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

THEME 1C

Table of contents

1. Team

Head of project-team
Thierry Jéron [CR INRIA]

administrative assistant
Lydie Mabil-Letort [TR INRIA]

Research Fellows (Inria)
Bertrand Jeannet [CR]
Hervé Marchand [CR]
Vlad Rusu [CR]

Visiting scientist
Ahmed Khoumsi [Professor Univ. Sherbrooke (Canada), October 2002-September 2003]

Post-doctoral fellow (Inria)
Wendelin Serwe [September 2002- January 2003, shared with LANDE]

Project technical staff
Erwan Demairy [INRIA, September 2002-August-2003]

Junior technical staff
François-Xavier Ponscarme [INRIA, since October 2002]

Ph. D. Students
Benoît Gaudin [MENRT]
Valery Tschaen [MENRT]
Elena Zinovieva [INRIA]

2. Overall Objectives

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

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

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

Our research is based on formal models and **verification** techniques such as model checking, theorem proving, abstract interpretation, the control theory of discrete event systems, and their underlying models and logics. The close connection between testing, control and verification produces a synergy between these research topics and allows us to share theories, models, algorithms and tools.

3. Scientific Foundations

3.1. Underlying models.

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

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 $M = (Q, A, \rightarrow, q_o)$ where Q is a non-empty set of states; $q_o \in Q$ is the initial state; A is the alphabet of actions, $\rightarrow \subseteq Q \times A \times Q$ is the transition relation.

To model reactive systems in the testing context, we use Input/Output labeled transition systems (IOLTS for short). In this setting, interactions between the system and its environment are modeled by input (controlled by the environment) and output events (observed by the environment), and the internal behavior of the system is modeled by internal (non observable) events. In the controller synthesis theory, we also distinguish between controllable and uncontrollable events, observable and unobservable events. In testing, we also manipulate input-output symbolic transition systems (IOSTS), which are extensions of IOLTS that operate on data (i.e., program variables, communication parameters, symbolic constants) through message passing, guards, and assignments. An alternative to IOSTS to specify systems with data variables is the model of synchronous dataflow equations.

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

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

3.2. Automatic Test Generation

Conformance testing consists in checking whether an implementation under test (abbreviated as *IUT*) behaves correctly with respect to its specification. In the line of model-based testing, we use formal specifications and their underlying models to unambiguously define conformance testing and test case generation. One difficult problem is to generate adequate test cases (the selection problem) that correctly identify faults (the oracle problem). We use *test purposes* for selection, which allow to generate tests targeted to specific behaviours. For solving the oracle problem we adapt a well established theory of conformanced testing [40], which formally defines conformance as a relation between formal models of specifications and implementations. This conformance relation, called **ioco** compares suspension traces of the specification and implementation. Suspension traces are sequence of input, output or quiescence (absence of action), thus abstract away internal behaviors that cannot be observed by testers. Roughly speaking, an implementation is conformant if after a suspension trace of the specification, the implementation can only show outputs and quiescences of specfication.

We use IOLTS (or IOSTS) as formal models for specifications, implementations, test purposes, and test cases. Most often, specifications are not directly given in such low-level models, but are written in higher-level specification languages (e.g. SDL, UML, Lotos). The tools associated with these languages often contain a simulation API that implements their semantics under the form of IOLTS. On the other hand, the IOSTS model is expressive enough to allow a direct representation of most constructs of the higher-level languages. Test purposes are specified directly as IOLTS (or IOSTS). They are associated with marked states, giving them the status of automata or observers. When IOLTS are considered, test purposes are accepting sequences of actions whose projection on visible actions are selected for test generation. When IOSTS are considered, test purposes are extended automata that observe behaviors, i.e. sequences of actions and vectors of variables of the specifications, whose projection is selected for test generation.

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

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

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

Our first test generation algorithms are based on enumerative techniques, optimized to fight the state-space explosion problem. We have developed on-the-fly algorithms, which consist in performing a lazy exploration of the set of states that are reachable in both the specification and the test purpose. The resulting test case is an IOLTS whose traces describe interactions with an implementation under test. This technique is now mature and implemented in the TGV tool, which we often use in industrial collaborations. We are continuously improving the technique. However, what characterizes this enumerative technique is that values of variables and communication parameters are instantiated at test generation time.

