INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team cassis

Combinaison d'Approches pour la Sécurité des Systèmes InfiniS

Lorraine

THEME 2A

Table of contents

1. Team **1**
2. Overall Objectives **2**
2.1. Background 2
2.2. Context 2
2.3. Challenge 3
3. Scientific Foundations **4**
3.1. Introduction 4
3.2. Automated deduction 4
3.3. Synthesizing and solving set constraints 5
3.4. Reachability analysis for infinite state systems 5
3.4.1. Context 5
3.4.2. Approximations and accelerations 6
4. Application Domains **6**
4.1. Verification of security protocols 6
4.2. Automated boundary testing from formal specifications 6
4.3. Program debugging and verification 7
5. Software **8**
5.1. CASRUL 8
5.2. BZ-Testing-Tools 9
5.3. haRVey and related tools 9
5.4. CLPS 10
5.5. daTac 10
5.6. SPIKE 10
6. New Results **10**
6.1. Automated deduction 10
6.1.1. Decision procedures and their extensions 10
6.1.2. Symbolic constraint solving 11
6.1.3. Verification of collaborative editors 12
6.2. Security protocol verification 12
6.2.1. Extension of the Dolev-Yao model 12
6.2.2. Intruder knowledge approximation 13
6.2.3. On-line intrusion detection 13
6.3. Reachability analysis 13
6.3.1. Reachability analysis for testing 13
6.3.2. Reachability computation for parametric systems 14
6.3.3. Test case and test driver generation from a formal model using constraint solving 15
6.3.4. Approximation of linear transition systems 15
7. Contracts and Grants with Industry **15**
7.1. Industrial contracts 15
7.2. RNTL 16
7.3. ANVAR 16
7.4. IST AVISPA 16
7.5. INTERREG Test-UML 16
8. Other Grants and Activities **16**
8.1. International grants 16
8.2. National grants 17
8.3. International grants 18

8.4. Individual involvement 18
8.5. Visits of foreign researchers 19
8.6. Visits of team members 19
9. Dissemination **20**
9.1. Ph. D. theses 20
9.2. Committees 20
9.3. Seminars, workshops, and conferences 20
10. Bibliography **22**

1. Team

CASSIS is a joint project between Laboratoire Lorrain de Recherche en Informatique et ses Applications (LORIA - UMR 7503) and Laboratoire d'Informatique de l'Université de Franche-Comté (LIFC - FRE 2661).

Head of project-team
M. Rusinowitch [LORIA]

Vice-Head of project-team
F. Bellegarde [Université Franche-Comté, LIFC]

Administrative assistant
S. Drouot [LORIA]

Staff member (INRIA)
S. Ranise [LORIA]

Staff member (CNRS)
V. Cortier [from Oct. 1, 2003, LORIA]

Faculty member (Université Nancy 2)
L. Vigneron [LORIA]

Faculty members (Université Franche-Comté)
F. Ambert [LIFC]
F. Bouquet [LIFC]
G. Cécé [LIFC]
A. Giorgetti [LIFC]
P.-C. Héam [LIFC]
O. Kouchnarenko [LIFC]
B. Legeard [LIFC]
F. Peureux [from Oct. 1, 2003, LIFC]

Post-doctoral fellows
D. Déharbe [Until Mar. 1, 2003, Brazil]
A. Imine [Algeria]
C. Qin [Until Oct. 1, 2003, China]
S. Stratulat [Teaching assistant until Oct. 1, 2003, LORIA]

Ph. D. Student
T. Abbes [BDI-PED, LORIA]
Y. Boichut [INRIA, LIFC]
C. Charlet [Teaching Assistant, LIFC]
S. Chemin [ANVAR, LIFC]
Y. Chevalier [Teaching Assistant, LORIA]
J.-F. Couchot [PRAG UFC, LIFC]
F. Dadeau [ACI, LIFC]
Y. Mainier [FEDER, LIFC]
J. Musset [DGA, LORIA]
F. Peureux [Teaching Assistant until Sept. 1, 2003, LIFC]
J. Santos Santiago [INRIA, LORIA]
M. Turuani [MENRT, LORIA]
N. Vacelet [MENRT, LIFC]

Project Technical Staff
M. Bouallagui [INRIA, until Sept. 1, 2003, LORIA]
S. Guenaud [ANVAR, LIFC]

2. Overall Objectives

2.1. Background

CASSIS is a joint project between *Laboratoire Lorrain de Recherche en Informatique et ses Applications (LORIA - UMR 7503)* and *Laboratoire d'Informatique de l'Université de Franche-Comté (LIFC - FRE 2661)*.

The objective of the project is to design and develop tools to verify the safety of systems with an infinite number of states. The analysis of such systems is based on a symbolic representation of sets of states in terms of formal languages or logical formulas. Safety is obtained via automatic proof, symbolic exploration of models or test generation. These validation methods are complementary. They rely on the study of accessibility problems and their reduction to constraint solving.

An originality of the project is its focus on infinite systems, parameterized or large scale, for which each technique taken separately shows its limits. Such is the case for example of protocols operating on topologies of arbitrary size (ring networks), systems handling data structures of any size (sets), or whose control is infinite (automata communicating through an unbounded buffer). Ongoing or envisioned applications concern embedded software (e.g. smart cards, automotive controllers), cryptographic protocols (IKE, SET, TLS, Kerberos) designed to ensure trust in electronic transactions, and distributed systems.

The problem of validating or verifying reactive systems is crucial with respect to the increasing number of security-sensitive systems. The failure of these critical systems can have dramatic consequences since for instance they are embedded in vehicles components, or they control power stations or telecommunication networks. Beside obvious security issues the reliability of products whose destination is millions of end-users has a tremendous economical impact.

There are several approaches to system verification: automated deduction, reachability analysis or model-checking, and testing. These approaches have different advantages and drawbacks. Automated deduction can address practical verification however it remains complex to handle and requires a lot of expertise and guidance from the user. Model-checking is exhaustive but must face combinatorial explosion and becomes problematic with large-size or infinite systems. Testing is fundamental for validating requirements since it allows for discovering many errors. However, it is almost never exhaustive and therefore only leads to partial solutions. Hence we believe that these approaches should not be considered as competing but complementary.

The goal of our project is to contribute to new combinations of these three verification techniques in a framework that would allow applying them in an industrial context. In particular we expect some breakthrough in the infinite-state verification domain by joint applications of deductive, model-checking and testing techniques.

2.2. Context

For verifying the security of infinite state systems we rely on

- Different ways to express the safety, reachability or liveness properties of systems, linear-time or branching-time logics, ..., the application of abstraction or abstract interpretation of models in order to accelerate (or make it feasible for infinite state systems) analysis based on model exploration.
- Test generation techniques and constraint-based model-checking.
- The modeling of systems whose verification needs to consider an infinite number of states. This modeling may be obtained by encoding states as words, terms or trees and by representing infinite sets of states by languages. To each of these structures corresponds appropriate action families, such as transductions or rewritings.

Our goal is to apply these different approaches for ensuring the security of industrial systems by providing adequate methods and tools. In more details we aimed at the following contributions (see continuous lines in Figure 1):

1. verification of abstract models derived from existing systems;
2. tests generation from the abstract model for validating the existing model;
3. cross-fertilization of the different validation techniques (deduction, model-checking, test) by taking advantage of the complementarity scopes and of their respective algorithmic contributions.

Let us mention that all these techniques comply with various development methodologies.

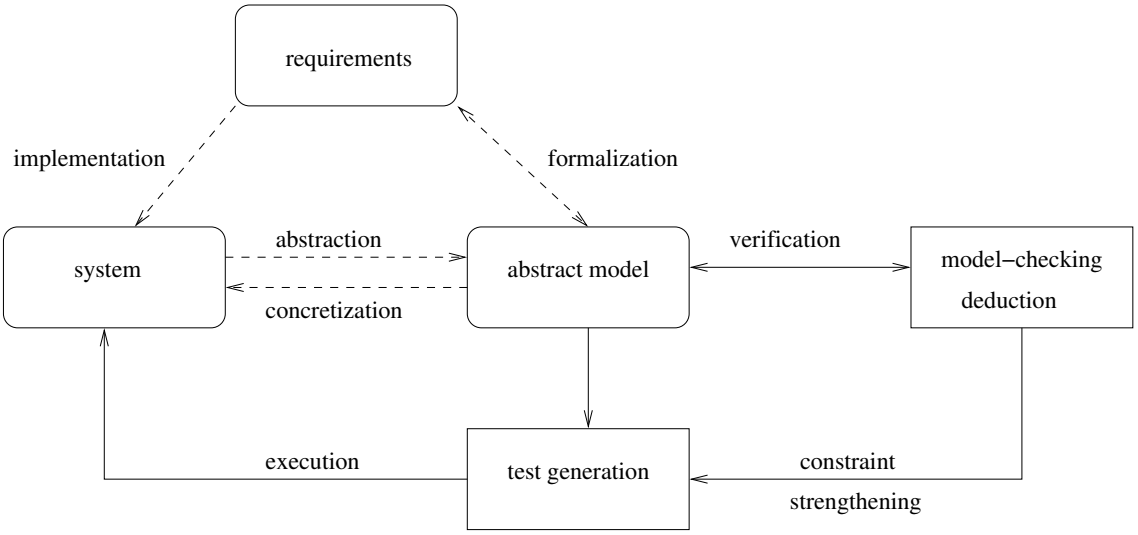

Figure 1. Software validation in CASSIS

2.3. Challenge

Verifying the safety of infinite state systems is a challenge: nowadays algorithmic techniques only apply to very specific infinite state systems. Let us mention *well-structured transition systems* [78] or systems with security properties that can be verified on finite abstractions of them. On the other hand the deductive approaches are good candidates to capture infinite system safety verification but are difficult to bring into operation and require a deep expertise. A solution consists of integrating several verification methods by combining theorem-proving and model-checking for instance.

The behavior of infinite states systems is expressed in the various models by composing or iterating actions. One of the main problems with algorithmic techniques is to compute the effect of these actions on the initial state. This computation is called *reachability analysis*. The verification of safety properties as well as the automatic generation of test cases rely heavily on the accuracy of reachability analysis.

The transverse goal is to push away the limitations on the use of formal verification techniques, to ease their applications, and to let them scale-up.

1. For properties that can be checked by reachability analysis we have proposed models based on regular languages and rational transductions, relying on works by Bouajjani *et al.* [68]. We have completed them by designing algorithms for verifying a refinement relation between two models $\mathcal{S}$ and $\mathcal{T}$ i.e. for checking that every behaviour of $\mathcal{T}$ is a behaviour of $\mathcal{S}$ [66]). This refinement relation when satisfied preserves the safety properties and therefore allows them to be inherited. We shall investigate this approach with other representations.

2. In order to generate boundary-value functional test cases [6], we abstract models as constrained states. These constraints are solved by a customized solver, called CLPS [19]. The test cases are derived in two steps:

   i. partitioning of the formal model and extraction of boundary values,
   ii. reachability graph exploration from constrained states in order to reach boundary values and generate state sequences (trace) as test cases with the oracle.

