Department of Computer Science, University of Sheffield

Type Analysis for Component-based Real-Time Programming

Investigator: Michael Mendler

1 Previous research track record

A Previous relevant work by investigator The project will build on substantial previous research carried out by the investigator in the area of real-time process algebras and type theory.

Modelling Real-Time Signal Processing with Process Algebras The semantic underpinning of the project will be the process algebra CSA (Calculus for Synchrony and Asynchrony) [CLM97] which was developed by the principal investigator in cooperation with Gerald Lüttgen (ICASE, USA) and Rance Cleaveland (Stony Brook, USA). CSA is a conservative extension of Milner's Calculus of Communicating Systems (CCS) by the notion of "local clocks" as a new primitive concept expressing global synchronisation in distributed systems. The predecessor of CSA is the process language PMC (Process Algebra with Multiple Clocks) [AM94] developed by the investigator and Henrik Andersen (Lyngby, DK) for the specification of a commercial measurement instrument, the *B&K 2145 Signal Analyzer* [AM95]. A clock in CSA is an abstract analogue to a system clock in hardware that can be used to express real-time constraints. The scope of a clock can be localised to a (logical or physical) part of the system that is controlled by this clock and in which assertions about the quantitative real-time behaviour of the clock, and the computations synchronized by it, can be made. The scope of different clocks may be disjoint, overlapping, or nested, so that the various different time layers of a distributed heterogeneous signal processing application can be modelled. This feature is unique for CSA and of central importance for the proposed project. The second distinguishing feature of CSA, contrasting to many other real-time process algebras, is that it does not require a complete and detailed specification of the timing properties of every single action in the system. The real-time behaviour is concentrated fully in the clocks which focus on a just a few abstract yet essential real-time constraints. Quantitative real-time behaviour may be linked very naturally with clocks. Timing parameters such as throughput, latency, minimum and maximum separation of events may be seen as quantitative measures relating the occurrence times of clock ticks, of one and the same or of different clocks. The separation of the quantitative from the qualitative, i.e. synchronising, aspect of clocks provides the abstraction necessary to master the technical complexity of signal processing and control applications, on which the proposed project focuses, where only few timing parameters are relevant but varying over several orders of magnitude.

Constraints and Type Theory In [Men91, Men93] is it shown how behavioural abstractions can be formalised in terms of type specifications by introducing a modal type constructor ($\bigcirc$) for "correctness-up-to-constraints." In this way type checking corresponds to the construction and verification of abstraction constraints. The close connection between Lax Logic and Constraint Logic Programming has been made precise in [FMW97]. The idea of integrating constraints into a type system has been further developed in [MF96, Men98, Men99] for the combined functional and temporal analysis of combinational systems. The work on Lax Logic provides the inspiration for this project, and the technical expertise to tackle the subtle issue of abstraction contraints within a type-theoretic framework.

Other relevant background Mendler has substantial experience in the application of formal methods to hardware and software design. He has published a formal semantics for VHDL based on streams [FM95], used the Concurrency Workbench for the formal verification of asynchronous circuits [RMS92], worked on formal step-wise refinement of layered communications protocols [SGM89] and on a mathematical framework for abstract program interpretation [SJM92]. He has also investigated formal methods for microprocessor synthesis [WM95, WM96]. More recently he is responsible for the theoretical underpinnings of MoSel [KMMG97], a model-based verification tool for the automatic analysis of finite-state systems in monadic second-order logic. Mendler is collaborator on the EPSRC funded project GR/L86180 "Lax Logic applied to Formal System Design" persued at Sheffield. During the period 1995–1998 Mendler held a personal fellowship from the *Deutsche Forschungsgemeinschaft* to apply Propositional Lax Logic to timing analysis of combinational systems. Until January 1999 Mendler worked at the University of Passau, Germany. There, he began a close cooperation with the group of Werner Grass on component-based programming environments for signal-processing applications [SBF$^+$98b]. The project will directly benefit from and continue this cooperation.

B Previous relevant work by collaborators The research group in the school of Mathematics and Computer Science at Passau University, which is headed by Werner Grass, is supporting this proposal—

see attached letter. Bernhard Sick in the Passau group has developed the graphical programming tools Sally [SFUS95] and Iconnect [SBF$^+$98a] for the interactive programming of signal-processing and control applications. These tools have been applied to industrial-sized case studies, *e.g.* for feature extraction of large data sets to train neural nets [Sic98]. Many other relevant publications may be found at Sick's http address, see [Sic98]. Iconnect has been developed in cooperation with Roland Mandl at Micro-Espilon GmbH, who is also collaborating.

C Institutional expertise and support Sheffield University has a large and diverse Department of Computer Science. There is expertise in the following four broad areas: Artificial Intelligence and Neural Nets (AINN); Language, Speech and Hearing (SPANDH); Communications and Distributed Systems (CDS), Verification and Testing (VT). Mendler is newly appointed reader (1st Feb 99) in the VT research group which is headed by Mike Holcombe. The department has recently achieved a rating of 4 in HEFCE's Research Selectivity Exercise and it currently holds in excess of £1.5M in terms of EPSRC grants and fellowships and in excess of £3.5M of other funding including European projects.

