Ready Simulation for Concurrency: It's Logical!

Gerald Lüttgen[1] and Walter Vogler[2]

[1] Department of Computer Science, University of York, York YO10 5DD, U.K.
`luettgen@cs.york.ac.uk`
[2] Institut für Informatik, Universität Augsburg, D–86135 Augsburg, Germany
`vogler@informatik.uni-augsburg.de`

Abstract. This paper provides new insight into the connection between the trace-based lower part of van Glabbeek's linear-time, branching-time spectrum and its simulation-based upper part. We establish that ready simulation is fully abstract with respect to failures inclusion, when adding the conjunction operator that was proposed by the authors in [TCS 373(1–2):19–40] to the standard setting of labelled transition systems with (CSP-style) parallel composition. More precisely, we actually prove a stronger result by considering a coarser relation than failures inclusion, namely a preorder that relates processes with respect to inconsistencies that may arise under conjunctive composition. Ready simulation is also shown to satisfy standard logic properties and thus commends itself for studying mixed operational and logic languages.

1 Introduction

Basic research in concurrency theory over the past 25 years has resulted in a wealth of process algebras [2, 8, 13] and temporal logics [4] for specifying and reasoning about concurrent processes. However, little research has been conducted on mixing process-algebraic and logic styles of specification in a single formalism. This is surprising since many popular software-engineering languages, including UML, permit such mixed specifications.

In [11] we proposed an approach to defining and reasoning about conjunction on labelled transition systems. Our setting consisted of standard labelled transition systems, augmented by an *inconsistency predicate* (cf. Sec. 2). While our conjunction operator is in essence a synchronous product on visible actions and an interleaving product on internal actions, the challenge was in dealing with inconsistencies. Inconsistencies may either arise when conjunctively composing two processes with different initial action sets (i.e., *ready sets*), or when a process has no other choice for some action than entering an inconsistent state. Our framework was equipped with *ready-tree semantics*, which is a variant of van Glabbeek's path-based possible-worlds semantics [6] that was inspired by Veglioni and De Nicola [17]. The resulting ready-tree preorder (for divergence-free consistent systems) turned out to be coarser than ready simulation and finer than failures inclusion and ready-trace inclusion, which implies the important feature that ready-tree semantics is sensitive to deadlock. We proved in [11] that the ready-tree preorder is fully abstract under conjunction with respect to a naive *inconsistency preorder*,[3] which allows an inconsistent specification only to be implemented by an inconsistent implementation.

This paper first shows that the ready-tree preorder is inadequate in the presence of concurrency, as it fails to be compositional for standard parallel composition, such as the parallel operator of CSP [8]. A different compositionality problem for the parallel composition of SCCS was already noted in [6]. We then establish our main result (cf. Sec. 3), namely that *ready simulation* [3], which adds to ordinary simulation the requirement that related processes must have identical ready sets, is fully abstract with respect to conjunction *and* parallel composition, for labelled transition systems with inconsistencies. Along the way, we adapt ready simulation to dealing with internal actions and inconsistencies. We also conduct several sanity checks: we verify that our conjunction operator indeed formalises *conjunction* regarding ready simulation, and prove further logic properties desired of ready simulation. Omitted proofs can be found in [12].

Our full-abstraction result provides an interesting insight into van Glabbeek's linear-time, branching-time spectrum [6], namely that conjunction on processes is a tool, via full abstraction, for relating the trace-based lower part of the spectrum to the simulation-based upper part. In addition, our results testify to the adequacy of ready simulation as the semantic basis for mixed process-algebraic and logic languages. Indeed, ready simulation eliminates the necessity for restrictions on the nesting of process-algebraic and logic constructs, such as the one employed by Olderog when embedding trace formulas into CSP [14].

2 Logic LTS, conjunction & parallel composition

This section recalls the definitions of *Logic Labelled Transition Systems*, or Logic LTS for short, and the conjunction operator on Logic LTS which were introduced in [11]. It also lifts the parallel composition operator of CSP [8] to Logic LTS.

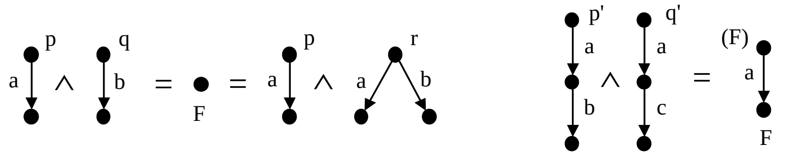

Fig. 1. Basic intuition behind conjunctive composition.

Key to our setting is the consideration of *inconsistencies* which may arise under conjunctive composition. The idea is to mark a composed state between

[3] I.e., the ready-tree preorder is the *coarsest precongruence* for conjunction which refines the inconsistency preorder.

two processes as inconsistent, if one offers an action that the other cannot perform, i.e., if the processes have different *ready sets* [15]. Consider the processes p, q and r of Fig. 1. Process p and q specify that exactly action a and respectively action b is offered initially, i.e., their ready sets are $\{a\}$ and respectively $\{b\}$. Similarly, process r specifies that a and b are offered initially and thus has ready set $\{a, b\}$. Hence, $p \wedge q$ as well as $p \wedge r$ are *inconsistent* (or *false*) and should be tagged as such. Formally, our variant of LTS will be augmented by an *inconsistency predicate*, or false-predicate, F, so that $p \wedge q$, $p \wedge r \in F$ in our example.

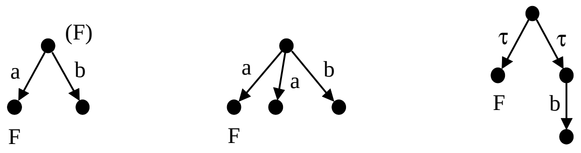

Fig. 2. Backward propagation of inconsistencies.

