04491 Abstracts Collection
Synchronous Programming - SYNCHRON'04
— Dagstuhl Seminar —

Stephen A. Edwards[1], Nicholas Halbwachs[2], Reinhard von Hanxleden[3] and Thomas Stauner[4]

[1] Columbia University, US
sedwards@cs.columbia.edu
[2] VERIMAG - IMAG, FR
[3] Universität Kiel, DE
rvh@informatik.uni-kiel.de
[4] BMW Car IT, DE
thomas.stauner@bmw-carit.de

Abstract. From 28.11.04 to 03.12.04, the Dagstuhl Seminar 04491 "Synchronous Programming - SYNCHRON'04" was held in the International Conference and Research Center (IBFI), Schloss Dagstuhl. During the seminar, several participants presented their current research, and ongoing work and open problems were discussed. Abstracts of the presentations given during the seminar as well as abstracts of seminar results and ideas are put together in this paper. The first section describes the seminar topics and goals in general. Links to extended abstracts or full papers are provided, if available.

Keywords. Synchronous languages, executive summary, Esterel, Lustre, Signal, semantics, programming languages, real-time systems

04491 Executive Summary – Synchronous Programming - SYNCHRON'04

This seminar was the 11th in a series of semi-annual workshops on the Synchronous Languages (Esterel, Lustre, and Signal). These languages were invented in the early 1980's to make the programming of reactive systems easier. The goal of the seminar was to bring together researchers and practitioners of synchronous programming, and furthermore to reach out to relevant related areas and industrial users.

Keywords: Synchronous languages, executive summary, Esterel, Lustre, Signal, semantics, programming languages, real-time systems

Joint work of: Edwards, Stephen A; Halbwachs, Nicholas; von Hanxleden, Reinhard; Stauner, Thomas

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2005/195

Dagstuhl Seminar Proceedings 04491
Synchronous Programming - SYNCHRON'04
http://drops.dagstuhl.de/opus/volltexte/2005/196

Playing Charts

Joaquin Aguado (Universität Bamberg, D)

This talk is about some results towards a game-theoretic account of the semantics of synchronous languages, which provides a framework encompassing both nondeterministic Statecharts (as per Pnueli and Shalev) and deterministic Esterel-SyncCharts.

Keywords: Game-theory, semantics, synchronous languages

Joint work of: Mendler, Michael; Aguado, Joaquin

Heterogeneous Reactive Systems

Albert Benveniste (IRISA/INRIA Rennes, F)

I shall present in detail our recent collective work on Heterogeneous reactive systems and tagged systems. This is a generalization and systematisation of our previous work on desynchronisation. It will be appropriate to put this presentation after the one on SHIM by Stephen Edwards, as I shall be using it.

Keywords: Synchronous, asynchronous, heterogeneous, tag, reactive systems, deployment

Joint work of: Benveniste, Albert; Caillaud, Benoit; Carloni, Luca; Caspi, Paul; Sangiovanni-Vincentelli, Alberto

Extending SCADE/LUSTRE with Automata.

Jean-Louis Colaco (Esterel Technologies - Toulouse, F)

This talk is about the extension of LUSTRE, some on the data flow side (merge and reset) and another more important: automata. A simple but not completely trivial example will be treated and the main ideas of the compilation will be sketched.

Keywords: LUSTRE, SCADE, data-flow, automata, compilation

Joint work of: Colaco, Jean-Louis; Pagano, Bruno

Nemo: A Domain-Specific Language for the Description of Multi-Task Systems

Gwenáël Delaval (INRIA Rhône-Alpes, F)

We propose a programming language specific to the domain of multi-task real-time control systems, such as in robotic, automotive or avionics systems. It can be used to specify a set of resources, with usage constraints, and a set of tasks, that use them, as well as imperative applications requesting them in sequence, and explicit temporal properties. We then obtain automatically, through a compilation process including a phase of discrete controller synthesis, an application-specific task handler that satisfies the constraints (if there exists one).