In 1996 Holcombe completed a large SERC project GR/H 73585. The project also involved the Department of Automatic Control and Systems Engineering and investigated formal specification and verification of hybrid systems. Holcombe was the principal investigator on the recent SERC project IED2/1/1031 "Functional testing of high integrity VLSI" which was highly rated on evaluation. VT's research work is also supported by Daimler-Benz. Fairtlough in the VT group is principal investigator on the £180K EPSRC funded research project GR/L86180 "Lax Logic applied to Formal System Design" which started in October 1998. Mendler is collaborator on this project. Simons in the VT group is principal investigator on the £221K EPSRC project GR/M56777 "MOTIVE - Method for Object Testing, Integration and Verification"

The case studies for the project proposed here will be drawn from robust mobile phone applications, specifically voice dialling in cars. The background and algorithmic know-how to realise these case studies will be provided Martin Cooke of the SPANDH group. Cooke who is a recognised expert in speech recognition is currently working on the problem of robust voice dialling and its applications. Cooke completed a £120K EPSRC funded project into Robust Signal Processing and Recognition of Occluded Speech, under GR/K 18962, in October 1998.

2 Description of proposed research

A Background

Application area Industrial software production has to cope with the ever growing complexity of software applications under increasingly tight time-to-market schedules. There is, of course, no general-purpose 'silver-bullet' solution to the fundamental trade-off between the efficiency and reliability of the software development process on the one hand and the efficiency and reliability of the produced software on the other. It is unlikely, in particular, that novel general-purpose programming paradigms such as object-oriented programming can resolve this inherent conflict. Case studies like [Hat98], in fact, suggest the contrary. For specialised application domains like control and signal processing, or intelligent networking, a new class of high-level graphic programming systems is beginning to establish itself as an alternative to the traditional low-level 'program-it-all-from-scratch-in-C^{++}' approach. These tools represent a component-based programming paradigm based on module abstraction and software reuse. They typically provide a rich repository of pre-compiled and well-tested software modules (such as FIR-filter, FFT-transforms, PID-controllers for signal processing). It is the restriction to their specific application domain that permit component-based programming systems to take over from the application programmer most of the error-prone programming tasks. By employing fixed design abstractions they can encapsulate the complexity within abstract modules with clear-cut interfaces. Such tools embody a wealth of domain-specific knowledge about the given application area. The enormous potential of component-based application-specific programming environments in increasing programming efficiency has been clearly demonstrated by academic and commercial projects. Their potential in increasing correctness and efficiency of the application software itself, however, is only beginning to be recognized, such as by [SMC$^+$96] in the domain of intelligent networking. Currently only few programming tools in signal processing and control exploit light-weight static validation techniques such as type-checking, timing analysis, or model-checking, which have been developed in Computer Science over the past two decades. Here lies a large potential for further research and development. Static validation methods can be an effective way to meet special demands of these application areas, in addressing the quality characteristics of accurateness, reliability, and efficiency, but without incurring prohibitive run-time overhead.

Related work

Graphic programming environments for signal processing and control applications with integrated run-time system. In [Sch97] an overview of the most important tools in this area is given. A widely used PC-based programming tool for measurement and control applications is LabVIEW (Laboratory Virtual Instruments Engineering Workbench) of National Instruments. It has a very powerful graphical user interface to define virtual instruments and build applications by connecting them [JP97]. The are many other commercial and noncommercial tools with graphical program specification. Examples are HP Vee of Hewlett Packard [HPVee], DASYLab [TL97] by Datalog GmbH, DIAdem [Dia96] of GfS mbh. The system Khoros [Fri97] for pattern recognition and image analysis is publicly available. Finally, we mention the tools Sally [SFUS95] and Iconnect [SBF$^+$98a] developed by Passau University, the latter in collaboration with Micro-Epsilon GmBH. Apart from a sophisticated GUI which supports a component-based programming style and the run-time system, these tools have in common a number of features specific to signal processing, measurement and control applications. They are typically based on a high-level data-flow model which may include:

• block-oriented data streams • different sampling rates • cycles in the data-flow graph, simple control-flow • synchronization of data streams • parallel execution of graphs • real-time constraints • hierarchical structuring • user defined data types and modules

In the following we shall simply refer to these as SGPR tools (for 'Signal Graph Programming and Run-time'). Besides the SGPR tools mentioned there exist other high-level systems without run-time system that generate executable code from data flow specifications (see *e.g.* [BML96]). A well-known example is Ptolemy [Pto] which was developed for the design and simulation of multiprocessor systems or DSPs. The kernel of Ptolemy has been used in various other systems (simulation of optical communication networks, design of special purpose processors [RGF97]) by a number of international companies and universities such as as HP, NEC, Cadence, Universities of Boston and Berkeley, as well as governments.

Types and Type-Checking. Some SGPR tools like Sally perform a limited form of type checking. For instance, they may produce an error message in case a fixed-point number input is connected to a floating-point output. Due to their high level of description the signal flow graphs of SGPR tools are rather similar to functional programs, and many type-theoretic features developed for functional programming may be exploited. Among them is the *let-polymorphism* [Mil78] which can be used for validating hierarchically structured flow graphs involving local declarations and instantiating of generic modules. *Dependent types* [CW85] are useful *e.g.* for generic modules whose number of inputs and outputs depend on a static parameter. Another example arises if module types include timing information: The propagation delay through a FIR filter depends on the the number of filter stages which is not known in advance but determined at programming time. A natural example for sub-types is easy to find, too: in order to connect a module A producing data in blocks of 100 integer values with another module B that accepts integer blocks of variable size the type system must be able to recognize the type integer[100] of A's output as a sub-type of integer[*] which is the type of B's input. There are a number of type analysis algorithms in the literature for type systems of different expressiveness. An excellent overview is given in [Mit90]. Traditionally, these focus on data structures only. There is, however, no intrinsic reason why types should not also capture abstract reactive and temporal semantic constraints. It is the goal of this project to demonstrate this. In SGPR tools type analysis may include the computation of sampling rates to instantiate generic modules and introducing antialiasing filters appropriately. Further, if types express worst-case propagation delays through modules, the type analysis for a composite signal flow graph might synthesize the overall latency of the graph. In general, assuming that types also capture abstract reactive and temporal behaviour, type synthesis would amount to computing the reactive and temporal properties of the composite system from that of its parts, as far as these can be expressed in the given type system. In including reactive and temporal aspects into the type checking and synthesis, this project conquers new terrain.