Inconsistency is a more tricky property, however, as it can propagate backwards along transitions. For example, in the conjunction $p' \wedge q'$ shown in Fig. 1, both conjuncts require action a to be performed, whence $p' \wedge q'$ should have an a-transition. But this transition does lead to an inconsistent state and, in the absence of any alternative a-transition leading to a consistent state, $p' \wedge q'$ must itself be considered as inconsistent. In this spirit, inconsistency propagates backwards for the left process in Fig. 2, whereas it does not for the middle and right processes, as they can engage in an a-transition, respectively τ-transition, that leads to a consistent state. As an aside, it is noted that the right process may be interpreted as a disjunction between the inconsistent process marked F which has empty behaviour, and the consistent process offering a b-transition.

Logic Labelled Transition Systems. Let $\mathcal{A}$ be an alphabet with representatives a and b, and let $\mathcal{A}_\tau$ denote $\mathcal{A} \cup \{\tau\}$ with representatives α and β. An LTS is a triple $\langle P, \longrightarrow, F\rangle$,[4] where P is the set of *processes* (states), $\longrightarrow \subseteq P \times \mathcal{A}_\tau \times P$ is the *transition relation*, and $F \subseteq P$ is the *inconsistency predicate*. We write (i) $p \xrightarrow{\alpha} p'$ instead of $\langle p, \alpha, p'\rangle \in \longrightarrow$, (ii) $p \xrightarrow{\alpha}$ instead of $\exists p' \in P.\, p \xrightarrow{\alpha} p'$ and (iii) $p \longrightarrow$ instead of $\exists p' \in P,\, \alpha \in \mathcal{A}_\tau.\, p \xrightarrow{\alpha} p'$. When $p \xrightarrow{\alpha} p'$, we say that process p can perform an α-step to p', and we call p' an α-derivative. A process p that cannot engage in a τ-transition, i.e., $p \not\xrightarrow{\tau}$, is called *stable*. The *sort* $\mathcal{A}_P$ of the LTS (and its processes) is the set of actions occurring in $\longrightarrow$.

We also require an LTS to satisfy the following τ-*purity* condition: $p \xrightarrow{\tau}$ implies $\nexists a \in \mathcal{A}.\, p \xrightarrow{a}$, for all $p \in P$. Hence, each process represents either an external or internal (disjunctive) choice between its outgoing transitions. This restriction reflects the fact that ready sets can only be observed at stable states,

[4] The additional, less relevant *true predicate* of [11] is omitted here for clarity.

so that visible transitions leaving instable states are outside our observation. The LTSs of interest to us need to satisfy two further properties:

Definition 1 (Logic LTS [11]). An LTS $\langle P, \longrightarrow, F\rangle$ is a *Logic LTS* if:

(LTS1) $F \subseteq P$ such that $p \in F$ if $\exists \alpha \in \mathcal{I}(p)\, \forall p' \in P.\ p \xrightarrow{\alpha} p'$ implies $p' \in F$;
(LTS2) p cannot stabilise implies $p \in F$.

The first condition formalises the backward propagation of inconsistencies as discussed above; here, $\mathcal{I}(p)$ stands for the ready set $\{\alpha \in \mathcal{A}_\tau \mid p \xrightarrow{\alpha}\}$ of process p. The second condition relates to *divergence*, i.e., infinite sequences of τ-transitions, where divergence is viewed as catastrophic if a process cannot *stabilise*.

Before formalising our notion of stabilisation, we introduce several variants of weak transition relations which will prove useful in the sequel. We write $p \overset{\epsilon}{\Longrightarrow} p'$ if $p \xrightarrow{\tau}{}^{*} p'$ and $p \overset{a}{\Longrightarrow} p'$ if $\exists \overline{p}.\, p \xrightarrow{a} \overline{p} \overset{\epsilon}{\Longrightarrow} p'$. Note that we do not consider τ-transitions preceding a visible transition as we only need weak a-transitions originating from stable processes. If all processes along a computation $p \overset{\epsilon}{\Longrightarrow} p'$ or $p \overset{a}{\Longrightarrow} p'$, including p and p', are consistent, then we write $p \overset{\epsilon}{\Longrightarrow}_{\mathrm{F}} p'$ and $p \overset{a}{\Longrightarrow}_{\mathrm{F}} p'$, respectively. If in addition, p' is stable, we write $p \overset{\epsilon}{\Longrightarrow}\!| \, p'$ and $p \overset{a}{\Longrightarrow}\!| \, p'$, respectively. We may now define that a process p *can stabilise* if $\exists p'.\, p \overset{\epsilon}{\Longrightarrow}\!| \, p'$.

We will denote a transition $p \xrightarrow{\alpha} p'$ with $p, p' \notin F$ by $p \xrightarrow{\alpha}_{\mathrm{F}} p'$. Moreover, whenever we mention a process p without stating a respective Logic LTS explicitly, we assume implicitly that such a Logic LTS $\langle P, \longrightarrow, F\rangle$ is given. Finally, we let *ff* stand for the only process of the LTS $\langle\{\mathit{ff}\}, \emptyset, \{\mathit{ff}\}\rangle$, which represents the boolean constant *false*. Intuitively, any given process is either inconsistent, in which case it is equivalent to *ff*, or it is equivalent to a process from which no inconsistent process can be reached; the latter can simply be achieved by omitting inconsistent processes in LTSs and all transitions leading to them.

Conjunction & parallel composition. Our conjunction operator is a synchronous product for visible transitions and an asynchronous product for τ-transitions, analogous to $\|_{\mathcal{A}}$ defined below. However, we need to take care of inconsistencies. This is because, otherwise, $p \wedge q$, with p and q defined as in Fig. 1, would be a consistent process without any transitions.

