIgn) we may conclude that $Q\!\downarrow\!\sigma \stackrel{\alpha}{\Longrightarrow} Q'$ which completes this case.
- **Case 2:** Let $P\!\downarrow\!\sigma \xrightarrow{\sigma} P\!\downarrow\!\sigma$ by Rule (tDIgn1). Because of the very same rule we have $Q\!\downarrow\!\sigma \xrightarrow{\sigma} Q\!\downarrow\!\sigma$, too. Additionally, $\langle P\!\downarrow\!\sigma, Q\!\downarrow\!\sigma\rangle \in \mathcal{R}$ holds by assumption, and $\mathit{\Pi}_\sigma(Q\!\downarrow\!\sigma) = \emptyset = \mathit{\Pi}_\sigma(P\!\downarrow\!\sigma)$ by the definition of $\mathit{\Pi}_\sigma(\cdot)$.
- **Case 3:** Let $P\!\downarrow\!\sigma \xrightarrow{\sigma'} P'\!\downarrow\!\sigma$ for some $\sigma' \not\equiv \sigma$. Thus, $P \xrightarrow{\sigma'} P'$ by Rule (tDIgn2). Since $P \underline{\approx} Q$ we know that $Q \xrightarrow{\sigma'} Q'$ for some $Q' \in \mathcal{P}$ satisfying $\mathit{\Pi}_{\sigma'}(Q) \subseteq \mathit{\Pi}_{\sigma'}(P)$ and $P' \underline{\approx} Q'$. Applying Rule (tDIgn2) again and because of the definitions of $\mathit{\Pi}_{\sigma'}(\cdot)$ and $\mathcal{R}$, we may conclude $Q\!\downarrow\!\sigma \xrightarrow{\sigma'} Q'\!\downarrow\!\sigma$, $\mathit{\Pi}_{\sigma'}(Q\!\downarrow\!\sigma) \subseteq \mathit{\Pi}_{\sigma'}(P\!\downarrow\!\sigma)$, and $\langle P'\!\downarrow\!\sigma, Q'\!\downarrow\!\sigma\rangle \in \mathcal{R}$.

In order to show that $\underline{\approx}$ is compositional with respect to recursion, we have to adapt a notion of "up to" again.

> A relation $\mathcal{R} \subseteq \mathcal{P}\times\mathcal{P}$ is a *temporal observational congruence up to* $\underline{\approx}$ if for every $\langle P, Q\rangle \in \mathcal{R}$, $\alpha \in \mathcal{A}$, and $\sigma \in \mathcal{T}$ the following conditions and their symmetric counterparts hold.
>
> 1. $P \xrightarrow{\alpha} P'$ implies $\exists Q'.\, Q \stackrel{\alpha}{\Longrightarrow} Q'$ and $P'\ \mathcal{R} \approx\ Q'$, and
> 2. $P \xrightarrow{\sigma} P'$ implies $\exists Q'.\, Q \xrightarrow{\sigma} Q'$, $\mathit{\Pi}_\sigma(Q) \subseteq \mathit{\Pi}_\sigma(P)$, and $P'\ \mathcal{R} \underline{\approx}\ Q'$.

The proof follows the standard lines [125] and, therefore, is omitted. □

Before stating the main theorem of this section we first relate temporal observational congruence to temporal weak bisimulation.

Proposition 7.6.7 *Temporal observational congruence* $\underline{\approx}$ *is the* largest *congruence contained in naive temporal weak bisimulation* $\approx$, *i.e.* $\underline{\approx} = \approx^{+}$.

Proof: By Proposition 2.4.3 we know that the *largest* congruence $\approx^+$ in $\approx$ exists and that $\approx^+ = \{\langle P, Q\rangle \mid \forall C[X].\, C[P] \approx C[Q]\}$. Since $\underline{\approx}$ is a congruence by Proposition 7.6.6 and obviously contained in $\approx$, the inclusion $\underline{\approx} \subseteq \approx^+$ holds. Thus, it remains to show $\approx^+ \subseteq \underline{\approx}$. Consider the symmetric relation $\underline{\approx}_{\mathrm{a}} =_{\mathrm{df}} \{\langle P, Q\rangle \mid P + c.\mathbf{0} \approx Q + c.\mathbf{0} \text{ for } c \notin \mathcal{S}(P) \cup \mathcal{S}(Q)\}$. By the definition of $\underline{\approx}_{\mathrm{a}}$ we have $\approx^+ \subseteq \underline{\approx}_{\mathrm{a}}$. We establish the other necessary inclusion $\underline{\approx}_{\mathrm{a}} \subseteq \underline{\approx}$ by proving that $\underline{\approx}_{\mathrm{a}}$ is a temporal observational congruence. Let $P \,\underline{\approx}_{\mathrm{a}}\, Q$, i.e. $P + c.\mathbf{0} \approx Q + c.\mathbf{0}$, and distinguish the following cases.

- **Case 1:** Let $P \xrightarrow{\alpha} P'$, i.e. $\alpha \not\equiv c$ and by Rule (Sum1) $P + c.\mathbf{0} \xrightarrow{\alpha} P'$. Since $P \,\underline{\approx}_{\mathrm{a}}\, Q$ we conclude the existence of a process $V \in \mathcal{P}$ satisfying $Q + c.\mathbf{0} \stackrel{\hat{\alpha}}{\Longrightarrow} V$ and $P' \approx V$. Since c is a distinguished action we have $V \not\equiv Q$ and, thus, $V \equiv Q'$ and $Q \stackrel{\alpha}{\Longrightarrow} Q'$ for some $Q' \in \mathcal{P}$.

- **Case 2:** Let $P \xrightarrow{\sigma} P'$. By Rules (tAct) and (tSum) $P + c.\mathbf{0} \xrightarrow{\sigma} P' + c.\mathbf{0}$ holds. Since $P \,\underline{\approx}_{\mathrm{a}}\, Q$ we know of the existence of processes $V, V', V'' \in \mathcal{P}$ such that $Q + c.\mathbf{0} \stackrel{\epsilon}{\Longrightarrow} V' \xrightarrow{\sigma} V'' \stackrel{\epsilon}{\Longrightarrow} V$, $\mathit{I\!I}_\sigma(V') \subseteq \mathit{I\!I}_\sigma(P)$, and $P' + c.\mathbf{0} \approx V$. Since c is a distinguished action not in the sorts of P and Q, we conclude $V' \equiv Q + c.\mathbf{0}$, $V'' \equiv Q' + c.\mathbf{0}$ for some $Q' \in \mathcal{P}$, $V \equiv V''$, $Q \xrightarrow{\sigma} Q'$, and $\mathit{I\!I}_\sigma(Q) \subseteq \mathit{I\!I}_\sigma(P)$. Moreover, $P' \approx_{\mathrm{a}} Q'$ by the definition of $\approx_{\mathrm{a}}$ and the fact that $\mathcal{S}(P') \subseteq \mathcal{S}(P)$ and $\mathcal{S}(Q') \subseteq \mathcal{S}(Q)$.

This shows that $\underline{\approx}_{\mathrm{a}}$ is a temporal observational congruence relation. Hence, $\underline{\approx}_{\mathrm{a}} \subseteq \underline{\approx}$. □

The following theorem is the main result of this section and "extends" the largest congruence result presented above with respect to temporal weak bisimulation to the coarser naive temporal weak bisimulation. Again, we use the same proof technique as for the analogous theorem in Chapters 2 and 6.

Theorem 7.6.8 *The relation* $\underline{\approx}$ *is the* largest congruence *contained in* $\approx_{\mathrm{n}}$, *i.e.* $\underline{\approx} = \approx_{\mathrm{n}}^+$.

Proof: We use Proposition 2.4.16 and choose $\mathcal{R}_1 = \approx_{\mathrm{n}}$ and $\mathcal{R}_2 = \approx$. The inclusion $\approx \subseteq \approx_{\mathrm{n}}$ follows immediately from the definition of the naive temporal weak and the temporal weak transition relation. In order to apply Proposition 2.4.16, we have to establish $\approx_{\mathrm{n}}^+ \subseteq \approx$. This inclusion turns out to be difficult to show directly. Instead, we introduce an auxiliary equivalence relation $\approx_{\mathrm{a}}$ and prove $\approx_{\mathrm{n}}^+ \subseteq \approx_{\mathrm{a}} \subseteq \approx$. In the following we define $\approx_{\mathrm{a}} =_{\mathrm{df}} \{\langle P, Q\rangle \mid C_{PQ}[P] \approx_{\mathrm{n}} C_{PQ}[Q]\}$. Here, the processes $C_{PQ}[X]$ are defined as in the proof of Theorem 7.5.4. By Proposition 2.4.3 we may immediately conclude that $\approx_{\mathrm{n}}^+ \subseteq \approx_{\mathrm{a}}$. The other necessary inclusion, $\approx_{\mathrm{a}} \subseteq \approx$, is established by proving that $\approx_{\mathrm{a}}$ is a temporal weak bisimulation. Let $P, Q \in \mathcal{P}$ satisfying $P \approx_{\mathrm{a}} Q$ and consider the following situations.

- **Situation 1:** Let $P \xrightarrow{\alpha} P'$ for some $P' \in \mathcal{P}$ and some $\alpha \in \mathcal{A}$. According to our operational semantics we may derive $C_{PQ}[P] \equiv P \mid H_{PQ} \xrightarrow{\alpha} P' \mid H_{PQ} \equiv C_{PQ}[P']$. This transition can only be matched by a corresponding weak transition of the process

Q, say $Q \stackrel{\hat{\alpha}}{\Longrightarrow} Q'$ for some $Q' \in \mathcal{P}$, since only the process H_{PQ} has the distinguished action e enabled. Therefore, we have $C_{PQ}[Q] \equiv Q \,|\, H_{PQ} \stackrel{\hat{\alpha}}{\Longrightarrow} Q' \,|\, H_{PQ} \equiv C_{PQ}[Q']$ and $C_{PQ}[P'] \approx_{\mathrm{n}} C_{PQ}[Q']$. Because $\mathcal{S}(P') \subseteq \mathcal{S}(P)$ and $\mathcal{S}(Q') \subseteq \mathcal{S}(Q)$, also $C_{P'Q'}[P'] \approx_{\mathrm{n}} C_{P'Q'}[Q']$ holds. Thus, $P' \approx_{\mathrm{a}} Q'$.

- **Situation 2:** Let $P \stackrel{\sigma}{\longrightarrow} P'$ for some $P' \in \mathcal{P}$ and some $\sigma \in \mathcal{T}$. As illustrated in Figure 7.4, $C_{PQ}[P]$ can perform a τ-transition to the process $P \,|\, H_L$, where $H_L \stackrel{\mathrm{def}}{=} D_L + d_L.H_{PQ}$ and $L =_{\mathrm{df}} \{\overline{c} \,|\, c \in (\mathcal{S}(P) \cup \mathcal{S}(Q)) \setminus \mathcal{I}_\sigma(P)\}$. Then, $P \,|\, H_L$ can engage in a σ-transition to $P' \,|\, H_L$ according to Rule (tCom). Finally, we consider the step $P' \,|\, H_L \stackrel{d_L}{\longrightarrow} P' \,|\, H_{PQ}$.

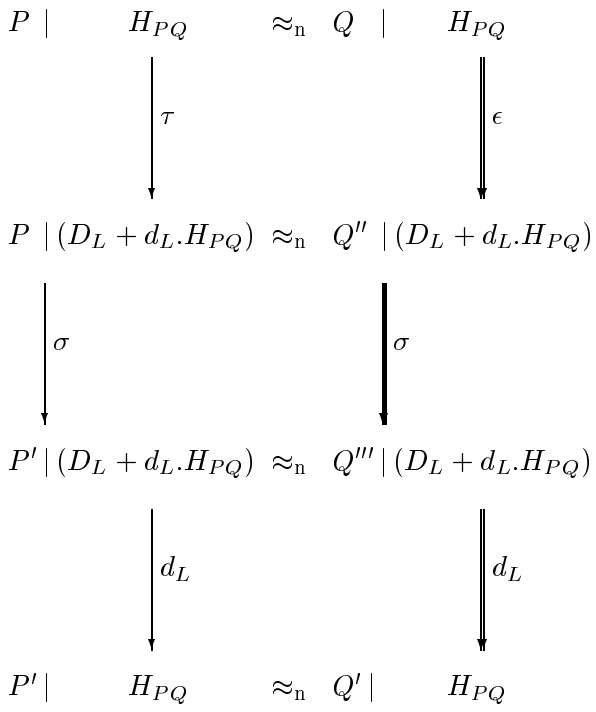

Figure 7.4: Largest congruence proof – illustration of Situation (2)

Let us have a look at the first step. Since $C_{PQ}[P] \approx_{\mathrm{n}} C_{PQ}[Q]$, we have $C_{PQ}[Q] \stackrel{\epsilon}{\Longrightarrow} W''$ for some $W'' \in \mathcal{P}$. We know that H_{PQ} has to perform a τ-transition to H_L since d_L is a distinguished action. However, Q may be able to perform some τ-transitions to some state $Q'' \in \mathcal{P}$, i.e. $Q \stackrel{\epsilon}{\Longrightarrow} Q''$ and $P \,|\, H_L \approx_{\mathrm{n}} Q'' \,|\, H_L$.

Now, we consider the more interesting second step. Since $P \,|\, H_L \approx_{\mathrm{n}} Q'' \,|\, H_L$, we know of the existence of some $W''' \in \mathcal{P}$ such that $Q'' \,|\, H_L \stackrel{\sigma}{\Longrightarrow} W'''$ and $P' \,|\, H_L \approx_{\mathrm{n}} W'''$. According to our operational semantics Q'' and H_L have to perform a naive temporal weak σ-transition. Since H_L can only engage in an idling σ-transition, i.e. $H_L \stackrel{\sigma}{\longrightarrow} H_L$,

we conclude $W''' \equiv Q''' \mid H_L$ for some process $Q''' \in \mathcal{P}$ such that $Q'' \stackrel{\sigma}{\Longrightarrow} Q'''$, i.e. $Q'' \stackrel{\epsilon}{\Longrightarrow} Q_1''' \stackrel{\sigma}{\longrightarrow} Q_2''' \stackrel{\epsilon}{\Longrightarrow} Q'''$ for some $Q_1''', Q_2''' \in \mathcal{P}$. Then, $Q'' \mid H_L \stackrel{\epsilon}{\Longrightarrow} Q_1''' \mid H_L \stackrel{\sigma}{\longrightarrow} Q_2''' \mid H_L \stackrel{\epsilon}{\Longrightarrow} Q''' \mid H_L$ must hold. According to Rule (tCom) the condition $\underline{\mathcal{I}_\sigma(Q_1''') \cap \mathcal{I}_\sigma(H_L)} = \emptyset$ has to be satisfied in order that the clock tick $\sigma \in \mathcal{T}$ may occur. By the choice of L this condition implies $\mathcal{I}_\sigma(Q_1''') \subseteq \mathcal{I}_\sigma(P)$, as desired.

Finally, let $P' \mid H_L \stackrel{d_L}{\longrightarrow} P' \mid H_{PQ} \equiv C_{PQ}[P']$. Since $P' \mid H_L \approx_{\mathrm{n}} Q''' \mid H_L$, we have $Q''' \mid H_L \stackrel{\epsilon}{\Longrightarrow} W'$ for some $W' \in \mathcal{P}$. We know that H_L performs its d_L-transition to H_{PQ} since e is a distinguished action. However, Q''' may engage in some τ-transitions to some $Q' \in \mathcal{P}$, i.e. $Q''' \stackrel{\epsilon}{\Longrightarrow} Q'$, and $C_{PQ}[P'] \equiv P' \mid H_{PQ} \approx_{\mathrm{n}} Q' \mid H_{PQ} \equiv C_{PQ}[Q']$.

We have established the existence of processes $Q', Q_1''', Q_2''' \in \mathcal{P}$ such that $Q \stackrel{\epsilon}{\Longrightarrow} Q_1''' \stackrel{\sigma}{\longrightarrow} Q_2''' \stackrel{\epsilon}{\Longrightarrow} Q'$ and $\mathcal{I}_\sigma(Q_1''') \subseteq \mathcal{I}_\sigma(P)$. Also $C_{P'Q'}[P'] \approx_{\mathrm{n}} C_{P'Q'}[Q']$ holds, i.e. $P' \approx_{\mathrm{a}} Q'$, since $C_{PQ}[P'] \approx_{\mathrm{n}} C_{PQ}[Q']$, $\mathcal{S}(P') \subseteq \mathcal{S}(P)$, and $\mathcal{S}(Q') \subseteq \mathcal{S}(Q)$.

Now, we are able to put our proof parts together in order to complete the proof. Proposition 7.6.7 states that $\underline{\approx}$ is the *largest* congruence contained in $\approx$. Moreover, $\approx_{\mathrm{n}}^{+}$ is contained in $\approx$ according to the above argumentation for $\approx_{\mathrm{a}} \subseteq \approx$ and the inclusion $\approx_{\mathrm{n}}^{+} \subseteq \approx_{\mathrm{a}}$. Therefore, we may conclude by Proposition 2.4.16 that $\underline{\approx} = \approx_{\mathrm{n}}^{+}$, i.e. $\underline{\approx}$ is the largest congruence contained in $\approx_{\mathrm{n}}$. □

A byproduct of the above proof is the proof of the second part of Proposition 7.6.3 which states that $\approx$ is the *largest* congruence contained in $\approx_{\mathrm{n}}$ with respect to the sub-algebra of CSA that consists only of prefixing, the static operators, and recursion.

7.6.3 Operational and Logical Characterizations

Similar to temporal strong bisimulation there is a choice as to whether to consider the additional structure of clock scoping as part of a new form of weak bisimulation, or as part of the underlying transition system. More precisely, we can again enrich the clock transitions by scoping information and use the following alternative temporal weak transition relation with respect to clock transitions:

$$P \stackrel{\sigma}{\underset{L}{\Longrightarrow}} P' \text{ if and only if } \exists P'', P''' \in \mathcal{P}.\ P \stackrel{\epsilon}{\Longrightarrow} P'' \stackrel{\sigma}{\underset{L}{\longrightarrow}} P''' \stackrel{\epsilon}{\Longrightarrow} P' ,$$

where $L \subseteq \mathcal{A} \setminus \{\tau\}$.

Definition 7.6.9 (Alternative Temporal Weak Bisimulation)
A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is an alternative temporal weak bisimulation *if for every $\langle P, Q\rangle \in \mathcal{R}$, $\alpha \in \mathcal{A}$, $\sigma \in \mathcal{T}$, and $L \subseteq \mathcal{A} \setminus \{\tau\}$ the following conditions hold.*

1. *$P \stackrel{\hat{\alpha}}{\Longrightarrow} P'$ implies $\exists Q'.\, Q \stackrel{\hat{\alpha}}{\Longrightarrow} Q'$ and $\langle P', Q'\rangle \in \mathcal{R}$, and*
2. *$P \stackrel{\sigma}{\underset{L}{\longrightarrow}} P'$ implies $\exists Q'.\, Q \stackrel{\sigma}{\underset{L}{\Longrightarrow}} Q'$ and $\langle P', Q'\rangle \in \mathcal{R}$.*

We write $P \approx_* Q$ if $\langle P, Q\rangle \in \mathcal{R}$ for some alternative temporal weak bisimulation $\mathcal{R}$.

In fact, this definition yields the same relation as the one introduced in Definition 7.6.2.

Theorem 7.6.10 (Operational Characterization) *The coincidence* $\approx_* = \approx$ *holds.*

The proof of this theorem uses of the following lemma.

Lemma 7.6.11 *Let* $P, P', Q \in \mathcal{P}$ *such that* $P \approx Q$ *and* $P \stackrel{\epsilon}{\Longrightarrow} P'$. *Then there exists some* $Q' \in \mathcal{P}$ *satisfying* $Q \stackrel{\epsilon}{\Longrightarrow} Q'$ *and* $P' \approx Q'$.

The validity of this lemma can easily be checked by induction on the length of the weak ϵ-transition of P. Now, we are able to prove Theorem 7.6.10.

Proof: We first establish the inclusion $\approx_* \subseteq \approx$ by showing that $\approx_*$ is a temporal weak bisimulation. Therefore, let $P, Q \in \mathcal{P}$ satisfying $P \approx_* Q$.

- Let $P \stackrel{\alpha}{\longrightarrow} P'$ for some $P' \in \mathcal{P}$ and $\alpha \in \mathcal{A}$. Then we also have $P \stackrel{\hat{\alpha}}{\Longrightarrow} P'$. Because of $P \approx_* Q$ we know of the existence of some $Q' \in \mathcal{P}$ such that $Q \stackrel{\hat{\alpha}}{\Longrightarrow} Q'$ and $P' \approx_* Q'$.
- Let $P \stackrel{\sigma}{\longrightarrow} P'$ for some $P' \in \mathcal{P}$ and $\sigma \in \mathcal{T}$. By the definition of $\stackrel{\sigma}{\underset{L}{\Longrightarrow}}$ we may conclude $P \stackrel{\sigma}{\underset{L}{\Longrightarrow}} P'$ for $L = \mathcal{U}_\sigma(P)$. Since $P \approx_* Q$ we know of the existence of some $Q' \in \mathcal{P}$ such that $Q \stackrel{\sigma}{\underset{L}{\Longrightarrow}} Q'$ and $P' \approx_* Q'$. Thus, there exist $Q'', Q''' \in \mathcal{P}$ satisfying $Q \stackrel{\epsilon}{\Longrightarrow} Q'' \stackrel{\sigma}{\underset{L}{\longrightarrow}} Q''' \stackrel{\epsilon}{\Longrightarrow} Q'$ which implies $Q'' \stackrel{\sigma}{\longrightarrow} Q'''$ and $\mathcal{U}_\sigma(Q'') \subseteq \mathcal{U}_\sigma(P) = L$, as desired.

Hence, $\approx_*$ is a temporal weak bisimulation, i.e. $\approx_* \subseteq \approx$ by Definition 7.6.2.

The reverse inclusion $\approx \subseteq \approx_*$ is more involved. In the following, we prove that $\approx$ is an alternative temporal weak bisimulation. Let $P, Q \in \mathcal{P}$ such that $P \approx Q$.

- Let $P \stackrel{\hat{\alpha}}{\Longrightarrow} P'$ for some $P' \in \mathcal{P}$ and $\alpha \in \mathcal{A}$, i.e. there exist some $P'', P''' \in \mathcal{P}$ such that $P \stackrel{\epsilon}{\Longrightarrow} P'' \stackrel{\hat{\alpha}}{\longrightarrow} P''' \stackrel{\epsilon}{\Longrightarrow} P'$ by the definition of the temporal weak transition relation. Let us assume $\alpha \not\equiv \tau$ since the other case follows pretty much the same lines but is simpler. Because of $P \approx Q$ and Lemma 7.6.11 we conclude the existence of some $Q'' \in \mathcal{P}$ satisfying $Q \stackrel{\epsilon}{\Longrightarrow} Q''$ and $P'' \approx Q''$. The latter implies $Q'' \stackrel{\hat{\alpha}}{\Longrightarrow} Q'''$ and $P''' \approx Q'''$ for some $Q''' \in \mathcal{P}$. Finally, $P''' \approx Q'''$ and Lemma 7.6.11 implies the existence of some $Q' \in \mathcal{P}$ such that $Q''' \stackrel{\epsilon}{\Longrightarrow} Q'$ and $P' \approx Q'$. Summarizing, we have established the existence of some $Q' \in \mathcal{P}$ satisfying $Q \stackrel{\hat{\alpha}}{\Longrightarrow} Q'$ and $P' \approx Q'$, as desired.
- Let $P \stackrel{\sigma}{\underset{L}{\Longrightarrow}} P'$ for some $P' \in \mathcal{P}$, $\sigma \in \mathcal{T}$, and $L \subseteq \mathcal{A} \setminus \{\tau\}$, i.e. $P \stackrel{\epsilon}{\Longrightarrow} P'' \stackrel{\sigma}{\underset{L}{\longrightarrow}} P''' \stackrel{\epsilon}{\Longrightarrow} P'$ for some $P'', P''' \in \mathcal{P}$ according to the definition of the alternative temporal weak transition relation. Moreover, $P'' \stackrel{\sigma}{\underset{L}{\longrightarrow}} P'''$ implies $P'' \stackrel{\sigma}{\longrightarrow} P'''$ and $\mathcal{U}_\sigma(P'') \subseteq L$. By the premise $P \approx Q$ and Lemma 7.6.11 we may conclude the existence of some $Q'' \in \mathcal{P}$ such that $Q \stackrel{\epsilon}{\Longrightarrow} Q''$ and $P'' \approx Q''$. Because of the latter, there also exist $Q''', Q'''_1, Q'''_2 \in \mathcal{P}$ satisfying $Q'' \stackrel{\epsilon}{\Longrightarrow} Q'''_1 \stackrel{\sigma}{\longrightarrow} Q'''_2 \stackrel{\epsilon}{\Longrightarrow} Q'''$, $\mathcal{U}_\sigma(Q'''_1) \subseteq \mathcal{U}_\sigma(P'') \subseteq L$ and $P''' \approx Q'''$.

Finally, $P''' \approx Q'''$ and Lemma 7.6.11 implies the existence of some $Q' \in \mathcal{P}$ such that $Q''' \stackrel{\epsilon}{\Longrightarrow} Q'$ and $P' \approx Q'$. Thus, we obtain $Q \stackrel{\sigma}{\underset{L}{\Longrightarrow}} Q'$ and $P' \approx Q'$ for some $Q' \in \mathcal{P}$, which finishes this case.

Summarizing we have shown that both relations, $\approx_*$ and $\approx$, coincide. □

Because of the above coincidence we may view $\approx$ as a naive temporal weak bisimulation on an enriched transition system. As for temporal strong bisimulation, the alternative definition suggests how standard partition-refinement algorithms [103, 138] can be adapted in order to compute temporal weak bisimulation.

A logical characterization of $\approx$ can be obtained by modifying the variant of the Hennessy-Milner logic presented in Section 7.5.4. More precisely, one needs to replace the enriched temporal strong transition relation in the definition of the semantics of the $\langle \sigma, L \rangle$-operators by the alternative temporal weak transition relation. The details are routine and, therefore, are omitted here.

