INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team VerTeCs

Verification models and techniques applied to the Testing and Control of reactive Systems

Rennes

THEME COM

Table of contents

1. Team

Head of project-team
Thierry Jéron [CR INRIA]

administrative assistant
Lydie Mabil-Letort [TR INRIA]

Research Fellows (Inria)
Bertrand Jeannet [CR]
Hervé Marchand [CR]
Vlad Rusu [CR]
Florimond Ployette [Technical staff, IR, since September 2004 (with ASCII)]

Post-doctoral fellow (Inria)
Jérôme Leroux [Since September 2004 (with Lande)]

Junior technical staff
François-Xavier Ponscarme [INRIA, until October 2004]

Ph. D. Students
Camille Constant [INRIA, since October 2004]
Benoît Gaudin [MENRT, ATER since October 2004]
Tristan Le Gall [MENRT, since October 2004]
Valery Tschaen [MENRT, ATER since October 2004]
Elena Zinovieva [INRIA, until March 2004]

2. Overall Objectives

The VERTECS team is focussed on the reliability of reactive software using formal methods. By reactive software we mean software that continuously reacts with its environment. The environment can be a human user for a complete reactive system, or another software using the reactive software as a component. Among these, critical systems are of primary importance, as errors occurring during their execution may have dramatic economical or human consequences. Thus, it is essential to establish their correctness before they are deployed in a real environment. Correctness is also essential for less critical applications, in particular for COTS components whose behavior should be trusted before integration in software systems.

For this, the VERTECS team promotes the use of formal methods, i.e. formal specification and mathematically founded analysis methods. During the analysis and design phases, correctness of specifications with respect to requirements or higher level specifications can be established by formal *verification*. Alternatively, *control* consists in forcing specifications to stay within desired behaviours by coupling with a supervisor. During validation, *testing* can be used to check the conformance of implementations with respect to their specifications. *Test generation* is the process of automatically generating test cases from specifications.

More precisely, the aim of the VERTECS project is to improve the reliability of reactive systems by providing software engineers with methods and tools for automating the **test generation** and **controller synthesis** from formal specifications. We adapt or develop formal models for the description of testing and control artifacts, e.g. specifications, implementations, test cases, supervisors. We formally describe correctness relations (e.g. conformance or satisfaction). We also formally describe interaction semantics between testing artifacts. >From these models, relations and interaction semantics, we develop algorithms for automatic test and controller synthesis that ensure desirable properties. We try to be as generic as possible in terms of models and techniques in order to cope with a wide range of specification languages and application domains. We implement prototype tools for distribution in the academic world, or for transfer to industry.

Our research is based on formal models and **verification** techniques such as model checking, theorem proving, abstract interpretation, the control theory of discrete event systems, and their underlying models

and logics. The close connection between testing, control and verification produces a synergy between these research topics and allows us to share theories, models, algorithms and tools.

3. Scientific Foundations

3.1. Underlying models.

Keywords: *controllable/uncontrollable events*, *implicit transition relation*, *input/output events*, *labeled transition systems*, *symbolic*.

The formal models we use are mainly automata-like structures such as labelled transition systems (LTS) and some of their extensions: an LTS is a tuple $M = (Q, \Lambda, \rightarrow, q_o)$ where Q is a non-empty set of states; $q_o \in Q$ is the initial state; A is the alphabet of actions, $\rightarrow \subseteq Q \times \Lambda \times Q$ is the transition relation.

To model reactive systems in the testing context, we use Input/Output labeled transition systems (IOLTS for short). In this setting, interactions between the system and its environment are modeled by input (controlled by the environment) and output events (observed by the environment), and the internal behavior of the system is modeled by internal (non observable) events. The alphabet Λ is then partitionned into $\Lambda_! \cup \Lambda_? \cup \Im$ where $\Lambda_!$ is the alphabet of outputs, $\Lambda_?$ the alphabet of outputs, and $\Im$ the alphabet of internal actions. In the controller synthesis theory, we also distinguish between controllable and uncontrollable events ($\Lambda = \Lambda_c \cup \Lambda_{uc}$), observable and unobservable events ($\Lambda = \Lambda_O \cup \Im$). In order to consider both control and data aspects, we also manipulate (input-output) symbolic transition systems ((IO)STS), which are extensions of (IO)LTS that operate on data (i.e., program variables, communication parameters, symbolic constants) through message passing, guards, and assignments. An IOSTS is a tuple (V, Θ, Σ, T), where V is a set of variables (including a counter variable encoding the control structure), Θ is the initial condition defined by a predicate on V, Σ is the finite alphabet of actions, where each action has a signature (just like in IOLTS, Σ can be partitionned as e.g. $\Sigma_? \cup \Sigma_! \cup \Sigma_\tau$), T is a finite set of symbolic transitions of the form $t = (a, p, G, A)$ where a is an action, p is a tuple of communication parameters, G is a guard defined by a predicate on p and V, and A is an meassignment of variables. The semantics of $IOSTS$ is defined in terms of (IO)LTS where states are vectors of values of variables, and transitions between them are labelled with instantiated actions. This (IO)LTS semantics allows us to perform syntactical transformations at the (IO)STS level while ensuring semantical properties at the (IO)LTS level. An alternative to IOSTS to specify systems with data variables is the model of synchronous dataflow equations.

Our research is based on well established theories: conformance testing, supervisory control, abstract interpretation, and theorem proving. Most of the algorithms that we employ take their origins in these theories:

- graph traversal algorithms (breadth first, depth first, strongly connected components, ...). We use these algorithms for verification as well as test generation and control synthesis.
- abstract interpretation algorithms, specifically in the abstract domain of polyhedraes (for example, Chernikova's algorithm for the computation of dual forms). Such algorithms are used in verification and test generation.
- logical decision algorithms, such as satifiability of formulas in Presburger arithmetics. We use these algorithms during generation and execution of symbolic test cases.

3.2. Automatic Test Generation

Conformance testing consists in checking whether an implementation under test abbreviated as *IUT*) behaves correctly with respect to its specification. In the line of model-based testing, we use formal specifications and their underlying models to unambiguously define conformance testing and test case generation. One difficult problem is to generate adequate test cases (the selection problem) that correctly identify faults (the oracle problem). We use *test purposes* for selection, which allows us to generate tests targeted to specific behaviours. For solving the oracle problem we adapt a well established theory of conformanced testing [48], which formally defines conformance as a relation between formal models of specifications and implementations. This conformance relation, called **ioco** is defined in terms of visible behaviors (called *suspension traces*) of the implementation I (denoted by $STraces(I)$) and those of the specification S (denoted by $STraces(S)$). Suspension traces are sequence of input, output or quiescence (absence of action denoted by δ), thus abstract away internal behaviors that cannot be observed by testers. The conformance relation $ioco$ can be formulated as a partial inclusion of visible behaviors

$$STraces(I) \cap STraces(S).\Lambda_!^\delta \subseteq STraces(S)$$

Roughly speaking, an implementation is conformant if after a suspension trace of the specification, the implementation can only show outputs and quiescences of specfication.

We use IOLTS (or IOSTS) as formal models for specifications, implementations, test purposes, and test cases. Most often, specifications are not directly given in such low-level models, but are written in higher-level specification languages (e.g. SDL, UML, Lotos). The tools associated with these languages often contain a simulation API that implements their semantics under the form of IOLTS. On the other hand, the IOSTS model is expressive enough to allow a direct representation of most constructs of the higher-level languages. Test purposes are specified directly as IOLTS (or IOSTS). They are associated with marked states, giving them the status of automata or observers accepting sequences of actions or visible behaviors (suspension traces), $ASTraces(TP)$. Selection of test cases roughly amounts at computing visible behaviors of the specification that are accepted by the test purpose i.e. $STraces(S) \cap ASTraces(TP)$.

A test case produces a *verdict* when executed on an implementation. These are formalized in IOLTS (or IOSTS) by special states (or locations). A *Fail* verdict means that the IUT is rejected, a Pass verdict means that the IUT exhibited a correct behavior and the test purpose has been satisfied, while an Inconclusive verdict is given to a correct behavior that is not accepted by the test purpose. Based on these models, an interaction semantics, and the conformance relation, one can then define correctness properties of test cases and test suites (sets of test cases). Typical properties are soundness (no conformant implementation may be rejected) and exhaustiveness (every non conformant implementation may be rejected).