Definition 2 (Conjunction operator [11]). The conjunction of two Logic LTSs $\langle P, \longrightarrow_P, F_P\rangle$ and $\langle Q, \longrightarrow_Q, F_Q\rangle$ is the Logic LTS $\langle P \wedge Q, \longrightarrow_{P\wedge Q}, F_{P\wedge Q}\rangle$:

- $P \wedge Q =_{\mathrm{df}} \{p \wedge q \mid p \in P,\ q \in Q\}$
- $\longrightarrow_{P\wedge Q}$ is determined by the following operational rules:

$$\begin{array}{rcl} p \xrightarrow{\tau}_P p' & \text{implies} & p \wedge q \xrightarrow{\tau}_{P\wedge Q} p' \wedge q \\ q \xrightarrow{\tau}_Q q' & \text{implies} & p \wedge q \xrightarrow{\tau}_{P\wedge Q} p \wedge q' \\ p \xrightarrow{a}_P p',\ q \xrightarrow{a}_Q q' & \text{implies} & p \wedge q \xrightarrow{a}_{P\wedge Q} p' \wedge q' \end{array}$$

– $F_{P \wedge Q}$ is the least set such that each $p \wedge q \in F_{P \wedge Q}$ satisfies at least one of the following conditions:
(C1) $p \in F_P$ or $q \in F_Q$;
(C2) $p \wedge q \not\xrightarrow{\tau}_{P \wedge Q}$ and $\mathcal{I}(p) \neq \mathcal{I}(q)$;
(C3) $\exists \alpha \in \mathcal{I}(p \wedge q)\, \forall p' \wedge q'.\, p \wedge q \xrightarrow{\alpha}_{P \wedge Q} p' \wedge q'$ implies $p' \wedge q' \in F_{P \wedge Q}$;
(C4) $p \wedge q$ cannot stabilise.

We are left with explaining Conds. (C1)–(C4). Firstly, a conjunction is inconsistent if a conjunct is inconsistent. Conds. (C2) and (C3) reflect our intuition of inconsistency and backward propagation. Cond. (C4) is added to enforce (LTS2).

Definition 3 (Witness). A *witness* is a set $W \subseteq P \wedge Q$ such that, for all $p \wedge q \in W$, the following conditions hold:

(W1) $p, q \notin F$;
(W2) $p \xrightarrow{\tau}$ or $q \xrightarrow{\tau}$ or $\mathcal{I}(p) = \mathcal{I}(q)$;
(W3) $\forall \alpha \in \mathcal{A}_\tau.\, p \wedge q \xrightarrow{\alpha}$ implies $\exists p' \wedge q' \in W.\, p \wedge q \xrightarrow{\alpha} p' \wedge q'$;
(W4) $p \wedge q$ can stabilise in W, i.e., $p \wedge q \xrightarrow{\tau} p_1 \wedge q_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} p_n \wedge q_n \not\xrightarrow{\tau}$ with all $p_i \wedge q_i \in W$.

It is easy to check that the set of consistent processes $\overline{F_{P \wedge Q}}$ of $P \wedge Q$, i.e., the complement of $F_{P \wedge Q}$, is a witness and is in fact the largest one in $P \wedge Q$. This implies the following straightforward proposition, giving us a useful tool for proving that the conjunction of two processes is consistent:

Proposition 4. *$p \wedge q \notin F_{P \wedge Q}$ if and only if $\exists$witness $W.\, p \wedge q \in W$.*

For example, the concept of witness may be employed to prove the following properties of conjunctive composition:

Lemma 5. *1. If $p \wedge q \xrightarrow{\tau} p' \wedge q' \notin F$ and $p, q \notin F$, then $p \wedge q \notin F$.*
2. Let $p \overset{\epsilon}{\Longrightarrow}| p'$, $q \overset{\epsilon}{\Longrightarrow}| q'$ and $p' \wedge q' \notin F$. Then, $p \wedge q \overset{\epsilon}{\Longrightarrow}| p' \wedge q'$.

Finally, we adapt the parallel operator $\|_A$ of CSP [8] to our setting, where $A \subseteq \mathcal{A}$ denotes the *synchronisation alphabet*. Naturally, the parallel composition of two processes is inconsistent if either process is inconsistent.

Definition 6 (Parallel operator). The parallel composition of two Logic LTS $\langle P, \longrightarrow_P, F_P \rangle$ and $\langle Q, \longrightarrow_Q, F_Q \rangle$ for the synchronisation set $A \subseteq \mathcal{A}$, is the Logic LTS $\langle P \|_A Q, \longrightarrow_{P \|_A Q}, F_{P \|_A Q} \rangle$:

– $P \|_A Q =_{\text{df}} \{p \|_A q \,|\, p \in P,\, q \in Q\}$
– $\longrightarrow_{P \|_A Q}$ is determined by the following operational rules:

$$\begin{array}{rcl}
p \xrightarrow{\alpha}_P p',\ \alpha \notin A,\ (\alpha = \tau \text{ or } q \not\xrightarrow{\tau}_Q) & \text{implies} & p \|_A q \xrightarrow{\alpha}_{P \|_A Q} p' \|_A q \\
q \xrightarrow{\alpha}_Q q',\ \alpha \notin A,\ (\alpha = \tau \text{ or } p \not\xrightarrow{\tau}_P) & \text{implies} & p \|_A q \xrightarrow{\alpha}_{P \|_A Q} p \|_A q' \\
p \xrightarrow{a}_P p',\ q \xrightarrow{a}_Q q',\ a \in A & \text{implies} & p \|_A q \xrightarrow{a}_{P \|_A Q} p' \|_A q'
\end{array}$$

– $p \parallel_A q \in F_{P \parallel_A Q}$ if $p \in F_P$ or $q \in F_Q$.