Processes and Model-Checking. In order to achieve a suitable level of flexibility in adjusting the descriptive level we will employ a process algebraic approach. Process algebras are an extremely well investigated framework for specifying and verifying concurrent systems at a high abstraction level. Their semantics typically is given in terms of *transition systems* describing the reactive behaviour of the system at hand, together with an equivalence relation that compares processes according to some notion of observation. Traditionally, process algebras are restricted to modelling the nondeterminism of distributed reactive systems. Recently, an increasing number of papers in the area is concerned with introducing implementation specific features into process algebras. Of those in particular the aspects of *time* [HR95] and *priorities* [CW95] are important for this project. However, most of these process algebras, as far as the mentioned extensions are concerned, deal with non-distributed systems, *i.e.* with *global* time and *global* priorities. Only recently process algebras with *distributed* time and priorities have been introduced [CLM97, CLN96]. Another characteristic of existing timed process algebras is that they are geared towards a maximally detailed and low-level description of a system's temporal behaviour. For complex signal flow graphs this must inevitably lead to unmanageable descriptions. Because of this these timed process algebras are not suitable as the basis of a notion of a reactive process type as envisaged by this project, which is to represent static information that abstracts from most timing parameters. The same applies to non-process-algebraic formalisms such as timed automata [AD94] or real-time logics [JM94, GMM90, OS95]. Therefore, as described above, we will use a more abstract approach based on the *clocked transition systems* of CSA. Efficient algorithms for checking the equivalence of (CSA) transition systems may be derived from the *partition refinement* method [PT87]. The other automatic analysis technique that has been very successful, which carries to CSA as well, is *model checking.* Its purpose is to verify abstract reactive properties expressed as logic formulas in a modal logic, such as the absence of *deadlock* or *livelock*, *mutual exclusion* or other *safety* and *liveness* properties. In this area a lot of progress has been made, which is documented, for instance, in the proceedings of the Springer LNCS conference series on *Computer Aided Verification.* An overview of tools and methods is given in [IP96]. Recent notable developments on the tool side include the *North Carolina Concurrency Workbench* [CS96], the *Concurrency Factory* [RPSO96] at SUNY, Stony Brook, SMV [CMCHG96], SPIN [HP96] and TempEst [JPvO95]. For timed process algebras and timed automata, too, model-checking algorithms have been developed, see *e.g.* the first issue of the STTT journal [SCM97]. It is expected that these can be adapted for CSA to include time parameters for clocks.

B Research programme and methodology

B.1 Aims and Objectives The overall aim of this project is to open up and explore a new route for transferring formal methods technology into software industry, by extending the conventional type-checking paradigm. Specifically we aim to develop an automatic validation method based on the notion of real-time process types in combination with rigorous semantic models that

• permits the specification of static functional, reactive, and temporal requirements • combines and adapts standard type-checking, model-checking, and timing-analysis techniques,

• is specialised for use in component-based interactive programming environments in the signal processing and control area.

By developing appropriate verification algorithms based on our method our long-term aim is to help practical programming tools in the targeted application area meet the high demands on correctness, real-time performance, and numeric precision of the resulting programs. The specific objectives are

• to devise a theory of real-time process types that combines reactive and temporal behaviour • to develop efficient automatic type analysis algorithms for this type theory based on the notion of abstraction (from control data and timing) and constraints • to produce a stand-alone type analysis tool implementing these algorithms • to demonstrate their usefulness and efficiency in the application context, by integrating the type analysis into an existing SGPR system and running application examples and use scenarios.

B.2 Methodology Historically, type-checking focuses on functional behaviour only, *i.e.* addresses the question "are the data of the right form and used consistently during the computation?" The project proposes to extend the notion of types also to capture abstract reactive and temporal semantic constraints, *e.g.* "are the data produced and consumed in the right order and properly synchronized with the computation phases?" or "are new input data accepted with sufficient speed?" These, too, are abstract properties of SGPR programs which may be verified statically at programming time. Using the close analogy between types and specifications the project will develop a theory of *real-time process types* that extends the standard functional type systems by reactive and temporal information, and devise appropriate type-analysis algorithms that extend the standard functional type checking by model-checking and timing analysis (*e.g.* MILP-solving).

In order to obtain rigorous statements about the soundness and (relative) completeness of the extended type-checking algorithms developed in this project an adequate semantic framework is needed. This will be provided by the *clocked transition systems* of CSA. The process algebraic semantics provides for a technically economical way to capture adequately the *nondeterminism* and *concurrency.* Both arise quite naturally in a static description of signal flow graphs that abstracts to a large extent from data dependence, as well as most details of the run-time system and the implementation platform. The role of abstraction and nondeterminism for static analyses is highlighted in [Cor96]. In [Cor96] it is shown for imperative programs how one can obtain sufficiently precise and yet compact real-time models, even considering priorities, run-time overhead, or resource limitations. Choosing a process-algebraic approach has the advantage that it is faithful to possibly distributed implementations, and that it permits the modelling of optional and non-optional module inputs as well as any other control-flow that may exist in a signal flow graph. For this many of the well-known data-flow semantics based on streams (which go back to [Kah74]) are insufficient.