Keywords: Real-time systems, safe design, domain-specific language, discrete control synthesis, synchronous programming

Joint work of: Delaval, Gwenaël; Rutten, Éric

SHIM: A Language for Hardware/Software Integration

Stephen A. Edwards (Columbia University, USA)

Virtually every system designed today is an amalgam of hardware and software. Unfortunately, software and circuits that communicate across the hardware/software boundary are tedious and error-prone to create. This suggests a more automatic way to synthesize them.

I present the SHIM language, which combines imperative C-like semantics for software and RTL-like semantics for hardware to allow a unified description of hardware/software systems. Hardware processes and software functions communicate through shared variables, hardware for which is automatically synthesized by the SHIM compiler, which generates C and synthesizable VHDL.

I demonstrate the effectiveness of the language by re-implementing an I2C bus controller. The SHIM source is half the size of an equivalent manual implementation, slightly faster, and has a smaller memory footprint. Partial and complete hardware implementations in SHIM are also presented, showing that SHIM is succinct and effective.

Keywords: Hardware/software codesign, synchronous, asynchronous, language design, integration

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2005/158

Uncertainties and Nondeterminism in UML 2.0 State Machines

Harald Fecher (Universität Kiel, D)

UML 2.0 state machines, which are modifications of Harel's statecharts, are used to model the behavior of objects. They allow the concepts of synchronous and asynchronous communication.

Unfortunately their semantics is only described informally in the UML 2.0 standard.

This talk presents uncertainties in the semantics of UML 2.0 state machines. Furthermore, (intended) nondeterminism of UML 2.0 statt machines are execution) and are formalized.

Keywords: UML, state machine, formal semantics

IMA-based Design within Polychrony: Current Status

Abdoulaye Gamatié (Université de Rennes, F)

The work exposed here concerns the design of avionic applications based on Integrated Modular Avionics (IMA) model, within Polychrony, the academic design environment associated with the synchronous language Signal.

We first give an overview of the previous results.

Then we address the ongoing part of this work. In particular, we focus on partitioning issues.

Keywords: Modeling, integrated modular vvionics, synchronous approach, ARINC, partitioning, signal, design methodology

Towards a Higher-Order Synchronous Data-Flow Language

Alain Girault (INRIA Rhône-Alpes, F)

The paper introduces a higher-order synchronous data-flow language in which communication channels may themselves transport programs. This provides a mean to dynamically reconfigure data-flow processes. The language comes as a natural and strict extension of both Lustre and Lucy. This extension is conservative, in the sense that a first-order restriction of the language can receive the same semantics.

We illustrate the expressivity of the language with some examples, before giving the formal semantics of the underlying calculus. The language is equipped with a polymorphic type system allowing types to be automatically inferred and a clock calculus rejecting programs for which synchronous execution cannot be statically guaranteed. To our knowledge, this is the first higher-order synchronous data-flow language where stream functions are first class citizens.

Keywords: Synchronous data-flow programming language, stream functions, Kahn processes, functional programming, dynamic reconfiguration, type systems

Joint work of: Colaço, Jean-Louis; Hamon, Grégoire; Girault, Alain; Pouzet, Marc

Runtime Verification for Detection of Scheduling Dependencies in SystemC

Claude Helmstetter (VERIMAG - IMPG, F)

We present a new approach to detect scheduling dependencies in SystemC programs. For a medium size SystemC program, there are too many possible schedulings to test them all. However, lots of distincts schedulings can be proved to produce the same result. Our approach is based on observing operations on communication objects during execution to generate only a small subset of all possible schedulings. With this subset, we are able to find all deadlocks and assertion violations. Our solution has some similarities with partial order reduction techniques.

