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 : Embedded and Real Time Systems

Table of contents

1. Team **1**
2. Overall Objectives **1**
2.1. Introduction 1
2.2. Highlights of the year 2
3. Scientific Foundations **2**
3.1. Underlying Models. 2
3.2. Verification 3
3.2.1. Abstract interpretation and Data Handling 4
3.2.2. Theorem Proving 4
3.2.3. Model-checking of infinite state and probabilistic systems 5
3.2.4. Analysis of infinite state systems defined by graph grammars 5
3.3. Automatic Test Generation 5
3.4. Controller Synthesis 7
4. Application Domains **8**
4.1. Panorama 8
4.2. Telecommunication Systems 8
4.3. Software Embedded Systems 8
4.4. Smart-card Applications 8
4.5. Control-command Systems 8
5. Software **9**
5.1. TGV 9
5.2. STG 9
5.3. SIGALI 9
6. New Results **10**
6.1. Verification and Abstract Interpretation 10
6.1.1. Analysis of probabilistic systems 10
6.1.1.1. Probabilistic Acceptors for Languages over Infinite Words 10
6.1.1.2. Probabilistic graph grammars 10
6.1.2. Analysis of Timed systems 10
6.1.2.1. Modal Specifications for Timed Systems 10
6.1.2.2. When are timed automata determinizable? 11
6.1.3. Characterization and Analysis of infinite systems 11
6.1.3.1. On external presentations of infinite graphs 11
6.1.3.2. Opacity and Abstraction 11
6.1.4. Equational Approximations for Tree Automata Completion 11
6.1.5. Verifying Invariants of Rewriting Specifications 12
6.2. Active and passive testing 12
6.2.1. Diagnosis of Pushdown Systems 12
6.2.2. Monitoring Confidentiality by Diagnosis Techniques 12
6.2.3. Testing security properties 12
6.3. Controller Synthesis and Game Theory 13
6.3.1. Stochastic games with partial information 13
6.3.2. Control of Infinite Symbolic Transitions Systems under Partial Observation 13
6.3.3. Discrete controller synthesis for modular reactive systems 13
6.3.4. Opacity Enforcing Control Synthesis 13
7. Other Grants and Activities **14**
7.1. National Grants & Contracts 14
7.1.1. RNTL TesTec: Test of Real-time and critical embedded System 14

7.1.2. RNRT POLITESS: Security Policies for Network Information Systems: Modeling, Deployment, Testing and Supervision 14
7.2. European and International Grants 15
7.2.1. Artist Design Network of Excellence 15
7.2.2. Combest. European Strep Project 15
7.2.3. PHC Procope PIPS: Partial Information Probabilistic Systems 15
7.2.4. DGRST-INRIA grant 15
7.2.5. Associated team (Equipe Associée) TReaTiES 16
7.3. Collaborations 16
7.3.1. Collaborations with other INRIA Project-teams 16
7.3.2. Collaborations with French Research Groups outside INRIA 16
7.3.3. International Collaborations 16
8. Dissemination **16**
8.1. University courses 16
8.2. PhD Thesis and Trainees 16
8.3. Scientific animation 17
9. Bibliography **17**

1. Team

Research Scientist

Thierry Jéron [Team Leader, 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]

Technical Staff

Florimond Ployette [Technical staff, IR (50% with ASCII) until September 2009]

PhD Student

Jérémy Dubreil [INRIA, since March 2006, until September 2009]
Sébastien Chédor [ENS CACHAN, since September 2009]

Post-Doctoral Fellow

Ylies Falcone [INRIA, since December 2009]

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

- Jérémy Dubreil defended his PhD thesis in November 2009 on *Monitoring and Supervisory Control for Opacity Properties*. It is the first thesis defended in the team on the subject of security analysis.
- The team obtained three international collaboration grants starting in 2009: a PHC Procope grant with Germany, a DGRST-INRIA grant with Tunisia, and an INRIA Associated Team with Brazil.