We have developed test generation algorithms. These algorithms are based on the computation of the visible behaviors of the specification $STraces(S)$, involving the identification of quiescence and determinization, the construction of a product between the specification and test purpose which accepted behavior is $STraces(S) \cap ASTraces(TP)$, and finally the selection of accepted behaviors. Selection can be seen as a model-checking problem where one wants to identify states (and transition between them) that are reachable from the initial state and co-reachable from the accepting states. We have proved that these algorithms ensure that the (infinite) set of all possibly generated test cases is sound and exhaustive.

Apart from these theoretical results, our algorithms are designed to be as efficient as possible in order to be able to scale up to real applications. Roughly speaking, test generation consists in computing the intersection of the observable behavior of the specification (traces and quiescence) and the language accepted by the test purpose. The computation of observable behaviors involves determinization, while language intersection is based on computing the set of states that are reachable from initial states and co-reachable states from accepting states.

Our first test generation algorithms are based on enumerative techniques, optimized to fight the state-space explosion problem. We have developed on-the-fly algorithms, which consist in performing a lazy exploration of the set of states that are reachable in both the specification and the test purpose. The resulting test case

is an IOLTS whose traces describe interactions with an implementation under test. This technique is now mature and implemented in the TGV tool, which we often use in industrial collaborations. We are continuously improving the technique. However, what characterizes this enumerative technique is that values of variables and communication parameters are instantiated at test generation time.

More recently, we have explored symbolic test generation techniques. This is a promising technique whose main objective is to avoid the state space explosion problem induced by the enumeration of values of variables and communication parameters. The idea consists in computing a test case under the form of an *IOSTS*, i.e., a reactive program in which the operations on data is kept in a symbolic form. The syntactical transformations that we define on IOSTS should ensure properties of their IOLTS semantics. However, most of the operations involved in test generation (determinization, product, reachability, and coreachability) become undecidable. For determinization we employ heuristics that allow us to solve the so-called bounded observable non-determinism (i.e., the result of an internal choice can be detected after finitely many observable actions). The product is defined syntactically. Finally test selection is performed as a syntactical transformation of transitions which is based on a semantical reachability and co-reachability analysis. As both problems are undecidable for IOSTS, we compute over-approximations by abstract interpretation techniques. Nevertheless, these overapproximations still ensure soundness of test cases. These techniques are implemented in the STG tool, with an interface with NBAC used for abstract interpretation.

3.3. Controller Synthesis

3.3.1. The Supervisory Control Problem

is concerned with ensuring (not only checking) that a computer-operated system works correctly. More precisely, given a specification model and a required property, the problem is to control the specification's behavior, by coupling it to a supervisor, such that the controlled specification satisfies the property [44]. The models used are LTS (and the associated language), which make a distinction between *controllable* and *non-controllable* actions and between *observable* and *non-observable* actions. Typically, the controlled system is constrained by the supervisor, which acts on the system's controllable actions and forces it to behave as specified by the property. The control synthesis problem can be seen as a constructive verification: building a supervisor that prevents the system from violating a property. Several kinds of properties can be ensured such as reachability, invariance, attractivity, etc. Techniques adapted from model checking are then used to compute the supervisor w.r.t. the objectives. Optimality must be taken into account as one often wants to obtain a supervisor that constrains the system as few as possible.

3.3.2. Optimal Control.

We are also interested in the Optimal Control Problem. The purpose of optimal control is to study the behavioral properties of a system in order to generate a supervisor that constrains the system to a desired behavior according to quantitative and qualitative requirements. In this spirit, we have been working on the optimal scheduling of a system through a set of multiple goals that the system had to visit one by one [6]. We have also extended the results of [47] to the case of partial observation in order to handle more realistic applications [41].

3.3.3. Control of Hierarchical Discrete Event System.

In many applications and control problems, FSM are the starting point to model fragments of a large scale system, which usually consists of several composed and nested sub-systems. Knowing that the number of states of the global systems grows exponentially with the number of parallel and nested sub-systems, we have been interested in designing algorithms that perform the controller synthesis phase by taking advantage of the structure of the plant without expanding the system [8]. In other words, given the modular structure of the system, it becomes of interest, for computational reasons, to be able to synthesize a supervisor on each sub-part of the system and then to infer a global supervisor from the local ones.

In order to reduce the complexity of the supervisor synthesis phase, several approaches have been considered in the literature. Modular control [51] and modular plant [52] are natural ways to handle this problem.

Similarly, in order to take into account nested behaviors, some techniques based on model aggregation methods [50][32] have been proposed to deal with hierarchical control problems. Another direction has been proposed in [31]. Brave and Heimann in [31] introduced Hierarchical State Machines which constitute a simplified version of the STATECHARTS. Compared to the classical state machines, they add orthogonality and hierarchy features. Some other works dealing with control and hierarchy can be found in [37][40]. This is the direction we have chosen in the VERTECS Team [8].

3.4. Verification

Both controller synthesis and conformance testing rely on the ability to solve reachability and coreachability problems on a formal model. These problems are particular, but important cases of verification problems. Verification in its full generality consists in checking that a system, which is specified in a formal model, satisfies a required property. When the state space of the system is finite and not too large, verification can be carried out by graph algorithms (model-checking). For large or infinite state spaces, we can perform approximate computations, either by computing a finite abstraction and resort to graph algorithms, or preferably by using more sophisticated abstract interpretation techniques. Another way to cope with large or infinite state systems is deductive verification, which, either alone or in combination with compositional and abstraction techniqes, can deal with complex systems that are beyond the scope of fully automatic methods.

3.4.1. Abstract interpretation and numerical data handling

The techniques described above, which are dedicated to the analysis of LTSs, are already mature. It seems natural to extend them to IOSTSs or data-flow applications that manipulate variables taking their values into possibly infinite data domains.

The techniques we develop for test generation or controller synthesis require to solve state reachability and state coreachability problems (a state s is *reachable from an initial stateb* s_i if an execution starting from s_i can lead to s; s is *coreachable from a final state* s_f if an execution starting from s can lead to s_f). These problems can be solved by fixpoint computations (and also by deductive methods). From an algorithmic point of view, those fixpoint computations are the core of the test generation and controller synthesis methods designed in the team.

The big change induces by taking into account the data and not only the (finite) control of the systems under study is that the fixpoints become not computable. The undecidability is overcomed by resorting to approximations, using the theoritical framework of Abstract Interpretation [34].

Abstract Interpretation is a theory of approximate solving of fixpoint equations applied to program analysis. Most program analysis problems, among others reachability analysis, come down to solving a fixpoint equation

$$x = F(x), x \in C$$

where C is a lattice. In the case of reachability analysis, if we denote by S the state space of the considered program, C is the lattice $\wp(S)$ of sets of states, ordered by inclusion, and F is roughly the "*successor states*" function defined by the program.

The exact computation of such an equation is generally not possible for undecidability (or complexity) reasons. The fundamental principles of Abstract Interpretation are:

1. to substitute to the *concrete domain* C a simpler *abstract domain* A (static approximation) and to transpose the fixpoint equation into the abstrat domain, so that one have to solve an equation $y = G(y), y \in A$;
2. to use a *widening operator* (dynamic approximation) to make the iterative computation of the least fixpoint of G converge after a finite number of steps to some upper-approximation (more precisely, a post-fixpoint).

Apprximations are conservative so that the obtained result is an upper-approximation of the exact result. Those two principles are well illustrated by the Interval Analysis [33], which consists in associating to each numerical variable of a program an interval representing an (upper) set of reachable values:

1. One subtitute to the concrete domain $\wp(R^n)$ induced by the numerical variables the abstract domain $(I_R)^n$, where I_R denotes the set of intervals on real numbers; a set of values of a variable is then represented by the smallest intervals containing it;
2. An iterative computation on this domain may not converge: it is indeed esasy to generate an infinite sequence of intervals which is strictly growing. The "standard" widening operator extrapolates by $+\infty$ the upper bound of an interval if the upper bound does not stabilize within a given number of steps (and similarly for the lower bound).

In this example, the state space $\wp(R^n)$ that should be abstracted has a simple structure, but this may be more complicated when variables belong to different data types (Booleans, numerics, arrays) and when it is necessary to establish *relations* between the values of different types.

