INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Team FORMES

FOrmal Methods for Embedded Systems

Liama - Beijing - Chine

THEME SYM

Table of contents

FORMES is a new project started in 2008 at LIAMA, and hosted by Tsinghua university, Beijing. FORMES is carried out in cooperation with the Beijing universities of Tsinghua -our main partner- and Beihang. The project DeviceWare started by Vania Joloboff in 2007 at LIAMA is now part of FORMES. The activity of FORMES therefore reduces to the activity of DeviceWare for the most part. The new members of the team joined LIAMA in September 2008, following 3 weeks preparation in March. This explains why the research results of 3 out of the 5 INRIA members of the team are limited, most of their activity in 2008 being described in other INRIA research reports.

1. Team

Research Scientist

Frédéric Blanqui [CR1 INRIA from 01/09/08]
Vania Joloboff [DR INRIA, Team leader until 30/08/09]
Jean Pierre Jouannaud [DR INRIA, Team leader from 01/09/08, HdR]

PhD Student

Fan Ni [Beijing University of Aeronautics and Astronautics]
Fengjuan Yao [Hunan University at Changsha]

Post-Doctoral Fellow

Claude Helmstetter [Postdoc INRIA until 31/10/08]
Pierre-Yves Strub [Postdoc INRIA since 01/10/08]
Li Qianqi [Tsinghua University Postdoc following our lectures since 01/10/08]

Other

Zhang Lianyi [Tsinghua University master student following our lectures since 01/10/08]
Xiao Hui [Beijing University of Aeronautics and Astronautics]
Jiajia Song [University of Science and Technology at Beijing]
Bing Liu [Beijing University of Aeronautics and Astronautics]
Wang Kejun [Tsinghua University master student following our lectures since 01/10/08]
Wang Qian [Tsinghua University master student following our lectures since 01/10/08]

2. Overall Objectives

2.1. Overall Objectives

FORMES stands for FORmal Methods for Embedded Systems. It is one of the projects run within the LIAMA Consortium, as a cooperation project between INRIA, Tsinghua and Beihang Universities. This project is aiming at making research advances towards the development of safe and reliable embedded systems, by exploiting synergies between two different approaches, namely (real time) hardware simulation and formal proofs development.

Embedded systems have become ubiquitous in our everyday life, ranging from simple sensors to complex systems such as mobile phones, network routers, airplane, aerospace and defense apparatus. As embedded devices include increasingly sophisticated hardware and software, the development of combined hardware and software has become a key to economic success.

The development of embedded systems uses hardware with increasing capacities. As embedded devices include increasingly sophisticated hardware running complex functions, the development of software for embedded systems is becoming a critical issue for the industry. There are often stringent time to market and quality requirements for embedded systems manufacturers. Safety and security requirements are satisfied by using strong validation tools and some form of formal methods, accompanied with certification processes such as DO 178 or Common Criteria certification. These requirements for quality of service, safety and security imply to have formally proved the required properties of the system before it is deployed.

Within the context described above, the FORMES project aims at addressing the challenges of embedded systems design with a new approach, combining fast hardware simulation techniques with advanced formal methods, in order to formally prove qualitative and quantitative properties of the final system. This approach requires the construction of a simulation environment and tools for the analysis of simulation outputs and proofs of properties of the simulated system. We therefore need to connect simulation tools with code-analyzers and easy-to-use theorem provers for achieving the following tasks:

- Enhance the hardware simulation technologies with new techniques to improve simulation speed, and produce program representations that are adequate for formal analysis and proofs of the simulated programs ;
- Connect validation tools that can be used in conjunction with simulation outputs that can be exploited using formal methods ;
- Extend and improve the theorem proving technologies and tools to support the application to embedded software simulation.

A main novelty of the project, besides improving the existing technologies and tools, relies in the application itself: to combine simulation technologies with formal methods in order to cut down the development time for embedded software and scale up its reliability. Apart from being a novelty, this combination is also a necessity: proving very large code is unrealistic and will remain so for quite some time; and relying only on simulation for assessing critical properties of embedded systems is unrealistic as well.

We assume that these properties can be localized in critical, but small, parts of the code, or dedicated hardware models. This nevertheless requires scaling up the proof activity by an order of magnitude with respect to the size of codes and the proof development time. We expect that it is realistic to rely on both combined. We plan to rely on formal proofs for assessing properties of small, critical components of the embedded system that can be analyzed independently of the environment. We plan to rely on formal proofs as well for assessing correctness of the elaboration of program representation abstractions from object code. We plan to rely on simulations for testing the whole embedded system. We finally plan to rely on formal proofs again for verifying completeness of test sets. Proving properties of these various abstractions requires using an interactive theorem prover.

