RealType Project Report:
Type Analysis for Component–Based Real–Time Programming

Grant: GR/M99637/01; PI: Prof. M. Mendler (Years 1 & 2) / Dr. G. Luettgen (Year 3); Date: July 2003

Background/Context. The theme of this research is to extend the well–known and successful technique of data types by reactive information, aiming at forming a semantic basis for static analysis in *component–based programming*. Since the research proposal's submission in 1999, this idea has been receiving increasing interest in the research community and, by now, has turned into a recognised research theme at international level, referred to by terms such as *behavioural types*, *interface types*, *reactive types* or *architectural types*. Notable researchers competing in this area are Luca de Alfaro, Thomas Henzinger and Ed Lee in the USA, Albert Benveniste and Jean–Pierre Talpin in France, Marco Bernardo in Italy and Franz Puntigam in Austria, to name but a few. In this context, the present work is distinguished methodologically by using a process–algebraic approach to reactive types and with respect to application in that it focuses on component–based programming in the domain of *digital signal processing and control* (DSPC).

Within the RealType project, ICONNECT, a tool for the visual development of signal processing applications, was taken as a typical member of the family of tools targeted. A detailed description of ICONNECT and a comparison to related tools (e.g., LabVIEW, Matlab/Simulink, Agilent Vee, DIAdem, and DASYlab) can be found in [24]. ICONNECT itself has descended from an academic tool for signal processing developed at the University of Passau, Germany; both the University of Passau and the commercial vendors of ICONNECT, Micro–Epsilon GmbH in Ortenburg, Germany, were partners on this project.

Example. As an example of the programming style adopted in DSPC tools consider the *digital spectrum analyser* modelled by the *hierarchical signal–flow graph* in Fig. 1. The task is to analyse an audio signal and continually show an array of bar–graphs representing the intensity of the signal in disjoint sections of the frequency range. The spectrum analyser is designed with the help of source components Soundcard and Const, and computation components Element and BarGraph. Each instance c1, c2, ... of Element, written as ck:Element or simply ck, is responsible for assessing the intensity of one frequency range, which is then displayed by component dk:BarGraph. The first input port ei_{k1} of ck:Element is connected to the output port $\overline{\mathrm{so}}$ of component s0:Soundcard, which generates the audio signal and provides exactly one audio value each time it is scheduled. As can be seen by the wire stretching from output port $\overline{\mathrm{co}}_k$ to input port ei_{k2}, ck:Element is also connected to component sk:Const, which initialises ck:Element by providing filter parameters when it is first scheduled. In contrast to components Soundcard and Const, Element is not a basic but a hierarchical component. Indeed, every ck encapsulates one instance of Filter, ck1:Filter, and one of Quantise, ck2:Quantise, as shown in Fig. 1 on the right–hand side.

