INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team FORMES

FOrmal Methods for Embedded Systems

Paris - Rocquencourt

Theme : Programs, Verification and Proofs

Activity Report

2009

Table of contents

1. Team

Research Scientist

Frédéric Blanqui [CR1 INRIA]
Vania Joloboff [DR INRIA]
Jean-Pierre Jouannaud [DR INRIA and Tsinghua Software Chair, Team Leader, HdR]
Jean-François Monin [DR CNRS on leave from the University Joseph Fourier since September 1st]

Faculty Member

Fei He [Tsinghua assistant professor]
Ming Gu [Tsinghua professor]

Technical Staff

Hui Xiao [Engineer]

PhD Student

Xiaomu Shi [University Joseph Fourier since November 1st]
Qian Wang [Tsinghua University since September 1st]
Liangze Yin [Tsinghua University]
Lianyi Zhang [Tsinghua University since September 1st]
Min Zhou [Tsinghua University]

Post-Doctoral Fellow

Claude Helmstetter [INRIA since November 17]
Jianqi Li [Tsinghua University]
Pierre-Yves Strub [INRIA]

Visiting Scientist

Bow-Yaw Wang [INRIA visiting professeur and Tsinghua invited professor since September 1st]

Administrative Assistant

Lin Cui [Tsinghua University, part time]
Lin Tang [CASIA, part time]

Other

Bin Liu [Master Beihang University]
Ming Liu [Master Guangxi University of Technology]
Huiying Luo [Master Tsinghua University Beijing]
Yuning Pang [Master Beihang University]
Lijun Sun [Master Northwestern Polytechnical University]
Bing Zhou [Master Guangxi University of Technology]
He Zhu [Master Tsinghua University]

2. Overall Objectives

2.1. Overall Objectives

Formes[1] is a joint project of CNRS, INRIA and Tsinghua University[2], located on Tsinghua's campus in Beijing. FORMES is therefore one of the projects of the LIAMA consortium[3]. Formes was created in 2008, and includes the activities developped since 2007 at LIAMA by Vania Joloboff (project DeviceWare).

FORMES stands for FORmal Methods for Embedded Systems. FORMES 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.

[1] http://formes.asia
[2] http://www.tsinghua.edu.cn/eng/index.jsp
[3] http://liama.ia.ac.cn/wiki/

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, and to formal proofs to verify the completeness of test sets. We finally plan to rely on formal proofs again for verifying the correct functionning of our tools. Proving properties of these various abstractions requires using a certified, interactive theorem prover.

2.2. Highlights of the year

- CoqMT, a new certified extension of Coq developped in the team, is now available at http://pierre-yves.strub.nu/coqmt/. This is the successful conclusion of a long term effort to improve the treatment of equality in Coq. Preliminary experiments show that CoqMT remedies as expected to the weaknesses of Coq in presence of dependent types.
- SimSoC, our simulator for embedded systems, will soon be released, see http://formes.asia/cms/softwares/simsoc. Its performance already approaches the performance of the simulated system in some cases.

- Rainbow, developped in the team, was the best certification back-end used by the termination tools which participated to the 2007 and 2008 editions of the international competition of certified termination provers[4].

3. Scientific Foundations

3.1. Simulation

The development of complex embedded systems platforms requires putting together many hardware components, processor cores, application specific co-processors, bus architectures, 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, (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, modular 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.

To fully simulate a complete hardware platform, one must simulate the processors, the co-processors, together with the peripherals such as network controllers, graphics controllers, USB controllers, etc. A commonly used solution is the combination of some ISS (Instruction Set Simulator) connected to a Hardware Description Language (HDL) simulator which can be implemented by software or by using a FPGA [42] simulator. These solutions tend to present slow iteration design cycles, (implementing the FPGA means the hardware has already been designed at low level) and become very costly when using large FPGA platforms. Others have implemented a co-simulation environment, using two separate technologies, typically one using a HDL and another one using an ISS [33], [35], [50]. Some communication and synchronization must be designed and maintained between the two using some inter-process communication (IPC), which slows down the process.

The idea we pursue is to combine hardware modeling and fast simulation into a fully integrated, software based (not using FPGA) simulation environment named SimSoC, which uses a single simulation loop thanks to Transaction Level Modeling (TLM) [29], [22] combined with a 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). 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 [28]. Another technique to implement a fast ISS is *dynamic translation* [30], [49], [32]. which has been favored by many [46], [32], [48], [49] in the past decade. With dynamic translation, the binary 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.

[4] http://termination-portal.org/wiki/Termination_Competition

There are typically two variants of the 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 [24] and GXEMUL [34] 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. Both QEMU and GXEMUL include many device models of open-source C code, but this code is hard to reuse. The functions that 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 re-usability of third-party models using the standards from the electronics industry (e.g. IEEE 1666)

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

- provide different levels of translation with different degrees of accuracy so that users can choose between accurate and slow (for debugging) or less accurate but fast simulation.
- to take advantage of multi-processor simulation hosts to parallelize the simulation;
- to define intermediate representations of programs that optimize the simulation speed and possibly provide a more convenient format for studying properties of the simulated programs.