7.7 Example (continued)

In this section we apply the semantic theory of temporal observational congruence for reasoning about `Module2` of our example system presented in Sections 7.2 and 7.4. We first define `Module2` formally in our language in accordance with the generic formalizations of registers and function blocks which we have presented so far. Consequently,

$$\texttt{Module2} \stackrel{\text{def}}{=} (\texttt{F} \,|\, \texttt{T} \,|\, \texttt{R}_3 \,|\, \texttt{R}_4) \setminus \{i, f, g, h\}$$

where

$$\begin{array}{lcllcll}
\texttt{R}_3 & \stackrel{\text{def}}{=} & \mu x.\lfloor h.x \rfloor \sigma_2(\texttt{R}'_3) & \text{and} & \texttt{R}'_3 & \stackrel{\text{def}}{=} & \mu y.\overline{i}.x + h.y \\
\texttt{R}_4 & \stackrel{\text{def}}{=} & \mu x.\lfloor e.x \rfloor \sigma_2(\texttt{R}'_4) & \text{and} & \texttt{R}'_4 & \stackrel{\text{def}}{=} & \mu y.\overline{f}.x + e.y \\
\texttt{F} & \stackrel{\text{def}}{=} & \mu x.i.f.\tau.\overline{g}.x & & & & \\
\texttt{T} & \stackrel{\text{def}}{=} & \mu x.g.\overline{h}.\texttt{T}' & \text{and} & \texttt{T}' & \stackrel{\text{def}}{=} & \mu y.\overline{\texttt{out}}.x + \tau.y\,.
\end{array}$$

For simplicity, we have left out channel `busy` since it is not important for the purposes of this section. Recall that `Module2` should intuitively behave as follows. It should always be able to receive a value over channel e. After a clock tick σ_2 `Module2` shall engage in *busy waiting* until it is able to output the computed value via channel `out`. This intuition is formalized by process `Spec2` where the τ-loop in state `Spec2'` models busy waiting.

$$\texttt{Spec2} \stackrel{\text{def}}{=} \mu x.\lfloor e.x \rfloor \sigma_2(\texttt{Spec2'}) \quad \text{where} \quad \texttt{Spec2'} \stackrel{\text{def}}{=} \mu y.\overline{\text{out}}.x + \tau.y + e.y$$

Observe that `Module2` essentially behaves like a register, similar to Registers $\texttt{R}_3$ and $\texttt{R}_4$ but possessing some internal computation modeled by the τ-loop. This kind of register may be called *function register* which on every clock tick σ_2 applies the function associated

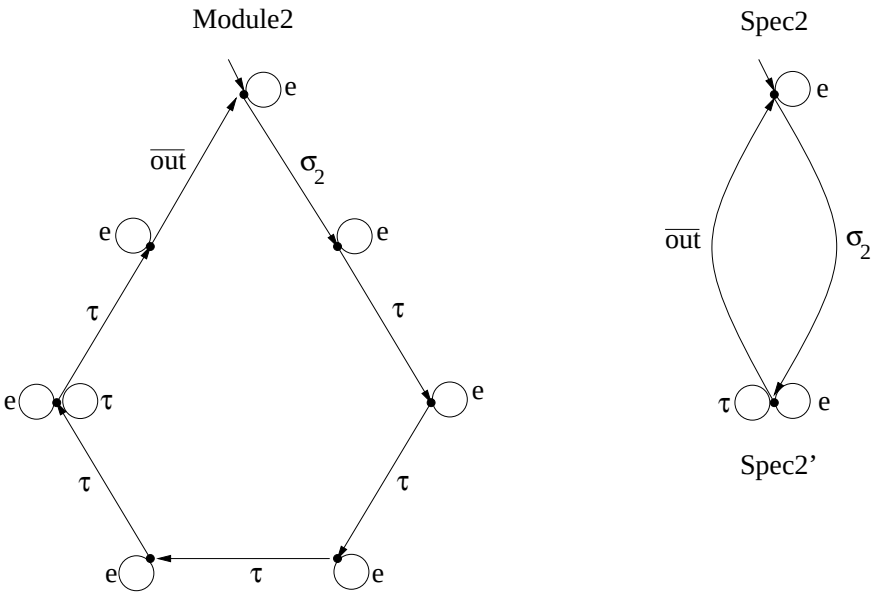

Figure 7.5: Semantics of `Module2` and `Spec2`

with `F` to its current state value and the other inputs of `F`. This typical behavior of function registers can be made even more visible if we would introduce value-passing [67] to CSA, which would enable us to deal with data explicitly. Considering more complex systems built from our generic hardware modules, their behavior can be compositionally reduced to function registers.

Table 7.8: A relation whose symmetric closure is a temporal weak bisimulation

{	$\langle$ `Module2`,	`Spec2` $\rangle$,
	$\langle\ (\mathtt{F} \mid \mathtt{T} \mid \mathtt{R}_3' \mid \mathtt{R}_4') \setminus \{i,f,g,h\}$,	`Spec2'` $\rangle$,
	$\langle\ (f.\tau.\overline{g}.\mathtt{F} \mid \mathtt{T} \mid \mathtt{R}_3 \mid \mathtt{R}_4') \setminus \{i,f,g,h\}$,	`Spec2'` $\rangle$,
	$\langle\ (\tau.\overline{g}.\mathtt{F} \mid \mathtt{T} \mid \mathtt{R}_3 \mid \mathtt{R}_4) \setminus \{i,f,g,h\}$,	`Spec2'` $\rangle$,
	$\langle\ (\overline{g}.\mathtt{F} \mid \mathtt{T} \mid \mathtt{R}_3 \mid \mathtt{R}_4) \setminus \{i,f,g,h\}$,	`Spec2'` $\rangle$,
	$\langle\ (\mathtt{F} \mid \overline{h}.\mathtt{T}' \mid \mathtt{R}_3 \mid \mathtt{R}_4) \setminus \{i,f,g,h\}$,	`Spec2'` $\rangle$,
	$\langle\ (\mathtt{F} \mid \mathtt{T}' \mid \mathtt{R}_3 \mid \mathtt{R}_4) \setminus \{i,f,g,h\}$,	`Spec2'` $\rangle$ }

In order to prove that the implementation `Module2` satisfies its specification `Spec2` we show that both are temporal observational congruent. Therefore, we first state that the symmetric closure of the relation depicted in Table 7.8 is a temporal weak bisimulation relating `Module2` and `Spec2`. This can easily be checked by considering the transition

systems of `Module2` and `Spec2` as depicted in Figure 7.5 where only clock transitions of σ_2 are drawn explicitly. All other clocks in the considered clock universe $\mathcal{T}$ idle in all states which do not possess an outgoing internal transition. Finally, observe that the initial action sets with respect to σ_2 in the initial states of implementation and specification are both equal to $\{e\}$. Since additionally every τ-transition of `Module2` can be matched by a τ-transition of `Spec2`, and vice versa, both processes are also temporal observational congruent according to Definition 7.6.5.

A similar, albeit more involved, reasoning also applies to `Module1` such that the overall system can be built up and analyzed compositionally. Since no new insights are obtained or new techniques need to be applied, the details are omitted.

7.8 Clock-Hiding

We introduce to our language a new family of unary operators, referred to as *clock-hiding* operators, which can be used for explicit clock scoping. We also comment on the effect of the new operators on the semantic theory of CSA. It should be noted that the discussions in this section are tentative and present first ideas only. These ideas are currently investigated further as part of a master's thesis at the University of Passau [107].

Up to now we have only considered the ignore operator for clock scoping. From a semantic point of view, any non-trivial σ-behavior of P is completely ignored in $P \uparrow \sigma$. Moreover, σ self-loops are added to every state of P. The latter is necessary since $P \uparrow \sigma$ should not have any influence on the clock σ since the concept of clocks is external. Instead of using the idea of *ignoring* clocks for clock scoping one can also adopt the view that clocks should be hidden, similarly to action scoping via the restriction operator. This approach is well motivated because one reason for introducing clocks to CCS is to express synchronization constraints. These constraints often need to be added when implementing sequential specifications within *distributed* environments. Hence, for relating these specifications and implementations, one needs to abstract from clocks. Obviously, the ignore operator, as defined in Section 7.3, does not have the desired semantic effect.

To this end, we introduce a family of *clock-hiding* operators $\langle\sigma\rangle$, indexed in $\sigma \in \mathcal{T}$, to CSA. Their operational semantics is defined via the operational rules presented in Table 7.9. Note that Rules (Hid), (tHid1), and (tHid2) correspond to Rules (Ign), (tIgn1), and (tIgn2) for the ignore operator. However, the additional Rule (tHid3) for clock-hiding makes sure that clock transitions of P labeled by σ are not cut off in $P\langle\sigma\rangle$ but relabeled to ι, where ι is a special action not in $\mathcal{A} \cup \mathcal{T}$. This special action, like the internal action τ, is supposed to be *unobservable* but, in contrast to τ, does not possess pre-emptive power since it stems from clock transitions. Moreover, a ι-transition itself cannot be pre-empted since ι stands for an internal clock which is not available for further synchronization. When computing the semantics of processes involving ι the same operational rules as for the special action τ can be applied. Due to the unobservability of ι, relabelings f have to satisfy $f(\iota) = \iota$, and restriction sets may not contain ι, either.

Table 7.9: Operational semantics for clock-hiding in CSA

$$\mathsf{Hid}\quad \frac{P \xrightarrow{\alpha} P'}{P\langle\sigma\rangle \xrightarrow{\alpha} P'\langle\sigma\rangle} \qquad \mathsf{tHid1}\quad \frac{-}{P\langle\sigma\rangle \xrightarrow{\sigma} P\langle\sigma\rangle}$$

$$\mathsf{tHid2}\quad \frac{P \xrightarrow{\sigma'} P'}{P\langle\sigma\rangle \xrightarrow{\sigma'} P'\langle\sigma\rangle}\ \sigma \not\equiv \sigma' \qquad \mathsf{tHid3}\quad \frac{P \xrightarrow{\sigma} P'}{P\langle\sigma\rangle \xrightarrow{\iota} P'\langle\sigma\rangle}$$

Regarding the consequences of introducing clock-hiding for our semantic theory based on the notion of bisimulation, observe that ι-transitions do not need to be parameterized by action sets but can be treated like action transitions since they cannot be pre-empted. It can be shown that the resulting variants of our behavioral relations, temporal strong bisimulation and temporal observational congruence, are still congruences and that they are the largest congruences contained in their naive versions, respectively. Our axiomatization of temporal strong bisimulation can also be extended to reflect clock-hiding. We leave out the details here since they can be found in the above mentioned master's thesis. This thesis is intended to demonstrate the utility of CSA and of clock-hiding by a case study concerned with a *pipelining technique* as used in INTEL's *Pentium processor* [96].

7.9 Discussion and Related Work

In this section we comment on general design decisions of temporal process algebras, relate our algebra CSA to relevant other process-algebraic approaches, and finish off with some remarks about axiomatizations and the role of the clock universe in CSA.

7.9.1 General Design Decisions

Real-time or temporal process algebras [135] can be classified according to (i) whether they deal with a single, global clock or multiple, local clocks, (ii) whether their semantics adopt the maximal progress assumption or not, and (iii) whether they have a quantitative or qualitative view of time. Relevant representatives of the different classes are Hennessy and Regan's TPL [92], Yi's real-time calculus [173], Nicollin and Sifakis' ATP [136], Moller and Toft's TCCS [128], and Andersen and Mendler's PMC [5]. The first four calculi adopt a single, global clock while PMC deals with multiple, local clocks. TCCS and PMC avoid the maximal progress assumption which is employed by the other approaches. Except for TPL semantic theories based on Milner and Parks's notion of bisimulation [125, 139] have been developed. So far, TPL only incorporates a testing semantics [72] which has been investigated and axiomatized. The five approaches are also distinguished by the fact that

ATP, TPL and PMC adopt a qualitative view of time, considering abstract clock ticks, whereas the others choose a quantitative notion, using real or natural numbers as time parameters. We believe that the approaches dealing with time very realistically, by working with numbers, have disadvantages for both the specification and the modeling of real-time systems due to the following reasons. First, for specifying real-time processes exact quantitative timing information must be available. This includes specification-relevant as well as specification-irrelevant timing parameters, though the latter of which one would like to abstract from. Second, exact delays are impossible to implement in practice due to uncontrollable operating conditions, such as temperature, and unavoidable quality variance in fabrication. Chen [45] has attempted to circumvent these problems by proposing an algebra which uses delay intervals instead of exact delays. However, since such delay intervals must be associated with every single action, this algebra turns out to be fairly complicated. Like qualitative time, maximal progress also supports the abstract view of systems for modeling and specification. It reflects our intuition, which is implemented in many distributed systems, that processes are able to wait for communication partners but immediately proceed with the communication when it is possible.

The presented temporal process algebra CSA proposes an abstract approach to the specification and modeling of real-time systems by adapting both a qualitative notion of time and the maximal progress assumption. However, it is even more general and realistic for distributed systems by allowing to deal with *multiple clocks* and by replacing the global notion of maximal progress found in TPL by a local one. In order to be more precise, recall that global maximal progress bundles together the phases of asynchronous cooperation of systems whenever all subcomponents synchronize to let time advance. Hence, the global maximal progress assumption focuses on a type of behavior which may be called *globally-synchronous, locally-asynchronous* and is also the scope of synchronous languages such as ESTEREL [20, 21], where the maximal progress assumption is referred to as *synchrony hypothesis*. In contrast, CSA adopts a more flexible scheme which allows us to localize the notion of time and maximal progress. We can thus obtain simpler and more abstract specifications for distributed systems covering not only globally-synchronous, locally-asynchronous systems but also the class of *globally-asynchronous, locally-synchronous* behavior.

7.9.2 Comparison to other Work

From a technical point of view, CSA extends Hennessy and Regan's TPL with operators from Andersen and Mendler's PMC [5] for managing multiple clocks. Both algebras descend from Nicollin and Sifakis' ATP [136]. As mentioned before, the syntax of CSA is the same as the one of PMC, i.e. it extends Milner's CCS [125] with a *timeout operator* and a clock *ignore operator*. The timeout operator also occurs in other real-time process algebras [5, 92] and was originally introduced in ATP, where it is called *unit-delay*. The ignore operator originates with PMC, though here it is a primitive operation, i.e. it is not derivable as in PMC.

In the following we investigate the semantic relationship of CSA to the process algebras CCS [125], TPL [92], TCCS [128], ATP [136], PMC [5], CSP [94], TCSP [71, 151], and a real-time variant of ACP [12, 16]. First, it is not difficult to see that CSA is a conservative extension of TPL if we drop TPL's *undefined* process Ω, which has been introduced to define a semantics based on *testing*. Restricting the clock universe $\mathcal{T}$ to a single clock, say $\mathcal{T} = \{\sigma\}$, and dropping the ignore operator, gives us precisely the syntax and operational semantics of TPL. Without the ignore operator we obtain $\mathcal{I}_\sigma(P) = \mathcal{I}(P)$, so that the side conditions in Rules (tCom) and (tTO1) degenerate to the global maximal progress assumption of TPL. Note that in this single-clock version of CSA the clock prefixing of TPL can be derived as $\sigma.P \stackrel{\text{def}}{=} \lfloor \mathbf{0} \rfloor \sigma(P)$. Moreover, our results show that CSA is a conservative extension of TPL not only in terms of operational semantics but also in terms of strong and weak bisimulation. This means that our main theorems apply to TPL, too. In particular, specializing Theorem 7.6.8 to the TPL fragment yields a characterization of observational congruence for TPL. Finally, CCS [125] can be identified as the sub-calculus of CSA which is obtained by defining $\mathcal{T} = \emptyset$.

Regarding the relationship to TCCS, ATP, and PMC, which mainly differ from each other by the fact that TCCS and ATP only deal with a single clock, there is an important difference in the way we control when a clock tick is possible and when it is blocked. In TCCS, ATP, and PMC the philosophy is that each subprocess must decide for itself when it is ready to participate in a clock tick and when it is not, which opposes our external view of clocks. In CSA this decision is made, according to maximal progress, indirectly by the presence and absence of communication. The only way to stop a clock is by internal divergence. However, it may be noted that the side conditions in Rules (tCom) and (tTO1), which enforce maximal progress, can be factored out in the following way. If $P \stackrel{\sigma}{\longrightarrow}_* P'$ is the weaker transition relation obtained by dropping the side conditions in Rules (tCom) and (tTO1) and adding the axiom $\tau.P \stackrel{\sigma}{\longrightarrow}_* \tau.P$, then $P \stackrel{\sigma}{\longrightarrow} P'$ if and only if $P \stackrel{\sigma}{\longrightarrow}_* P'$ and $\tau \notin \mathcal{I}_\sigma(P)$. Further, we observe that the transition relation $\stackrel{\sigma}{\longrightarrow}_*$ is precisely the operational semantics of PMC [5] where prefixes and nil are *relaxed* with respect to all clocks, i.e. processes may idle for all clocks in every state. This means, essentially, that CSA can be viewed as the local maximal progress version of PMC with relaxed prefixes. In this precise sense CSA is a specialization of PMC with a built-in semantic connection between clocks and actions. One may also observe that many axioms presented for CSA are identical to the ones given in [5] for PMC. The additional axioms for CSA make the algebra CSA a *quotient algebra* of PMC and hence a semantic abstraction.

In TCCS [128] relaxed behavior is introduced by a nonstandard primitive, the *weak summation* operator. It behaves as the standard summation for ordinary actions, but in contrast forces components to take part in a clock transition only if both argument processes can do a clock tick together. If one of these processes does not admit a clock transition it is considered stopped, in which case the other process can engage in a clock transition all by itself while the stopped process is simply dropped from the weak summation. Weak summation plays an important role in obtaining an expansion law for the axiomatization

of strong bisimulation in TCCS.

As PMC, the temporal process algebra ATP [136] defines only *insistent* prefixing explicitly, where clock ticks are prohibited, but relaxed versions may be expressed by using the *unit-delay* operator, a timeout construct. However, it is surprising that ATP's inaction process may engage in idling. The bisimulation-based semantic theory developed for ATP only focuses on processes that satisfy a certain liveness property which essentially avoids time-stops. Unfortunately, an expansion theorem for ATP fails to hold due to technical subtleties. Therefore, an axiomatization involving parallel composition is only given by introducing auxiliary operators together with some equations which these operators are required to fulfill.

Although clocks synchronize in a broadcasting fashion, they cannot be mimicked in CSP [94]. The reason is that clocks do not only have to synchronize over parallel composition but also over the summation operator, with the consequence that the time determinism property of CSA cannot be simulated in CSP. Moreover, the notion of (local) maximal progress distinguishes the semantic nature of clock transitions in CSA from action transitions in CSP. However, this assumption can explicitly be modeled in a version of CSP having a two-level (distributed) priority-scheme. Unfortunately, up to our knowledge no approach, based on a semantic theory of bisimulation, for investigating priority in CSP exists. Therefore, CSA semantics cannot be encoded in existing variants of CSP semantics. In fact, CSP itself has been extended to cover timing aspects, yielding *Timed CSP* (TCSP) [151], by introducing a global notion of time over a dense time domain as well as new operators expressing delays and timeouts. Semantically, the *failure semantics* of CSP has been carried over to TCSP using a notion of *timed failures* and adapting a kind of maximal progress assumption.

In the synchronous programming language ESTEREL [20, 21] time is related with computation steps in the sense that each step, which in turn consists of several substeps as in *Statecharts* [91], is supposed to consume one time unit (cf. Section 8.2). Thus, the execution model implicitly postulates a single, global clock which indicates the beginning and the end of each step. Semantically, the conceptual clock bundles substeps together to steps. This behavior corresponds to the maximal progress assumption adopted in temporal process algebra. More recent developments regarding ESTEREL focus on implementing dialects of the language within distributed computing environments [22]. Hence, a distributed notion of time implicitly arises. However, up to our knowledge no formal semantics for the distributed approach has been given, yet.

The process algebra ACP [12, 16] is extended with dense time in [9]. More precisely, actions are annotated by time parameters denoting absolute and exact occurrences of time. Semantically, labeled transition systems cover time as part of the state, and the semantic theory is built on the concept of bisimulation. In contrast to CSA, the real-time semantics for versions of ACP obey neither the idling property nor maximal progress.

Finally, we comment on an emerging parallel programming paradigm called *bulk synchronous programming* (BSP) [165], which is based on a high-level synchronization mecha-

nism that corresponds essentially to logical clocks. BSP is proposed as a natural high-level programming model with efficient implementation in hardware. We believe that CSA could be used as a formal process-algebraic model that captures the essence of BSP according to the principle "every bulk one clock."

7.9.3 Remarks on Axiomatization

The axiomatization of temporal strong bisimulation for rs-free CSA processes shows that such processes have a finite number of states under temporal strong bisimulation, which has also been pointed out in [5] with respect to the PMC setting. This is a remarkable fact since rs-free processes in CCS always have a *finite* syntactic unfolding which is in general not true for rs-free processes in CSA. Consider for instance the rs-free process $P \stackrel{\text{def}}{=} \mu x.\lfloor \mathbf{0} \rfloor \sigma(x + (Q\uparrow\sigma))$ taken from [5], where Q is an arbitrary rs-free process. P possesses an infinite unfolding as can be seen by the infinite transition sequence

$$P \xrightarrow{\sigma} P + (Q\uparrow\sigma) \xrightarrow{\sigma} \cdots \xrightarrow{\sigma} P + Q\uparrow\sigma + \cdots + Q\uparrow\sigma \xrightarrow{\sigma} \cdots \ .$$

In contrast to Milner [124] we show completeness only with respect to closed terms and not also for open terms. The problem for extending our approach to open terms is how to deal with unguarded recursion since the standard axiom $\mu x.(x + t) = \mu x.t$, often called Axiom (R3), is not valid in the CSA framework. In order to see this choose $t \equiv \mathbf{0}$ and use Axiom (A4) to conclude $\vdash_r \mu x.(x + \mathbf{0}) = \mu x.x$ and Axiom (R1) for obtaining $\vdash_r \mu x.\mathbf{0} = \mathbf{0}$. Hence, if Axiom (R3) would be valid, then $\mu x.x \simeq \mathbf{0}$ must hold. However, $\mu x.x$ does not possess any transitions while $\mathbf{0}$ can idle for all clocks. Thus, allowing unguardedness enables one to specify processes that disallow any clock tick, i.e. that stop time. In order to deal with those processes in our axiomatization, one could introduce a process Δ to CSA, that is not able to engage in any transition, and re-define Axiom (R3) to $\mu x.(x + t) = \Delta + \mu x.t$. Yet, this does not suffice; an even more elaborate version of this axiom is needed to obtain the desired completeness result. Additionally, the completeness proof would have to make all clock transitions in terms explicit, which we have avoided in Section 7.5.2 since it would require to restrict ourselves to a *finite* clock universe $\mathcal{T}$. Regardless these problems, the presence of the time-stop process Δ would allow us to embed the algebra PMC in CSA since the semantics of Δ coincides with the one of the process *nil* in PMC and PMC's insistent interpretation of the prefix $\alpha.P$ can be encoded as $\alpha.P_\Delta + \Delta$ where P_Δ is the embedding of the PMC process P in CSA.

Finally, we leave an axiomatization of temporal observational congruence (cf. [4]) for future work because of the following reason. Traditional techniques usually employed for proving axiomatizations for observational congruences complete with respect to regular processes rely on the fact that τ-cycles can be eliminated or inserted [126]. Unfortunately, since the internal action τ has pre-emptive power over clocks such a "straightforward" τ-cycle elimination or insertion is unsound in CSA. Thus, providing a complete axiomatization of temporal observational congruence for regular processes in CSA seems to be a very challenging task.

7.9.4 The Impact of the Choice of the Clock Universe

To finish off, we remark on the impact of the choice of the clock universe $\mathcal{T}$ on the semantic theory of CSA. Unlike actions in $\mathcal{A}$, clocks are an external concept, i.e. they originate in the environment rather than in processes. Thus, even if a clock $\sigma \in \mathcal{T}$ is not mentioned explicitly in a term, the corresponding transition system may have σ-loops. Hence, our semantic theory is implicitly parameterized by the clock universe. Consequently, the question arises as to whether $\mathcal{T}$ has to contain all clocks which may occur in a system. If so, $\mathcal{T}$ would have to be a potentially *infinite* universe since systems can be embedded in arbitrary other systems. In practice however, a system designer defines $\mathcal{T}$ to be the *finite* set of all clocks of the system under consideration. If several systems with different universes of clocks are to be composed, the designer might want to extend $\mathcal{T}$ to contain the clock universes of all components. Fortunately, our semantic theory is coherent with respect to this kind of extension of $\mathcal{T}$. In order to make this more precise, we write $\simeq_{\mathcal{T}}$, $\approx_{\mathcal{T}}$, and $\underline{\approx}_{\mathcal{T}}$, explicitly visualizing the clock universe as parameter of our behavioral relations.

Proposition 7.9.1 (Extending the Clock Universe)
Let $\sigma \notin \mathcal{T}$ be a clock and $P, Q \in \mathcal{P}$ such that $\mathcal{C}(P) \subseteq \mathcal{T}$ and $\mathcal{C}(Q) \subseteq \mathcal{T}$. Then

1. *$P \simeq_{\mathcal{T}} Q$ implies $P \simeq_{\mathcal{T}\cup\{\sigma\}} Q$,*
2. *$P \approx_{\mathcal{T}} Q$ implies $P{\uparrow}\sigma \approx_{\mathcal{T}\cup\{\sigma\}} Q{\uparrow}\sigma$, and*
3. *$P \underline{\approx}_{\mathcal{T}} Q$ implies $P{\uparrow}\sigma \underline{\approx}_{\mathcal{T}\cup\{\sigma\}} Q{\uparrow}\sigma$.*

Consider the example of Section 7.7. From the equivalence $\texttt{Module2} \underline{\approx}_{\{\sigma_2\}} \texttt{Spec2}$, which has been established in Section 7.7, we may conclude $\texttt{Module2} \underline{\approx}_{\{\sigma_1,\sigma_2,\rho\}} \texttt{Spec2}$. It is easy to see that the reverse directions of the above statements are also true. Thus, the validity of $P \simeq Q$ is invariant under extension and shrinking of the clock universe $\mathcal{T}$ as long as $\mathcal{T}$ contains all clocks which are explicitly mentioned in the syntax of P or Q. In contrast to the case for temporal strong bisimulation the $\uparrow\sigma$ operators are really needed with respect to temporal weak bisimulation and temporal observational congruence. For example, one may choose $\mathcal{T} =_{\mathrm{df}} \{\rho\}$, where $\rho \not\equiv \sigma$, and consider the processes $P \stackrel{\mathrm{def}}{=} \mathbf{0}$ and $Q \stackrel{\mathrm{def}}{=} (\mu x.\tau.x){\uparrow}\sigma$. According to Definition 7.6.2 we have $P \approx_{\mathcal{T}} Q$ but $P \not\approx_{\mathcal{T}\cup\{\sigma\}} Q$ since P can engage in a σ-transition but Q pre-empts any σ-transition. The same is also true with respect to temporal observational congruence. Hence, one can only add new clocks to $\mathcal{T}$ if those are "ignored."

Finally, we want to note that the choice of $\mathcal{T}$ has an impact neither on the compositionality of the operators of CSA nor on the largest congruence results. Choosing a *finite* clock universe $\mathcal{T} = \{\sigma_1, \ldots, \sigma_n\}$ has the pleasant side effect that we can derive clock prefixing by using recursion and timeout in the following fashion:

$$\sigma_i.P \stackrel{\mathrm{def}}{=} \mu x.\, \lfloor \mathbf{0} \rfloor \sigma_1(x) \cdots \sigma_{i-1}(x) \sigma_i(P) \sigma_{i+1}(x) \cdots \sigma_n(x)\,,$$

i.e. the process $\sigma_i.P$ can engage in a clock transition to P labeled by σ_i and can idle on all other clocks $\sigma \not\equiv \sigma_i$.

7.10 Summary

We have presented the temporal process algebra CSA with multiple clocks and a local maximal progress assumption. CSA is closely related to the process algebras TPL and PMC. Whereas TPL does not deal with multiple clocks, and the semantics of PMC does not ensure maximal progress, CSA combines both features under the special consideration of the distribution of systems. By means of a generic example we have demonstrated the utility of CSA as a semantic framework for modeling synchrony and asynchrony in which one can express various levels of synchronization and time, including independent, hierarchical, and overlapping time scales.

For CSA we have developed an abstract semantic theory based on the notion of bisimulation. The characterizations of the largest congruences in the naive temporal strong and weak bisimulations are of special importance for compositional reasoning, which underlies most of the existing verification techniques. The algebraic flavor of our calculus is made precise by an axiomatization of temporal strong bisimulation for a class of finite-state processes and by alternative characterizations of our behavioral relations, which allow us to adapt standard partition-refinement algorithms for their computation as well as the Hennessy-Milner logic [125] for logical reasoning about processes.