3. Scientific Foundations

3.1. Underlying Models.

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 [37]. 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.

Also, for verification purposes, we use graph grammars that are a general tool to define families of graphs. Such grammars are formed by a set of rules, left-hand sides being simply hyperedges and right-hand sides hypergraphs. For finite degree, these graph grammars characterise transition graphs of pushdown automata (each graph generated by such a grammar correspond to the transition graph of a pushdown automaton). They provide a simple yet powerfull setting to define and study infinite state systems.

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 four verification topics we are interested in: abstract interpretation, theorem proving, model-checking of infinite state and probabilistic systems and analysis of systems defined by graph grammars.

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 [39]. 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 [43] and COQ [44] 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.

3.2.4. Analysis of infinite state systems defined by graph grammars

Currently, many techniques (reachability, model checking, ...) from finite state systems have been generalised to pushdown systems, that can be modeled by graph grammars. Several such extensions heavily depend on the actual definition of the pushdown automata, for example, how many top stack symbols may be read, or whether the existence of ε-transitions (silent transitions) is allowed. Many of these restrictions do not affect the actual structure of the graph, and interesting properties like reachability or satisfiability (of a formula) only depend on the structure of a graph.

Deterministic graph grammars enable to focus on structural properties of systems. The connexion with finite graph algorithms is often straightforward: for example reachability is simply the finite graph algorithm iterated on the right hand sides. On the other hand, extending these grammars with time or probabilities is not straightforward: qualitative values associated to each copy (in the graph) of the same vertex (in the grammar) is different, introducing more complex equations. Furthermore, the fact that the left-hand sides are single hyperarcs is a very strong restriction. But removing this restriction leads to non-recursive graphs. Identifying decidable families of graphs defined by contextual graph grammars is also very challenging.

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 [47], 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 [46]. 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 [45]. 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 [45]. 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 [45] 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 [45] 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.

4. Application Domains

4.1. Panorama

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 [42], 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

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

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.

Finally, in collaboration with ULB, we implemented a prototype SMACS, derived from STG, that is devoted to the control of infinite system modeled by STS.

5.3. SIGALI

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], [38]. 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).

6. New Results

6.1. Verification and Abstract Interpretation

6.1.1. Analysis of probabilistic systems

6.1.1.1. Probabilistic Acceptors for Languages over Infinite Words

Participant: Nathalie Bertrand.

Probabilistic omega-automata are variants of nondeterministic automata for infinite words where all choices are resolved by probabilistic distributions. Acceptance of an infinite input word requires that the probability for the accepting runs is positive. In [14] and [34], we provide a summary of the fundamental properties of probabilistic omega-automata concerning expressiveness, efficiency, compositionality and decision problems.

6.1.1.2. Probabilistic graph grammars

Participants: Nathalie Bertrand, Christophe Morvan.

We currently study a probabilistic extension of regular graphs (i.e. graphs generated by deterministic graph grammars). These graphs form a structural extension of configuration graphs of pushdown systems whose probabilistic version has already been studied by Esparza *et al* [41]. We propose an algorithm to perform on probabilistic regular graphs the approximate verification of quantitative formulae expressed in the probabilistic logic PCTL. Moreover, we prove that the exact model-checking problem for PCTL on probabilistic regular graphs is undecidable, unless if we restrict to qualitative properties. Our results generalise [41] using similar methods combined with techniques of graph grammars.

6.1.2. Analysis of Timed systems

Participant: Nathalie Bertrand.

6.1.2.1. Modal Specifications for Timed Systems

On the one hand, modal specifications are classic, convenient, and expressive mathematical objects to represent interfaces of component-based systems. On the other hand, time is a crucial aspect of systems for practical applications, e.g. in the area of embedded systems. And yet, only few results exist on the design of timed component-based systems. In [17], we remedy this lack and define timed modal specifications, an automata-based formalism combining modal and timed aspects, as a stepping stone to compositional approaches of timed systems. We define the notions of refinement and consistency, and establish their decidability. This work, in collaboration with S. Pinchinat (S4 EPI) and J-B. Raclet (Pop-Art EPI) has been continued in [16] together with A. Legay (S4 EPI).