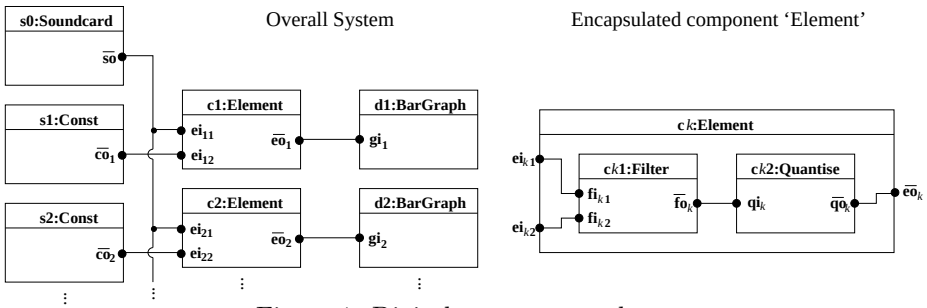

Figure 1: Digital spectrum analyser

According to the DSPC scheduling policy adopted in ICONNECT, which is targeted at uni–processor architectures, the example application will be serialised as follows within each execution cycle. First, each source component instance gets the chance to execute. In the first cycle, this will be s0:Soundcard and all sk:Const. In all subsequent cycles, only s0:Soundcard will request to be scheduled, since sk:Const can only produce a value once. Each produced sound value will be instantaneously propagated from output port $\overline{\mathrm{so}}$ of s0 to the input port ei_{k1} of each ck:Element, for all $k \geq 1$, according to the principle of *isochronic broadcast*. This guarantees that all ck must have received the new value from s0:Soundcard before any cl:Element may be scheduled, and ensures that the array of bar–graphs display a consistent state synchronous with the

environment. The scheduler then switches to scheduling computation components. Since all necessary inputs of each ck are available in each cycle, every ck will request to be scheduled. The scheduler will serialise these requests, each ck will execute accordingly, and the synthesised frequency–strength signal will be emitted by component ck2:Quantise via port $\overline{\text{qo}}_k$ and propagated by ck through port $\overline{\text{eo}}_k$. Upon reception of this signal by dk:BarGraph at port gi_k, this computation component instance will also request to be scheduled and, according to the synchrony principle of *run–to–completion*, granted execution within the same cycle. When all components dk have executed, the current cycle ends since these do not generate outputs that need to be propagated to the system environment. It is important to note that, since each ck encapsulates further computation component instances, its execution is non–trivial and involves a sub–scheduler that will schedule ck1:Filter and ck2:Quantise in such a way that an execution of these instances will appear atomic outside of ck. This ensures that the scheduling of the inner ck1 and ck2 will not be interleaved with any sibling cl of ck, or any component dk.

Let us now consider what happens if instances s0:Soundcard and s1:Const are accidently connected the wrong way around, i.e., output port $\overline{\text{so}}$ is connected to input port ei_{12}, and output port $\overline{\text{co}}_1$ of s1:Const to input port ei_{11} of c1:Element. Recall that c11:Filter within c1:Element will only read a value, an initialisation value, from port ei_{12} in the first execution cycle and never again afterwards. Thus, when the value of s0:Soundcard produced in the second cycle is propagated to port ei_{12} and further to fi_{12}, the system *jams*. This is because the value that has been produced in the second cycle and stored at this latter port, has not yet been read by c11:Filter. According to Micro–Epsilon GmbH, jams were the single most prominent problem in the use of the ICONNECT tool and were only dealt with in an ad–hoc fashion by assigning execution priorities to components; however, no compile–time checks were available to demonstrate the absence of jams.

Objectives. The project's main objective, as reaffirmed at the initial project meeting with the industrial and academic collaborators [1], was the development of a *compositional interface model*, or *coordination model*, of synchronous scheduling that (i) admits hierarchical run–to–completion abstraction (ii) captures isochronous broadcast, (iii) facilitates static jam checks, and (iv) is applicable to the ICONNECT tool. The RealType project envisioned that this coordination model captures the interfaces of components together with the imposed scheduling constraints; these interfaces could then be understood as a *reactive type*, with the checking of jam–freeness corresponding to type checking. In the course of the project, it was realised that the underlying semantic foundations were not unique to tools such as ICONNECT but apply to many *synchronous languages*, including *Esterel* [3] and *Statecharts* [11]. Consequently, the project's objectives were generalised accordingly, thus widening the impact of RealType.

What was not any more a priority for the collaborators in Passau since the submission of the project proposal was *timing analysis*. This is because all implementations of the ICONNECT technology had been on multi–tasking operating systems, while experiments with real–time operating systems proved unpersuasive. Given the change of interests of the collaborators that are documented in the minutes of the first project meeting [1], the project's focus had to shift from 'timing analysis' to 'coordination and semantics'.