Programs performing dynamic allocation of objects in memory have an even more complex state space. Solutions have been devised to represent in an approximate way the memory heap and pointers between memory cells by graphs (*shape analysis* [46][45]). Values contained in memory cells are however generally ignored.

In the same way, programs with recursive procedure calls, parameter passing and local variables are more delicate to analyse with precision. The difficulty is to abstract the execution stacks which may have a complex structure, particularly whenparameter passing by reference is allowed, as it induces aliasing on the stack [30].

3.4.2. Theorem Proving

For verification we also use theorem proving and more particularly the PVS [42] and COQ [43] proof assistants. These are two general-purpose systems based on two different versions of higher-order logic. A verification task in such a proof assistant consists in encoding the system under verification and its properties into the logic of the proof assistant, together with verification *rules* that allow to prove the properties. Using the rules usually requires input from the user; for example, proving that a state predicate holds in every reachable state of the system (i.e., it is an *invariant*) typically requires to provide a stronger, *inductive* invariant, which is preserved by every execution step of the system. Another type of verification problem is proving *simulation* between a concrete and an abstract semantics of a system. This too can be done by induction in a systematic manner, by showing that, in each reachable state of the system, each step of the concrete system is simulated by a corresponding step at the abstract level.

4. Application Domains

4.1. Panorama

Keywords: *Telecommunication*, *control-command Systems*, *smart-cards*, *software embedded systems*, *transportation systems*.

The methods and tools developed by the VERTECS project-team for test generation and control synthesis of reactive systems are quite generic. This allows us to apply them in many application domains where the presence of software is predominant and its correctness is essential. In particular, we apply our research in the context of telecommunication systems, for embedded systems, for smart-cards application, and control-command systems.

4.2. Telecommunication systems

Our research on test generation was initially proposed for conformance testing of telecommunication protocols. In this domain, testing is a normalized process [29], and formal specification languages are widely

used (SDL in particular). Our test generation techniques have already proved useful in this context, going up to industrial transfer. New standardized component-based design methodologies such as UML and OMG's MDE will certainly increase the need for formal techniques in order to ensure the composability of components, by verification and testing. We believe that our techniques, by their genericity and adaptativity, will also prove useful at different levels of these methodologies, from component testing to system testing.

4.3. Software Embedded Systems

In the context of transport, software embedded systems are increasingly predominant. This is particularly important in automotive systems, where software replace electronics for power train, chassis (e.g. engine control, steering, brakes) and cabin (e.g. wiper, windows, air conditioning) or new services to passengers are increasing (e.g. telematics, entertainment). Car manufacturers have to integrate software components provided by many different suppliers, according to specifications. One of the problems is that testing is done late in the life cycle, when the complete system is available. Faced with these problems, but also complexity of systems, compositionality of components, distribution, etc, car manufacturers now try to promote standardized interfaces and component-based design methodologies. They also develop virtual platforms which allows for testing components before the system is complete. It is clear that software quality and trust are one of the problems that have to be tackled in this context. This is why we believe that our techniques (testing and control) can be useful in such a context.

4.4. Smart-card applications

We have also applied our test generation techniques in the context of smart-card applications. Such applications are typically reactive as they describe interactions between a user, a terminal and a card. The number and complexity of such applications is increasing, with more and more services offered to users. The security of such applications is of primary interest for both users and providers and testing is one the means to improve it.

4.5. Control-command Systems

The main application domain for controller synthesis is control-command systems. In general, such systems control costly machines (see e.g. robotic systems, flexible manufacturing systems), that are connected to an environment (e.g. a human operator). Such systems are often critical systems and errors occurring during their execution may have dramatic economical or human consequences. In this field, the controller synthesis methodology (CSM) is useful to ensure by construction the interaction between 1) the different components 2) the environment and the system itself. For the first point, the CSM is often used as a safe scheduler, whereas for the second one, the supervisor can be interpreted as a safe discrete tele-operation system.

5. Software

5.1. Tgv

Keywords: *CADP*, *Conformance Testing*, *IF*, *Lotos*, *ObjectGéode*, *SDL*, *TGV*, *TTCN*, *UML*.

Participants: Thierry Jéron [contact], Valéry Tschaen.

TGV (Test Generation with Verification technology) is a tool for test generation of conformance test suites from specifications of reactive systems [4]. It is based on the IOLTS model, a well defined theory of testing, and on-the-fly test generation algorithms coming from verification technology. Originally, TGV allows test generation focussed on well defined behaviors formalized by test purposes. The main operations of TGV are (1) a synchronous product which identifies sequences of the specification accepted by a test purpose, (2) abstraction and determinization for the computation of next visible actions, (3) selection of test cases by the computation of reachable states from the initial states and co-reachable states from accepting states. TGV has

been developed in collaboration with Vérimag Grenoble and uses libraries of the CADP toolbox (VERIMAG and VASY). TGV can be seen as a library that can be linked to different simulation tools through well defined APIs. An academic version of TGV is distributed in the CADP toolbox and allows test generation from Lotos specifications by a connection to its simulator API. The same API is used for a connection with the UMLAUT validation framework of UML models. This version has been transfered in the SDL ObjectGéode toolset as part of the TestComposer tool. A new version of TGV is currently adapted to a new API of the IF simulator (VERIMAG) allowing test generation from IF and UML models (via a compilation from UML to IF). This new version extends the previous one with new functionalities for coverage based test generation combined with test purposes based test generation. The first version of TGV is protected by APP (Agence de Protection des Programmes) Number IDDN.FR.001.310012.00.R.P.1997.000.2090. The new version is currently being deposit at APP.

5.2. Nbac

Keywords: *Abstract Interpretration*, *Reactive Systems*, *boolean and numerical types*, *polyhedra*.

Participant: Bertrand Jeannet [contact].

NBAC is a verification/slicing tool developed in collaboration with Vérimag. This tool analyses synchronous and deterministic reactive systems containing combination of Boolean and numerical variables and continuously interacting with an external environment. Its input format is directly inspired by the low-level semantics of the LUSTRE dataflow synchronous language. Asynchronous and/or non-deterministic systems can be compiled in this model. The kind of analyses performed by NBAC are: reachability analysis from a set of initial states, which allows to compute invariants satisfied by the system; coreachability analysis from a set of final states, which allows to compute sets of states that may lead to a final state; and combination of the above. The result of an analysis is either a set of states together with a necessary condition on states and inputs to stay in this set during an execution, either a verdict of a verification problem. The tool is founded on the theory of abstract interpretation: sets of states are approximated by abstract values belonging to an abstract domain, on which fix-point computations are performed. The originality of NBAC resides in

- the use of a very general notion of control structure in order to very precisely tune the tradeoff between precision and efficiency;
- the ability to dynamically refine the control structure, and to guide this refinement by the needs of the analysis.
- sophisticated methods for computing postconditions and preconditions of abstract values.

5.3. Stg

Keywords: *test de conformité*, *test et vérification symboliques*.

Participants: Vlad Rusu [contact], Elena Zinovieva, François-Xavier Ponscarme, Thierry Jéron.

STG (Symbolic Test Generation) [2] is a prototype tool for the generation and execution of test cases using symbolic techniques. It takes as input a specification and a test purpose described as IOSTS, and generates a test case program also in the form of IOSTS. Test generation in STG is based on a syntactic product of the specification and test purpose IOSTS, an extraction of the subgraph corresponding to the test purpose, elimination of internal actions, determinization, and simplification. The simplification phase now relies on NBAC, which approximates reachable and coreachable states using abstract interpretation. It is used to eliminate unreachable states, and to strengthen the guards of system inputs in order to eliminate some Inconclusive verdicts. After a translation into C++ or Java, test cases can be executed on an implementation in the corresponding language. Constraints on system input parameters are solved on-the-fly during execution using the Omega constraint solver. The first version of STG was developped in C++. This version is currently being deposit at APP. A new version in OCaml is now under developpement. This version is more generic and will serve as a library for symbolic operations on IOSTS.

5.4. Sigali

Keywords: *Controller Synthesis*, *symbolic techniques*, *verification*.

Participant: Hervé Marchand [contact].