Based on the previous paper [17], we propose a timed extension of modal specifications, together with fundamental operations (conjunction, product, and quotient) that enable to reason in a compositional way about timed system. The specifications are given as modal event-clock automata, where clock resets are easy to handle. We develop an entire theory that promotes efficient incremental design techniques.

6.1.2.2. When are timed automata determinizable?

In [13], we propose an abstract procedure which, given a timed automaton, produces a language-equivalent deterministic infinite timed tree. We prove that under a certain boundedness condition, the infinite timed tree can be reduced into a classical deterministic timed automaton. The boundedness condition is satisfied by several subclasses of timed automata, some of them were known to be determinizable (event-clock timed automata, automata with integer resets), but some others were not. We prove for instance that strongly non-Zeno timed automata can be determinized. As a corollary of those constructions, we get for those classes the decidability of the universality and of the inclusion problems, and compute their complexities (the inclusion problem is for instance EXPSPACE-complete for strongly non-Zeno timed automata). This work was done in collaboration with C. Baier (Universität Dresden), P. Bouyer (LSV) and Th. Brihaye (Université de Mons).

6.1.3. Characterization and Analysis of infinite systems

6.1.3.1. On external presentations of infinite graphs

Participant: Christophe Morvan.

The vertices of a finite state system are usually a subset of the natural numbers. Most algorithms relative to these systems only use this fact to select vertices.

For infinite state systems, however, the situation is different: in particular, for such systems having a finite description, each state of the system is a configuration of some machine. Then most algorithmic approaches rely on the structure of these configurations. Such characterisations are said *internal*. In order to apply algorithms detecting a structural property (like identifying connected components) one may have first to transform the system in order to fit the description needed for the algorithm. The problem of internal characterisation is that it hides structural properties, and each solution becomes *ad hoc* relatively to the form of the configurations. On the contrary, *external* characterisations avoid explicit naming of the vertices. Such characterisation are mostly defined via graph transformations. In [24], we present two kind of external characterisations: deterministic graph rewriting, which in turn characterise regular graphs, deterministic context-free languages, and rational graphs. Inverse substitution from a generator (like the complete binary tree) provides characterisation for prefix-recognizable graphs, the Caucal Hierarchy and rational graphs. We illustrate how these characterisation provide an efficient tool for the representation of infinite state systems.

Finally, deterministic graph grammars generate a family of infinite graphs which characterize context-free (word) languages. In [33], we presents a context-sensitive extension of these grammars. We achieve a characterization of context-sensitive (word) languages. It is shown that this characterization is not straightforward and that unless having some rigorous restrictions, contextual graph grammars generate non-recursive graphs.

6.1.3.2. Opacity and Abstraction

Participant: Jérémy Dubreil.

The opacity property characterizes the absence of confidential information flow towards inquisitive attackers. Verifying opacity is well established for finite automata but is known to be not decidable for more expressive models like Turing machines or Petri nets. As a consequence, for a system dealing with confidential information, certifying its confidentiality may be impossible, but attackers can infer confidential information by approximating systems' behaviours. Taking such attackers into account, we investigate the verification of opacity using abstraction techniques to compute executable counterexamples (attack scenarios). Considering a system and a predicate over its executions, attackers are modeled as semi-conservative decision process determining from observed traces the truth of that predicate. Moreover, we show that the most precise the abstraction is, the most accurate (and then dangerous) the corresponding class of attackers will be. Consequently, when no attack scenario is detected on an approximate analysis, we know that this system is safe against all "less precise" attackers. This can therefore be used to provide a level of certification relative to the precision of abstractions.

6.1.4. Equational Approximations for Tree Automata Completion

Participant: Vlad Rusu.