More recently, we have explored symbolic test generation techniques. This is a promising technique whose main objective is to avoid the state space explosion problem induced by the enumeration of values of variables and communication parameters. The idea consists in computing a test case under the form of an *IOSTS*, i.e., a reactive program in which the operations on data is kept in a symbolic form. However, most of the operations involved in test generation (determinization, reachability, and coreachability) become undecidable. For determinization we employ heuristics that allow us to solve the so-called bounded observable non-determinism (i.e., the result of an internal choice can be detected after finitely many observable actions). For the remaining operations, reachable and co-reachable sets of states are approximated using abstract interpretation techniques. These techniques are implemented in the STG tool.

3.3. Controller Synthesis

3.3.1. The Supervisory Control Problem

is concerned with ensuring (not only checking) that a computer-operated system works correctly. More precisely, given a specification model and a required property, the problem is to control the specification's

behavior, by coupling it to a supervisor, such that the controlled specification satisfies the property [36]. The models used are LTS (and the associated language), which make a distinction between *controllable* and *non-controllable* actions and between *observable* and *non-observable* actions. Typically, the controlled system is constrained by the supervisor, which acts on the system's controllable actions and forces it to behave as specified by the property. The control synthesis problem can be seen as a constructive verification: building a supervisor that prevents the system from violating a property. Several kinds of properties can be ensured such as reachability, invariance, attractivity, etc. Techniques adapted from model checking are then used to compute the supervisor w.r.t. the objectives. Optimality must be taken into account as one often wants to obtain a supervisor that constrains the system as few as possible.

3.3.2. Optimal Control.

We are also interested in the Optimal Control Problem. The purpose of optimal control is to study the behavioral properties of a system in order to generate a supervisor that constrains the system to a desired behavior according to quantitative and qualitative requirements. In this spirit, we have been working on the optimal scheduling of a system through a set of multiple goals that the system had to visit one by one [6]. We have also extended the results of [39] to the case of partial observation in order to handle more realistic applications [33].

3.3.3. Control of Hierarchical Discrete Event System.

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

In order to reduce the complexity of the supervisor synthesis phase, several approaches have been considered in the literature. Modular control [42] and modular plant [44] are natural ways to handle this problem. Similarly, in order to take into account nested behaviors, some techniques based on model aggregation methods [41][27] have been proposed to deal with hierarchical control problems. Another direction has been proposed in [26]. Brave and Heimann in [26] introduced Hierarchical State Machines which constitute a simplified version of the STATECHARTS. Compared to the classical state machines, they add orthogonality and hierarchy features. Some other works dealing with control and hierarchy can be found in [30][32]. This is the direction we have chosen in the VERTECS Team [8].

3.4. Verification

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

3.4.1. Abstract interpretation and numerical data handling

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

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

The big change induces by taking into account the data and not only the (finite) control of the systems under study is that the fixpoints become not computable. The undecidability is overcomed by resorting to approximations, using the theoritical framework of Abstract Interpretation [29].

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

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

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

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

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

Apprximations are conservative so that the obtained result is an upper-approximation of the exact result. Those two principles are well illustrated by the Interval Analysis [28], which consists in associating to each numerical variable of a program an interval representing an (upper) set of reachable values:

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

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

Programs performing dynamic allocation of objects in memory have an even more complex state space. Solutions have been devised to represent in an approximate way the memory heap and pointers between memory cells by graphs (*shape analysis* [38][37]). Values contained in memory cells are however generally ignored.

In the same way, programs with recursive procedure calls, parameter passing and local variables are more delicate to analyse with precision. The difficulty is to abstract the execution stacks which may have a complex structure, particularly whenparameter passing by reference is allowed, as it induces aliasing on the stack [25].

3.4.2. Theorem Proving

For verification we also use theorem proving and more particularly the PVS [34] and COQ [35] proof assistants. These are two general-purpose systems based on two different versions of higher-order logic. A verification task in such a proof assistant consists in encoding the system under verification and its properties into the logic of the proof assistant, together with verification *rules* that allow to prove the properties. Using the rules usually requires input from the user; for example, proving that a state predicate holds in every reachable state of the system (i.e., it is an *invariant*) typically requires to provide a stronger, *inductive* invariant, which is preserved by every execution step of the system. Another type of verification problem is proving *simulation* between a concrete and an abstract semantics of a system. This too can be done by induction in a systematic manner, by showing that, in each reachable state of the system, each step of the concrete system is simulated by a corresponding step at the abstract level.