Both conjunction and parallel composition are well-defined, i.e., the compositions of two Logic LTSs satisfy the conditions of Def. 1. In the sequel, we leave out indices of relations and predicates whenever the context is clear.

Ready-tree semantics. Our previous work [11] focused only on studying conjunction on Logic LTSs. It characterised the largest precongruence contained in the *inconsistency preorder*, which states that a consistent implementation p does never refine an inconsistent specification q.[5]

Definition 7 (Inconsistency preorder [11]). The *inconsistency preorder* $\sqsubseteq_F$ on processes is defined by $p \sqsubseteq_F q$ if $p \notin F$ implies $q \notin F$.

This definition agrees with the standard verification question whether an implementation satisfies its specification. When reading 'satisfies' logically as 'implies', it is clear that an inconsistent (i.e., 'false') specification can only be met by an inconsistent implementation.

Obviously, the inconsistency preorder is not compositional with respect to conjunction. Our characterisation of the fully-abstract preorder contained in $\sqsubseteq_F$ and presented in [11] is based on a variant of the path-based possible-worlds semantics of [6, 17], to which we refer as *ready-tree semantics*. This semantics employs the notion of *observation tree*. An observation tree is a Logic LTS $\langle V, \longrightarrow, \emptyset \rangle$ whose processes and transitions form a *deterministic tree* and whose processes (vertices) are stable; we refer to the tree's root as v_0. We may now formalise our desired observations of a process p, called *ready trees*:

Definition 8 (Ready tree [11]). An observation tree v_0 is a *ready tree of* p, if there is a labelling $h : V \longrightarrow P$ satisfying the following conditions:

(RT1) $\forall v \in V.\ h(v)$ stable and $h(v) \notin F$;
(RT2) $p \overset{\epsilon}{\Longrightarrow}\!| \, h(v_0)$;
(RT3) $\forall v \in V,\ a \in \mathcal{A}.\ v \xrightarrow{a} v'$ implies $h(v) \overset{a}{\Longrightarrow}\!| \, h(v')$;
(RT4) $\forall v \in V. \mathcal{I}(v) = \mathcal{I}(h(v))$.

Intuitively, nodes v in a ready tree represent stable states $h(v)$ of p (cf. the first part of Cond. (RT1)) and transitions represent stable, consistent computations (cf. Cond. (RT3)). Since such computations do not contain inconsistent states, no represented state must be in F (cf. the second part of Cond. (RT1)). Since p might not be stable, the root v_0 of a ready tree represents a stable process reachable from p via some internal computation (cf. Cond. (RT2)). Moreover, v must mimic the ready set of $h(v)$ (cf. Cond. (RT4)). In the following, we write $\mathrm{RT}(p)$ for the set of all ready trees of p; note that *ff* has no ready tree.

[5] The reader familiar with [11] should note that we now write the implementation to the left and the specification to the right of the preorder symbol, in order to be consistent with the notational conventions of simulation-based preorders.

Definition 9 (Ready-tree preorder [11]). The *ready-tree preorder* $\subseteq_{\mathrm{RT}}$ on processes is defined as ready-tree inclusion, i.e., $p \subseteq_{\mathrm{RT}} q$ if $\mathrm{RT}(p) \subseteq \mathrm{RT}(q)$.

Theorem 10 (Full-abstraction wrt. conjunction [11]). *$\subseteq_{RT}$ is the largest precongruence in $\sqsubseteq_F$, when considering conjunction as the only operator.*

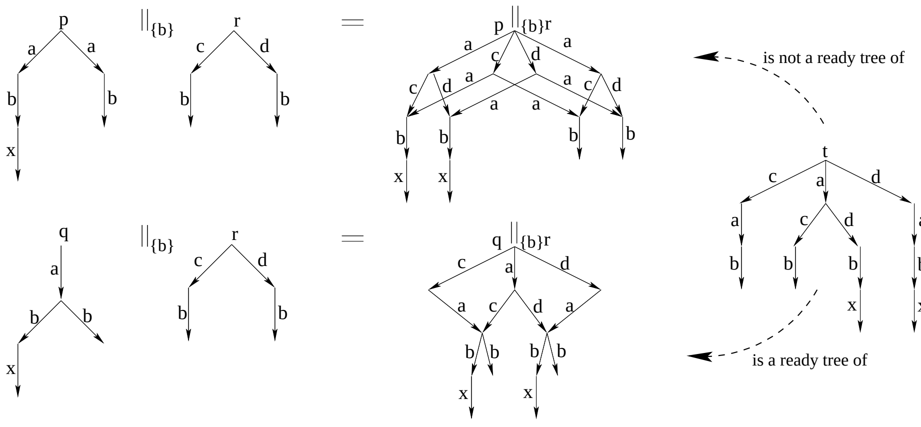

Fig. 3. Ready-tree semantics is not compositional for parallel composition.

Unfortunately, $\subseteq_{\mathrm{RT}}$ is *not* a precongruence for parallel composition $\|_A$, which makes the preorder unsuitable for reasoning about concurrency. To see this, consider the Logic LTSs p, q and r of Fig. 3. Here, p and q have the same ready trees, but t is a ready tree of $q \|_{\{b\}} r$ but not of $p \|_{\{b\}} r$.

3 Full abstraction via ready simulation

We now establish our full-abstraction result of ready simulation wrt. the inconsistency preorder, when considering both conjunction and parallel composition.