   After the generation phase, a reification is used to produce the test drivers from the generated test cases in order to perform it on the system under test and to provide automatic verdict assignment [31]. Furthermore, the kernel of the engine allows one to perform specification animations in order to validate the model [70].
   The kernel of the engine allows also to perform specification animations too [84] for validating the model. Moreover the abstraction allows verifying linear temporal logic properties for systems with a large number of states [83].
3. For the safety on infinite state systems we have designed automated deduction tools based on term rewriting **SPIKE** [2], **daTac** [7], **haRVey** [39] and an extensible and modular platform for detecting flaws and potential attacks on security protocols [5] [72]. The main techniques involved in these systems are deduction, constraint solving and induction. The tools have been built on the modeling of systems by terms and rewrite rules. Our works with other models based on regular languages of words or trees and of transducers should complement these term rewriting models.

In order to address this challenge, we rely on complementary skills within the project. We believe that each of the three techniques will benefit from concepts and algorithms designed for the two others.

3. Scientific Foundations

3.1. Introduction

Our main goal is to design techniques and to develop tools for the verification of (safety-critical) systems, such as programs or protocols. To this end, we develop a combination of techniques based on automated deduction for program verification, constraint resolution for test generation, and reachability analysis for the verification of infinite state systems.

3.2. Automated deduction

The main goal is to prove the validity of assertions obtained from program analysis. To this end, we develop techniques and automated deduction systems based on rewriting and constraint solving. The verification of recursive data structures relies on inductive reasoning or the manipulation of equations and it also exploits some form of reasoning modulo properties of selected operators (such as associativity and/or commutativity).

Rewriting, which allows us to simplify expressions and formulae, is a key ingredient for the effectiveness of many state-of-the-art automated reasoning systems. Furthermore, a well-founded rewriting relation can be also exploited to implement reasoning by induction. This observation forms the basis of the approach to inductive reasoning implemented by **SPIKE**. This system also features a combination of many simplification techniques and allows us to detect invalid conjectures. There exist other reasoning systems providing a high degree of automation for induction such as NqThm and its descendent Acl2. However, these systems do not allow the user to refute false conjectures as it is the case for **SPIKE**.

The constraints are the key ingredient to postpone the activity of solving complex symbolic problems only when this is really necessary. They also allow us to increase the expressivity of the specification language and to refine theorem-proving strategies. As an example of this, the handling of constraints for unification problems or for the orientation of equalities in the presence of interpreted operators (e.g. commutativity and/or

associativity function symbols) will possibly yield shorter automated proofs. We investigate these techniques both from a conceptual and a practical viewpoint by implementing them in the system **daTac**. This tool, to the best of our knowledge, is the only one which integrates refutationally complete equational reasoning based on rewriting techniques modulo associative-commutative operators. However, when considering the pure equational case, the E prover [85] is more efficient thanks to its powerful redundancy elimination techniques and it is also used in this project.

Finally, decision procedures are being considered as a key ingredient for the successful application of automated reasoning systems to verification problems. A decision procedure is an algorithm capable of efficiently deciding whether formulae from certain theories (such as Presburger arithmetic, lists, arrays, and their combination) are valid or not. We intend to develop techniques to build and combine decision procedures for the domains which are relevant to verification problems. We also want to perform experimental evaluation of the proposed techniques by implementing them within the system **haRVey**, which features a combination of propositional reasoning (implemented by means of Binary Decision Diagrams or BDD) and of decision procedures, their combinations, and their extensions to semi-decision procedures handling larger (possibly undecidable) fragments of first-order logic.

Finally, in order to better understand the various sources of problems in automated theorem proving, we want to study the complexity of constraint solving problems such as unification and orientability of equations.

3.3. Synthesizing and solving set constraints

The aim of this research is the evaluation of set-oriented formal specifications. The current work concerns the development of a set constraint solving system based on the CLPS core [19].

By evaluation, we mean the rewriting of the formal model into a constraint system, and the ability for the solver to verify the invariant on the current constraint graph, to propagate preconditions or guards, and to apply the substitution calculus on this graph. The constraint solver makes it possible to look into the reachability graph, representing the states defined by the specification. It is directly used for animating specifications and automatically generating abstract test cases.

Applying constraint logic programming technology in the validation and verification area is currently an active way of research. It usually requires concepts on domain finiteness and also the writing of specific solvers to deal with the description language's vocabulary.

The technology used in the CLPS solver is based on rewriting set formulae into disjunctive normal form formulae based on set operators ($\in$, $\notin$, $=$, etc). Usual propagation rules on finite domains have been revisited with the addition of domains able to contain variables, the V-domains. This allows us to take advantage of the structure of the formula extracted from the formal model. But this approach requires a finiteness assumption, and in particular, the infinite terms, like the universe sets—*abstract sets in B*—must be replaced by a finite enumerated set before calling the CLPS solver. Moreover, the current version of the solver is able to deal with B notation, and the Z notation is currently being integrated, whereas the principles used in animation, invariant verification and test generation can be generalized to every set-oriented notation.

3.4. Reachability analysis for infinite state systems

The main objective of this task is deciding if non-desirable states can or cannot be reached by a huge size or infinite state system. This reachability problem is obviously crucial to guarantee the safety of critical systems.

3.4.1. Context

Presently, infinite state system verification is done on a (finite state) abstraction of the infinite system. These abstractions use symbolic representations coming from different theories such as languages and automata, various logics, term rewriting systems, constraint systems, etc.

We adapt and specialize these techniques to answer the particular question of the reachability analysis. This question is directly linked with some objectives of the other tasks. Indeed, before testing a limit state, one

might like to know if it is reachable. Also, a non-practical proof can be improved and assisted by knowing some useful sets of reachable states.

In order to improve the efficiency of the reachability analysis, the (semi-)algorithms which perform a symbolic exploration of infinite state systems need to be examined and adapted to huge size and parameterized systems. These algorithms are based on logical abstractions and theories requiring their combination with automatic demonstration tools.

3.4.2. Approximations and accelerations

The reachability sets of an infinite kind (communicating automata systems, parameterized systems) are in general non recursive. Our pragmatic goal is then to compute a sequence of sub-approximations of the set of reachable states. Such a sequence has to converge "often" and in a finite number of steps towards the exact reachability state set. This set is accurate since the application systems which need to be verified are far from having the power of Turing machines.

The first problem to be solved is how to obtain a finite representation of sets of infinite states. This representation has to be compatible with the way we compute the reachable state space (traversal of the control flow graph of the system and approximation of the effect of the loops in this control flow graph).

The second problem we have is how to compute in a finite number of steps an infinite number of reachable states [67][68]. This can only be solved by computing in one step the effect of infinite sequences of transitions of the system (called *meta-transitions*). The abstraction of infinite sequences of transitions into a *meta-transition* has to be adapted to the properties to be verified on the system.

4. Application Domains

4.1. Verification of security protocols

Security protocols such as SET, TLS and Kerberos, are designed for establishing the confidence of electronic transactions. They rely on cryptographic primitives the purpose of which is to ensure integrity of data, authentication or anonymity of participants, confidentiality of transactions, etc.

The experience has showed that the design of those protocols is often erroneous, even when admitting that cryptographic primitives are perfect, i.e. that an encoded message cannot be decrypted without the appropriate key. An intruder can intercept, analyze and modify the exchanged messages with very few computation means and therefore, for example, generate important economic damage.

Analyzing cryptographic protocols is complex because the set of configurations to consider is very large, and can even be *infinite*: one has to consider any number of sessions, any size of messages, sessions interleaving, algebraic properties of encryption or data structures.

Our objective is to automatize as much as possible the analysis of protocols starting from their specification. This consists in bringing a tool easy to use, permitting to specify a large number of protocols thanks to a standard high-level language, and permitting either to look for flaws in a given protocol or to check that it satisfies a given property. Such a tool is essential for verifying existing protocols, but also for helping in designing new ones. For our tool to be easy to use, it has to provide a graphical interface allowing a user to do only click-button.

The system **CASRUL** [5] that we are developing is a first prototype, giving an idea of what will be the final tool. It permits to consider many protocols, but its specification language has to be extended for handling e-business protocols for example.

4.2. Automated boundary testing from formal specifications

In [6], we have presented a new approach for test generation from set-oriented formal specifications: the BZ-TT method. This method is based on Constraint Logic Programming (CLP) techniques. The goal is to test every operation of the system at every *boundary state* using all *input boundary values* of that operation. The unique features of the BZ-TT method are that it:

- takes both B [62] and Z [87] specifications as input;

- avoids the construction of a complete finite state automaton for the system;
- produces boundary-value test cases (both boundary states and boundary input values);
- produces both negative and positive test cases;
- is fully supported by tools;
- has been validated in several industry case studies for smart card OS and application validation (GSM 11-11 standard [18] and Java Card Virtual Machine Transaction mechanism [31]) and for embedded automotive software (an automobile wind-screen wiper controller).

This test generation method can be summed up as follows: from the formal model, the system computes boundary values to create boundary states; test cases are generated by traversal of the state space with a preamble part (sequences of operations from the initial state to a boundary state), a body part (critical invocations), an identification part (observation and Oracle state computation) and a post-amble part (return path to initial or boundary state). Then, an executable test scripts file is generated using a test pattern and a table of correspondence between abstract operations (from the model) and concrete ones. This approach differs on several main points from the work of Dick, Faivre *et al*: first, using boundary goals as test objectives avoids the complete construction of the reachability graph; second, this process is fully automated and the test engineer could just drive it at the boundary value computation level or for the path computation.

The BZ-TT method is fully supported by the BZ-Testing-Tools tool-set. This environment is a set of tools dedicated to animation and test cases generation from B or Z formal specifications. It is based on the CLPS constraint solver, able to simulate the execution of the specification. By execution, we mean that the solver computes a so-called constrained state by applying the pre- and post-condition of operations. A constrained state is a constraint store where state variables and also input and output variables support constraints.

A unique feature of the BZ-Testing-Tools approach is its strong boundary testing orientation. Despite the enormous popularity of boundary-value testing strategy for black-box testing in the software practitioners guide, currently this approach has not been widely investigated for automated formal model-based test generation. In the BZ-TT approach, boundaries are computed both for state variables in each effect predicates and for the input variables in the computation of the body part of the test.

One orientation of the current work is to go beyond the finiteness assumption limitations by using symbolic constraint propagation during the test generation process.

4.3. Program debugging and verification

Catching bugs in programs is difficult and time-consuming. The effort of debugging and proving correct even small units of code can surpass the effort of programming. Bugs inserted while "programming in the small" can have dramatic consequences for the consistency of a whole software system as shown, e.g., by viruses which can spread by exploiting buffer overflows, a bug which typically arises while coding a small portion of code. To detect this kind of errors, many verification techniques have been put forward.

Static analysis has been used to automatically detect common bugs such as out-of-range array subscripts and variables used before initialization. Many commercial tools (such as PREfix [71]) are based on such techniques. More recently, in the static analysis community, there seems to be a growing demand for a more declarative approach.[1] In fact, while static analysis techniques are quite efficient, they are described operationally by means of the rules used in the analysis and the domains of abstract values. As a consequence, in some cases, the properties checked by the analysis may not match the expectations of the programmer. A declarative approach seems mandatory to enable the programmer to express the properties to be checked so that the results of the analysis can be confronted against his expectations. In this direction, some tools based on (extensions of) first-order logic have been developed (e.g. [81][79][77]). These tools take a program with some annotations written in (an extension of) first-order logic and they produce a set of formulae of (a fragment of)

[1] See, for example, the challenge at http://research.microsoft.com/specncheck/consel_challenge.htm.

first-order logic whose satisfiability implies that a bug is present in the code. In order to check for satisfiability, a theorem prover capable of handling the generated proof obligations must be available.