SIGALI is a model-checking tool that operates on ILTS (Implicit Labeled Transition Systems, an equational representation of an automaton), an intermediate model for discrete event systems. It offers functionalities for verification of reactive systems and discrete controller synthesis. It is developed jointly by the ESPRESSO and VERTECS teams. The techniques used consist in manipulating the system of equations instead of the sets of solution, which avoids the enumeration of the state space. Each set of states is uniquely characterized by a predicate and the operations on sets can be equivalently performed on the associated predicates. Therefore, a wide spectrum of properties, such as liveness, invariance, reachability and attractivity can be checked. Algorithms for the computation of predicates on states are also available [7]. SIGALI is connected with the Polychrony environment (as well as the Matou environment), thus allowing the modeling of reactive systems by means of Signal Specification or Mode Automata and the visualization of the synthesized controller by an interactive simulation of the controlled system. SIGALI is protected by APP (Agence de Protection des Programmes).

5.5. Syntool

Keywords: *Controller Synthesis*, *structured systems*.

Participants: Benoit Gaudin, Hervé Marchand [contact].

SYNTOOL is a tool dedicated to the control of structured discrete systems event. It implements the theory developed in [11]. SYNTOOL has an API allowing the user to graphically describe the different LTS modeling the plant, to perform some controller synthesis computations solving e.g. the forbidden state avoidance problem for structured systems and finally to simulate the result (i.e. the behavior of the controlled system). This tool is currently under testing.

5.6. Rapture

Keywords: *Markov Decision Processes*, *Probabilistic verification*.

Participant: Bertrand Jeannet [contact].

RAPTURE is a verification tool developed jointly by BRICS and INRIA [39]. The tool is designed to verify reachability properties on Markov Decision Processes (MDP), also known as Probabilistic Transition Systems. This model can be viewed both as an extension to classical (finite-state) transition systems extended with probability distributions on successor states, or as an extension of Markov Chains with non-determinism. We have developed a simple automata language that allows to describe a set of processes communicating over a set of channels *à la* CSP. Processes can also manipulate local and global variables of finite type. Probabilistic reachability properties are specified by defining two sets of initial and final states together with a probability bound. The originality of the tool is to provide two reduction techniques that limit the state space explosion problem: automatic abstraction and refinement algorithms, and the so-called essential states reduction [35].

6. New Results

6.1. Controller Synthesis

Keywords: *Hierarchical models*, *controller synthesis methodology*, *symbolic methods*.

6.1.1. Supervisory Control of Structured Discrete Event Systems

Participants: Benoit Gaudin, Hervé Marchand.

We investigated the supervisory control of a class of Discrete Event Systems modeled either by a collection of Finite State Machines that behave asynchronously or by a Hierarchical Finite State Machine. The basic

problem of interest is to ensure the invariance of a set of particular configurations in the system. When the system is modeled as asynchronous FSMs, we provide algorithms that, based on a particular decomposition of the set of forbidden configurations, solve the control problem locally (i.e. on each component without computing the whole system) and produce a global supervisor ensuring the desired property. We then provide sufficient conditions under which the obtained controlled system is non-blocking. This kind of objectives may be useful to perform dynamic interactions between different parts of a system. Finally, we apply these results to the case of Hierarchical Finite State Machines. [14].

We also developed a methodology allowing to control structured discrete event systems, when the control objectives are expressed in terms of languages. We investigated the computation of the supremal controllable language contained in the one of the specification. We do not adopt the decentralized approach. Instead, we have chosen to perform the control on some approximations of the plant derived from the behavior of each component. The behavior of these approximations is restricted so that they respect a new language property for discrete event systems called *partial controllability condition* that depends on the specification. It is shown that, under some assumptions, the intersection of these "controlled approximations" corresponds to the supremal controllable language contained in the specification with respect to the plant. This computation is performed without having to build the whole plant, hence avoiding the state space explosion induced by the concurrent nature of the plant [19][23].

6.1.2. A Model for the Automatic Generation of Safe Task Handlers

Participant: Hervé Marchand.

In collaboration with the POP-ART Project, we are been interested in the programming of real-time control systems, such as in robotic, automotive or avionics systems. These systems are designed with multiple tasks, each with multiple modes. implementing a functionality with different levels of quality (e.g., computation approximation), and cost (e.g., computation time, energy). Due to the interactions between components, it is complex to design task handlers that control the switching of activities in order to insure safety properties of the global system. We proposed a model of tasks in terms of transition systems, designed especially with the purpose of applying existing (optimal) discrete controller synthesis techniques. This provides us with a systematic methodology, for the automatic generation of safe task handlers, with the support of synchronous languages and associated tools for compilation and formal computation [27].

6.1.3. Supervisory Control of Symbolic and hybrid Transition systems

Participants: Tristan Le Gall, Bertrand Jeannet, Hervé Marchand.

This year, we have been interested in solving the safety controller synthesis problem for various models (from finite transition systems to hybrid systems). Within this framework, we have been mainly interested in an intermediate model: the symbolic transition system. We first redefined the concept of controllability by introducing the notion of uncontrollable transitions. We then defined synthesis algorithms based on abstract interpretation techniques so that we can ensure finiteness of the computations. We finally generalized our methodology to the control of hybrid systems, which gives an unified framework to the supervisory control problem for several classes of models.

6.2. Test generation on enumerative and symbolic models

Keywords: *models*, *symbolic transition systems*, *test generation*, *testing*, *transition systems*.

6.2.1. Test generation based on coverage directives

Participants: Thierry Jéron, Valery Tschaen.

Our test generation techniques were previously based on a selection by test purposes. However, this approach necessitates to specify those test purposes. Users sometimes want more automatic ways to generate test cases, from more general selection mechanisms. In the context of the Agedis european project (see 7.1), we have defined more general selection mechanisms, called test selection directives. They allow to describe

both coverage directives (on states, transitions and more generally expressions on variables), test purposes (extended to more general observers) and constraints on data values, and to combine them. Taking into account these test directives involved a deep modification in some test generation algorithms. We also designed algorithms that generate test cases randomly, without any test directive. All these algorithms are incremental in the sense that they produce test cases when these are computed, without waiting the end of the process, thus allowing users to interupt the process with a partial result [28]. We are now formalizing these notions and algorithms. The starting point is a variation of the IOSTS model and its IOLTS semantics where information in states (composed of vectors of variable values) is explicitely considered for test generation. Operations performed during test generation with various selection directives including coverage can then be defined at this semantical level. This formalization should bridge the gap between enumerative test generation and symbolic test generation.

6.2.2. Symbolic test generation

Participants: Elena Zinovieva, Vlad Rusu, Bertrand Jeannet, Thierry Jéron.

In our seminal work [9], we presented a framework for symbolic test generation. These were preliminary ideas which have been improved since then. First the PhD. thesis of Elena Zinovieva presents a complete formalization of these ideas, and describes the integration of the approximated reachability and co-reachability algorithms embodied into the NBac tool, into the general symbolic test generation algorithm of the STG tool. The new reachability/co-reachability algorithms allow for a precise handling of data, and as a result, the new version of the tool is able to generate better test cases (i.e., having less Inconclusive verdicts). The Habilitation of Thierry Jéron [12] presents another viewpoint and new results with a more abstract IOSTS model and a language point of view allowing to completely explain IOSTS transformations in terms of their IOLTS semantics. Finally, in [24] we reformulate these results and define new qualitative properties of test cases. We precisely show what is lost by approximate analysis compared to exact analysis (e.g. in the finite IOLTS case) : inconclusive verdict can be delayed, while pass and fail verdicts are exact.

6.2.3. From Safety Verification to Safety Testing

Participants: Vlad Rusu, Hervé Marchand, Valery Tschaen, Thierry Jéron, Bertrand Jeannet, Camille Constant.

In this work, we present combinations of verification and conformance testing techniques for the formal validation of reactive systems. A formal specification of a system - an input-output automaton with variables that may range over infinite domains - is assumed. Additionally, a set of safety properties for the specification are given under the form of observers described in the same formalism. Then, each property is verified on the specification using automatic techniques (e.g., abstract interpretation) that are sound but not necessarily complete for the class of safety properties considered here. Next, for each property, a test case is automatically generated from the specification and the property and is executed on a black-box implementation of the system. If the verification step was successful, that is, it has established that the specification satisfies the property, then the test execution may detect the violation of the property by the implementation and the violation of the standard IOCO conformance relation [49] between implementation and specification. On the other hand, if the verification step did not conclude (i.e., it did not allow to prove or to disprove the property), then the test execution may additionally detect a violation of the property by the specification. The informations about the relative (in)consistencies between specification, implementation, and properties are reported to the user as test verdicts. The approach is illustrated on the BRP protocol [25]. This work proposes a symbolic approach that extends [22], that were based on finite model using enumerative methods.

