INRIA

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 - Bretagne-Atlantique

THEME COM

Table of contents

1. Team

Research Scientist

Thierry Jéron [Research Director (DR) Inria, HdR]
Nathalie Bertrand [Research Associate (CR) Inria]
Hervé Marchand [Research Associate (CR) Inria]
Vlad Rusu [Research Associate (CR) Inria, partly in DART EPI since Otober 2008, HdR]

Faculty Member

Camille Constant [INRIA, ATER until August 2008]
Tristan Le Gall [MENRT, ATER until August 2008]

Technical Staff

Florimond Ployette [Technical staff, IR (50% with ASCII)]

PhD Student

Jéremy Dubreil [INRIA, since March 2006]
Hatem Hamdi [in collaboration with UNIVERSITY OF SFAX, since October 2005]

Post-Doctoral Fellow

Gwenaël Delaval

Visiting Scientist

Christophe Morvan [Assistant Professor, Univ. de Marne-la-Vallée]

Administrative Assistant

Lydie Mabil [TR INRIA, (80%)]

2. Overall Objectives

2.1. Introduction

The VerTeCs team is focused on the use of formal methods to assess the reliability, safety and security of reactive software systems. By reactive software system we mean a system controlled by software which reacts with its environment (human or other reactive software). 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, or at least detect incorrectness during execution and take appropriate action. For this aim, the VerTeCs team promotes the use of formal methods, i.e. formal specification of software and their required properties and mathematically founded validation methods. Our research covers several validation methods, all oriented towards a better reliability of software systems:

- Verification, which is used during the analysis and design phases, and whose aim is to establish the correctness of specifications with respect to requirements, properties or higher level specifications.
- Control synthesis, which consists in "forcing" (specifications of) systems to stay within desired behaviours by coupling them with a supervisor.
- Conformance testing, which is used to check the correctness of a real system with respect to its specification. In this context, we are interested in model-based testing, and in particular automatic test generation of test cases from specifications.
- Diagnosis and monitoring, which are used during execution to detect erroneous behaviour.
- Combinations of these techniques, both at the methodological level (combining several techniques within formal validation methodologies) and at the technical level (as the same set of formal verification techniques - model checking, theorem proving and abstract interpretation - are required for control synthesis, test generation and diagnosis).

Our research is thus concerned with the development of formal models for the description of software systems, the formalization of relations between software artifacts (e.g. satisfaction, conformance between properties, specifications, implementations), the interaction between these artifacts (modelling of execution, composition, etc). We develop methods and algorithms for verification, controller synthesis, test generation and diagnosis that ensure desirable properties (e.g. correctness, completeness, optimality, etc). We try to be as generic as possible in terms of models and techniques in order to cope with a wide range of application domains and specification languages. Our research has been applied to telecommunication systems, embedded systems, smart-cards application, and control-command systems. We implement prototype tools for distribution in the academic world, or for transfer to the industry.

Our research is based on formal models and our basic tools are **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.

2.2. Highlights of the year

N. Bertrand received the EATCS best theory paper at ETAPS for her work on Decision Problems for Probabilistic Büchi Automata [13] .

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. These models are adapted to testing and controller synthesis.

To model reactive systems in the testing context, we use Input/Output labeled transition systems (IOLTS for short). In this setting, the interactions between the system and its environment (where the tester lies) must be partitioned into inputs (controlled by the environment), outputs (observed by the environment), and internal (non observable) events modeling the internal behavior of the system. The alphabet Λ is then partitioned into $\Lambda_! \cup \Lambda_? \cup \mathcal{T}$ where $\Lambda_!$ is the alphabet of outputs, $\Lambda_?$ the alphabet of inputs, and $\mathcal{T}$ 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 ($\Lambda = \Lambda_O \cup \mathcal{T}$).

In the context of verification, we also use Timed Automata. A timed automaton is a tuple $A = (L, X, E, \mathcal{I})$ where L is a set of locations, X is a set of clocks whose valuations are positive real numbers, $E \subseteq L \times \mathcal{G}(\mathcal{X}) \times 2^X \times L$ is a finite set of edges composed of a source and a target state, a guard given by a finite conjunction of expressions of the form $x \sim c$ where x is a clock, c is a natural number and $\sim \in \{<, \leq, =, \geq, >\}$, a set of resetting clocks, and $\mathcal{I} : \mathcal{L} \rightarrow \mathcal{G}(\mathcal{X})$ assigns an invariant to each location [28]. The semantics of a timed automaton is given by a (infinite states) labelled transition system whose states are composed of a location and a valuation of clocks.

In order to cope with more realistic models, closer to real specification languages, we also need higher level models that consider both control and data aspects. We defined (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. Formally, 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 partitioned 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 (possibly with a polarity reflecting its input/output/internal nature), p is a tuple of communication parameters, G is a guard defined by a predicate on p and V, and A is an assignment 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 (action with valued communication parameter). This (IO)LTS semantics allows us to perform syntactical transformations at the (IO)STS level while ensuring semantical properties at the (IO)LTS level. We also consider extensions of these models with added features such as recursion, fifo channels, etc. 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.
- BDDs (Binary Decision Diagrams) algorithms, for manipulating Boolean formula, and their MTBDDs (Multi-Terminal Decision Diagrams) extension for manipulating more general functions. We use these algorithms for verification and test generation.
- abstract interpretation algorithms, specifically in the abstract domain of convex polyhedra (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 satisfiability of formulas in Presburger arithmetics. We use these algorithms during generation and execution of symbolic test cases.

3.2. Verification

Verification in its full generality consists in checking that a system, which is specified by a formal model, satisfies a required property. Verification takes place in our research in two ways: on the one hand, a large part of our work, and in particular controller synthesis and conformance testing, relies on the ability to solve some verification problems. Many of these problems reduce to reachability and coreachability questions on a formal model (a state s is *reachable from an initial state* 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 are important cases of verification problems, as they correspond to the verification of safety properties.

On the other hand we investigate verification on its own in the context of complex systems. For expressivity purposes, it is necessary to be able to describe faithfully and to deal with complex systems. Some particular aspects require the use of infinite state models. For example asynchronous communications with unknown transfer delay (and thus arbitrary large number of messages in transit) are correctly modeled by unbounded FIFO queues, and real time systems require the use of continuous variables which evolve with time. Apart from these aspects requiring infinite state data structure, systems often include uncertain or random behaviours (such as failures, actions from the environment), which it make sense to model through probabilities. To encompass these aspects, we are interested in the verification of systems equipped with infinite data structures and/or probabilistic features.