Keywords: SystemC, Scheduling, Runtime verification, Test
Joint work of: Helmstetter, Claude (Verimag & STMicroelectronics); Maraninchi, Florence (Verimag); Maillet-Contoz, Laurent (STMicroelectronics)

Sanity Checks for Stateflow Diagrams

Ralf Huuck (National ICT Australia - Eveleigh, AU)

Simulink/Stateflow is the most accepted programming and design language in the area of control engineering for safety critical systems. The semantics is to a certain extend clearly defined, but often complex and rather counter-intuitive. We present ongoing work on detecting stateflow diagrams exhibiting potential flaws due to those aspects.

Lurette V2 : Hispano-SUIza Cases Studies

Erwan Jahier (VERIMAG - IMPG, F)

We will illustrate the use of the automated testing tool Lurette with some the case studies we have worked on within the SAFEAIR II project, in particular the ones provided by hispano-suiza.

Keywords: Automated testing, tool environment, real-time embedded systems, reactive programs, synchronous languages, stochastic machines

Joint work of: Jahier, Erwan; Raymond, Pascal

Towards a Proof Theory for UML 2.0 Action Semantics

Marcel Kyas (Universität Kiel, D)

This presentation depends on the presentation of Harald Fecher

In this talk we present the basic concepts of the UML 2.0 Action Semantics (AS), which is an abstract syntax of a programming language together with an informally defined semantics. We discuss the notion of atomicity of actions and the notion of progress of time, as well as a formal semantics for actions, as a structured operational semantics and as an axiomatic semantics.

We observe that AS shares properties with SDL but also with data flow oriented languages like Lustre. In AS the notion of an atomic action is defined in a flexible manner, we propose constraints which may allow the use of AS as a synchronous language.

The contents of this presentation is work in progress.

Keywords: UML 2.0 Action Semantics

The Kiel Esterel Processor—A Semi-Custom, Configurable Reactive Processor

Xin Li (Universität Kiel, D)

The synchronous language Esterel is an established language for developing reactive systems. It gives an abstract, well-defined and executable description of the application, and can be synthesized into hardware and software. Typically, an Esterel program is first translated into other, lower-level languages (such as VHDL or C), and then compiled further. However, there is also the alternative of executing Esterel-like instructions directly. For example, in the REFLIX and RePIC projects, Roop et al. have augmented traditional processors with custom hardware to execute Esterel instructions. This patch strategy is a convenient approach, but has some shortages.

We present the Kiel Esterel Processor (KEP), a semi-custom, configurable reactive processor for the direct execution of Esterel programs. It consists of a reactive core and scalable peripheral elements. KEP supports standard Esterel statements directly, except (so far) for the concurrency operator. Valued signals and counter functions in Esterel statements are supported by KEP. Due to its control path and its cooperation with elements, KEP obeys exact Esterel (preemption and priority) rules, including for example abort/weak abort (nests).

Keywords: Esterel, synchronous languages, reactive programming, ASIPs

Joint work of: Li, Xin; Von Hanxleden, Reinhard

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2005/159

Removing Cycles in Esterel Programs

Jan Lukoschus (Universität Kiel, D)

Programs written in the synchronous programming language Esterel may contain statically cyclic dependencies of signals, which inhibits the application of certain compilation approaches that rely on static scheduling. This talk proposes an algorithm which, given a constructive synchronous program, performs a semantics-preserving source-level code transformation that removes cyclic signal dependencies. The transformation exploits the monotonicity of constructive programs, and is illustrated in the context of Esterel, but should be applicable to other synchronous languages as well.

Keywords: Synchronous Languages, compilation, cyclic circuits, constructiveness, Esterel

Joint work of: Lukoschus, Jan; von Hanxleden, Reinhard

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2005/160

Simulation of Age- and Position-Based Protocols for Mobile Ad-hoc Networks in ReactiveML

Louis Mandel (Université Paris VI, F)

This talk presents a programming experiment of a complex network routing protocol for mobile ad hoc networks within the ReactiveML language.

Mobile ad hoc networks are highly dynamic networks characterized by the absence of physical infrastructure. In such networks, nodes are able to move, evolve concurrently and synchronize continuously with their neighbors.