Chapter 8

Discussion

This chapter discusses various notions of pre-emption in traditional process algebras, as developed in the previous chapters, and in existing programming and specification languages. The motivation for introducing pre-emptive constructs for modeling priority and time into these settings is to extend their expressiveness and to improve their notational convenience.

8.1 Concepts of Pre-emption in Process Algebras

Traditionally, process algebras such as CCS [125], CSP [94], or ACP [16, 17] only allow one to model nondeterminism. This dissertation focuses on the most practically-relevant aspects of quantitative and qualitative time which reduce nondeterminism, namely timing constraints and priorities. Whereas in prioritized settings high prioritized transitions are specified to have precedence over lower prioritized ones, in timed settings action transitions may have precedence over clock transitions according to the maximal progress assumption. In contrast to the approaches presented in Chapters 2 and 4, which investigate global pre-emption for reasoning about processes in centralized, uni-processor environments, the previous two chapters have introduced process algebras focusing on distributed systems whose semantics is based on local pre-emption. In the following we consider global and local concepts of pre-emption from various angles.

8.1.1 Principles of Pre-emption

In order to be precise about what it means for a transition to have pre-emptive power, i.e. to prevent other transitions, we need to distinguish between *potential* and *actual* transitions. A transition is called *potential* whenever it offers a communication to the environment, and *actual* if it models a factual communication, thus representing "real" progress. Accordingly, in CCS-based algebras, transitions that are labeled by visible actions are potential whereas transitions labeled by the internal action τ are actual.

The point for developing a semantic theory for pre-emption is that only *actual* high-prioritized transitions (action transitions) can globally or locally pre-empt lower-prioritized

transitions (clock transitions). Giving pre-emptive power also to potential transitions would result in semantically identifying processes which just differ by some lower-prioritized transitions (clock transitions). However, when restricting all transitions having pre-emptive power in some context, the considered processes can semantically be distinguished, i.e. the underlying behavioral relation would not be a congruence. Thus, only those transitions should have pre-emptive power which may not be restricted in any context. As mentioned before, in CCS-like languages these transitions are exactly the ones labeled by τ. When dealing with local pre-emption an additional condition has to be taken care of. Both, the pre-empting and the pre-empted transitions, have to arise from the same state of the system. Note that, whenever we are talking about a localized form of pre-emption, we do not refer by "state" to a *global* state of the underlying distributed system but to a *local* state defined by its parallel subcomponents or by subcomponents which are connected to a common clock. In fact, two transitions can only compete for execution if they both arise from the same local state of the distributed system.

8.1.2 Priority and Communication Schemes

One difference of the algebras $\mathsf{CCS}^{\mathsf{ch}}$, $\mathsf{CCS}^{\mathsf{dp}}$, $\mathsf{CCS}^{\mathsf{prio}}$, $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$, and CSA arises by considering their priority-schemes. These schemes can be distinguished by the number of "priority" levels on which they are built. The process algebras $\mathsf{CCS}^{\mathsf{prio}}$ and CSA have a fixed two-level priority-scheme, where actions are considered to be more important than clocks in CSA. The calculi $\mathsf{CCS}^{\mathsf{ch}}$, $\mathsf{CCS}^{\mathsf{dp}}$, and $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ incorporate multi-level schemes by providing an arbitrary number of priority levels.

Regarding communication schemes, a communication on complementary actions belonging to conceptually different priority levels may take place according to $\mathsf{CCS}^{\mathsf{dp}}$ and $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ semantics. Note that this statement is only implicitly true for $\mathsf{CCS}^{\mathsf{dp}}$ due to its "relaxed" semantics for prefixing. In contrast, $\mathsf{CCS}^{\mathsf{ch}}$ and $\mathsf{CCS}^{\mathsf{prio}}$ only allow communication on complementary actions having the same priority, thereby adopting the view that priorities are part of channels and not explicitly assigned to transitions. The same communication scheme is naturally valid for CSA since communications are allowed on complementary actions whereas synchronizations are enforced on clocks.

During our research we have experienced that the main semantic insights of $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ and CSA concerning a localized view of pre-emption have already been obtained in the simple, convenient framework of $\mathsf{CCS}^{\mathsf{prio}}$ possessing the above mentioned characteristics. Subsequently, the gained knowledge of local pre-emption has been applied for developing semantic theories for $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ and CSA. In the following, we substantiate this observation by working out the unifying semantic concepts of local pre-emption for $\mathsf{CCS}^{\mathsf{prio}}$ and CSA.

8.1.3 Architectural Information

In order to treat qualitative timing aspects for distributed systems, we have developed a *local* view of pre-emption that exploits *architectural information* of the distributed system

under consideration. Architectural information refers to the logical structure of the system and *not* to its physical distribution on a particular hardware platform. The logical structure, which is used for describing the behavior of *parts* of a system, is expressed differently in $\mathsf{CCS}^{\mathsf{prio}}$, respectively $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$, and CSA. Whereas in $\mathsf{CCS}^{\mathsf{prio}}$ architectural information is implicitly encoded in the operational semantics of the calculus and the comparability relation (cf. Definition 6.3.1), by viewing processes on different sides of the parallel composition operator to be logically scheduled on different processors, it is explicitly specified in the syntax of CSA via the ignore operator. By means of generic examples we have illustrated why these two abstract approaches to taking into account distribution are suitable for reasoning about priority and time in distributed systems, respectively.

8.1.4 Pre-emption and Behavioral Relations

The foundation of our semantic theories are behavioral relations that relate processes given as terms of the language under consideration. In order to obtain behavioral relations that are compositional with respect to the operators of the language, one has to take care of all semantic aspects. Consequently, we have to reflect the aspect of pre-emption in our semantic theories for $\mathsf{CCS}^{\mathsf{ch}}$, $\mathsf{CCS}^{\mathsf{dp}}$, $\mathsf{CCS}^{\mathsf{prio}}$, and CSA. These theories are based on the notion of bisimulation [125], which defines an equivalence on processes that is also a congruence for CCS. However, if we straightforwardly adapt strong bisimulation [125] to our process algebras dealing with local pre-emption, we lose this congruence property. Therefore, the challenge is to identify the *largest congruences* contained in the naive adaptations of strong bisimulation for $\mathsf{CCS}^{\mathsf{prio}}$ and CSA. We have shown that this can be done by a careful analysis of the concept of local pre-emption.

The key issue for identifying the relevant information concerning local pre-emption, which we refer to as *local pre-emption potential*, is the observation that potential transitions may turn to actual ones when plugging processes into contexts, e.g. some environment running in parallel. For instance, an environment may accept offers for communication, thereby introducing the internal action τ (cf. Rule (Com3)). Note that a similar argumentation is also true with respect to other languages which may have other mechanisms for turning potential transitions to actual ones. For instance, when integrating aspects of distributed priority and distributed time in CSP-like [94] process algebras involving *broadcast communication*, the switch from potential to actual transitions arises by *hiding* visible actions, i.e. by turning them into the internal action τ. Thus, the principles we have developed for defining bisimulation-based semantic theories dealing with local pre-emption are not restricted to CCS-based process algebras but can be adapted to other popular process-algebraic settings, too.

As suggested above, we need to take care of labels of potential transitions which may turn to τ within some context. Regarding strong bisimulation and *global* pre-emption this is not an issue since every (action) transition is executed from the same, single location (connected to all clocks). Moreover, strongly bisimular processes possess the same initial actions, i.e. their *global pre-emption potentials* are identical. Hence, standard strong bisim-

ulation is a congruence for the algebras $\mathsf{CCS}^{\mathsf{ch}}$ and $\mathsf{CCS}^{\mathsf{dp}}$. According to our semantics for *local* pre-emption, a low-prioritized transition (clock transition) can only be executed if no higher-prioritized transition (action transition) at the same location (connected to the same clock) may engage in a communication. Requiring compositional behavioral relations respecting local pre-emption forces us to consider enriched transition relations. In the following, we make this precise for the calculi $\mathsf{CCS}^{\mathsf{prio}}$ and CSA, for which we have introduced enriched transition relations $\xrightarrow[L]{\alpha}$ for unprioritized actions and $\xrightarrow[L]{\sigma}$ for clocks, respectively. Here, α is an unprioritized action, σ a clock, and L a set of visible (prioritized) actions.

In the following we concentrate on the enriched transition relation for $\mathsf{CCS}^{\mathsf{prio}}$. We have defined $P \xrightarrow[L]{\alpha} P'$ if there exists a location $m \in \mathcal{L}oc$ such that $P \xrightarrow{m,\alpha} P'$ and $\underline{\underline{\Pi}}_{[m]}(P) \subseteq L$ (cf. Section 6.4.3). Intuitively, P may engage in an α-transition at location m to P' whenever the local pre-emption potential $\underline{\underline{\Pi}}_{[m]}(P)$ of P is at most L. Alternatively, one may read $P \xrightarrow[L]{\alpha} P'$ as follows. Whenever the environment refuses to communicate *at least* on the ports contained in L the process P may perform an α-transition to P'; the phrase "at least" is justified since $P \xrightarrow[L]{\alpha} P'$ and $L \subseteq L'$ implies $P \xrightarrow[L']{\alpha} P'$ by definition. The reader may already have observed that our notion of enriched transitions gives rise to an *assumption/commitment*–style for reasoning. Thus, $P \xrightarrow[L]{\alpha} P'$ means that the process P *commits* itself to a transition to P' with action α *assuming* that the *environment* does not want to communicate with any actions from L. A similar intuition can be developed in the timed setting for enriched transitions of the form $P \xrightarrow[L]{\sigma} P'$ where the local pre-emption potential is described by the set $\underline{\Pi}_{\sigma}(P)$, i.e. the set of labels of all action transitions from P that are connected to clock σ. It is worth mentioning that the above observations do not only reflect the coincidence of the prioritized and timed philosophies for local pre-emption in $\mathsf{CCS}^{\mathsf{prio}}$ and CSA but also explain their uniform technical treatment. The semantic analogies are reflected in the abstractness proofs of our behavioral relations, too, where the same idea for the constructions of the contexts for the *largest* congruence proofs is used. The technical framework of enriched transition systems is also the key for developing logical characterizations of our behavioral relations in style of the Hennessy-Milner logic [125] and for applying standard partition-refinement algorithms [103, 138] for their computation.

8.1.5 Pre-emption and Abstraction

Behavioral equivalences that are directly defined on (strong) transition relations are often too fine for reasoning about systems in practice. Usually, specifications are given as simple sequential processes only involving visible actions. Implementations are more complex and often composed of several parallel processes whose interactions are coordinated by internal synchronizations. These bookkeeping synchronizations are invisible for external observers and need to be abstracted away when relating specifications and implementations. Therefore, a notion of *weak bisimulation* or *observational equivalence* is introduced in CCS [125]. This is defined as standard bisimulation on a *weak transition relation* that abstracts from unobservable transitions, namely τ-transitions. Algebraically, observational equivalence has proved to be a congruence with respect to all CCS operators except summation.

When carrying over the CCS procedure for defining (distributed) prioritized and temporal weak bisimulation for $\mathsf{CCS}^{\mathsf{ch}}$, $\mathsf{CCS}^{\mathsf{prio}}$, and CSA, respectively, we have encountered slight differences concerning our global and local views of pre-emption. Whereas the straightforward definition of the temporal weak transition relation (cf. Definition 7.6.2) based on our notion of *enriched* temporal transition relation works fine in CSA, we have experienced difficulties when following the analogous approach for $\mathsf{CCS}^{\mathsf{ch}}$ and $\mathsf{CCS}^{\mathsf{prio}}$, because the resulting (distributed) prioritized weak bisimulation is not compositional with respect to parallel composition (and recursion). The reason concerning parallel composition is that as soon as a process is plugged into an environment, which includes processes running in parallel, high-prioritized transitions located in the process may *delay* lower prioritized transitions that the environment is capable of. Those delays are taken care of by an extra condition in the definition of (distributed) prioritized weak bisimulation (cf. Condition (1) of Definition 2.4.5 and Condition (1) of Definition 6.5.4). Regarding the compositionality problem with respect to recursion, observe that – similar to the reasoning above – an enriched weak transition $P \stackrel{\alpha}{\underset{L}{\Longrightarrow}} P'$ does not cover all necessary aspects of *local* pre-emption. In fact, the parameter L just considers the pre-emptive influence of the environment on the parallel component of P, which may engage in the α-transition. However, symmetrically P itself exerts a pre-emptive impact on its environment. In order to obtain a distributed prioritized weak bisimulation which is compositional with respect to all operators of $\mathsf{CCS}^{\mathsf{prio}}$ except (distributed) summation, an additional parameter, $\underline{\underline{\Pi}}(P)$, capturing this sort of pre-emption potential of P is needed in the definition of the distributed prioritized weak transition relation. Summarizing, in the distributed priority framework we have two different views of pre-emption and, accordingly, two kinds of pre-emption potential that are made explicit as two separate parameters of the distributed prioritized weak transition relation for unprioritized actions. The first kind of pre-emption potential is *local* to P and reflected by some distributed prioritized initial action set $\underline{\underline{\Pi}}_{[m]}(P)$, where m is an appropriate location. The second is *global* to P and given by the global prioritized initial action set $\underline{\underline{\Pi}}(P)$.

One may wonder why the latter technical difficulty does not arise for (distributed) prioritized strong bisimulation and for the algebras $\mathsf{CCS}^{\mathsf{ch}}$ and CSA. Regarding (distributed) prioritized strong bisimulation, we observe that only those processes are related which have identical (distributed) prioritized initial action sets. Thus, it is implicitly taken care of global pre-emption potentials. For CSA the reason is that a clock tick may only be inhibited by a communication whenever *both* communication partners are attached to that clock. Similarly, in $\mathsf{CCS}^{\mathsf{ch}}$ all communication partners reside in the same, single location. In contrast, an unprioritized transition from some location m gets pre-empted in $\mathsf{CCS}^{\mathsf{prio}}$ whenever there exists a prioritized communication where *at least one* communication partner is located at a comparable location of m.

The technical subtleties are reversed when defining observational congruences, i.e. the largest congruences contained in the (distributed) prioritized and temporal weak bisimulations, respectively. Whereas the traditional summation fix for CCS works just fine for

$\mathsf{CCS}^{\mathsf{ch}}$ and $\mathsf{CCS}^{\mathsf{prio}}$ when obeying pre-emption potentials (cf. Sections 2.4.2 and 6.5.2), it does not for CSA (cf. Section 7.6.2). This is due to the special nature of nondeterministic choices in timed settings in which the summation operator behaves differently with respect to actions and clocks. On the one hand, it is *dynamic* for actions and resolves choices. On the other hand, it is *static* for clocks since clock ticks require global synchronization. Therefore, the definition of temporal observational congruence is more involved than the one of (distributed) prioritized observational congruence.

To close this section we comment on abstractions from clocks. Similar to auxiliary communications, synchronization constraints modeled by clocks often need to be added when implementing (sequential) specifications within distributed environments. Hence, beside abstracting from the internal action τ, one would also like to abstract from these clocks in order to be able to relate specifications and implementations by means of temporal weak bisimulation or temporal observational congruence. Since the ignore operator, which is responsible for clock scoping, does not support any form of abstraction we have proposed to insert a separate *clock-hiding* operator in CSA. Fortunately, this new operator is consistent with our semantic theory for CSA and does not require any special attention.

8.2 Concepts of Pre-emption in Existing Languages

In this section we shed some light on the practical relevance of the aspects of priority and time and the underlying concepts of pre-emption by considering the relevant mechanisms in existing programming and specification languages that are successfully employed by engineers. More specifically, we focus on the languages *ADA* [105], `occam` [95], *SDL* [99, 162], *E-LOTOS* [98], *Statecharts* [91], *ESTEREL* [21, 34], *LUSTRE* [88], and *SIGNAL* [87] which can be categorized as follows: (1) ADA and `occam` are variants of imperative, parallel programming languages with no precise formal semantics. (2) SDL, E-LOTOS, and Statecharts are specification languages for which formal semantics have been defined. (3) ESTEREL, LUSTRE, and SIGNAL are mathematically well-founded, *synchronous* programming languages having a very different semantic character from representatives of the first two categories and the process algebras developed in this dissertation. They are only presented here in order to complete the picture of languages incorporating priority and timing aspects.

Unfortunately, many of these languages do not come with a formal semantics, and their reference manuals only provide a vague intuition about the semantic aspects we are concerned with. Consequently, our aim is to stress the necessity for developing semantic theories of pre-emption, not to present a complete comparison of the concepts of priority and time advocated in this dissertation with the ones implemented in existing languages. In fact, the present dissertation should be considered as basic research on the semantics of pre-emption that establish a basis upon which semantic theories for existing and new languages can be founded. Most languages dealt with in this section either provide no such semantic theories at all, or the existing theories are not satisfying since they do not fit together elegantly with other important concepts. Therefore, it is often impossible to

classify semantic properties concerning priority and time according to the lines of this dissertation. However, we try to give the reader some intuition about the utility of the languages under consideration and their implemented constructs for expressing priority and time. Our main interest is in the specification language *Statecharts*, because of its success in industrial applications. Although E-LOTOS, which is a newly revised version of LOTOS [30, 97], has been widely inspired by process algebras, we have not been able to examine precisely its concepts of priority and time. Unfortunately, E-LOTOS is still under development, and documents adequately presenting its formal semantics concerning pre-emption are missing.

8.2.1 Imperative Parallel Programming Languages

This section is concerned with the languages ADA and `occam` belonging to the first category of the considered languages.

ADA

ADA is an imperative, parallel, object-oriented, high-level programming language [105] which is internationally standardized (ISO/IEC 8652). The high complexity of ADA stems from combining features of various imperative programming languages. ADA also provides a mechanism for running processes, called *tasks*, in parallel. Similarly to the process algebra CCS, its communication principle is synchronous and handshake, i.e. a handshaking mechanism which in ADA is also referred to as *rendezvous*. Priorities can be expressed via a variety of programming constructs. One may define *interrupts*, along a multi-level priority-scheme, and corresponding interrupt handlers, which specify the program code that is executed whenever an interrupt is invoked. Just like in the second part of this thesis, ADA adopts a distributed view of priority since a multi-processor system may have more than one interrupt subsystem (cf. Chapter 6). Another similarity to our process algebras $\mathsf{CCS}^{\mathsf{ch}}$, $\mathsf{CCS}^{\mathsf{prio}}$, and $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ is that the priority of the execution of an interrupt handler is allowed to vary from one system state to another, i.e. priorities are *globally dynamic* in the sense of [154].

Regarding *qualitative* aspects of time, ADA provides a powerful mechanism for task scheduling. It works by defining *task priorities* where an integer value indicates the degree of urgency of a task. Task priority is a basis for resolving competing demands for system resources. ADA is a good example confirming our view adopted in Chapter 4 that priorities dealing with scheduling in real-time have a global and dynamic character. Indeed global pre-emption information is extracted out of task priorities which may be re-assigned as the system evolves and, thus, are dynamic. This *user-guided* task of re-assigning priority values distinguishes dynamic-priority in ADA from dynamic-priority in $\mathsf{CCS}^{\mathsf{dp}}$ (cf. Chapter 4). The global pre-emption mechanism for task scheduling is implemented by *priority-ordered ready queues* together with concepts of *pre-emptable system resources* and *pre-emptive abort*. These realizations are typical for many languages dealing with real-time scheduling.

Additionally, ADA allows one to specify aspects of *quantitative* time since it provides a *clock package* together with delay and timeout statements. Here, *time* is measured with respect to the physical processor on which the considered task is running and, therefore, has a distributed character (cf. Chapter 7). Finally, the language ADA also includes mechanisms for synchronous and asynchronous task control, which would lead too far to be discussed here.

`occam`

The process-oriented, asynchronous, high-level programming language `occam` is based on the principle of sequential communicating processes [95]. Usually, `occam` programs are physically mapped onto *transputers*, a hardware platform for the parallel execution of processes. The concept of communication in `occam` is similar to CCS and ADA, i.e. a synchronous handshake communication. Consequently, each communication channel connects two ports, an *input* and an *output port*.

As already mentioned in Chapter 6, the `occam` language contains a specific operator, called `PRIALT`, which allows one to express *prioritized choices*. This operator is typically used as follows

$$\texttt{PRIALT}\ \ g_1\, P_1 \mid g_2\, P_2 \mid \ \ldots\ \mid g_n\, P_n$$

where $P_1, P_2, \ldots, P_n$ are processes and $g_1, g_2, \ldots, g_n$ are *guards*, usually of the form $c?$, that encode waiting for an input on port c. The guard $c?$ is satisfied whenever a process running in parallel is ready to communicate on channel c. Semantically, the above process behaves as the process P_i, where $1 \leq i \leq n$, provided that i is the smallest index such that guard g_i is satisfied. Hence, `PRIALT` is a choice operator which resolves nondeterminism. In a process-algebraic framework, involving multi-level, distributed priorities, this operator has been investigated by Camilleri and Winskel [43]. Since we have shown that our process algebra $\mathsf{CCS}_{\mathsf{pp}}^{\mathsf{prio}}$ is more general than their approach, we have already covered the semantic aspects of `PRIALT` in Chapter 6.

However, it is important to emphasize that guards in `PRIALT` constructs may only contain input ports but *not* output ports. This restriction greatly simplifies the implementation of `occam` on transputer platforms by avoiding the need to implement technically complex communication protocols. Observe that a similar restriction is responsible for the algebraic tractability of the operational semantics for $\mathsf{CCS}_{\mathsf{pp}}^{\mathsf{prio}}$ (cf. Section 6.7.1). This again stresses the importance of the mentioned restriction in order to obtain a semantics that is efficiently implementable.

Regarding timing aspects, `occam` provides a possibility to influence the scheduling of processes via the construct `PRIPAR`, a prioritized version of parallel composition. Its semantics is defined in an analogous fashion to that of `PRIALT`, i.e. it prefers to execute the argument process which comes first in the program text. However, processes which have to wait for a communication partner are not considered for selection. Jensen [102] has extensively considered `PRIPAR` in a process-algebraic context. Also the calculus $\mathsf{CCS}_{\mathsf{pp}}^{\mathsf{prio}}$ can

easily be extended by this kind of operator without introducing new difficulties or obtaining new insights concerning the theory of local pre-emption. In fact, it turns out that the semantic theory developed in Chapter 6 can be adapted straightforwardly by just changing the definition of the comparability relation (cf. Definition 6.3.1).

In order to implement applications that react on time-dependent events, `occam` includes a special construct, called `AFTER`, that allows one to read the value of a *local clock*, where "local" refers to the specific physical processor on which the underlying process is running. Therefore, similar to ADA, time has a distributed character although the distribution concept is not as general as for CSA since one may not express overlapping and nested scales of time (cf. Chapter 7). However, the `AFTER` construct can be used together with *timer channels* for expressing delays. For instance, one may write *clock ? **AFTER** v* where *clock* is a timer channel and v an integer value. Semantically, this expression is a process which does not terminate unless the value of the *clock* is greater than v. Summarizing, `occam` possesses all important features for dealing with priority and real-time. Moreover, those features have semantics that can nicely be reflected in process-algebraic frameworks.

8.2.2 Specification Languages

In the following, we deal with members of the second category of languages, namely the specification languages SDL, E-LOTOS, and Statecharts.

SDL

The abbreviation SDL stands for *Specification and Description Language* [99, 162]. This language is internationally standardized and widely used by engineers for specifying and implementing software in a tool-supported, graphical and textual manner. It contains many rich constructs of parallel, imperative programming languages. However, SDL does not provide a mechanism for modeling interrupts or similar kinds of priority, but it possesses a concept of *timeout*.

The asynchronous programming model underlying SDL is a variant of other models for parallel, imperative languages. It is built of three hierarchical layers with asynchronous concepts of handshake and broadcast communication via buffered channels, in contrast to synchronous communication adopted for the process algebras dealt with in this thesis. The lowest layer contains *process instances* which correspond to finite-state machines together with a local memory and an input queue. Process instances may perform actions such as assignments and procedure calls. Several process instances can be grouped, i.e. composed in parallel, to *blocks*. Within blocks they may instantaneously exchange data and names, e.g. process identifiers, via *signal routes*. Furthermore, blocks may be combined to *systems* and connected by *channels*. For a basic variant of SDL, called Basic SDL, a formal operational semantics has been defined using structured operational rules [85].

In order to deal with *timeouts*, every SDL system implicitly contains a single, global clock which measures time. In each process instance one may set arbitrarily many timers

together with an expiration value and a name of a signal which is invoked when the timer expires. At first sight, one may think that this timeout concept permits one to deal with quantitative, global time in a very general fashion. However, the bottom line is that the signal generated by an expired timer does not interfere the current execution of the process instance since it is only appended to the input queue and processed at some undetermined time later. In other words, SDL does not provide a concept of pre-emption with respect to time. Consequently, time is a very subtle, unintuitive concept in SDL which does not allow a similarly clean mathematical treatment of time as process-algebraic approaches do. Therefore, the gap between the concepts of time in SDL and the ones advocated in this dissertation (see Chapters 4 and 7) is too wide in order to allow us a fair comparison. In fact, SDL is a good example for demonstrating that specification languages successfully used in industrial contexts do exist that lack any conceptually view of pre-emption, although they claim to provide such features as timeouts.

E-LOTOS

E-LOTOS [98] (*Enhanced Language of Temporal Ordering Specifications*) is a formal specification language for specifying distributed systems and currently on its way for standardization. As its predecessor LOTOS [30, 97, 162] (ISO standard 8807), E-LOTOS is based on traditional process algebras such as CCS [125] and CSP [94], and on an algebraic description language for describing abstract data types. The successful history of LOTOS has shown that a specification language developed on process-algebraic ideas can play an important role even in industrial contexts. Thus, studying process algebras is not a waste of academic resources but basic research in language design.

(E-)LOTOS is given a behavioral semantics in terms of labeled transition systems via structural operational rules. The semantics of parallel composition is based on interleaving, and broadcast communication is achieved as in CSP via simultaneous synchronization on common ports. However, there exists also a mechanism for asynchronous communication via buffered channels. The old standard LOTOS does not provide any mean for specifying priority or timing aspects in the spirit of this thesis. It only includes a so-called *disabling* operator that can be used for specifying interrupt handlers. However, this operator does not have a notion of pre-emption but is solely based on the concept of nondeterminism which makes it unsuitable for dealing with interrupts adequately. Disabling can better be used for defining a notion of *state refinement* as has been done for Statecharts (see next section). However, people have realized that priority and time (cf. [111, 147]) are important concepts for specification languages and, hence, these concepts can be found in the draft for the recently proposed E-LOTOS standard [98].

E-LOTOS allows one to describe behaviors or expressions raising *exceptions* which may be trapped by *exception handlers*. Technically, this is implemented via *urgent actions* [31] which include exception raising actions as well as *internal* and *termination actions*. The urgent semantics of exceptions given in E-LOTOS is basically the same as the *signal* model of Timed CSP [70]. Intuitively, a behavior is urgent if it cannot delay. Additionally,

E-LOTOS also includes a mechanism of *suspend/resume* by extending the LOTOS disabling together with a new *raise* construct allowing the resumption of interrupted behavior. It is worth noting that behavior is blocked whenever it is suspended, i.e. it does *not* evolve in time or actions. Whereas the concept of exception handling vaguely corresponds to our approach to priority, the process algebra $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ possesses no constructs to model the suspend/resume facilities of E-LOTOS directly. However, from a theoretical point of view $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ is expressive enough to encode stacks, since stacks can also be modeled in plain CCS, and to mimic the suspend/resume mechanism, because it incorporates a concept of pre-emption.