The SimSoC simulator is based on the TLM standard from OSCI [47]. The hardware components are modeled as TLM models, and since TLM is itself based on SystemC, the simulation is driven by the SystemC [38] 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 [43]. 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 via TLM transactions. Research work has been done regarding TLM models such as [44], [52], [45].

3.2. Formal proofs

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 some natural number n, 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 two lists of identical content but respective lengths $m + n$ 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 $reverse(l :: l') = reverse(l') :: reverse(l)$

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. The development of this new paradigm is our first research challenge here.

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 FRAMA-C) program into an ML-like program. The correctness of this first step is ensured by proving in Coq verification conditions 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 in association with CEA. Part of our second challenge is to reuse these tools to prove properties at the source code level of programs used in an embedded application. As part of this effort, we are interested in the development of termination tools and automatic provers, in particular an SMT prover which is indeed complementary of our first challenge. The second part of the challenge is to ensure that these properties are still satisfied by the machine code executed on the embedded cpu. Here, we are going to rely on a different technology, certified compilers, and reuse the certified compilers from CLight to ARM or PowerPC developped in the COMPCERT INRIA project. We will be left with the development of certified compilers from source languages which are frequently used for developping embedded applications into CLight. These languages are either variants of C, or languages for the description of automata with timers in the case of Program Logic Controllers.

Our last challenge is to rely on certified tools only. In particular, we decided to certify in Coq all extensions of Coq developped in the project: the core logic of CoqMT has been certified with Coq. The most critical parts of the simulator will also be certified. As for compilers, there are two ways to certify tools: either, the code is proved correct, or it outputs a certificate that can be checked. The second approach demands less man-power, and has the other advantage to be compatible with the use of tools taken from the shelves, provided these tools are open-source since they must be equipped with a mechanism for generating certificates. This is the approach we will favour for the theories to be used in CoqMT, as well as for the SMT prover to be developped. For the simulator SimSoC itself, we shall probably combine both approaches.

4. Application Domains

4.1. Application Domains

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

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

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 a serial line controller, a flash memory controller, an interrupt controller.
- Utilities software such as a 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. See http://formes.asia/cms/softwares/simsoc for more information.

5.2. CoqMT

Participant: Pierre-Yves Strub [correspondant].

CoqMT is a modification of the Coq proof assistant allowing to dynamically load decision procedures for first-order theories in the conversion checker of the Coq kernel. Users decide which Coq symbols are handled by the decision procedures through the use of mapping primitives. Having dynamic loading and mapping facilities allows users to write their own decision procedures or take anyone from the shelves and use them in Coq without any additional modification of the Coq source code.

For the moment, CoqMT comes with a predefined decision procedure for integer linear arithmetic which generates small certificates (unlike previously existing procedures).

CoqMT (along with the decision procedure for integer linear arithmetic and the theory of dependent lists) is accessible via its GIT repository at http://pierre-yves.strub.nu/coqmt/.

5.3. aCiNO

Participants: Fei He [correspondant], Min Zhou.

aCiNO is a C++ implementation of Nelson-Oppen's architecture. It is intended to be a new and efficient SMT (Satisfiability Modulo Theory) solver. SMT is the problem of determining the satisfiability of a first-order logic formula in one or more decidable theories. It has been considered as the next generation of verification engines. We are going to develop this solver in an incremental way. We first aim at 2 popular theories, LRA (Linear Arithmetic) over real numbers and UF (Uninterpreted Functions). We use the simplex method to solve LRA and the congruence closure algorithm to solve UF. Both theories are combined under the Nelson-Oppen architecture. Newly discovered equalities between variables are therefore propagated to the other theory, so that the SMT solver is both SOUND and COMPLETE. We will integrate more theories into our solver on a by need basis.

5.4. CoLoR and Rainbow

Participants: Frédéric Blanqui [correspondant], Pierre-Yves Strub, Qian Wang, Lianyi Zhang.

CoLoR is a Coq [31] library on rewriting theory and termination [17]. It is intended to serve as a basis for certifying the output of automated termination provers like AProVE, MatchBox, TTT2, Torpa, TPA, etc. It contains libraries on:

- Mathematical structures: relations, (ordered) 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, dependency graph decomposition.
- Termination criteria: polynomial interpretations, multiset ordering, lexicographic ordering, first and higher order recursive path ordering, matrix interpretations.

Rainbow is a tool for automatically certifying termination proofs expressed in a given XML format. Termination proofs are translated and checked in Coq by using the CoLoR library.

Rainbow was the best certification back-end in the 2007 and 2008 editions of the international competition of certified termination provers[5].

Since then, we improved the efficiency of proof checking and extended Rainbow and CoLoR with syntactic first-order matching, the verification of loops in term and string rewrite systems (to certify non-termination), and semantic labelling [53].

We also started to formalize the termination of Haskell programs [37] (internship of Julien Bureaux, ENS Paris, from June 1st to July 31), and to formalize Rainbow itself in order to certify it and improve the efficiency of proof checking by using the extraction mechanism of Coq to OCaml.

CoLoR and Rainbow are distributed under the CeCILL license on http://color.inria.fr/. Various European people participated to its development (see the website for more information).