When the state space of the system is infinite, or when we try to evaluate performances, standard model-checking techniques (essentially graph algorithms) are not sufficient. For large or infinite state spaces, symbolic model-checking or approximation techniques are used. Symbolic verification is based on efficient representations of set of states and permits exact model-checking of some well-formed infinite-state systems. However, for feasibility reasons, it is often mandatory to make the use of 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 techniques, can deal with complex systems that are beyond the scope of fully automatic methods. For systems with stochastic aspects, a quantitative analysis has to be performed, in order to evaluate the performances. Here again, either symbolic techniques (e.g. by grouping states with similar behaviour) or approximation techniques should be used.

We detail below the three verification topics we are interested in: abstract interpretation, theorem proving and model-checking of infinite state and probabilistic systems.

3.2.1. Abstract interpretation and Data Handling

Most problems in test generation or controller synthesis reduce to state reachability and state coreachability problems which can be solved by fixpoint computations of the form $x = F(x), x \in C$ 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 big change induced by taking into account the data and not only the (finite) control of the systems under study is that the fixpoints become uncomputable. The undecidability is overcome by resorting to approximations, using the theoretical framework of Abstract Interpretation [32]. 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 abstract domain, so that one has 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).

Approximations are conservative so that the obtained result is an upper-approximation of the exact result. In simple cases the state space 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.

3.2.2. Theorem Proving

For verification we also use theorem proving and more particularly the PVS [38] and COQ [39] 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 can also 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.

3.2.3. Model-checking of infinite state and probabilistic systems

Model-checking techniques for finite state probabilistic systems are now quite developed. Given a finite state Markov chain, for example, one can check whether some property holds almost surely (i.e. the set of executions violating the property is negligible), and one can even compute (or at leat approximate as close as wanted) the probability that some property holds. In general, these techniques cannot be adapted to infinite state probabilistic systems, just as model-checking algorithms for finite state systems do not carry over to infinite state systems. For systems exhibiting complex data structures (such as unbounded queues, continuous clocks) and uncertainty modeled by probabilities, it can thus be hard to design model-checking algorithms. However, in some cases, especially when considering qualitative verification, symbolic methods can lead to exact results. Qualitative questions do not aim at computing neither approximating a probability, but are only concerned with almost-sure or non negligible behaviours (that is events either of probability one, or non zero). In some cases, qualitative model-checking can be derived from a combination of techniques for infinite state systems (such as abstractions) with methods for finite state probabilistic systems. However, when one is interested in computing (or rather approximating) precise probability values (neither 0 nor 1), exact methods are scarce. To deal with these questions, we either try to restrict to classes of systems where exact computations can be made, or look for approximation algorithms.

3.3. Automatic Test Generation

We are mainly interested in conformance testing which consists in checking whether a black box implementation under test (the real system that is only known by its interface) behaves correctly with respect to its specification (the reference which specifies the intended behavior of the system). In the line of model-based testing, we use formal specifications and their underlying models to unambiguously define the intended behavior of the system, to formally define conformance and to design test case generation algorithms. The difficult problems are to generate test cases that correctly identify faults (the oracle problem) and, as exhaustiveness is impossible to reach in practice, to select an adequate subset of test cases that are likely to detect faults. Hereafter we detail some elements of the models, theories and algorithms we use.

We use IOLTS (or IOSTS) as formal models for specifications, implementations, test purposes, and test cases. We adapt a well established theory of conformance testing [42], which formally defines conformance as a relation between formal models of specifications and implementations. This conformance relation, called **ioco** compares the visible behaviors (called *suspension traces*) of the implementation I (denoted by $STraces(I)$) with those of the specification S ($STraces(S)$). Suspension traces are sequence of inputs, outputs or quiescence (absence of action denoted by δ), thus abstract away internal behaviors that cannot be observed by testers. Intuitively, I $ioco$ S if after a suspension trace of the specification, the implementation I can only show outputs and quiescences of the specification S. We re-formulated ioco as a partial inclusion of visible behaviors as follows:

$$I \; ioco \; S \Leftrightarrow STraces(I) \cap [STraces(S).\Lambda_!^\delta \setminus STraces(S)] = \varnothing.$$

In other words, suspension traces of I which are suspension traces of S prolongated by an output or quiescence, should still be suspension traces of S.

Interestingly, this characterization presents conformance with respect to S as a safety property of suspension traces of I. The negation of this property is charaterized by a *canonical tester* $Can(S)$ which recognizes exactly $[STraces(S).\Lambda_!^\delta \setminus STraces(S)]$, the set of non-conformant suspension traces. This *canonical tester* also serves as a basis for test selection.

Test cases are processes executed against implementations in order to detect non-conformance. They are also formalized by IOLTS (or IOSTS) with special states indicating *verdicts*. The execution of test cases against implementations is formalized by a parallel composition with synchronization on common actions. A *Fail* verdict means that the IUT is rejected and should correspond to non-conformance, a *Pass* verdict means that the IUT exhibited a correct behavior and some specific targeted behaviour has been observed, while an *Inconclusive* verdict is given to a correct behavior that is not targeted.

Test suites (sets of test cases) are required to exhibit some properties relating the verdict they produce to the conformance relation. Soundness means that only non conformant implementations should be rejected by a test suite and exhaustiveness means that every non conformant implementation may be rejected by the test suite. Soundness is not difficult to obtain, but exhaustiveness is not possible in practice and one has to select test cases.
Test selection is often based on the coverage of some criteria (state coverage, transition coverage, etc). But test cases are often associated with *test purposes* describing some abstract behaviors targeted by a test case. In our framework, test purposes are specified as IOLTS (or IOSTS) associated with marked states or dedicated variables, giving them the status of automata or observers accepting runs (or sequences of actions or suspension traces). Selection of test cases amounts to selecting traces of the canonical tester accepted by the test purpose. The resulting test case is then both an observer of the negation of a safety property (non-conformance wrt. S), and an observer of a reachability property (acceptance by the test purpose). Selection can be reduced to a model-checking problem where one wants to identify states (and transitions between them) which are both reachable from the initial state and co-reachable from the accepting states. We have proved that these algorithms ensure soundness. Moreover the (infinite) set of all possibly generated test cases is also 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.