The *clocks* of CSA will be the interface between the qualitative (clock ticks as synchronisation event) and the quantitative (relative distance between clocks ticks) real-time behaviour. The interface between reactive and functional behaviour is given by the *value-passing* principle that combines in an orthogonal way the communication action with the data transmitted in the synchronisation event. This separation of concerns makes process algebras in general and CSA in particular an ideal semantic formalism for the project. The abstraction level can be adjusted in several directions (data, timing, control state abstraction) by introducing additional nondeterminism and also by trading between local (actions) and global (clocks) synchronisation.

A distinct methodological feature of the project will be the use of *abstractions* and *constraints* as a means for the semantically consistent combination of dedicated validation algorithms. Dedicated algorithms for functional, reactive, or temporal properties will need to abstract further from the CSA process types (which themselves are abstractions of module behaviour). Efficient model-checking, for instance, will have to abstract from data and possibly also from timing parameters. However, to maintain a consistent interpretation of the model-checking result the data and timing constraints that are left behind must be taken care of. Similarly, efficient timing analysis may require abstraction from control states and data. To trade complexity between different verification methods and to obtain a consistent combination of type-checking, model-checking, and timing analysis we intend to follow up ideas from Lax Logic and constraint programming.

B.3 Originality and timeliness Model checking, timing analysis, and (ordinary) type checking are three of the most well-known and successfully mechanized formal verification techniques developed in Computer Science. They have been developed, however, as separate methods. Their combination within one coherent specification and validation system has not been attempted before.

A new generation of formal methods tools discussed in the literature combines different verification methods to overcome their inherent limitations and expand their applicability. Recent work advocates hybrid specification and verification methods, such as combining theorem-proving and model-checking [ORR$^+$96, Yu99]. In another direction some workers advocate data-abstraction and generation of data constraints to extend model-checking with functional verification [HGD95]. Our project, too, falls into this class of hybrid methods, but focuses on static validation rather than formal verification. It is not the generality and expressiveness of one single formalism that is at issue in this project but the semantically consistent combination of different static analysis methods.

On the theoretical side the project will highlight the importance of intensional soundness and completeness theorems to deal with abstractions in the presence of constraints. So far most abstractions considered, *e.g.* in the model-checking literature, evade the problem of constraints simply by applying abstractions only in situations in which the property one is interested in is known to be preserved. This, however, is a rare case in practice and rules out many interesting applications of model-checking. The results of this project, which takes abstraction constraints seriously, therefore may be expected to conquer new terrain for model-checking applications. The same applies to the timing analyses considered by the project.

B.4 Programme of work We are requesting funding to develop the proposed theory of real-time types to include static reactive and temporal information, and associated abstractions. These are the main novel features of our type theory. Conventional data types will also be considered, though not be the main focus. The work will proceed in three phases, as described in the following.

Phase I: Type System In the first year the type system of *real-time process types* PT, specialised for the class of SGPR programming tools, will be developed. Its semantics is defined as an extension of CSA to capture static constraints on the reactive and temporal behaviour of modules and signal flow graphs. The extension consists in adding *timing parameters* to CSA clocks, and *finite value passing* to CSA actions. Clock timing parameters are to measure the relative speed of modules at different synchronization levels. In terms of clock timing parameters it will be possible to express a number of standard performance requirements such as lag time, throughput, response time constraints. It will be shown how typical real-time scheduling strategies employed by SGPR tools can be expressed by CSA processes that enforce execution priorities on clocks. In the project we will restrict ourselves to value passing over finite data types, which is adequate to model the control flow in SGPR graphs. According to the static analysis paradigm all non-finite influence of signal data on the control flow and performance will be subsumed by nondeterminism and worst-case timing approximations. There exist standard techniques for adding value passing (*e.g.* [HL95]) to CCS-based process languages such as CSA. More specifically, we can build on work done at the Technical University of Denmark [Mør99] for the related language PMC.

Based on suitably formalised languages PT of process types and SFG of signal flow graphs, the type system will consist, at its heart, of a *typing relation* $\mathsf{m}_1 : \mathsf{pt}_1, \ldots, \mathsf{m}_n : \mathsf{pt}_n \vdash_{\mathsf{PT}} \mathsf{sfg} : \mathsf{pt}$, specifying that a given signal flow graph $\mathsf{sfg} \in \mathsf{SFG}$ posesses the process type pt, assuming that all the modules $\mathsf{m}_1, \ldots \mathsf{m}_n$ have the process types $\mathsf{pt}_1, \ldots, \mathsf{pt}_n$, respectively. The relation $\vdash_{\mathsf{PT}}$ embodies the CSA semantics of process types. It *defines* the typing, and serves as the formal basis to establish soundness and completeness of specific type analysis algorithms to be developed in phases II and III that *implement* the type system.