2.2. Highlights of the year

- The project has developed an integrated simulation framework, named SimSoC, for simulation of System-On-Chips [14]. This framework is based on SystemC and Transaction Level Modeling. The ISS are using dynamic translation of the processor binary software into an executable representation. A complete ISS has been developed for the ARM instruction set Version 5, as well as some hardware peripherals models, such as ARM interrupt controller, ARM serial line controller and ST Microelectronics flash memory controllers. executable representation, with parallel translation steps.
- The tool Rainbow/CoLoR developed by the team has been, for the second consecutive year, the best certification back-end in the 2008 international competition on certified termination provers. See http://color.loria.fr/comp.html.
- Martin-Löf first remark that typing becomes undecidable in a type-theory with an extensionnal equality. We have solved this long-standing problem in the context of Coq, a work published in the theoretical computer science conference of the World Computer Congress in Milan [11].

3. Scientific Foundations

3.1. Simulation

Keywords: *ISS*, *Instruction Set Simulator*, *System-on-Chip*, *SystemC*, *TLM*, *Transaction Level Modeling*, *cached translation*, *co-simulation*, *compilation*, *dynamic partial order reduction*, *dynamic translation*, *loose timing*, *multi-threads*, *parallelization*, *partial evaluation*, *run-time verification*, *scheduling*, *simulation*, *specialization*, *test*, *trace*, *trace analysis*.

modeling the methods and languages used to define executable abstractions for modeling hardware or embedded software components, making it possible to simulate the system.

TLM Transaction Level Modeling is an approach for modeling hardware components where details of communication among modules are abstracted. Transactions refer to operations carried by the module.

SystemC SystemC is the common name for the IEEE 1666 standard modeling language

simulation Software technique used to run the simulation of executable models, producing output and trace information during simulation sessions.

dynamic translation Dynamic translation consists in dynamically translating the machine code of programs compiled for some architecture into another representation to run the program on a different machine architecture.

The development of complex embedded systems platforms requires the assembly of many hardware components, processor cores, application specific co-processors, bus architectures, and peripherals, etc. The hardware platform of a project is seldom entirely new. In fact, in most cases, 80 percent of the hardware components are re-used from previous projects or simply are COTS (Commercial Off-The-Shelf) components. There is no need to simulate in great detail these already proven components, whereas there is a need to run fast simulation of the software using these components.

These requirements call for an integrated, modular, simulation environment where already proven components can be simulated quickly, (even possibly including real hardware in the loop), new components under design can be tested more thoroughly, and the software can be tested on the complete platform with reasonable speed.

Modularity and fast prototyping also have become important aspects of simulation frameworks, for investigating alternative designs with easier re-use and integration of third party components.

The project aims at developing such a rapid prototyping simulation platform, combining new hardware components modeling, verification techniques, fast software simulation for proven components, capable of running the real embedded software application without any change.

Co-simulation has been a growing area of interest in the past decade, as hardware-software co-design is becoming a key industrial factor. Co-simulation usually implies two separate technologies, typically one using a Hardware Description Language, and another one using an Instruction Set Simulator (ISS) [27], [30], [44]. Some communication and synchronization must be designed and maintained between the two using some inter-process communication (IPC).

A commonly used solution is the combination of some ISS, interpreting or compiling, with an HDL simulator which can be implemented by software or by using an FPGA [35]. These solutions tend to present slow iteration design cycles, and become very costly when using large FPGA platforms. Others have implemented a co-simulation environments that reach fair integration, however there are still two main simulation loops interconnected, using adhoc mechanisms.

The idea pursued in SimSoC simulator is to combine hardware modeling and fast simulation into a fully integrated simulation environment, software based (not using FPGA) named SimSoC, using a single simulation loop, thanks to Transaction Level Modeling (TLM) [22], [18], combined with new ISS technology designed specifically to fit within the TLM environment.

The most challenging way to enhance simulation speed is to simulate the processors. Processor simulation is achieved with Instruction Set Simulation (ISS). In the past decade, dynamic translation technology has been favored many ISS [38], [25], [40], [41]. The binary target code to be executed is dynamically translated into an executable representation. There are typically two variants of dynamic translation technology: the target code is translated either directly into machine code for the simulation host, or into an intermediate representation that makes it possible to execute the code with fast speed. Dynamic translation introduces a compile time phase as part of the overall simulation time. But as the resulting cached code is re-used the compilation time is amortized over time.

Processor simulation is also achieved in Virtual Machines such as QEMU [20] and GXEMUL [29] that emulate to a large extent the behavior of a particular hardware platform. The technique used in QEMU is a form of dynamic translation. The target code is translated directly into machine code using some pre-determined code patterns that have been pre-compiled with the C compiler. QEMU and GXEMUL each include many device models of open-source C code; but this code is hard to reuse. The functions to emulate device accesses do not have the same profile. The scheduling process of the parallel hardware entities is not specified well enough to guarantee the compatibility between several emulators or third-party models.

There are several alternatives to achieve such simulation. In *interpretive simulation*, each instruction of the target program is fetched from memory, decoded, and executed. This method is flexible and easy to implement, but the simulation speed is slow as it wastes a lot of time in decoding. Interpretive simulation is used in Simplescalar [21]. Another technique to implement ISS is *dynamic translation* [23], [41], [25]. With dynamic translation, the target instructions are fetched from memory at run-time, like in interpretive simulation. They are decoded on the first execution and the simulator translates these instructions into another representation which is stored into a cache. On further execution of the same instructions, the translated cached version is used. If the code is modified during run-time, the simulator invalidates the cached representation. Dynamic translation provides much faster simulation while keeping the advantage of interpretive simulation as it supports the simulation of programs that have either dynamic loading or self-modifying code.