In [11], we deal with the verification of safety properties of infinite-state systems modeled by term-rewriting systems. An over-approximation of the set of reachable terms of a term-rewriting system R is obtained by automatically constructing a finite tree automaton. The construction is parameterized by a set E of equations on terms, and we also show that the approximating automata recognize at most the set of R/E-reachable terms. Finally, we perform some experiments carried out with the implementation of our algorithm. In particular, we show how some approximations from the literature can be defined using equational approximations. This work was done in collaboration with Th. Genest (Celtique EPI).

6.1.5. Verifying Invariants of Rewriting Specifications

Participant: Vlad Rusu.

In [29],[26], we present an approach based on inductive theorem proving for verifying invariants of dynamic systems specified in rewriting logic, a formal specification language implemented in the Maude system. An invariant is a property that holds on all the states that are reachable from a given class of initial states. Our approach consists in encoding the semantic aspects that are relevant for our task (namely, verifying invariance properties of the specified systems) in membership equational logic, a sublogic of rewriting logic. The invariance properties are then formalized over the encoded rewrite theories and are proved using an inductive theorem prover for membership equational logic also implemented in the Maude system using its reflective capabilities. We illustrate our approach by verifying mutual exclusion in an n-process Bakery algorithm. This work was done in collaboration with M. Clavel (University of Madrid).

6.2. Active and passive testing

6.2.1. Diagnosis of Pushdown Systems

Participant: Christophe Morvan.

Diagnosis problems of discrete-event systems consist in detecting unobservable defects during system execution. For finite-state systems, the theory is well understood and a number of effective solutions have been developed. For infinite-state systems, however, there are only few results, mostly identifying classes where the problem is undecidable. In [25], [36], we consider higher-order pushdown systems and investigate two basic variants of diagno- sis problems: the diagnosability, which consists in deciding whether defects can be detected within a finite delay, and the bounded-latency problem, which consists in determining a bound for the delay of detecting defects. This work was done in collaboration with S. Pinchinat (S4 EPI).

6.2.2. Monitoring Confidentiality by Diagnosis Techniques

Participants: Jérémy Dubreil, Thierry Jéron, Hervé Marchand.

In [20], we have been interested in constructing monitors for the detection of confidential information flow in the context of partially observable discrete event systems. We first characterize the set of observations allowing an attacker to infer the secret information. 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 construct a monitor allowing an administrator to detect it.

6.2.3. Testing security properties

Participants: Jérémy Dubreil, Thierry Jéron, Hervé Marchand.

In [23][28], we investigate the combination of controller synthesis and test generation techniques for the testing of open, partially observable systems with respect to security policies. We consider two kinds of properties: integrity properties and confidentiality properties. We assume that the behavior of the system is modeled by a labeled transition system and assume the existence of a black-box implementation. We first outline a method allowing to automatically compute an ideal access control ensuring these two kinds of properties. Then, we show how to derive testers that test the conformance of the implementation with respect to its specification, the correctness of the real access control that has been composed with the implementation in order to ensure a security property, and the security property itself.

6.3. Controller Synthesis and Game Theory

6.3.1. Stochastic games with partial information

Participant: Nathalie Bertrand.

In [15], we consider the standard model of finite two-person zero-sum stochastic games with signals. We are interested in the existence of almost-surely winning or positively winning strategies, under reachability, safety, Büchi or co-Büchi winning objectives. We prove two qualitative determinacy results. First, in a reachability game either player 1 can achieve almost-surely the reachability objective, or player 2 can ensure surely the complementary safety objective, or both players have positively winning strategies. Second, in a Büchi game if player 1 cannot achieve almost-surely the Büchi objective, then player 2 can ensure positively the complementary co-Büchi objective. We prove that players only need strategies with finite-memory, whose sizes range from no memory at all to doubly-exponential number of states, with matching lower bounds. Together with the qualitative determinacy results, we also provide fix-point algorithms for deciding which player has an almost-surely winning or a positively winning strategy and for computing the finite memory strategy. Complexity ranges from EXPTIME to 2-EXPTIME with matching lower bounds, and better complexity can be achieved for some special cases where one of the players is better informed than her opponent. This work was done in collaboration with B. Genest (Distribcom EPI) and H. Gimbert (Labri).