4. Application Domains

4.1. Panorama

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

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

4.2. Telecommunication systems

Our research on test generation was initially proposed for conformance testing of telecommunication protocols. In this domain, testing is a normalized process [24], and formal specification languages are widely used (SDL in particular). Our test generation techniques have already proved useful in this context, going up to industrial transfer. New standardized component-based design methodologies such as UML and OMG's MDE will certainly increase the need for formal techniques in order to ensure the composability of components, by verification and testing. We believe that our techniques, by their genericity and adaptativity, will also prove useful at different levels of these methodologies, from component testing to system testing.

4.3. Software Embedded Systems

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

4.4. Smart-card applications

We have also applied our test generation techniques in the context of smart-card applications. Such applications are typically reactive as they describe interactions between a user, a terminal and a card. The number and

complexity of such applications is increasing, with more and more services offered to users. The security of such applications is of primary interest for both users and providers and testing is one the means to improve it.

4.5. Control-command Systems

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

5. Software

5.1. Tgv

Participants: Thierry Jéron [contact], Valéry Tschaen, Erwan Demairy, Simon Pickin.

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

TGV (Test Generation with Verification technology) is a tool for test generation of conformance test suites from specifications of reactive systems [3]. It is based on the IOLTS model, a well defined theory of testing, and on-the-fly test generation algorithms coming from verification technology. Originally, TGV allows test generation focussed on well defined behaviors formalized by test purposes. The main operations of TGV are (1) a synchronous product which identifies sequences of the specification accepted by a test purpose , (2) abstraction and determinization for the computation of next visible actions, (3) selection of test cases by the computation of reachable states from the initial states and co-reachable states from accepting states. TGV has been developed in collaboration with Vérimag Grenoble and uses libraries of the CADP toolbox (VERIMAG and VASY). TGV can be seen as a library that can be linked to different simulation tools through well defined APIs. An academic version of TGV is distributed in the CADP toolbox and allows test generation from Lotos specifications by a connection to its simulator API. The same API is used for a connection with the UMLAUT validation framework of UML models. This version has been transfered in the SDL ObjectGéode toolset as part of the TestComposer tool. A new version of TGV is currently adapted to a new API of the IF simulator (VERIMAG) allowing test generation from IF and UML models (via a compilation from UML to IF). This new version extends the previous one with new functionalities for coverage based test generation combined with test purposes based test generation. TGV is protected by APP (Agence de Protection des Programmes) Number IDDN.FR.001.310012.00.R.P.1997.000.2090.

5.2. Nbac

Participant: Bertrand Jeannet [contact].

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

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

abstract interpretation: sets of states are approximated by abstract values belonging to an abstract domain, on which fix-point computations are performed. The originality of NBAC resides in

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

5.3. Stg

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

Key words: *test de conformité, test et vérification symboliques.*

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

5.4. Sigali

Participant: Hervé Marchand [contact].

Key words: *Controller Synthesis, verification, symbolic techniques.*

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

5.5. Syntool

Participants: Benoit Gaudin, Hervé Marchand [contact].

Key words: *Controller Synthesis, structured systems.*

SYNTOOL is a tool dedicated to the control of structured discrete systems event. It implements the theory developed in [16][22]. SYNTOOL has an API allowing the user to graphically describe the different LTS modeling the plant, to perform some controller synthesis computations solving e.g. the forbidden state avoidance problem for structured systems and finally to simulate the result (i.e. the behavior of the controlled system). This tool is currently under testing.

5.6. rapture

Participant: Bertrand Jeannet [contact].

Key words: *Markov Decision Processes*, *Probabilistic verification*.

RAPTURE is a verification tool developed jointly by BRICS and INRIA [31]. The tool is designed to verify reachability properties on Markov Decision Processes (MDP), also known as Probabilistic Transition Systems. This model can be viewed both as an extension to classical (finite-state) transition systems extended with probability distributions on successor states, or as an extension of Markov Chains with non-determinism. We have developed a simple automata language that allows to describe a set of processes communicating over a set of channels *à la* CSP. Processes can also manipulate local and global variables of finite type. Probabilistic reachability properties are specified by defining two sets of initial and final states together with a probability bound. The originality of the tool is to provide two reduction techniques that limit the state space explosion problem: automatic abstraction and refinement algorithms, and the so-called essential states reduction [43].

6. New Results

6.1. Controller Synthesis

Key words: *controller synthesis methodology*, *Hierarchical models*, *symbolic methods*.

6.1.1. Supervisory Control of Structured Discrete Event Systems

Participants: Benoit Gaudin, Hervé Marchand.