A challenge in the development of simulator is to maintain simultaneously fast speed and simulation accuracy. In the FORMES project, we expect to develop a dynamic translation technology with additional objectives:

- to take advantage of multi-processor simulation hosts to parallelize the simulation;
- to define intermediate representations of programs that optimize the simulation speed and provide a convenient format for making proofs about the simulated programs.

The SimSoC simulator is based on the TLM standard from OSCI [39]. The hardware components are modeled as TLM models, and since TLM is itself based on SystemC, the simulation is driven by the SystemC [32] kernel. We use standard, unmodified, SystemC (version 2.2), hence the simulator has a single simulation loop. The interconnection between components is an abstract bus similar to the TLM TAC abstract bus open sourced by ST Microelectronics [42]. Each processor simulated in the platform is abstracted as a particular TLM class. This class is both an initiator (it can initiate transactions) and a target (it can process transactions). It acts as an initiator to initiate I/Os and it behaves as a target essentially to receive the boot or halt signals and interrupt notifications from the interrupt controller. Memory and I/O controllers are also modeled as TLM classes. The simulated platform can include multiple heterogeneous processors, for example a general purpose CPU and a DSP. Then each processor is abstracted by a TLM class and they communicate among themselves and I/O controllers through TLM transactions. Research work has been done regarding TLM models such as [36], [45], [37].

3.2. Formal proofs

verification the theoretical and practical tools to verify that the simulation results are consistent with expected properties of the system.

proof assistant the software tools that allow the user to build certified formal proofs interactively in some given, expressive logical system.

kernel or proof-checker the part of a proof assistant that checks the correctness of a formal proof.

decision procedure A dedicated prover for a decidable fragment of logic.

Presburger arithmetic the most well-known decidable fragment of integer arithmetic.

tactic a software tool that helps the user in building a proof of some kind.

formalization a description of a problem in the logical system of the proof assistant.

development a set of formalizations and proofs achieving a particular goal.

library a set of developments relating to each-other.

Coq is one of the most popular proof assistant, in the academia and in the industry. Based on the Calculus of Inductive Constructions, Coq has three kinds of basic entities: objects are used for computations (data, programs, proofs are objects); types express properties of objects; kinds categorize types by their logical structure. Coq's type checker can decide whether a given object satisfies a given type, and if a given type has a logical structure expressed by a given kind. Because it is possible to (uniformly) define inductive types such as lists, dependent types such as lists-of-length-n, parametric types such as lists-of-something, inductive properties such as $(even\ n)$ for $n : nat$, etc, writing small specifications in Coq is an easy task. Writing proofs is a harder (non-automatable) task that must be done by the user with the help of tactics. Automating proofs when possible is a necessary step for dissemination of these techniques, as is scaling up. These are the problems we are interested in.

Modeling in Coq is not always as easy as argued: Coq identifies expressions up to computation. Identifying lists of zeros of length $m + n$ and $n + m$ is no problem if m and n are given integers, but does not work if m and n are unknowns, since $n + m = m + n$ is a valid theorem of arithmetic which cannot be proved by mere computation. It follows that the statement $reverse(l \mathbin{@} l') = reverse(l') \mathbin{@} reverse(l)$ is not typable, $@$ standing for appending two lists. This problem that seemingly innocent statements cannot be written in Coq because they do not type-check has been considered a major open problem for years. Blanqui, Jouannaud and Strub have recently developed *Coq modulo Theories*, in which computations do not operate only on closed terms (as are $1 + 2$ and $2 + 1$) but on open expressions of a decidable theory (as is $n + m = m + n$ in Presburger arithmetic). This preliminary work addresses three problems at once: decidable goals become solved automatically by a program taken from the shelves; writing specifications and proofs becomes easier and closer to the mathematical practice; assuming that calls to a decision procedure return a *proof certificate* in case of success, the correctness of a Coq proof now results from type checking the proof as well as the various certificates generated along the proof. Trusting Coq becomes incremental, resulting from trusting each certificate checker when added in turn to Coq's kernel. Developing this new paradigm is our first challenge.

Scaling up is yet another challenge. Modeling a large, complex software is a hard task which has been addressed within the Coq community in two different ways. By developing a module system for Coq in the OCaML style, which makes it possible to modularize proof developments and hence to develop modular libraries. By developing a methodology for modeling real programs and proving their properties with Coq. This methodology allows to translate a JavaCard (tool Caduceus) or C (tool Krakatoa) program into an ML-like program. The correctness of this first step is ensured by proving in Coq requirements generated along the translation. The correctness of the ML-like program annotated by the user is then done by Coq via another tool called Why. This methodology and the associated tools are developed by the INRIA project PROVAL. Our second challenge is to develop an analog of Caduceus for the abstract representation to be used by our dynamic translation technology. The challenge here lies in the fact that abstract representations generated by simulation are inherently low level.