Our first test generation algorithms are based on enumerative techniques, thus adapted to IOLTS models, and optimized to fight the state-space explosion problem. On-the-fly algorithms where designed and implemented in the TGV tool (see 5.1), which consist in computing co-reachable states from a target state during a lazy exploration of the set of reachable states in a product of the specification and the test purpose [4]. However, this enumerative technique suffers from some limitations when specification models contain data.

More recently, we have explored symbolic test generation techniques for IOSTS specifications [41]. The 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 are kept in a symbolic form. Test selection is still based on test purposes (also described as IOSTS) and involves syntactical transformations of IOSTS models that should ensure properties of their IOLTS semantics. However, most of the operations involved in test generation (determinisation, reachability, and coreachability) become undecidable. For determinisation 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, syntactical transformations are guided by over-approximations using abstract interpretation techniques. Nevertheless, these over-approximations still ensure soundness of test cases [5]. These techniques are implemented in the STG tool (see 5.2), with an interface with NBAC used for abstract interpretation.

3.4. Controller Synthesis

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 [40]. The models used are LTSs and the associated languages, 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 problem: building a supervisor that prevents the system from violating a property. Several kinds of properties can be ensured such as reachability, invariance (i.e. safety), 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.

The Supervisory Control Theory overview. Supervisory control theory deals with control of Discrete Event Systems. In this theory, the behavior of the system S is assumed not to be fully satisfactory. Hence, it has to be reduced by means of a feedback control (named Supervisor or Controller) in order to achieve a given set of requirements [40]. Namely, if S denotes the specification of the system and Φ is a safety property that has to be ensured on S (i.e. $S \neg\models \Phi$), the problem consists in computing a supervisor $\mathcal{C}$, such that

$$S \| \mathcal{C} \models \Phi \tag{1}$$

where $\|$ is the classical parallel composition between two LTSs. Given S, some events of S are said to be uncontrollable (Σ_{uc}), i.e. the occurrence of these events cannot be prevented by a supervisor, while the others are controllable (Σ_c). It means that all the supervisors satisfying (1) are not good candidates. In fact, the behavior of the controlled system must respect an additional condition that happens to be similar to the $ioco$ conformance relation that we previously defined in 3.3. This condition is called the *controllability condition* and is defined as follows.

$$\mathcal{L}(S \| \mathcal{C}) \Sigma_{uc} \cap \mathcal{L}(S) \subseteq \mathcal{L}(S \| \mathcal{C}) \tag{2}$$

Namely, when acting on S, a supervisor is not allowed to disable uncontrollable events. Given a safety property Φ, that can be modeled by an LTS A_Φ, there actually exist many different supervisors satisyfing both (1) and (2). Among all the valid supervisors, we are interested in computing the supremal one, ie the one that restricts the system as few as possible. It has been shown in [40] that such a supervisor always exists and is unique. It gives access to a behavior of the controlled system that is called the supremal controllable sub-language of A_Φ w.r.t. S and Σ_{uc}. In some situations, it may also be interesting to force the controlled system to be non-blocking (See [40] for details).

The underlying techniques are similar to the ones used for Automatic Test Generation. It consists in computing a product between the specification and A_Φ and to remove the states of the obtained LTS that may lead to states that violate the property by triggering only uncontrollable events.

Control of Structured Discrete Event System. In many applications and control problems, LTS 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 system 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 [3].

Similarly, in order to take into account nested behaviors, some techniques based on model aggregation methods [43], [31] have been proposed to deal with hierarchical control problems. Another direction has been proposed in [30]. Brave and Heimann in [30] introduced Hierarchical State Machines which constitute a simplified version of the STATECHARTS. Compared to the classical state machines, they add concurrency and hierarchy features. Some other works dealing with control and hierarchy can be found in [34], [37]. This is the direction we have chosen in the VERTECS Team [33].

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 intended to be as generic as possible. 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 [35], 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 increase the need for formal techniques in order to ensure the compositionality of components, by verification and testing. Our techniques, by their genericity and adaptativity, have also proved useful at different levels of these methodologies, from component testing to system testing. The telecommunication industry now also tries to provide more and more services to the users. These services must be validated. We are involved with France Telecom R & D in a project on the validation of vocal services. Very recently, we also started to study the impact of our test generation techniques in the domain of network security. More specifically, we believe that testing that a network or information system meets its security policy is a major concern, and complements other design and verification techniques.

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 replaces 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 allow 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 of 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, and 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: *Conformance Testing*, *IF*, *Lotos*, *SDL*, *TGV*, *TTCN*, *UML.*

Participant: Thierry Jéron [contact].

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 focused 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 determinisation 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 transferred in the SDL ObjectGéode toolset as part of the TestComposer tool. A new version of TGV has been 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 TGV-IF extends the previous one with new functionalities for coverage based test generation combined with test purposes based test generation. This year some CADP libraries used in TGV-IF have been replaced with STL libraries in order to gain some independency with respect to CADP and allow easier porting. 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. TGV-IF is currently being deposited at APP.

5.2. STG

Keywords: *Conformance Testing*, *Symbolic Testing*, *Symbolic Verification*.

Participants: Vlad Rusu [contact], Florimond Ployette, Thierry Jéron.

STG (Symbolic Test Generation) 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, determinisation, 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 (i.e. during execution) using a constraint solver. The first version of STG was developed in C++, using Omega as constraint solver during execution. This version has been deposit at APP (IDDN.FR.001.510006.000.S.P.2004.000.10600).

A new version in OCaml has been developed in the last two years. This version is more generic and will serve as a library for symbolic operations on IOSTS. Most functionalities of the C++ version have been re-implemented. Also a new translation of abstract test cases into Java executable tests has been developed, in which the constraint solver is LUCKYDRAW (VERIMAG). This version has also been deposit at APP and is available for download on the web as well as its documentation and some examples.

5.3. 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 set of solutions, 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 [6], [29]. SIGALI is connected with the Polychrony environment (ESPRESSO project-team) as well as the Matou environment (VERIMAG), 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).

5.4. Ctrl-S

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

Participant: Hervé Marchand [contact].