5.5. Moca

Participant: Frédéric Blanqui [correspondant].

Moca is developed by Pierre Weis (INRIA Rocquencourt) and Frédéric Blanqui.

It is a general construction functions generator for OCaML [40] 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).

[5] http://termination-portal.org/wiki/Termination_Competition

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. Simulation

6.1.1. SimSoC Full System Simulation

Participants: Vania Joloboff, Helmstetter Claude, Hui Xiao.

The SimSoC simulator software has been developed. The ARM simulator has been completed to include the MMU and simulate ARM 9 subsystem, including a simulation model for the PrimeCell interrupt controller.

A UART controller and a flash memory controller simulation model have been also been implemented simulating some flash memory models from ST Microelectronics. Gathering the simulation models for processor, interrupt controller and other peripherals, a full system simulator has been developed for a specific System On Chip from ST Microelectronics. Embedded Linux as distributed on ST web site for this SoC can now be run over that simulator [14]. In this simulator, the simulated UART is connected to a window of the Graphical User Interface so that users can login on Linux using the simulated serial line.

6.1.2. SimSoC network simulation and debug

Participants: Vania Joloboff, Hui Xiao, Seng Patrice, Combier Pascal, Yuning Pang.

An Ethernet controller simulator has been developed, allowing for the connection of two simulated systems running TCP/IP stack. A small network simulation layer was developed so that N simulated systems using the simulated Ethernet controller can connect together as if they were connected to the real network. It is therefore possible to have several simulated systems running on the same host machine or on multiple networked machines to communicate with Ethernet frames over a simulated network. In addition, this simulated network can be connected to the real world, for example to ping 'inria.fr' from a simulated system.

A framework was added to the simulator such that simulated programs can be debugged from any debugger compliant with the GDB protocol for remote debugging. This framework is mostly architecture independent, with only architecture dependent plug-ins. A complete implementation was developed for debugging ARM platforms.

6.1.3. PowerPC and MIPS simulation with debug

Participants: Vania Joloboff, Bin Liu, Ming Liu, Hui Xiao, Bing Zhou.

An experiment has started to explore parallell simulation in the case of multi-core System-On-Chips. The idea explored is to parallelize simulation of processors while maintaining a serialized simulation of devices in SystemC.

The simulation of PowerPC and MIPS architectures has started. A complete ISS in interpreted mode has been developed for PowerPC, including the Memory Management Unit (MMU). A subset of the dual core Freescale 82641D SoC has been simulated. This simulator can can run U-Boot and Embedded Linux with limited pheripheral devices.

6.1.4. ARMv6 instruction set formalization

Participants: Frédéric Blanqui, Jean-François Monin, Xiaomu Shi.

We recently started to work on the certification of our simulator SimSoC by formalizing in Coq the ARMv6 instruction set, its binary encoding/decoding, and its semantics, reusing Xavier Leroy's work on logical and arithmetic operations on 32-bits words for CompCert[6] [41]. We cannot reuse CompCert's simplified formalization of the ARM instruction set and semantics as it is, since we want to be able to simulate every detail of an ARM processor, including exceptions and other imperative mechanisms.

6.2. Certified provers

6.2.1. Calculus of Congruent Constructions

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

In [26], [51], we described a modification of the Calculus of Inductive Constructions allowing the use of decision procedures in the computation mechanism. In [21], we gave a new definition of the calculus without most of the restrictions made in [26], [51], and proved its core logic in Coq. This development has been the basis of CoqMT, our new version of Coq. As a paradigmatic example, we developed the basic theory of dependent lists with CoqMT. Compared with the same development for non-dependent lists, very few modifications were necessary to carry out the proofs.

We also started several generalisations of the previous work. Two are especially important: the ability to consider polymorphic first-order theories, and the extraction of equations from pattern matching.

6.2.2. Certification of SAT solvers

Participants: Frédéric Blanqui, Jean-Pierre Jouannaud, Pierre-Yves Strub, Bow-Yaw Wang, Lianyi Zhang.

We started to work on the certification of unsatisfiability proofs given by a set of regular input resolution proofs as provided by the PicoSAT solver[7] and described in http://fmv.jku.at/tracecheck/README.tracecheck. In order to make experiments, we also developped a new version of MiniSAT in OCaml, outputting a trace in the PicoSAT format.

6.2.3. Maxterm covering for satisfiability

Participants: Ming Gu, Fei He, Liangze Yin.

Boolean satisfiability (SAT) is to find if there is a true interpretation for a Boolean formula. Many real-world problems can be transformed into SAT problems and many of these problem instances can be effectively solved via satisfiability, such as testing, formal verification, synthesis, various routing problems, etc. In [18], we present a novel efficient SAT algorithm based on maxterm covering. The satisfiability of a clause set is determined in terms of the number of relative maxterms of the empty clause with respect to the clause set. If the number of relative maxterms is zero, it is unsatisfiable, otherwise satisfiable. A set of synergic heuristic strategies are presented and elaborated. We conduct a number of experiments on 3-SAT problems at the phase transition region of density 4.3, which have been cited as the hardest group of SAT problems. Our experimental results on public benchmarks attest to the fact that, by incorporating our proposed heuristic strategies, our enhanced algorithm can handle 3-SAT problems with 400 variables. The approach runs 3 to 40 times faster than zChaff does for both satisfiable and unsatisfiable problems.