Key Advances and Supporting Methodology. The novel approach of the RealType project has resulted from understanding hierarchical dataflow–oriented systems not as an extended form of pure dataflow, as was done in Ptolemy [4] and in previous work of our collaborators [18], but as a new form of synchronous programming language that needs to be grounded in a more general set of semantic primitives. In the Ptolemy approach, the validation against behavioural faults such as jams and the formation of synchronous steps of behaviour are dealt with together, by finding a solution to a system of linear equations. This technology requires that the number of inputs of each component per execution is fixed, which is an assumption that is not valid in tools such as ICONNECT, as shown in the above example regarding the inputs ei_{k2}. To generalise the Ptolemy approach to include statefulness of components as well as nondeterministic responses to input values, probabilistic assumptions are necessary [4]. Since these are difficult to validate in practice, the confidence in the correctness of jam analyses in Ptolemy is often compromised.

In contrast, the view taken by RealType is that the operations of systems in ICONNECT are a series of synchronous steps, in each of which a *fixed point* of communications is reached by run–to–completion, i.e., by asynchronous chain reaction. The new form of synchronous coordination model envisaged thus involves a *globally–synchronous and locally–asynchronous* (GSLA) scheduling discipline. To achieve this, the temporal process algebra CSA, which had previously been co–developed by Luettgen and Mendler [7], was adapted. This algebra is general enough to express a mixture of synchronous and asynchronous behaviour. For doing

so, it employs the concept of *abstract clocks* that are governed by an implicit *maximal–progress assumption* reflecting run–to–completion. The unique contribution of the RealType project was to present, for the first time, a *hierarchical* and *compositional* model for GSLA scheduling. The developed CSA–based model decomposes the global run–to–completion scheduling into local interface descriptions, i.e., reactive types, assigned to individual components and permits jam analysis via *behavioural equivalence checking* [15, 22, 23].

As in other synchronous languages, such as Esterel and Statecharts, the global nature of synchronous encapsulation in GSLA scheduling naturally leads to positive and negative dependencies of control on the occurrence of signals. This poses a tangled and deep compositionality problem which had not been satisfactorily addressed in the literature before. In the context of the RealType project, therefore, this problem was studied extensively, covering not just ICONNECT but synchronous languages in general. This resulted in novel *constructive semantics* based on intuitionistic Kripke models and game graphs published in [2, 13, 14, 15, 16, 17]. That work proposes solutions to the compositionality problem and provides a semantic framework that can serve as a basis of a component model for the Esterel and Statecharts' family of languages, which addresses a similar application domain in embedded systems design as the ICONNECT tool. The model adopted for ICONNECT [15, 22, 23] differs from that for Esterel, in particular, in that *recurrence* of signals within one synchronous step is also explicitly dealt with.

In view of the fact that reactive types are static descriptions of interface behaviour and as such always constitute an abstraction in time and data of the actual component, it was anticipated in the project proposal that *Lax Logic* [8] would be used to tackle the subtle issue of abstraction constraints in a type–theoretic framework. This track has been followed, too, but not yet been integrated with the coordination model mentioned above. In [9], it was demonstrated that the modal operator of Lax Logic indeed captures a general notion of constraint context, thereby establishing Lax Logic as a *logic of constraints* and solving an open problem originally raised by Curry on constructive modalities. Relating to the envisaged application within RealType, a special constraint interpretation of the Lax Modality was investigated in [10], where it was shown that complete timing information can be extracted from proofs in Lax Logic.

GSLA Modelling — Course of Actions. To build a compositional coordination model, within which each component may exhibit independent state and make nondeterministic choices, we followed as proposed an approach based on the temporal process algebra CSA [7]. In the following, a more detailed description shall be given of the developments of our process–algebraic coordination model for compositional GSLA scheduling, which may be understood as a reactive–types language. These developments are listed below in chronological order, highlighting how the project progressed over time and pointing our relevant publications, talks and visits within the RealType project.