Finally, E-LOTOS includes a notion of *quantitative* time, based on real-time extensions of LOTOS [114], thereby allowing one to give precise descriptions of real-time systems. Technically, communications can be made sensitive to time by adding an $@@t$ annotation to the communication action which binds the variable t to the time at which the communication occurs. This time is measured relatively from when the communication has been offered first. Moreover, the E-LOTOS statement `wait`(k) enables the specification of a delay by k time units. These concepts have already been introduced by Yi [173] in his real-time process algebra, which obeys maximal progress similar to $\mathsf{CCS}^{\mathsf{rt}}$ (see Chapter 4). Consequently, the behavioral semantics in form of labeled transition systems also follows Yi's approach. Hence the transition relation is split into two parts by distinguishing between action and clock transitions as has been done for $\mathsf{CCS}^{\mathsf{rt}}$ and CSA. However, in contrast to CSA but similar to $\mathsf{CCS}^{\mathsf{rt}}$, the notion of time in E-LOTOS is quantitative and global, instead of qualitative and local.

Statecharts

The *graphical* specification language *Statecharts* extends *communicating finite automata* by the concepts of *hierarchy* and *priority*. Statecharts have been introduced by Harel about a decade ago [91] for modeling and reasoning about reactive systems. Since then this language has attracted the attention of many software engineers who appreciate its graphical nature. Consequently, a lot of different dialects and specification tools supporting these dialects have been developed [14, 78]. However, the semantic foundations of most variants have some disturbing shortcomings, especially concerning compositionality [47, 117]. This is mainly due to the complicated semantics of parallel composition, based on a kind of asynchronous broadcast communication, and the chosen approach to priority.

Before we discuss how priorities can be expressed in Statecharts, we remark on the semantic deficiencies concerning parallel composition. This also leads us to discuss a special qualitative timing constraint implicitly present in Statecharts which is closely related to the maximal progress assumption adopted by process-algebraic approaches to time (cf. Chapters 4 and 7). First, we need to introduce the form of labels used in the syntax and semantics of Statecharts. Statechart labels are split into two parts, a trigger part and an action part, which both consist of a conjunction of *events*, as port names are called within the framework of Statecharts. A statechart may engage in a transition, also referred to as

micro step, whenever all events of the trigger are provided by the environment, thereby generating the events of the output part of the action. These generated events may cause the trigger of a transition in a different parallel component to be satisfied. Thus, this transition is now allowed to be executed, i.e. probably new events are generated, and this kind of "chain reaction" may continue until possibly all parallel components have taken part in this system step, which in Statecharts is called *macro step*.

In order for this semantics to be well-defined, a *synchrony hypothesis* is introduced which ensures that the considered system is infinitely faster than its environment. More precisely, a chain reaction, initialized by some events of the environment, is guaranteed to be completed before the environment produces the next initial trigger events. Thus, the synchrony hypothesis allows one to differentiate between micro and macro steps. More precisely, it is used for determining the start and end points of macro steps, thereby introducing implicit synchronizations which could also be encoded via a global clock in temporal process algebras. Except the fact that Statecharts only allow a parallel component to take part *at most once* in every macro step, the synchrony hypothesis corresponds one-to-one to the maximal progress assumption of traditional temporal process algebras where clock ticks may only occur if no further communication is possible within the system.

Unfortunately, the two-level semantics of Statecharts, which is reflected by two different *atomicity levels* (micro and macro steps), gives rise to problems concerning *compositionality*, which is a mandatory requirement for any reasonable algebraic framework. In many process-algebraic approaches, including the ones presented in this thesis, the problems experienced in Statecharts are avoided by choosing a single, fine-granular atomicity level and a semantics for parallel composition that does not introduce subtle issues of causality. As those details are not central for this thesis, we do not consider them further but refer the reader to [117, 164].

In the following, let us present the priority mechanism, which is adopted by many variants of Statecharts, including the original one [91]. Intuitively, in Statecharts priorities can be expressed via the absence of events. Technically, this is achieved by permitting negated events as part of triggers (see also [84]). For example, consider the following term-based notation of a simple statechart: $a : b.P + \neg b : c.Q$. This term consists of a nondeterministic choice between a b-transition with trigger a to process P, and a c-transition with trigger $\neg b$ to Q. Intuitively, a statechart may only engage in the latter transition if it cannot execute the former one since this produces the event b which falsifies the trigger of the c-transition. Thus, the b-event is given precedence over the c-event. In the following we investigate how this approach to priority goes along with the concept of *hiding* which is used in a very popular variant of Statecharts, called ARGOS [119, 120].

Hiding enables one to relabel a visible action into a distinguished invisible action (cf. the internal action τ in CCS). Unfortunately, the approach to priority via negated events does not fit to the concept of hiding. The reason is that negated events allow one to specify different priority values for different events. However, when several events are hidden, all are relabeled to the same event and, thus, have the same priority value. In other words,

hiding destroys the priority structure of a statechart, i.e. the concepts of priority and hiding are not orthogonal in ARGOS. However, orthogonality is mandatory for obtaining mathematically tractable semantics. In our process-algebraic approach to priority this problem does not occur since we do not use port names to express priority-schemes, but we introduce priorities to transitions by a more fine-granular mechanism, namely by assigning explicit priority values, e.g. natural numbers, to transitions (cf. Chapters 2, 4, and 6).

Summarizing, the existing semantics of Statecharts is not satisfactory from an algebraic point of view. Especially, the two-level micro/macro-step approach to parallel composition and the priority concept via negated events are not semantically elegant and do not harmonize with other concepts, such as hiding, which are desired in specification languages.

Several approaches have been suggested for introducing time in Statecharts (e.g. [140]). We concentrate on a particular well-known variant, called *Timed Statecharts*, developed by Kesten and Pnueli [106]. They introduce time in the sense of process-algebraic real-time, i.e. in a global and quantitative way which is similar to the one adopted for $\mathsf{CCS}^{\mathsf{rt}}$ (cf. Section 4.3) but which opposes our local, qualitative view of time advocated by CSA (cf. Chapter 7). Formally, Timed Statecharts rely on a second variant of triggers which consists of a time interval that specifies lower and upper time bounds on when a transition may be taken in a step. In Timed Statecharts the notions of micro/macro-steps do not any longer play such an important role as in "plain" Statecharts. Any generated event continues to exist as long as time does not progress. In particular, the reaction to a generation of an event may happen some (macro) steps later. Possibly, a parallel component may take part several times in that reaction, provided it is still within the same time-stamp. Thus, the introduction of time in Statecharts completely changes its semantic theory, thereby justifying our criticism of the semantics of Statecharts.

More precisely, in contrast to usual Statecharts, Timed Statecharts have a uniform *asynchronous* semantics as labeled transition systems which are defined via structural operational rules. As for real-time process algebras, these transition systems possess a timed and an untimed transition relation, where timed transitions are synchronous in the sense that they require the cooperation of all concurrent processes of the considered system. Moreover, similar to $\mathsf{CCS}^{\mathsf{rt}}$ and CSA, a notion of *maximal progress* is imposed on Timed Statecharts, i.e. time cannot progress until all transitions that can respond to a generated event have done so. Consequently, from a semantic point of view, Timed Statecharts are closer related to process algebras than "plain" Statecharts.

8.2.3 Synchronous Programming Languages

Whereas ADA, `occam`, SDL, and E-LOTOS are *asynchronous* languages and Statecharts is asynchronous on the micro-step but synchronous on the macro-step level, we now consider the languages ESTEREL [21, 34], LUSTRE [88], and SIGNAL [87], which are based on the principle of *synchrony* [15]. Consequently, they follow a very different philosophy than those process algebras that we have developed in this dissertation, also concerning semantic aspects of priority and time. Therefore, it is not the purpose of this section to relate the

semantic concepts of synchronous languages with those of asynchronous process algebras but to complete the picture of programming languages that involve aspects of pre-emption. Moreover, we want to found our point of view presented in Chapter 7, that the principles synchrony and asynchrony should be uniformly combined in programming languages.

Synchronous languages are suitable for modeling and reasoning about reactive systems under the assumption that the system is fast enough to respond to each input of the environment before another input arrives. ESTEREL, LUSTRE, and SIGNAL obey this assumption, which is referred to as *synchrony hypothesis*, and has also been adopted in Statecharts, as already seen in the previous section. At first sight, the synchrony hypothesis permits an easier mathematical treatment of languages. This is especially true with respect to timing information since each synchronous step is assumed to consume exactly one time unit. However, at second sight, the synchrony hypothesis also introduces *causality problems* which can only be detected via sophisticated algorithms during the compilation of programs to *finite-state machines* or *equation systems*. This stresses the importance of developing variants of asynchronous languages which have the capability to express synchronous behavior, too, thereby remedying the shortcomings of pure synchronous languages. Therefore, we have introduced the asynchronous process algebra CSA which provides a very general mean to specify global *and* local synchronization constraints (cf. Chapter 7). Another important semantic aspect of ESTEREL, LUSTRE, and SIGNAL, beside synchrony, is *determinism*. In fact, nondeterminism is completely thrown out of these languages in order to enable an easy implementation of programs. Now, let us take a closer look at ESTEREL, LUSTRE, and SIGNAL.

ESTEREL

ESTEREL is a reactive model of synchronous parallel systems with instantaneous broadcasting of signals which may be emitted, tested for presence, and which may have a value with them. ESTEREL programs, having both a textual and a graphical syntax, are given an operational semantics as labeled transition systems and are compiled into finite-state sequential machines. ESTEREL divides the life of a reactive system into *instants*, i.e. moments where the system reacts, and, therefore, implicitly assumes an associated global clock. Moreover, ESTEREL provides an interrupt mechanism via the *watching statement* that has the form "`do` *body* `watching` *signal*." Intuitively, the *body* is killed as soon as the *signal* watched for becomes present. By nesting watching statements, one can express multi-level interrupts within this framework (cf. Chapters 2 and 6). Since the handling of watching statements implicitly involves negation, the corresponding comments for Statecharts regarding negated events are also valid for ESTEREL.

The concept of pre-emption has also been investigated by Berry [19] within ESTEREL's *zero-delay process calculus* which is a theoretical version of ESTEREL. Berry emphasizes the importance of pre-emption in control-dominated reactive and real-time programming. He suggests pre-emption operators to be considered as first-class operators which are fully orthogonal with respect to all other primitives such as concurrency and communication.

Several examples of useful pre-emption operators are presented and axiomatized in [19], all of which are based on the ideas of *abortion* and *suspension*.

LUSTRE

LUSTRE is a synchronous *data flow* language which allows one to deal with parallelism and timing constraints within a mathematical, functional model that is based on *equations* and closely related to temporal logics. Similar to process algebras, this enables one to express specifications and implementations within the same language. More precisely, a *flow* is a pair consisting of a sequence of values and a clock, i.e. a sequence of times. Thus, functional and timing behaviors are *explicitly* combined. This is in contrast to the maximal progress assumption adapted by $\mathsf{CCS}^{\mathsf{rt}}$ and CSA (cf. Chapters 4 and 7, respectively) that *implicitly* relates functional behavior and clock ticks. For convenience, LUSTRE allows one to define several granularities of the underlying, global, quantitative clock, thereby introducing a kind of multiple clocks. However, those clocks are still mutually dependent and interact synchronously, whereas in CSA different clocks are independent and interact asynchronously. Priority may be expressed in LUSTRE using a kind of *watchdog* construct that provides a mechanism for modeling timeout and, therefore, a concept of pre-emption.

SIGNAL

SIGNAL is a block-diagram oriented language for programming in *constraints*. Its semantics is defined by a mathematical model of multiple-clocked flows of data and events. The SIGNAL compiler performs equational, formal calculations on synchronizations and other dependencies in order to check the correctness of programs, e.g. for guaranteeing determinism, and to produce executable code. Clocks in SIGNAL have a similar flavor than clocks in CSA since they are a technical mean for encoding synchronization constraints. Intuitively, clocks may be considered as equivalence classes of signals that are always guaranteed to occur simultaneously. In SIGNAL, priority cannot be expressed directly but some kinds of priority may be encoded via clock constraints. Regarding time, *delays* are an elementary concept in SIGNAL and enable one to deal with various timing aspects.

Summarizing, the languages ESTEREL, LUSTRE, and SIGNAL are based on the concepts of synchrony and determinism. This distinguishes them from our process algebras which follow the principles of asynchrony and (reduced) nondeterminism. Therefore, the aspects of (distributed) priority and time have a different quality for the languages considered above which makes a fair comparison to the work presented in this dissertation difficult.

Chapter 9

Conclusions and Future Work

We conclude this dissertation by summarizing its main contributions, critically remarking on the design decisions and the technical developments of the presented semantic theories, and providing starting points for future research.

9.1 Contributions of this Dissertation

This dissertation has investigated concepts of pre-emption in process algebras with priority and time. Considering these practically-relevant aspects of system behavior enhances the expressiveness of traditional process algebras and, thereby, makes them more attractive to system designers as pointed out by several case studies and examples in the previous chapters. We first have presented a version of an existing approach to priority in process algebras, $\mathsf{CCS}^{\mathsf{ch}}$, which has been altered here in order to enhance the applicability of its semantic theory involving *global* pre-emption. The algebra's utility for modeling and verification has been proved by several examples and a medium-scale case study dealing with part of a safety-critical railway signaling system. Moreover, we have demonstrated how priority and time are related by developing a calculus with dynamic-priority, $\mathsf{CCS}^{\mathsf{dp}}$, which is suitable for efficiently encoding real-time constraints as shown by formally modeling and reasoning about the `SCSI-2` bus-protocol. This dissertation has also presented a new process-algebraic approach to developing semantic theories for distributed priority and distributed time based on the idea of *localizing pre-emption* which is suitable for modeling and verifying *distributed* systems. More precisely, we have extended the well-studied process algebra CCS [125] by mechanisms for expressing qualitative timing aspects, namely priority and synchronization constraints. The developed process algebras $\mathsf{CCS}^{\mathsf{prio}}$ and CSA, dealing with distributed priority and distributed time, respectively, do not only remedy shortcomings of existing theories but also present a basis for specification and programming languages that are implementable within distributed environments. The attractiveness of the calculi introduced in this thesis stems from their well-founded mathematical semantics based on the notion of bisimulation [125, 139], which do not sacrifice the simplicity and the elegance that have made traditional process-algebraic approaches successful. In the

following, we summarize the contributions of this dissertation in more detail.

In Chapter 2 we have adapted a version of a classical CCS-based approach, called $\mathsf{CCS}^{\mathsf{ch}}$, to priority in process algebras. In particular, this version includes a *multi-level* priority-scheme which is needed for modeling many practically relevant systems. Moreover, leaving out the prioritization and deprioritization operators allows us to develop a coarser notion of prioritized observational congruence (cf. Definition 2.4.10), a bisimulation that abstracts from internal computation. Accordingly, the semantic theory for observational congruence in $\mathsf{CCS}^{\mathsf{ch}}$ has been re-developed. First, prioritized observational congruence has been characterized as the largest congruence contained in the naive adaptation of standard weak bisimulation (see Theorem 2.4.11). Such an abstractness result indicates that our behavioral relation is semantically adequate and useful for formally reasoning about concurrent systems. Second, we have encoded prioritized weak bisimulation (cf. Definition 2.4.5), a behavioral relation upon which prioritized observational congruence is defined, as standard bisimulation on enriched transition systems (cf. Proposition 2.4.21). As a consequence, existing *partition-refinement algorithms* [103, 138] can be used for the computation of prioritized weak bisimulation, which is an essential requirement for efficiently implementing our semantic theory in verification tools [64, 65, 153, 156]. Finally, a logical characterization of prioritized weak bisimulation in terms of a variant of the Hennessy-Milner logic [125] provides a basis for system verification via *model checking* [40, 42, 50, 77, 109, 160].

Chapter 3 has demonstrated the importance of priority for modeling and verifying concurrent systems by means of a medium-scale case study dealing with the design of a railway signaling system as used by British Rail [39, 69]. The case study has pointed out that in practice, potential nondeterminism is often resolved by using priorities and timing constraints. If these properties cannot be modeled within the languages used in verification frameworks, models may not accurately reflect the real-world systems under consideration. Consequently, verification results obtained for models need not necessarily be valid in the real world. However, there exist two ways to circumvent this problem. On the one hand, undesired interleavings can be filtered out in the premises of system properties which are to be verified. However, this task turns out to be extremely complicated even for small systems. On the other hand, notions of priority can explicitly be included in the underlying modeling language, as shown by the process algebra $\mathsf{CCS}^{\mathsf{ch}}$. This allows an intuitive modeling and keeps models, e.g. the model of the railway signaling system, small, thereby enhancing the efficiency of verification procedures, such as model checking.

In Chapter 4 we have formally proved that priority and time are related aspects of system behavior when adopting a *dynamic* view of priority and when considering priority values as *time-stamps*. We have introduced the CCS-based process algebra $\mathsf{CCS}^{\mathsf{dp}}$ for modeling and reasoning about *dynamic*-priority and related its semantic theory to the one of the real-time process algebra $\mathsf{CCS}^{\mathsf{rt}}$, a version of Moller and Toft's Temporal CCS [128]. More precisely, we have established a one-to-one correspondence between both semantics in terms of bisimulation (cf. Theorem 4.5.3) and temporal logics (cf. Theorem 4.5.4). Hence, real-time constraints can be encoded by dynamic priorities. The utility of this encoding

stems from the fact that the state space of $\mathsf{CCS}^{\mathsf{dp}}$ models is much smaller and the size of the transition relation is at least not worse, but in practice often better, than the one of corresponding $\mathsf{CCS}^{\mathsf{rt}}$ models. This has been demonstrated by formally modeling and verifying several aspects of the widely-used *SCSI-2 bus-protocol* in Chapter 5, where the state space of the dynamic-priority model is almost an order of magnitude smaller than the one resulting from traditional real-time semantics. A variant of Theorem 4.5.4 provides the formal foundation that all properties proved for our model of the bus-protocol carry over to the real world.

When examining existing approaches to priority and time in process algebras, we have observed that all of them, except Camilleri and Winskel's work [43], incorporate a global notion of pre-emption. We have illustrated that this design decision leads to a counter-intuitive and possibly wrong semantics when reasoning about *distributed* systems. We have overcome this shortcoming by introducing a more natural, *local view* of pre-emption, which has inspired the process algebras $\mathsf{CCS}^{\mathsf{prio}}$ (and $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$, cf. Chapter 6) with distributed priority and CSA (cf. Chapter 7) with distributed time, respectively. Altering the concept of pre-emption has far-reaching consequences on the semantic theories.

Subsequently, we have re-developed the bisimulation-based semantic theory for $\mathsf{CCS}^{\mathsf{prio}}$. It includes a characterization of the largest congruences, distributed prioritized strong bisimulation (cf. Definition 6.4.2) and distributed prioritized observational congruence (cf. Definition 6.5.10), contained in the naive adaptations of the standard strong and weak bisimulations, respectively (see Theorems 6.4.5 and 6.5.11). Moreover, we have encoded our behavioral relations as standard bisimulations on enriched transition systems (cf. Theorems 6.4.16 and 6.5.16). Finally, we have presented an axiomatic characterization of distributed prioritized strong bisimulation for finite processes (cf. Theorem 6.4.14), which points to the mathematical tractability and elegance of the developed theory, and logical characterizations of distributed prioritized strong and distributed prioritized weak bisimulation (see Theorem 6.4.17). Additionally, we have generalized $\mathsf{CCS}^{\mathsf{prio}}$ to $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ by introducing a multi-level priority-scheme and by allowing inter–priority-level communication (cf. Section 6.7.2). By means of an example we have demonstrated the utility of our approach for verifying and reasoning about priority in theoretical as well as practical settings.

Existing approaches to quantitative or qualitative time have concentrated on modeling centralized systems, as opposed to distributed systems, too. All of them, besides Andersen and Mendler's PMC [5, 6], consider only a single, global clock. However, when it comes to networks or hardware, most systems contain multiple clocks. Unfortunately, PMC is not flexible enough for designing systems because processes are forced to deadlock whenever an expected communication partner is not yet ready. Our temporal process algebra CSA, as introduced in Chapter 7, circumvents this shortcoming by allowing processes to wait until a communication partner becomes available. However, if two processes are able to communicate, this communication must take place instantaneously, as formalized in the notion of *maximal progress* which has also been adopted by other temporal process algebras such as TPL [92]. Maximal progress corresponds to priority in the way that actions are given

precedence over clock ticks. By looking at generic examples drawn from hardware design we have motivated that maximal progress needs to be localized when dealing with multiple clocks, thus allowing one to reason about synchrony and asynchrony within a common framework. The qualitative notion of time incorporated in CSA allows us to specify abstract synchronization constraints which are often needed to model the functional behavior of systems accurately. Our technical results for the presented bisimulation-based semantic theory for CSA are similar to the ones for $\mathsf{CCS}^{\mathsf{prio}}$. They include abstract behavioral congruences (see Definition 7.5.2 and Theorem 7.5.4, as well as Definition 7.6.5 and Theorem 7.6.8) together with their logical characterizations (cf. Sections 7.5.4 and 7.6.3) and a sound and complete axiomatization of temporal strong bisimulation for several classes of processes (cf. Section 7.5.2).

Moreover, we have formally compared our theories to the most relevant existing approaches. The process algebra $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ has been found to be strongly connected to Camilleri and Winskel's approach to priority [43], $\mathsf{CCS}^{\mathsf{cw}}$, which is also based on the process-algebraic framework of CCS (cf. Section 6.7.3). However, Camilleri and Winskel introduce a new choice operator favoring one summand over the other instead of explicitly attaching priority values to transitions as is done in $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$. Although both concepts are different at first sight, they semantically coincide on the class of processes that can be expressed in $\mathsf{CCS}^{\mathsf{cw}}$ (cf. Theorem 6.7.9). Thus, our approach is more general, especially concerning the important aspect of *compositionality*.

Our calculus of synchrony and asynchrony, CSA, is a conservative extension of Hennessy and Regan's TPL by multiple clocks and closely related to PMC. Essentially, CSA combines the features of both TPL and PMC while maintaining a clean semantic treatment. Additionally, we have discussed the similarities and differences of the semantic concept of local pre-emption in $\mathsf{CCS}^{\mathsf{prio}}$ and CSA in Section 8.1. We have found out that the basic idea of localizing pre-emption is the same since in both process algebras we take into account the explicit or implicit distribution of processes. However, priorities and clocks have two slightly different views of localities, which is reflected in the different theories for observational congruence in $\mathsf{CCS}^{\mathsf{prio}}$ and CSA, respectively. Because of this we believe that it is very challenging to develop a uniform theory for local pre-emption.

Finally, we want to emphasize that the insights about global and local pre-emption gained in this dissertation are of general nature and not restricted to process-algebraic languages based on CCS. We believe that our approach to qualitative timing aspects can be exploited for (re-)defining intuitive and mathematically tractable semantic extensions to priority and time in existing programming and specification languages (cf. Chapter 8.2). We conjecture that our frameworks are general and flexible enough to remedy certain problems in other languages, as e.g. encountered for the priority approach taken in Statecharts [91].

9.2 Critical Remarks

It seems appropriate to make some critical remarks concerning both the design decisions and the technical developments of the presented semantic theories for priority and time.

We have made supreme efforts to intuitively justify every design decision made in this dissertation. However, there are some elementary design decisions for our semantic theories on which we have, so far, not commented at all. They can be summarized by the following question. Why have we chosen CCS as the base upon which we have built our theories? In doing so, of course, we have accepted all design decisions already made in CCS [125]. Especially we have restricted ourselves to a semantics based on interleaving and the principle of handshake communication. The answer to the above question is manifold. First, there are historical reasons and reasons of personal preference, since the author of this dissertation has a strong background in CCS and feels most comfortable within this calculus. Second, the semantic framework of CCS, which is known among a large community of researchers, provides well-established mathematical foundations and proof methods for the semantic theory of bisimulation. Although numerous research results have been obtained in or have been adapted to CCS, it is surprising that nobody – to the best of our knowledge – has considered the important semantic phenomenon of pre-emption in as much detail as is done in this thesis. Third, our main objective has been to make process algebras more applicable for practice. It is our conviction that the design decisions underlying CCS are desirable for implementable distributed programming languages. Interleaving semantics is very simple as it comes with labeled transition systems as semantic objects. This is of utmost importance since most existing efficient analysis and verification algorithms work on finite-state automata. In contrast, *true concurrency* semantics [171] are defined on less elementary structures for which less analysis and verification techniques have been developed [75]. Regarding easily *implementable* communication mechanisms, handshake communication has been found to be the most adequate one and is implemented as a communication primitive in many existing distributed computing environments. Multi-party communication mechanisms, based on the principle of synchronous broadcasting, are extremely complicated to implement within distributed environments since they usually have to be encoded using complex protocols. Note that synchronizations on clocks also have a broadcasting character. However, in the algebra CSA clocks are local. Hence, they can naturally be implemented using either common signals produced by a hardware component or indirectly in terms of constraints involving quantitative time, depending on the distribution of the system under consideration. It is because of the above arguments that we feel that the process algebra CCS provides a solid semantic basis for our extensions by priority and time.

A few design decisions, especially restricting the syntax of $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ when generalizing $\mathsf{CCS}^{\mathsf{prio}}$ to inter–priority-level communication, have purely technical reasons and are less intuitive. Nevertheless, we have sketched how the operational semantics of $\mathsf{CCS}^{\mathsf{prio}}_{\mathsf{pp}}$ has to be defined if one wants to avoid the syntax restriction. The price one has to pay is that the side condition of the operational rule concerning the restriction operator becomes extremely complex, thereby probably preventing the development of an elegant algebraic theory. The

chosen syntax restriction is inspired by programming languages like `occam` whose implementability would otherwise be infeasible. However, if one just wants to program a (global) state-space generator [35, 62, 153] and is not interested in algebraic properties of the underlying semantics, our proposal for avoiding the syntax restriction should be just fine. Since this dissertation focuses on algebraic theories, we have been forced to accept the syntax restriction. We are not aware of any solution to circumvent the intrinsic problem of contradicting priority information within a system state, which also occurs in other assume-guarantee paradigms.

In this dissertation we have not presented a *general* theory for the concept of pre-emption in process algebras whose semantics is based on the principle of interleaving with handshake communication. Instead we have developed theories concerning two different application areas for pre-emption, namely theories for priority and time. In fact, it has not been intended to develop a general theory because of the following two reasons. First, to the best of our knowledge, pre-emption only arises in frameworks either related to priority or to time. Second, a general theory therefore would have to cover at least both, the prioritized and the timed case. Whereas this seems possible with respect to global pre-emption, local pre-emption has a different flavor in these two settings. For distributed time it is implemented by observing the communication potential with respect to each clock individually, i.e. by observing if both partners of a potential communication are connected to the considered clock. In contrast, when dealing with distributed priority one also observes local communication potential, however, only one of the considered communication partners has to be located in the observed locality. Because of these facts, finding a general framework for a theory of pre-emption seems to be difficult.

Finally, the reader might have noticed that the semantic theories for $\mathsf{CCS}^{\mathsf{ch}}$, $\mathsf{CCS}^{\mathsf{prio}}$, and CSA are not completed yet, since we have not provided sound and complete axiomatizations of the *observational* congruences. For *finite* processes, one should be able to establish these axiomatizations using standard techniques [126]. However, for *regular* processes, i.e. the class of finite-state processes not containing recursion through static operators, it is not clear how to obtain a completeness result. The point is that existing methods for proving completeness of axiomatizations with respect to observational congruences rely on the possibility to remove or to insert τ-cycles in processes [126]. In the context of pre-emption, however, this would possibly change the pre-emption potential of the processes under consideration which is semantically incompatible with the underlying observational congruences. Thus, it remains an open problem how to prove axiomatizations of our observational congruences complete for regular processes.