Definition 11 (Ready simulation on Logic LTS). Let $\langle P, \longrightarrow_P, F_P \rangle$ and $\langle Q, \longrightarrow_Q, F_Q \rangle$ be two Logic LTS. A relation $\mathcal{R} \subseteq P \times Q$ is a *stable ready simulation relation*, if the following conditions hold, for any $\langle p, q \rangle \in \mathcal{R}$ and $a \in \mathcal{A}$:

(RS1) p, q stable;
(RS2) $p \notin F_P$ implies $q \notin F_Q$;
(RS3) $p \overset{a}{\Longrightarrow}| \, p'$ implies $\exists q'.\, q \overset{a}{\Longrightarrow}| \, q'$ and $\langle p', q' \rangle \in \mathcal{R}$;
(RS4) $p \notin F$ implies $\mathcal{I}(p) = \mathcal{I}(q)$.

We say that p is *stable ready simulated* by q, in symbols $p \precsim_{\mathrm{RS}} q$, if there exists a stable ready simulation relation $\mathcal{R}$ with $\langle p, q \rangle \in \mathcal{R}$. Further, p is *ready simulated* by q, written $p \sqsubseteq_{\mathrm{RS}} q$, if $\forall p'.\, p \overset{\epsilon}{\Longrightarrow}| \, p'$ implies $\exists q'.\ q \overset{\epsilon}{\Longrightarrow}| \, q'$ and $p' \precsim_{\mathrm{RS}} q'$. We write $\approx_{\mathrm{RS}}$ and $=_{\mathrm{RS}}$ for the kernel of $\precsim_{\mathrm{RS}}$ and $\sqsubseteq_{\mathrm{RS}}$, respectively.

It is easy to see that $\underset{\sim}{\sqsubset}_{\text{RS}}$ and $\sqsubseteq_{\text{RS}}$ are preorders, and that $p \sqsubseteq_{\text{RS}} q$ trivially holds if $p \in F$. Moreover, ready simulation $\sqsubseteq_{\text{RS}}$ is contained in the ready-tree preorder $\subseteq_{\text{RT}}$, as essentially stated in [6], and conjunction and parallel composition are associative and commutative with respect to $=_{\text{RS}}$. Note that there are several ways how to define ready simulation [3, 6] for settings with internal actions [5]. Our variant is an adaptation of Glabbeek's *stability respecting ready simulation may preorder* to Logic LTS. Observe that replacing the premise $p \overset{a}{\Longrightarrow}| \, p'$ of (RS3) by $p \xrightarrow{a}_{\text{F}} p'$ results in a finer preorder, unlike for some other simulation-based behavioural relations [13].

Theorem 12 (Compositionality).

1. *Let $p \underset{\sim}{\sqsubset}_{RS} q$, r be stable and $A \subseteq \mathcal{A}$. Then, (a) $p \parallel_A r \underset{\sim}{\sqsubset}_{RS} q \parallel_A r$ as well as (b) $p \wedge r \underset{\sim}{\sqsubset}_{RS} q \wedge r$.*
2. *Let $p \sqsubseteq_{RS} q$, r be an arbitrary process and $A \subseteq \mathcal{A}$. Then, (a) $p \parallel_A r \sqsubseteq_{RS} q \parallel_A r$ and (b) $p \wedge r \sqsubseteq_{RS} q \wedge r$.*

Regarding the proof, we only want to point out that it employs the proof tool of witness in order to reason about the consistency of conjunctively composed processes in Part (1). The following witness turns out to be sufficient for our purpose:

Lemma 13. *The set $W =_{df} W_1 \cup W_2$ is a witness, where*

$$W_1 \;=_{df}\; \{q \wedge r \,|\, \exists p.\, p \underset{\sim}{\sqsubset}_{RS} q \text{ and } p \wedge r \notin F\};$$

$$W_2 \;=_{df}\; \{\overline{q} \wedge \overline{r} \,|\, \exists p, q, r, p', r', q', a.\; p \underset{\sim}{\sqsubset}_{RS} q,\; p \wedge r \notin F,\; p \wedge r \overset{a}{\Longrightarrow}|\, p' \wedge r',\; p' \underset{\sim}{\sqsubset}_{RS} q', \text{ and } q \overset{a}{\Longrightarrow}_F \overline{q} \overset{\epsilon 2}{\Longrightarrow}|\, q' \text{ and } r \overset{a}{\Longrightarrow}_F \overline{r} \overset{\epsilon 1}{\Longrightarrow}|\, r' \text{ with } \{\epsilon 1, \epsilon 2\} = \{\epsilon, \tau\}\}\,.$$

Full-abstraction result. To prove our main result we encode the full behaviour of a stable process p into a single ready tree. The idea is to unfold p to a tree and to eliminate any nondeterminism by placing fresh actions into the tree.

Definition 14 (Characteristic ready tree & context). Let p be a process with Logic LTS $\langle P, \longrightarrow, F\rangle$ having sort $\mathcal{A}_P$, let q be a process with sort $\mathcal{A}_Q$, and let $p \overset{\epsilon}{\Longrightarrow}|\, p_0$.

1. The *characteristic ready tree* P_0 of p with respect to p_0 and q is a Logic LTS whose states are paths $\pi \in P \times (\mathcal{A}_P \times P)^*$ of p originating in p_0, as well as such paths concatenated with *selection sets* D which are subsets of $\mathcal{A}_P \times P$. Formally, the state set P_0 and transition relation $\longrightarrow_{P_0}$ are inductively defined as follows, where $\text{last}(\pi)$ denotes the last process on path π and the $x_D \notin \mathcal{A}_P \cup \mathcal{A}_Q$ are fresh actions with respect to p and q:
   - $p_0 \in P_0$;
   - $\pi \xrightarrow{x_D}_{P_0} \pi D$ and $\pi D \in P_0$, if $\pi \in P_0$, $\forall \langle a, p\rangle \in D.\ \text{last}(\pi) \overset{a}{\Longrightarrow}|\, p$ in P and $\forall a \in \mathcal{I}(\text{last}(\pi))\, \exists_1 \langle a, p\rangle \in D$;
   - $\pi D \xrightarrow{a}_{P_0} \pi a p$ and $\pi a p \in P_0$, if $\pi D \in P_0$ and $\langle a, p\rangle \in D$.