4. Application Domains

4.1. Application Domains

Keywords: *embedded systems*, *process engineering*, *telecommunications*, *transportation*.

Simulation is relevant to most areas where complex embedded systems are used, not only to the semiconductor industry for System-on-Chip modeling, but also to any application where a complex hardware platform must be assembled to run the application software. It has applications for example in industry automation, digital TV, telecommunications and transportation.

5. Software

5.1. SimSoC

Keywords: *Co-design*, *Co-simulation*, *Dynamic Translation*, *Simulation*, *SystemC*, *TLM*.

Participants: Vania Joloboff [correspondant], Claude Helmstetter, Jiajia Song.

The simulation software made by the FORMES Team is called SimSoC. It is based on SystemC kernel and uses Transaction Level Modeling for interactions between the hardware models. The software includes:

- Instruction Set Simulators. The ARM Version 5 has been implemented. Other architectures are under development.
- A dynamic translator from binary programs to an internal representation. For the ARM architecture a compiler has been developed that generates the C++ translated code, using parameterized specialization options.
- Some peripheral models such as serial line controller, flash memory controller, interrupt controller.
- Utilities software such as utility to generate permanent storage for flash memory simulation, or a compiler tool to generate instruction binary decoder.

It is intended that the software will be distributed under open source license. Please contact correspondant contact if you are interested in this software.

See http://liama.ia.ac.cn/wiki/projects:formes:simulator.

5.2. CoLoR and Rainbow

Keywords: *Coq*, *certification*, *proof*, *rewriting*, *termination*.

Participant: Frédéric Blanqui [correspondant].

CoLoR and Rainbow are mainly developed by Adam Koprowski (Eindhoven University of Technology until 30/11/08, and Radbound University, Nijmegen, The Netherlands now) and Frédéric Blanqui, and various European researchers and master and PhD students made important contributions: Sébastien Hinderer (LORIA, France), Solange Coupet-Grimal and William Delobel (Université de Provence Aix-Marseille I, France), Stéphane Le Roux (ENS Lyon, France), Léo Ducas (ENS Paris, France), and Johannes Waldmann (Leipzig HTWK, Germany).

It is a Coq [24] library on rewriting and termination. It is intended to serve as a basis for certifying the output of automated termination provers like TPA, AProVE, Torpa, etc. It contains libraries on:

- Mathematical structures: relations, semi-rings.
- Data structures: lists, vectors, integer polynomials with multiple variables, finite multisets, matrices.
- Term structures: strings, algebraic terms with symbols of fixed arity, algebraic terms with varyadic symbols, simply typed lambda-terms.
- Transformation techniques: conversion from strings to algebraic terms, conversion from algebraic to varyadic terms, arguments filtering, rule elimination, dependency pairs.
- Termination criteria: polynomial interpretations, multiset ordering, lexicographic ordering, first and higher order recursive path ordering, matrix interpretations, dependency graph decomposition.

Rainbow is a tool for automatically certifying termination proofs expressed in a standardized format called termination proof grammar (TPG). Termination proofs are translated and checked in Coq by using the CoLoR library.

CoLoR and Rainbow are distributed under CeCILL license on http://color.loria.fr/.

5.3. Moca

Keywords: *Non-free data types*, *completion*, *functional programming*, *rewriting*.

Participant: Frédéric Blanqui [correspondant].

Moca is mainly developed by Pierre Weis (INRIA Rocquencourt) and Frédéric Blanqui, and benefited from a postdoc (Richard Bonichon, 09/07-08/08) and an Argentin master student (Laura Lowenthal, 10/07-02/08).

It is a general construction functions generator for OCaML [26] data types with invariants.

Moca allows the high-level definition and automatic management of complex invariants for data types. In addition, Moca provides the automatic generation of maximally shared values, independently or in conjunction with the declared invariants.

A relational data type is a concrete data type that declares invariants or relations that are verified by its constructors. For each relational data type definition, Moca compiles a set of construction functions that implements the declared relations.

Moca supports two kinds of relations:

- algebraic relations (such as associativity or commutativity of a binary constructor),
- general rewrite rules that map some pattern of constructors and variables to some arbitrary user's define expression.

Algebraic relations are primitive, so that Moca ensures the correctness of their treatment. By contrast, the general rewrite rules are under the programmer's responsibility, so that the desired properties must be verified by a programmer's proof before compilation (including for completeness, termination, and confluence of the resulting term rewriting system).

Algebraic invariants are specified by using keywords denoting equational theories like commutativity and associativity. Moca generates construction functions that allow each equivalence class to be uniquely represented by their canonical value.

Moca is distributed under QPL on http://moca.inria.fr/.

6. New Results

6.1. Coq modulo theories

Participants: Frédéric Blanqui, Jean-Pierre Jouannaud, Pierre-Yves Strub.

The main achievement of the team is the Calculus of Presburger Constructions [11], which allows to have an extensional equality for Presburger arithmetic instead of an intensional equality as is the rule since the very early days of Martin-Löf's type theory who first recognized that type checking becomes undecidable in presence of an extensional equality at all types.