6.3. Decision procedures

6.3.1. Decidable fragment of array theory

Participants: Fei He, Bow-Yaw Wang, Min Zhou.

[6] http://compcert.inria.fr
[7] http://fmv.jku.at/picosat/

Array theory is very useful in program verification. The most popular technique for deciding array theory is to reduce to the theory of uninterpreted functions. Although the technique is widely used in SMT solvers, it has several drawbacks. In program verification, one often needs array theory with quantified indices. Enriching the theory of uninterpreted functions with quantifiers leads to undecidability easily. Recently, a new technique that applies counter automata allows more general quantification in array theory [27]. But its algorithm requires two reductions: one from array theory to the reachability of counter automata; the other from counter automata to Presburger arithmetic. In this project, we reduce array theory to the logic of weakly monadic second-order with one successor (WS1S). We have developed an equi-satisfiable reduction and conducted preliminary experiments with Mona.

6.3.2. *Certified simplex*

Participant: Pierre-Yves Strub.

For the use of CoqMT (or SMT), we need the availability of decision procedures either certified or generating certificates. We studied the certification of decision procedures for the case of (integer or rational) linear arithmetic. For that purpose, we developed in Coq the theory related to the simplex method, a well known linear optimization algorithm over linear constraints. This development [?] includes:

i) the definition and proof of basic properties of the ordered rings and fields,

ii) the definition and proof of basic properties of polytopes (weak Krein-Milman theorem),

iii) the correction of the simplex algorithm steps and

iv) the correction and completeness of the halting conditions.

This work is based of the SsReflect tactic language and libraries developed in the team Mathematical Components of the Microsoft Research-INRIA Joint Centre. It is done in cooperation with Assia Mahboubi, from the TypiCal group at INRIA Saclay - Ile de France.

6.3.3. *Certificates of a small size for linear arithmetic*

Participant: Pierre-Yves Strub.

For solving large problems, the use of decision procedures using complex optimizations and heuristics is necessary. Proving correctness of such programs can be very tedious. Moreover, proofs have to be updated each time the decision procedure algorithm is modified. A workaround is to write non verified algorithm generating certificates at each run. These certificates must be small and easily verifiable.

We studied [20] this problem for the case of linear arithmetic. We instrumented a general simplex algorithm so that it can generate certificates. Our certificates for the harder case (non-satisfiable problems) are i) linear in the number of involved equations in the rational case, ii) linear in the number of involved equations and number of Gomory cuts in the integer case: the better your cutting heuristic, the smallest your certificate. Certificates can then be checked by using very simple, inexpensive computations.

This implementation is currently used in the CoqMT kernel to check conversion goals involving arithmetic constraints. We also wrote a new Coq tactic for the resolution of arithmetic goals. We expect far better results than the current tactics of Coq.

This work is done in cooperation with Assia Mahboubi, from the TypiCal group at INRIA Saclay - Ile de France.

6.4. Termination

6.4.1. *Certification of termination proofs*

Participant: Frédéric Blanqui.

In [17], Frédéric Blanqui and Adam Koprowski (Radboud University) present a methodology and a tool, Rainbow, for the automated verification of the results of such automated termination provers. This is accomplished by means of termination certificates that can be easily generated by termination provers, and by the transformation of these certificates into full formal proofs in some proof assistant/checker. This last step is done by formalizing (in Coq) the proofs of termination criteria used in modern termination provers. The above paper describes the formalization of some of these criteria in the proof assistant Coq (the CoLoR library) and the application of those formalizations in the transformation of termination certificates into termination proofs verifiable by Coq.

Since then, we improved the efficiency of proof checking and extended Rainbow and CoLoR with syntactic first-order matching, the verification of loops in term and string rewrite systems (to certify non-termination), and semantic labelling [53].

We also started to formalize the termination of Haskell programs [37] (internship of Julien Bureaux, ENS Paris, from June 1st to July 31), and to formalize Rainbow itself in order to certify it and improve the efficiency of proof checking by using the extraction mechanism of Coq to OCaml.

6.4.2. Computability Path Ordering

Participants: Frédéric Blanqui, Jean-Pierre Jouannaud, Jianqi Li.

The Computability Path Ordering of Blanqui, Jouannaud and Rubio [25] is a well founded order on algebraic lambda terms aiming at proving strong normalization of higher-order rewrite rules. CPO accepts weakly polymorphic algebraic signatures only. We are currently generalizing the well-foundedeness proof of CPO to the more general case of fully polymorphic signatures before to consider the case of dependently typed disciplines.

6.4.3. Higher-order dependency pairs

Participant: Frédéric Blanqui.

Higher-order rewrite systems (HRSs) and simply-typed term rewriting systems (STRSs) are computational models of functional programs. In [12], together with Y. Isogai, K. Kusakari and M. Sakai (Nagoya University), we proposed an extremely powerful method, the static dependency pair method, which is based on the notion of strong computability, to prove termination of STRSs. In this paper, we extend the method to HRSs. 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.