CTRL-S is a tool dedicated to the control and simulation of structured discrete event systems. CTRL-S is a graphical tool connected with Oris dedicated to (1) the synchronous products of finite state machines and the simulation of the global behavior, and (2) the integration of toolboxes that compute their controllers. It now encompasses the former tool Syntool that was developped in our team during the past years.

6. New Results

6.1. Verification and Abstract Interpretation

Keywords: *Abstract Interpretation*, *Communicating Finite State Machines*, *FIFO channels*, *Reachability Analysis*, *rewriting logic*, *theorem proving*.

6.1.1. Verification of Communication Protocols using Abstract Interpretation of FIFO queues

Participants: Tristan Le Gall, Thierry Jéron.

Many scientific studies analysed the FIFO channel systems, but none offered a fully satisfying solution. We propose to tackle this problem within the abstract interpretation framework, by defining some abstract lattices adapted to this kind of systems. We first consider systems with a finite alphabet of messages, then we consider more complex systems, with an infinite alphabet of messages. This leads us to define and to study a new kind of automata: the lattice automata. Those automata are also useful for the analysis of programs with a call stack. This work has been done in the context of Tristan Le Gall's PhD [9] under the supervision of Bertrand Jeannet and Thierry Jéron.

6.1.2. Executable algebraic semantics for conformance and model transformations in the MOF framework

Participant: Vlad Rusu.

We give executable formal semantics to the notions of conformance and model transformations in the MOF framework. Conformance is relation between a model at a given MOF level and a meta-model at the next MOF level, possibly constrained by OCL invariants. Model transformations are functions that map a source model, which conforms to a source meta-model (+ OCL invariants), to a target model, which must conform to a target meta-model (+ OCL invariants). Our approach consists in representing models, meta-models, and OCL invariants into Membership Equational Logic (MEL), a logic implemented in the Maude tool. Then, conformance is defined as an inclusion between the semantics of a model and that of a meta-model (+ OCL), whereas model transformations are defined as functions between the semantics of two meta-models(+ OCL) involved in the transformation. Conformance can be checked automatically, and model transformations can be automatically [18]

6.1.3. Decision Problems for Probabilistic Büchi Automata

Participant: Nathalie Bertrand.

Probabilistic Büchi automata (PBA) are finite-state acceptors for infinite words where all choices are resolved by fixed distributions and where the accepted language is defined by the requirement that the measure of the accepting runs is positive. The main contribution of this paper is a complementation operator for PBA and a discussion on several algorithmic problems for PBA. All interesting problems, such as checking emptiness or equivalence for PBA or checking whether a finite transition system satisfies a PBA-specification, turn out to be undecidable. An important consequence of these results are several undecidability results for stochastic games with incomplete information, modelled by partially-observable Markov decision processes and omega-regular winning objectives. Furthermore, we discuss an alternative semantics for PBA where it is required that almost all runs for an accepted word are accepting, which turns out to be less powerful, but has a decidable emptiness problem [13].

[13] obtained the EATCS best theory paper at ETAPS.

6.1.4. Model Checking of Timed Automata

Participant: Nathalie Bertrand.

6.1.4.1. Almost-Sure Model Checking of Infinite Paths in One-Clock Timed Automata.

We define here two relaxed semantics (one based on probabilities and the other one based on the topological notion of largeness) for LTL over infinite runs of timed automata which rule out unlikely sequences of events. We prove that these two semantics match in the framework of single-clock timed automata (and only in that framework), and prove that the corresponding relaxed model-checking problems are PSPACE-Complete. Moreover, we prove that the probabilistic non-Zenoness can be decided for single-clock timed automata in NLOGSPACE [12].

6.1.4.2. Quantitative Model-Checking of One-Clock Timed Automata under Probabilistic Semantics

In [12], a probabilistic semantics for timed automata has been defined in order to rule out unlikely (sequences of) events. The qualitative model-checking problem for LTL properties has been investigated, where the aim is to check whether a given LTL property holds with probability 1 in a timed automaton, and solved for the class of single-clock timed automata. In this paper, we consider the quantitative model-checking problem for omega-regular properties: we aim at computing the exact probability that a given timed automaton satisfies an omega-regular property. We develop a framework in which we can compute a closed-form expression for this probability; we furthermore give an approximation algorithm, and finally prove that we can decide the threshold problem in that framework [14].

6.2. Test Generation on Enumerative and Symbolic Models

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

6.2.1. Integrating formal verification and conformance testing for reactive systems

Participants: Camille Constant, Thierry Jéron, Hervé Marchand, Vlad Rusu.

In this work, we describe a methodology integrating verification and conformance testing for the formal validation of reactive systems. A specification of a system - an extended input-output automaton, which may be infinite-state - and a set of safety properties ("nothing bad ever happens") and possibility properties ("something good may happen") are assumed. The properties are first tentatively verified on the specification using automatic techniques based on approximated state-space exploration, which are sound, but, as a price to pay for automation, are not complete for the given class of properties. Because of this incompleteness and of state-space explosion, the verification may not succeed in proving or disproving the properties. However, even if verification did not succeed, the testing phase can proceed and provide useful information about the implementation. Test cases are automatically and symbolically generated from the specification and the properties, and are executed on a black-box implementation of the system. The test execution may detect violations of conformance between implementation and specification; in addition, it may detect violation/satisfaction of the properties by the implementation and by the specification. In this sense, testing completes verification. The approach is illustrated on a Bounded Retransmission Protocol [19] This work is also part of Camille Constant's PhD [8].

6.2.2. *Automatic test generation from interprocedural specifications*

Participants: Camille Constant, Thierry Jéron.

This work done addresses the problem of automatic test case generation for testing the conformance of a reactive implementation with respect to a recursive interprocedural specification. The test generation method we propose is based on coreachability analysis, which allows deciding whether and how the test purpose can still be satisfied. However, although it is possible to carry out an exact analysis, the inability of test cases to inspect their own stack prevents them from fully using the coreachability information. We discuss this partial observation problem, its consequences, and how to minimise its impact. Finally, we experiment these methods of test generation on several examples and a case study. This work is also part of Camille Constant's PhD [8] and done under the co-supervision of Bertrand Jeannet (Pop Art EPI, Inria Rhônes-Alpes).

6.2.3. *Application to security*

Participants: Jérémy Dubreil, Thierry Jéron, Hervé Marchand.