Following our preliminary results on control of hierarchical systems [8], we first enriched the model by considering interactions between the components of the system in order to handle more realistic applications. In particular, given a system modeled as a structured discrete event systems, we have proposed a method solving the State avoidance Control Problem for a set of configurations of the system. Based on a particular decomposition of the set of forbidden configurations, we locally solve the control problem (i.e. on each component without computing the whole system) and we produce a global supervisor ensuring the desired property. This supervisor can be seen as an oracle that will activate/deactivate local supervisors according to the current configuration of the global system and some conditions that can be easily computed on the fly. Moreover, we make the necessary efforts to keep the structure of the plant in the global supervisor, hence improving the readability and the understanding of the supervisor effect as well as its memory storage [15][16][22]. We also developed a methodology allowing to control structured discrete event systems, when the control objectives are expressed in terms of languages. This work is still under progress.

These techniques has been implemented in a prototype, named SYNTOOL. It is currently under testing.

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

Participant: Hervé Marchand.

In collaboration with the BIP Project (now POP-ART) and VERIMAG, we have been interested in the programming of real-time control systems, such as in robotics or avionics. These systems are designed with multiple tasks, each with multiple modes. We propose a model of tasks in terms of transition systems, designed especially with the purpose of applying existing discrete controller synthesis techniques (based on the SIGALI framework). This provides us with a systematic methodology, for the automatic generation of safe task handlers, with the support of synchronous languages and associated tools for compilation and formal computation. This work is still under progress.

6.2. Test generation on enumerative and symbolic models

Key words: *testing*, *test generation*, *models*, *transition systems*, *symbolic transition systems*.

6.2.1. Test generation based on coverage directives

Participants: Thierry Jéron, Valery Tschaen, Erwan Demairy.

Our test generation techniques were previously based on a selection by test purposes. However, this approach necessitates to specify those test purposes. Users sometimes want more automatic ways to generate test cases, from more general selection mechanisms. In the context of the Agedis european project (see 7.1), we have defined more general selection mechanisms, called test selection directives. They allow to describe both coverage directives (on states, transitions and more generally expressions on variables), test purposes (extended to more general observers) and constraints on data values, and to combine them. Taking into account these test directives involved a deep modification in some test generation algorithms. We also designed algorithms that generate test cases randomly, without any test directive. All these algorithms are incremental in the sense that they produce test cases when these are computed, without waiting the end of the process, thus allowing users to interupt the process with a partial result. These results are still unpublished.

6.2.2. From Safety Verification to Safety Testing

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

In this work, we define a methodology that combines verification and conformance testing for validating safety requirements of reactive systems. The safety requirements, specified as observers of visible behaviors, are first automatically verified on the system's specification. Then, test cases are automatically derived from the specification and the requirements, and executed on a black-box implementation of the system. This allows to check whether the requirements hold on the implementation as well. It is shown that an implementation conforms to its specification (for the conformance relation of Tretmans) if and only if it satisfies all the relevant safety requirements that are satisfied by the specification. The main differences with our previous works on test generation from test purposes is that test purposes express reachability properties, while requirements express safety properties, which are the most used type of property used in verification, and our methodology establishes a direct link between verification and test generation as what is tested on the implementation is exactly what is verified on the specification. This work will be presented in Testcom in 2004.

6.2.3. Symbolic test generation

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

The PhD. thesis of Elena Zinovieva describes the integration of the approximated reachability and co-reachability algorithms embodied into the NBac tool, into the general symbolic test generation algorithm of the STG tool. The new reachability/co-reachability algorithms allow for a precise handling of data, and as a result, the new version of the tool is able to generate better test cases (i.e., having less Inconclusive verdicts).

6.3. Interaction between test, control and verification

6.3.1. Ensuring the conformance by means of supervisors

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

The problem addressed here is how to force a known implementation model to conform to a reference specification. The proposed solution is to control the implementation with an internal controller and to compute it by control synthesis techniques, considering conformance with the specification as a control objective. The problem is attacked in the context of total and partial observation of the controller on the implementation [18][17]. This work is still under progress.

6.3.2. Robustness testing.

Participants: Thierry Jéron, Hervé Marchand, Valery Tschaen.

Robustness is the ability of a system to behave acceptably in the presence of hazards. One problem is the generation of test cases for testing robustness. Given a specification with modes (normal and degraded) and hazards, and required robustness properties, we propose an approach in two phases. First the specification should at least ensure robustness properties, which is achieved by control. Then test cases are selected, focussed on robustness. This is done by using a TGV-like technique on the controlled specification, but

focussed on mode changes and hazards, using test purposes derived from properties, and results of the control problem.