9.3 Future Work and Outlook

Future work should be done along several orthogonal directions from both, a theoretical and an application-oriented point of view. Thereby, the strength of the presented semantic theories should be developed further, whereas their weaknesses might be eliminated.

Concerning theoretical work, the following four aspects seem interesting candidates for further investigations. First, we have encountered that our semantic theories of observational congruence are rather complicated due to the nature of pre-emption in our interleaving-based semantics. Especially, for our approach to distributed priority it turns out that the usual interleaving law, which shows how parallelism can be reduced to nondeterminism, is not valid. Therefore, it is worth pursuing the aspect of pre-emption in process-algebraic frameworks dealing with non-interleaving semantics. Preliminary considerations have been made in Jensen's PhD thesis [102]. However, the insights obtained by Jensen are restricted to a structural operational semantics using *asynchronous transition systems* [171]. They do not comprise a semantic theory based on a behavioral relation such as bisimulation [132]. Second, the semantic theories for $\mathsf{CCS}^{\mathsf{ch}}$, $\mathsf{CCS}^{\mathsf{prio}}$ and CSA need to be completed by providing sound and complete axiomatizations of the observational congruences. Regarding $\mathsf{CCS}^{\mathsf{ch}}$ an axiomatization of prioritized observational congruence for *finite* processes would be particularly helpful since it would allow a precise semantic comparison with the corresponding relation presented in [133, 134]. As already mentioned, standard techniques for proving axiomatizations complete for *regular* processes are not applicable directly with respect to the presented process algebras. Therefore, developing complete axiomatizations of observational congruences in process-algebraic frameworks involving pre-emption constitutes a challenge for regular processes and may gain new insights concerning both, global and local, pre-emption, technically as well as conceptually. Recently, a similar problem has been attacked in [93] for a stochastic timed version of CCS with maximal progress. However, the notion of observational congruence employed in this paper differs from Milner's original by adding a notion of fairness which is sensitive to escaping divergence, i.e. infinite internal computation. The authors conjecture that their techniques can be adapted to priority frameworks, too. Third, introducing multiple clocks in process algebras opens the door for abstractions going further than temporal observational congruence. Synchronization constraints are often needed when developing a distributed implementation that satisfies some (sequential) specification. However, when implementation and specification should be related by a behavioral equivalence, one has to abstract from clocks again. Hence, clock ticks have to be made unobservable. It needs to be further investigated if our proposal of introducing a *clock-hiding* operator meets this need. Moreover, one may also think of a new form of parallel composition parameterized by clock constraints, expressing e.g. that one clock of the left argument process should be identified with another clock of the right argument process. This additional synchronization, corresponding to a kind of *clock relabeling*, would certainly enhance the utility of CSA for the *modular* construction of temporal distributed systems. Fourth, having studied concepts of pre-emption within relatively simple CCS-based frameworks, it would be interesting to see if our approaches can be carried over to more expressive calculi involving *mobile processes* [80, 127].

Suggestions for more practical future work aim at transferring the insights obtained in this dissertation, especially concerning well-founded semantic theories for distributed

priority and distributed time, to practice. As a first step towards applicability, the process algebras $\mathsf{CCS}^{\mathsf{prio}}$ and CSA, or a combined version of them, need to be implemented in existing verification tools [48, 53, 65, 156, 159]. Along with these implementations, our calculi could be enhanced by introducing *value-passing* along the lines of [67]. As has been done for the process algebras $\mathsf{CCS}^{\mathsf{ch}}$, $\mathsf{CCS}^{\mathsf{rt}}$, and $\mathsf{CCS}^{\mathsf{dp}}$, the languages $\mathsf{CCS}^{\mathsf{prio}}$ and CSA could be integrated as front-ends of the *Concurrency Workbench of North-Carolina* [65, 153] using the *Process Algebra Compiler* [62]. This integration is currently on its way. Once having also implemented the corresponding semantic theories, one is able to evaluate the relevance of our process algebras by applying them for modeling and verifying large-scale, distributed systems in practice. Implementing the various bisimulations requires to adapt Paige and Tarjan's partition-refinement algorithm [138]. The resulting *bisimulation checkers* can also be used for reducing system sizes by identifying semantically equivalent system states [86], thereby enabling verification tools to handle larger systems.

Our outlook is inspired by the observation that semantically well-founded specification languages are of need in industrial contexts. In fact, many software engineers are continuing to use notations with no or insufficient underlying semantic theories, thereby making a formal analysis and verification of their products impossible. Although our approaches to priority and time in process algebras have impaired the gap between theoretically tractable and practically implementable specification languages, various other aspects have to be considered in order to attract engineers. On the one hand, soft- and hardware is often developed in a *top-down* fashion which needs to be supported by introducing a *hierarchy* (*state-refinement*) *operator* as is done in Statecharts [91]. In contrast, our approach follows the process-algebraic traditions and focuses on compositionality, i.e. on *bottom-up* designs. Thus, it would be useful to incorporate a notion of state-refinement in the presented process algebras. On the other hand, in order to make these languages user-friendly, graphical interfaces [1, 53, 158] need to be developed, thereby wrapping up their "process algebra feel" by a "Statecharts-like look."

Bibliography

[1] H. Ben Abdallah. *Graphical Communicating Shared Resources: A Language for the Specification, Refinement and Analysis of Real-Time Systems.* PhD thesis, University of Pennsylvania, August 1996.

[2] J. Leach Albert, B. Monien, and M. Rodríguez Artalejo, editors. *Automata, Languages and Programming (ICALP'91)*, volume 510 of *Lecture Notes in Computer Science*, Madrid, July 1991. Springer-Verlag.

[3] American National Standard Institute. *ANSI X3.131–1994, Information Systems — Small Computer Systems Interface–2.* ANSI, 1994. A draft of this standard is also available at http://abekas.com:8080/SCSI2/.

[4] H.R. Andersen and M. Mendler. Complete axiomatization of observational congruence for PMC. Technical Report ID-TR:1993-126, Department of Computer Science, Technical University of Denmark, December 1993.

[5] H.R. Andersen and M. Mendler. An asynchronous process algebra with multiple clocks. In Sannella [150], pages 58–73.

[6] H.R. Andersen and M. Mendler. Describing a signal analyzer in the process algebra PMC — A case study. In P.D. Mosses, M. Nielsen, and M.I. Schwartzbach, editors, *Theory and Practice of Software Development (TAPSOFT'95)*, volume 915 of *Lecture Notes in Computer Science*, pages 620–635. Springer-Verlag, 1995.

[7] K.R. Apt and E.-R. Olderog. *Verification of Sequential and Concurrent Programs.* Springer, 1991.

[8] J.C.M. Baeten, editor. *Applications of Process Algebra*, volume 17 of *Cambridge Tracts in Theoretical Computer Science.* Cambridge University Press, Cambridge, England, 1990.

[9] J.C.M. Baeten and J.A. Bergstra. Real time process algebra. *Formal Aspects of Computing*, 3(2):142–188, 1991.

[10] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Syntax and defining equations for an interrupt mechanism in process algebra. *Fundamenta Informaticae IX*, pages 127–168, 1986.

[11] J.C.M. Baeten and J.W. Klop, editors. *CONCUR'90 (Concurrency Theory)*, volume 458 of *Lecture Notes in Computer Science*, Amsterdam, The Netherlands, August 1990. Springer-Verlag.

[12] J.C.M. Baeten and W.P. Weijland. *Process Algebra*, volume 18 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, England, 1990.

[13] G. Barrett. The semantics of priority and fairness in occam. In M. Main, A. Melton, M. Mislove, and D. Schmidt, editors, *Proceedings of Mathematical Foundations of Programming Semantics*, volume 442 of *Lecture Notes in Computer Science*, pages 194–208. Springer-Verlag, 1989.

[14] M. v.d. Beeck. *Ein Kontrollmodell für die Strukturierte Analyse*. PhD thesis, Aachen University of Technology, Aachen, Germany, July 1995.

[15] A. Benveniste and G. Berry. The synchronous approach to reactive and real-time systems. *Proceedings of the IEEE*, 79(9):1270–1282, September 1991.

[16] J.A. Bergstra and J.W. Klop. Process algebra for synchronous communication. *Information and Computation*, 60:109–137, 1984.

[17] J.A. Bergstra and J.W. Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, 37:77–121, 1985.

[18] M. Bernardo and R. Gorrieri. Extended markovian process algebra. In Montanari and Sassone [130], pages 315–330.

[19] G. Berry. Preemption in concurrent systems. In R.K. Shyamasundar, editor, *Foundations of Software Technology and Theoretical Computer Science*, volume 761 of *Lecture Notes in Computer Science*, pages 72–93. Springer-Verlag, 1993.

[20] G. Berry and L. Cosserat. The ESTEREL synchronous programming language and its mathematical semantics. In S. D. Brookes, A. W. Roscoe, and G. Winskel, editors, *Seminar on Concurrency*, Lecture Notes in Computer Science, pages 389–448. Springer-Verlag, 1984.

[21] G. Berry and G. Gonthier. The ESTEREL synchronous programming language: Design, semantics, implementation. *Science of Computer Programming*, 19:87–152, 1992.

[22] G. Berry, S. Ramesh, and R.K. Shyamasundar. Communicating reactive processes. In *Twentieth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'93)*, pages 85–98, Charleston, South Carolina, January 1993. IEEE Computer Society Press.

[23] E. Best, editor. *CONCUR'93 (Concurrency Theory)*, volume 715 of *Lecture Notes in Computer Science*, Hildesheim, Germany, August 1993. Springer-Verlag.

[24] E. Best and M. Koutny. Petri net semantics of priority systems. *Theoretical Computer Science*, 96:175–215, 1992.

[25] G. Bhat. *Tableau-based Approaches to Model Checking*. PhD thesis, North Carolina State University, Raleigh, NC, USA, December 1997.

[26] G. Bhat and R. Cleaveland. Efficient local model-checking for fragments of the modal μ-calculus. In Margaria and Steffen [121], pages 107–126.

[27] G. Bhat, R. Cleaveland, and G. Lüttgen. Dynamic priorities for modeling real-time. In T. Mizuno, N. Shiratori, T. Higashino, and A. Togashi, editors, *Formal Desciption Techniques and Protocol Specification, Testing and Verification (FORTE X/PSTV XVII '97)*, pages 321–336, Osaka, November 1997. Chapman & Hall.

[28] G. v. Bochmann and D.K. Probst, editors. *Computer Aided Verification (CAV'92)*, volume 663 of *Lecture Notes in Computer Science*, Montréal, Canada, June/July 1992. Springer-Verlag.

[29] R.N. Bol and J.F. Groote. The meaning of negative premises in transition system specifications. In Albert et al. [2], pages 481–494.

[30] T. Bolognesi and E. Brinksma. Introduction to the ISO specification language LOTOS. *Computer Networks and ISDN Systems*, 14:25–59, 1987.

[31] T. Bolognesi and F. Lucidi. LOTOS-like process algebras with urgent or timed interactions. In K. Parker and G. Rose, editors, *Proceedings of the 4^{th} International Conference on Formal Description Techniques, FORTE'91*. North-Holland, 1992.

[32] G. Boudol and I. Castellani. A non-interleaving semantics for CCS based on proved transitions. *Fundamenta Informaticae*, XI(4):433–452, 1988.

[33] G. Boudol, I. Castellani, M. Hennessy, and A. Kiehn. Observing localities. *Theoretical Computer Science*, 114(1):31–61, June 1993.

[34] F. Boussinot and R. De Simone. The ESTEREL language. *Proceedings of the IEEE*, 79(9):1293–1304, September 1991.

[35] V. Braun. Ein Transitionssystem-Generator für Prozeßalgebren. Master's thesis, Technical University of Aachen, Aachen, Germany, October 1994.

[36] P. Brémond-Grégoire, J.-Y. Choi, and I. Lee. A complete axiomatization of finite-state ACSR processes. *Information and Computation*, 138(2):124–159, November 1997.

[37] P. Brémond-Grégoire, S. Davidson, and I. Lee. CCSR92: Calculus for communicating shared resources with dynamic priorities. In Purushothaman and Zwarico [146], pages 65–85.

[38] P. Brémonde-Grégoire, I. Lee, and R. Gerber. A process algebra of communicating shared resources with dense time and priorities. *Theoretical Computer Science*, 189(1-2):179–219, December 1997.

[39] G. Bruns. A case study in safety-critical design. In Bochmann and Probst [28], pages 220–233.

[40] O. Burkart and Y.-M. Quemener. Model-checking of infinite graphs defined by graph grammars. In *International Workshop on Verification of Infinite State Systems (INFINITY'96)*, Electronic Notes of Theoretical Computer Science (ENTCS). Elsevier Science B.V., 1997.

[41] O. Burkart and B. Steffen. Model checking for context-free processes. In Cleaveland [51], pages 123–137.

[42] O. Burkart and B. Steffen. Composition, decomposition, and model-checking pushdown processes. *Nordic Journal of Computing*, 2:89–125, 1995.

[43] J. Camilleri and G. Winskel. CCS with priority choice. *Information and Computation*, 116(1):26–37, January 1995.

[44] D.M. Chapiro. Reliable high-speed arbitration and synchronization. *IEEE Transaction on Computers*, C-36(10):1251–1255, October 1987.

[45] L. Chen. An interleaving model for real-time systems. Technical Report ECS-LFCS-91-184, Laboratory for Foundations of Computer Science, November 1991.

[46] L. Christoff. *Specification and Verification Methods for Probabilistic Processes.* PhD thesis, Uppsala University, Uppsala, Sweden, 1993.

[47] A. Claßen. Modulare Statecharts: Ein formaler Rahmen zur hierarchischen Prozeßspezifikation. Master's thesis, Technical University of Aachen, Aachen, Germany, March 1993.

[48] A. Claßen. *Component Integration in METAFrame.* PhD thesis, University of Passau, Passau, Germany, 1997. Published by Shaker Verlag.

[49] A. Claßen, M. Klein, J. Knoop, T. Margaria, and B. Steffen. The fixpoint-analysis machine. In Lee and Smolka [112], pages 72–87.

[50] R. Cleaveland. Tableau-based model checking in the propositional mu-calculus. *Acta Informatica*, 27(8):725–747, September 1990.

[51] R. Cleaveland, editor. *CONCUR'92 (Concurrency Theory)*, volume 630 of *Lecture Notes in Computer Science*, Stony Brook, New York, August 1992. Springer-Verlag.

[52] R. Cleaveland. Analyzing concurrent systems using the Concurrency Workbench. In P.E. Lauer, editor, *Functional Programming, Concurrency, Simulation and Automated Reasoning*, volume 693 of *Lecture Notes in Computer Science*, pages 129–144. Springer-Verlag, 1993.

[53] R. Cleaveland, J. Gada, P. Lewis, S. Smolka, O. Sokolsky, and S. Zhang. The concurrency factory—practical tools for the specification, simulation, verification, and implementation of concurrent systems. In G.E. Blelloch, K.M. Chandy, and S. Jagannathan, editors, *Specification of Parallel Algorithms*, volume 18 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 75–90, Piscataway, NJ, May 1994. American Mathematical Society.

[54] R. Cleaveland and M.C.B. Hennessy. Priorities in process algebra. *Information and Computation*, 87(1/2):58–77, July/August 1990.

[55] R. Cleaveland and M.C.B. Hennessy. Testing equivalence as a bisimulation equivalence. *Formal Aspects of Computing*, 5:1–20, 1993.

[56] R. Cleaveland, M. Klein, and B. Steffen. Faster model checking for the modal mu-calculus. In Bochmann and Probst [28], pages 410–422.

[57] R. Cleaveland, G. Lüttgen, and M. Mendler. An algebraic theory of multiple clocks. In A. Mazurkiewicz and J. Winkowski, editors, *CONCUR'97 (Concurrency Theory)*, volume 1243 of *Lecture Notes in Computer Science*, pages 166–180, Warsaw, Poland, July 1997. Springer-Verlag.

[58] R. Cleaveland, G. Lüttgen, and V. Natarajan. A process algebra with distributed priorities. In Montanari and Sassone [130], pages 34–49.

[59] R. Cleaveland, G. Lüttgen, and V. Natarajan. A process algebra with distributed priorities. *Theoretical Computer Science*, 195(2):227–258, March 1998.

[60] R. Cleaveland, G. Lüttgen, and V. Natarajan. Priority in process algebra. In J.A. Bergstra, A. Ponse, and S.A. Smolka, editors, *Handbook of Process Algebra*. Elsevier, January 1999. In preparation.

[61] R. Cleaveland, G. Lüttgen, V. Natarajan, and S. Sims. Priorities for modeling and verifying distributed systems. In Margaria and Steffen [121], pages 278–297.

[62] R. Cleaveland, E. Madelaine, and S. Sims. Generating front-ends for verification tools. In E. Brinksma, W.R. Cleaveland, K.G. Larsen, T. Margaria, and B. Steffen, editors, *First International Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'95)*, volume 1019 of *Lecture Notes in Computer Science*, pages 153–173, Aarhus, Denmark, May 1995. Springer-Verlag.

[63] R. Cleaveland, V. Natarajan, S. Sims, and G. Lüttgen. Modeling and verifying distributed systems using priorities: A case study. *Software–Concepts and Tools*, 17(2):50–62, 1996.

[64] R. Cleaveland, J. Parrow, and B. Steffen. The Concurrency Workbench: A semantics-based tool for the verification of finite-state systems. *ACM Transactions on Programming Languages and Systems*, 15(1):36–72, January 1993.

[65] R. Cleaveland and S. Sims. The NCSU Concurrency Workbench. In R. Alur and T. Henzinger, editors, *Computer Aided Verification (CAV'96)*, volume 1102 of *Lecture Notes in Computer Science*, pages 394–397, New Brunswick, New Jersey, July 1996. Springer-Verlag.

[66] R. Cleaveland and B. Steffen. A linear-time model-checking algorithm for the alternation-free modal mu-calculus. *Formal Methods in System Design*, 2:121–147, 1993.

[67] R. Cleaveland and D. Yankelevich. An operational framework for value-passing processes. In *21st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'94)*, pages 326–338, Portland, Oregon, January 1994. IEEE Computer Society Press.

[68] P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the 4^{th} Annual ACM Symposium on Principles of Programming Languages (POPL'77)*, pages 238–252, Los Angeles, CA, 1977.

[69] A.H. Cribbens. Solid-state interlocking (SSI): An integrated electronic signalling system for mainline railways. *IEE Proceedings*, 134, Pt. B(3), May 1987.

[70] J. Davies, D. Jackson, and S. Schneider. Broadcast communication for real-time processes. In Vytopil [169], pages 149–170.

[71] J. Davies and S. Schneider. A brief history of Timed CSP. *Theoretical Computer Science*, 138(2):243–271, February 1995.

[72] R. De Nicola and M.C.B. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1983.

[73] P. Degano, R. De Nicola, and U. Montanari. A partial ordering semantics for CCS. *Theoretical Computer Science*, 75:223–262, 1990.

[74] P. Degano and C. Priami. Causality of mobile processes. In Z. Fülöp and F. Gécseg, editors, *Automata, Languages and Programming (ICALP'95)*, volume 944 of *Lecture Notes in Computer Science*, pages 660–671, Szeged, Hungary, July 1995. Springer-Verlag.

[75] V. Diekert and G. Rozenberg. *The book of traces*. World Scientific, Singapore, 1995.

[76] W. Elseaidy, J. Baugh, and R. Cleaveland. Verification of an active control system using temporal process algebra. *Engineering with Computers*, 12:46–61, 1996.

[77] E.A. Emerson and C.-L. Lei. Efficient model checking in fragments of the propositional mu-calculus. In *Symposium on Logic in Computer Science (LICS'86)*, pages 267–278, Cambridge, Massachusetts, June 1986. IEEE Computer Society Press.

[78] D. Harel et al. STATEMATE: A working environment for the development of complex reactive systems. *IEEE Transactions on Software Engineering*, 16(4):403–413, April 1990.

[79] C.J. Fidge. A formal definition of priority in CSP. *ACM Transactions on Programming Languages and Systems*, 15(4):681–705, September 1993.

[80] C. Fournet, G. Gonthier, J. Lévy, L. Maranget, and D. Rémy. A calculus of mobile agents. In Montanari and Sassone [130], pages 406–421.

[81] N. Francez. *Fairness*. Springer-Verlag, New York, 1986.

[82] R. Gerber and I. Lee. CCSR: A calculus for communicating shared resources. In Baeten and Klop [11], pages 263–277.

[83] R. Gerber and I. Lee. A resourced-based prioritized bisimulation for real-time systems. *Information and Computation*, 113:102–142, 1994.

[84] S.M. German. Programming in a general model of synchronization. In Cleaveland [51], pages 534–549.

[85] J.C. Godskesen. An operational semantic model for Basic SDL. In O. Færgemand and R. Reed, editors, *Proceedings of the 5th SDL forum (SDL'91 – Evolving Methods)*, pages 15–22. North-Holland, 1991.

[86] S. Graf, B. Steffen, and G. Lüttgen. Compositional minimisation of finite state systems using interface specifications. *Formal Aspects of Computing*, 8:607–616, 1996.

[87] P. Le Guernic, T. Gautier, M. Le Borgne, and C. Le Maire. Programming real-time applications with SIGNAL. *Proceedings of the IEEE*, 79(9):1321–1336, September 1991.

[88] N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud. The synchronous data flow programming language LUSTRE. *Proceedings of the IEEE*, 79(9):1305–1320, September 1991.

[89] H. Hansson and F. Orava. A process calculus with incomparable priorities. In Purushothaman and Zwarico [146], pages 43–64.

[90] H.A. Hansson. *Time and Probability in Formal Design of Distributed Systems.* PhD thesis, Uppsala University, September 1991.

[91] D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8:231–274, 1987.

[92] M. Hennessy and T. Regan. A process algebra for timed systems. *Information and Computation*, 117:221–239, 1995.

[93] H. Hermanns and M. Lohrey. Observational congruence in a stochastic timed calculus with maximal progress. Technical Report IMMD-VII/7-97, Department of Computer Science, University of Erlangen, Germany, 1997. To appear in Proceedings of CONCUR'98 (Concurrency Theory).

[94] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, London, 1985.

[95] INMOS Limited. *Occam Programming Manual.* International Series in Computer Science. Prentice Hall, 1984.

[96] Intel Corporation. Pentium(R) processor family developer's manual, 1997.

[97] ISO. Information processing systems. definition of the temporal ordering specification language LOTOS. TC97/16 N1987, 1984.

[98] ISO/IEC. Working draft on enhancements on LOTOS. JTC1/SC21/WG7, January 1997.

[99] ITU-T. Specification and description language (SDL). Recommendation Z.100, Revision 1, ITU, 1994.

[100] R. Janicki and M. Koutny. Semantics of inhibitor nets. *Information and Computation*, 123(1):1–17, 1995.

[101] A. Jeffrey. Translating timed process algebra into prioritized process algebra. In Vytopil [169], pages 493–506.

[102] C.-T. Jensen. *Prioritized and Independent Actions in Distributed Computer Systems.* PhD thesis, Aarhus University, August 1994.

[103] P. Kanellakis and S.A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. *Information and Computation*, 86(1):43–68, May 1990.

[104] P. Kelb. *Abstraktionstechniken für automatische Verifikationsmethoden.* PhD thesis, University of Oldenburg, Oldenburg, Germany, 1996. Published by Shaker Verlag.

[105] Kempe Software Capital Enterprises. Ada95 reference manual: Language and standard libraries, 1995. Available at http://www.adahome.com.

[106] Y. Kesten and A. Pnueli. Timed and hybrid statecharts and their textual representation. In Vytopil [169], pages 591–620.

[107] M. Kick. Modelling synchrony and asynchrony with multiple clocks. Master's thesis, University of Passau, Passau, Germany, 1998. To appear.

[108] C. Delgado Kloos and P. T. Breuer, editors. *Formal Semantics for VHDL*. Kluwer, 1995.

[109] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[110] L. Lamport. What it means for a concurrent program to satisfy a specification: Why no one has specified priority. In *Twelfth Annual ACM Symposium on Principles of Programming Languages (POPL'85)*, pages 78–83, Orlando, Florida, January 1985. IEEE Computer Society Press.

[111] G. Leduc. An upward compatible timed extension to LOTOS. In K. Parker and G. Rose, editors, *Proceedings of the 4^{th} International Conference on Formal Description Techniques, FORTE'91*. North-Holland, 1992.

[112] I. Lee and S. A. Smolka, editors. *CONCUR'95 (Concurrency Theory)*, volume 962 of *Lecture Notes in Computer Science*, Philadelphia, PA, USA, August 1995. Springer-Verlag.

[113] C. Lengauer. Loop parallelization in the polytope model. In Best [23], pages 398–416.

[114] L. Léonard and G. Leduc. An introduction to ET-LOTOS for the description of time-sensitive systems. *Computer Networks and ISDN Systems*, 29:271–292, 1997.

[115] J. Loeckx and K. Sieber. *The Foundations of Program Verification*. Wiley-Teubner series in computer sience. Teubner, Stuttgart, 1987.

[116] G. Lowe. Probabilistic and prioritized models of timed CSP. *Theoretical Computer Science*, 138(2):315–352, February 1995.

[117] A. Maggiolo-Schettini, A. Peron, and S. Tini. Equivalences of statecharts. In Montanari and Sassone [130], pages 687–702.

[118] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, Berlin, 1992.

[119] F. Maraninchi. The ARGOS language: Graphical representation of automata and description of reactive systems. In *IEEE Workshop on Visual Languages*, October 1991.

[120] F. Maraninchi. Operational and compositional semantics of synchronous automaton compositions. In Cleaveland [51], pages 550–564.

[121] T. Margaria and B. Steffen, editors. *Second International Workshop on Tools and Algorithms for the Construction and Analysis of Systems TACAS'96*, volume 1055 of *Lecture Notes in Computer Science*, Passau, Germany, March 1996. Springer-Verlag.

[122] A. Mazurkiewicz. Trace theory. In M.P. Chytil et al., editor, *11th Symposium on Mathematical Foundations of Computer Science*, volume 176 of *Lecture Notes in Computer Science*, pages 115–133. Springer-Verlag, 1984.

[123] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25:267–310, 1983.

[124] R. Milner. A complete inference system for a class of regular behaviours. *Journal of Computer and System Sciences*, 28:439–466, 1984.

[125] R. Milner. *Communication and Concurrency*. Prentice-Hall, London, 1989.

[126] R. Milner. A complete axiomatization for observational congruence of finite-state behaviours. *Information and Computation*, 81:227–247, 1989.

[127] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, I and II. *Information and Computation*, 100(1):1–77, September 1992.

[128] F. Moller and C. Tofts. A temporal calculus of communicating systems. In Baeten and Klop [11], pages 401–415.

[129] F. Moller and C. Tofts. Relating processes with respect to speed. In J.C.M. Baeten and J.F. Groote, editors, *CONCUR'91 (Concurrency Theory)*, volume 527 of *Lecture Notes in Computer Science*, pages 424–438, Amsterdam, The Netherlands, August 1991. Springer-Verlag.