While a lot of work has been done on formal verification of security, in particular for cryptographic protocols, very little has been done on formal security testing. As a consequence, testing security often resort on the expert knwoledge and leads to ad hoc solutions. The general challenge is to study how formalization of security policies and information systems can help in automatically (or systematically) performing security testing. Several approaches are already investigated. In the context of RNRT Politess, we first investigate the use of test generation and controller synthesis techniques for the testing of security policies. We assume the existence of a model of the system and consider two kinds of properties: integrity properties and confidentiality properties. We first outline the methodology allowing to automatically compute access controls ensuring these two kinds of properties. We then show how to derive testers that not only test the security properties and the conformance of the implementation, but try to test the access controls that have been plugged with the implementation in order to ensure security properties [27].

Further, in collaboration with Loic Helouet (Distribcom EPI), We describe a technique to test if a covert channel detected using a formal model is effective in an implementation. The technique consists in testing whether the observations of the implementation under test conform to the input/outputs behavior of a covert channel contained in the model [26].

6.3. Controller Synthesis

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

6.3.1. *Control of Infinite Symbolic Transitions Systems under Partial Observation*

Participants: Tristan Le Gall, Hervé Marchand.

Following our preliminary result [36], we have proposed algorithms for the synthesis of memoryless controllers through partial observation of infinite state systems modelled by STS. We provide models of safe controllers both for potentially blocking and non blocking controlled systems. To obtain algorithms for this problems, we use abstract interpretation techniques which provide overapproximations of the transitions set to disable. To our knowledge, with the hypotheses taken, the improved version of our algorithm provides a better solution than what was previously proposed in the literature. Our tool SMACS allowed us to make an empirical validation of our methods to show their feasibility and usability [24]. This work has been done in cooperation with T. Massart and G. Kalyon (Université libre de Bruxelles, Belgium).

6.3.2. Discrete controller synthesis for modular reactive systems

Participants: Gwenaël Delaval, Hervé Marchand.

We here focused on the exploitation of particularities of modular reactive systems for the application of discrete controller synthesis (DCS). We have proposed a schema of integration of DCS techniques into the modular compilation of an extended synchronous language, following the methodology described in [15]. In this extended language, the modularity is expressed by nodes, representing components associated with modular synthesis objectives ; we can then obtain, by application of DCS tools on these components, some synchronous controllers controlling parts of programs. In this framework, we implemented a translation schema of a subset of the Lucid Synchrone language into dynamic systems, for further application of Sigali, as DCS tool. Future work will consist in applying decentralized control methods, together with modular distribution of synchronous programs, in order to obtain automatically, from an annotated synchronous program, a distributed controlled system.

This work is part of the post-doc of Gwenaël Delaval and a cooperative work between the Pop Art EPI (Inria Rhônes-Alpes) and the VerTeCs EPI (Inria Rennes).

6.3.3. Opacity Enforcing Control Synthesis

Participants: Jérémy Dubreil, Hervé Marchand.

Given a system modeled by a finite transition system and a secret modeled by a regular predicate, we address the problem of computing a controller enforcing the opacity (a particular notion of confidentiality) of the secret against an attacker (that partially observes the system), supposedly trying to push the system to reveal the secret. Assuming that the controller can only control a subset of the events it observes (possibly different from the ones of the attacker), we show that an optimal control always exists and provide sufficient conditions under which it is regular and effectively computable. These conditions rely on the inclusion relationships between the observable alphabets of the attacker and the controller and the controllable alphabet [16]. This work has been done in cooperation with Ph. Darondeau (S4 EPI).

6.4. Diagnosis of discrete event systems

Keywords: *Diagnosis*, *Discrete event system*, *Information flow*, *Security*.

6.4.1. Predictability of Sequence Patterns in Discrete Event Systems

Participants: Thierry Jéron, Hervé Marchand.

Following our preliminary results on diagnosis of discrete event systems, we studied in [17] the problem of predicting the occurrences of a pattern in a partially-observed discrete-event system. The system is modeled by a labeled transition system. The pattern is a set of event sequences modeled by a finite-state automaton. The occurrences of the pattern are predictable if it is possible to infer about any occurrence of the pattern before the pattern is completely executed by the system. An off-line algorithm to verify the property of predictability is presented. The verification is polynomial in the number of states of the system. An on-line algorithm to track the execution of the pattern during the operation of the system is also presented. This algorithm is based on the use of a diagnoser automaton. The results are illustrated using an example from computer systems. This work has been done in cooperation with S. Lafortune and S. Genc (University of Michigan, USA).

6.4.2. Diagnosis of Pushdown Systems

Participant: Christophe Morvan.

On-line applications, such as diagnosis, involve partial observation of dynamical discrete event systems that raises worth understanding theoretical questions. Whereas the theory of diagnosis for finite state systems is well understood and yields computable solutions, thesis for infinite state systems are very few and mostly display undecidability results. In [25], we consider the infinite hierarchy of higher order pushdown systems together with the two the important problems of diagnosability and bounded response delay. We establish that diagnosability is decidable for arbitrary sub-classes of visibly higher order pushdown systems whenever unobservable events the leave the stacks unchanged; when this restriction is relaxed, diagnosability becomes undecidable already at the first level of the hierarchy. Regarding the bounded response delay problem, we show how the finiteness problem for visibly higher order pushdown languages is related to the former.

This work was done in collaboration with S. Pinchinat (S4 EPI).

6.4.3. Construction of monitor for the supervision of security properties

Participants: Jérémy Dubreil, Thierry Jéron, Hervé Marchand.

Regarding security, besides our work on test generation and controller synthesis for security properties, we have been interested in constructing monitors for the detection of confidential information flow in the context of partially observable discrete event systems. We focus on the case where the secret information is given as a regular language. We first characterize the set of observations allowing an attacker to infer the secret behaviors. We consider the general case where the attacker and the administrator have different partial views of the system. Further, based on the diagnosis of discrete event systems, we provide necessary and sufficient conditions under which detection and prediction of secret information flow can be ensured and a construction of a monitor ensuring this task [23].

7. Other Grants and Activities

7.1. National Grants & Contracts

7.1.1. RNTL TesTec: Test of Real-time and critical embedded System

Participants: Nathalie Bertrand, Thierry Jéron, Hervé Marchand, Vlad Rusu.