After the defense of his PhD [8], Pierre-Yves Strub started to develop a formal proof in Coq for his decidability result of type checking in the Calculus of Presburger Inductive Constructions. Rather than an academic project, this work should be seen as a preliminary step towards more effective algorithms in the future implementation of the calculus. This preliminary work is scheduled to be completed by the end of the year. The new implementation of Coq should start right after this phase.

6.2. Confluence of first and higher-order rewriting

Participant: Jean-Pierre Jouannaud.

In [9], we settled the problem of modularity of confluence for first-order rewriting systems, by allowing for arbitrary rewriting modulo equations, and for extra variables in righthand sides.

With Femke van Ramsdong and Vincent van Oostrom, we continue our work on confluence of normal rewriting, with application, in particular, to the confluence of higher-order rewriting.

6.3. Termination of higher-order rewriting

Participants: Frédéric Blanqui, Jean-Pierre Jouannaud.

In [10], we briefly survey automated termination proof methods for higher-order calculi. We then concentrate on the higher-order recursive path ordering, for which we provide an improved definition, the Computability Path Ordering. This new definition appears indeed to capture the essence of computability arguments à la Tait and Girard, therefore explaining the name of the improved ordering.

In [16], we extend the static dependency pair method for simply-typed term rewriting systems (STRSs) [43] to higher-order rewrite systems (HRSs), using the notion of strong computability. Since HRSs include lambda-abstraction, but STRSs do not, we restructure the static dependency pair method to correspond to lambda-abstraction, and show that the static dependency pair method also works well on HRSs without new restrictions.

6.4. Certification of termination certificates

Participant: Frédéric Blanqui.

The CoLoR library and the Rainbow program have been further developed. Adam Koprowski and Hans Zantema have added support for matrix interpretations [34], and Adam Koprowski and Johannes Waldmann have added support for arctic matrix interpretations [33]. Frédéric Blanqui added support for dependency graph decomposition and unification [19]. Frédéric Blanqui and Adam Koprowski have written some manuscript describing the general approach implemented in the CoLoR library and the Rainbow program for automatically certifying termination certificates [17]. Rainbow is now used as a certification back-end by 4 automated termination provers: AProVE[1] (best prover for TRSs), Matchbox[2] (best prover for SRSs), TPA[3] and TTT2[4]. And Rainbow was again the best certification back-end in the 2008 international contest on certified termination provers. See http://color.loria.fr/comp.html.

6.5. SimSoC software

Participants: Vania Joloboff, Claude Helmstetter, Jiajia Song.

In order to compare different techniques, we have implemented three kinds of instruction simulation corresponding to three modes that the simulator can run in, for the ARM architecture.

The first mode, named D0, is interpretive simulation. This is the basis from which we can compare performance. The second mode (D1) is dynamic translation with no specialization. This mode shows the performance improvement obtained with dynamic translation compared to interpretive simulation. The third mode (D2) is dynamic translation using *partial evaluation*, a compiling optimization technique, also known as specialization [28]. The basic concept of specialization is to transform a generic program P, when operating on some data d into a faster specialized program Pd that executes specifically for this data. Specialization can be advantageously used in processor simulation [38], because data can often be computed at decoding time, and a specialized version of the generic instruction can be used to execute it. The simulation code then uses fewer tests, fewer memory accesses and more immediate instructions.

In both D1 and D2 mode of our simulator, the dynamic translation process constructs an intermediate representation (IR) for the original instructions that is suitable for execution on the simulation machine.

The SimSoC simulator is described in two publications [14], [13].

6.6. Model Scheduling

Participants: Vania Joloboff, Claude Helmstetter, Jiajia Song.

TLM has a thread-model that is in fact a co-routine model. The main parallel entities of hardware (processors, DMA, bus arbiter, ...) are modeled in TLM by asynchronous processes, which have to be scheduled at simulation time. All TLM models are being scheduled sequentially. The behavior of the models become different, possibly incoherent, if the threads are truly run in parallel.

[1] http://aprove.informatik.rwth-aachen.de/
[2] http://dfa.imn.htwk-leipzig.de/matchbox/
[3] http://www.win.tue.nl/tpa/
[4] http://colo6-c703.uibk.ac.at/ttt2/

This induces a problem for validation by simulations: we have to cover the set of valid schedulings in addition to the set of data. Indeed, a deterministic scheduler will miss some bugs. Random schedulings will show more possible behaviors but the coverage is still uncertain. The valid schedulings of a real model are too numerous to try them all.

We worked with the validation of System-on-a-Chip models at the transaction level (TLM) [31]. These models are used for the development of embedded software. Hardware is highly parallel but the simulator runs on a single processor. The specification of this scheduling is non-deterministic in order to represent the physical parallelism faithfully.

In [4], we presented a solution to cover effectively the set of schedulings. Our solution is based on dynamic partial order reduction. The idea is to look at the actions performed by the processes, in order to guess whether a change in their order (as what would be produced by distinct scheduler choices) could affect the final state. Successive iterations eventually give a complete scheduling set, which guarantees the detection of all local errors and deadlocks for a fixed data set.