Theorem proving is also important to other systems for model checking software. For example, SLAM [65] implements the *abstract-check-refine* paradigm: build an abstract model, check the desired property and, if the check fails, refine the model, and repeat. The abstract model is a program with the same control-flow as the original, but only boolean variables tracking the state in the original program. Checking the property is done by reachability analysis: if a certain statement is not reachable in the abstracted program, then it is not reachable in the original one. However, the abstraction may be too coarse, so that a statement is reachable in the abstract program but not in the original one. In such a case, *counterexample-driven refinement* is applied to generate a new abstract program without the spurious path. SLAM relies on theorem proving, to detect spurious error paths, and give hints to refine the abstraction.

It is well known that discharging the proof obligations arising in software verification reduces to the problem of proving the unsatisfiability of a first-order formula ϕ with a complex Boolean structure modulo a background theory. For example, ϕ can encode an execution path e of a program and the theory can specify its domains of computation. In this situation, the unsatisfiability of ϕ modulo the theory implies the unfeasibility of e. **haRVey**—the prototype we are developing—is a system based on a combination of BDD (for the propositional reasoning) and superposition theorem proving (for the first-order equational reasoning) and has been designed to be open and easily embeddable in larger verification systems in order to test our ideas in a flexible way.

The goals of our research are two. First, we will perform theoretical investigations of various combinations of propositional and first-order satisfiability checking so to automate the theorem proving activity required to solve a large class of program analysis problems. Second, we will perform experimental investigations to make such techniques scale up significantly, so that real programs (of thounds or even millions of lines) can be precisely analized. In fact, the actual state-of-the-art of program analysis techniques shows a dramatic trade-off between precision and performance; for example, there exist pointer analysis techniques capable of handling huge programs whose results are not very informative, see e.g. [80] for a discussion on this issue.

5. Software

5.1. CASRUL

Key words: *Protocols*, *cryptography*, *verification*.

Participants: M. Bouallagui, Y. Chevalier, M. Rusinowitch, M. Turuani, L. Vigneron.

CASRUL[2] is a system that encompasses several translation and constraint solving techniques for automating protocol analysis in Dolev-Yao model and some extensions. Protocol specifications as they can be found in white papers are compiled by the **CASRUL** system and then are passed on decision procedures for checking whether they are exposed to flaws.

The **CASRUL** compiler performs a static analysis to check the executability of the protocol (i.e. whether each principal has enough knowledge to compose the messages he is supposed to send), and then compiles the protocol and intruder activities into an Intermediate Format based on first-order multiset rewriting. The Intermediate Format unambiguously defines an operational semantics for the protocol. Afterwards, different translators can be employed for translating the Intermediate Format into the input language of different analysis tools. The Intermediate Format can be also generated in a typed mode (the untyped one is the default), which leads to smaller search spaces at the cost of abstracting away type-flaws from the protocol. This compiler is the front-end of several verification tools designed in the context of the European projects AVISPA.

The detection of flaws in a hostile environment can be reduced to solving symbolic constraints in the message space. The intruder should send messages that comply with the protocol specification in order to

[2] http://www.loria.fr/equipes/cassis/softwares/casrul/

get undetected by honest agents. After the last protocol step, the intruder should be able to derive the secret from the replies sent by the honest agents (and possibly using some initial knowledge).

The **CASRUL** implementation of these symbolic constraints solving is efficient and has been successful with many security problems. Among the 51 protocols in the Clark/Jacob library [73], 46 can be expressed in the tool specification language. Among the 51 protocols 35 are flawed, and **CASRUL** can find a flaw in 32 of them, including a previously unknown type confusion attack in Denning Sacco Protocol.

5.2. BZ-Testing-Tools

Key words: *Test generation*, *formal specification*, *animation of specifications*.

Participants: F. Ambert, F. Bouquet, S. Guenaud, B. Legeard, F. Peureux, N. Vacelet.

BZ-Testing-Tools[3] (BZ-TT, for short) is a tool-set for animation and test generation from B, Z and State-chart specifications. BZ-Testing-Tools provides several testing strategies (partition analysis, cause-effect testing, boundary-value testing and domain testing), and several test model coverage criteria (multiple condition coverage, boundary coverage and transition coverage). The tool-set is composed of three graphic user interfaces, for animation, test generation and reification. Animation is used to validate the model. Test generation is used to define the test to validate. Reification is used to translate abstract tests into concrete executable scripts.

5.3. haRVey and related tools

Key words: *Automated deduction*, *saturation theorem proving*, *satisfiability*, *equational theories*, *boolean reasoning*, *BDDs*.

Participants: J.-F. Couchot, D. Déharbe, A. Giorgetti, S. Ranise [correspondent].

haRVey[4] is a theorem prover for first-order logic with equality [39]. Its main feature is the capability of behaving as a decision procedure for the problem of checking the validity of certain classes of (ground) formulae modulo some theories of relevance in verification such as lists, arrays, and their combinations. The system is based on a combination of BDDs to compactly represent the boolean structure of formulae and superposition theorem proving to efficiently and flexibly reason in many first-order equational theories. If a formula is not a logical consequence of a theory, then **haRVey** returns a set of literals from which a counter-example can be built.

The system has been especially designed to be integrated in larger verification systems, such as interactive theorem provers, symbolic simulators, and verification condition generators. The following tools (developed within the CASSIS project) have been combined with **haRVey**:

- **bam2rv** takes a B abstract machine containing an invariant clause and generates proof obligations encoding the fact that the invariant is inductive (for details, see [38][58]).
- **rvqe**[5] performs some transformations on a validity problem so that **haRVey** can process it.

Also the tool *Why*[6] (developed by J.-C. Filliâtre of LRI, Université Paris Sud, Orsay) can generate proof obligations for our system to check the correctness of ML and C programs.

We have successfully applied **haRVey** (in combination with the tools listed above) to the verification and the debugging of imperative programs [48][42] as well as of B specifications [38].

[3] http://lifc.univ-fcomte.fr/~bztt
[4] http://www.loria.fr/equipes/cassis/softwares/haRVey/
[5] http://lifc.univ-fcomte.fr/~couchot/rvqe/
[6] http://why.lri.fr/

5.4. CLPS

Key words: *Constraint Logic Programming*, *Set constraints*, *Solver*.

Participants: F. Ambert, F. Bouquet, S. Guenaud, B. Legeard, F. Peureux, N. Vacelet.

CLPS [19] is a set constraint solver dedicated to the evaluation of set-oriented formal specifications. Its constraint domain uses Homogeneous Hereditarily Finite Set Universe with set operators and relations ($\in, \notin, \subseteq, =, \neq$, etc) and allows mapping functions (total or partial injection, surjection or bijection) and relations. CLPS features customized partial consistency techniques using specific inference rules in order to maintain arc consistency.

5.5. daTac

Key words: *Automatic deduction*, *first-order logic*, *equational logic*, *associative-commutative theories*, *symbolic constraints*.

Participant: L. Vigneron [correspondent].

The **daTac** [88] system (*Déduction Automatique dans des Théories Associatives-Commutatives*) is a software for theorem proving and completion in associative and commutative theories, in first-order logic. The deduction techniques implemented are combining selection strategies, deduction steps, redundant information deletion, and symbolic constraints. **daTac** is documented, maintained and accessible on the web (version 0.94).[7] Recent modifications have been done for giving a better control of deduction and simplification strategies. **daTac** is intensively used in the CASSIS group for verifying cryptographic protocols.

5.6. SPIKE

Key words: *Induction*, *consistency of specifications*, *completeness of function definitions*.

Participants: S. Stratulat, M. Rusinowitch [correspondent].

The **SPIKE** system is a software verification tool. It is able to perform proofs by induction, consistency checks of specifications and completeness tests of function definitions. **SPIKE** provides heuristics for the generation of test sets and for the automatic generation of lemmas. It also allows for minimal user interaction by automatizing the easy inference steps in the proofs. A reasonable number of examples have been automatically treated by **SPIKE** with a lower interaction degree than other similar systems (for example Gilbreath card trick, Ramsey theorem, binomial theorem, verification of digital systems).

A new prototype (written in OCAML), integrating proof strategies and decision procedures, is under development by Sorin Stratulat. It has been already successfully applied to the validation of the JavaCard platform by the LEMME project and to the design to an XML collaborative editor and a file synchronizer by the ECOO project.

SPIKE is available on the web.[8]

6. New Results

6.1. Automated deduction

6.1.1. Decision procedures and their extensions

Key words: *Propositional and first-order satisfiability*, *quantifier instantiation*, *integration*, *combination of decision procedures*.

Participants: F. Bellegarde, D. Déharbe, S. Ranise, M. Rusinowitch, L. Vigneron.

[7] http://www.loria.fr/equipes/cassis/softwares/daTac/
[8] http://www.loria.fr/equipes/cassis/softwares/spike/

In many cases, applying software analysis techniques reduces to the problem of proving the unsatisfiability of a formula of first-order logic with a complex Boolean structure modulo a background theory. So, we have considered the problem of checking that a Boolean combination of ground literals is satisfiable modulo a theory. Our technique is based on a simple yet efficient combination of BDDs and superposition theorem proving. The idea is to abstract ground atoms to propositional letters and then let BDDs represent the Boolean structure of (an abstraction of) the formula. Since it is easy to extract the Disjunctive Normal Form (DNF) of ϕ_g from its BDD representation, we check the satisfiability modulo the theory of each disjunct in the DNF by invoking a superposition theorem prover. A refinement of this schema which dramatically improves performances (based on the capability of generating suitable lemmas to *prune* the BDD) has also been devised. In order to build procedures which check for satisfiability modulo a certain theory, we have adopted our work in [16]. This allows us the flexible implementation of many decision (and semi-decision) procedures by simply feeding a superposition theorem prover with the axioms of the theory and the literals to be proved satisfiable modulo the theory [48]. It is also an efficient alternative to specialized decision procedures as described in [63].

We have also considered the problem of reducing the satisfiability of a first-order formula ϕ (possibly containing quantifiers) modulo a theory $\mathcal{T}$ to the satisfiability of a ground formula ϕ_g modulo a theory $\mathcal{T}'$ s.t. ϕ is satisfiable modulo $\mathcal{T}$ iff ϕ_g is satisfiable modulo $\mathcal{T}'$ and $\mathcal{T}$ is contained in $\mathcal{T}'$. This can be seen as a pre-processing step so that the technique developed for ground formulae can be applied also to non-ground ones [39].

The efficiency of the proposed algorithms is shown by comparing an implementation of our techniques in the system **haRVey** with the state-of-the-art theorem prover *Simplify* (used in the ESC/Java tool [79]) on a set of benchmarks extracted from the debugging and verification of C functions manipulating pointers. The results show that our techniques scale better and always return correct results (*Simplify* may return false negatives given its heuristic nature). For details of the experiments see [39].