The first project year was largely spent by the post–graduate RA, B. Norton, to familiarise himself with DSPC tools, in particular with ICONNECT and GSLA scheduling, and with temporal process algebra, in particular CSA. At the same time, Norton developed first CSA models that captured various aspects of GSLA scheduling, including run–to–completion and isochrony, and demonstrated how these may be implemented in verification tools. These ideas were relayed to the collaborators at a visit to Passau (27th January – 3rd February 2001), which was attended by personnel from Micro–Epsilon GmbH and at which two talks were given. The first talk on the use of CSA to model ICONNECT was refined into a presentation for BCTCS17 in Glasgow [20]; technical details can be found in Norton's MPhil/PhD Transfer Report [21]. This work shows how both the synchronous nature of systems scheduled in ICONNECT and the *isochronic broadcast* nature of the produced system outputs could be captured using CSA. The second talk focused on how the CSA model could be implemented in the functional language Haskell, how this model could be interfaced to the Microsoft software platform employed by Micro–Epsilon GmbH, and also on how this model could form the basis of a GSLA scheduler itself. The strategy for implementation was further discussed in meetings at Micro–Epsilon GmbH's offices in Ortenburg (30th July — 12th August 2001), where it was also negotiated that source code for various ICONNECT modules would be made available to the project in order to program a realistic demonstrator. The details on the resulting implementation of this demonstrator became a report which is available on the RealType web site [19] that is maintained by Norton.

The second project year focused on elaborating on the idea of reactive types and on exploring the utility of *temporal observation equivalence* [7], which underlies the semantic theory of CSA, for checking jams at compile–time rather than at run–time. As basis of this work served the insight that CSA models describing the *interfaces* of components, together with the constraints imposed by the GSLA scheduling discipline,

may be viewed as behavioural types to those components. In this context, *type inference* would be the interpretation of the act of compositionally deriving a model for a DSPC system alongside its creation, that *type assignment* would be natural alongside encapsulation in creating hierarchical systems, and that *type checking* would be appropriate at this point. It was observed that type checking would naturally be based on CSA's temporal observation equivalence since this is insensitive to internal actions whilst at the same time being sensitive to jams. In further analogy to the fundamental ideas of type theory we noted that just as well–typedness can guarantee termination in certain lambda calculi where reduction is the fundamental computational operation, so the aim of well–typedness in this system would be *reactivity*. These ideas have been refined and presented at the workshop on Automated Verification of Critical Systems (AVoCS '02) [22].

The beginning of the final project year saw the hand–over of the project's leadership from Mendler, who took up a chair at the University of Bamberg, Germany, to Luettgen. After assessing the status of the project, the new PI decided on finishing off the work of the first two project years by eliminating two weaknesses of the developed CSA model for GSLA scheduling, which already emerged during the second project year. First, the model needed to be extended to a fully–fledged coordination model. In particular, this concerned including the ability of modelling hierarchy, or encapsulation, in signal–flow graphs, which had been ignored so far in order not to overly complicate matters for the quite inexperienced RA. Second, the semantic theory underlying jam detection needed to be completed by providing an *axiomatisation* of temporal observation equivalence. Unexpectedly, dealing with both issues proved to be extremely challenging and thus filled out the full final project year. Addressing the first weakness involved introducing a *clock–hiding* operator to CSA, which had not been attempted before for temporal process algebras incorporating the maximal–progress assumption. Several possible operators, reflecting different intuitions about encapsulation in the presence of time determinism, needed to be evaluated for their utility for the DSPC domain and for their implications, regarding compositionality and full–abstraction, for CSA's semantic theory. Dealing with the second issue on axiomatisation turned out to be quite problematic, before it was found that no simple finitary axiomatisation of CSA could exist for which the traditional proof schemes for Milner's CCS, of which CSA is a conservative extension, would work. This insight came as a surprise as it contradicts what related work [5] suggests. To keep the technical framework of RealType simple, it was decided to identify a suitable fragment of CSA, expressive enough for modelling ICONNECT's GSLA scheduling but which admits a conventional style of axiomatising behavioural congruence. This fragment was found during a visit Luettgen and Norton made to Prof. Cleaveland in Washington (13th September 2002) and to the ICASE Research Institute at NASA Langley Research Center (14th – 26th September 2002). In a nutshell, the notion of maximal progress in CSA was strengthened in a way that still permits one to model centralised systems, which is sufficient since DSPC applications in ICONNECT are scheduled on uniprocessor architectures. This approach avoids the experienced difficulties in the observational theory mentioned above and also simplified the definition of a clock–hiding operator. In a subsequent visit by Norton to Mendler in Bamberg (30th September – 9th October 2002), the semantic theory for the new calculus was largely completed. This included the first proof of *full–abstractness* of an observational theory with clock hiding. The resulting algebra, CaSE, was successfully employed to compositionally model GSLA scheduling for signal–flow graphs with hierarchy. This work will be presented at the annual international conference on Concurrency Theory (CONCUR '03) [23]; it pioneers the CSA–based GSLA scheduling model as a framework for reactive types. The original Haskell implementation, developed at the end of the first project year, has also been reworked as a demonstrator, now implementing CaSE. This demonstrator has been presented successfully to Micro–Epsilon GmbH (11th July 2003), as is documented by the collaborator's attached letter of appraisal. The axiomatisation of CaSE will be included in Norton's PhD thesis, which should be submitted by the end of this year.