6.3.2. Control of Infinite Symbolic Transitions Systems under Partial Observation

Participant: Hervé Marchand.

We provide models of safe controllers both for potentially blocking and non blocking controlled systems. To obtain algorithms for these problems, we make the use of abstract interpretation techniques which provide over-approximations of the transitions set to be disabled. 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 [22]. This work has been extended to the case of decentralized control in [27]. Finally, on the same model, but assuming that the system is finite, we have studied the computational complexity of several decision and optimization control problems arising in partially observed discrete event systems [21]. This work has been done in cooperation with T. Massart, G. Kalyon and T. Le Gall (Université libre de Bruxelles).

6.3.3. Discrete controller synthesis for modular reactive systems

Participant: Hervé Marchand.

Following preliminaries results [12], we have been interested in the extension of a reactive programming language with a behavioral contract construct [31]. It is particularly dedicated to the programming of reactive control of applications in embedded systems, and involves principles of the supervisory control of discrete event systems. Our contribution is in a language approach where modular discrete controller synthesis (DCS) is integrated, and it is concretized in the encapsulation of DCS into a compilation process. From transition system specifications of possible behaviors, DCS automatically produces controllers that make the controlled system satisfy the property given as objective. Our language features and compiling technique hence provide correctness-by-construction in that sense, and enhance reliability and verifiability. An application domain particularly targeted at is that of adaptive and reconfigurable systems: closed-loop adaptation mechanisms enable flexible execution of functionalities w.r.t. changing resource and environment conditions. This language can serve programming such adaption controllers. This work has been done in cooperation with E. Rutten and G. Delaval (INRIA Grenoble).

6.3.4. Opacity Enforcing Control Synthesis

Participants: Jérémy Dubreil, Hervé Marchand.

In the field of computer security, a problem that received little attention so far is the enforcement of confidentiality properties by supervisory control. Given a critical system G that may leak confidential information (a secret), the problem consists in designing a controller C, possibly disabling occurrences of a fixed subset of events of G, so that the closed-loop system G/C does not leak confidential information. We consider this problem in the case where G is a finite transition system with set of events Σ and an inquisitive user, called the adversary, observes a subset Σ_a of Σ. When the secret can be disclosed. We present an effective algorithm for computing the most permissive controller C such that S is opaque w.r.t. G/C and Σ_a. This algorithm subsumes two earlier algorithms presented in [40] working under the strong assumption that the alphabet Σ_a of the adversary and the set of events that the controller can disable are comparable. This work publised as a research report ([32]) has been accepted for publication in 2010 in IEEE Transaction Automatic and Control [9]. This work has been done in cooperation with Ph. Darondeau (S4 EPI).

In [18], we followed a different approach. We introduced the notion dynamic partial observability where the set of events the user can observe changes over time. We have shown how to check that a system is opaque w.r.t. to a dynamic observer and also addressed the corresponding synthesis problem: given a system G and secret states S, compute the set of dynamic observers under which S is opaque. It turned out that this problem can be reduced to a two-players safety game and that the set of such observers can be finitely represented and can be computed in EXPTIME. This work has been done in cooperation with F. Cassez (IRCCyN).

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.

The TESTEC project is a three years [2008-2010] 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, 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. The final review of POLITESS took place in February 2009.

7.2. European and International Grants

7.2.1. 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.

7.2.2. 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.

7.2.3. PHC Procope PIPS: Partial Information Probabilistic Systems

Participant: Nathalie Bertrand.

The objective of this bilateral collaboration [2009-2010] with the group of Prof. Christel Baier in TU Dresden (Germany) is to study partially observable probabilistic systems. M. Groesser visited us during one week and N. Bertrand visited Dresden for three weeks.

7.2.4. DGRST-INRIA grant

Participants: Nathalie Bertrand, Thierry Jéron, Hervé Marchand.