6.2.4. Test generation from interprocedural specifications

Participants: Bertrand Jeannet, Liva Randriamanohisoa.

The project has mainly studied the problem of conformance test generation in the case where the specification is intraprocedural, *i.e.,* it does not contain procedure calls and returns. However, allowing such a feature in specifications would allow more expressiveness, both theoritically and from a specification language point of

view. The master thesis of L. Randriamanohisoa investigated the problem of test generation in the case where the specification is defined by a pushdown system (PDS), which extend finite transition systems (LTS) with a stack of symbols belonging to a finite set, and which generates context-free languages. PDS allows to model any recursive programs manipulating finite-state variables and can thus be viewed as the interprocedural generalization of LTS for our application. Two important problems for test generation are still decidable with a low polynomial complexity: the language intersection with a regular language, which allows to consider test purposes modeled with IOLTS, and coreachability (and reachability) analysis [36], which allows to perform test selection. This allowed us to design an exact test generation algorithm resulting in an IOPDS test case. Interesting problems of partial observation are raised by this work: a PDS test case indeed observes only the topmost symbol of its stack.

6.3. Interaction between testing, control and verification

6.3.1. Ensuring the conformance by means of supervisors

Participants: Thierry Jéron, Hervé Marchand, Vlad Rusu, Valéry Tschaen.

We studied the problem of controlling a plant of a system by means of an automatically computed supervisor, in order to ensure a certain conformance relation between the plant and its formal specification. The supervisor can be seen as a device that automatically fixes errors, which otherwise should have been discovered by testing and fixed by hand. The resulting controlled plant conforms to the specification and is maximal in terms of observable behavior [16].

6.3.2. Coverage in model-checking and testing

Participants: Thierry Jéron, Sophie Pinchinat, Sebastien Saudrais.

This work was part of the master trainee of Sebastien Saudrais. Model checking consists in checking that a model of a system satisfies some property. Typically properties are expressed with a temporal logic (CTL, LTL, etc), and systems are modelled with Kripke structures (LTS where states carry atomic propositions true in those states). Coverage in model checking consists in checking that a property covers the model of the system. Intuitively, a state is covered if it plays a role in the satisfaction of the property, i.e. if a mutant obtained by inverting an atomic proposition in this state violates the property. After a study of the state of the art, we tried to transpose the problem in the model of IOLTS. Now information is carried by transitions We first defined mutants for these models and their unfoldings by addition, suppression and modification of transitions. We then defined adequate notions of coverage for LTL properties on traces and compared these notions. This work constitutes a first step towards a new formal definitions of coverage for test generation.

6.4. Applications of theorem proving

6.4.1. Compositional verification of an ATM protocol

Participant: Vlad Rusu.

This works [26] describes a methodology and a case study in formal verification. The case study is the protocol, a member of the adaptation layer whose main role is to perform a reliable data transfer over an unreliable communication medium. The methodology involves: (1) a state-space exploration for initial debugging; (2) a partial-order abstraction that preserves the properties of interest; and (3) a compositional verification of the properties at the abstract level using the theorem prover. Steps (2) and (3) guarantee that the properties still hold on the whole (composed, concrete) system as well. The value of the approach lies in adapting and integrating several formal techniques for verifying a real case study.

6.4.2. Extracting a Data Flow Analyser in Constructive Logic

Participant: Vlad Rusu.

This work has been done in cooperation with David Cachera, Thomas Jensen, and David Pichardie from the Lande project-team of Irisa. We show how to formalise a constraint-based data flow analysis in the

specification language of the COQ proof assistant. This involves defining a dependent type of lattices together with a library of lattice functors allowing for a modular construction of complex abstract domains. Constraints are expressed via an intermediate representation that allows for efficient constraint resolution. Correctness with respect to an operational semantics is proved formally. The proof of existence of a correct, minimal solution of the constraints is constructive, which means that the extraction mechanism of COQ provides a provably correct data flow analyser in OCAML. The library of lattices together with the intermediate representation of constraints are defined in an analysis-independent fashion, thus providing a generic framework for proving and extracting static analysers in COQ [18].

6.5. Verification and Abstract Interpretation

6.5.1. Abstracting Call-Stacks for interprocedural verification of imperative programs

Keywords: *Interprocedural Verification*.

Participants: Bertrand Jeannet, Wendelin Serwe.

In the context of the ARC Modocop, we proposed [21] a new approach to interprocedural analysis and verification, consisting of deriving an interprocedural analysis method by abstract interpretation of the standard operational semantics of programs. The advantages of this approach are twofold. From a methodological point of view, it provides a direct connection between the concrete semantics of the program and the effective analysis, which facilitates implementation and correctness proofs. This method also integrates two main, distinct methods for interprocedural analysis, namely the call-string and the functional approaches introduced by Sharir and Pnueli. This enables strictly more precise analyses and additional flexibility in the tradeoff between efficiency and precision of the analysis.

A tool is currently being implemented. Our final goal is to extend to recursive programs the sophisticated techniques implemented in the tool NBAC for reactive programs.

6.5.2. A Relational Approach to Interprocedural Shape Analysis

Keywords: *Interprocedural Verification*, *Shape Analysis*.

Participant: Bertrand Jeannet.

This work addresses the verification of properties of imperative programs with recursive procedure calls, heap-allocated storage, and destructive updating of pointer-valued fields-i.e., interprocedural shape analysis. It presents a way to apply some previously known approaches to interprocedural dataflow analysis-which in past work have been applied only to a much less rich setting-so that they can be applied to programs that use heap-allocated storage and perform destructive updating [20].

7. Contracts and Grants with Industry

7.1. IST-Agedis

Keywords: *UML*, *distributed software*, *test generation*, *testing*.

Participants: Thierry Jéron, Valery Tschaen.

Agedis [11/2000-01/2004] is an European project IST (http://www.agedis.de/) on the automated generation and execution of test suites for distributed component based software. The goal of the project is to propose a toolset for test generation, test execution and test result analysis, starting from UML models of distributed software. This project allows us to improve the TGV technology by extending it with coverage based test generation (see 6.2.1) using a more complete API to the IF simulation tool of VERIMAG. It also gives us a new opportunity to connect our technology to the UML world. Partners are IBM Haifa, IBM Hursley, France Télécom R& D, IntraSoft, Imbus, Oxford University and Vérimag. We are subcontractors of Vérimag in this project. Agedis lasted in the beginning of year 2004 with demonstrations of the complete chain, from UML design models and test selection directives to test generation and execution on target.

7.2. France Telecom R&D

Keywords: *test generation*, *testing*, *vocal phone services*.

Participants: Camille Constant, Thierry Jéron, Vlad Rusu.

The goal of this 3-year project (starting October 2004) is to build a platform for the formal validation of France Telecom's vocal phone services. Vocal services are based on speech recognition and synthesis algorithms, and they include automatic connection to the callee's phone number by pronouncing her name, or automatic pronuncation of the callee's name whose phone number was dialed in by the user. Here, we are not interested in validation the voice recognition/synthesis algorithms, but on the logic surrounding them. For example, the system may allow itself a certain number of attempts for recognizing a name, after which it switches to normal number-dialing mode, during which the user may choose to go back to voice-recognition mode by pronouncing a certain keyword. This logic may become quite intricate, and this complexity is multiplied by the number of clients that may be using the service at any given time. Its correctness has been identified by France Telecom as a key factor in the success of the deployment of voice-based systems. To validate them we are planning to apply a combination of formal verification and conformance testing techniques (cf. Section 6.2.3).

8. Other Grants and Activities

8.1. National grants & contracts

8.1.1. CNRS ACI Sécurité V3F : Validation and Verification of programs with floating point numbers

Participants: Bertrand Jeannet, Thierry Jéron, Jérome Leroux.

V3F (http://lifc.univ-fcomte.fr/~v3f/)[2003-2005] is a project involving LIFC Besançon, Inria-I3S Nice, LIST-CEA Saclay and project teams Lande and Vertecs in Irisa. The goal of this project is to provide tools to support the verification and validation process of programs with floating-point numbers. More precisely, project 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 will be based on constraint programming. Constraints solving techniques have been successfully used during the last years for automatic test data generation, model-checking and static analysis. However in all these applications, the domains of the constraints were restricted either to finite subsets of the integers, rational numbers or intervals of real numbers. Hence, the investigation of solving techniques for constraint systems over floating-point numbers is an essential issue for handling problems over the floats.