We will write $\langle p_0 \rangle$ instead of p_0 when we wish to highlight that not the process p_0 is meant, but the path consisting only of p_0.

2. The *characteristic context* K of p with respect to p_0 and q is defined as the Logic LTS P_0 augmented with the fresh process 0 and transitions
   - $\pi D \xrightarrow{a}_K 0$, if $\pi D \in P_0$, $a \in \mathcal{A}_Q$ and $\nexists p. \langle a, p \rangle \in D$.

Proposition 15. *Let P_0 be the characteristic ready tree of a process p wrt. some p_0 and q, and let K be the respective characteristic context of p. Then, P_0 is a ready tree of $p \parallel_A \langle p_0 \rangle$, where $A =_{df} \mathcal{A}_P \cup \mathcal{A}_Q$ and $\langle p_0 \rangle$ is the root of K.*

Proof. P_0 is an observation tree by construction, since it is a deterministic tree and since all its vertices are stable processes. We define a mapping h_0 from the vertices in P_0 to processes in $P \parallel_A K$ by $h_0(\pi) =_{\text{df}} \text{last}(\pi) \parallel_A \pi$ and $h_0(\pi D) =_{\text{df}} \text{last}(\pi) \parallel_A \pi D$, and verify Conds. (RT1)–(RT4) of Def. 8:

(RT1) This is trivial since last(π), π and πD are all stable and not in F.

(RT2) We have $p \parallel_A \langle p_0 \rangle \overset{\epsilon}{\Longrightarrow}\!| \, p_0 \parallel_A \langle p_0 \rangle$ by construction.

(RT3) If $\pi \xrightarrow{x_D}_{P_0} \pi D$, then $\pi \xrightarrow{x_D}_K \pi D$ by construction of K. Since x_D is a "fresh" action, $h_0(\pi) = \text{last}(\pi) \parallel_A \pi \xrightarrow{x_D}_{\text{F}} \text{last}(\pi) \parallel_A \pi D = h_0(\pi D)$. If $\pi D \xrightarrow{a}_{P_0} \pi a p$, then $\text{last}(\pi) \overset{a}{\Longrightarrow}\!| \, p$ and $\pi D \xrightarrow{a}_K \pi a p$ by the construction of K. As $a \in A$, we have $h_0(\pi D) = \text{last}(\pi) \parallel_A \pi D \overset{a}{\Longrightarrow}\!| \, p \parallel_A \pi a p = h_0(\pi a p)$.

(RT4) Observe that the ready set of state πD in K is the initial action set $\mathcal{I}(\text{last}(\pi))$ of the last process of path π in P plus all actions in $\mathcal{A}_Q$, whereas the same state in P_0 has only ready set $\mathcal{I}(\text{last}(\pi))$. By the operational rules for parallel composition we obtain:
   - $\mathcal{I}_{P \parallel_A K}(\text{last}(\pi) \parallel_A \pi) = (\mathcal{I}_P(\text{last}(\pi)) \cap \mathcal{I}_K(\pi) \cap A) \cup (\mathcal{I}_P(\text{last}(\pi)) \setminus A) \cup (\mathcal{I}_K(\pi) \setminus A) = \emptyset \cup \emptyset \cup \mathcal{I}_K(\pi) = \mathcal{I}_{P_0}(\pi)$.
   - $\mathcal{I}_{P \parallel_A K}(\text{last}(\pi) \parallel_A \pi D) = (\mathcal{I}_P(\text{last}(\pi)) \cap \mathcal{I}_K(\pi D) \cap A) \cup (\mathcal{I}_P(\text{last}(\pi)) \setminus A) \cup (\mathcal{I}_K(\pi D) \setminus A) = (\mathcal{I}_P(\text{last}(\pi)) \cap (\mathcal{I}_P(\text{last}(\pi)) \cup \mathcal{A}_Q) \cap A) \cup \emptyset \cup ((\mathcal{I}_P(\text{last}(\pi)) \cup \mathcal{A}_Q) \setminus A) = \mathcal{I}_P(\text{last}(\pi)) = \mathcal{I}_{P_0}(\pi D)$; note that the last equality is due to the construction of P_0 from P. □

Observe that P_0 is not a ready tree of p itself due to the fresh actions inserted in P_0; these actions are added to p via the parallel context K. Together, characteristic ready trees and Prop. 15 are the key for proving our main result:

Theorem 16 (Full abstraction). *The largest precongruence contained in $\sqsubseteq_F$, with respect to parallel composition and conjunction, equals $\sqsubseteq_{RS}$.*

Proof. Because of Thm. 12 and Thm. 10 [11], as well as the fact that ready simulation is contained in the ready-tree preorder $\subseteq_{\text{RT}}$ and thus in $\sqsubseteq_F$ [11], it is sufficient to prove that $\sqsubseteq_{\text{RS}}$ subsumes the largest precongruence $\subseteq^+_{\text{RT}}$ contained in $\subseteq_{\text{RT}}$. Consider processes p and q with Logic LTSs P and Q and sorts $\mathcal{A}_P$ and $\mathcal{A}_Q$. We let $\mathcal{A}_{PQ}$ stand for $\mathcal{A}_P \cup \mathcal{A}_Q$, and abbreviate $\parallel_{\mathcal{A}_{PQ}}$ by $\parallel$.