Phase II: Type Analysis In year 2 the type analysis algorithms for the type system are developed. Full type checking in terms of $\vdash_{\mathsf{PT}}$ (*i.e.* using the transition system semantics) would combine all three behavioural aspects of control data, reaction, and time together. This is likely to be too inefficient for practical SGPR tools. Instead, in phase II, the project will identify fragments PT_r, PT_t, PT_d of PT that focus on reaction, time, and data, respectively. The fragment PT_r would abstract (to an adjustable degree) from data and timing and thus reduce PT to the standard CTL *model-checking* problem; Similarly, PT_t will be the fragment of pure (combinational) *timing analysis*, and PT_d of conventional type checking problems. To make use of these simpler fragments *abstraction maps* $\alpha_{dt} : \mathsf{PT} \rightarrow \mathsf{PT}_\mathsf{r}$, $\alpha_{dr} : \mathsf{PT} \rightarrow \mathsf{PT}_\mathsf{t}$, and $\alpha_{rt} : \mathsf{PT} \rightarrow \mathsf{PT}_\mathsf{d}$ will be defined. There may be several such maps, e.g. different α_{dr} for response-time and lag-time analysis. The purpose of α_{dt}, α_{dr}, α_{rt} is to reduce the general type checking problem for PT to that for PT_r, PT_t, PT_d, respectively, for which efficient modifications of well-known static validation algorithms can be implemented. Since these abstractions loose information their correctness and completenss is only *up to constraints*. For instance, where α_{dt} abstracts from control data and timing it will increase the nondeterminism of a type pt and introduce into $\alpha_{dt}(\mathsf{pt})$ new potential execution paths, which are executable (or sensitizable) only if certain associated data and timing constraints are met. If the abstract PT_r model-checking $\mathsf{m}_1 : \alpha_{dt}(\mathsf{pt}_1), \ldots, \mathsf{m}_n : \alpha_{dt}(\mathsf{pt}_n) \vdash_{\mathsf{PT}_\mathsf{r}} \mathsf{sfg} : \alpha_{dt}(\mathsf{pt})$ finds that these execution paths are relevant for the given reactivity property $\alpha_{dt}(\mathsf{pt})$ to hold, then these data and timing constraints need to be validated separately outside of the model-checking. This means that the model-checking algorithm must be implemented so that it keeps track of all the relevant constraints. These depend both on the abstractions $\alpha_{dt}(\mathsf{pt}_i)$ and on the property checked, $\alpha_{dt}(\mathsf{pt})$. The same applies to abstract PT_t and PT_d analyses. We will develop the type-checking algorithms so they generate abstraction constraints and prove soundness and completeness properties for them relative to the generated constraints. This corresponds to the notion of *intensional soundness* and *intensional completeness* of Lax Logic type theory [Men98, MF96].

Phase III: Implementation and Integration In the last phase we will implement the type-checking algorithms and integrate them into one of the reference tools provided by Passau and Micro-Epsilon to produce a demonstrator. The most suitable one of the tools will be selected in meetings PM3 and PM4.

We plan to implement a *type analysis machine* centered around a constraint handling system kernel that serves to keep track of and solve constraints arising from the abstractions α_{dt}, α_{dr}, α_{tr}. This will exploit standard constraint programming technology. Relevant constraint-solving packages will need to cover inequations for time parameters, Boolean equations for control data, and term unification for ordinary data types. The type analysis machine will be integrated with the tool's GUI, so that it accumulates and solves typing constraints as the user incrementally builds up a signal flow graph. Because of the interactive nature of this process, special attention needs to be paid to error recovery and backtracking.

A demonstrator will be built to evaluate the usefulness and efficiency of the new validation features, based on an application from speech recognition. The case study planned is a system for robust voice dialling in mobile car phones. The algorithmic solution to this problem will be provided by Martin Cooke in the Speech and Hearing Group. The experiments will explore different ways of modularising and distributing the complex speech recognition algorithms developed by Cooke and perform static validation of reactive and timing constraints for these.

B.5 Workplan The work plan defining the work packages is given in Fig. 1.

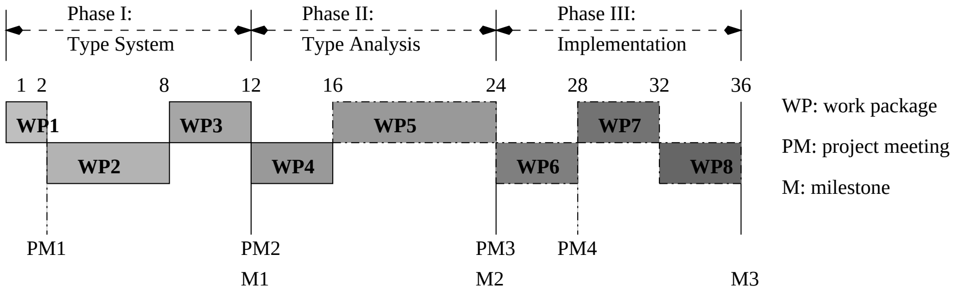

Figure 1: Work Plan

Phase I: • **WP 1**: Analyse requirements for SGPR tools; RA to attend training course at Passau to study the project's reference tools Sally/Iconnect; simple application case study. • **WP 2**: Extend CSA by finite value-passing and clock timing parameters; Develop semantic framework of clocked symbolic transition systems. • **WP 3**: Define formal languages SFG and PT together with the $\vdash_{\mathsf{PT}}$ typing relation, and its CSA semantics; Specify a representative number of Sally/Iconnect modules and signal flow graphs in PT to demonstrate adequacy of the type system as a static specification language for SGPR tools. • **Milestone M1**: Interim report on **WP1–WP3**. • **Project meetings**: **PM1** at Passau to discuss modelling requirements; **PM2** to review CSA model and discuss useful abstractions.

Phase II: • **WP 4**: Define fragments PT_r, PT_t, PT_d of PT; reformulate standard CTL model-checking for PT_r, floating mode timing analysis algorithms for PT_t, and standard type checking for PT_d. • **WP 5**: Define suitable abstractions α_{dt}, α_{dr}, α_{tr} and identify associated abstraction constraints; extend type-checking algorithms of **WP4** to generate constraints; establish relative (= intensional) soundness and completenss theorems for them. • **Milestone M2**: Interim report on **WP4–WP5**. • **Project meeting**: **PM3** to review theoretical results and plan implementation.