So, the expected results of project V3F are a clean design of constraint solving techniques over floating-point number, and a deep study of the capabilities of these techniques in the software validation and verification process. More precisely, we will develop an open and generic prototype of a constraint solver over the floats. We will also pay a special attention on the integration of floats into various formal notations (e.g., B, Lustre, UML/OCL) to allow an effective use of the constraint solver in formal model verification, automatic test data generation (functional and structural) and static analysis.

Our contribution to this project is first to precisely formalize a conformance testing theory for programs with floating point with respect to their specifications, and second, to describe test generation algorithms in this framework. We currently investigate several possibiliies for the first point, where the conformance testing theory should take into account imprecisions due to floating point calculations.

8.1.2. CNRS ACI Sécurité Potestat : Security policies : test directed analysis of open network systems

Participants: Thierry Jéron, Hervé Marchand, Vlad Rusu.

The Potestat project (http://www-lsr.imag.fr/POTESTAT/) [2004-2006] involves LSR-IMAG Grenoble, Verimag Grenoble and Lande and Vertecs project teams in Irisa.

In the framework of open service implementations, based on the interconnection of heterogeneous systems, the security managers lack of well-formalized analysis techniques. The security of such systems is therefore organized from pragmatic elements, based on well-known vulnerabilities and their associated solutions. It then remains to verify if such security policies are correctly and effectively implemented in the actual system. This is usually carried out by auditing the administrative procedures and the system configuration. Tests are then performed, for instance by probing, to check the presence of some particular vulnerabilities. Although some tools are already available for specific tests (like password crackers), there is no solution to analyze the whole system conformance with respect to a security policy. This lack may be explained by several factors. First, there is currently no complete study about the formal modeling of a security policy, even if some particular aspects have been more thoroughly studied. Furthermore, verification based researches about security usually concerned more precise elements, like cryptographic protocols or code analysis. Finally, most of these works are dedicated to a priori verification of the coherency of security policies before their implementation. We are concerned here by the conformance of a system configuration with respect to a given policy. In the framework of the POTESTAT project we plan to tackle this problem according to the following items: - Formal modelization of security policies, allowing a test directed analysis. - Definition of a conformance notion between a system configuration and some security policies elements. The goal is to obtain a test theory similar to the one existing in the protocol testing area (like the Z.500 norm). - Definition of methods to test this conformance notion, including the testability problems, the environment of execution, code analysis and test selection. A long-term of this project is to offer some tools allowing security managers: - to modelize information flow, network elements (protocols, node types and their associated security policy, etc) to better describe the security policy for conformance testing; - to provide some practical tools to perform coherency verification and vulnerabilities detection.

8.1.3. CNRS ACI Sécurité APRON : Analysis of Numerical Programs

Participant: Bertrand Jeannet.

The APRON (Analyse de PROgrammes Numériques) project (http://www.cri.ensmp.fr/apron/) [2004-2006] involves ENSMP, LIENS-ENS, LIX-Polytechnique, VERIMAG and Vertecs-Irisa.

The goal is to develop methods and tools to analyze statically embedded software with high-level of criticity for which the detection of errors at run-time is unacceptable for safety or security reasons. Such safety and security software is found in the context of transportation, automotive, avionics, space, industrial process control and supervision, etc. One characteristics of such software is that it is based on physical models whence involve a lot of numerical computations. Moreover, *counters* play an important role in the control of reactive programs (e.g., delay counting in synchronous programming). Critical properties depending on these counters are generally outside the scope of model-checking approaches, while being simple enough to be accurately analyzed by more sophisticated numerical analyses.

The goal of the project is the static analysis of large specifications (e.g. à la LUSTRE) and corresponding programs (e.g. of 100 to 500 000 LOCs of C), made of thousands of procedures, involving a lot of numerical floating-point computations, as well as boolean and counter-based control in order to:

- verify critical properties (including the detection of possible runtime errors), and
- help in automatically locating the origin of critical property potential violation.

An example of such critical properties, as found in control/command programs, is of the form “under a condition holding on boolean and numerical variables for some time, the program must imperatively establish a given boolean and/or numerical property, in given bounded delay”.

Vertecs contributes to the following topics within the APRON project:

- The design and implementation of a common interface to several abstraction libraries (intervals, linear equalities, octagons, polyhedra, ...and their combination).
- The study of adaptative techniques for adjusting the tradeoff between the efficiency and the precision of analyses, among other dynamic partitioning techniques [38]. Results have already been obtained in the intraprocedural case, but to a less extend in the interprocedural case.

Vertecs focus mainly on LUSTRE specifications and provides with the NBAC tool the main experimental platform of the project for the verification of critical properties on such specifications.

8.2. Collaborations

8.2.1. Collaborations with other INRIA project-teams:

We collaborate with several Inria project-teams. We collaborate with the LANDE project-team in two ACI-Securité grants (V3F and POTESTAT). With ESPRESSO project-team for the development of the SIGALI tool inside the Polychrony environment. With the POP-ART project-team on the use of the controller synthesis methodology for the control of control-command systems (e.g. robotic systems). With the TRISKELL project-team we have different collaborations on testing, in particular on the connection of TGV and UMLAUT. With DISTRIBCOM on symbolic distributed test generation and securtity testing. With the S4 project-team on the use of control and game theory for test generation. With the VASY project-team on the use of CADP libraries in TGV and the distribution of TGV in the CADP toolbox.

8.2.2. Collaborations with French research groups outside INRIA:

Our main collaborations are with Vérimag. Beyond formalized collaborations (IST Agedis, CNRS AS Test, ARC Modocop), we also collaborate on the connection of NBAC with Lurette for the analysis of Lustre programs, as well as the connection of SIGALI and Matou.

8.2.3. International Collaborations

Aalborg University *in Denmark (K. G. Larsen) and University of Twente (P. Katoen) on probabilistic verification. We participate in the development of the Rapture tool.*

University of South Carolina *in USA (D. Clarke) on symbolic test generation with emphasis on the STG tool.*

CNR Pisa *in Italy (A. Bertolino) on using TGV for test generation for software architectures.*

University of Wisconsin *(T. Reps) on shape analysis. Bertrand Jeannet visited Tom Reps during summer 2004.*

ENIS Sfax *in Tunisia (M. Tahar Bhiri). Thierry Jéron is co-supervisor of a master student Hatem Hamdi working on robustness testing.*

8.2.4. ARTIST Network of Excellence

We are partners of the ARTIST Network of Excellence (http://www.artist-embedded.org/) on Embedded Systems, involved in the Testing and Verification cluster with Brics in Aalborg (DK), University of Twente (NL), University of Liège (B), Upsalla (SE), Verimag Grenoble, ENS Cachan, LIAFA Paris, EPFL Lausanne (S). The aim of the cluster is to develop a theoretical foundation for real-time testing, real-time monitoring and optimal control, to design data structures and algorithms for quantitative analysis, and to apply testing and verification tools in industrial settings. For security, we plan to create a common semantic framework for describing security protocols including notion of "trust" and to develop tools and methods for the verification of security protocols. Test and verication tools developed by partners will be made available via web-portal and with dedicated verfication servers. A first cluster meeting took place in Bruxelles in december 04 during which T. Jéron and V. Rusu gave presentations.

9. Dissemination

9.1. University courses

- B. Gaudin is teaching in DEUG and DIIC 2 in University of Rennes 1 (64h/year).
- B. Jeannet teaches in the Master of Computer Science in Rennes, on abstract interpretation,
- T. Jéron is responsible of a course on Testing in Master of Computer Science at the University of Rennes 1. He also teaches in the engineering school EnstB in Rennes and Brest. The topics of both lectures are testing and model-checking,
- V. Rusu teaches in the Master of Computer Science in Rennes, on deductive verification methods.
- V. Tschaen is teaching in DEUG and DIIC 3 in University of Rennes 1 (64h/year).

9.2. PhD Thesis and Trainees

PhD theses defended in 2004

1. Benoit Gaudin, "*Control of Structured Discrete Event Systems*". November 15th 2004
2. Elena Zinovieva "*Symbolic methods in test generation for reactive systems with data*". November 22th 2004.