This two years collaboration [2009-2010] with ENIS Sfax Tunisia (Maher Ben Jemaa and Moez Krichen) is targetted on testing embedded systems and adaptability (with the Paris project team). It is funded by an DGRST - INRIA grant which involves visits on both sides and scholarships for Tunisian students. M. Krichen visited Vertecs during one week and T. Jéron and H. Marchand visited ENIS Sfax during one week.

7.2.5. Associated team (Equipe Associée) TReaTiES

Participants: Nathalie Bertrand, Thierry Jéron, Hervé Marchand.

This associated team with the Federal University of Campina Grande (Prof. Patrícia D. L. Machado) and University Pernambuco (Prof. Augusto Sampaio) in Brazil started in 2009. The objective is to work on test case generation, selection and abstraction for embedded real-time systems. In 2009 we had the visit of Sidney Nogueira and N. Bertrand, T. Jéron and H. Marchand visited the Brazilian team.

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 and on stochastic games with partial observation. 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, 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

We collaborate with LIG (Vasco teams and Verimag) in Grenoble in the context of the RNRT Politess grant. 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 in the context of the RNTL TesTec grant.

7.3.3. International Collaborations

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

Université Libre Bruxelles in Belgium (Prof. Thierry Massart) on testing and control of symbolic transitions systems. Gabriel Kalyon visited us for 1 week in september.

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

ETH Zurich (Marina Egea) on formal semantics conformance in model-driven engineering

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

8. Dissemination

8.1. University courses

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

Nathalie Bertrand taught modelisation in computer science to students of ENS Ker Lann in March 2009 and gave a lecture on Timed Automata in Master 2 Research at the University of Rennes 1 in October 2009.

Christophe Morvan is teaching at the University of Marne La Vallée (192h/year).

8.2. PhD Thesis and Trainees

PhD. thesis defended in 2009:

Jérémy Dubreil : "*Monitoring and Supervisory Control for Opacity Properties*", November 2009.

Current PhD. thesis:

Sébastien Chédor: "*Verification and Test of systems modeled by regular graphs*", first year.

Trainees 2008-2009:

Sébastien Chédor: "*Infinite discrete event systems and partial information*"

8.3. Scientific animation

Nathalie Bertrand was PC member of QAPL'09 workshop, and QEST'09 international conference. She was invited to give seminars on "When are timed automata determinizable?" at LaBRI Bordeaux and LIF Marseille (may and october 2009 respectively). She visited TU Dresden twice 2 weeks in february and november 2009.

Thierry Jéron was PC member of Testcom/Fates'09 and ICFEM'09 and SC member of Movep 2010. He gave an invited talk at MSR'09, and a tutorial at the ETR'09 summer school (*Ecole d'été Temps réel*) on "Automatic test generation of Reactive and timed systems". He was reviewer of the PhD defense of Eduardo Almeida (Université de Nantes, February 2009), of Yliès Falcone (Université Joseph Fourier, Grenoble, November 2009) and of the HDR defense of Hélène Collavizza (Université de Nice, December 2009). He is member of the IFIP Working Group 10.2 on Embedded Systems.

Hervé Marchand is Associate Editor of the IEEE Transactions on Automatic Control journal since October 2009. He is member of the IFAC Technical Committees (TC 1.3 on Discrete Event and Hybrid Systems) since 2005. He was PC member of the ICINCO'09 and MSR'09 Conferences. He was invited to give a seminar at INRIA Grenoble on "optimal control of discrete event systems" in december 2009. He visited ULB (Bruxelles) for one week in June 2008 during which he gave a seminar on "security analysis of information systems".

Christophe Morvan was invited to give a seminar "Regular graphs : a perfect model for infinite state systems?" at LSV, Cachan (Februar 2009). He gave a talk: "Diagnosability of pushdown systems" at the conference AutomathA 2009 (for the AutomathA European project).

Vlad Rusu organized the EJCP (Ecole Jeunes Chercheurs en Programmation) in June 2009.

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° 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° 6, November 2006, p. 710–730.