6.4.4. Size-based termination

Participant: Frédéric Blanqui.

In [13], with Cody Roux (INRIA Pareo), we have investigated the relationship between two independently developed termination techniques. On the one hand, sized-types based termination (SBT) uses types annotated with size expressions and Girard's reducibility candidates, and applies on systems using constructor matching only. On the other hand, semantic labelling transforms a rewrite system by annotating each function symbol with the semantics of its arguments, and applies to any rewrite system.

First, we introduce a simplified version of SBT for the simply-typed lambda-calculus. Then, we give new proofs of the correctness of SBT using semantic labelling, both in the first and in the higher-order case. As a consequence, we show that SBT can be extended to systems using matching on defined symbols (e.g. associative functions).

In addition, we started to study how we could use this size information in order to check the correctness of upper bounds on the complexity of functions (internship of Antoine Taveneaux, ENS Lyon, from May 15 to July 31).

6.5. Confluence

6.5.1. Confluence of parameterized rewrite systems

Participant: Jean-Pierre Jouannaud.

Together with Benjamin Monate (CEA), we are interested in proofs of properties of infinite families of specifications, like the family of dihedral groups of order n for some natural number n, or the family of a multicore harware with 2^n cores for some natural number n. So far, we have shown the decidability of confluence when these families can be presented by parameterised words over a finite alphabet of parameterized size, as in the case of the example of dihedral groups.

6.5.2. Confluence of higher-order rewrite systems

Participant: Jean-Pierre Jouannaud.

Together with Femke van Raamsdonk from the Free University of Amsterdam, we have started a program to investigate decidable sufficient conditions for the confluence of higher-order rewrite systems when lefthand sides are patterns “a la Miller” which can be fired by using higher-order pattern matching [39]. The difficulty here lies in the fact that it is difficult to abstract from a particular syntax for lambda-terms, such as de Bruijn numbers, localy nameless variables, or freshness conditions. We have not been able yet to prove our results in an axiomatic setting capturing all these various syntax for binders.

6.5.3. Certified confluence

Participants: Jean-Pierre Jouannaud, Huiying Luo.

Decreasing diagrams are a technique due to van Oostrom for proving confluence results for abstract relations which captures both styles of proofs based respectively on strong and local confluence. Last year, Van Oostrom and Jouannaud developped a refinement of this technique in order to handle relations defined by rewrite systems [15]. We continue this work in order to get rid of some linearity restrictions, and plan to develop a Coq library in order to search for and certify confluence proofs.

6.6. Assume-Guarantee reasoning

6.6.1. Simultaneous Assume-Guarantee reasoning through learning

Participants: Fei He, Bow-Yaw Wang, He Zhu.

To generate finite automata as contextual assumptions, the exact learning algorithm L^* for finite automata is used. In the simplest setting, the system under verification is decomposed into two components. An instance of the L^* algorithm is deployed to find a proper contextual assumption to verify the system. In more realistic settings, the optimal decomposition may consist of several components [54]. One could deploy several independent instances of the L^* algorithm to find assumptions for these components. The naïve deployment however would disregard semantic information among components. In this project, we would like to incorporate such information into instances of the L^* algorithm. We have discussed ideas to coordinate the construction of contextual assumptions in each L^* algorithm.

6.6.2. Implicit Assume-Guarantee reasoning through learning

Participant: Bow-Yaw Wang.

Contextual assumptions are required to apply assume-guarantee reasoning. Previously, assumptions are computed explicitly [36]. It has been reported that the explicit assume-guarantee reasoning is less efficient than explicit monolithic algorithms. To address this problem, we apply an exact learning algorithm for Boolean formulae to generate assumptions implicitly. For the invariant checking problem, our new algorithm derives initial predicates and transition relations represented by Boolean formulae implicitly. We have implemented a prototype. Preliminary experiments show that our algorithm is comparable to monolithic SAT-based algorithms for small cases.

6.6.3. Data mining based decomposition for Assume-Guarantee reasoning

Participants: Ming Gu, Fei He, He Zhu.

Automated compositional reasoning using assume-guarantee rules plays a key role in large system verification. A vexing problem is to discover fine decomposition of system contributing to appropriate assumptions. In [16], we present with William N. N. Hung and Xiaoyu Song an automatic decomposition approach in compositional reasoning verification. The method is based on data mining algorithms. An association rule algorithm is harnessed to discover the hidden rules among system variables. A hypergraph partitioning algorithm is proposed to incorporate these rules as weight constraints for system variable clustering. The experiments demonstrate that our strategy leads to order-of-magnitude speedup over previous.

6.7. Distributed algorithms

Participant: Jean-François Monin.

The population protocol model has emerged as a new computation paradigm for describing mobile *ad hoc* networks that consist of a number of mobile nodes interacting with each other to carry out a computation [23]. Correctness proofs of such protocols involve intricate arguments on infinite sequences of events. We formalize such proofs in the constructive framework given by Coq. Preliminary results on the leader election problem show that we gain interesting insights on the behaviour of such algorithms.

This work is done in cooperation with Deng Yuxin, from the BASICS group at Shanghai Jiaotong University.