The TESTEC project is an industrial research project that gathers two companies: an end-user (EDF R&D) and one software editor for embedded real-time systems and automation systems (Geensys), and four laboratories from automation engineering and computer science (I3S, INRIA Rennes, LaBRI, LURPA). This project focuses on automatic generation and execution of tests for the class of embedded real-time systems. They are highly critical. Such systems can be found in many industrial domains, such as energy, transport systems. More precisely the project TESTEC will address two crucial technological issues:

- optimisation of tests generation techniques for large size systems, in particular by an explicit modelling of time and by simultaneous management of continuous and discrete variables in hybrid applications;
- reduction of the size of the tests derived from specification models by using the results of formal verification of implementation models.

The overall aim of this project is to propose a software tool for generation and execution of tests; this tool will be based on an existing environment for embedded systems design and will implement the scientific results of the project.

7.1.2. RNRT POLITESS: Security Policies for Network Information Systems: Modeling, Deployment, Testing and Supervision

Participants: Jérémy Dubreil, Hatem Hamdi, Thierry Jéron, Hervé Marchand, Vlad Rusu.

The POLITESS project (http://www.rnrt-politess.info/) [2006-2008] involves GET (INT Evry and ENST Rennes), INPG-IMAG (LSR and VERIMAG laboratories), France Telecom R&D Caen, Leyrios Technologies, SAP Research, AQL Silicomp Rennes and Irisa. In a sense, this project is an extension of the Potestat project. The objective of the project is to study and provide methodological guidelines and software solutions for a formal approach to security of networks. This encompasses the specification of high level security policies with clear semantics, their deployment on the network in terms of security artifacts and the analysis of this deployment, testing and monitoring of security based on models of security policies and abstract models of networks. Our team is involved in several activities, in particular in modelling (defining adequate models for both the system and security policies), testing (modelling security testing, test generation/selection), supervision (intrusion detection, diagnosis) and case studies.

7.2. European and International Grants

7.2.1. ARTIST2 Network of Excellence

Participants: Thierry Jéron, Hervé Marchand, Vlad Rusu.

We are partners of the ARTIST2 Network of Excellence on Embedded Systems (http://www.artist-embedded.org/), involved in the Testing and Verification cluster with Brics in Aalborg (DK), University of Twente (NL), University of Liège (B), Uppsala (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 verification tools developed by partners will be made available via a web-portal and with dedicated verification servers.

In ARTIST2, the main role of VERTECS is to integrate our research on testing and test generation based on symbolic transition systems with other works based on timed models.

7.2.2. Artist Design Network of Excellence

Participants: Nathalie Bertrand, Thierry Jéron, Hervé Marchand, Vlad Rusu.

The central objective for ArtistDesign http://www.artist-embedded.org/artist/-ArtistDesign-Participants-.html is to build on existing structures and links forged in Artist2, to become a virtual Center of Excellence in Embedded Systems Design. This will be mainly achieved through tight integration between the central players of the European research community. Also, the consortium is smaller, and integrates several new partners. These teams have already established a long-term vision for embedded systems in Europe, which advances the emergence of Embedded Systems as a mature discipline.

The research effort aims at integrating topics, teams, and competencies, grouped into 4 Thematic Clusters: “Modelling and Validation”, “Software Synthesis, Code Generation, and Timing Analysis”, “Operating Systems and Networks”, “Platforms and MPSoC”. “Transversal Integration” covering both industrial applications and design issues aims for integration between clusters.

The Vertecs EPI is a partner of the “Validation” activity of the “Modeling and Validation” cluster. The objective is to address the growth in complexity of future embedded products while reducing time and cost to market. This requires methods allowing for early exploration and assessment of alternative design solutions as well as efficient methods for verifying final implementations. This calls for a range of model-based validation techniques ranging from simulation, testing, model-checking, compositional techniques, refinement as well as abstract interpretation. The challenge will be in designing scalable techniques allowing for efficient and accurate analysis of performance and dependability issues with respect to the various types of (quantitative) models considered. The activity brings together the leading teams in Europe in the area of model-based validation.

7.2.3. Combest. European Strep Project

Participant: Nathalie Bertrand.

We are partners of the Combest European Strep Project http://www.combest.eu/home/. The aim of this project is to provide a theoretical framework as well as implemented methods and tools for the component-based design of embedded systems. Our role in Combest is to work on timed components, and more precisely develop a theory around timed modal specifications.

7.3. Collaborations

7.3.1. Collaborations with other INRIA Project-teams

We collaborate with several Inria project-teams. We collaborate with the ESPRESSO EPI for the development of the SIGALI tool inside the Polychrony environment. With the POP ART EPI on the use of the controller synthesis methodology for the control of control-command systems (e.g. robotic systems). With DISTRIBCOM on security testing in the context of the Politess grant. With the S4 EPI on the use of control, game theory and diagnosis for test generation as well as on the study of timed modal specifications and in the context of the Combest grant. With the VASY EPI on the use of CADP libraries in TGV and the distribution of TGV in the CADP toolbox.

7.3.2. Collaborations with French Research Groups outside INRIA

Our main collaborations in France are with LIG (Vasco teams and Verimag) in Grenoble in the context of the RNRT Politess grant, 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. We also work in collaboration with the LSV Cachan on topological and probabilistic semantics for timed automata. With LURPA Cachan, LaBRI Bordeaux and I3S Nice we collaborate on testing control-command systems.

7.3.3. International Collaborations

ENIS Sfax in Tunisia (M. Tahar Bhiri) on security testing and (Maher Ben Jemaa and Moez Krichen) on testing embedded systems. Thierry Jéron is co-supervisor of a PhD student Hatem Hamdi working on robustness and security testing.

Université Libre Bruxelles in Belgium on testing and control of symbolic transitions systems. Thierry Massart visited us for one month in April 2008, and Gabriel Kalyon visited us for 3 months (March to May) and both one week in November.

Institute of Mathematics, Czech Academy of Sciences (Jan Komenda) on supervisory control of concurrent systems.

Technische Universität Dresden (Prof. Christel Baier) on verification of probabilistic systems.

Université Mons-Hainaut (Prof. Thomas Brihaye) on verification of timed systems.

University of Madrid (Prof. Manuel Clavel) on theorem proving for rewriting logic.

University of Michigan in USA (Prof. Stéphane Lafortune) on control and diagnosis of discrete event systems.