Phase III: • **WP 6**: Implement a stand-alone system combining the type analyses and abstractions of **WP5** with constraint solving techniques. • **WP 7**: Integrate the type analysis machine with the user interface of Sally/Iconnect; Characterize a complete set of Sally/Iconnect modules in terms of PT types; extend module library by type-information • **WP 8**: Use the demonstrator on synthetic examples, as well as the speech recognition case study; Evaluate efficiency and usefulness of the type analysis machine as a static and automatic validation facility. • **Milestone M3**: Demonstrator, case study, final report. • **Project meeting**: **PM4** at Passau after completion of **WP6** to decide on which implementation to use for the demonstrator, and to prepare integration.

C Relevance to beneficiaries and collaborators

Beneficiaries of this work are the community of engineers and scientists who seek to make significant progress in the application of formal methods in industrial practice by exploiting the potential of large grain verification in terms of domain-specific formal methods [LG97]. The challenge that must be met is the conflict between efficiency on the one hand, and expressiveness and correctness on the other. It is here that the project attempts to make a contribution. It attempts to show that it is possible to combine dedicated and efficient verification methods without compromising efficiency and correctness. By demonstrating the feasibility and usefulness of a combined static validation in a class of domain-specific programming environments the project introduces a natural "interpolation" point between formal methods and software engineering practice, and thus opens up a new path for technology transfer.

By integrating model-checking and timing analysis within one coherent specification and validation system, as well as by specialising to a particular application domain the project is likely to raise new questions of both practical and theoretical nature that are of interest to the Computer Science community in general. Specifically, it will show how the notions of type-theoretic abstraction and constraints may permit to trade the complexity between different static verification techniques, and to allow for efficient implementations without jeopardizing the overall semantic correctness of the analysis.

Immediate beneficiaries of this project will be the group of Sick who are developing SGPR tools, and the group of Cooke who are using such tools to experiment with different voice recognition algorithms. The type checking and timing analysis methods developed in the project, when implemented in the SGPR tool, would assist Cooke to establish the static correctness of complex voice recognition algorithms, and to study the effect of different algorithmic solutions on their performance.

D Dissemination and exploitation The annual project reports will form the basis of papers presented at conferences such as Tools and Algorithms for the Construction and Analysis of Systems, Parallel and Real Time Systems, Computer Aided Verification, as well as for papers in journals such as Formal Methods in System Design, Software Tools for Technology Transfer, Automated Analysis of Software, and the IEEE Transactions.

A successful conclusion of the project would establish a solid case for commercial development of the validation framework in SGPR tools, but the theoretical nature of the research precludes industrial involvement at this stage. The commercial potential will be assessed as part of the final project meeting **PM4** when the integration is discussed.

E Justification of resources The work will require a substantial amount of research which Mendler is not in a position to undertake. We therefore request funding of a RA1B researcher to work on the project for 3 years. The named candidate for the post is Barry Norton, who has finished at Sheffield this summer, with a 1st class BSc in Software Engineering. Norton was awarded the Engineering faculty's Mappin Medal, as the best student of Software Engineering. He is an exceptional candidate for this project. He has not only the theoretical background (Mathematics, Logic and Logic Programming) but also considerable experience in industrial C++ programming and component-based software developments. More details can be found in the attached CV. Norton will be expected also to do a (part-time) Ph.D in Computer Science on a topic closely related to the project. However, in view of the many and extremely well-paid positions in Industry offered to a 1st class candidate we would not be able to keep him merely on a studentship. He will accept, however, an RA position. Norton has now joined our Verification and Testing group on a short term research project funded by Daimler-Benz and supervised by Mike Holcombe, which will terminate in March 2000. The Daimler-Benz project specifically involves model-checking and automata-based verification methods, and thus will provide Norton with an ideal preparation towards the proposed project.

The project requires a dedicated machine for full-time use by the researcher. To build the final demonstrator for the speech recognition case studies a high-quality microphone and an audio card are required. To have available sufficient power for numerical computations we request funding for a dual-processor machine. Also, in order to store the large amounts of signal and parameter data necessary for the application the machine will need to be equipped with a second disk drive.

The project will build on existing constraint programming techniques. We plan to use the commercial C++ constaint solving library (ILOG Solver/Planner/Scheduler) developed by ILOG Ltd. Bracknell to be customised and suitably extended by us, for integration with our implementation. The RA1B will attend a 3-day ILOG training course at Bracknell.

We are requesting support for the RA1B to travel to Passau to work with Bernhard Sick in years 1 and 3. On the first of these visits the RA1B will be given 6 weeks training on the lconnect tool and its implementation, including a two-day professional crash course run by Micro-Epsilon. This will establish a good link with Passau and reduce the time needed for the RA1B to become a skilled user of SGPR tools. The investigator will also need to travel to Passau, for a shorter period of time, to attend the PM1 and PM4 project meetings. To assure the scientific quality of his work and to scrutinise the theoretical methodology the RA1B will need to undertake a short term visit to an internationally leading research center in model-checking and logic programming, such as the groups of Rance Cleaveland and Scott Smolka at SUNY, Stony Brook, N.J. USA. We request £1,000 for this. We further request funding to attend European conferences and workshops within the UK, and international conferences in year 3.

In addition to routine office expenses of £500 p.a., we will need £200 p.a. for telephone contact between Sheffield and Passau.

References

[AD94] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, April 1994.

[AM94] H. R. Andersen and M. Mendler. An asynchronous process algebra with multiple clocks. In *ESOP'94*, pages 58–73. Springer, LNCS 788, 1994.