Another viewpoint on decision procedures is to generalize the congruence closure algorithm with the aim of building a compact representation of an equational theory. In collaboration with Stony Brook University, NY, we have used this technique to define rewriting based decision procedures which are crucial ingredients in developing efficient automated reasoning systems [17]. This work can also be seen as a rational reconstruction of previous results in the literature and allows us to understand the relationships between such results (Nelson et Oppen [82], Downey, Sethi, and Tarjan [76], Shostak [86]). Furthermore, all the results carry over to associative-commutative operators by using combination algorithms for the completion of theories having disjoint signatures so to generate a convergent term rewriting system on an extended signature. This approach can also be used to solve word problems for theories containing associative-commutative symbols without resorting to simplification orderings which must be compatible with associative-commutative operators [64]. With this approach, we plan to investigate different approaches to fragments of set theory so to extend the techniques used in the CLPS system. In particular, we will try to obtain modularity results for combining decision procedures for different theories sharing some symbols. We will also try to propose techniques to smoothly integrate decision procedures within the general strategies of the automated theorem provers.

In [50][60], we are also investigating an alternative approach to combining decision procedures whereby key ingredients of the Shostak schema (i.e. the canonizer and the solver) are extended and used to refine the Nelson-Oppen approach for certain classes of theories. This work is being carried over in D.-K. Tran's thesis work. In this context, we plan to implement our ideas in **haRVey** so to test their practical value as well as to extend the system to reason about theories which are difficult, if not impossible, to handle by superposition (e.g. Presburger arithmetic). Finally, in [42], we have investigated the integration between the Nelson-Oppen combination schema and the activity of instantiating quantifiers so to automatically prove formulae containing quantifiers in the presence of a combination of theories.

6.1.2. Symbolic constraint solving

Key words: *Unification*, *constraint solving*.

Participant: M. Rusinowitch.

Equational unification problems are central in automated deduction. In collaboration with Siva Anantharaman (LIFO, Orléans) and Paliath Narendran (SUNY, Albany) we have considered unification modulo theories that extend the well-known ACI (associative, commutative, idempotent) by adding a binary symbol '$*$' that distributes over the ACI-symbol '$+$'. If this distributivity is one-sided we get a DEXPTIME-complete unification problem [29]. If '$*$' is distributive on both side over '$+$', we get a theory denoted $ACID$; we show unification modulo $ACID$ to be NEXPTIME-decidable and DEXPTIME-hard [28]. These theories seem to be related to the analysis of programs modeled in terms of process algebras and to the theory of set constraints. We have also proved that modulo the theory which adds on to $ACID$ the assumption that '$*$' is associative-commutative unification is undecidable.

6.1.3. Verification of collaborative editors

Key words: *Collaborative*, *editor synchronization*, *consistency*, *proof*.

Participants: A. Imine, M. Rusinowitch.

Collaborative Editors allow a group of users to edit a shared document at the same time from physically dispersed sites that are interconnected by a supposed reliable network. The interactive nature of Collaborative Editors requires that the effect of a user's action be seen by the local user and distant users in a timely way. Therefore, the state-of-the-art collaborative Editors usually *replicate* the shared document at each site and use Operational Transformation (OT) algorithms to achieve convergence copies of cooperating sites and high responsiveness. OT algorithms is based on optimistic concurrency control: local operations are executed immediately on the local replica of the document without being nondeterministically blocked or delayed as in many pessimistic schemes. On the other hand, remote operations are executed after being *transformed* against those that have been executed and that are concurrent. Using these algorithms, however, imposes the verification of convergence conditions. Verifying these conditions is often difficult – even impossible – to produce by hand.

In this respect, the objective of this work is to study the convergence problem in order to propose a formal framework for developing correct OT algorithms. This work is developed in collaboration with the ECOO project.

Initially, we have constructed a transition system from a Collaborative Editor, where the transition steps were expressed as rewrite rules. In order to verify the convergence conditions from this specification we have used the **SPIKE** prover. In that way we have detected bugs in several Collaborative Editors designed by the specialists of the domain [40].

To simplify the verification task, we have developed a prototype tool, **VOTE** (Validation of Operational Transformation Environment), for automatically checking convergence conditions [43][40]. The input of our tool consists of a formal specification written in algorithmic style. The tool compiles the description in conditional equations that can be analyzed by **SPIKE**.

Furthermore, we have proposed to use the transformational approach for building a generic and safe synchronizer [46]. This one forces convergence in all cases and reconciles divergent data at all possible levels of granularity. The synchronizer resolves *all* conflicts automatically without delegating them to users or the administrator. Users can compensate system decisions after synchronization completion.

6.2. Security protocol verification

Key words: *Protocol*, *security*, *verification*, *exclusive-or*, *exponentiation*.

6.2.1. Extension of the Dolev-Yao model

Participants: M. Bouallagui, Y. Chevalier, M. Rusinowitch, J. Santiago, M. Turuani, L. Vigneron.

In the domain of protocol verification, we have obtained this year the first results relaxing the *perfect cryptography* assumption [35]. This hypothesis was a limitation on most formal protocol analysis techniques. We have also studied security for a model allowing an infinite number of sessions for finite messages. Finally we have enhanced a validation technique based on over-approximations of the intruder knowledge.

We have designed an algorithm for deciding the insecurity of cryptographic protocols in presence of the standard Dolev-Yao intruder (with a finite number of sessions) extended with so-called oracle rules, i.e. deduction rules that satisfy certain conditions. As an instance of this general framework, we obtain that protocol insecurity is in NP for an intruder that can exploit the properties of the exclusive **or** operator. This operator is frequently used in cryptographic protocols but cannot be handled in most protocol models. An immediate consequence of our proof is that checking whether a message can be derived by an intruder (using exclusive **or**) is in PTIME. We also apply our framework to an intruder that exploits properties of certain encryption modes such as *cipher block chaining*.

By taking a different oracle rule we have obtained an NP decision procedure for the formal analysis of protocols in presence of modular exponentiation with products allowed in exponents [36]. Unlike other works, the number of factors that may appear in the products is unlimited. This procedure is quite useful to analyze protocols based on Diffie-Hellman technique. For instance we have shown that our model is powerful enough to uncover known attacks on the Authenticated-Group Diffie-Hellman.2 protocol suite.

We have proposed a protocol model which integrates two different ways of analyzing cryptographic protocols: i) analysis w.r.t. an unbounded number of sessions and bounded message size, and ii) analysis w.r.t. an a priori bounded number of sessions but with messages of unbounded size. We have shown that in this model secrecy is DEXPTIME-complete [34]. This result is obtained by extending the Dolev-Yao intruder to simulate unbounded number of sessions.

6.2.2. Intruder knowledge approximation

Participants: Y. Boichut, G. Cécé, P.C. Héam, O. Kouchnarenko.

We have investigated the tree automata method by Genet and Klay for the validation of protocols in collaboration with Dublin City University and based on the tool Is2Tif they have developed for Isabelle prover syntax. The idea is to over-estimate the intruder knowledge by using regular tree languages. This method allows one to show that some states are unreachable, and hence that the intruder will never be able to know certain terms. Regular tree-languages can be used here to effectively model the knowledge that the intruder might have acquired from previous sessions.

A translator to ISABELLE -Is2Tif input syntax- has been developed in order to link **CASRUL** together with Is2Tif. This work gives to **CASRUL** the capability to verify security protocols with an unbounded number of sessions. Furthermore the class of verifiable protocols w.r.t. Is2Tif has been extended. We intend to develop a **CASRUL** package based on the intermediate format and using the principles of Is2tif.

6.2.3. On-line intrusion detection

Participants: T. Abbes, M. Rusinowitch.

Intrusion Detection Systems are becoming necessary tools for system administrators to protect their network. However they find more and more difficulties with high speed networks. To enhance their capacity and deal with evasion techniques, frequently used by hackers, we have introduced a new method to filter the network traffic. The detection method, while being stateful, processes each packet as soon as it is received. We have employed this strategy after a new classification of detection rules. Then, we have used efficient multi-search methods and suitable data structure for signatures. The method has been successfully implemented as an extension of the Intrusion Detection System "Snort" [25][24].

Nevertheless, IDS are still plagued with false positive alerts, letting security professionals to be overwhelmed by the analysis tasks. To overcome this problem, we have proposed a combination of pattern matching and protocol analysis approaches. While the first method of detection relies on a multi pattern matching strategy, the second one benefits from an efficient decision tree adapted to the network traffic characteristics.

6.3. Reachability analysis

6.3.1. Reachability analysis for testing

Key words: *Set-theoretic specification*, *boundary test*, *first-order logic*.

Participants: J.-F. Couchot, D. Déharbe, A. Giorgetti, S. Ranise.

Given a B model of an (already implemented) industrial system, the software called *BZ-Testing-Tools* automates the generation of functional boundary tests. However, it follows a process that does not guarantee their reachability. We plan to add a component to check whether these testing objectives are reachable or not. The set-theoretic specifications built at LIFC are large, but their reasonably simple structure suggest us to treat the reachability question with automatic proving techniques like those implemented in **haRVey**.

The work described in [74] aimed to bridge the gap between this need with the techniques developed for infinite systems (such as abstraction, acceleration, ...). It defines a reduced language for models and properties to check and details how to translate a B abstract machine into this language. Algorithms are given to build a symbolic representation of reachable states, through an abstraction mechanism. The toy example of a process scheduler is work out entirely to illustrate the method. As a prerequisite to reachability analysis, we showed how to prove the correctness of a class of B specifications by checking their invariant [38]. When the proof fails, the question arises to check the reachability of a counter-example exhibited by some model finder. This question brings us back to our initial task.

This work continues within the Ph. D. thesis of J.-F. Couchot, strengthening the cooperation between the *BZ-Testing-Tools* and **haRVey**.

6.3.2. Reachability computation for parametric systems

Key words: *Parametric systems*, *regular languages*, *reachability*.

Participants: F. Bellegarde, C. Charlet, G. Cécé, P.-C. Héam, O. Kouchnarenko, Y. Mainier.

Regular languages can be used for reachability analysis. We use this language-based model in two directions.

In the first direction, we propose a verification approach for a class of parameterized systems which are composed of an arbitrary number of similar processes. For that, in C. Charlet's thesis [10], states of symbolic transition systems are represented by regular languages and the transitions by transducers over regular languages. If a symbolic model of the systems can be computed by acceleration of the actions, then we can also verify safety and liveness properties, and a refinement relation $\mathcal{R}$ between these symbolic models.

In [30][52], we present a refinement verification for this class of parameterized systems. The verification of the refinement relation $\mathcal{R}$ by Regular Model Checking [61] is done on transition systems where states are regular languages and transitions are labelled by accelerated actions. We show that, under some conditions, if $\mathcal{R}$ is verified between two symbolic models, then refinement is verified between concrete parameterized systems without accelerated actions. Then, we can take advantage of the property preservation by refinement for their verification since the model checking verification at the abstract level is more likely to terminate. We have implemented a prototype tool to verify the refinement relation for parameterized systems. Our tool is based on the $LASH$[9] libraries to represent automata and on the RMC[10] tool to compute transducers acceleration. Presently, our tool generates symbolic reachability graphs, verifies refinement between two symbolic transition systems and also verifies safety properties for these systems. When refinement is not verified, our tool finds the origin of the error. We applied our tool to verify automatically the refinement for abstract and refined specifications of parameterized systems like the PID [75] and an elevator. We must however complete this framework by adding the verification of safety and liveness properties.