[130] U. Montanari and V. Sassone, editors. *CONCUR'96 (Concurrency Theory)*, volume 1119 of *Lecture Notes in Computer Science*, Pisa, Italy, August 1996. Springer-Verlag.

[131] U. Montanari and D. Yankelevich. Location equivalence in a parametric setting. *Theoretical Computer Science*, 149:299–332, 1995.

[132] M. Mukund and M. Nielsen. CCS, locations and asynchronous transition systems. In R.K. Shyamasundar, editor, *Foundations of Software Technology and Theoretical Computer Science*, volume 652 of *Lecture Notes in Computer Science*, pages 328–341. Springer-Verlag, 1992.

[133] V. Natarajan. *Degrees of Delay: Semantic Theories for Priority, Efficiency, Fairness, and Predictability in Process Algebras*. PhD thesis, North Carolina State University, Raleigh, NC, USA, August 1996.

[134] V. Natarajan, L. Christoff, I. Christoff, and R. Cleaveland. Priorities and abstraction in process algebra. In P.S. Thiagarajan, editor, *Foundations of Software Technology and Theoretical Computer Science*, volume 880 of *Lecture Notes in Computer Science*, pages 217–230, Madras, India, December 1994. Springer-Verlag.

[135] X. Nicollin and J. Sifakis. An overview and synthesis on timed process algebras. In J.W. de Bakker, C. Huizing, W.P. de Roever, and G. Rozenberg, editors, *REX Workshop on Real-Time: Theory in Practice*, volume 600 of *Lecture Notes in Computer Science*, pages 526–548. Springer-Verlag, 1991.

[136] X. Nicollin and J. Sifakis. The algebra of timed processes, ATP: Theory and application. *Information and Computation*, 114:131–178, 1994.

[137] M. Nielsen, G. Plotkin, and G. Winskel. Petri nets, event structures and domains, Part 1. *Theoretical Computer Science*, 13:85–108, 1981.

[138] R. Paige and R.E. Tarjan. Three partition refinement algorithms. *SIAM Journal of Computing*, 16(6):973–989, December 1987.

[139] D.M.R. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Proceedings of 5th G.I. Conference on Theoretical Computer Science*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer-Verlag, 1981.

[140] A. Peron and A. Maggiolo-Schettini. A new semantics for timed statecharts. In M. Hagiya and J.C. Mitchell, editors, *Proceedings of International Symposium on Theoretical Aspects of Computer Systems (TACS'94)*, volume 789 of *Lecture Notes in Computer Science*, pages 806–821. Springer-Verlag, 1994.

[141] G.D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI-FN-19, Computer Science Department, Aarhus University, Denmark, 1981.

[142] A. Pnueli, editor. *Linear and Branching Structures in the Semantics and Logics of Reactive Systems*, volume 194 of *Lecture Notes in Computer Science*. Springer-Verlag, 1985.

[143] A. Pnueli and M. Shalev. What is in a step: On the semantics of statecharts. In *Theoretical Aspects of Computer Science*, pages 244–264, 1991.

[144] K.V.S. Prasad. Programming with broadcasts. In Best [23], pages 173–187.

[145] K.V.S. Prasad. Broadcasting with priority. In Sannella [150], pages 469–484.

[146] P. Purushothaman and A. Zwarico, editors. *First North American Process Algebra Workshop*, Workshops in Computing, Stony Brook, New York, August 1992. Springer-Verlag.

[147] J. Quemada, A. Azcorra, and A. Fernandez. TIC: A timed calculus for LOTOS. In Son T. Vuong, editor, *Proceedings of the 2nd International Conference on Formal Description Techniques, FORTE'89*. North-Holland, 1990.

[148] W. Reisig. *Petri Nets: An Introduction*. Springer, Berlin, 1985.

[149] D. Sangiorgi and R. Milner. The problem of 'weak bisimulation up to'. In Cleaveland [51], pages 32–46.

[150] D. Sannella, editor. *Proceedings of the 5th European Symposium on Programming (ESOP'94)*, volume 788 of *Lecture Notes in Computer Science*, Edinburgh, U.K., April 1994. Springer-Verlag.

[151] S. Schneider, J. Davies, D.M. Jackson, G.M. Reed, J.N. Reed, and A.W. Roscoe. Timed CSP: Theory and practice. In J.W. de Bakker, C. Huizing, W.P. de Roever, and G. Rozenberg, editors, *Real-Time: Theory in Practice*, volume 600 of *Lecture Notes in Computer Science*, pages 526–548. Springer-Verlag, 1991.

[152] K. Seidel. *Probabilistic Communicating Processes*. PhD thesis, University of Oxford, Oxford, England, 1992.

[153] S. Sims. *Customizable Tools for Verifying Concurrent Systems*. PhD thesis, North Carolina State University, Raleigh, NC, USA, November 1997.

[154] S.A. Smolka and B. Steffen. Priority as extremal probability. *Formal Aspects of Computing*, 8:585–606, 1996.

[155] B. Steffen. Model checking in practice. In Lee and Smolka [112]. Tutorial.

[156] B. Steffen. METAFrame in practice. In *Electronic Notes of Theoretical Computer Science (ENTCS)*. Elsevier Science B.V., June 1996.

[157] B. Steffen and A. Ingolfsdottir. Characteristic formulae for finite state processes. *Information and Computation*, 110(1):149–163, April 1994.

[158] B. Steffen and T. Margaria. Tools get formal methods into practice. *ACM Computing Surveys*, 28A(4):126, December 1996.

[159] B. Steffen, T. Margaria, and A. Claßen. Heterogeneous analysis and verification for distributed systems. *Software—Concepts and Tools*, 17:13–25, 1996.

[160] C. Stirling and D. Walker. Local model checking in the modal mu-calculus. *Theoretical Computer Science*, 89:161–177, 1991.

[161] C. Tofts. Processes with probablities, priority and time. *Formal Aspects of Computing*, 6(5):536–564, 1994.

[162] K.J. Turner, editor. *Using Formal Description Techniques*, Series in Communication and Distributed Systems. John Wiley & Sons, 1993.

[163] W.M. Turski. Time considered irrelevant for real-time systems. *BIT*, 28:473–486, 1988.

[164] A.C. Uselton and S.A. Smolka. A compositional semantics for statecharts using labeled transition systems. In B. Jonsson and J. Parrow, editors, *CONCUR'94 (Concurrency Theory)*, volume 836 of *Lecture Notes in Computer Science*, pages 2–17, Uppsala, Sweden, August 1994. Springer-Verlag.

[165] L. Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33:103–111, 1990.

[166] R. van Glabbeek, S.A. Smolka, B. Steffen, and C.M.N. Tofts. Reactive, generative and stratified models of probabilistic processes. In *Fifth Annual Symposium on Logic in Computer Science (LICS'90)*, pages 130–141, Philadelphia, USA, June 1990. IEEE Computer Society Press.

[167] C. Verhoef. A congruence theorem for structured operational semantics with predicates and negative premises. *Nordic Journal of Computing*, 2:274–302, 1995.

[168] W. Vogler. *Modular Construction and Partial Order Semantics of Petri Nets*, volume 625 of *Lecture Notes in Computer Science*. Springer-Verlag, 1992.

[169] J. Vytopil, editor. *Proceedings of Symposium on Real-Time and Fault-Tolerant Systems (FTRTFT'92)*, volume 571 of *Lecture Notes in Computer Science*, Nijmegen, The Netherlands, January 1992. Springer-Verlag.

[170] G. Winskel. *The Formal Semantics of Programming Languages: An Introduction*. MIT Press, Cambridge, MA, 1993.

[171] G. Winskel and M. Nielsen. Models for concurrency. In S. Abramsky, D.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic Computer Science*, volume 4, pages 1–148. Oxford Science Publications, 1995.

[172] N. Wirth. Towards a discipline of real-time programming. *Communications of the ACM*, 20(8):577–583, 1977.

[173] W. Yi. CCS + time = an interleaving model for real time systems. In Albert et al. [2], pages 217–228.

Appendix A

Auxiliary Proof for Chapter 2

In this chapter we prove Proposition 2.4.19 which states that $\underline{\approx}_{\mathrm{a}} \subseteq \underline{\approx}$. Therefore, let $P, Q \in \mathcal{P}$ satisfying $P \underline{\approx}_{\mathrm{a}} Q$, i.e. $C_{PQ}[P] \approx_{\times} C_{PQ}[Q]$ by the definition of $\underline{\approx}_{\mathrm{a}}$. In the following we consider three situations, each illustrated by an accompanying diagram. The double arrows in the diagrams symbolize the naive prioritized weak transition relation. The left hand side of each diagram displays a sequence of transitions which we may choose. According to the definition of $\approx_{\times}$, matching sequences exist which mimic each step by a corresponding naive prioritized weak transition. From these transitions we may extract additional conditions which have to be satisfied according to $\mathsf{CCS}^{\mathsf{ch}}$ semantics. Those conditions are sufficient to conclude that $\underline{\approx}_{\mathrm{a}}$ is a prioritized weak bisimulation. Hence, $P \underline{\approx} Q$ as desired.

Situation 1

Let $P \xrightarrow{\tau:k} P'$ for some $P' \in \mathcal{P}$. First, let $C_{PQ}[P] \xrightarrow{\tau:0} P \,|\, H_L$ where $H_L \stackrel{\text{def}}{=} \tau:k.\mathbf{0} + d_L:0.H_{PQ} + D_L + e:k.H_{PQ} + \overline{b}:k.H_{PQ}$ for $b:k \equiv \tau:k$ and $L =_{\text{df}} \{\overline{c}:l \,|\, c:l \in (\mathcal{S}(P) \cup \mathcal{S}(Q)) \setminus \mathcal{I}\!\!\mathcal{L}^{<k}(P),\ l < k\}$.

Since $C_{PQ}[P] \approx_{\times} C_{PQ}[Q]$, we have $C_{PQ}[Q] \stackrel{\epsilon}{\Longrightarrow}_{\times} W$ for some $W \in \mathcal{P}$. We know that H_{PQ} has to perform a $\tau:0$-transition to H_L since the action $d_{L,b:k}:0$ is unique. However, Q may be able to execute several internal transitions to some state $\overline{Q} \in \mathcal{P}$, i.e. we have the following situation:

$$\exists s, t \in \mathbb{N}\ \forall 0 \leq i < s\ \forall 0 < j < t\ \exists Q_i, Q_s, Q_{s+j}, Q_{s+t} \in \mathcal{P}\ \exists l_i, l_s, l_{s+j} \in \mathbb{N}.\ \ Q_0 \equiv Q,\ Q_{s+t} \equiv \overline{Q}$$

and

1. $Q_i \,|\, H_{PQ} \xrightarrow{\tau:l_i} Q_{i+1} \,|\, H_{PQ}$
2. $Q_s \,|\, H_{PQ} \xrightarrow{\tau:l_s} Q_{s+1} \,|\, H_L$, and
3. $Q_{s+j} \,|\, H_L \xrightarrow{\tau:l_{s+j}} Q_{s+j+1} \,|\, H_L$.

According to the definition of our semantics, the following conditions must be satisfied.

- $l_i = 0$ and $l_s = 0$ since H_{PQ} can engage in a $\tau:0$-transition,
- $l_{s+j} \leq k$ because H_L can execute a $\tau:k$-transition, and
- $\tau \notin \mathcal{I}^{<l_{s+j}}(Q_{s+j})$ which implies $\mathcal{U}^{<l_{s+j}}(Q_{s+j}) \cap \overline{\mathcal{U}(H_L)} = \emptyset$; hence, $\mathcal{U}^{<l_{s+j}}(Q_{s+j}) \subseteq \mathcal{U}^{<k}(P)$ by our choice of L.

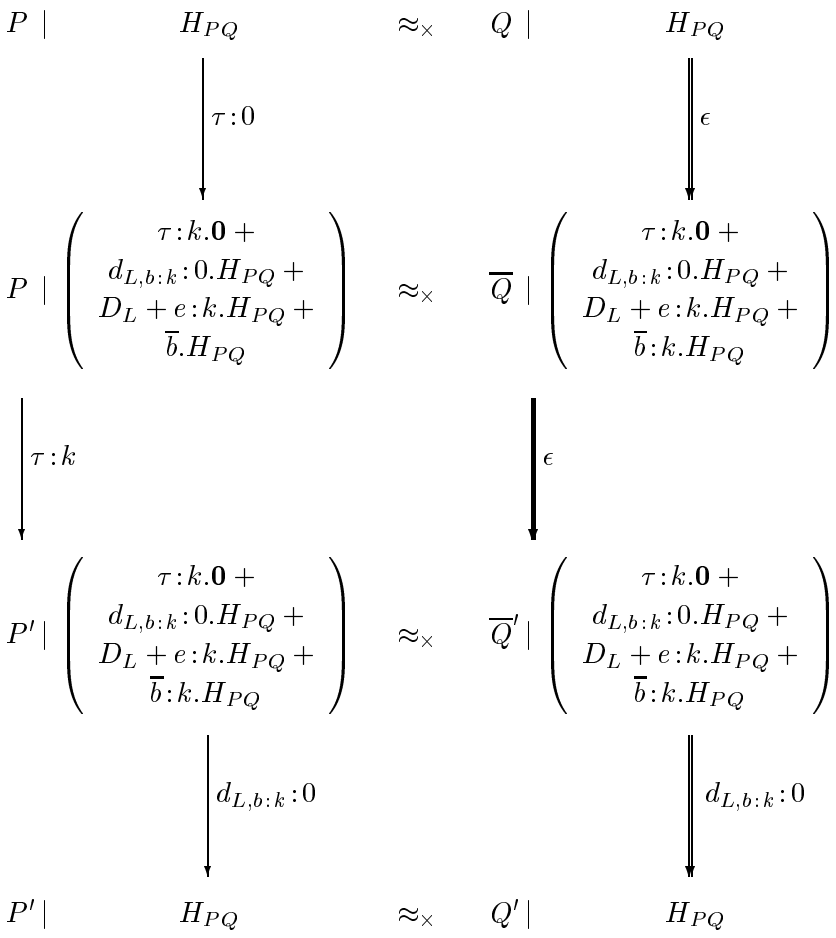

Figure A.1: Largest congruence proof – illustration of Situation (1)

In the second step, let $P \mid H_L \xrightarrow{\tau:k} P' \mid H_L$. Note that this transition is possible according to the semantics of $\mathsf{CCS}^{\mathsf{ch}}$ and the choice of L. Since $P \mid H_L \approx_\times \overline{Q} \mid H_L$, there exists some $\overline{W} \in \mathcal{P}$ such that $\overline{Q} \mid H_L \stackrel{\epsilon}{\Longrightarrow}_\times \overline{W}$ and $P' \mid H_L \approx_\times \overline{W}$. Now, observe that H_L cannot engage or participate in an internal transition yielding a process equivalent to $P' \mid H_L$ because the latter process is distinguished by its initial $d_{L,b:k}:0$-action. This leads to the following two possible cases:

1. $\overline{W} \equiv C_{PQ}[\overline{Q}]$, and we define $\overline{Q}' \equiv \overline{Q}$, or

2. $\overline{W} \equiv \overline{Q}' \,|\, H_L$ for some $\overline{Q}' \in \mathcal{P}$ satisfying $\overline{Q} \stackrel{\epsilon}{\Longrightarrow}_{\times} \overline{Q}'$.

The first case corresponds to an "idling" and does not need any special attention. In the second case we have the following situation according to the definition of the naive prioritized weak transition relation:

$$\exists \overline{s} \in \mathbb{N}\ \forall 0 \leq \overline{i} < \overline{s}\ \exists \overline{Q}_{\overline{i}}, \overline{Q}_{\overline{s}} \in \mathcal{P}\ \exists \overline{l}_{\overline{i}} \in \mathbb{N}.\ \overline{Q}_0 \equiv \overline{Q},\ \overline{Q}_{\overline{s}} \equiv \overline{Q}' \text{ such that } \overline{Q}_{\overline{i}} \stackrel{\tau:\overline{l}_{\overline{i}}}{\longrightarrow} \overline{Q}_{\overline{i}+1}\ .$$

Hence, $\tau \notin \mathcal{I}^{<\overline{l}_{\overline{i}}}(\overline{Q}_{\overline{i}} \,|\, H_L)$ according to $\mathsf{CCS}^{\mathsf{ch}}$ semantics and, therefore, $\overline{l}_{\overline{i}} \leq k$ and $\mathcal{U}^{<\overline{l}_{\overline{i}}}(\overline{Q}_{\overline{i}}) \cap \overline{\mathcal{U}(H_L)} = \emptyset$. Because of the choice of L we conclude $\mathcal{U}^{<\overline{l}_{\overline{i}}}(\overline{Q}_{\overline{i}}) \subseteq \mathcal{U}^{<k}(P)$. Additionally, $P' \,|\, H_L \approx_{\times} \overline{Q}' \,|\, H_L$ holds.

Finally, let $P' \,|\, H_L \stackrel{d_{L,b:k}:0}{\longrightarrow} C_{PQ}[P']$. In order for $\overline{Q}' \,|\, H_L$ to match this step with a naive prioritized weak $d_{L,b:k}:0$-transition to some $W' \in \mathcal{P}$ satisfying $C_{PQ}[P'] \approx_{\times} W'$, the process H_L has to perform a $d_{L,b:k}:0$-transition to the process H_{PQ}. Moreover, H_{PQ} cannot perform a $\tau:0$-transition because of the distinguished action $c:0$. However, $\overline{Q}$ may be able to perform a sequence of internal transitions to some process $Q' \in \mathcal{P}$, i.e.

$$\exists s', t' \in \mathbb{N}\ \forall 0 \leq i' < s'\ \forall 0 \leq j' < t'\ \exists \overline{Q}'_{i'}, \overline{Q}'_{s'+j'}, \overline{Q}'_{s'+t'} \in \mathcal{P}\ \exists l'_{i'}, l'_{s'+j'} \in \mathbb{N}$$

such that $\overline{Q}'_0 \equiv \overline{Q}'$, $\overline{Q}'_{s'+t'} \equiv Q'$, $l'_{s'} = 0$, and

1. $\overline{Q}'_{i'} \,|\, H_L \stackrel{\tau:l'_{i'}}{\longrightarrow} \overline{Q}'_{i'+1} \,|\, H_L$,
2. $\overline{Q}'_{s'} \,|\, H_L \stackrel{d_{L,b:k}:0}{\longrightarrow} \overline{Q}'_{s'} \,|\, H_{PQ}$, and
3. $\overline{Q}'_{s'+j'} \,|\, H_{PQ} \stackrel{\tau:l'_{s'+j'}}{\longrightarrow} \overline{Q}'_{s'+j'+1} \,|\, H_{PQ}$.

According to the definition of our semantics, the following conditions must be satisfied.

- $l'_{i'} \leq k$ since H_L can engage in a $\tau:k$-transition,
- $\tau \notin \mathcal{I}^{<l'_{i'}}(\overline{Q}'_{i'} \,|\, H_L)$ which implies $\mathcal{U}^{<l'_{i'}}(\overline{Q}'_{i'}) \cap \overline{\mathcal{U}(H_L)} = \emptyset$ and, due to the choice of L and $l'_{i'} \leq k$, $\mathcal{U}^{<l'_{i'}}(\overline{Q}'_{i'}) \subseteq \mathcal{U}^{<k}(P)$, and
- $l'_{s'+j'} = 0$ because H_{PQ} can execute a $\tau:0$-transition.

Moreover, we have $W' \equiv C_{PQ}[Q']$. Summarizing, we have shown that $Q \stackrel{\epsilon:k}{\Longrightarrow}_{L'} Q'$ for $L' = \mathcal{U}^{<k}(P)$ according to the definition of the prioritized weak transition relation. Because of $C_{PQ}[P'] \approx_{\times} C_{PQ}[Q']$, $\mathcal{S}(P') \subseteq \mathcal{S}(P)$, and $\mathcal{S}(Q') \subseteq \mathcal{S}(Q)$ we obtain $C_{P'Q'}[P'] \approx_{\times} C_{P'Q'}[Q']$, too. Therefore, $P' \underline{\approx}_{\mathrm{a}} Q'$.

Situation 2

Let $P \xrightarrow{a:k} P'$ for some process $P' \in \mathcal{P}$ and some action $a:k \in \mathcal{A} \setminus \{\tau:l \,|\, l \in \mathbb{N}\}$. As illustrated in Figure A.2, we let $C_{PQ}[P]$ perform a $\tau:0$-transition to the process $P \,|\, H_L$, where $H_L \stackrel{\text{def}}{=} \tau:k.\mathbf{0} + d_{L,a:k}:0.H_{PQ} + D_L + e:k.H_{PQ} + \overline{a}:k.H_{PQ}$ for $L =_{\text{df}} \{\overline{c}:l \,|\, c:l \in (\mathcal{S}(P) \cup \mathcal{S}(Q)) \setminus \mathcal{I}\!\!\mathcal{I}^{<k}(P),\ l < k\}$. Now, $P \,|\, H_L$ can perform a $\tau:k$-transition to $C_{PQ}[P']$ according to $\mathsf{CCS}^{\mathsf{ch}}$ semantics.

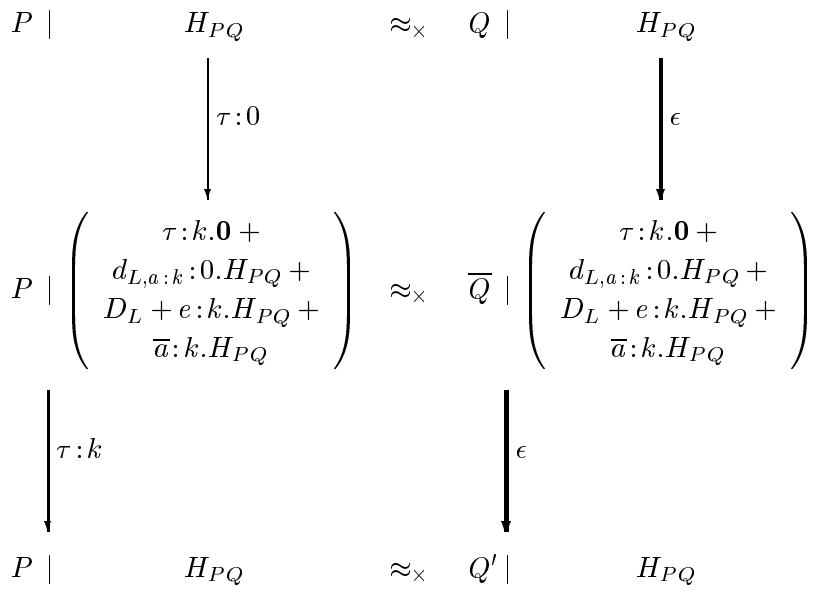

Figure A.2: Largest congruence proof – illustration of Situation (2)

Consider the first step. Since $C_{PQ}[P] \approx_\times C_{PQ}[Q]$, we have $C_{PQ}[Q] \stackrel{\epsilon}{\Longrightarrow}_\times W$ for some $W \in \mathcal{P}$. We know that H_{PQ} has to perform a $\tau:0$-transition to H_L since the action $d_{L,a:k}:0$ is unique. However, Q may be able to perform internal transitions to some state $\overline{Q} \in \mathcal{P}$, i.e. we have the following situation:

$$\exists s,t \in \mathbb{N}\ \forall 0 \le i \le s\ \forall 0 < j < t\ \exists Q_i, Q_{s+j}, Q_{s+t} \in \mathcal{P}\ \exists l_i, l_{s+j} \in \mathbb{N}.\ \ Q_0 \equiv Q,\ Q_{s+t} \equiv \overline{Q}$$

and

1. $Q_i \,|\, H_{PQ} \xrightarrow{\tau:l_i} Q_{i+1} \,|\, H_{PQ}$,
2. $Q_s \,|\, H_{PQ} \xrightarrow{\tau:l_s} Q_{s+1} \,|\, H_L$, and
3. $Q_{s+j} \,|\, H_L \xrightarrow{\tau:l_{s+j}} Q_{s+j+1} \,|\, H_L$.

According to the definition of our semantics, the following conditions must be satisfied.

- $l_i = l_s = 0$ since H_{PQ} can engage in a $\tau:0$-transition,
- $l_{s+j} \leq k$ because H_L possesses an initial $\tau:k$-transition, and
- $\tau \notin \mathcal{I}^{<l_{s+j}}(Q_{s+j} \mid H_L)$ which implies $\mathcal{IL}^{<l_{s+j}}(Q_{s+j}) \cap \overline{\mathcal{IL}(H_L)} = \emptyset$; hence, $\mathcal{IL}^{<l_{s+j}}(Q_{s+j}) \subseteq \mathcal{IL}^{<k}(P)$ by the choice of L and since $l_{s+j} \leq k$.

Now, we take a closer look at the second step. Since $P \mid H_L \approx_\times \overline{Q} \mid H_L$, there exists some $\overline{W}' \in \mathcal{P}$ satisfying $\overline{Q} \mid H_L \stackrel{\epsilon}{\Longrightarrow}_\times \overline{W}'$ and $C_{PQ}[P'] \approx_\times \overline{W}'$. We first argue that the matching transition must arise from a communication between $\overline{Q}$ and H_L. First, H_L must perform an $\overline{a}:k$-transition to H_{PQ} since only that process can engage in the distinguished $c:0$-transition. Because we have to match a $\tau:k$-transition, we may conclude that $\overline{Q} \stackrel{a:k}{\Longrightarrow}_\times \overline{Q}'$ for some $\overline{Q}' \in \mathcal{P}$, i.e.

$$\exists \overline{s}, \overline{t} \in \mathbb{N}\ \forall 0 \leq \overline{i} < \overline{s}\ \forall 0 < \overline{j} < \overline{t}\ \exists \overline{Q_{\overline{i}}}, \overline{Q_{\overline{s}}}, \overline{Q_{\overline{s}+\overline{j}}}, \overline{Q_{\overline{s}+\overline{t}}}, \in \mathcal{P}\ \exists \overline{l_{\overline{i}}}, \overline{l_{\overline{s}+\overline{j}}} \in \mathbb{N}.\ \ \overline{Q}_0 \equiv \overline{Q},\ \overline{Q_{\overline{s}+\overline{t}}} \equiv \overline{Q}'$$

and

1. $\overline{Q_{\overline{i}}} \mid H_L \stackrel{\tau:\overline{l_{\overline{i}}}}{\longrightarrow} \overline{Q_{\overline{i}+1}} \mid H_L$,
2. $\overline{Q_{\overline{s}}} \mid H_L \stackrel{\tau:k}{\longrightarrow} \overline{Q_{\overline{s}+1}} \mid H_{PQ}$, and
3. $\overline{Q_{\overline{s}+\overline{j}}} \mid H_{PQ} \stackrel{\tau:\overline{l_{\overline{s}+\overline{j}}}}{\longrightarrow} \overline{Q_{\overline{s}+\overline{j}+1}} \mid H_{PQ}$.

Moreover, we conclude that $\overline{W}' \equiv \overline{Q}' \mid H_{PQ}$. According to the definition of our semantics, the following conditions must be satisfied.

- $\overline{l_{\overline{i}}} \leq k$ since H_L can engage in a $\tau:k$-transition.
- $\tau \notin \mathcal{I}^{<\overline{l_{\overline{i}}}}(\overline{Q_{\overline{i}}} \mid H_L)$ which implies $\mathcal{IL}^{<\overline{l_{\overline{i}}}}(\overline{Q_{\overline{i}}}) \cap \overline{\mathcal{IL}(H_L)} = \emptyset$; hence $\mathcal{IL}^{<\overline{l_{\overline{i}}}}(\overline{Q_{\overline{i}}}) \subseteq \mathcal{IL}^{<k}(P)$,
- $\tau \notin \mathcal{I}^{<k}(\overline{Q_{\overline{s}}} \mid H_L)$ which implies $\mathcal{IL}^{<k}(\overline{Q_{\overline{s}}}) \subseteq \mathcal{IL}^{<k}(P)$, and
- $\overline{l}_{\overline{s}+\overline{j}} = 0$ because H_{PQ} can initially execute a $\tau:0$-transition.