6.3.3. Testing and Control of Real-Time Systems

Participants: Ahmed Khoumsi, Thierry Jéron, Hervé Marchand, Vlad Rusu.

Before visiting us, Ahmed Khoumsi had contributed to the testing and control of real-time systems, based on a transformation of classes of Timed Automata (TA) into equivalent finite state automata called SetExp Automata (SEA). During his visit, we used this transformation as a basis for the generalization of two problems adressed in the project. First, for a class of Determinizable TA, we extended the TGV method to the real-time case [20]. Second, we extended the work on control for conformance for real-time discrete event systems and for a real-time extension of the **ioco** conformance relation.

6.3.4. Combining Formal Verification and Conformance Testing

Participant: Vlad Rusu.

This work [13] presents yet another combination of verification and conformance testing techniques to support the formal validation of reactive systems. The idea is to use symbolic test selection techniques to extract subgraphs (*components*) from a specification, and to perform the verification on the components rather than on the whole specification. Under reasonable sufficient conditions, this constitutes a sound compositional verification technique, in the sense that a property verified on the components also holds on the whole specification. This may considerably reduce the global verification effort. Moreover, once verified, a component forms the basis of an adequate test case, i.e., when executed on an implementation, it will not issue false positive or negative verdicts with respect to the verified properties. The approach has been implemented using the STG test selection tool STG and the PVS theorem prover, and demonstrated on a smart-card application (an electronic purse system).

6.4. Applications of theorem proving

Participant: Vlad Rusu.

6.4.1. Compositional verification of an ATM protocol

In this work [21] an approach based on compositionality, partial-order verification, and interactive theorem proving for verifying communication protocols is presented. The approach is implemented in PVS. Its originality lies more in the combination of the methods than in the methods themselves; and its value is that it scales up to real-size systems. It is demonstrated by verifying a real ATM protocol whose main requirement is to perform a reliable data transfer over an unreliable communication medium.

6.4.2. Extracting a Data Flow Analyser in Constructive Logic

This work has been done in cooperation with David Cachera, Thomas Jensen, and David Pichardie from the Lande project-team of Irisa. We show how to formalise a constraint-based data flow analysis in the specification language of the COQ proof assistant. This involves defining a dependent type of lattices together with a library of lattice functors allowing for a modular construction of complex abstract domains. Constraints are expressed via an intermediate representation that allows for efficient constraint resolution. Correctness with respect to an operational semantics is proved formally. The proof of existence of a correct, minimal solution of the constraints is constructive, which means that the extraction mechanism of COQ provides a provably correct data flow analyser in OCAML. The library of lattices together with the intermediate representation of constraints are defined in an analysis-independent fashion, thus providing a generic framework for proving and extracting static analysers in COQ. This work will be presented at ESOP in 2004.

6.5. Verification and Abstract Interpretation

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

Participants: Bertrand Jeannet, Wendelin Serwe.

Key words: *Interprocedural Verification*.

In the context of the ARC Modocop, we have proposed a new approach to interprocedural analysis/verification of programs, consisting in deriving an interprocedural analysis method by abstract interpretation of the standard operational semantics of programs [23]. The advantages of this approach are twofold. From a methodological point of view, it provides a direct connection between the concrete semantics of the program and the effective analysis, which facilitates implementation and correction proofs. Moreover, this method subsumes/integrates two main, distincts methods for interprocedural analysis, namely the call-string and the functional approaches introduced by Sharir and Pnueli. This enables strictly more precise analysis and additional flexibility in the tradefoff between efficiency and precision of the analysis.

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

6.5.2. Interprocedural Shape Analysis

Participant: Bertrand Jeannet.

Key words: *Interprocedural Verification*, *Shape Analysis*.

This work has been done in cooperation with Thomas Reps, of the University of Wisconsin–Madison, during my three months stay in this place. The goal of the shape analysis is to analyze the possible memory configurations occuring during the execution of a program performing dynamic allocation of objects in the memory heap. Of course the configurations computed by such an analysis abstracts the concrete memory configurations, usually by using graphs representing memory cells and their pointer relations. We have applied the interprocedural analysis method described above to shape analysis, using the abstract lattice of 3-valued logical structures developped by Thomas Reps, M. Sagiv and R. Wilhelm. The challenge was to apply the interprocedural method with a very complex abstract lattice, and to extend the abstract lattice with interprocedural operations.