The second direction is for computing the image of a rational language by the transitive closure of a relation. This is a central question in a Regular Model Checking. In a recent paper [69], Bouajjani, Muscholl and Touili proved that the class of rational languages L of the form $\cup L_0 L_1 L_1 ... L_k$ is closed under all semi-commutation relations R, where the union $\cup$ is finite and the L_i's are either letters or of the form B^* with B a subset of the alphabet. Moreover, a recursive algorithm on the regular expression allows computing $R^*(L)$. We developed a more efficient algorithm to compute $R^*(L)$ [53]. Moreover, we answer an open question asked in the Bouajjani and al.'s paper by giving a larger class of rational languages closed under union, intersection,

[9] Available at http://www.montefiore.ulg.ac.be/~boigelot/research/lash/.
[10] Available at http://www.regularmodelchecking.com/.

semi-commutative relations and conjugacy. Our algorithm is also effective for this new class. This approach has been implemented in OCAML. The experimental tests we have done so far are conclusive.

In Y. Mainier's current thesis work, we are investigating the synchronization of parametric systems using a Presburger formula. The challenge is to be able to limit the problem sufficiently to obtain acceleration semi-algorithms. Accelerations and regular model checking are both based on inclusion between regular languages so another challenge is the decidability of the inclusion in the considered class of languages. The study of some toys examples seems to answer both requirements. We now have to look for the formal definition of the corresponding class of applications.

6.3.3. Test case and test driver generation from a formal model using constraint solving

Key words: *Model-Based Testing*, *Formal Specifications*, *Test Case Generation*, *test Driver Generation*.

Participants: F. Ambert, F. Bouquet, S. Guenaud, B. Legeard, F. Peureux, N. Vacelet.

The need to offer better methods and tools for functional black-box testing of large scale system has given rise to a large amount of research on generating tests from formal specifications. The BZ-TT approach is based on an original method of boundary-value extraction and preamble computation based on a customized constraint logic programming technology. This method has been validated on several real-size industrial applications. The new research directions that we follow concern various research challenges.

Firstly, we have addressed the problem of mastering the test number explosion. To do that, we formalize different model coverage criteria to allow the test engineer to choose the level of model coverage he/she wants during test generation.

Secondly, we have tried to improve search algorithm during the preamble calculus: we are studying a backward-chaining algorithm using the constraint solver CLPS [37].

Thirdly, we have introduced the Generation of Test Drivers from the abstract generated test cases [31][59].

Finally, we currently work to improve the customized set constraint solver CLPS [19][18] in order to complete the data-structure and to optimize the resolution.

6.3.4. Approximation of linear transition systems

Key words: *Linear transitions systems*, *model-checking*, *acceleration rules*.

Participants: J. Musset, M. Rusinowitch.

We have proposed in [12][47] a construction for approximating the reachability sets of linear transition relations. Our approach takes into considerations the asymptotic behavior of the dynamic system depending on the eigenvalues and the dimension of the characteristic space associated. Compared to previous works, we improve the class of transition relations for which we can construct a meta-transition. Finally, we use polynomials to represent these meta-transitions. We have illustrated this method on a train controller specified in Lustre.

7. Contracts and Grants with Industry

7.1. Industrial contracts

This project is validated by industrial case studies. These case studies help to identify the problem in the verification of infinite system. During 2003, we have worked in three new industrial case studies:

- Schlumberger e-city (Besançon) with MagIC500, bank paid terminal (january 2002 to September 2003),
- Schlumberger Smart Cards (Montrouge) key management (October 2002 to October 2003),
- PSA with interfaces for the car industry (November 2002 to October 2003).

7.2. RNTL

In the area of structural testing using constraint solving, LIFC has been working in RNTL INKA "Génération automatique déterministe de données de tests selon des critères de couverture structurelle" driven by THALES division Systèmes Aéroportés with the laboratories I3S/Nice (Michel Rueher) et LSR/Grenoble (Farid Ouabdessalam). It is a pre-project financed by Industry Ministry with 780 KF to LIFC 2001/2003.

7.3. ANVAR

The French government-organization ANVAR supports the development of the tool-set BZ-Testing-Tools with 500Keuros from September 2001 to October 2003. This funding was given in order to create a new company, Leirios Technologies, able to commercialize and to transfer to the market the BZ-Testing-Tools technology.

7.4. IST AVISPA

AVISPA[11] is a shared-cost RTD (FET open) project, funded by the European Commission under the Information Society Technologies Programme operating within the Fifth Framework Programme, started on January 1st, 2003. The participants are: Mechanized Reasoning Group at DIST, Università di Genova (Genova, Italy), CASSIS project at INRIA, Information Security Group at ETHZ (Zürich, Switzerland) and Siemens AG (Munich, Germany).

AVISPA aims at developing a push-button, industrial-strength technology for the analysis of large-scale Internet security-sensitive protocols and applications. This technology will speed up the development of the next generation of network protocols, improve their security, and therefore increase the public acceptance of advanced, distributed IT applications based on them. A central aim of the project is to integrate this technology into a robust automated tool, tuned on practical, large-scale problems collected from IETF drafts, and migrated to standardization bodies.

7.5. INTERREG Test-UML

In the European Interreg III project (Suisse - Franche-Comté), LIFC is a member of the TestUML project with the École Polytechnique Fédérale de Lausane - EPFL - concerning Test generation from UML/OCL formal models. The duration of the project is 2 years and it was started in November 2002.

8. Other Grants and Activities

8.1. International grants

- SECURYPTO,[12] a new France-Quebec research network between LIFO (Orléans), IMAG (Grenoble), LORIA (Nancy), UQAM (Montréal) investigates the specification and verification of security properties in process algebras, using notions and techniques from: rewriting, constraints, information flow and interference, process localities and performance.
- We (Nancy) are collaborating with David Deharbe of Department of Computer Science and Applied Mathematics, of UFRN, Federal University of Rio Grande do Norte (Natal, Brazil) on the development of the system **haRVey** and its application to various verification problems.

[11] http://www.avispa-project.org/
[12] http://www.hains.org/security.html

8.2. National grants

- We have been invited by the members of the INRIA Research action (ARC) ModoCop to participate to their activities. The goal of this action is to verify object-oriented concurrent programs.[13]
- ACI V3F—"Validation & Verification of programs with floating-point numbers".
  The goal of this project is to provide tools to support the verification and validation process of programs with floating-point numbers. More precisely, V3F will investigate techniques to check that a program satisfies the calculations hypothesis on the real numbers that have been done during the modelling step. The underlying technology is based on constraint solving.
  Partners:
  - I3S-INRIA Sophia Antipolis - Michel Rueher,
  - IRISA-INRIA Vertecs & Lande - Thierry Jeron & Arnaud Gotlieb,
  - CEA-LIST - Bruno Marre.

  Duration: 3 years.
  Starting: October 2003.
- ACI EDEMOI—"Formal Modelling and Verification of Airport Security"
  The EDEMOI project aims at defining an approach for the construction and analysis of a precise reference document that models and structures current standards and associated recommendations. The exploitation of this model by the civil aviation authorities will improve airport security.
  Partners:
  - LSR-IMAG - Yves Ledru,
  - CEDRIC-CNAM - Veronique Donzeau-Gouge,
  - ONERA Toulouse - Michel Lemoine,
  - GET ENST Paris - Sylvie Vignes.

  Duration: 3 years.
  Starting: October 2003.
- ACI Securité GECCOO—"Génération de code certifié pour des applications orientées objet (Spécification, raffinement, preuve et détection d'erreurs)".
  This project aims at developing methods and tools for the design of object-oriented systems that require a high degree of security. The methods and tools will be developed to be integrated, *i.e.* together they will form a coherent design method from specification to certified code generation using refinement, simulation, testing and verification techniques. In particular, the project focuses on the design of smart card applications, written in a subset of Java (like JavaCard), annotated with JML specifications. Several tools exist which manipulate JML annotated programs. However, experiences with using JML for industrial applications also have revealed several of its shortcomings which this project proposes to attack.
  The teams collaborating in this project (Équipe TFC (LIFC), Projet Lemme (INRIA), Projet LogiCal (LRI), Équipe VASCO (LSR)) have complementary skills in the domains of security, modeling object oriented programs, and interactive and automatic program verification.
  Duration: July 2003 - July 2006.

[13] http://www-sop.inria.fr/lemme/modocop/

- ACI Cryptologie VERNAM—"Decidable subclasses of cryptographic protocols"
  Academic partners: University of Provence (R. Amadio and D. Lugiez) and ENS Cachan (H. Comon and J. Goubault-Larrecq)
  Duration: fall 2000 - fall 2003.
- We are members of the working group *Vérification des systèmes à nombre infini d'états* of the GDR *Spécifications, Preuves et Tests*, directed by A. Bouajjani et A. Finkel.
- We participate to the working group *Sémantiques Logiques et Opérationnelles, Vérification et Optimisation* (SLOVO) of GDR *Spécifications, Preuves et Tests*.
- With respect to the *Contrat de Plan État-Région Lorraine 2000-2006*, we are working in the *Pôle de Recherche Scientifique et Technologique Intelligence Logicielle* within the theme: - Qualité et sûreté des logiciels et systèmes informatiques - with the action VALDA (2002-2003).

8.3. International grants

- PAI PROCOPE: Combining automata-theoretic and rewriting techniques for the analysis of cryptographic protocols. The participants are the CASSIS project and the team of professor Thomas Wilke, Institute of Computer Science and Applied Mathematics, Christian-Albrechts-University of Kiel.
  The aim of this project is to combine the rewriting approach, pursued by the French partner, and the automata-based approach, studied on the German side, in order to develop algorithms and tools for the automated analysis of a class of protocols containing many real-world protocols that are out of the scope of current methods and tools.
- We collaborate with SUP'COM (École Supérieure des Communications de Tunis) with A. Bouhoula, on the formal verification for telecommunication software.
- In the area of automated test generation from a formal model, we have an active collaboration with Dr Mark Utting from the Formal Method group from the University of Waikato.[14] This cooperation is supported by the France-New-Zealand scientific program.
- In the context of the verification of cryptographic protocols, we are collaborating with D. Sinclair and F. Oehl of Dublin City University, School of Computer Applications, Ireland.

8.4. Individual involvement

- The TFC group is organizing and will host the national conference "Approches Formelles dans l'Assistance au Développement de Logiciel 2004" (AFADL'2004) on June 16–18, 2004.
- Francoise Bellegarde: director of the research team *Techniques formelles et à contraintes (TFC)* of the laboratory *Laboratoire d'Informatique de l'Université de Franche-Comté (LIFC)*, member of the laboratory board.
- Fabrice Bouquet: elected member in UFR board, elected member in laboratory board. In charge of the Mobilization area (9 research projects, with 46 researcher and 8 laboratories) in ISTI Institute.[15]
- Olga Kouchnarenko: elected member of the laboratory board, Program Committee member of the International Workshop on Formal Methods 2003 (IWFM'03), held in Dublin, Ireland on July 11, 2003.
- Bruno Legeard : elected member of the University of Franche-Comté Scientific Council. Assistant-Director of the LIFC. Reviewer for the national program RNTL (1999 to 2003).