Federal University of Campina Grande in Brazil (Prof. Patrícia D. L. Machado) Sampaio) on test case generation, selection and abstraction for embedded real-time systems.

8. Dissemination

8.1. University courses

C. Constant is teaching in License and Master at the University of Rennes 1 (96h/year).

J. Dubreil is teaching in INSA of Rennes (40h in 2007-2008), on the algorithm introduction with Scheme.

T. Jéron is teaching in Model-based Testing in Research Master of Computer Science at the University of Rennes 1.

T. Le Gall is teaching in License and Master at the Univeristy of Rennes 1 (96h/year).

C. Morvan is teaching in License and Master at the University of Rennes 1 (192h/year).

8.2. PhD Thesis and Trainees

PhD. thesis defended in 2008:

Camille Constant: "*Verification and symbolic test generation for reactive systems*", November 2008.

Tristan Le Gall: "*Abstract lattice of fifo channels for verification and control synthesis*", July 2008.

Current PhD. thesis:

Hatem Hamdi: "*Testing of network security*", In collaboration with University of Sfax, third year.

Jérémy Dubreil: "*Formal methods for testing and monitoring security of open networks*", third year.

Trainees 2007-2008:

Wassim Wehbi: "*A tool for the construction of monitors for the supervision of security properties*", Internship-ESIB (Liban) (3 months).

8.3. Scientific animation

Nathalie Bertrand was PC member of QAPL'08 workshop. She was invited to give seminars on "Probabilistic Büchi automata" at LSV Cachan and LaBRI Bordeaux (beginning of 2008), on "Probabilistic Games on Lossy Channel Systems" during the GT Jeux meeting in Bordeaux (june 2008), and on "Verification of Probabilistic Systems" at ENS Cachan Bretagne (september 2008).

Thierry Jéron was PC member of Testcom/Fates'08 (Tokyo, Japan) in June 2008, IEEE ICST 2008 (Lillehammer, Norway) in April 2008, and PC member for the special number on Components, Services and Aspects of the journal L'Objet. He is member of the steering committee and co-organiser of Movep 2008 (Orléans) in June 2008. He gave an invited talk at the Brazilian Symposium on Formal Methods (SBMF2008) in Salvador de Bahia (Brazil) in August 2008. He was reviewer of the PhD defense of Muzzammil Shahbaz (INPG, Grenoble, December 2008). He is member of the IFIP Working Group 10.2 on Embedded Systems (http://jerry.c-lab.de/ifip-wg-102/).

Hervé Marchand was PC member of the Wodes'08, ICINCO'08 and Vecos'08 Conferences. He is member of the IFAC Technical Committees (TC 1.3 on Discrete Event and Hybrid Systems) for the 2005-2008 triennium. He is PC member of the forthcoming ICINCO'09 and MSR'09 conferences. He was member of the PhD commitee of Loic Strus (Univ. Joseph Fourier, Grenoble, September 2008). He was member of the "Commissions de Spécialistes 27e section" at the University of Rennes 1 until 2008. He visited ULB (Bruxelles) for one week in October 2008 and the Ecole Nationale des Sciences de l'Informatique (Tunis) for one week in December 2008.

Christophe Morvan was invited to give a seminar on "Regular graphs : a perfect model for infinite state systems?" in the seminar CFV, Belgium (October 2008).

Vlad Rusu Vlad Rusu gave invited talks in at the University of Madrid in September 2008, and at Inria Lille in February 2008. He organized the EJCP (Ecole Jeunes Chercheurs en Programmation) in June 2008.

Gwenaël Delaval gave a talk during the workshop SYNCHRON 2008 on modular synthesis of reactive systems.

Tristan Le Gall was invited to give seminars on "Lattice automata and their application to verification of systems with stacks and queues" in LABRI Bordeaux (April 2008) and ULB Bruxelles (June 2008).

Jérémy Dubreil gave a talk on "Opacity enforcing by synthesis" during the school Movep 2008.

9. Bibliography

Major publications by the team in recent years

[1] C. BAIER, N. BERTRAND, PH. SCHNOEBELEN. *Verifying nondeterministic probabilistic channel systems against ω-regular linear-time properties*, in "ACM Transactions on Computational Logic", vol. 9, n^{o} 1, 2007.

[2] C. CONSTANT, T. JÉRON, H. MARCHAND, V. RUSU. *Integrating formal verification and conformance testing for reactive systems*, in "IEEE Transactions on Software Engineering", vol. 33, n^{o} 8, August 2007, p. 558-574.

[3] B. GAUDIN, H. MARCHAND. *An Efficient Modular Method for the Control of Concurrent Discrete Event Systems: A Language-Based Approach*, in "Discrete Event Dynamic System", vol. 17, n^{o} 2, 2007, p. 179-209.

[4] 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, October 2004.

[5] B. JEANNET, T. JÉRON, V. RUSU, E. ZINOVIEVA. *Symbolic Test Selection based on Approximate Analysis*, in "11th Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'05), Volume 3440 of LNCS, Edinburgh (Scottland)", April 2005, p. 349-364, http://www.irisa.fr/vertecs/Publis/Ps/tacas05.pdf.

[6] 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^{o} 4, Octobre 2000, p. 347-368, http://www.irisa.fr/vertecs/Publis/Ps/2000-J-DEDS.pdf.

[7] V. RUSU. *Verifying an ATM Protocol Using a Combination of Formal Techniques*, in "Computer Journal", vol. 49, n^{o} 6, November 2006, p. 710–730.

Year Publications

Doctoral Dissertations and Habilitation Theses

[8] C. CONSTANT. *Génération automatique de tests pour des modèles avec variables ou récursivité*, Ph. D. Thesis, Ecole doctorale Matisse, Université de Rennes I, Novembre 2008.

[9] T. LE GALL. *Abstract Lattices for the Verification of Systems with Queues and Stacks*, Ph. D. Thesis, Ecole doctorale Matisse, Université de Rennes 1, July 2008.

Articles in International Peer-Reviewed Journal

[10] J. KOMENDA, J. VAN SCHUPPEN, B. GAUDIN, H. MARCHAND. *Supervisory Control of Modular Systems with Global Specification Languages*, in "Automatica", vol. 44, 2008, p. 1127–1134, http://www.irisa.fr/vertecs/Publis/Ps/2008-Automatica.pdf.