Due to mobility, connections in the network can change dynamically and nodes can be added or removed at any time. All these characteristics – concurrency with many communications and the need of complex data-structure – combined to our routing protocol specifications make the use of standard simulation tools (e.g., NS, OPNET) inadequate and network protocols appear to be very hard to program efficiently in conventional programming languages.

We show that the synchronous reactive model, as introduced in the pioneering work of Boussinot, matters for programming such systems. This model provides adequate programming constructs – namely synchronous parallel composition, broadcast communication and dynamic creation – which allow for a natural implementation of the hard part of the simulation.

The implementation has been done in ReactiveML, an embedding of the reactive model inside a statically typed, strict functional language. ReactiveML provides reactive programming constructions with most of the features of OCaml. Moreover, it provides an efficient execution scheme for reactive constructs which

made the simulation of real-size examples feasible. Experimental results show that the ReactiveML implementation is two orders of magnitude faster than the original C version; it was able to simulate more than 1000 nodes where the original C version failed (after 200 nodes) and is twice faster than the version programmed in NAB.

Joint work of: Mandel, Louis; Benbadis, Farid; Pouzet, Marc

Temporal Refinement for Lustre

Jan Mikac (VERIMAG - IMPG, F)

We propose a general stepwise refinement scheme for a large class of systems including Lustre programs. Then, we "customise" the calculus to Lustre by restraining the scope. Practical considerations on the effective feasability of the refinement proof obligations lead us to further constraints on the calculus, so that we obtain a final form which provides (i)an effective stepwise refinement (ii)with the possibility of replacing a program on some clock by a program on a “quicker” clock, thus realizing a refinement of time.

Keywords: Lustre, refinement

Joint work of: Mikac, Jan; Caspi, Paul

A Generalised Synchronous Model for Software Coordination

Barry Norton (University of Sheffield, GB)

We will present a synchronous model that generalises both an existing dataflow-oriented model for component-based software development and the synchronous model of Esterel. The resulting model allows the zero-time abstraction to be made on multiple levels, hierarchically, and therefore allows both absence of signals and signal recurrence to be treated systematically.

Joint work of: Norton, Barry

Realization Theory for Linear Switched Systems

Mihaly Petreczky (CWI - Amsterdam, NL)

Realization theory is one of the major problems of system theory.The aim of the realization theory is to construct a system with an input/output behavior given in advance. In particular, an important problem is to construct the minimal system having a certain input/output behavior. Here minimality can be understood in many different ways. For those systems which are defined on a state-space for which the notion of dimension is defined, minimal systems are understood to be the systems with the smallest possible state-space dimension. An example of a successfully developed realization theory is the realization of regular languages by finite automata.Realization theory has been developed for linear, general non-linear, bilinear systems and automata.

The talk deals with the realization theory of linear switched systems. Linear switched systems are a special case of hybrid systems. A linear switched system is just a finite collection of continuous-time time-invariant linear systems.

The system evolution takes place by switching between the linear systems. The switching is initiated externally, in fact it can be regarded as part of the input.

In this talk necessary and sufficient conditions are formulated for a family of input-output maps to be realizable by a linear switched system. Characterization of minimal realizations is presented.The paper treats two types of linear switched systems. The first one is when all switching sequences are allowed. The second one is when only a subset of switching sequences is admissible, but within this restricted set the switching times are arbitrary. The latter one covers the case of linear switched systems with switching controlled by a finite automaton such that the automaton is known in advance.

Keywords: Realization theory, hybrid systems, switched linear systems, rational formal power series, minimal realization

Joint work of: Petreczky, Mihaly

Correct-by-Construction Asynchronous Implementation of Modular Synchronous Specifications

Dumitru Potop-Butucaru (Université de Rennes, F)

In this paper, we introduce a new model for the representation of asynchronous implementations of synchronous specifications.

The model covers classical implementations, where a notion of global synchronization is preserved by means of signalling, and globally asynchronous, locally synchronous (GALS) implementations where the global clock is removed.