Now assume $p \subseteq^+_{\text{RT}} q$, and consider some p_0 such that $p \overset{\epsilon}{\Longrightarrow}\!| \, p_0$. Because of $p \subseteq^+_{\text{RT}} q$ and Prop. 15, we have $P_0 \in \text{RT}(q \parallel \langle p_0 \rangle)$ due to some mapping h (cf.

Def. 8); in particular, $q \notin F$. Here, P_0 is the characteristic ready tree of p with respect to p_0 and q. To prove our claim, it is sufficient to establish that

$$\mathcal{R}_0 =_{\text{df}} \{\langle p', q' \rangle \,|\, \exists \pi.\, \text{last}(\pi) = p' \text{ and } h(\pi) = q' \parallel \pi\}$$

is a stable ready simulation relation. Thus, let $\langle p', q' \rangle \in \mathcal{R}_0$ due to π.

(RS1) $h(\pi)$ is stable, whence q' is. Moreover, last(π) is stable by construction.

(RS2) $h(\pi) \notin F$ implies $q' \notin F$.

(RS3) Let $p' \overset{a}{\Longrightarrow}\!| \, p''$ and $\pi \xrightarrow{x_D} \pi D$ with $\langle a, p'' \rangle \in D$ for some p''. Then, $\pi D \xrightarrow{a} \pi a p''$. Moreover, $h(\pi D) = q' \parallel \pi D$, whence $q' \parallel \pi D \overset{a}{\Longrightarrow}\!| \, h(\pi a p'') = q'' \parallel \pi a p''$ for some q'' by (RT3), as well as $q' \overset{a}{\Longrightarrow}\!| \, q''$ and $\langle p'', q'' \rangle \in \mathcal{R}_0$ due to $\pi a p''$.

(RS4) We have $p' \notin F$ by construction. Choose some D with $\pi \xrightarrow{x_D} \pi D$, whence $h(\pi D) = q' \| \pi D$. Now, $\mathcal{I}(p') = \mathcal{I}(\pi D)$ in P_0 by construction of P_0. The latter equals $\mathcal{I}(q' \| \pi D)$ by (RT4), which in turn equals the set $\mathcal{I}(q')$ since $\mathcal{A}_Q \subseteq \mathcal{I}(\pi D) \subseteq \mathcal{A}_{PQ}$, for $\mathcal{I}(\pi D)$ in the characteristic context. Hence, $\mathcal{I}(p') = \mathcal{I}(q')$.

Thus, $\mathcal{R}_0$ is a stable ready simulation relation. Finally observe $h(p_0) = q_0 \parallel \langle p_0 \rangle$ for some q_0 such that $q \| \langle p_0 \rangle \overset{\epsilon}{\Longrightarrow}\!| \, q_0 \| \langle p_0 \rangle$ (by (RT2)); therefore, $q \overset{\epsilon}{\Longrightarrow}\!| \, q_0$ and $\langle p_0, q_0 \rangle \in \mathcal{R}_0$ due to $\langle p_0 \rangle$.

Summarising, we have shown that, for each p_0 with $p \overset{\epsilon}{\Longrightarrow}\!| \, p_0$, there exists some q_0 satisfying $q \overset{\epsilon}{\Longrightarrow}\!| \, q_0$ and $p_0 \precsim_{\text{RS}} q_0$. Hence, $p \sqsubseteq_{\text{RS}} q$. □

One way to guarantee the existence of the fresh actions required in the construction of characteristic ready trees is to assume an uncountable alphabet $\mathcal{A}$ and to restrict ourselves to those processes that are finitely branching with respect to $\overset{a}{\Longrightarrow}\!|$, for all $a \in \mathcal{A}$, and have a countable sort. Then, context K and the characteristic ready trees are also finitely branching and have countable sorts.

Logic properties of ready simulation. We conclude this section by highlighting some logic properties of ready simulation.

Theorem 17 ($\wedge$ is And). *(1) $r \precsim_{RS} p \wedge q$ if and only if $r \precsim_{RS} p$ and $r \precsim_{RS} q$; (2) $r \sqsubseteq_{RS} p \wedge q$ if and only if $r \sqsubseteq_{RS} p$ and $r \sqsubseteq_{RS} q$.*

As for the compositionality proof of ready simulation wrt. conjunction, the proof of this theorem uses the concept of witness for reasoning about inconsistencies:

Lemma 18. *The set $W' =_{df} W'_1 \cup W'_2$ is a witness, where*
$W'_1 =_{df} \{p \wedge q \,|\, \exists r.\, r \precsim_{RS} p,\ r \precsim_{RS} q \text{ and } r \notin F\}$
$W'_2 =_{df} \{\overline{p} \wedge \overline{q} \,|\, \exists r, p, q, r', p', q', a.\ r \precsim_{RS} p,\ r \precsim_{RS} q,\ r \overset{a}{\Longrightarrow}\!| \, r',\ p \overset{a}{\Longrightarrow}_F \overline{p} \overset{\epsilon 1}{\Longrightarrow}\!| \, p' \text{ and }$
$q \overset{a}{\Longrightarrow}_F \overline{q} \overset{\epsilon 2}{\Longrightarrow}\!| \, q' \text{ with } \{\epsilon 1, \epsilon 2\} = \{\epsilon, \tau\},\ r' \precsim_{RS} p' \text{ and } r' \precsim_{RS} q'\}$.

Conjunction also satisfies further standard logic properties:

Proposition 19 (Logic properties of ready simulation).
1. $p \wedge ff =_{RS} ff$; $p \wedge ff \approx_{RS} ff$ if p stable;
2. $p \wedge q \sqsubseteq_{RS} p$; $p \wedge q \precsim_{RS} p$ if p, q stable;
3. $p \wedge p =_{RS} p$;
4. $p \wedge q =_{RS} p$ if and only if $p \sqsubseteq_{RS} q$.