Project Plan Review. The project has undergone a major shift in objectives during the period of funding. As a result, the project could not maintain the workplan and work packages originally announced in the proposal. The reasons for the deviation were twofold:

First, the original plan of extending CSA by quantitative temporal information and, based on that, the development of timing analysis algorithms was given up early on. This became necessary after both our industrial partner and the academic collaborator abandoned plans for re–implementing the reference tools on a real–time platform during the first year [1]. Instead, their focus shifted to the problem of statically analysing *jams*. Since this required a faithful and much deeper modelling of the GSLA scheduling mechanism, as opposed to abstract worst–case timing analysis, the project had to embark on more serious studies into

process–algebraic theory than was anticipated. Along with that, behavioural equivalences and axiomatisation of a language of reactive types based on CSA clocked transition systems became central to the work, while the idea to cover ordinary functional data types and timing information was postponed. As a result of the shift in focus, it also became necessary for the RA to spend much more time for acquiring a deep theoretical knowledge in process theory, before he could contribute substantially to the project.

Second, the PI Mendler took up a chair at the University of Bamberg, Germany, in April 2002. Although new Anglo–German agreements would have permitted the transfer of the grant to Bamberg, Mendler handed over to Luettgen as PI since the employed RA Norton was not able to move due to personal reasons. This had a couple of implications for the project's focus. On the one hand, Luettgen's expertise in temporal process algebra greatly helped with developing CaSE. On the other hand, Mendler's experience in type theory was no longer present, which implied a further shift of the project towards the semantic foundations for GSLA scheduling. In December 2002, unforeseen by both Mendler and Luettgen, the new PI left Sheffield for the University of York. This time, Norton was prepared to move, but unfortunately the EPSRC, despite our request, did not judge it possible to transfer the grant to York. Obviously, these circumstances caused some disruptions in the management of RealType.

Despite the deviation from the original objectives and workplan, the RealType project has made substantial scientific progress in the compositional semantics of synchronous languages. This laid the foundation for developing a synchronous, compositional and hierarchical coordination model for DSPC applications and ICONNECT, which establishes a framework for reactive types. As this was at the core of the project proposal, RealType's main objectives have clearly been achieved. Unfortunately, the necessary change in workplan meant that there was not enough time and resources to investigate other matters originally proposed, including the conduct of a real–world case study in collaboration with Prof. M. Cooke at Sheffield.