Finally, in [12], [15], we also studied SystemC and TLM semantics with respect to other formalisms.

7. Contracts and Grants with Industry

7.1. Schneider Electric

Participants: Vania Joloboff, Claude Helmstetter, Jiajia Song.

The simulation part of the project is sponsored by Schneider Electric China.

8. Other Grants and Activities

8.1. International Initiatives

8.1.1. Action in China

The FORMES project is a project run in cooperation with the University of Tsinghua which hosts the project, Beijing University of Aeronautics and Astronautics and the University of Science and Technology at Beijing.

8.2. National Initiatives

8.2.1. ARC Quotient (2007-2008)

Participant: Frédéric Blanqui [leader].

This project gathers people from INRIA Nancy - Grand Est (Frédéric Blanqui), INRIA Paris - Rocquencourt (Pierre Weis and Damien Doligez), Université Paris 6 (Thérèse Hardin, Renaud Rioboo) and CNAM (David Delahaye, Catherine Dubois). Its aim is to study and certify the use of non-free concrete data types in functional programming and develop an extension of OCaML providing such types. It benefited from a postdoc (Richard Bonichon, 09/07-08/08) and an Argentin master student (Laura Lowenthal, 10/07-02/08).

8.2.2. ANR SIVES (2009-2011)

Participants: Frédéric Blanqui [leader], Vania Joloboff, Jean-Pierre Jouannaud, Pierre-Yves Strub.

SIVES is a recently awarded project, with INRIA as leader on the French side, Tsinghua University on the Chinese side, and with the participation of Beihang university. SIVES is funded by ANR and the National Science Foundation of China.

8.2.3. Research networks (GDR)

Participants: Frédéric Blanqui, Jean-Pierre Jouannaud, Pierre-Yves Strub.

- GPL network on software engineering, working group LTP on Languages, Types and Proofs.
- IM network on mathematics and computer science, working group LAC on Logic, Algebra and Calculus.

8.3. Exterior research visitors

- Jean-François Monin, Verimag.
- Yijia Chen, Associate professor, Jiao Tong University, Shanghai.

9. Dissemination

9.1. Presentations

- Jean-Pierre Jouannaud gave a presentation at Jiao Tong University (december) and Fudan University (december), Shanghai, at National Taiwan University (november), Taipei (ROC). He was invited speaker at the Computer Science Logic Conference in Bertinoro (september, Italy), and at the Colloquium in Honor of Hubert Comon at ENS-Cachan (november, France).
- Frédéric Blanqui gave a presentation at the East China Normal University (december), Shanghai.
- Vania Joloboff gave presentations at Zhejiang University, Xian Northwest Polytechnic University, Chengdu UESTC University and was invited speaker at the Embedded Forum 2008 in Shanghai.

9.2. Teaching

- Vania Joloboff contributed to set up the ARTIST2 Summer School in China. See http://www.artist-embedded.org/artist/Organisation,1355.html.
- Frédéric Blanqui and Jean-Pierre Jouannaud gave a two weeks course on formal methods for the master students of Tsinghua in March. Four students follow now an intensive training (8 hours a week) combining foundations of formal methods with Coq programming. See http://liama.ia.ac.cn/wiki/projects:formes:class.
- We also started putting up a Coq summer school targeting South-East-Asian students to be held at Tsinghua the last week of August 2009.

10. Bibliography

Major publications by the team in recent years

[1] F. BLANQUI. *Definitions by rewriting in the Calculus of Constructions*, in "Mathematical Structures in Computer Science", vol. 15, n^{o} 1, 2005, p. 37-92.

[2] F. BLANQUI. *Inductive types in the Calculus of Algebraic Constructions*, in "Fundamenta Informaticae", vol. 65, n^{o} 1-2, 2005, p. 61-86.

[3] B. DELSART, V. JOLOBOFF, E. PAIRE. *JCOD: A Lightweight Modular Compilation Technology for Embedded Java*, in "Second International Conference on Embedded Software", Lecture Notes in Computer Science, ISBN 3-540-44307-X, vol. 2491, Springer-Verlag, 2002, p. 197–212.

[4] C. HELMSTETTER, F. MARANINCHI, L. MAILLET-CONTOZ. *Test Coverage for Loose Timing Annotations*, in "11th International Workshop on Formal Methods for Industrial Critical Systems", Springer-Verlag, August 2006.

[5] J.-P. JOUANNAUD, A. RUBIO. *Higher-Order Orderings for Normal Rewriting*, in "Rewriting Techniques and Applications", Lectures Notes in Computer Science, vol. 4098, Springer-Verlag, 2006, p. 387–399.

[6] J.-P. JOUANNAUD, A. RUBIO. *Polymorphic Higher-Order Recursive Path Orderings*, in "Journal of the ACM", vol. 54, n^{o} 1, 2007, p. 1-48.

[7] J.-P. JOUANNAUD, F. VAN RAAMSDONG. *Confluence Properties of Terminating Higher-Order Rewrite Relations*, in "Mathematical Theories of Abstraction, substitution and naming in Computer Science", ICMS, mai 2007.