[14] http://www.cs.waikato.ac.nz/Research/fm/index.html
[15] http://www.isti.info

- Silvio Ranise: *Trustee* of the european project CALCULEMUS (Systems for Integrated Computation and Deduction).[16] Local coordinator of the ACI GECCOO. Co-chair of the First Workshop on *Pragmatics of Decision Procedures in Automated Deduction (PDPAR'03)*, affiliated to the *19th International Conference on Automated Deduction (CADE'19)*, held in Miami, Fl. (USA). Coordinator (with Cesare Tinelli) of the Satisfiability Modulo Theories Library (SMT-LIB) initiative.[17] Organizer of the *Theoretical Computer Science Seminar/Séminaire d'informatique fondamentale (SIF)* at LORIA. Web master of the site for the CASSIS project.
- Michaël Rusinowitch: member of the IFIP Working Group 1.6 (Rewriting); member of the steering committee of RTP 19 of CNRS: SECC (Complex or Constrained Embedded Systems); member of the scientific committee of the CRIL (CNRS, Computer Science Laboratory of Lens); Chairman of *Security Protocol Verification*, CONCUR Satellite Workshop, Marseille, september 6, 2003.
  Chairman of IJCAR 2004 (with D. Basin) of International Joint Conference on Automated Reasoning, 04 July - 08 July, 2004, Cork, Ireland. PC member of "Rencontres Sécurité et Architecture Réseaux", SAR 2003, july 2003, Nancy. PC member of Sixth International Workshop in Formal Methods (IWFM'03). Dublin City University, 11 July 2003.
  Editor of a special issue of *Technique et Science Informatiques*, 2003 on computer security.
- Laurent Vigneron: elected member of the LORIA council; member (secretary) of the IFIP Working Group 1.6; web master of the site *Rewriting Home Page*, of the Rewriting Techniques and Applications (RTA) Conference site, and of the web page for the IFIP Working Group 1.6; member of the scientific council of Université Nancy 2; chairman of the conference *FTP: First-Order Theorem Proving*, 2003; member of the FTP steering committee.

8.5. Visits of foreign researchers

- Adel Bouhoula (Professor, Tunisia) visited our group (Nancy) for two weeks in July. The subject of the collaboration is how to detect intruders.
- Ralf Küsters (Christian-Albrechts-Universität, Kiel) visited our group (Nancy) for one week in September. The subject of the collaboration is the verification of security protocols.
- Mark Utting (University of Waikato, New-Zealand) visited LIFC twice in 2003: 2 weeks in June and 2 weeks in November.
- Thomas Wilke (University of Kiel, Germany) visited our group (Nancy) for one week in March. The subject of the collaboration is the verification of security protocols.

8.6. Visits of team members

- Yohan Boichut visited the laboratory "Conception et Réalisation des Application Complexes", University of Montreal, Canada from November 9 to November 29, 2003. This visit is part of the bi-lateral project Securypto between Canada and France.
- Julien Musset has been hosted from April to August 2003 by the University of Karlsruhe (Group of Prof. J. Calmet). He was supported by a CALCULEMUS/Marie Curie Grant.

[16] http://www.calculemus.net
[17] http://combination.cs.uiowa.edu/smtlib

9. Dissemination

9.1. Ph. D. theses

Here follows a list of the Ph. D. theses defended this year:

1. **Célina Charlet**, title: "*Raffiner pour Vérifier des systèmes paramétrés*", supervisors: F. Bellegarde and O. Kouchnarenko, defended on December 19th, 2003.
2. **Yannick Chevalier**, title: "*Résolution de problemes d'accessibilité pour la compilation et la validation de protocoles cryptographiques*", supervisors: M. Rusinowitch and L. Vigneron, defended on December 9th, 2003.
3. **Julien Musset**, title: "*Approximation de relations de transition : application à la vérification de systèmes infinis*", supervisor: M. Rusinowitch, defended on July 15th, 2003.
4. **Mathieu Turuani**, title: "*Sécurite des Protocoles Cryptographiques : Décidabilité et Complexité*", supervisor: M. Rusinowitch, defended on December 11th, 2003.

9.2. Committees

Michael Rusinowitch is a member of the AFIT committee to award the best Ph. D. theses in theoretical informatics of the year.

9.3. Seminars, workshops, and conferences

- Y. Boichut has given a seminar at the University of Montreal, entitled "Une méthode de vérification des protocoles cryptographiques basée sur les automates d'arbres" on November 21, 2003.
- F. Bouquet has attended the following events:
  - Conference AFADL'03 Rennes, 15-17 january,
  - INRIA Sophia Antipolis (ARC Modocop), 17 April,
  - Conference FME'03 Pisa (Italia), 8-13 September,
  - IRISA Rennes (ARC Modocop), 17 September,
  - Software Testing workshop CNAM Paris, 16 October,
- Y. Chevalier has attended the following events:
  - Logic in Computer Science, Ottawa (Canada), June;
  - FST-TCS (Bombay, India), December;
  - AVISPA meetings: January (Nancy), March (Zurich), May (Nancy), July (Munich), October (Zurich);
  - VERNAM meeting, (Grenoble), March;
  - seminar at Montreal Polytechnic (group headed by John Mullins), entitled "Complexité de l'étude d'un nombre non-borné de sessions", June;
  - seminar at Concordia University (group headed by Sofiène Tahar (HVG)), entitled "An NP Decision Procedure for Protocol Insecurity with XOR", June.
- F. Bouquet and A. Giorgetti have presented two alternative B specifications of an electronic purse at Modocop Workshop on Smart Card Specification, Verification and Testing, at INRIA Rhône-Alpes, December 4, 2003.

- G. Cécé has participated in regular meetings with G. Sutre and A. Finkel at the LSV of Cachan, Paris during 2003.
- J.-F. Couchot and A. Giorgetti have taken part in the B day organized by the B group of GDR ALP at CNAM, 20 november 2003. They presented a talk entitled “Preuve automatique de (certaines) machines B”.
- F. Dadeau has attended FATES (Formal Approach to Testing of Softwares) workshop and the ASE (Automated Software Engineering) conference in Montréal, 6-10 October 2003. He presented the BZ-TT tool-set during the demo session of ASE.
- O. Kouchnarenko has attended the following events:
  - VERNAM meeting, March 2003, to give a presentation entitled ”Automatic Approximation for Security Protocols Verification”;
  - Seminar at the LIAFA, May 2003, to give a presentation entitled ”Automatic Approximation for Security Protocols Verification: a Way to Combine Model-Checking and Theorem Proving ”.
- B. Legeard has attended the following events:
  - Software Testing workshop CNAM Paris, 15-16 October 2003,
  - Cartes & IT security Paris - Workshop on new paradigm for testing, 19 November 2003.
- M. Rusinowitch has attended the following events:
  - WFLP’03, 12th International Workshop on Functional and (Constraint) Logic Programming Valencia, Spain, June 12-13, 2003 to give an invited presentation entitled “Automated Analysis of Security Protocols”;
  - Sixth International Workshop in Formal Methods (IWFM’03). School of Computing, Dublin City University, 11 July 2003 to give an invited presentation entitled “Deciding Security of Cryptographic Protocols”;
  - Seminar at the Max Planck Institute, july 2003,
  - Formal Methods Europe 2003, Pisa, september 2003;
  - Logic in Computer Science 2003, Ottawa, june 2003.
- S. Ranise has attended the following events:
  - Workshop on Infinite Systems and Verification of Quantitative Properties (JSIVPQ’03), Grenoble, March 2003;
  - Fourth Workshop on First-Order Theorem Proving (FTP’03), Valencia, Spain, June 12-14, 2003;
  - Eleventh Symposium on the Integration of Symbolic Computation and Mechanized Reasoning (CALCULEMUS2003), Rome, Italy, September 10-12, 2003;
  - International Conference on Software Engineering and Formal Methods (SEFM03), Brisbane, Australia, September 24-26, 2003;
  - Sixth Workshop on Formal Methods, Campina Grande, PB, Brazil, October 12-14, 2003;
  - Meeting to prepare an European project, Grenoble, November 24-25, 2003.

- M. Turuani has attended the following events:
  - Logic in Computer Science 2003, Ottawa (Canada), June 2003;
- L. Vigneron has attended the following events:
  - 14th International Conference on Rewriting Techniques and Applications (RTA'03), Valencia, Spain, June 9-11, 2003;
  - IFIP Working Group 1.6 on Term Rewriting, Valencia, Spain, June 12, 2003;
  - Fourth Workshop on First-Order Theorem Proving (FTP'03), Valencia, Spain, June 12-14, 2003.
- F. Bellegarde, F. Bouquet, F. Dadeau, A. Giorgetti, B. Legeard, and S. Ranise have attended the start-up meeting of the ACI GECCOO in Paris on September 8, 2003.

10. Bibliography

Major publications by the team in recent years

[1] A. ARMANDO, S. RANISE, M. RUSINOWITCH. *Uniform Derivation of Decision Procedures by Superposition.* in « Proceedings of Annual Conference of the European Association for Computer Science Logic (CSL'01) », series Lecture Notes in Computer Science, volume 2142, Springer, L. FRIBOURG, editor, pages 513–528, Paris, France, September, 2001.

[2] A. BOUHOULA, M. RUSINOWITCH. *Implicit Induction in Conditional Theories.* in « Journal of Automated Reasoning », number 2, volume 14, 1995, pages 189–235.

[3] G. CÉCÉ. *Vérification, analyse et approximations symboliques des automates communicants.* Ph. D. Thesis, LSV – URA 2236 CNRS, 61, Av. du Pdt. Wilson; 94235 Cachan Cedex, janvier, 1998.

[4] M. HIBTI, B. LEGEARD, H. LOMBARDI. *Une procédure de décision pour un problème de satisfiabilité dans un univers ensembliste héréditaire.* in « Revue informatique théorique et applications / Theorical Informatics and Applications », number 3, volume 31, 1997, pages 205–236.

[5] F. JACQUEMARD, M. RUSINOWITCH, L. VIGNERON. *Compiling and Verifying Security Protocols.* in « Logic for Programming and Automated Reasoning (LPAR'00) », series Lecture Notes in Computer Science, volume 1955, Springer, A. VORONKOV, M. PARIGOT, editors, pages 131–160, Reunion Island, France, 2000.

[6] B. LEGEARD, F. PEUREUX, M. UTTING. *Automated Boundary Testing from Z and B.* in « Formal Methods Europe (FME 2002) », series Lecture Notes in Computer Science, volume 2391, Springer, L.-H. ERIKSSON, P. LINDSAY, editors, pages 21–40, 2002.

[7] L. VIGNERON. *Positive Deduction Modulo Regular Theories.* in « Proceedings of Computer Science Logic (CSL'95) », series Lecture Notes in Computer Science, volume 1092, Springer, H. K. BÜNING, editor, pages 468–485, Paderborn, Germany, 1995.

Books and Monographs

[8] *FTP'2003, 4th International Workshop on First-Order Theorem Proving.* I. DAHN, L. VIGNERON, editors, series Electronic Notes in Theoretical Computer Science, number 1, volume 86, Elsevier Science Publishers, 2003, http://www.elsevier.com/locate/entcs/volume86.html.