Explanation of Expenditure. Since the employment of a named RA was the main item of expenditure in the proposal, this has been substantially as planned. The decision to investigate in depth the process–algebraic approach and to put aside the analysis part have saved on software costs, since the actual tools used, such as Haskell and Microsoft Visual Studio, were either free–of–charge or available to the project on campus licences. The anticipated expensive licence for iLogic's constraint–solver library was not acquired. Therefore, consumables and equipment ended up underspent. Furthermore, travel costs were reduced by a collaboration grant from the British Council under project number PRO1163/CH, which funded some visits to the collaborators in Passau.

Research Impact and Benefits. The potential research impact of the project on component–based development is very high. Tools like Ptolemy and ICONNECT have proven very successful and useful to the signal–processing and embedded–systems communities, yet most of them still are without the validation support for dataflow–oriented systems in which state and nondeterministic choice play a major part. As well as enabling such validation in a static fashion, the semantic basis established in this project has been shown capable of dealing with the other problems that the application of our coordination model to ICONNECT has brought out [21]. The accompanying letter of appraisal from Micro–Epsilon GmbH that is attached to this Grant Review Report may be taken to witness the success of this programme. As indicated above, the *behavioural types* approach is clearly now gaining credibility and acceptance in the application of formal methods to software development [12]; we also cite its addition as a new topic to the call for papers of the international conference on Formal Methods for Open Object–based Distributed Systems (FMOODS) and anticipate more such interest in the future. Specifically, the GSLA reactive type developed in the project, viewed as a compositional coordination model, opens up a distinct new research direction that would mirror current activities in the area of globally asynchronous and locally synchronous systems (GALS).

The RealType project has produced several peer–reviewed publications, among which are 7 papers in international workshops and conferences as well as 2 journal articles, and was the centre of several talks at university seminars, including one at the University of Sussex (27th June 2003). During the project's funding period, an international workshop on *Semantic Foundations of Engineering Design Languages* (SFEDL '02) was organised successfully by the PIs as a satellite event to ETAPS '02 in Grenoble, France. A second workshop on the same topic is now scheduled in conjunction with ETAPS '04. Moreover, Luettgen and Mendler, in cooperation with Manfred Broy, are guest editors for a special issue of the *Formal Aspects of Computing Journal* on this topic [6]. This special issue attracted a large number of high–quality contributions, so that it will be bound in two volumes.

Further Research and Dissemination. A few work packages within the RealType project are still in the process of being finalised, including an axiomatisation of CaSE's semantic theory. Their results will first be published as part of Norton's PhD thesis, and later in the form of conference papers and journal articles. The work conducted within RealType already pushes our research towards the up and coming field of *coordination languages*, especially since our coordination model may easily be extended to one where optional inputs may appear as a first–class feature. Thus, a coordination language for synchronous systems with possible serialised execution, with instantiation and encapsulation as first–class operations, and where optional input and isochronic output are paired and taken together, may be imagined. As stated in the attached letter, our industrial partners Micro–Epsilon GmbH have expressed their continued support of these activities in the future.

References

[1] Minutes of the first project meeting at Passau, 29th January – 2nd February 2001.

[2] J. Aguado, G. Lüttgen, and M. Mendler. A-mazeing Esterel. In *Workshop on Synchronous Languages, Applications, and Programming (SLAP '03)*, vol. 88 of *ENTCS*. Elsevier Science, 2003.

[3] G. Berry and G. Gonthier. The Esterel synchronous programming language: Design, semantics, implementation. *SCP*, 19(2):87–152, 1992.

[4] S. Bhattacharya. Software synthesis and code generation for signal processing systems. *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, 47(9):849–875, 2000.

[5] M. Bravetti and R. Gorrieri. A complete axiomatization for observational congruence of prioritized finite-state behaviours. In *27th Intl. Coll. on Automata, Languages and Programming (ICALP '00)*, vol. 1853 of *LNCS*, pp. 744–755. Springer-Verlag, 2000.

[6] M. Broy, G. Lüttgen, and M. Mendler. Semantic foundations of engineering design languages. *Formal Aspects of Computing*, 2003. Special issue; to appear.