In our previous work we also considered a disjunction operator $\vee$ on Logic LTSs. This operator was defined as internal choice, i.e., $p \vee q$ can perform an internal τ-transition to both p and q, where $p \vee q$ is considered to be inconsistent if both p and q are. Due to space constraints we do not include disjunction here, but simply note that ready simulation is compositional for disjunction and that the dual properties to the ones of Prop. 19 hold. The validity of these statements is not difficult to check. Moreover, the distributivity laws hold, too.

4 Related work

This section briefly discusses related work; a full discussion can be found in [11]. Firstly, our ready-tree semantics is in essence the *path-based possible-worlds semantics* of van Glabbeek [6] which goes back to Veglioni and De Nicola [17], and our ready simulation was first suggested by Bloom et al. [3]. However, in contrast to the standard notions of these semantics, our setting deals with internal actions as well as inconsistencies.

Traditional research has often avoided explicitly mixing operational and logic styles of specification by translating one style into the other. Operational content may be translated into logic formulas, as is implicitly done in [7, 10], where logic implication serves as refinement relation [1]. Dually, logic content may be translated into operational content. This is the case in automata-theoretic work, such as in Kurshan's work on ω-automata [9], which includes synchronous and asynchronous composition operators and uses maximal trace inclusion as refinement relation. However, both logic implication and trace inclusion are insensitive to deadlock and are thus inadequate in the presence of concurrency.

A seminal approach to compositional refinement in a mixed setting was proposed by Olderog in [14], where process-algebraic constructs are combined with trace formulas expressed in a predicate logic and where failure semantics forms the semantic basis of refinement. In this approach, trace formulas can serve as processes, but not vice versa. Thus, and in contrast to our present work, freely mixing operational and logic specification styles is not supported and, in particular, conjunction cannot be applied to processes.

Finally, it should be noted that the term *consistency* as used here is different from the one in [16], where two specifications are defined as consistent if they have at least one implementation in common. In our setting, a process $p \notin F$ is called consistent, while $p \wedge q$ implements both p and q, for arbitrary p, q. Thm. 17 also implies that p and q are consistent in the sense of [16], if $p \wedge q \notin F$ in our setting.

5 Conclusions & future work

This paper proved that ready simulation [3] is fully abstract with respect to conjunction and parallel composition on Logic LTS. In this sense, ready simulation is indeed a "logical" semantics. Establishing this result was non-trivial due to the challenges that arise when dealing with inconsistencies under conjunctive composition. This is evidenced by the complex compositionality proof with respect to conjunction, as well as the two-step "largest" precongruence proof that relied on our previous full-abstraction work on ready-tree semantics [11].

Our results show that conjunction is a tool for relating trace-based semantics to simulation-based semantics, via the concept of full abstraction. This sheds additional light onto van Glabbeek's linear-time, branching-time spectrum [6]. Moreover, our results imply that ready simulation commends itself as a suitable behavioural relation for reasoning about specifications given in a mixed operational and logic style. Indeed, future work shall employ ready simulation within novel algebras that will combine process-algebraic and temporal-logic operators.

Acknowledgements. We wish to thank the anonymous referees for their insightful and constructive comments. The first author also acknowledges support by the EPSRC under grant no. EP/E034853/1.

References

[1] M. Abadi and G.D. Plotkin. A logical view of composition. *TCS*, 114(1):3–30, 1993.
[2] J.A. Bergstra, A. Ponse, and S.A. Smolka. *Handbook of Process Algebra*. Elsevier, 2001.
[3] B. Bloom, S. Istrail, and A.R. Meyer. Bisimulation can't be traced. *J. ACM*, 42(1):232–268, 1995.
[4] E.A. Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science*, vol. B, pp. 995–1072. North-Holland, 1990.
[5] R. van Glabbeek. The linear time – branching time spectrum II, 1993. Available at http://theory.stanford.edu/~rvg/abstracts.html#26.
[6] R. van Glabbeek. The linear time – branching time spectrum I. In *Handbook of Process Algebra*, ch. 1, pp. 3–99. Elsevier, 2001.
[7] S. Graf and J. Sifakis. A logic for the description of non-deterministic programs and their properties. *Inform. & Control*, 68(1–3):254–270, 1986.
[8] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.
[9] R.P. Kurshan. *Computer-Aided Verification of Coordinating Processes: The Automata-Theoretic Approach*. Princeton Univ. Press, 1994.
[10] L. Lamport. The temporal logic of actions. *TOPLAS*, 16(3):872–923, 1994.
[11] G. Lüttgen and W. Vogler. Conjunction on processes: Full-abstraction via ready-tree semantics. *TCS*, 373(1–2):19–40, 2007.
[12] G. Lüttgen and W. Vogler. Ready simulation for concurrency: It's logical! Techn. rep. 2007-4, Inst. f. Informatik, Univ. Augsburg, 2007
http://www.informatik.uni-augsburg.de/forschung/reports/
[13] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.
[14] E.-R. Olderog. *Nets, Terms and Formulas*. Cambridge Tracts in Theoretical Computer Science 23. Cambridge Univ. Press, 1991.
[15] E.-R. Olderog and C.A.R. Hoare. Specification-oriented semantics for communicating processes. *Acta Informatica*, 23(1):9–66, 1986.
[16] M. Steen, J. Derrick, E. Boiten, and H. Bowman. Consistency of partial process specifications. In *AMAST '98*, vol. 1548 of *LNCS*, pp. 248–262. Springer, 1999.
[17] S. Veglioni and R. De Nicola. Possible worlds for process algebras. In *CONCUR '98*, vol. 1466 of *LNCS*, pp. 179–193. Springer, 1998.