The new model offers a unified framework for reasoning about two essential correctess properties of an implementation: the preservation of semantics and the absence of deadlocks. We use it to derive criteria insuring the correct deployment of synchronous specifications over GALS architectures.

As the model captures the internal concurrency of the synchronous specification, our criteria support implementations that are less constrained and more efficient than existing ones.

Keywords: Globally Asynchronous Locally Synchronous, GALS, synchronous, asynchronous, microstep, desynchronization, deadlock-free, correct-by-construction deployment, concurrency

Joint work of: Potop-Butucaru, Dumitru; Caillaud, Benoit; Benveniste, Albert

Towards Robust Distribution of Synchronous Programs

Jan Romberg (TU München, D)

Towards Robust Distribution of Synchronous Programs Jan Romberg, TU Muenchen

For developing real-time control systems software, the synchronous paradigm is potentially attractive for both small-scale and large-scale development: the uniform logical time axis and deterministic notion of composition inherent in synchronous formalisms has the potential to considerably facilitate behavioral analysis. However, in the Automotive sector, the existing implementation architectures such as CAN for communication and OSEK VDX operating system are not particularly suited for implementation of synchronous designs.

We propose a method where each distributable synchronous flow in a program is associated with a particular degradation strategy. For most of the system's running time, the behavior of the distributed executive will implement the synchronous semantics correctly. The degradation strategy is of importance in cases when the real-time executive fails to meet a given deadline, loses a message, or similar. We show how the method could be put to work with an implementation scheme for programs with state-semantics flows on top of event-triggered protocols such as CAN.

Nemo: A Domain-Specific Language for the Description of Multi-Task Systems

Eric Rutten (INRIA Rhône-Alpes, F)

We propose a programming language specific to the domain of multi-task real-time control systems, such as in robotic, automotive or avionics systems. It can be used to specify a set of resources, with usage constraints, and a set of tasks, that use them, as well as imperative applications requesting them in sequence, and explicit temporal properties. We then obtain automatically, through a compilation process including a phase of discrete controller synthesis, an application-specific task handler that satisfies the constraints (if there exists one).

Keywords: Discrete control synthesis, multi-task system, domain-specific language

Joint work of: Delaval, Gwenaël; Rutten, Eric

Software Development from the Point of View of Process Development

Stefan-Alexander Schneider (BMW AG - München, D)

Main tasks for production code generation in an industry environment.

Keywords: Software development, process development

RISE Project Summary

Norman R. Scaife (VERIMAG - IMPG, F)

We summarize our experience in the IST RISE (Reliable Inovative Software for Embedded systems) project. Firstly, we show analysis and translation of imperative features from Stateflow into Lustre.

Secondly, we present a communications protocol which allows preservation of idealised semantics in applications derived from it.

Keywords: Stateflow Lustre Event-triggered Time-triggered

Joint work of: Scaife, Norman R.; Caspi, Paul; Maraninchi, Florence; Tripakis, Stavros; Sofronis, Christos

Challenges of Automotive Software Engineering

Thomas Stauner (BMW Car IT, D)

The talk outlined the main characteristics of current automotive software systems. Software today is a driving factor for innovations in the automotive domain. There are already about 100MB of Software in current premium cars and the past exponential growth in memory size and complexity is expected to continue. Automotive software engineering has to cope with this complexity in an environment that is heterogeneous, ranging from multimedia entertainment to hard real-time applications, like engine control. From the technological side, trends are centralization of functionality on less computation units, usage of open standards, model-based development and layering/abstraction (cf. Autosar standard). The increasing degree of abstraction together with the ongoing move to deterministic architectures, like Flexray, provides a basis for the application of synchronous languages.