[7] R. Cleaveland, G. Lüttgen, and M. Mendler. An algebraic theory of multiple clocks. In *8th Intl. Conf. on Concurreny Theory (CONCUR '97)*, vol. 1243 of *LNCS*, pp. 166–180. Springer-Verlag, 1997.

[8] M. Fairtlough and M. Mendler. Propositional lax logic. *Inform. and Comp.*, 137(1):1–33, 1997.

[9] M. Fairtlough and M. Mendler. On the logical content of computational type theory: A solution to Curry's problem. In *Types for Proofs and Programs*, vol. 2277 of *LNCS*, pp. 63–78. Springer-Verlag, 2002.

[10] M. Fairtlough and M. Mendler. Intensional completeness in a fragment of Gödel-Dummett logic. *Studia Logica*, 73:51–80, 2003.

[11] D. Harel. Statecharts: A visual formalism for complex systems. *SCP*, 8:231–274, 1987.

[12] E.A. Lee and Y. Xiong. A behavioural type system and its application to Ptolemy II for component-based design. In SFEDL [6]. Submitted.

[13] G. Lüttgen and M. Mendler. Fully-abstract Statecharts semantics via intuitionistic Kripke models. In *27th Intl. Coll. on Automata, Languages and Programming (ICALP '00)*, vol. 1853 of *LNCS*, pp. 163–174. Springer-Verlag, 2000.

[14] G. Lüttgen and M. Mendler. Statecharts: From visual syntax to model-theoretic semantics. In *Workshop on Integrating Diagrammatic and Formal Specification Techniques (IDFST '01)*, vol. 1, pp. 615–621. Austrian Computer Society, 2001.

[15] G. Lüttgen and M. Mendler. Axiomatizing an algebra of step reactions for synchronous languages. In *13th Intl. Conf. on Concurrency Theory (CONCUR '02)*, vol. 2421 of *LNCS*, pp. 386–401. Springer-Verlag, 2002.

[16] G. Lüttgen and M. Mendler. The intuitionism behind Statecharts steps. *ACM Trans. on Computational Logic*, 3(1):1–41, 2002.

[17] G. Lüttgen and M. Mendler. Towards a model theory for Esterel. In *Workshop on Synchronous Languages, Applications, and Programming (SLAP '02)*, vol. 65,5 of *ENTCS*. Elsevier Science, 2002.

[18] W. Maydl, B. Sick, and W. Grass. Towards a specification technique for component-based measurement and control software for embedded systems. In *28th EUROMICRO Conf.*, pp. 74–80. IEEE Press, 2002.

[19] B. Norton. Haskell as a controller for reactive components. www.dcs.shef.ac.uk/~barry/RealType/Nor00.pdf, 2000.

[20] B. Norton. Clocked transition systems and the compositional modelling of reactive components under synchronous scheduling. *EATCS Bulletin*, 74, 2001.

[21] B. Norton. Reactive types for component-based development. www.dcs.shef.ac.uk/~barry/RealType/Nor01.pdf, 2001.

[22] B. Norton. Reactive types for dataflow-oriented component-based development. In *2nd Workshop on Automated Verification of Critical Systems (AVoCS '02)*, vol. CSR-02-6 of *Techn. Report Series*, pp. 195–209. Univ. of Birmingham, 2002.

[23] B. Norton, G. Lüttgen, and M. Mendler. A compositional semantic theory for synchronous component-based design. In *14th Intl. Conf. on Concurreny Theory (CONCUR '03)*, LNCS. Springer-Verlag, 2003. To appear.

[24] A. Sicheneder, A. Bender, E. Fuchs, M. Mendler, and B. Sick. Tool-supported design and program execution for signal processing applications using modular software components. In *Intl. Workshop on Software Tools for Technology Transfer (STTT '98)*, vol. NS-98-4 of *BRICS Notes Series*, pp. 61–70. BRICS, 1998.