7. Contracts and Grants with Industry

7.1. Schneider Electric

The goal of this project contracted with Schneider Electric China is to develop a full system simulator for a System-On-Chip used by Schneider Electric in their automation product line.

7.2. Orange IT Labs

The goal of this project is to complete the PowerPC simulator and compare its performance with another simulator used internally by Orange IT Labs.

8. Other Grants and Activities

8.1. International Initiatives

- SIVES is a French-Chinese ANR project for 2009-2011 between INRIA FORMES, Verimag, ST Microelectronics, Tsinghua University and Beihand University on the development of a “SImulation and Verification based platform for Embedded Systems” (coordinated by Frédéric Blanqui).
- FORMES is part of the Sino-French Laboratory for Computer Science, Automation and Applied Mathematics (LIAMA)[8] whose French director is Jean-Pierre Jouannaud since November 11.
- FORMES co-organized the 1st Workshop on Simulation Based Development of Certified Embedded Systems[9] with CVS/AIST (Japan) from October 5 to 7.
- FORMES organized the 1st Asian-Pacific Summer School on Formal Methods[10] late august, 2009. The school attracted over 60 participants from China (mostly), Corea, Macao and Taiwan, see http://formes.asia/cms/coqschool/2009.

[8]http://liama.ia.ac.cn/wiki/
[9]http://unit.aist.go.jp/cvs/workshop/SBDCES.html
[10]http://formes.asia/cms/coqschool/2009

- Ming Gu and Vania Joloboff participated into the scientific committee and provided local organization for the 2009 ARTIST Summer School in China[11], a summer school organized by the ARTIST Design European Network of Excellence on the topics of Embedded Systems, which was held at Tsinghua University from July 19 to July 24.
- Last but not least, FORMES organizes a weekly seminar which has been proved to be a major local forum in the area of formal methods, with a steady participation of colleagues who come from the other nearby research institutions, CASIA, ISCAS and Pekin University, to attend the presentations. All seminars are announced on our website, as well as the other relevant local seminars or events, in particular those taking place at ISCAS.

8.2. National Initiatives

- FORMES is part of the working group LTP on Languages, Types and Proofs of the GDR GPL[12], the French research network on software engineering.
- FORMES is part of the working group LAC on Logic, Algebra and Calculus of the GDR IM[13], the French research network on mathematics and computer science.

9. Dissemination

9.1. Visits and presentations

- Jean-François Monin visited Yuxi Fu (Jiaotong University, Shanghai) from October 11 to October 16, and from December 14 to December 18.
- Vania Joloboff was invited to give presentations at the Shanghai Jiaotong University, the Hunan University and the National University of Defense Technology (Changsha).
- Jean-Pierre Jouannaud gave a seminar presentation at Ecole Polytechnique, in June 2009, on his work on confluence with van Oostrom.
- Jean-Pierre Jouannaud gave an invited presentation at the 10th Anniversary of the France-Taiwan Scientific Prize, September 2009, NTU, Taipei, Taiwan.
- Jean-Pierre Jouannaud was guest speaker in the Distinguished lecture series of Academia Sinica, October 2009, Taipei, Taiwan.
- Jean-Pierre Jouannaud was keynote speaker at the IEEE Open Source Software Conference, October 2009, Guiyang, China.
- Vania Joloboff and Jean-Pierre Jouannaud gave an invited presentation at the Microsoft Research Asia Verified Software Workshop, October 2009, Beijing, China.
- Frédéric Blanqui, Jean-Pierre Jouannaud and Vania Joloboff gave an invited presentation at the workshop SBDCES, October 2009, Osaka, Japan.
- Jean-Pierre Jouannaud and Pierre-Yves Strub gave a seminar presentation at USTC, November 2009, Suzhou, China.

9.2. Editorial duties

- Jean-Pierre Jouannaud is a member of the editorial board of IJSI.

9.3. Program committees

[11] http://www.artist-embedded.org/artist/Overview,1630.html
[12] http://gdr-gpl.cnrs.fr/
[13] http://www.gdr-im.fr/

- Frédéric Blanqui was PC member of the 4th International Workshop on Logical Frameworks and Meta-languages: Theory and Practice (LFMTP'09), the 10th International Workshop on Termination (WST'09) and the 1st Coq Workshop (Coq'09).
- Jean-Pierre Jouannaud was PC member of Developments in Computational Models 2009: Computational Models From Nature, ICALP workshop, July 11, Rhodes, Greece.
- Jean-Pierre Jouannaud is PC Chair of LICS 2010 in Edinburgh, UK.

9.4. Internships

- Frédéric Blanqui supervised the 2-months internship of Julien Bureaux (ENS Paris) on the certification in Coq of the termination of Haskell programs.
- Frédéric Blanqui supervised the 2-months internship of Antoine Taveneaux (ENS Lyon) on the automated verification of complexity bounds using sized types.
- Frédéric Blanqui supervised the internships of Lianyi Zhang and Qian Wang on the certification of loops in rewrite systems.
- Vania Joloboff supervised the 4-month internships of Pascal Combier, Patrice Seng and Ren Ligang, as well as the one-year internships of Bin Liu, Yuning Pang, Ming Liu and Bing Zhou.