Year Publications

Doctoral Dissertations and Habilitation Theses

[8] P.-Y. STRUB. *Théorie des Types et Procédures de Décision*, Ph. D. Thesis, Ecole Polytechnique X, 07 2008, http://tel.archives-ouvertes.fr/tel-00351837/en/.

Articles in International Peer-Reviewed Journal

[9] J.-P. JOUANNAUD, Y. TOYAMA. *Modular Church-Rosser Modulo: the Full Picture*, in "International Journal of Software and Informatics", 2008, http://hal.inria.fr/inria-00350970/en/.

Invited Conferences

[10] F. BLANQUI, J.-P. JOUANNAUD, A. RUBIO. *The computability path ordering: the end of a quest*, in "7th EACSL Annual Conference on Computer Science Logic - CSL'08 LNCS, Italie Bertinoro", LNCS, vol. 5213, 2008, http://hal.inria.fr/inria-00288209/en/.

International Peer-Reviewed Conference/Proceedings

[11] F. BLANQUI, J.-P. JOUANNAUD, P.-Y. STRUB. *From formal proofs to mathematical proofs: a safe, incremental way for building in first-order decision procedures*, in "5th IFIP International Conference on Theoretical Computer Science - TCS 2008 IFIP, Italie Milan", IFIP, vol. 273, 2008, http://hal.inria.fr/inria-00275382/en/.

[12] C. HELMSTETTER, O. PONSINI. *A Comparison of Two SystemC/TLM Semantics for Formal Verification*, in "Formal Methods and Models for Codesign (MEMOCODE), États-Unis d'Amérique Anaheim", 2008, http://hal.inria.fr/inria-00275456/en/.

[13] V. JOLOBOFF, C. HELMSTETTER, H. HAO, J. SONG. *Generation of Executable Representation for Processor Simulation with Dynamic Translation*, in "2008 International Conference on Computer Science and Software Engineering, Chine Wuhan", IEEE Computer Society, 2008, http://hal.inria.fr/inria-00350519/en/.

[14] V. JOLOBOFF, C. HELMSTETTER. *SimSoC: A SystemC TLM integrated ISS for full system simulation*, in "Asia Pacific Conference on Computer Architecture and Systems, Macao Macao", 2008, http://hal.inria.fr/inria-00349846/en/.

[15] F. MARANINCHI, M. MOY, J. CORNET, L. MAILLET-CONTOZ, C. HELMSTETTER, C. TRAULSEN. *SystemC/TLM Semantics for Heterogeneous System-on-Chip Validation*, in "2008 Joint IEEE-NEWCAS and TAISA Conference, Canada Montréal", IEEE (editor), June 2008, unknown, http://hal.archives-ouvertes.fr/hal-00311011/en/.

Other Publications

[16] F. BLANQUI, Y. ISOGAI, K. KUSAKARI, M. SAKAI. *Static dependency pair method based on strong computability for higher-order rewrite systems*, 2008, http://www.loria.fr/~blanqui/papers/hodp.pdf.

[17] F. BLANQUI, A. KOPROWSKI. *Automated verification of termination certificates*, 2008.

References in notes

[18] F. GHENASSIA (editor). *Transaction-Level Modeling with SystemC. TLM Concepts and Applications for Embedded Systems*, ISBN 0-387-26232-6, Springer, June 2005.

[19] T. ARTS, J. GIESL. *Termination of Term Rewriting Using Dependency Pairs*, in "Theoretical Computer Science", vol. 236, 2000, p. 133-178.

[20] F. BELLARD. *QEMU, A Fast And Portable Dynamic Translator*, in "USENIX Annual Technical Conference, Philadelphia, PA, USA", 2005.

[21] D. BURGER, T. M. AUSTIN. *The SimpleScalar tool set, version 2.0*, in "SIGARCH Comput. Archit. News", vol. 25, n^{o} 3, 1997, p. 13–25, http://doi.acm.org/10.1145/268806.268810.

[22] L. CAI, D. GAJSKI. *Transaction level modeling: an overview*, in "CODES+ISSS '03: Proceedings of the 1st IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis, New York, NY, USA", ACM Press, 2003, p. 19–24, http://doi.acm.org/10.1145/944645.944651.

[23] B. CMELIK, D. KEPPEL. *Shade: a fast instruction-set simulator for execution profiling*, in "SIGMETRICS Perform. Eval. Rev.", vol. 22, n^{o} 1, 1994, p. 128–137, http://doi.acm.org/10.1145/183019.183032.

[24] COQ DEVELOPMENT TEAM. *The Coq Proof Assistant Reference Manual, Version 8.1*, INRIA Rocquencourt, France, 2006, http://coq.inria.fr/.

[25] J. D'ERRICO, W. QIN. *Constructing portable compiled instruction-set simulators: an ADL-driven approach*, in "DATE '06: Proceedings of the conference on Design, automation and test in Europe, 3001 Leuven, Belgium, Belgium", European Design and Automation Association, 2006, p. 112–117.

[26] D. DOLIGEZ, J. GARRIGUE, X. LEROY, D. RÉMY, J. VOUILLON. *The Objective Caml system release 3.10, Documentation and user's manual*, INRIA, France, 2007, http://caml.inria.fr/.