[AM95] H. R. Andersen and M. Mendler. Describing a signal analyzer in the process algebra PMC — A case study.

In *TAPSOFT'95*, pages 620–635. Springer, LNCS 915, 1995.

[BML96] S. S. Bhattacharryya, P. K. Murthy, and E. A. Lee. *Software Synthesis from Dataflow Graphs.* Kluwer, 1996.

[CLM97] R. Cleaveland, G. Lüttgen, and M. Mendler. An algebraic theory of multiple clocks. In *CONCUR'97*, pages 166–180. Springer, LNCS 1243, 1997.

[CLN96] R. Cleaveland, G. Lüttgen, and V. Natarajan. A process algebra with distributed priorities. In *CONCUR'96*, pages 34–49, 1996.

[CMCHG96] E. M. Clarke, K. McMillan, S. Campos, and V. Hartonas-GarmHausen. Symbolic model checking. In *CAV'96*, pages 419–422. Springer, LNCS 1102, 1996.

[Cor96] James C. Corbett. Constructing abstract models of concurrent real-time software. In Steven J. Zeil, editor, *Proc. Symposium on Software Testing and Analysis*, pages 250–260, New York, ACM Press, 1996.

[CS96] R. Cleaveland and S. Sims. The NCSU concurrency workbench. In *CAV'96*, pages 394–397. Springer, LNCS 1102, 1996.

[CW85] L. Cardelli and P. Wegner. On understanding types, data abstraction, and polymorphism. *Computing Surveys*, 17(4):471–522, 1985.

[CW95] J. Camilleri and G. Winskel. CCS with priority choice. *Information and Computation*, 116(1):26–37, January 1995.

[Dia96] DIAdem Homepage. Gfs mbH, Aachen, 1996. `http://www.gfs-ac.de`.

[FM95] M. Fuchs and M. Mendler. Functional semantics for delta-VHDL based on FOCUS. In C. Delgado Kloos and P. T. Breuer, editors, *Formal Semantics for VHDL*, chapter 1. Kluwer, March 1995.

[FMW97] M. Fairtlough, M. Mendler, and M. Walton. First-order lax logic as a framework for constraint logic programming. Technical Report MIP-9714, University of Passau, July 1997. Postscript available through `http://www.dcs.shef.ac.uk/~mendler`.

[Fri97] D. Frieauff. Kraftpaket–Bildverarbeitung mit Khoros 2.1. *iX-Magazin für professionelle Informationsverarbeitung*, 1997.

[GMM90] C. Ghezzi, D. Mandriolli, and A. Morzenti. Trio: A logic language for executable specifications of real-time systems. *Journal of Systems and Software*, 12(2):107–123, 1990.

[Hat98] Les Hatton. Does OO sync with how we think? *IEEE Software*, pages 46–54, May/June 1998.

[HGD95] H. Hungar, O. Grumberg, and W. Damm. What if model checking must be truly symbolic. In *C HARME'95*, pages 1–20. Springer, LNCS 987, 1995.

[HL95] M. Hennessy and H. Lin. Symbolic bisimulation. *Theoretical Computer Science*, 138:353–389, 1995.

[HP96] G. J. Holzmann and D. Peled. The state of SPIN. In *CAV'96*, pages 385–389. Springer, LNCS 1102, 1996.

[HPVee] Hewlett-Packard, HP-Vee, `http://www.tmo.hp.com/tmo/pia/HPVEE/PIATop/English/`.

[HR95] M. Hennessy and T. Regan. A process algebra for timed systems. *Information and Computation*, 117:221–239, 1995.

[IP96] P. Inverardi and C. Priami. Automatic verification of distributed systems: The process algebra approach. *Formal Methods in System Design*, 8:7–38, 1996.

[JM94] F. Jahanian and A. K. Mok. Modechart: A specification language for real-time systems. *IEEE Transactions on Software Engineering*, 20(12):933–947, December 1994.

[JP97] R. Jamal and H. Pichlik. *LabVIEW-Programmiersprache der vierten Generation.* Prentice Hall, 1997. (see `http://vanegl.ecs.umass.edu/Socratis/LabVIEW/`)

[JPvO95] L. J. Jagadeesan, C. Puchol, and J. E. von Olnhausen. Safety property verification of ESTEREL programs and applications to telecommunications software. In *CAV'95*, pages 127–140. Springer, LNCS 933, 1995.

[Kah74] G. Kahn. The semantics of a simple language for parallel programming. In J. L. Rosenfeld, editor, *Information Processing '74*, pages 471–475. North Holland, 1974.

[KMMG97] P. Kelb, T. Margaria, M. Mendler, and C. Gsottberger. MOSEL: A flexible toolset for monadic second-order logic. In *TACAS'97*, pages 183–202. Springer LNCS 1217, April 1997.

[LG97] Luqi and J. Goguen. Formal methods: Promises and problems. *IEEE Software*, pages 73–85, January 1997.

[Men91] M. Mendler. Constrained proofs: a logic for dealing with behavioural constraints in formal hardware verification. In G. Jones and M. Sheeran, editors, *Workshop on Designing Correct Circuits.* Springer, 1991.

[Men93] M. Mendler. *A Modal Logic for Handling Behavioural Constraints in Formal Hardware Verification.* ECS-LFCS-93-255, Department of Computer Science, University of Edinburgh, March 1993.