6.5.3. Automatic state reaching for debugging reactive programs

Participant: Bertrand Jeannet.

Key words: *Debugging*, *Verification*, *Testing*.

This work has been done in collaboration with the synchronous team of VERIMAG [14]. Reactive systems are made of programs that permanently interact with their environment. Debuggers generally provide support for data and state inspection, given a sequence of inputs. But, because the reactive programs and their environments are interdependent, a very useful feature is to be able go the other way around; namely, given a state, obtain a sequence of inputs that leads to that state. The main technical contribution of this work is to propose an efficient solution to this problem for systems with numerical variables and inputs, which works well in practice, although the problem is equivalent to general safety properties verification, which is notoriously undecidable in presence of numeric variables. Three tools cooperate in order to solve the problem: the debugger LUDIC isolates the relevant parts of the program using slicing techniques. Then the verification tool NBAC computes the set of states that are both reachable from the starting (or initial) state and coreachable from the states satisfying the property P (or final states). Last, the sequence generator LURETTE uses this information for an efficient search for a (short) execution starting from the initial state and leading to some final state.

7. Contracts and Grants with Industry

7.1. Agedis

Participants: Thierry Jéron, Valery Tschaen, Erwan Demairy.

Key words: *test*, *test generation*, *distributed softwares*, *UML*.

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

7.2. ITEA EAST-EEA, Embedded Electronic Architecture

Participants: Thierry Jéron, Hervé Marchand.

The EAST-EEA project (ITEA-Project-No. 00009, http://east-eea.test.k-k.de/) is about the Embedded Electronic Architecture study with European leading automotive industry (PSA, Renault, Daimler, Volvo, etc) and suppliers (Valéo, Siemens, Bosch, etc) and academic groups. We were previously partners in **AEE Development Initiative** (htmladdnormallinkhttp://aee.inria.fr/http://aee.inria.fr/) 2000-2001 on Embedded Electronic Architecture with French automotive industry and suppliers. In both cases, the budget is managed by Inria Rocquencourt and for the moment we use it for travel only. Our motivation in this project is to improve the validation of embedded software, and in particular improve robustness testing of such software. In this context, we try to introduce automatic test generation in the design process of the automotive industry.

8. Other Grants and Activities

8.1. National grants & contracts

8.1.1. Carroll-Mutation

Participant: Thierry Jéron.

The objective of this project with CEA LIST, Thalès Airborne Systems and Thalès R & D, is to automate testing from UML models in the context of Model Driven Engeneering. Our participation consists in expertise for test generation and the use of the TGV tool, in collaboration with the Triskell project-team.

8.1.2. INRIA ARC Modocop

Participants: Bertrand Jeannet, Thierry Jéron, Vlad Rusu, Wendelin Serwe, François-Xavier Ponscarme.

Modocop (http://www-sop.inria.fr/lemme/modocop) is an ARC (Action de Recherche Coopérative de l'Inria), that started in January 2002. Modocop deals with the Model checking Of Concurrent Object-oriented Programs. The focus is on analysis, verification and test generation for Java programs. Partners are Inria project-teams LANDE, LEMME, OASIS, and VASY and CNRS Lab VERIMAG. VERTECS works on the analysis of Java programs and interfaces STG with (Synchronous) Java, as output for execution of test cases, and as input to allow a direct specification in a Java-like language. W. Serwe made his Post-Doc in this context.

8.1.3. CNRS Action Spécifique Test no. 23

Participants: Thierry Jéron, Hervé Marchand.

AS STIC 23 (http://www.laas.fr/TSF/AS23/) is a working group ("Action spécifique") in the context of CNRS networks RTP SECC and SdF, on "Advanced test techniques for complex systems". Partners are LAAS (Toulouse), LABRI (Bordeaux), LRI (Orsay), VERIMAG (Grenoble) and TRISKELL (Inria Rennes, 1C).
The objective is to define properly what robustness testing is, and to define new techniques for test generation for robustness. In this context, we proposed a new approach based on a combination of control synthesis and conformance test generation (see 6.3.2).

8.2. Collaborations

8.2.1. Collaborations with other INRIA project-teams:

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

8.2.2. Collaborations with French research groups outside INRIA:

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

8.2.3. International Collaborations

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

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

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

University of Wisconsin (T. Reps) on shape analysis. Bertrand Jeannet visited Tom Reps during spring 2003.