Current PhD. theses:

1. Camille Constant, "*Verification and symbolic test generation for reactive systems*",
2. Tristan Le Gall, "*Abstract lattice of fifo channels for verification and control synthesis*"
3. Valéry Tschaen, "*Automatic test generation: Models and Techniques*"

Trainees 2003-2004:

1. Camille Constant, "*Verification and Test of safety properties*", Master student (6 months).
2. Tristan Le Gall, "*Supervisory Control of symbolic and hybrid transition systems*", Master student (6 months).
3. Sebastien Saudrais, "*Coverage in model checking and testing*", Master student (6 months).
4. Liva Randriamanohisoa, "*Test generation for interprocedural specifications*", Master student (6 months).
5. Hatem Hamdi, "*Generation and execution of test cases with Agedis*", Master student ENIS Sfax.

Trainees 2004-2005:

1. Sophie Quinton, "*Vérification à l'éxécution et test de conformité*", Master student (6 months).

9.3. Participation to jurys

- T. Jéron has been member of the “Comissions de Spécialistes 27e” University Rennes 1 until 2004. He was member of the local selection jury for Inria researchers.
- Hervé Marchand is member of the “Comissions de Spécialistes 27e” at University of Rennes 1.

9.4. Conferences, Seminars, invited talks

- Bertrand Jeannet is the organizer of the thematic seminar “68 NQRT” (http://www.irisa.fr/NQRT/index.html) in Irisa.
- Thierry Jéron is PC member of Fates’04 (Linz, Austria 09/04) and Fates’05 (Edinburgh, 07/05), Testcom 2004 (Oxford, 03/04) and Testcom 2005 (Montreal, 06/05), Tacas’05 (Edinburgh, 04/04) and Tool Chair of Tacas’06 (Vienna, 03/06). He is member of the Organizing Committee of the Movep School (Bruxelles, 12/04) . He is Financial Chair of ISSRE 2004 (St Malo, 11/04). He is reviewer for Zentralblatt Math.
  Thierry Jéron gave a presentation on test generation in the Inria-Industry seminar in January 2004. He was also invited to give a seminar in Labri Bordeaux in January 2004 on test generation and control synthesis.
- Jérome Leroux was invited to give a seminar on “The convex hull of a Number Decision Diagran is an effectively computable polyhedron” at LORIA, Nancy (10/2004)
- Hervé Marchand is PC member of MSR’05 (Grenoble, 10/05) and is member of the Organizing Committee of ACSD’05 (St Malo, 06/05).
  Hervé marchand was invited to give a seminar on “the optimal control of discrete event systems” during the "journée GDR MACS" (Aix-en-Provence, 10/04) and on “the control of structured discrete event systems” during the "Journée QSL" (LORIA, Nancy, 10/05).
- Vlad Rusu gave invited talks at the Dagstuhl seminar “Perspectives in Model-Based Testing” (12/09) and at the British FORTEST seminar on model-based testing (IBM Hursley, 21/09).
  Vlad Rusu organized a meeting (25-26/11) at IRISA on “Verification, Testing, and Synthesis” around the visit of Doron Peled, chair of Software Engineering, University of Warwick. The meeting includes a talk by Doron; talks by members of several project-teams of IRISA that are active in the fields of verification, testing, and synthesis; and discussions on collaborations.
- Valery Tschaen gave an invited talk on “Test generation algorithms based on preorders” at the Dagstuhl seminar “Model-based Testing of Reactive Systems” (12-15/01)

10. Bibliography

Major publications by the team in recent years

[1] M. BOZGA, J.-C. FERNANDEZ, L. GHIRVU, C. JARD, T. JÉRON, A. KERBRAT, P. MOREL, L. MOUNIER. *Verification and test generation for the SSCOP protocol*, in "Journal of Science of Computer Programming, special issue on Formal Methods in Industry", vol. 36, n° 1, Janvier 2000, p. 27-52.

[2] D. CLARKE, T. JÉRON, V. RUSU, E. ZINOVIEVA. *STG: a Symbolic Test Generation tool*, in "(Tool paper) Tools and Algorithms for the Construction and Analysis of Systems (TACAS'02)", LNCS, vol. 2280, Springer-Verlag, 2002, http://www.irisa.fr/vertecs/Publis/Ps/2002-TACAS.ps.gz.

[3] B. JEANNET, N. HALBWACHS, P. RAYMOND. *Dynamic Partitioning in Analyses of Numerical Properties*, in "Static Analysis Symposium, SAS'99", LNCS, vol. 1694, 1999.

[4] T. JÉRON. *TGV: théorie, principes et algorithmes*, in "Techniques et Sciences Informatiques, numéro spécial Test de Logiciels", n° 21, 2002.

[5] T. JÉRON, P. MOREL. *Test generation derived from model-checking*, in "CAV'99, Trento, Italy", N. HALBWACHS, D. PELED (editors)., LNCS, vol. 1633, Springer-Verlag, Juillet 1999, p. 108-122.

[6] H. MARCHAND, O. BOIVINEAU, S. LAFORTUNE. *On the Synthesis of Optimal Schedulers in Discrete Event Control Problems with Multiple Goals*, in "SIAM Journal on Control and Optimization", vol. 39, n° 2, 2000, p. 512-532.

[7] H. MARCHAND, P. BOURNAI, M. LE BORGNE, P. LE GUERNIC. *Synthesis of Discrete-Event Controllers based on the Signal Environment*, in "Discrete Event Dynamic System : Theory and Applications", vol. 10, n° 4, Octobre 2000, p. 347-368, http://www.irisa.fr/vertecs/Publis/Ps/J-DEDS.ps.gz.

[8] H. MARCHAND, B. GAUDIN. *Supervisory Control Problems of Hierarchical Finite State Machines*, in "41th IEEE Conference on Decision and Control, Las Vegas, USA", December 2002, http://www.irisa.fr/vertecs/Publis/Ps/2002-CDC.ps.gz.

[9] V. RUSU, L. DU BOUSQUET, T. JÉRON. *An approach to symbolic test generation*, in "International Conference on Integrating Formal Methods (IFM'00)", LNCS 1945, Springer Verlag, 2000, p. 338-357.

[10] V. RUSU. *Verifying a Sliding-Window Protocol using PVS*, in "Formal Techniques for Networked and Distributed Systems (FORTE'01)", Kluwer Academic Publishers, 2001, p. 251-266.

Doctoral dissertations and Habilitation theses

[11] B. GAUDIN. *Contrôle de systèmes à événements discrets structurés*, Ph.D. Thesis, Université de Rennes 1, November 2004.

[12] T. JÉRON. *Contribution à la génération automatique de tests pour les systèmes réactifs*, Habilitation à diriger des Recherches, Université de Rennes 1, March 2004, http://www.irisa.fr/prive/jeron/HDR/doc_hdr_sent2.pdf.

[13] E. ZINOVIEVA-LEROUX. *Méthodes symboliques pour la génération de tests de systèmes réactifs comportant des données*, PhD Thesis, Université de Rennes 1, November 2004.

Articles in referred journals and book chapters

[14] B. GAUDIN, H. MARCHAND. *Supervisory Control of Product and Hierarchical Discrete Event Systems*, in "European Journal of Control", vol. 10, n° 2, 2004.

[15] C. JARD, T. JÉRON. *TGV: theory, principles and algorithms, A tool for the automatic synthesis of conformance test cases for non-deterministic reactive systems*, in "Software Tools for Technology Transfer (STTT)", vol. 6, 2004, http://www.springerlink.com/index/10.1007/s10009-004-0153-x.

[16] T. JÉRON, H. MARCHAND, V. RUSU, V. TSCHAEN. *Ensuring the conformance of reactive discrete-event systems by means of supervisory control*, in "International Journal of Production Research", vol. 42, n° 14, 2004, p. 2809–2826.

[17] V. TSCHAEN. *MOTRES, Model-Based Testing of Reactive systems*, M. BROY, B. JONSSON, J.-P. KATOEN, M. LEUCKER, A. PRETSCHNER (editors)., Common book resulting from GI/Dagstuhl research seminar on Model-Based Testing of Reactive Systems, to appear, Springer, 2004.