9.5. Teaching

- Jean-Pierre Jouannaud participated in january to the selection of chinese candidates to the postdoc program of the "Fondation Franco-Chinoise pour la Science et ses Applications".
- Frédéric Blanqui, Jean-François Monin and Pierre-Yves Strub participated to the classes in the 1st Asian-Pacific Summer School on Formal Methods.
- Frédéric Blanqui gave lectures on "System F *à la* Curry" at Tsinghua University.
- Jean-Pierre Jouannaud gave lectures at Tsinghua university, at both the graduate and undergraduate level.
- Frédéric Blanqui and Pierre-Yves Strub gave lectures on Coq at Tsinghua university (graduate class).
- Jean-François Monin participated to the oral examinations and selection of chinese students applicants to the Polytech'Network.

9.6. Long-term visitors

- Cody Roux, PhD at INRIA Nancy in the Pareo team, supervised by Claude Kirchner (INRIA Bordeaux), Gilles Dowek (INRIA Saclay, École Polytechnique) and Frédéric Blanqui visited Formes from June 15 to July 5, and from November 18 to December 16, to work on his thesis with Frédéric Blanqui.
- Pierre-Louis Curien (CNRS), leader of INRIA πr^2 project-team, visited FORMES from August 18 to October 11 and gave lectures on "Proof theory: sequent calculus and focalisation".
- Jean-Jacques Lévy (INRIA), director of the MSR-INRIA Joint Center, visited FORMES from October 13 to December 5, gave lectures on "Caml, OCaml and Jocaml" and a talk at ISCAS on "Three years of research at the MSR-INRIA Joint Centre". His visit was partially supported by ISCAS.

9.7. Short-term visitors

- Kokichi Futatsugi (JAIST) gave a talk on April 3 on "Combining Inference and Search in Verification with CafeOBJ".

- Gilles Dowek (Ecole Polytechnique) gave a talk on April 17 on “Polarized Resolution Modulo”.
- Yoshiki Kinoshita (AIST) gave a talk on April 24 on “Introduction to Agda language and system”.
- Christian Urban (TU Muenchen) gave a talk on May 22 on “Nominal Techniques in the Theorem Prover Isabelle or, How Not to be Intimidated by the Variable Convention”.
- Hubert Comon-Lundh (INRIA & AIST) gave a talk on May 26 on “Models for security protocols”.
- Hugo Herbelin (INRIA) gave a talk on June 16 on “Coq: current development issues”.
- Gilles Barthe (IMDEA Software) gave a talk on September 1st on the “Certification of code-based cryptographic proofs”.
- Laurent Fribourg (CNRS & ENS Cachan) gave a talk on October 12 on “Detecting Race Condition in Concurrent Systems with the Inverse Method”.
- Joseph Sifakis (CNRS & INRIA-Schneider), Turing Award 2007, gave a talk on October 26 on “Embedded Systems Design: Scientific Challenges and Work Directions”, and on October 27 on “Component-based Construction of Heterogeneous Real-time Systems in BIP”.
- T. John Koo, Director of the Center for Embedded Software Systems, Shenzhen Institute of Advanced Technology (SIAT), Chinese Academy of Sciences, gave a talk on November 13 about“Model-based tool-chain for the design and analysis of embedded hybrid systems”.

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] 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 "Proceedings of the 5th International Conference on Theoretical Computer Science, International Federation for Information Processing 273", 2008.

[4] B. BÉRARD, L. FRIBOURG, F. KLAY, J.-F. MONIN. *A compared study of two correctness proofs for the standardized algorithm of ABR conformance*, in "Formal Methods in System Design", january 2003.

[5] 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, vol. 2491, Springer-Verlag, 2002, p. 197–212, ISBN 3-540-44307-X.

[6] F. HE, X. SONG, M. GU, J. SUN. *Heuristic-Guided Abstraction Refinement*, in "Computer Journal", vol. 52, n^{o} 3, May 2009, p. 280-287.

[7] 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.

[8] C. JARD, J.-F. MONIN, R. GROZ. *Development of Veda, a Prototyping Tool for Distributed Algorithms*, in "IEEE Transactions on Software Engineering", vol. 14, n^{o} 3, march 1988, p. 339–352.

[9] 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.

[10] 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.

[11] Y.-K. TSAY, B.-Y. WANG. *Automated Compositional Reasoning of Intuitionistically Closed Regular Properties*, in "International Journal on Foundation of Computer Science", vol. 20, n^{o} 4, 2009, p. 747-762.

Year Publications

Articles in International Peer-Reviewed Journal

[12] K. KUSAKARI, Y. ISOGAI, M. SAKAI, F. BLANQUI. *Static Dependency Pair Method based on Strong Computability for Higher-Order Rewrite Systems*, in "IEICE Transactions on Information and Systems", 2009, http://hal.inria.fr/inria-00397820/en/CNJP.

International Peer-Reviewed Conference/Proceedings

[13] F. BLANQUI, C. ROUX. *On the relation between sized-types based termination and semantic labelling*, in "18th EACSL Annual Conference on Computer Science Logic - CSL 09, Portugal Coimbra", 2009, http://hal.inria.fr/inria-00397689/en/CN.