Year Publications

Doctoral Dissertations and Habilitation Theses

[8] J. DUBREIL. *Monitoring and Supervisory Control for Opacity Properties*, Université de Rennes 1, November 2009, Ph. D. Thesis.

Articles in International Peer-Reviewed Journal

[9] J. DUBREIL, P. DARONDEAU, H. MARCHAND. *Supervisory Control for Opacity*, in "IEEE Transactions on Automatic Control", 2010, to appear.

[10] M. EGEA, V. RUSU. *Formal executable semantics for conformance in the MDE framework*, in "Innovations in Systems and Software Engineering", 2009.

[11] T. GENEST, V. RUSU. *Equational Approximations for Tree Automata Completion*, in "Journal of Symbolic Computation", 2010, to appear.

[12] E. RUTTEN, H. MARCHAND. *Automatic generation of safe handlers for multi-task systems*, in "Journal of Embedded Computing", vol. 3, n° 4, 2009, to appear.

International Peer-Reviewed Conference/Proceedings

[13] C. BAIER, N. BERTRAND, P. BOUYER, T. BRIHAYE. *When are timed automata determinizable?*, in "36th International Colloquium on Automata, Languages and Programming (ICALP'09), Rhodes, Greece", LNCS, vol. 5556, July 2009, p. 43-54 DE BE .

[14] C. BAIER, N. BERTRAND, M. GRÖSSER. *Probabilistic Acceptors for Languages over Infinite Words*, in "35th Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'09), Spindleruv Mlyn, Czech", LNCS, vol. 5404, Springer, 2009, p. 19-33 DE .

[15] N. BERTRAND, B. GENEST, H. GIMBERT. *Qualitative Determinacy and Decidability of Stochastic Games with Signals*, in "24th Annual IEEE Symposium on Logic in Computer Science (LICS'09), Los Angeles, CA, USA", IEEE Computer Society Press, August 2009, p. 319-328.

[16] N. BERTRAND, A. LEGAY, S. PINCHINAT, J.-B. RACLET. *A Compositional Approach on Modal Specifications for Timed Systems*, in "Proceedings of the 11th International Conference on Formal Engineering Methods (ICFEM'09)", Lecture Notes in Computer Science, vol. 5885, Springer, 2009, p. 679-697.

[17] N. BERTRAND, S. PINCHINAT, J.-B. RACLET. *Refinement and Consistency of Timed Modal Specifications*, in "Proceedings of the 3rd International Conference on Language and Automata Theory and Applications (LATA'09), Tarragona, Spain", LNCS, vol. 5457, April 2009, p. 152-163.

[18] F. CASSEZ, J. DUBREIL, H. MARCHAND. *Dynamic Observers for the Synthesis of Opaque Systems*, in "7th International Symposium on Automated Technology for Verification and Analysis (ATVA'09), Macao SAR, China", Z. LIU, A. RAVN (editors), LNCS, vol. 5799, Springer-Verlag, October 2009, p. 352-367.

[19] J. DUBREIL. *Opacity and Abstraction*, in "Proceedings of the First International Workshop on Abstractions for Petri Nets and Other Models of Concurrency (APNOC'09), Paris, France", June 2009.

[20] J. DUBREIL, T. JÉRON, H. MARCHAND. *Monitoring Confidentiality by Diagnosis Techniques*, in "European Control Conference, Budapest, Hungary", August 2009, p. 2584-2590.

[21] G. KALYON, T. LE GALL, H. MARCHAND, T. MASSART. *Computational Complexity for State-Feedback Controllers with Partial Observation*, in "7th International Conference on Control and Automation, ICCA'09, Christchurch, New Zealand", December 2009 BE .

[22] G. KALYON, T. LE GALL, H. MARCHAND, T. MASSART. *Control of Infinite Symbolic Transition Systems under Partial Observation*, in "European Control Conference, Budapest, Hungary", August 2009, p. 1456-1462 BE .