According to the definition of the prioritized weak transition relation we have shown that $Q \stackrel{a:k}{\underset{L'}{\Longrightarrow}} Q'$ where $L' = \mathcal{IL}^{<k}(P)$. Since $C_{PQ}[P'] \approx_\times C_{PQ}[Q']$, $\mathcal{S}(P') \subseteq \mathcal{S}(P)$, and $\mathcal{S}(Q') \subseteq \mathcal{S}(Q)$ also $C_{P'Q'}[P'] \approx_\times C_{P'Q'}[Q']$ holds, i.e. $P' \underline{\approx}_{\mathrm{a}} Q'$, and this part of the proof is finished.

Situation 3

Here, we are going to establish Condition (1) of Definition 2.4.5. We choose the transition sequence displayed in Figure A.3, where L is defined in the previous situations, and $b:k \equiv \tau:k$ for some $k \leq \mathtt{min}$. Again, H_L denotes the process $\tau:k.\mathbf{0} + d_{L,\tau:k}:0.H_{PQ} + D_L + e:k.H_{PQ} + \tau:k.H_{PQ}$.

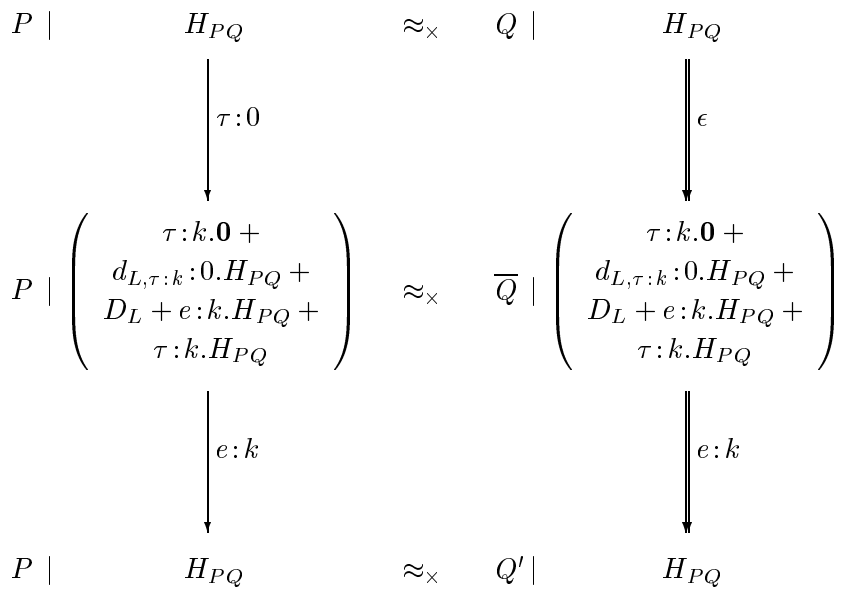

Figure A.3: Largest congruence proof – illustration of Situation (3)

Therefore, most proof parts are slight modifications of the proofs of the preceding situations. The interesting part of this situation is the matching of the $e:k$-transition. Let $Q'' \in \mathcal{P}$ be the process such that $\overline{Q} \stackrel{\epsilon}{\Longrightarrow}_\times Q'' \stackrel{\epsilon}{\Longrightarrow}_\times Q'$ and $Q'' \,|\, H_L \stackrel{e:k}{\longrightarrow} W$ for some $W \in \mathcal{P}$. According to $\mathsf{CCS}^{\mathsf{ch}}$ semantics we have the constraint $\tau \notin \mathcal{I}^{<k}(H_L \,|\, Q'')$ which implies $\mathcal{I}\!\!\mathcal{I}^{<k}(Q'') \cap \overline{\mathcal{I}\!\!\mathcal{I}(H_L)} = \emptyset$ and further $\mathcal{I}\!\!\mathcal{I}^{<k}(Q'') \subseteq \mathcal{I}\!\!\mathcal{I}^{<k}(P)$, as desired. Considering also the similarity to the situations discussed above, we conclude that $Q \stackrel{\epsilon:k}{\underset{L'}{\Longrightarrow}} Q'$ where $L' = \mathcal{I}\!\!\mathcal{I}^{<k}(P)$, $\mathcal{I}\!\!\mathcal{I}^{<k}(Q') \subseteq \mathcal{I}\!\!\mathcal{I}^{<k}(P)$, and $C_{P'Q'}[P'] \approx_\times C_{P'Q'}[Q']$, i.e. $P' \underline{\approx}_{\mathrm{a}} Q'$.

Summarizing, we have shown that $\underline{\approx}_{\mathrm{a}}$ is a prioritized weak bisimulation according to Definition 2.4.5. Therefore, $P \underline{\approx} Q$, as desired. □

Appendix B

Modeling LUN1 of the SCSI-2 Bus-Protocol

In this chapter of the appendix we present the model of the logical unit LUN1 of the SCSI-2 bus-protocol which needs to be modeled separately since the behaviors of LUN0 and LUN1 are not symmetric in the Arbitration phase. The asymmetries arise from the different priority values assigned to both devices. In the Arbitration phase, LUN0 has to check if LUN1 has set its id on the bus. If so, LUN0 has lost arbitration. However, LUN1 needs not to check if LUN0 has set its id on the bus because LUN0 is assigned to the lower SCSI id. Moreover, since we are assuming only two devices, there is no necessity for LUN1 to check any SCSI id asserted on the bus. The formal specification of LUN1, as accepted by the CWB-NC, is the following.

```
proc LUN1  =   t(start1):9.'relIO:0.(BusFree1 + GetSelected1)
             + t(start1):9.'setIO(obs_setIO):0.(BusFree1 + GetSelected1)
             + t:9.LUN1
             + GetSelected1

proc GetSelected1 = isATN:0.( isSEL:0.noBSY:0.readtarget1:0.'setBSY(obs_setBSY):9.
                              'release:0.'clear:0.noSEL:0.
                              (noIO:0.Target1 + isIO:0.Initiator1)
                            + noSEL:0.LUN1
                            )

* BusFree Phase

proc BusFree1 =   t(busfree):80.'setBSY(obs_setBSY):80.'setid1:0.Arbitrate1
                + isSEL(obs_isSEL):0.LUN1
                + isBSY(obs_isBSY):0.LUN1
```

```
* Arbitration and Selection Phase

proc Arbitrate1 =   noSEL:80.'setSEL(obs_setSEL):0.Selection1
                  + isSEL:80.LUN1

proc Selection1 = 'writetarget0:240.'setATN:9.'relBSY(obs_relBSY):18.
                  isBSY:80.'relSEL(obs_relSEL):9.t(begin_ITP):0.
                  (noIO:0.Initiator1 + isIO:0.Target1)

* Initiator and Target

proc Initiator1 = H1 [> noBSY(obs_noBSY):0.'relATN:0.LUN1

proc H1 =   t:9.'setATN(obs_setATN):9.H1
          + isREQ(obs_isREQ):9.
            ( noMSG:0.( noCD:0.(noIO:0.DataOutI1 + isIO:0.DataInI1)
                      + isCD:0.(noIO:0.CommandI1 + isIO:0.StatusI1)
                      )
            + isMSG:0.isCD:0.(noIO:0.MsgOutI1 + isIO:0.MsgInI1)
            )

proc Target1 =    (noIO:0.MsgOutT1 + isIO:0.'relATN:0.MsgInT1)
               [> noBSY:0.'relATN:0.LUN1

* MessageIn Phase

proc MsgInI1 =  isREQ:0.( readmsgIn:0.'setACK:0.noREQ:0.'relACK:0.MsgInI1
                        + readfinished:0.'setACK:0.noREQ:0.'relACK:0.H1
                        + readcomplete:0.'setACK:0.noREQ:0.'relACK:0.nil
                        + readdisconnect:0.'setACK:0.noREQ:0.'relACK:0.nil
                        )

proc MsgInT1  = 'setMSG:0.'setCD:0.'setIO(begin_MsgIn):0.t(begin_Phase):0.
                MsgInT1'

proc MsgInT1' =   'placemsgIn:0.'setREQ(obs_setREQ):9.
                  isACK(obs_isACK):0.'release:0.'relREQ(obs_relREQ):0.noACK:0.
                  MsgInT1'
                + 'placefinished:0.'setREQ(obs_setREQ):9.isACK(obs_isACK):0.
                  'release:0.'relREQ(obs_relREQ):0.noACK(end_Phase):0.
                  (MsgOutT1 + DataOutT1 + DataInT1 + CommandT1 + StatusT1)
                + 'sentcomplete:0.'setREQ:9.isACK(obs_isACK):0.'release:0.
                  'relREQ(obs_relREQ):0.noACK(end_Phase):0.t(end_ITP):0.
                  'relBSY(obs_relBSY):0.nil
                + 'sentdisconnect:0.'setREQ:9.isACK(obs_isACK):0.'release:0.
                  'relREQ(obs_relREQ):0.noACK(end_Phase):0.t(end_ITP):0.
                  'relBSY(obs_relBSY):0.nil
```

```
* MessageOut Phase

proc MsgOutI1  = isREQ:0.
                 ( 'placemsgOut:0.'setACK:9.noREQ:0.'release:0.
                   'relACK:0.MsgOutI1

                 + 'placefinished:0.'relATN:9.'setACK:0.noREQ:0.'release:0.
                   'relACK:0.H1
                 )

proc MsgOutT1  = isATN:0.'setMSG:0.'setCD:0.'relIO(begin_MsgOut):0.
                 t(begin_Phase):0.MsgOutT1'

proc MsgOutT1' = 'setREQ:0.isACK(obs_isACK):0.
                 ( readmsgOut:0.'relREQ(obs_relREQ):0.noACK(obs_noACK):0.MsgOutT1'
                 + readfinished:0.'relREQ(obs_relREQ):0.noACK:0.
                   ( t(end_Phase):0.
                     (MsgInT1 + DataOutT1 + DataInT1 + CommandT1 + StatusT1)
                   + t(more_read):0.MsgOutT1'
                   )
                 )

* Command Phase

proc CommandI1 = isREQ:0.( 'placecmd:0.'setACK:9.noREQ:0.'release:0.
                           'relACK:0.CommandI1
                         + 'placefinished:0.'setACK:9.noREQ:0.'release:0.
                           'relACK:0.H1
                         )

proc CommandT1  = 'relMSG:0.'setCD:0.'relIO(begin_Command):0.t(begin_Phase):0.
                  CommandT1'

proc CommandT1' = 'setREQ:0.isACK(obs_isACK):0.
                  ( readcmd:0.'relREQ(obs_relREQ):0.noACK:0.CommandT1'
                  + readfinished:0.'relREQ(obs_relREQ):0.noACK(end_Phase):0.
                    (MsgOutT1 + MsgInT1 + DataOutT1 + DataInT1 + StatusT1)
                  )

* DataIn Phase

proc DataInI1  = isREQ:0.( readdata:0.'setACK:0.noREQ:0.'relACK:0.DataInI1
                         + readfinished:0.'setACK:0.noREQ:0.'relACK:0.H1
                         )

proc DataInT1  = 'relMSG:0.'relCD:0.'setIO(begin_DataIn):0.t(begin_Phase):0.
                 DataInT1'
```

```
proc DataInT1' =   'placedata:0.'setREQ:9.isACK(obs_isACK):0.'release:0.
                   'relREQ(obs_relREQ):0.noACK:0.
                    DataInT1'
                 + 'placefinished:0.'setREQ:9.isACK(obs_isACK):0.'release:0.
                   'relREQ(obs_relREQ):0.noACK(end_Phase):0.
                   (MsgOutT1 + MsgInT1 + StatusT1)

* DataOut Phase

proc DataOutI1  = isREQ:0.
                  ( 'placedata:0.'setACK:9.noREQ:0.'release:0.'relACK:0.
                    DataOutI1
                  + 'placefinished:0.'setACK:9.noREQ:0.'release:0.'relACK:0.
                    H1
                  )

proc DataOutT1  = 'relMSG:0.'relCD:0.'relIO(begin_DataOut):0.t(begin_Phase):0.
                  DataOutT1'
proc DataOutT1' = 'setREQ:0.isACK(obs_isACK):0.
                  ( readdata:0.'relREQ(obs_relREQ):0.noACK:0.
                    DataOutT1'
                  + readfinished:0.'relREQ(obs_relREQ):0.noACK(end_Phase):0.
                    (MsgOutT1 + MsgInT1 + StatusT1)
                  )

* Status Phase

proc StatusI1 = readstatus:0.'setACK:0.noREQ:0.'relACK:0.H1

proc StatusT1 = 'relMSG:0.'setCD:0.'setIO(begin_Status):0.t(begin_Phase):0.
                'placestatus:0.'setREQ:9.isACK(obs_isACK):0.'release:0.
                'relREQ(obs_relREQ):0.noACK(end_Phase):0.
                (MsgOutT1 + MsgInT1)
```

Appendix C

Auxiliary Proof for Chapter 6

Here we prove Proposition 6.5.14, which states that $\underline{\approx}_{\mathrm{a}} \subseteq \underline{\approx}$, by establishing that $\underline{\approx}_{\mathrm{a}}$ is a distributed prioritized weak bisimulation. Therefore, let $P, Q \in \mathcal{P}$ satisfying $P \,\underline{\approx}_{\mathrm{a}}\, Q$, i.e. $C_{PQ}[P] \approx_{\times} C_{PQ}[Q]$ by the definition of $\underline{\approx}_{\mathrm{a}}$. The structure of the proof and its notation follow the lines of the proof presented in Appendix A.

Situation 1

Let $P \xrightarrow{m,\tau} P'$ for some $P' \in \mathcal{P}$ and some $m \in \mathcal{L}oc$. First, let $C_{PQ}[P] \xrightarrow{\tau} P \,|\, H_{LM}$ where $H_{LM} \stackrel{\text{def}}{=} (\underline{d}_{L,M,b}.H_{PQ} + \underline{D}_L + e.H_{PQ} + \overline{b}.(\underline{f}.H_{PQ} + \underline{D}_S)) \oplus \underline{D}_M$ for some $b \in \mathcal{S}(P) \cup \mathcal{S}(Q)$, $L =_{\mathrm{df}} \{\overline{c} \,|\, \underline{c} \in (\underline{\mathcal{S}(P)} \cup \underline{\mathcal{S}(Q)}) \setminus \underline{\Pi}_{[m]}(P)\}$, and $M =_{\mathrm{df}} \emptyset$.

Since $C_{PQ}[P] \approx_{\times} C_{PQ}[Q]$, we have $C_{PQ}[Q] \stackrel{\epsilon}{\Longrightarrow}_{\times} W$ for some $W \in \mathcal{P}$. The process H_{PQ} has to perform a $\underline{\tau}$-transition to H_{LM} since $\underline{d}_{L,M,b}$ is a unique action. However, Q may engage in some prioritized or unprioritized internal transitions to some $\overline{Q} \in \mathcal{P}$, i.e.

$$\exists s, t \in \mathbb{N}\ \forall 0 \leq i < s\ \ \forall 0 < j < t\ \ \exists Q_i, Q_s, Q_{s+j}, Q_{s+t} \in \mathcal{P}.\ \ Q_0 \equiv Q,\ Q_{s+t} \equiv \overline{Q}$$

and

1. $Q_i \,|\, H_{PQ} \xrightarrow{n_i \cdot L,\tau} Q_{i+1} \,|\, H_{PQ}$ for some $n_i \in \mathcal{L}oc$ or $Q_i \,|\, H_{PQ} \xrightarrow{\tau} Q_{i+1} \,|\, H_{PQ}$,
2. $Q_s \,|\, H_{PQ} \xrightarrow{n_s \cdot L,\tau} Q_{s+1} \,|\, H_{LM}$ for some $n_s \in \mathcal{L}oc$ or $Q_s \,|\, H_{PQ} \xrightarrow{\tau} Q_{s+1} \,|\, H_{LM}$, and
3. $Q_{s+j} \,|\, H_{LM} \xrightarrow{n_{s+j} \cdot L,\tau} Q_{s+j+1} \,|\, H_{LM}$ for some $n_{s+j} \in \mathcal{L}oc$ or
   $Q_{s+j} \,|\, H_{LM} \xrightarrow{\tau} Q_{s+j+1} \,|\, H_{LM}$.

According to the definition of our semantics, the following conditions must be satisfied if unprioritized τ-transitions are participating.

1. $\underline{\Pi}_{[n_i]}(Q_i) \cap \overline{\underline{\Pi}(H_{PQ})} = \emptyset$, which implies $\underline{\Pi}_{[n_i]}(Q_i) = \emptyset$,
2. $\underline{\Pi}_{[n_s]}(Q_s) \cap \overline{\underline{\Pi}(H_{PQ})} = \emptyset$, which implies $\underline{\Pi}_{[n_s]}(Q_s) = \emptyset$, and

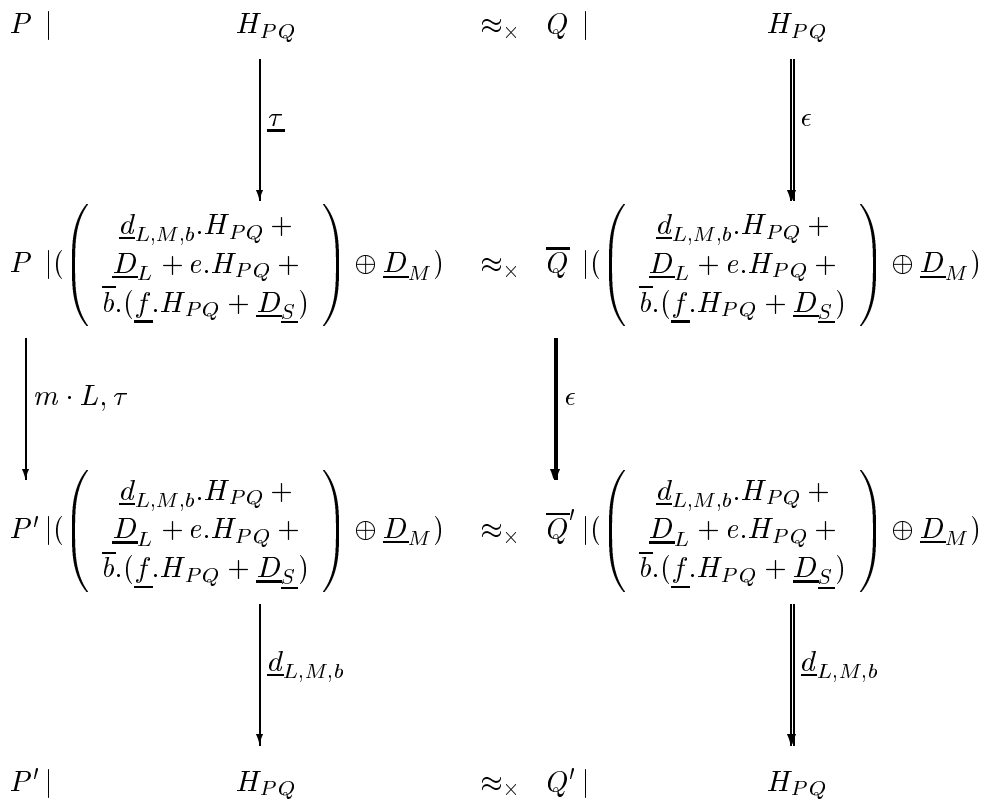

Figure C.1: Largest congruence proof – illustration of Situation (1)

3. $\underline{\mathit{If}}_{[n_{s+j}]}(Q_{s+j}) \cap \overline{\underline{\mathit{If}}(H_{LM})} = \emptyset$, which implies $\underline{\mathit{If}}_{[n_{s+j}]}(Q_{s+j}) \subseteq \underline{\mathit{If}}_{[m]}(P)$ by our choice of L and M.

In the second step, let $P \mid H_{LM} \xrightarrow{m \cdot L, \tau} P' \mid H_{LM}$, which is a possible transition according to $\mathsf{CCS}^{\mathsf{prio}}$ semantics and the choice of L and M. Since $P \mid H_{LM} \approx_\times \overline{Q} \mid H_{LM}$, we know of the existence of some $\overline{W} \in \mathcal{P}$ such that $\overline{Q} \mid H_{LM} \stackrel{\epsilon}{\Longrightarrow}_\times \overline{W}$ and $P' \mid H_{LM} \approx_\times \overline{W}$. Now, observe that H_{LM} has no τ-transitions. Moreover, a communication involving H_{LM} yielding a state which has the distinguished action $\underline{f}$ enabled and, therefore, cannot be equivalent to $P' \mid H_{LM}$. This leads to the following two possible cases:

1. $\overline{W} \equiv C_{PQ}[\overline{Q}]$, and we define $\overline{Q}' \equiv \overline{Q}$, or
2. $\overline{W} \equiv \overline{Q}' \mid H_{LM}$ for some $\overline{Q}' \in \mathcal{P}$ satisfying $\overline{Q} \stackrel{\epsilon}{\Longrightarrow}_\times \overline{Q}'$.

The first case corresponds to "idling" and needs no special attention. In the second case we have the following situation according to the definition of the naive distributed prioritized weak transition relation:

$$\exists \overline{s} \in \mathbb{N}\ \forall 0 \leq \overline{i} < \overline{s}\ \exists \overline{Q_i}, \overline{Q_{\overline{s}}} \in \mathcal{P}.\ \overline{Q}_0 \equiv \overline{Q},\ \overline{Q_{\overline{s}}} \equiv \overline{Q}'$$

such that $\overline{Q}_{\bar{i}} \xrightarrow{\overline{n}_{\bar{i}},\tau} \overline{Q}_{\bar{i}+1}$ for some $\overline{n}_{\bar{i}} \in \mathcal{L}oc$ or $\overline{Q}_{\bar{i}} \xrightarrow{\tau} \overline{Q}_{\bar{i}+1}$.

If $\overline{Q}_{\bar{i}} \,|\, H_{LM} \xrightarrow{\overline{n}_{\bar{i}}\cdot L,\tau} \overline{Q}_{\bar{i}+1} \,|\, H_{LM}$ then $\underline{\mathcal{IL}}_{[\overline{n}_{\bar{i}}]}(\overline{Q}_{\bar{i}}) \cap \overline{\underline{\mathcal{IL}}(H_{LM})} = \emptyset$ according to $\mathsf{CCS}^{\mathsf{prio}}$ semantics. Because of the choice of L and M we obtain $\underline{\mathcal{IL}}_{[\overline{n}_{\bar{i}}]}(\overline{Q}_{\bar{i}}) \subseteq L'$ for $L' = \underline{\mathcal{IL}}_{[m]}(P)$, i.e. $\overline{Q}_{\bar{i}} \xrightarrow[L']{\overline{n}_{\bar{i}},\tau} \overline{Q}_{\bar{i}+1}$, and also $P' \,|\, H_{LM} \approx_{\times} \overline{Q}' \,|\, H_{LM}$ holds.

Finally, let $P' \,|\, H_{LM} \xrightarrow{\underline{d}_{L,M,b}} C_{PQ}[P']$. The process $\overline{Q}' \,|\, H_{LM}$ has to match this step with a naive distributed prioritized weak $\underline{d}_{L,M,b}$-transition to some $W' \in \mathcal{P}$ satisfying $C_{PQ}[P'] \approx_{\times} W'$. Thus, H_{LM} has to perform a $\underline{d}_{L,M,b}$-transition to process H_{PQ}. Observe that H_{PQ} cannot perform a $\underline{\tau}$-transition since $\underline{c}$ is a distinguished action. However, $\overline{Q}$ may be able to engage in a sequence of prioritized and unprioritized internal transitions to some $Q' \in \mathcal{P}$, i.e.

$$\exists s', t' \in \mathbb{N}\ \forall 0 \leq i' < s'\ \forall 0 < j' < t'\ \exists \overline{Q}'_{i'}, \overline{Q}'_{s'}, \overline{Q}'_{s'+j'}, \overline{Q}'_{s'+t'} \in \mathcal{P}.\ \overline{Q}'_0 \equiv \overline{Q}',\ \overline{Q}'_{s'+t'} \equiv Q'$$

and

1. $\overline{Q}'_{i'} \,|\, H_{LM} \xrightarrow{\overline{n}'_{i'}\cdot L,\tau} \overline{Q}'_{i'+1} \,|\, H_{LM}$ for some $\overline{n}'_{i'} \in \mathcal{L}oc$ or $\overline{Q}'_{i'} \,|\, H_{LM} \xrightarrow{\tau} \overline{Q}'_{i'+1} \,|\, H_{LM}$,
2. $\overline{Q}'_{s'} \,|\, H_{LM} \xrightarrow{\underline{d}_{L,M,b}} \overline{Q}'_{s'} \,|\, H_{PQ}$, and
3. $\overline{Q}'_{s'+j'} \,|\, H_{PQ} \xrightarrow{\overline{n}'_{s'+j'}\cdot L,\tau} \overline{Q}'_{s'+j'+1} \,|\, H_{PQ}$ for some $\overline{n}'_{s'+j'} \in \mathcal{L}oc$ or $\overline{Q}'_{s'+j'} \,|\, H_{PQ} \xrightarrow{\tau} \overline{Q}'_{s'+j'+1} \,|\, H_{PQ}$.

According to the definition of our semantics, the following conditions must be satisfied with respect to unprioritized τ-transitions.

1. $\underline{\mathcal{IL}}_{[\overline{n}'_{i'}]}(\overline{Q}'_{i'}) \cap \overline{\underline{\mathcal{IL}}(H_{LM})} = \emptyset$, which implies $\underline{\mathcal{IL}}_{[\overline{n}'_{i'}]}(\overline{Q}'_{i'}) \subseteq \underline{\mathcal{IL}}_{[m]}(P)$, and
2. $\underline{\mathcal{IL}}_{[\overline{n}'_{s'+j'}]}(\overline{Q}'_{s'+j'}) \cap \overline{\underline{\mathcal{IL}}(H_{PQ})} = \emptyset$, which implies $\underline{\mathcal{IL}}_{[\overline{n}'_{s'+j'}]}(\overline{Q}'_{s'+j'}) = \emptyset$.

Additionally, we have $W' \equiv C_{PQ}[Q']$. Summarizing, $Q \overset{\epsilon}{\underset{L'}{\Longrightarrow}} Q'$ according to the definition of the distributed prioritized weak transition relation. Since $C_{PQ}[P'] \approx_{\times} C_{PQ}[Q']$, $\underline{\mathcal{S}(P')} \subseteq \underline{\mathcal{S}(P)}$, and $\underline{\mathcal{S}(Q')} \subseteq \underline{\mathcal{S}(Q)}$, we also obtain $C_{P'Q'}[P'] \approx_{\times} C_{P'Q'}[Q']$. Hence, $P' \underline{\approx}_{\mathrm{a}} Q'$.