[Men98] M. Mendler. Characterising timing analysis in intuitionistic modal logic. In R.J.G.B. de Queiroz and M. Finger, editors, *Proc. WOLLIC'98*, pages 132–140. University of São Paulo, Brazil, 1998. Invited paper to *Logic Journal of the IGPL*, Postscript available through `http://www.dcs.shef.ac.uk/~mendler`.

[Men99] M. Mendler. Timing analysis of combinational circuits in intuitionistic propositional logic. *To appear in: Formal Methods in System Design*, 1999. A short version appeared at TABLEAUX'96, Springer, LNAI

1071, pp. 261–277. Postscript available through `http://www.dcs.shef.ac.uk/~mendler`.

[MF96] M. Mendler and M. Fairtlough. Ternary simulation: A refinement of binary functions or an abstraction of real-time behaviour? In M. Sheeran and S. Singh, editors, *Proc. 3rd Workshop on Designing Correct Circuits (DCC96)*. Springer, October 1996. Springer Electronic Workshops in Computing.

[Mil78] R. Milner. A theory of type polymorphism in programming. *J. Comp. Sys. Sci.*, 17(3):348–375, 1978.

[Mit90] J. C. Mitchell. *Type Systems for Programming Languages*, chapter 8, pages 365–458. Elsevier, 1990.

[Mør99] S. Mørk. *High-level Design of Embedded Systems.* PhD thesis, Danish Technical University, 1999. To appear.

[ORR+96] S. Owre, S. Rajan, J. M. Rushby, N. Shankar, and M. Srivas. PVS: Combining specification, proof checking, and model checking. In *CAV'96*, pages 411–414. Springer LNCS 1102, 1996.

[OS95] E.-R. Olderog and M. Schenke. Design of real-time systems: The interface between Duration Calculus. In J. Desel, editor, *Structures in Concurrency Theory*, Workshops in Computing, pages 32–54. Springer-Verlag, 1995.

[PT87] R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM Journal of Computing*, 16(6):973–989, 1987.

[Pto] The almagest: A manual for Ptolemy. `http://ptolemy.berkeley.edu`.

[RGF97] M. A. Richards, A. J. Gadient, and G. A. Frank. *Rapid Prototyping of Application Specific Signal Processors.* Kluwer, 1997.

[RMS92] T. Reinhardt, M. Mendler, and T. Stroup. Die formale Validierung einer "Bausteintafel" delay-insensitiver Grundelemente. In *ITG/GME/GI Fachtagung Rechnergestützter Entwurf u. Architektur Mikroelektronischer Systeme*, pages 273–274. VDE Verlag, 1992.

[RPSO96] R. Cleaveland, P. M. Lewis, S. A. Smolka, and O. Sokolsky. The concurrency factory: a development environment for concurent systems. In *CAV'96*, pages 398–401. Springer LNCS 1102, 1996.

[SBF+98a] A. Sicheneder, A. Bender, E. Fuchs, R. Mandl, and B. Sick. A framework for the specification and execution of complex signal processing applications. In *ICASSP '98*, Seattle, May 1998, Vol. 3, pages 1757 - 1760.

[SBF+98b] A. Sicheneder, A. Bender, E. Fuchs, M. Mendler, and B. Sick. Tool-supported software design and program execution for signal processing appliations using modular software components. In T. Margaria and B. Steffen, editors, *International Workshop on Software Tools for Technology Transfer STTT'98.* Springer, 1998.

[Sch97] P.G. Schreier. Users adopt new technologies, return to familiar suppliers. *Personal Engineering*, pages 22–25, January 1997.

[SCM97] B. Steffen, W.R. Cleaveland, and T. Margaria, editors. *Software Tools for Technology Transfer*, volume 1. Springer, December 1997.

[SFUS95] B. Sick, E. Fuchs, A. Ulrich, and G. P. Supe. SALLY – ein Tool zur Signalanalyse. Technical report, Department of Computer Science, Passau University, 1995.

[SGM89] T. Stroup, N. Götz, and M. Mendler. Stepwise refinement of layered protocols by formal program development. In *9th IFIP WG6.1 Int'l Symposium on Protocol Specification, Testing, and Verification*, 1989.

[Sic98] B. Sick. Online tool wear monitoring in turning using time-delay neural networks. In *ICASSP'98*, Seattle, May 1998, Vol. 1, pages 445–448. See also `http://lrs.fmi.uni-passau.de/~sick/-private/publications.html`.

[SJM92] B. Steffen, C. B. Jay, and M. Mendler. Compositional characterization of observable program properties. *Theoretical Informatics and Applications*, 26(5):403–424, 1992.

[SMC+96] Bernhard Steffen, Tiziana Margaria, Andreas Claßen, Volker Braun, and Manfred Reitenspieß. An environment for the creation of intelligent network services. In *Intelligent Networks: IN/AIN Technologies, Operations, Services, and Applications – A Comprehensive Report*, pages 287–300. IEC: International Engineering Consortium, Chicago, 1996.

[TL97] N. Trevarthen and S. Leigh. 16 and 32 bit data acquisition systems with multiboard drivers. *Adept Scientific*, July 1997. (see *e.g.* `http://www.datalog-dasytec.de/`).

[WM95] L. G. Wang and M. Mendler. The formal design of a class of computers — its high stage: Abstract microprogramming. In *CHARME'95*, pages 84–102. Springer LNCS 987, 1995.

[WM96] L. G. Wang and M. Mendler. Abstraction of hardware construction. In *HOA'95*, pages 264–287. Springer LNCS 1074, 1996.

[Yu99] Shen-Wei Yu. *Formal Verification of Concurrent Programs in Type Theory.* PhD thesis, Department of Computer Science, University of Durham, 1999. To appear.