[9] *Pragmatics of Decision Procedures in Automated Reasoning (PDPAR'03).* S. RANISE, C. TINELLI, editors, July, 2003, http://www.loria.fr/~ranise/pdpar03/, Workshop affiliated to the 19th Conference on Automated Deduction (CADE-19).

Doctoral dissertations and "Habilitation" theses

[10] C. CHARLET. *Raffiner pour Vérifier des systèmes paramétrés.* Thèse de doctorat, Université de Franche-Comté, Besançon, December, 2003.

[11] Y. CHEVALIER. *Résolution de problèmes d'accessibilité pour la compilation et la validation de protocoles cryptographiques.* Thèse de doctorat, Université Henri Poincaré, Nancy, décembre, 2003.

[12] J. MUSSET. *Approximation of transition relations. Application to infinite states systems verification.* Ph. D. Thesis, Université Henri Poincaré, Nancy, juillet, 2003.

[13] M. TURUANI. *Sécurité des Protocoles Cryptographiques: Décidabilité et Complexité.* Thèse de doctorat, Université Henri Poincaré, Nancy, décembre, 2003.

Articles in referred journals and book chapters

[14] A. ARMANDO, L. COMPAGNA, S. RANISE. *Rewriting and Decision Procedure Laboratory: Combining Rewriting, Satisfiability Checking, and Lemma Speculation.* in « In Festschrift in Honour of Prof. Joerg Siekmann », series Lecture Notes in Artificial Intelligence, Springer Verlag, 2003, http://www.loria.fr/~ranise/pubs/lnai60.ps.gz, To appear.

[15] A. ARMANDO, S. RANISE. *Constraint Contextual Rewriting.* in « J. of Symbolic Computation », number 1–2, volume 36, 2003, pages 193–216, http://www.mrg.dist.unige.it/~silvio/pubs/ftp-jsc.ps.gz.

[16] A. ARMANDO, S. RANISE, M. RUSINOWITCH. *A Rewriting Approach to Satisfiability Procedures.* in « Journal of Information and Computation—Special Issue on Rewriting Techniques and Applications (RTA'01) », number 2, volume 183, June, 2003, pages 140–164, http://www.loria.fr/~rusi/pub/longcsl01.ps.

[17] L. BACHMAIR, A. TIWARI, L. VIGNERON. *Abstract Congruence Closure.* in « Journal of Automated Reasoning », 2003, To appear. Also available as Technical Report A01-R-266, LORIA, Nancy (France).

[18] E. BERNARD, B. LEGEARD, X. LUCK, F. PEUREUX. *Generation of Test Sequences from Formal Specifications: GSM 11-11 Standard Case-Study.* in « International Journal on Software - Practice and Experience », 2003, Accepted for publication.

[19] F. BOUQUET, B. LEGEARD, F. PEUREUX. *A constraint solver to animate a B specification.* in « International Journal on Software Tools for Technology Transfer », 2003, Accepted for publication.

[20] H. COMON, P. NARENDRAN, R. NIEUWENHUIS, M. RUSINOWITCH. *Deciding the Confluence of Ordered Term Rewrite Systems.* in « ACM Transactions on Computational Logic », number 1, volume 4, January, 2003, pages 33-55, http://www.loria.fr/~rusi/pub/tocl.ps.

[21] A. GIORGETTI. *An asymptotic study for path reversal.* in « Theoretical Computer Science », volume 299, 2003, pages 585–602.

[22] M. RUSINOWITCH, S. STRATULAT, F. KLAY. *Mechanical Verification of an Ideal ABR Conformance Algorithm.* in « Journal of Automated Reasoning », number 2, volume 30, February, 2003, pages 153-177, http://www.loria.fr/~rusi/pub/sorinjar.ps.

[23] M. RUSINOWITCH, M. TURUANI. *Protocol Insecurity with Finite Number of Sessions and Composed Keys is NP-complete.* in « Theoretical Computer Science », volume 299, April, 2003, pages 451-475, http://www.loria.fr/~rusi/pub/tcsprotocol.ps.gz.

Publications in Conferences and Workshops

[24] T. ABBES, A. BOUHOULA. *Détection d'Interaction de Services de Télécommunications.* in « Premier Congrès Francophone MAJECSTIC 03 », octobre, 2003.

[25] T. ABBES, A. BOUHOULA, M. RUSINOWITCH. *Filtrage efficace pour la détection d'intrusions.* in « Conférence Sécurité et Architecture Réseaux, SAR 2003 », Juillet, 2003.

[26] F. AMBERT, F. BOUQUET, B. LEGEARD, F. PEUREUX. *Automated Boundary-Value Test Generation from Specification - Method and Tools.* in « 4th International Conference on Software Testing (4th IC TEST) », pages 52–68, April, 2003.

[27] F. AMBERT, S. CHEMIN, B. LEGEARD. *Intégration de domaines à variables dans un solveur de contraintes ensemblistes.* in « Programmation en Logique avec Contraintes, JFPLC'2003 », volume Hors serie RSTI (Revue des Sciences et Technologie de l'Information), Hermes Lavoisier, M. DUCASSÉ, editor, pages 217–222, Amiens, Juin, 2003, http://lifc.univ-fcomte.fr/~bztt/docs/article/jfplc03ACL.ps.

[28] S. ANANTHARAMAN, P. NARENDRAN, M. RUSINOWITCH. *AC(U)ID-Unification is NEXPTIME-Decidable.* in « Proc. of the Int. Conf. MFCS 2003, Bratislava (Slovak Rep.) », series Lecture Notes in Computer Science, volume 2747, Springer, B. ROVAN, editor, August, 2003.

[29] S. ANANTHARAMAN, P. NARENDRAN, M. RUSINOWITCH. *Unification over ACUI plus Distributivity/Homomorphisms.* in « Proc. of the Int. Conf. on Automated Deduction, CADE-19 », series Lecture Notes in Computer Science, volume 2741, Springer, F. BAADER, editor, pages 442-457, 2003, http://www.loria.fr/~rusi/pub/cade2003.ps.

[30] F. BELLEGARDE, C. CHARLET, O. KOUCHNARENKO. *How to Compute the Refinement Relation for Parameterized Systems.* in « Proc. First Int. ACM & IEEE Conf. on Formal Methods and Models for Codesign (MEMOCODE'2003), Mont St-Michel, France », pages 103–112, June, 2003.

[31] F. BOUQUET, B. LEGEARD. *Reification of Executable Test Scripts in Formal Specification-Based Test Generation: The Java Card Transaction Mechanism Case Study.* in « Formal Methods, FME 2003 », volume

2805, Springer-Verlag, K. ARAKI, S. GNESI, D. MANDRIOLI, editors, pages 778–795, September, 2003.

[32] F. BOUQUET, B. LEGEARD. *Réification de scripts exécutables en génération de tests à partir de spécification formelles: application aux mécanismes de transaction de la Java Card.* in « AFADL'2003 », pages 141–156, Rennes, Janvier, 2003, http://lifc.univ-fcomte.fr/~bztt/docs/article/afadl03.pdf.

[33] F. BOUQUET, B. LEGEARD, N. VACELET. *Un format fédérateur pour l'évaluation de spécifications formelles en programmation logique avec contraintes.* in « Programmation en Logique avec Contraintes, JFPLC'2003 », volume Hors serie RSTI (Revue des Sciences et Technologie de l'Information), Hermes Lavoisier, M. DUCASSÉ, editor, pages 203–216, Amiens, Juin, 2003, http://lifc.univ-fcomte.fr/~bztt/docs/article/jfplc03BLV.ps.

[34] Y. CHEVALIER, R. KÜSTERS, M. RUSINOWITCH, M. TURUANI, L. VIGNERON. *Extending the Dolev-Yao Intruder for Analyzing an Unbounded Number of Sessions.* in « Computer Science Logic (CSL 03) and 8th Kurt Gödel Colloquium (8th KCG) », series Lecture Notes in Computer Science, volume 2803, Springer, M. BAAZ, editor, Vienna, Austria, August, 2003.

[35] Y. CHEVALIER, R. KÜSTERS, M. RUSINOWITCH, M. TURUANI. *An NP Decision Procedure for Protocol Insecurity with XOR.* in « Proceedings of the Logic In Computer Science Conference LICS'03 », pages 261-270, June, 2003, http://www.loria.fr/~chevalie/Research/xor-oracle.pdf, Long version available as Technical Report RR-4697, INRIA, France.

[36] Y. CHEVALIER, R. KÜSTERS, M. RUSINOWITCH, M. TURUANI. *Deciding the Security of Protocols with Diffie-Hellman Exponentiation and Products in Exponents.* in « Proceedings of the Foundations of Software Technology and Theoretical Computer Science FSTTCS'03 », December, 2003, http://www.loria.fr/~chevalie/Research/expo-oracle.pdf, Long version available as Christian-Albrecht Universität IFI-Report 0305, Kiel (Germany).

[37] S. COLIN, B. LEGEARD, F. PEUREUX. *Preamble computation in automated test generation using Constraint Logic Programming.* in « Proceedings of UK-Test Workshop », York, UK, September, 2003, http://lifc.univ-fcomte.fr/~bztt/docs/article/UK-Test03.ps.

[38] J.-F. COUCHOT, F. DADEAU, D. DÉHARBE, A. GIORGETTI, S. RANISE. *Proving and Debugging Set-Based Specifications.* in « Proc. of the 6th Workshop on Formal Methods », UFCG, Campina Grande, PB, Brazil, October, 2003, http://www.loria.fr/~ranise/pubs/rvgo2B.ps.gz, Also to appear in Electronic Notes on Theoretical Computer Science (ENTCS).

[39] D. DÉHARBE, S. RANISE. *Light-Weight Theorem Proving for Debugging and Verifying Units of Code.* in « Proc. of the International Conference on Software Engineering and Formal Methods (SEFM03) », IEEE Computer Society Press., Brisbane, Australiab, September, 2003, http://www.loria.fr/~ranise/pubs/sefm03.ps.gz.

[40] A. IMINE, P. MOLLI, G. OSTER, M. RUSINOWITCH. *Proving Correctness of Transformation Functions in Real-time Groupware.* in « Proceedings of the 8th European Conference on Computer Supported Cooperative Work, 14-18 September 2003, Helsinki, Finland », Kluwer, K. KUUTTI, AL., editors, pages 277-293, 2003.

[41] A. IMINE, P. MOLLI, G. OSTER, P. URSO. *VOTE: Group Editors Analyzing Tool.* in « Proc. of the 4th Workshop on First-Order Theorem Proving (FTP'03), Electronic Notes in Theoretical Computer Science »,

volume 86, Elsevier, I. DAHN, L. VIGNERON, editors, 2003.

[42] A. IMINE, S. RANISE. *Building Satisfiability Procedures for Verification: The Case Study of Sorting Algorithms.* in « Proc. of the International Symposium on Logic-based Program Synthesis and Transformation (LOPSTR'03) », 2003.

[43] A. IMINE, P. URSO. *Automatic Detection of Copies Divergence in Collaborative Editing Systems.* in « Formal Methods for Industrial Critical Systems (FMICS'03), Electronic Notes in Theoretical Computer Science », volume 80, Elsevier, T. ARTS, W. FOKKINK, editors, 2003.

[44] O. KOUCHNARENKO, A. LANOIX. *Refinement and Verification of Synchronized Component-based Systems.* in « Formal Methods, FM 2003 », series Lecture Notes in Computer Science, volume 2805, Springer, K. ARAKI, S. GNESI, M. D., editors, pages 341–358, Pisa, Italy, September, 2003.