Situation 2

Let $P \xrightarrow{m,a} P'$ for some process $P' \in \mathcal{P}$, some location $m \in \mathcal{L}oc$, and some action $a \in A \setminus \{\tau\}$. As shown in Figure C.2, $C_{PQ}[P]$ may perform a $\underline{\tau}$-transition to $P \,|\, H_{LM}$, where $H_{LM} \overset{\mathrm{def}}{=} (\underline{d}_{L,M,b}.H_{PQ} + \underline{D}_L + e.H_{PQ} + \overline{b}.(\underline{f}.H_{PQ} + \underline{D}_S)) \oplus \underline{D}_M$ for $b \equiv a$, $L =_{\mathrm{df}} \{\overline{\underline{c}} \,|\, \underline{c} \in (\underline{\mathcal{S}(P)} \cup \underline{\mathcal{S}(Q)}) \setminus \underline{\mathcal{IL}}(P)\}$, and $M =_{\mathrm{df}} \{\overline{\underline{c}} \,|\, \underline{c} \in (\underline{\mathcal{S}(P)} \cup \underline{\mathcal{S}(Q)}) \setminus \underline{\mathcal{IL}}_{[m]}(P)\} \setminus L$. Moreover, $P \,|\, H_{LM}$ can engage in a τ-transition from location $\langle m \cdot L, o \cdot L \cdot R\rangle$ to $P' \,|\, (\underline{f}.H_{PQ} + \underline{D}_S)$ according to $\mathsf{CCS}^{\mathsf{prio}}$ semantics. Here, the location $o \in \{l, r\}^*$ does not need to be specified more precisely because priorities on different sides of the summation operator are comparable.

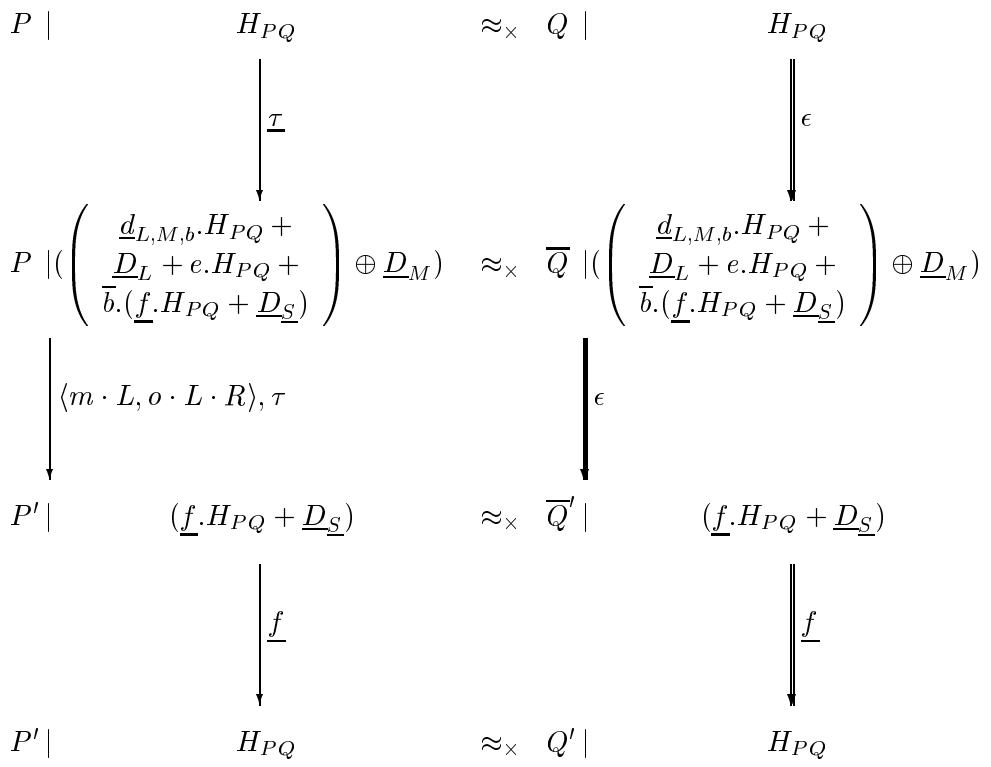

Figure C.2: Largest congruence proof – illustration of Situation (2)

Since $C_{PQ}[P] \approx_\times C_{PQ}[Q]$ there exists some $W \in \mathcal{P}$ satisfying $C_{PQ}[Q] \stackrel{\epsilon}{\Longrightarrow}_\times W$. Moreover, H_{PQ} has to perform a $\underline{\tau}$-transition to H_{LM} since $\underline{d}_{L,M,b}$ is a distinguished action. However, Q may engage in some prioritized or unprioritized internal transitions to some $\overline{Q} \in \mathcal{P}$, i.e.

$$\exists s,t \in \mathbb{N}\ \forall 0 \leq i < s\ \ \forall 0 < j < t\ \ \exists Q_i, Q_s, Q_{s+j}, Q_{s+t} \in \mathcal{P}.\ \ Q_0 \equiv Q,\ Q_{s+t} \equiv \overline{Q}$$

and

1. $Q_i \mid H_{PQ} \stackrel{n_i \cdot L, \tau}{\longrightarrow} Q_{i+1} \mid H_{PQ}$ for some $n_i \in \mathcal{L}oc$ or $Q_i \mid H_{PQ} \stackrel{\tau}{\rightarrow} Q_{i+1} \mid H_{PQ}$,
2. $Q_s \mid H_{PQ} \stackrel{n_s \cdot L, \tau}{\longrightarrow} Q_{s+1} \mid H_{LM}$ for some $n_s \in \mathcal{L}oc$ or $Q_s \mid H_{PQ} \stackrel{\tau}{\rightarrow} Q_{s+1} \mid H_{LM}$, and
3. $Q_{s+j} \mid H_{LM} \stackrel{n_{s+j} \cdot L, \tau}{\longrightarrow} Q_{s+j+1} \mid H_{LM}$ for some $n_{s+j} \in \mathcal{L}oc$ or
   $Q_{s+j} \mid H_{LM} \stackrel{\tau}{\rightarrow} Q_{s+j+1} \mid H_{LM}$.

According to the definition of our semantics, the following conditions must be satisfied if τ-transitions are involved.

1. $\underline{\Pi}_{[n_i]}(Q_i) \cap \overline{\underline{\Pi}(H_{PQ})} = \emptyset$, which implies $\underline{\Pi}_{[n_i]}(Q_i) = \emptyset$,
2. $\underline{\Pi}_{[n_s]}(Q_s) \cap \overline{\underline{\Pi}(H_{PQ})} = \emptyset$, which implies $\underline{\Pi}_{[n_s]}(Q_s) = \emptyset$, and

3. $\underline{\mathcal{IL}}_{[n_{s+j}]}(Q_{s+j}) \cap \overline{\underline{\mathcal{IL}}(H_{LM})} = \emptyset$, which implies $\underline{\mathcal{IL}}_{[n_{s+j}]}(Q_{s+j}) \subseteq \underline{\mathcal{IL}}_{[m]}(P)$.

Regarding the second step, we conclude from $P \,|\, H_{LM} \approx_\times \overline{Q} \,|\, H_{LM}$ that there exists some $\overline{W}' \in \mathcal{P}$ satisfying $\overline{Q} \,|\, H_{LM} \stackrel{\epsilon}{\Longrightarrow}_\times \overline{W}'$ and $C_{PQ}[P'] \approx_\times \overline{W}'$. It is easy to see that the matching transition must arise from a communication between $\overline{Q}$ and H_{LM}. In fact, H_{LM} has to engage in an $\overline{a}$-transition to $\underline{f}.H_{PQ} + \underline{D_S}$ since only this process can engage in the distinguished $\underline{f}$-transition. Because we have to match a τ-transition, we may conclude that $\overline{Q} \stackrel{a}{\Longrightarrow}_\times \overline{Q}'$ for some $\overline{Q}' \in \mathcal{P}$, i.e.

$$\exists \overline{s}, \overline{t} \in \mathbb{N}\ \forall 0 \leq \overline{i} < \overline{s}\ \forall 0 < \overline{j} < \overline{t}\ \exists \overline{Q_{\overline{i}}}, \overline{Q_{\overline{s}}}, \overline{Q_{\overline{s}+\overline{j}}}, \overline{Q_{\overline{s}+\overline{t}}} \in \mathcal{P}.\ \overline{Q_0} \equiv \overline{Q},\ \overline{Q_{\overline{s}+\overline{t}}} \equiv \overline{Q}'$$

such that

1. $\overline{Q_{\overline{i}}} \,|\, H_{LM} \stackrel{\overline{n_{\overline{i}}} \cdot L, \tau}{\longrightarrow} \overline{Q_{\overline{i}+1}} \,|\, H_{LM}$ for some $\overline{n_{\overline{i}}} \in \mathcal{L}oc$ or $\overline{Q_{\overline{i}}} \,|\, H_{LM} \stackrel{\tau}{\Rightarrow} \overline{Q_{\overline{i}+1}} \,|\, H_{LM}$,
2. $\overline{Q_{\overline{s}}} \,|\, H_{LM} \stackrel{\langle \overline{n_{\overline{s}}} \cdot L, o \cdot L \cdot R \rangle, \tau}{\longrightarrow} \overline{Q_{\overline{s}+1}} \,|\, (\underline{f}.H_{PQ} + \underline{D_S})$ for some $\overline{n_{\overline{s}}} \in \mathcal{L}oc$, and
3. $\overline{Q_{\overline{s}+\overline{j}}} \,|\, (\underline{f}.H_{PQ} + \underline{D_S}) \stackrel{\overline{n}_{\overline{s}+\overline{j}} \cdot L, \tau}{\longrightarrow} \overline{Q}_{\overline{s}+\overline{j}+1} \,|\, (\underline{f}.H_{PQ} + \underline{D_S})$ for some $\overline{n}_{\overline{s}+\overline{j}} \in \mathcal{L}oc$ or $\overline{Q_{\overline{s}+\overline{j}}} \,|\, (\underline{f}.H_{PQ} + \underline{D_S}) \stackrel{\tau}{\Rightarrow} \overline{Q}_{\overline{s}+\overline{j}+1} \,|\, (\underline{f}.H_{PQ} + \underline{D_S})$.

We also conclude that $\overline{W}' \equiv \overline{Q}' \,|\, (\underline{f}.H_{PQ} + \underline{D_S})$. The following conditions must be satisfied according to the definition of our semantics if not a $\underline{\tau}$-transition is chosen.

1. $\underline{\mathcal{IL}}_{[\overline{n}_{\overline{i}}]}(\overline{Q}_{\overline{i}}) \cap \overline{\underline{\mathcal{IL}}(H_{LM})} = \emptyset$, which implies $\underline{\mathcal{IL}}_{[\overline{n}_{\overline{i}}]}(\overline{Q}_{\overline{i}}) \subseteq \underline{\mathcal{IL}}_{[m]}(P)$,
2. $\underline{\mathcal{IL}}_{[\overline{n}_{\overline{s}}]}(\overline{Q}_{\overline{s}}) \cap \overline{\underline{\mathcal{IL}}(H_{LM})} = \emptyset$ and $\underline{\mathcal{IL}}_{[o]}(H_{LM}) \cap \overline{\underline{\mathcal{IL}}(\overline{Q}_{\overline{s}})} = \emptyset$, which implies $\underline{\mathcal{IL}}_{[\overline{n}_{\overline{s}}]}(\overline{Q}_{\overline{s}}) \subseteq \underline{\mathcal{IL}}_{[m]}(P)$ and $\underline{\mathcal{IL}}(\overline{Q}_{\overline{s}}) \subseteq \underline{\mathcal{IL}}(P)$, respectively, and
3. $\underline{\mathcal{IL}}_{[\overline{n}_{\overline{s}+\overline{j}}]}(\overline{Q}_{\overline{s}+\overline{j}}) \cap \overline{\underline{\mathcal{IL}}(\underline{f}.H_{PQ} + \underline{D_S})} = \emptyset$, which implies $\underline{\mathcal{IL}}_{[\overline{n}_{\overline{s}+\overline{j}}]}(\overline{Q}_{\overline{s}+\overline{j}}) = \emptyset$.

Let $P' \,|\, (\underline{f}.H_{PQ} + \underline{D_S}) \stackrel{\underline{f}}{\Rightarrow} C_{PQ}[P']$. Since $P' \,|\, (\underline{f}.H_{PQ} + \underline{D_S}) \approx_\times \overline{Q}' \,|\, (\underline{f}.H_{PQ} + \underline{D_S})$ we know of the existence of some $W' \in \mathcal{P}$ such that $\overline{Q}' \,|\, (\underline{f}.H_{PQ} + \underline{D_S}) \stackrel{\underline{f}}{\Longrightarrow}_\times W'$ and $C_{PQ}[P'] \approx_\times W'$. Additionally, $\underline{f}$ and $\underline{c}$ are distinguished actions, i.e. $\underline{f}.H_{PQ} + \underline{D_S}$ has to perform its $\underline{f}$-transition to process H_{PQ}. However, $\overline{Q}'$ may be able to engage in some prioritized or unprioritized internal transitions to some $Q' \in \mathcal{P}$, i.e.

$$\exists s', t' \in \mathbb{N}\ \forall 0 \leq i' < s'\ \forall 0 < j' < t'\ \exists \overline{Q}'_{i'}, \overline{Q}'_{s'}, \overline{Q}'_{s'+j'}, \overline{Q}'_{s'+t'} \in \mathcal{P}.\ \overline{Q}'_0 \equiv \overline{Q}',\ \overline{Q}'_{s'+t'} \equiv Q'$$

and

1. $\overline{Q}'_{i'} \,|\, (\underline{f}.H_{PQ} + \underline{D_S}) \stackrel{\overline{n}'_{i'} \cdot L, \tau}{\longrightarrow} \overline{Q}'_{i'+1} \,|\, (\underline{f}.H_{PQ} + \underline{D_S})$ for some $\overline{n}'_{i'} \in \mathcal{L}oc$ or $\overline{Q}'_{i'} \,|\, (\underline{f}.H_{PQ} + \underline{D_S}) \stackrel{\tau}{\Rightarrow} \overline{Q}'_{i'+1} \,|\, (\underline{f}.H_{PQ} + \underline{D_S})$,
2. $\overline{Q}'_{s'} \,|\, (\underline{f}.H_{PQ} + \underline{D_S}) \stackrel{\underline{f}}{\Rightarrow} \overline{Q}'_{s'+1} \,|\, H_{PQ}$, and

3. $\overline{Q}'_{s'+j'} \mid H_{PQ} \xrightarrow{\overline{n}'_{s'+j'}\cdot L,\tau} \overline{Q}'_{s'+j'+1} \mid H_{PQ}$ for some $\overline{n}'_{s'+j'} \in \mathcal{L}oc$ or
$\overline{Q}'_{s'+j'} \mid H_{PQ} \xRightarrow{\tau} \overline{Q}'_{s'+j'+1} \mid H_{PQ}$.

If τ-transitions are involved, the following conditions must be satisfied.

1. $\underline{\Pi}_{[\overline{n}'_{i'}]}(\overline{Q}'_{i'}) \cap \overline{\underline{\Pi}(\underline{f}.H_{PQ} + \underline{D_S})} = \emptyset$, which implies $\underline{\Pi}_{[\overline{n}'_{i'}]}(\overline{Q}'_{i'}) = \emptyset$, and
2. $\underline{\Pi}_{[\overline{n}'_{s'+j'}]}(\overline{Q}'_{s'+j'}) \cap \overline{\underline{\Pi}(H_{PQ})} = \emptyset$, which implies $\underline{\Pi}_{[\overline{n}'_{s'+j'}]}(\overline{Q}'_{s'+j'}) = \emptyset$.

By the definition of the distributed prioritized weak transition relation we have established $Q \overset{a}{\underset{L',M'}{\Longrightarrow}} Q'$ where $L' = \underline{\Pi}_{[m]}(P)$ and $M' = \underline{\Pi}(P)$. Since $C_{PQ}[P'] \approx_\times C_{PQ}[Q']$, $\underline{\mathcal{S}(P')} \subseteq \underline{\mathcal{S}(P)}$, and $\underline{\mathcal{S}(Q')} \subseteq \underline{\mathcal{S}(Q)}$, also $C_{P'Q'}[P'] \approx_\times C_{P'Q'}[Q']$ holds, i.e. $P' \underline{\approx}_{\mathrm{a}} Q'$, and this part of the proof is finished. After discussing the previous situations carefully, we omit the details of similar parts in the remaining situations.

Situation 3

Let $P \xRightarrow{\underline{\alpha}} P'$ for some process $P' \in \mathcal{P}$ and some action $\underline{\alpha} \in \underline{A}$. We first choose the transition $C_{PQ}[P] \xRightarrow{\tau} P \mid H_{LM}$ where $H_{LM} \overset{\text{def}}{=} (\underline{d}_{L,M,b}.H_{PQ} + \underline{D}_L + e.H_{PQ} + \overline{b}.(\underline{f}.H_{PQ} + \underline{D_S})) \oplus \underline{D}_M$ for some $b \in \mathcal{S}(P) \cup \mathcal{S}(Q)$, $L =_{\mathrm{df}} \underline{S}$, and $M =_{\mathrm{df}} \emptyset$. Analogously to Situation (1) we may conclude the existence of some process $\overline{Q} \in \mathcal{P}$ such that $C_{PQ}[Q] \overset{\epsilon}{\Longrightarrow} \overline{Q} \mid H_{LM}$ and $P \mid H_{LM} \approx_\times \overline{Q} \mid H_{LM}$. In the second step let $P \mid H_{LM} \xRightarrow{\underline{\alpha}} P' \mid H_{LM}$. Since $P \mid H_{LM} \approx_\times \overline{Q} \mid H_{LM}$ we have $\overline{Q} \mid H_{LM} \overset{\hat{\underline{\alpha}}}{\Longrightarrow}_\times \overline{W}$ and $C_{PQ}[P'] \approx_\times \overline{W}$ for some $\overline{W} \in \mathcal{P}$. Because of the distinguished action $\underline{d}_{L,M,b}$ we are in the following situation:

$$\exists s,t \in \mathbb{N}\ \forall 0 \le i < s\ \forall 0 < j < t\ \exists Q_i, Q_s, Q_{s+j}, Q_{s+t} \in \mathcal{P}.\ Q_0 \equiv \overline{Q},\ Q_{s+t} \equiv \overline{Q}'$$

such that

1. $Q_i \mid H_{LM} \xrightarrow{n_i\cdot L,\tau} Q_{i+1} \mid H_{LM}$ for some $n_i \in \mathcal{L}oc$ or $Q_i \mid H_{LM} \xRightarrow{\tau} Q_{i+1} \mid H_{LM}$,
2. $Q_s \mid H_{LM} \xRightarrow{\underline{\alpha}} Q_{s+1} \mid H_{PQ}$ or, if $\underline{\alpha} \equiv \underline{\tau}$, $Q_s \equiv Q_{s+1}$, and
3. $Q_{s+j} \mid H_{PQ} \xrightarrow{n_{s+j}\cdot L,\tau} Q_{s+j+1} \mid H_{PQ}$ for some $n_{s+j} \in \mathcal{L}oc$ or
$Q_{s+j} \mid H_{PQ} \xRightarrow{\tau} Q_{s+j+1} \mid H_{PQ}$.

According to the definition of our semantics, the following conditions must be satisfied with respect to unprioritized τ-transitions.

1. $\underline{\Pi}_{[n_i]}(Q_i) \cap \overline{\underline{\Pi}(H_{LM})} = \emptyset$, which implies $\underline{\Pi}_{[n_i]}(Q_i) = \emptyset$,
2. $\underline{\Pi}_{[n_{s+j}]}(Q_{s+j}) \cap \overline{\underline{\Pi}(H_{PQ})} = \emptyset$, which implies $\underline{\Pi}_{[n_{s+j}]}(Q_{s+j}) = \emptyset$.

Therefore, we have $\overline{Q} \overset{\epsilon}{\Longrightarrow} \overline{Q}'$ and $\overline{W} \equiv \overline{Q}' \mid H_{LM}$. Finally, let $P' \mid H_{LM} \xrightarrow{\underline{d}_{L,M,b}} C_{PQ}[P']$. Similar to Situation (1) we can conclude the existence of some process $Q' \in \mathcal{P}$ satisfying $\overline{Q}' \overset{\epsilon}{\Longrightarrow} Q'$ and $C_{PQ}[P'] \approx_\times C_{PQ}[Q']$. Hence, we have shown $Q \overset{\hat{\underline{\alpha}}}{\Longrightarrow} Q'$. Since $C_{PQ}[P'] \approx_\times C_{PQ}[Q']$, $\underline{\mathcal{S}(P')} \subseteq \underline{\mathcal{S}(P)}$, and $\underline{\mathcal{S}(Q')} \subseteq \underline{\mathcal{S}(Q)}$, also $C_{P'Q'}[P'] \approx_\times C_{P'Q'}[Q']$ can be established. Thus, $P' \underline{\approx}_{\mathrm{a}} Q'$ holds.

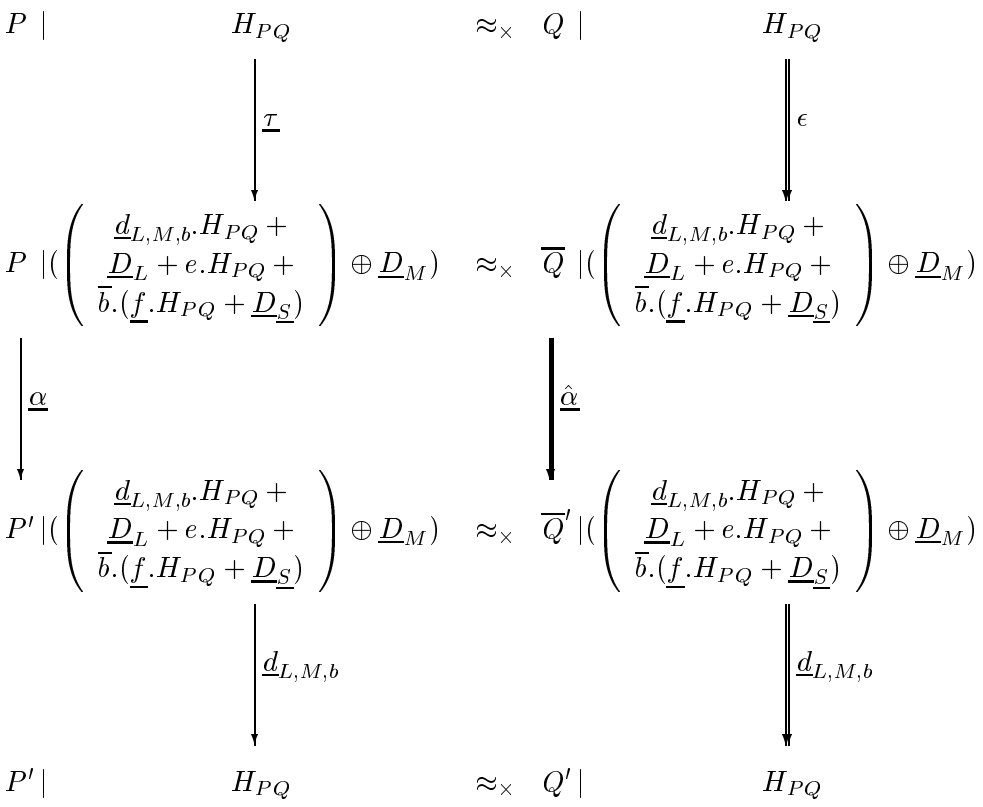

Figure C.3: Largest congruence proof – illustration of Situation (3)

Situation 4

It remains to establish Condition (1) of Definition 6.5.4. Therefore, we take a look at the transition sequence displayed in Figure C.4, where $L =_{\mathrm{df}} \underline{S} \setminus \overline{\underline{\Pi}(P)}$, $M =_{\mathrm{df}} \underline{S}$, $b \in \mathcal{S}(P) \cup \mathcal{S}(Q)$ arbitrary, and $H_{LM} \stackrel{\mathrm{def}}{=} (\underline{d}_{L,M,b}.H_{PQ} + \underline{D}_L + e.H_{PQ} + \overline{b}.(\underline{f}.H_{PQ} + \underline{D}_{\underline{S}})) \oplus \underline{D}_M$. Observe that in each step in Figure C.4 the prioritized initial actions in our context are always the complete universe $\underline{S}$. This is important in order to conclude the following property. Whenever the left parallel component at the right side engages in an unprioritized τ-transition, the corresponding initial action set is empty. However, the choice of L and the use of the *distributed* summation operator in the context ensure that the e-transition on the left side in Figure C.4 is possible.

The second step drawn in Figure C.4 is the only one which requires our attention. Let $Q'' \in \mathcal{P}$ be the process such that $\overline{Q} \stackrel{\epsilon}{\Longrightarrow}_\times Q'' \stackrel{\epsilon}{\Longrightarrow}_\times Q'$ and $Q'' \mid H_{LM} \stackrel{o \cdot L \cdot R, e}{\longrightarrow} W$ for some $W \in \mathcal{P}$. The location $o \in \{l, r\}^*$ needs not to be specified more precisely since priorities on different sides of the summation operator are comparable. According to $\mathsf{CCS}^{\mathsf{prio}}$ semantics $\underline{\Pi}_{[o \cdot L]}(H_{LM}) \cap \overline{\underline{\Pi}(Q'')} = \emptyset$ holds which implies $\underline{\Pi}(Q'') \subseteq \underline{\Pi}(P)$, as desired. Hence, $Q \stackrel{\epsilon}{\Longrightarrow} Q'' \stackrel{\epsilon}{\Longrightarrow} Q'$, $\underline{\Pi}(Q'') \subseteq \underline{\Pi}(P)$, and $C_{P'Q'}[P'] \approx_\times C_{P'Q'}[Q']$, i.e. $P' \underline{\approx}_{\mathrm{a}} Q'$.

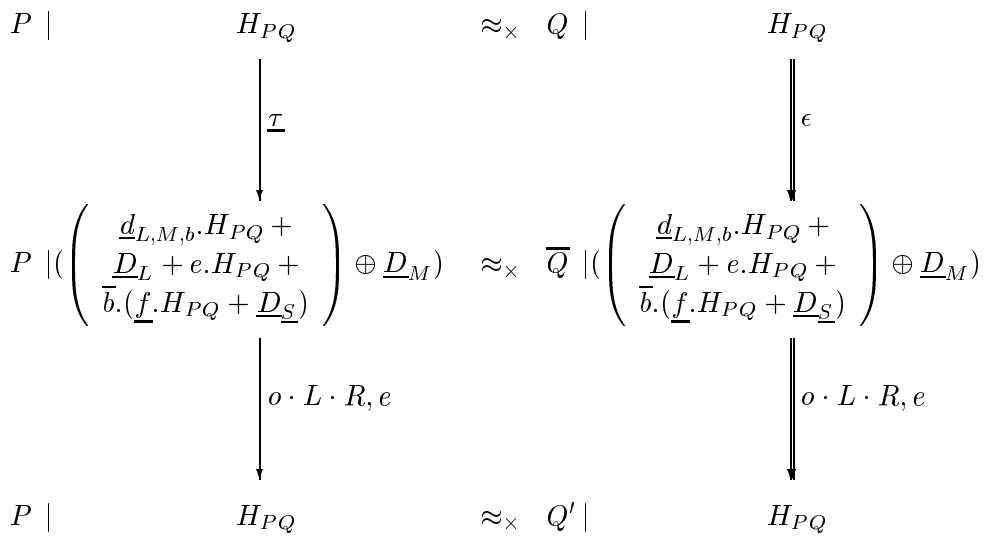

Figure C.4: Largest congruence proof – illustration of Situation (4)

Summarizing, we have shown $\underline{\approx}_{\mathrm{a}}$ to be a distributed prioritized weak bisimulation according to Definition 6.5.4. Therefore, $P \underline{\approx} Q$, as desired. □

Index