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

8.2.4. Networks of Excellence

We have been involved in three proposals for Networks of Excellence in the 6th PCRD: Define (http://www.laas.fr/DeSIRE&DeFINE/index.html) on Dependability that was not selected for the 1st call, and Artist (http://www.artist-embedded.org/) on Embedded Systems, and Testnet (http://www-lor.int-evry.fr/testnet/) on Integration of Testing Methodologies which have been submitted to the 2nd call.

9. Dissemination

9.1. University courses

- T. Jéron: is responsible of a course in Master of Computer Science in University of Rennes 1, and of a course in the engineering school DIIC in University of Rennes 1. He also teaches in the engineering school EnstB in Rennes and Brest. The topics of all lectures are testing and model-checking, He is also member of the “Comissions de Spécialistes” of ENS Cachan and University Rennes 1.
- V. Rusu teaches in the Master of Computer Science in Rennes, on deductive verification methods.
- B. Jeannet teaches in the Master of Computer Science in Rennes, on abstract interpretation,
- B. Gaudin is teaching in DEUG and DIIC 2 in University of Rennes 1 (64h/year).
- V. Tschaen is teaching in DEUG and DIIC 3 in University of Rennes 1 (64h/year).
- E. Demairy is teaching compilation and project management in IUP-2 and IUP-3 in University of Rennes 1 (40h).

9.2. PhD Thesis and Trainees

Current PhD. thesis:

1. Benoit Gaudin, « Control of Structured Discrete Event Systems »
2. Valéry Tschaen, « Automatic test generation: Models and Techniques »
3. Elena Zinovieva « Generation and simplification of symbolic testing »

Trainees :

1. Laurent Payen, « Syntool a Tool dedicated to the controller synthesis for Structured Discrete Event Systems », DESS Trainee (6 months).
2. Sophie Quinton, « Deductive proof with PVS », ENS Cachan (3 months).
3. Mbolatiana Rafamantanantsoa, « Robustness testing », Master student (6 months).

9.3. Participation to jurys

Thierry Jéron was member of the PhD. jurys of Manuel Aguilar (INPG Grenoble) and Armelle Prigent (Ecole Centrale de Nantes).

9.4. Conferences, Seminars, invited talks

Thierry Jéron is PC member of Fates'03 (Montréal, 10/03), Testcom 2004 (Oxford, 3/04), Tacas'05 (Edinburgh, 4/04). He is Financial Chair of ISSRE 2004 (St Malo, 11/04). He is member of the Organizing Committee of the Movep School (Liège, 12/04) .

Thierry Jéron was invited to give a tutorial on test generation in the school ETR'03 (Ecole Temps Réél, Toulouse).

10. Bibliography

Major publications by the team in recent years

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

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

[3] T. JÉRON. *TGV: théorie, principes et algorithmes.* in « Techniques et Sciences Informatiques, numéro spécial Test de Logiciels », number 21, 2002.

[4] T. JÉRON, P. MOREL. *Test generation derived from model-checking.* in « CAV'99, Trento, Italy », series LNCS, volume 1633, Springer-Verlag, N. HALBWACHS, D. PELED, editors, pages 108-122, Juillet, 1999.

[5] B. JEANNET, N. HALBWACHS, P. RAYMOND. *Dynamic Partitioning in Analyses of Numerical Properties.* in « Static Analysis Symposium, SAS'99 », series LNCS, volume 1694, 1999.

[6] H. MARCHAND, O. BOIVINEAU, S. LAFORTUNE. *On the Synthesis of Optimal Schedulers in Discrete Event Control Problems with Multiple Goals.* in « SIAM Journal on Control and Optimization », number 2, volume 39, 2000, pages 512-532.

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

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

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

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

Articles in referred journals and book chapters

[11] 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) », 2004, to appear.

[12] B. JEANNET. *Dynamic Partitioning In Linear Relation Analysis. Application To The Verification Of Reactive Systems.* in « Formal Methods in System Design », number 1, volume 23, July, 2003, pages 5–37.

[13] V. RUSU. *Combining formal verification and conformance testing for validating reactive sstems.* in « Journal of Software Testing, Verification, and Reliability », number 3, volume 13, September, 2003.

Publications in Conferences and Workshops

[14] F. GAUCHER, E. JAHIER, B. JEANNET, F. MARANINCHI. *Automatic State Reaching for Debugging Reactive Programs.* in « 5th Int. Workshop on Automated and Algorithmic Debugging, AADEBUG'03 », September, 2003.