[45] O. KOUCHNARENKO, A. LANOIX. *SynCo: a Refinement Analysis Tool for Synchronized Component-based Systems.* in « FM'2003 Tool Exhibition Notes », T. MARGARIA, editor, pages 47–51, Pisa, Italy, September, 2003.

[46] P. MOLLI, G. OSTER, H. SKAF-MOLLI, A. IMINE. *Using the Transformational Approach to Build a Safe and Generic Data Synchronizer.* in « Proceedings of GROUP 2003, ACM 2003 International Conference on Supporting Group Work, November 9-12, 2003, Sanibel Island, Florida, USA », ACM, 2003.

[47] J. MUSSET, M. RUSINOWITCH. *Computing Metatransitions for Linear Transition Systems.* in « FME 2003, 12th International FME Symposium », series Lecture Notes in Computer Science, volume 2805, Springer, pages 562-581, Pisa, Italy, September, 2003.

[48] S. RANISE, D. DÉHARBE. *Applying Light-Weight Theorem Proving to Debugging and Verifying Pointer Programs.* in « Proc. of the 4th Workshop on First-Order Theorem Proving (FTP'03) », series Electronic Notes in Theoretical Computer Science, number 1, volume 86, Valencia, Spain, May, 2003, http://www.elsevier.nl/gejng/31/29/23/135/23/show/Products/notes/index.htt#010.

[49] S. RANISE. *Building Convex Hulls by Combining SAT Solving and Algebric Computing.* in « Proceedings of the 11th Symposium on the Integration of Symbolic Computation and Mechanized Reasoning (CALCULEMUS2003) », Rome, Italy, September, 2003, http://www.loria.fr/~ranise/pubs/calculemus2003.ps.gz.

[50] S. RANISE, C. RINGEISSEN, D. TRAHN. *Rule-Based Algorithms for Combining Nelson-Oppen Theories and Shostak Theories.* in « Proc. of the Third Workshop on Cooperative Solvers in Constraint Programming (Satellite Event to CP'2003) », Kinsale, County Cork, Ireland, September, 2003.

[51] M. RUSINOWITCH. *Automated Analysis of Security Protocols.* in « Proceedings of the 12th International Workshop on Functional and (Constraint) Logic Programming, WFLP'03 », number 3, volume 86, Electronic Notes in Theoretical Computer Science, G. VIDAL, editor, June, 2003, http://www.elsevier.com/locate/entcs.

Internal Reports

[52] F. BELLEGARDE, C. CHARLET, K. KOUCHNARENKO. *Using Acceleration to Compute Parameterized System Refinement.* Research Report, number RR-4716, INRIA, January, 2003, http://www.inria.fr/rrrt/rr-4716.html.

[53] G. CÉCÉ, P.-C. HÉAM, Y. MAINIER. *Efficiency of Automata in Semi-Commutation Verification Techniques.* Research Report, number RR-5001, INRIA, December, 2003, http://www.inria.fr/rrrt/rr-5001.html.

[54] I. DAHN, L. VIGNERON. *FTP'2003, 4th International Workshop on First-Order Theorem Proving.* Technical report, number DCSI-II/10/03, Universidad Politécnica de Valencia, Valencia, Spain, 2003.

[55] O. KOUCHNARENKO, A. LANOIX. *Refinement and Verification of Synchronized Component-based Systems.* Technical report, number 4862, INRIA Research Report, Juin, 2003, http://www.inria.fr/rrrt/rr-4862.html.

[56] P. MOLLI, G. OSTER, H. SKAF-MOLLI, A. IMINE. *Safe Generic Data Synchronizer.* Research Report, number A03-R-062, LORIA, Nancy (France), May, 2003, http://www.loria.fr/publications/2003/A03-R-062/A03-R-062.ps.

Miscellaneous

[57] Y. BOICHUT. *Combinaison d'approches pour la vérification de protocoles cryptographiques.* Rapport de DEA, Laboratoire d'Informatique de l'Université de Franche-Comté, Besançon, France, 2003.

[58] F. DADEAU. *Vérification automatique de machines abstraites B paramétrées—Exemples, preuves et outils.* Rapport de DEA, Laboratoire d'Informatique de l'Université de Franche-Comté, Besançon, France, 2003.

[59] B. GOGNIAT. *Pilotage de bancs de test de systèmes embarqués à partir de l'environnement State-Chart-Testing-Tools.* Rapport de DEA, Laboratoire d'Informatique de l'Université de Franche-Comté, Besançon, septembre, 2003.

[60] D. K. TRAN. *Coopération de procédures de décision: Etude et implantation.* Rapport de DEA, LORIA – Université Henri Poincaré, Nancy, 2003.

Bibliography in notes

[61] P. A. ABDULLA, B. JONSSON, M. NILSSON, J. D'ORSO. *Regular Model Checking Made Simple and Efficient.* in « Proc. of CONCUR 2002 - Concurrency Theory, 13th International Conference », series Lecture Notes in Computer Science, volume 2421, Springer, pages 116–130, Brno, Czech Republic, August, 2002.

[62] J.-R. ABRIAL. *The B-Book: Assigning Programs to Meanings.* Cambridge University Press, 1996.

[63] A. ARMANDO, M. P. BONACINA, A. K. SEHGAL, S. RANISE, M. RUSINOWITCH. *High-performance deduction for verification: a case study in the theory of arrays.* in « Proceedings of the 2nd Verification Workshop (VERIFY02) », Copenhagen, Denmark, 2002.

[64] L. BACHMAIR, I. V. RAMAKRISHNAN, A. TIWARI, L. VIGNERON. *Congruence Closure modulo Associativity-Commutativity.* in « 3rd International Workshop on Frontiers of Combining Systems

(FroCoS'2000) », series Lecture Notes in Computer Science, volume 1794, Springer-Verlag, K. KIRCHNER, C. RINGEISSEN, editors, pages 242–256, Nancy, France, 2000.

[65] T. BALL, S. K. RAJAMANI. *Automatically Validating Temporal Safety Properties of Interfaces.* in « SPIN 2001 », series LNCS, volume 2057, pages 103–122, 2001.

[66] F. BELLEGARDE, C. DARLOT, J. JULLIAND, O. KOUCHNARENKO. *Reformulation: a Way to Combine Dynamic Properties and Refinement.* in « International Symposium Formal Methods Europe (FME 2001) », series Lecture Notes in Computer Science, Springer-Verlag, Berlin, Germany, 2001.

[67] B. BOIGELOT, P. WOLPER. *Symbolic Verification with Periodic Sets.* in « Proceedings of the sixth International Conference on Computer-Aided Verification CAV », volume 818, Springer-Verlag, DILL, D. L., editor, pages 55–67, Standford, California, USA, 1994.

[68] A. BOUAJJANI, B. JONSSON, M. NILSSON, T. TOUILI. *Regular Model Checking.* in « Computer Aided Verification (CAV'00) », series Lecture Notes in Computer Science, volume 1855, Springer, pages 403–418, Chicago, IL, USA, 2000.

[69] A. BOUAJJANI, A. MUSCHOLL, T. TOUILI. *Permutation rewriting and algorithmic verification.* in « LICS'01 », series IEEE Computer Society, pages 399-408, 2001.

[70] F. BOUQUET, B. LEGEARD, F. PEUREUX. *Constraint Logic Programming with Sets for Animation of B formal Specifications.* in « Workshop on (Constraint) Logic Programming and Software Engineering (CLPSE'2000) », 2000.

[71] W. R. BUSH, J. D. PINCUS, D. J. SIELAFF. *A static analyzer for finding dynamic programming errors.* in « Software–Practice & Experience », number 30, 2000, pages 775–802.

[72] Y. CHEVALIER, L. VIGNERON. *A Tool for Lazy Verification of Security Protocols.* in « 16th IEEE International Conference Automated Software Engineering », 2001.

[73] J. CLARK, J. JACOB. *A Survey of Authentication Protocol Literature.* 1997, http://www.cs.york.ac.uk/~jac/papers/drareviewps.ps.

[74] J.-F. COUCHOT. *Atteignabilité d'états et spécifications logico-ensemblistes.* Mémoire de DEA, Laboratoire d'Informatique de l'Université de Franche-Comté, Besançon, France, octobre, 2002.

[75] J. DICK, A. FAIVRE. *Automating the Generation and Sequencing of Test Cases from Model-Based Specifications.* in « FME'93: Industrial-Strength Formal Methods », series Lecture Notes in Computer Science, volume 670, Springer-Verlag, pages 268–284, April, 1993.

[76] P. J. DOWNEY, R. SETHI, R. E. TARJAN. *Variations on the Common Subexpressions Problem.* in « Journal of the Association for Computing Machinery », number 4, volume 27, 1980, pages 758–771.

[77] D. EVANS. *Using Specifications to Check Source Code.* Technical report, MIT, 1994, MIT/LCS/TR-628.

[78] A. FINKEL, P. SCHNOEBELEN. *Well Structured Transition Systems Everywhere!*. in « Theoretical Computer Science », number 1–2, volume 256, 2001, pages 63–92.

[79] C. FLANAGAN, K. R. LEINO, M. LILLIBRIDGE, G. NELSON, J. B. SAXE, R. STATA. *Extended Static Checking for Java.* in « Proc. ACM PLDI », pages 234–245, 2002.

[80] M. HIND. *Pointer Analysis: Haven't We Solved This Problem Yet?.* in « 2001 ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering (PASTE'01) », Snowbird, UT, 2001.

[81] D. JACKSON, M. VAZIRI. *Finding Bugs with a Constraint Solver.* in « Proc. of Intl. Symp. on Soft. Test. and Anal. », 2000.

[82] G. NELSON, D. OPPEN. *Fast Decision Procedures Based on Congruence Closure.* in « Journal of the Association for Computing Machinery », number 2, volume 27, April, 1980, pages 356–364.

[83] B. PARREAUX. *Vérification de systèmes d'événements B par model-checking PLTL - Contribution à la réduction de l'explosion combinatoire en utilisant de la résolution de contraintes ensemblistes.* Thèse de doctorat, Université de Franche-Comté, Besançon, 2000.

[84] L. PY, B. LEGEARD, B. TATIBOUET. *Évaluation de spécifications formelles en programmation logique avec contraintes ensemblistes – Application à l'animation de spécification B.* in « AFADL'2000 », pages 21–35, Grenoble, France, 2000.

[85] S. SCHULZ. *System Abstract: E 0.3.* in « Proceedings of the 16th International Conference on Automated Deduction », series Lecture Notes in Artificial Intelligence, volume 1632, Springer-Verlag, H. GANZINGER, editor, pages 297–301, Trento, Italy, 1999.

[86] R. E. SHOSTAK. *Deciding Combinations of Theories.* in « Journal of the Association for Computing Machinery », number 7, volume 21, 1984, pages 583–585.

[87] J. M. SPIVEY. *The Z notation: A Reference Manual.* edition 2nd, Prentice-Hall, 1993.

[88] L. VIGNERON. *Déduction automatique avec contraintes symboliques dans les théories équationnelles.* Thèse d'université, Université Henri Poincaré - Nancy 1, novembre, 1994.