Keywords: Automotive software, trends, application of synchronous languages

A Generic Framework to Model Embedded Software using Multi-Clocked Synchrony

Jean-Pierre Talpin (INRIA Rennes, F)

We propose a framework based on a synchronous multi-clocked model of computation to support the inductive and compositional construction of scalable behavioral models of embedded systems engineered with de facto standard design and programming languages. Behavioral modeling is seen under the paradigm of type inference. The aim of the proposed type system is to capture the behavior of a system under design and to re-factor it by performing global optimizing and architecture-sensitive transformations on it. It allows to modularly express a wide spectrum of static and dynamic behavioral properties and automatically or manually scale the desired degree of abstraction of these properties for efficient verification. The type system is presented using a generic and language-independent static single assignment intermediate representation.

References

– “Compositional behavioral modeling of embedded systems and conformance checking”. Talpin, J.-P, Le Guernic, P., Shukla, S., Gupta, R. In International Journal on Parallel processing, special issue on testing of embedded systems. Kluwer Academic Publishers, 2005.

– “Formal refinement checking in a system-level design methodology”. Talpin, J.-P., Le Guernic, P., Shukla, S. K., Gupta, R., Doucet, F. Special Issue of Fundamenta Informaticae on Applications of Concurrency to System Design. IOS Press, 2004.

- "Polychrony for system design". Le Guernic, P., Talpin, J.-P., Le Lann, J.-C. Journal for Circuits, Systems and Computers. Special Issue on Application Specific Hardware Design. (c) World Scientific, 2003.
- "Modular design through component abstraction". Berner, D., Talpin, J.-P., Le Guernic, P., Shukla, S. K. International conference on compilers, architectures and synthesis for embedded systems. ACM Press, 2004
- "A behavioral type inference system for compositional system-on-chip design". Talpin, J.-P., Berner, D., Shukla, S. K., Gamatié, A., Le Guernic, P., Gupta, R. Application of Concurrency to System Design. IEEE Press, 2004.

Joint work of: Talpin, Jean-Pierre;Le Guernic, Paul; Shukla, Sandeep; Gupta, Rajesh

SharpHDL—A Hardware Description Language Embedded in C#

Christine Vella (University of Malta, SEU)

Embedded domain specific languages have been shown to be useful in various domains. One particular domain in which this approach has been applied is hardware description languages. We present SharpHDL, a language embedded in C# which enable us to describe structurally large regular circuits in an intuitive way. Descriptions can then be automatically used in simulators and verification tools. We show the versatility of the approach by writing a hardware compiler for regular expressions circuits in our language.

We also discuss future work with this language. This consists of extending it to include placement information and description of other non-functional circuit properties. SharpHDL will also be used to implement FFT algorithms using butterfly circuits and to develop a framework where behavioural languages can be embedded and combined.

Keywords: HDL, embedded languages, C#, hardware compilation

Accessing Databases within Esterel

David White (University of York, GB)

A current limitation of the Esterel language for reactive-systems »design is its lack of support for accessing databases. This talk »presents the results of a summer student project which investigated »a way of integrating databases and Esterel by providing an API for »database use inside Esterel. » »A case study, involving a warehouse storage system built using Lego »Mindstorms robotics kits, demonstrates the utility of the API. This »system employs an Esterel-programmed robot whose task it is to collect »various items from a customer's order and assemble them in one place. »To do so, the robot accesses customer-order data and floor-plan data »stored in a database.

Keywords: Database esterel lego mindstorms

Joint work of: White, David; Luettgen, Gerald

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2005/161

Timing Analysis of Synthesized Code

Reinhard Wilhelm (Universität Saarbrücken, D)

Hard real-time systems need sound methods to compute bounds on execution times. We have used Abstract Interpretation combined with Integer Linear Programming successfully and hand-written and on synthesized code.

Although it is tempting to believe that synthesized code is easier to analyze due to its regularity, this is not always true. Some code synthesizers obfuscate knowledge they have about the algorithm by generating code that confuses compiler and timing analysis.

We present several case studies using popular specification languages.

Keywords: Timing analysis, hard real time, embedded system, specification language, SCADE, ASCET-SD

Joint work of: Wilhelm, Reinhard; Thesing, Stephan