Publications in Conferences and Workshops

[18] D. CACHERA, T. JENSEN, D. PICHARDIE, V. RUSU. *Extracting a data flow analyser in constructive logic*, in "European Symposium on Programming,ESOP'04", LNCS, vol. 2986, Springer-Verlag, February 2004, p. 385–400.

[19] B. GAUDIN, H. MARCHAND. *Modular Supervisory Control of a class of Concurrent Discrete Event Systems*, in "Workshop on Discrete Event Systems, WODES'04", September 2004, p. 181-186.

[20] B. JEANNET, A. LOGINOV, T. REPS, M. SAGIV. *A Relational Approach to Interprocedural Shape Analysis*, in "11th Static Analysis Symposium - SAS 2004, Verona, Italy", LNCS, vol. 3148, Springer-Verlag, August 2004.

[21] B. JEANNET, W. SERWE. *Abstracting Call-Stacks for Interprocedural Verification of Imperative Programs*, in "10th International Conference on Algebraic Methodology And Software Technology AMAST'2004", LNCS, vol. 3116, Springer-Verlag, July 2004.

[22] V. RUSU, H. MARCHAND, V. TSCHAEN, T. JÉRON, B. JEANNET. *From Safety Verification to Safety Testing*, in "The 16th IFIP International Conference on Testing of Communicating Systems (TestCom04), Oxford, UK", LNCS, vol. 2978, Springer-Verlag, March 2004.

Internal Reports

[23] B. GAUDIN, H. MARCHAND. *Supervisory control of Concurrent Discrete Event Systems*, Technical report, n° 1593, IRISA, January 2004.

[24] B. JEANNET, T. JÉRON, V. RUSU, E. ZINOVIEVA. *Symbolic test selection using approximate*

analysis, Accepted for publication in TACAS'05, Technical report, n° 1649, IRISA, October 2004, http://www.irisa.fr/bibli/publi/pi/2004/1649/1649.html.

[25] V. RUSU, H. MARCHAND, T. JÉRON. *Verification and Symbolic Test Generation for Safety Properties*, Technical report, n° 5285, INRIA, August 2004, http://www.inria.fr/rrrt/rr-5285.html.

[26] V. RUSU. *Verifying an ATM Protocol Using a Combination of Formal Techniques*, Technical report, n° 5089, INRIA, January 2004, http://www.inria.fr/rrrt/rr-5089.html.

[27] E. RUTTEN, H. MARCHAND. *Automatic Generation of Safe Handlers for Multi-Task Systems*, Technical report, n° 5345, INRIA, October 2004, http://www.inria.fr/rrrt/rr-5345.html.

Miscellaneous

[28] T. JÉRON, V. TSCHAEN. *Test Generation Engine Documentation*, IST-AGEDIS European contract report, 2004, AGEDIS : Automated Generation and Execution of test suites for DIstributed component-based Software.

Bibliography in notes

[29] I. 9646. *Information Technology - Open Systems Interconnection Conformance Testing Methodology and Framework - Part 1 : General Concept - Part 2 : Abstract Test Suite Specification - Part 3 : The Tree and Tabular Combined Notation (TTCN)*, in "International Standard ISO/IEC 9646-1/2/3", 1992.

[30] F. BOURDONCLE. *Sémantique des langages impératifs d'ordre supérieur et interprétation abstraite*, Ph. D. Thesis, Ecole Polytechnique, Paris, 1992.

[31] Y. BRAVE, M. HEIMANN. *Control of Discrete Event Systems Modeled as hierarchical State Machines*, in "IEEE Transacations on Automatic Control", vol. 38, n° 12, December 1993, p. 1803–1819.

[32] P. CAINES, V. GUPTA, G. SHEN. *The hierarchical control of ST-finite-state machines*, in "Systems and Control Letters", vol. 32, 1997, p. 185-192.

[33] P. COUSOT, R. COUSOT. *Static determination of dynamic properties of programs*, in "2nd Int. Symp. on Programming", Dunod, Paris, 1976.

[34] P. COUSOT, R. COUSOT. *Abstract intrepretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints*, in "Conference Record of the 4th ACM Symposium on Principles of Programming Languages, Los Angeles, CA", January 1977, p. 238-252.

[35] P. D'ARGENIO, B. JEANNET, H.E. JENSEN, K.G. LARSEN. *Reduction and Refinement Strategies for Probabilistic Analysis*, in "Process Algebra and Probabilistic Methods - Performance Modelling and Verification, PAPM-PROBMIV 2002, Copenhagen, Denmark", LNCS, vol. 2399, July 2002, http://www.irisa.fr/prive/bjeannet/djjl02.ps.gz.

[36] J. ESPARZA, D. HANSEL, P. ROSSMANITH, S. SCHWOON. *Efficient Algorithms for Model Checking Pushdown Systems*, in "Proceedings of Computer Aided Verification (CAV'2000)", LNCS, vol. 1855, July

2000.

[37] P. GOHARI-MOGHADAM, W.M. WONHAM. *A linguistic Framework for controller hierarchical DES*, in "4th International Workshop on Discrete Event Systems, Cagliari, Italy", August 1998, p. 207-212.

[38] B. JEANNET. *Dynamic Partitioning In Linear Relation Analysis. Application To The Verification Of Reactive Systems*, in "Formal Methods in System Design", vol. 23, n° 1, July 2003, p. 5–37.

[39] B. JEANNET, P. D'ARGENIO, K.G. LARSEN. *RAPTURE: A tool for verifying Markov Decision Processes*, in "Tools Day, International Conference on Concurrency Theory, CONCUR'02, Brno, Czech Republic", August 2002, http://www.irisa.fr/prive/bjeannet/rapture02.ps.gz.

[40] R.J. LEDUC. *Hierarchical Interface Based Supervisory Control*, Ph. D. Thesis, Dept. of Elec. & Comp. Engrg., Univ. of Toronto, 2002.

[41] H. MARCHAND, O. BOIVINEAU, S. LAFORTUNE. *On Optimal Control of a class of partially-Observed Discrete Event Systems*, in "Automatica", vol. 38, n° 11, October 2002, p. 1935-1943.

[42] S. OWRE, J. RUSHBY, N. SHANKAR, F. VON HENKE. *Formal Verification for Fault-Tolerant Architectures: Prolegomena to the Design of PVS*, in "IEEE Transactions on Software Engineering", vol. 21, n° 2, feb 1995, p. 107-125.

[43] C. PAULIN-MOHRING. *Le système Coq (habilitation thesis, in French)*, Technical report, ENS Lyon, 1997.

[44] P. J. RAMADGE, W. M. WONHAM. *The Control of Discrete Event Systems*, in "Proceedings of the IEEE; Special issue on Dynamics of Discrete Event Systems", vol. 77, n° 1, 1989, p. 81-98.

[45] M. SAGIV, T. REPS, R. WILHELM. *Parametric shape analysis via 3-valued logic*, in "ACM Transactions on Programming Languages and Systems", vol. 24, n° 3, 2002.

[46] M. SAGIV, T. REPS, R. WILHELM. *Solving shape-analysis problems in languages with destructive updating*, in "ACM Transactions on Programming Languages and Systems", vol. 20, n° 1, 1998.

[47] R. SENGUPTA, S. LAFORTUNE. *An Optimal Control Theory for Discrete Event Systems*, in "SIAM Journal on Control and Optimization", vol. 36, n° 2, march 1998.

[48] J. TRETMANS. *Testing Labelled Transition Systems with Inputs and Outputs*, in "8th International Workshop on Protocols Test Systems, Evry - France", September 1995.

[49] J. TRETMANS. *Testing concurrent systems: A formal approach*, in "CONCUR'99", LNCS, n° 1664, 1999, p. 46-65.

[50] K. C. WONG, W. M. WONHAM. *Hierarchical Control of Discrete-Event Systems*, in "Discrete Event Dynamic Systems", vol. 6, 1996, p. 241-273.

[51] W. M. WONHAM, P. J. RAMADGE. *Modular Supervisory Control of Discrete Event Systems*, in "Mathematics of Control Signals and Systems", vol. 1, 1988, p. 13–30.

[52] M.H. DEQUEIROZ, J.E.R. CURY. *Synthesis and implementation of local modular supervisory control for a manufacturing cell*, in "Proceedings of the 6th International Workshop on Discrete Event Systems", October 2002, p. 377-382.