[14] V. JOLOBOFF, C. HELMSTETTER, H. XIAO. *SimSoC: A full system simulation software for embedded systems*, in "2009 International Workshop on Open-source Software for Scientific Computation (OSSC-2009), Chine Guiyang", IEEE (editor), LIAMA, Institute of Automation, CAS, Beijing, China Guizhou Normal University, China, 2009-09-20, http://hal.inria.fr/inria-00435247/en/, ossc2009CN.

[15] J.-P. JOUANNAUD, V. VAN OOSTROM. *Diagrammatic Confluence and Completion*, in "International Conference in Automata, Languages and Programming, Grèce Rhodes", W. THOMAS (editor), vol. 2, Springer Berlin/Heidelberg, Paul Spirakis, 2009-07-06, http://hal.inria.fr/inria-00436070/en/CN.

[16] H. ZHU, F. HE, W. N. N. HUNG, X. SONG, M. GU. *Data Mining Based Decomposition for Assume-Guarantee Reasoning*, in "FMCAD, Austin, Texas", 2009.

Research Reports

[17] F. BLANQUI, A. KOPROWSKI. *Automated Verification of Termination Certificates*, INRIA, 2009, http://hal.inria.fr/inria-00390902/en/, RR-6949, Rapport de rechercheCNNL.

[18] L. YIN, F. HE, W. N. N. HUNG, X. SONG, M. GU. *Maxterm Covering for Satisfiability*, Tsinghua University, 2009, Technical report.

Other Publications

[19] A. MAHBOUBI, P.-Y. STRUB. *A Formalization of the simplex algorithm theory*, 2009, Draft.

[20] A. MAHBOUBI, P.-Y. STRUB. *Efficient certificates for linear arithmetic problems*, 2009, Draft.

[21] P.-Y. STRUB. *Coq Modulo Theory: an implementation of the Calculus of Inductive Congruent Constructions*, 2009, Draft.

References in notes

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

[23] D. ANGLUIN, J. ASPNES, M. J. FISCHER, H. JIANG. *Self-stabilizing population protocols*, in "Proceedings of the 9th International Conference on Principles of Distributed Systems", Lecture Notes in Computer Science, vol. 3974, Springer, 2005, p. 103-117.

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

[25] F. BLANQUI, J.-P. JOUANNAUD, A. RUBIO. *The Higher-Order Computability Path Ordering: the end of a Quest*, in "Proceedings of the 22nd International Conference on Computer Science Logic, Lecture Notes in Computer Science 5213", 2008, Invited paper.

[26] 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 "Proceedings of the 5th International Conference on Theoretical Computer Science, International Federation for Information Processing 273", 2008.

[27] M. BOZGA, P. HABERMEHL, R. IOSIF, F. KONECNÝ, T. VOJNAR. *Automatic Verification of Integer Array Programs*, in "CAV", 2009, p. 157-172.

[28] 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.

[29] 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.

[30] 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.

[31] COQ DEVELOPMENT TEAM. *The Coq Reference Manual, Version 8.2*, INRIA Rocquencourt, France, 2008, http://coq.inria.fr/.

[32] 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.

[33] 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.

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

[35] 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.

[36] D. GIANNAKOPOULOU, C. S. PASAREANU. *Special issue on learning techniques for compositional reasoning*, in "Formal Methods in System Design", vol. 32, n^o 3, 2008, p. 173-174.

[37] J. GIESL, S. SWIDERSKI, P. SCHNEIDER-KAMP, R. THIEMANN. *Automated Termination Analysis for Haskell: From Term Rewriting to Programming Languages*, in "Proceedings of the 17th International Conference on Rewriting Techniques and Applications, Lecture Notes in Computer Science 4098", 2006.

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

[39] J.-P. JOUANNAUD, F. V. RAAMSDONK. *Confluence Properties of Terminating Higher-Order Rewrite Relations*, in "Mathematical Theories of Abstraction, substitution and naming in Computer Science", ICMS, may 2007.

[40] X. LEROY, D. DOLIGEZ, J. GARRIGUE, D. RÉMY, J. VOUILLON. *The Objective Caml system release 3.11, Documentation and user's manual*, INRIA, France, 2008, http://caml.inria.fr/.

[41] X. LEROY. *A formally verified compiler back-end*, in "Journal of Automated Reasoning", vol. 43, n^o 4, 2009, p. 363-446.

[42] 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.

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

[44] 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.

[45] 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.

[46] 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.

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

[48] 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.

[49] 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.

[50] 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.

[51] P.-Y. STRUB. *Type Theory and Decision Procedures*, École Polytechnique, July 2008, Ph. D. Thesis.

[52] 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.

[53] H. ZANTEMA. *Termination of Term Rewriting by Semantic Labelling*, in "Fundamenta Informaticae", vol. 24, 1995, p. 89-105.

[54] H. ZHU, F. HE, W. N. N. HUNG, X. SONG, M. GU. *Data Mining Based Decomposition for Assume-Guarantee Reasoning*, in "FMCAD, Austin, Texas", 2009.