[15] B. GAUDIN, H. MARCHAND. *Contrôle de systèmes à événements discrets hiérarchiques.* in « 4ième Colloque Francophone sur la Modélisation des Systèmes Réactifs, MSR'03 », Metz, France, October, 2003, (Version française de ECC'03).

[16] B. GAUDIN, H. MARCHAND. *Modular Supervisory Control of Asynchronous and Hierarchical Finite State Machines.* in « European Control Conference, ECC 2003 », Cambridge, UK, September, 2003, http://www.irisa.fr/vertecs/Publis/Ps/2003-ECC.pdf.

[17] T. JÉRON, H. MARCHAND, V. RUSU, V. TSCHAEN. *Ensuring the conformance of reactive discrete-event systems using supervisory control.* in « 42nd IEEE Conference on Decision and Control », Hawaii, USA, December, 2003.

[18] T. JÉRON, H. MARCHAND, V. RUSU, V. TSCHAEN. *Synthèse de contrôleurs pour une relation de conformité.* in « 4ième Colloque Francophone sur la Modélisation des Systèmes Réactifs, MSR'03 », Metz, France, October, 2003.

[19] A. KHOUMSI, J. DRISSI. *Méthode de construction de sous-module utilisant la théorie du contrôle des systèmes à événements discrets.* in « 4ième Colloque Francophone sur la Modélisation des Systèmes Réactifs, MSR'03 », Metz, France, October, 2003.

[20] A. KHOUMSI, T. JÉRON, H. MARCHAND. *Test Cases Generation for Nondeterministic Real-time Systems.* in « 3rd International Workshop on Formal Approaches To Testing of Software (FATES 2003) », Montréal, Québec, Canada, October, 2003.

[21] V. RUSU. *Compositional verification of an ATM protocol.* in « Formal Methods Europe (FME'03) », 2003, http://www.irisa.fr/vertecs/Publis/Ps/2003-FM.ps.gz.

Internal Reports

[22] B. GAUDIN, H. MARCHAND. *Supervisory Control of Structured Discrete Event Systems.* Technical report, number 1569, IRISA, November, 2003.

[23] B. JEANNET, W. SERWE. *Abstracting Call-Stacks for Interprocedural Verification of Imperative Programs.* Technical report, number 1543, IRISA, July, 2003, ftp://ftp.irisa.fr/techreports/2003/PI-1543.ps.gz.

Bibliography in notes

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

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

[26] Y. BRAVE, M. HEIMANN. *Control of Discrete Event Systems Modeled as hierarchical State Machines.* in « IEEE Transacations on Automatic Control », number 12, volume 38, December, 1993, pages 1803–1819.

[27] P. CAINES, V. GUPTA, G. SHEN. *The hierarchical control of ST-finite-state machines.* in « Systems and Control Letters », volume 32, 1997, pages 185-192.

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

[29] 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 », pages 238-252, Los Angeles, CA, January, 1977.

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

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

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

[33] H. MARCHAND, O. BOIVINEAU, S. LAFORTUNE. *On Optimal Control of a class of partially-Observed Discrete Event Systems.* in « Automatica », number 11, volume 38, October, 2002, pages 1935-1943.

[34] 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 », number 2, volume 21, feb, 1995, pages 107-125.

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

[36] 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 », number 1, volume 77, 1989, pages 81-98.

[37] M. SAGIV, T. REPS, R. WILHELM. *Parametric shape analysis via 3-valued logic.* in « ACM Transactions on Programming Languages and Systems », number 3, volume 24, 2002.

[38] M. SAGIV, T. REPS, R. WILHELM. *Solving shape-analysis problems in languages with destructive updating.* in « ACM Transactions on Programming Languages and Systems », number 1, volume 20, 1998.

[39] R. SENGUPTA, S. LAFORTUNE. *An Optimal Control Theory for Discrete Event Systems.* in « SIAM Journal on Control and Optimization », number 2, volume 36, march, 1998.

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

[41] K. C. WONG, W. M. WONHAM. *Hierarchical Control of Discrete-Event Systems.* in « Discrete Event Dynamic Systems », volume 6, 1996, pages 241-273.

[42] W. M. WONHAM, P. J. RAMADGE. *Modular Supervisory Control of Discrete Event Systems.* in « Mathematics of Control Signals and Systems », volume 1, 1988, pages 13–30.

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

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