[27] F. FUMMI, G. PERBELLINI, M. LOGHI, M. PONCINO. *ISS-centric modular HW/SW co-simulation.*, in "ACM Great Lakes Symposium on VLSI", 2006, p. 31-36.

[28] Y. FUTURAMA. *Partial Evaluation of Computation Process-An Approach to a Compiler-compiler*, in "Systems, Computers, Controls 2 (5)", 1971, p. 45–50.

[29] A. GAVARE. *GXemul Documentation*, 2007, http://gavare.se/gxemul/gxemul-stable/doc/index.html.

[30] P. GERIN, S. YOO, G. NICOLESCU, A. A. JERRAYA. *Scalable and flexible cosimulation of SoC designs with heterogeneous multi-processor target architectures*, in "ASP-DAC '01: Asia South Pacific Design Automation Conference", ACM, 2001, p. 63–68.

[31] C. HELMSTETTER, F. MARANINCHI, L. MAILLET-CONTOZ, M. MOY. *Automatic Generation of Schedulings for Improving the Test Coverage of Systems-on-a-Chip*, in "FMCAD", vol. 0, 2006, p. 171-178, http://doi.ieeecomputersociety.org/10.1109/FMCAD.2006.10.

[32] IEEE. *IEEE Standard 1666 - SystemC Language Reference Manual*, Technical report, IEEE, 2006.

[33] A. KOPROWSKI, J. WALDMANN. *Arctic Termination...Below Zero*, in "Proceedings of the 19th International Conference on Rewriting Techniques and Applications, Lecture Notes in Computer Science 5117", 2008.

[34] A. KOPROWSKI, H. ZANTEMA. *Certification of Proving Termination of Term Rewriting by Matrix Interpretations*, in "Proceedings of the 34th International Conference on Current Trends in Theory and Practice of Computer Science, Lecture Notes in Computer Science 4910", 2008.

[35] M. MEERWEIN, C. BAUMGARTNER, T. WIEJA, W. GLAUERT. *Embedded systems verification with FGPA-enhanced in-circuit emulator*, in "ISSS '00: Proceedings of the 13th international symposium on System synthesis, Washington, DC, USA", IEEE Computer Society, 2000, p. 143–148, http://doi.acm.org/10.1145/501790.501821.

[36] M. MOY, F. MARANINCHI, L. MAILLET-CONTOZ. *LusSy: A Toolbox for the Analysis of Systems-on-a-Chip at the Transactional Level*, in "International Conference on Application of Concurrency to System Design", June 2005.

[37] M. MOY, F. MARANINCHI, L. MAILLET-CONTOZ. *Pinapa: An Extraction Tool for SystemC descriptions of Systems-on-a-Chip*, in "EMSOFT", September 2005, http://www-verimag.imag.fr/~moy/publications/sc-compil.pdf.

[38] A. NOHL, G. BRAUN, O. SCHLIEBUSCH, R. LEUPERS, H. MEYR, A. HOFFMANN. *A universal technique for fast and flexible instruction-set architecture simulation*, in "DAC '02: Proceedings of the 39th conference on Design automation, New York, NY, USA", ACM, 2002, p. 22–27, http://doi.acm.org/10.1145/513918.513927.

[39] OPEN SYSTEMC INITIATIVE (OSCI). *SystemC TLM Library*, 2007, http://www.systemc.org.

[40] M. PONCINO, J. ZHU. *DynamoSim: a trace-based dynamically compiled instruction set simulator*, in "ICCAD '04: Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design, Washington, DC, USA", IEEE Computer Society, 2004, p. 131–136, http://dx.doi.org/10.1109/ICCAD.2004.1382557.

[41] M. RESHADI, P. MISHRA, N. DUTT. *Instruction set compiled simulation: a technique for fast and flexible instruction set simulation*, in "DAC '03: Proceedings of the 40th conference on Design automation, New York, NY, USA", ACM, 2003, p. 758–763, http://doi.acm.org/10.1145/775832.776026.

[42] ST MICROELECTRONICS. *TAC: Transaction Accurate Communication/Channel*, 2006, http://www.greensocs.com/TACPackage.

[43] M. SAKAI, K. KUSAKARI. *On Dependency Pair Method for Proving Termination of Higher-Order Rewrite Systems*, in "IEICE Transactions on Information and Systems", vol. E88-D, n^{o} 3, 2005, p. 583-593.

[44] P. SCHAUMONT, D. CHING, I. VERBAUWHEDE. *An interactive codesign environment for domain-specific coprocessors*, in "ACM Trans. Des. Autom. Electron. Syst.", vol. 11, n^{o} 1, 2006, p. 70–87, http://doi.acm.org/10.1145/1124713.1124719.

[45] E. VIAUD, F. PÊCHEUX, A. GREINER. *An efficient TLM/T modeling and simulation environment based on conservative parallel discrete event principles*, in "DATE '06: Proceedings of the conference on Design, automation and test in Europe, 3001 Leuven, Belgium, Belgium", European Design and Automation Association, 2006, p. 94–99.