Invited Conferences

[11] T. JÉRON. *Symbolic model-based test selection*, in "Proceedings of the Brazilian Symposium on Formal Methods (SBMF 2008), Salvador, Bahia, Brazil", P. MACHADO, A. ANDRADE, A. DURAN (editors), Electronic version will appear in ENTCS, UFBA, August 2008, p. 17–32.

International Peer-Reviewed Conference/Proceedings

[12] C. BAIER, N. BERTRAND, P. BOUYER, T. BRIHAYE, M. GRÖSSER. *Almost-Sure Model Checking of Infinite Paths in One-Clock Timed Automata*, in "Proceedings of the 23rd Annual IEEE Symposium on Logic in Computer Science (LICS'08), Pittsburgh, PA, USA", IEEE Computer Society Press, June 2008.

[13] C. BAIER, N. BERTRAND, M. GRÖSSER. *On Decision Problems for Probabilistic Büchi Automata*, in "Proceedings of the 11th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'08), Budapest, Hungary", Lecture Notes in Computer Science, Springer, March 2008, http://www.irisa.fr/prive/nbertran/BBG-fossacs08.pdf.

[14] N. BERTRAND, P. BOUYER, T. BRIHAYE, N. MARKEY. *Quantitative Model-Checking of One-Clock Timed Automata under Probabilistic Semantics*, in "Proceedings of the 5th International Conference on the Quantitative Evaluation of SysTems (QEST'08), Saint Malo, France", IEEE Computer Society Press, September 2008.

[15] G. DELAVAL. *Modular Distribution and Application to Discrete Controller Synthesis*, in "International Workshop on Model-driven High-level Programming of Embedded Systems (SLA++P'08), Budapest, Hungary", April 2008.

[16] J. DUBREIL, P. DARONDEAU, H. MARCHAND. *Opacity Enforcing Control Synthesis*, in "Workshop on Discrete Event Systems, WODES'08, Gothenburg, Sweden", A shorter version also appeared in the proceeding of Movep'08, March 2008, http://www.irisa.fr/vertecs/Publis/Ps/2008-Wodes-Opacity.pdf.

[17] T. JÉRON, H. MARCHAND, S. GENC, S. LAFORTUNE. *Predictability of Sequence Patterns in Discrete Event Systems*, in "IFAC World Congress, Seoul, Korea", July 2008, http://www.irisa.fr/vertecs/Publis/Ps/2008-IFAC-Predictability.pdf.

Workshops without Proceedings

[18] M. CLAVEL, M. EGEA, V. RUSU. *Executable Algebraic Semantics for Refinement and Conformance in the MOF Framework*, in "1st International Workshop on Algebraic Methods in Model-Based Software Engineering (AMMSE 2008), Universidad Complutense, Madrid, Spain", 2008.

Scientific Books (or Scientific Book chapters)

[19] C. CONSTANT, T. JÉRON, H. MARCHAND, V. RUSU. *2*, in "Validation of Reactive Systems", S. MERZ, N. NAVET (editors), Hermès Science, January 2008, p. 51-76.

Books or Proceedings Editing

[20] J.-M. COUVREUR, T. JÉRON (editors). *Proceedings of 8th School on MOdeling and VErifying parallel Process (MOVEP'08)*, 2008.

Research Reports

[21] C. BAIER, N. BERTRAND, P. BOUYER, T. BRIHAYE, M. GRÖSSER. *A Probabilistic Semantics for Timed Automata*, Research Report, n° LSV-08-13, Laboratoire Spécification et Vérification, ENS Cachan, France, April 2008, http://www.lsv.ens-cachan.fr/Publis/RAPPORTS_LSV/PDF/rr-lsv-2008-13.pdf.

[22] J. DUBREIL, P. DARONDEAU, H. MARCHAND. *Opacity Enforcing Control Synthesis*, Technical report, n° 1887, IRISA, March 2008, http://www.irisa.fr/vertecs/Publis/Ps/PI-Opacity.pdf.

[23] J. DUBREIL, T. JÉRON, H. MARCHAND. *Monitoring Information flow by Diagnosis Techniques*, Technical report, n° 1901, IRISA, August 2008, http://www.irisa.fr/vertecs/Publis/Ps/2008-PI-1901-Detec-Opacity.pdf.

[24] G. KALYON, T. LE GALL, H. MARCHAND, T. MASSART. *Control of Infinite Symbolic Transitions Systems under Partial Observation*, Technical report of the verification group, n° 103, Univeristé Libre de Bruxelles, October 2008.

[25] C. MORVAN, S. PINCHINAT. *Diagnosis of Pushdown Systems*, Technical report, n° 1904, IRISA, October 2008, ftp://ftp.irisa.fr/techreports/2008/PI-1904.pdf.

Other Publications

[26] L. HELOUET, H. MARCHAND, T. JÉRON. *Testing Cover Channel*, 2008, Delivrable, Politess Project.

[27] H. MARCHAND, J. DUBREIL, T. JÉRON. *Automatic Test Generation for Security Property*, 2008, Delivrable, Politess Project.

References in notes

[28] R. ALUR, D. L. DILL. *A Theory of Timed Automata*, in "Theor. Comput. Sci.", vol. 126, n° 2, 1994, p. 183-235.

[29] L. BESNARD, H. MARCHAND, E. RUTTEN. *The Sigali Tool Box Environment*, July 2006, p. 465-466.

[30] 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.

[31] 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.

[32] 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, USA)", January 1977, p. 238-252.

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

[34] 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.

[35] ISO/IEC 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.

[36] T. LE GALL, B. JEANNET, H. MARCHAND. *Supervisory Control of Infinite Symbolic Systems using Abstract Interpretation*, in "44nd IEEE Conference on Decision and Control (CDC'05) and Control and European Control Conference ECC 2005, Seville (Spain)", December 2005, p. 31–35.

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

[38] 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^{o} 2, feb 1995, p. 107-125.

[39] C. PAULIN-MOHRING. *Le système Coq (Habilitation Thesis, in French)*, Technical report, ENS Lyon, 1997.

[40] 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^{o} 1, 1989, p. 81-98.

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

[42] J. TRETMANS. *Test Generation with Inputs, Outputs and Repetitive Quiescence.*, in "Software - Concepts and Tools", vol. 17, n^{o} 3, 1996, p. 103-120.

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