[23] H. MARCHAND, J. DUBREIL, T. JÉRON. *Automatic Testing of Access Control for Security Properties*, in "TestCom'09/FATES'09", LNCS, vol. 5826, November 2009, p. 113-128.

[24] C. MORVAN. *On external presentations of infinite graphs*, in "11th International Workshop on Verification of Infinite-State Systems, INFINITY'09, Bologna, Italy", n^{O} 10, August 2009, p. 23-35.

[25] C. MORVAN, S. PINCHINAT. *Diagnosability of pushdown systems*, in "HVC2009, Haifa Verification Conference, Haifa, Israel", October 2009, to appear in LNCS.

[26] V. RUSU. *Formal Executable Semantics for Conformance in the MDE Framework*, in "UML and FM workshop", 2009.

National Peer-Reviewed Conference/Proceedings

[27] G. KALYON, T. LE GALL, H. MARCHAND, T. MASSART. *Contrôle décentralisé de systèmes symboliques infinis sous observation partielle*, in "7ème Colloque Francophone sur la Modélisation des Systèmes Réactifs", November 2009, p. 805-820 BE .

[28] H. MARCHAND, J. DUBREIL, T. JÉRON. *Génération automatique de tests pour des propriétés de sécurité*, in "4ème Conférence sur la Sécurité des Architectures Réseaux et des Systèmes d'Information", June 2009, p. 157-174.

[29] V. RUSU, M. CLAVEL. *Vérification d'invariants pour des systèmes spécifiés en logique de réécriture*, in "Vingtièmes Journées Francophones des Langages Applicatifs, JFLA 2009, Saint Quentin sur Isère, France", A. SCHMITT (editor), Studia Informatica Universalis, vol. 7.2, February 2009, p. 317-350 ES .

Research Reports

[30] F. CASSEZ, J. DUBREIL, H. MARCHAND. *Dynamic Observers for the Synthesis of Opaque Systems*, n^{o} 1930, IRISA, May 2009, Technical report.

[31] G. DELAVAL, H. MARCHAND, E. RUTTEN. *BZR Contracts for Modular Discrete Controller Synthesis*, n^{o} 7111, INRIA, November 2009, Research Report.

[32] J. DUBREIL, P. DARONDEAU, H. MARCHAND. *Supervisory Control for Opacity*, n^{o} 1921, IRISA, February 2009, Technical report.

[33] C. MORVAN. *Contextual graph grammars characterizing context-sensitive languages*, n^{o} 1926, IRISA, March 2009, Technical report.

Other Publications

[34] C. BAIER, N. BERTRAND, M. GRÖSSER. *The Effect of Tossing Coins in Omega-Automata*, in "Proceedings of the 20th International Conference on Concurrency Theory (CONCUR'09)", Lecture Notes in Computer Science, vol. 5710, Springer, 2009, Invited talk (C. Bayer).

[35] T. JÉRON. *Génération de tests pour les systèmes réactifs et temporisés*, in "Ecole d'Eté Temps-Réel, Télécom ParisTech, Paris", September 2009, Invited talk.

[36] C. MORVAN, S. PINCHINAT. *Diagnosability of pushdown systems*, in "AutomathA, Liège, Belgique", June 2009.

References in notes

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

[38] L. BESNARD, H. MARCHAND, E. RUTTEN. *The Sigali Tool Box Environment*, in "Workshop on Discrete Event Systems, WODES'06 (Tool Paper), Ann-Arbor (MI, USA)", July 2006, p. 465-466.

[39] 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.

[40] J. DUBREIL, P. DARONDEAU, H. MARCHAND. *Opacity Enforcing Control Synthesis*, in "Workshop on Discrete Event Systems, WODES'08, Gothenburg, Sweden", March 2008, p. 28–35.

[41] J. ESPARZA, A. KUCERA, R. MAYR. *Model Checking Probabilistic Pushdown Automata*, in "Logical Methods in Computer Science", vol. 2, n^{o} 1, 2006.

[42] 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.

[43] 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.

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

[45] 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.

[46] 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.

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