Proposition de Projet

VerTeCS
Modèles et techniques de VÉRification appliquées au TEst et au Contrôle de Systèmes réactifs

VERification models and techniques applied to the TEsting and Control of reactive Systems

Thème Inria 1C, Rennes

T. Jéron, H. Marchand, V. Rusu

27 avril 2001

Mots clés : Test, contrôle, vérification, conformité, énuméré, symbolique, réparti, commande optimale, couverture, systèmes réactifs et critiques, télécommunications, systèmes embarqués, contrôle commande.

Ce document présente une proposition de projet de recherche dans le programme 1C. Les membres de ce projet proviennent des deux projets Pampa et Ep-Atr, tous deux également du programme 1C.

1 Personnels

Permanents

- Thierry Jéron (CR Inria), responsable scientifique,
- Hervé Marchand (CR Inria),
- Vlad Rusu (CR Inria)

Assistante de projet

- Marie-Noëlle Georgeault, Chrystelle Soulas

Doctorants

- Elena Zinovieva (boursière Inria)

Ingénieurs experts

- Duncan Clarke (Inria pour FormalCard)
- Simon Pickin (CNRS pour projet COTE; partagé avec Triskell)

2 Introduction

2.1 Objectif scientifique

L'objectif scientifique de notre projet est d'améliorer la fiabilité des systèmes réactifs et critiques, en fournissant à l'ingénieur logiciel des méthodes et des outils de génération de tests et de synthèse de contrôleurs pour ce type de systèmes.

Par systèmes réactifs on entend des logiciels qui interagissent avec leur environnement. Citons par exemple les protocoles de télécommunication, les logiciels embarqués, les systèmes de contrôle/commande, etc. Ces systèmes réactifs sont aussi souvent des systèmes critiques, au sens où l'occurrence d'erreurs au cours du fonctionnement du logiciel peut avoir des conséquences graves du point de vue économique ou de la sécurité des personnes.

La correction fonctionnelle de ce type de logiciel est donc primordiale. Cette correction peut être envisagée dès la conception et les techniques de **vérification** diverses (preuve, model-checking, analyse statique, etc) peuvent être très utiles dans cette phase. Toutes ces techniques ont pour but d'assurer qu'un logiciel ou, plus souvent, une spécification d'un logiciel, satisfait certaines propriétés fondamentales de bon fonctionnement. Si des erreurs sont découvertes, ces techniques permettent en général de les diagnostiquer.

Une solution alternative à la vérification est celle du **contrôle**: plutôt que de vérifier la correction du logiciel, on l'assure en interdisant les comportements qui risqueraient de sortir de l'ensemble des comportements acceptables. Cette restriction des comportements s'effectue en interdisant le tir de certaines actions dites contrôlables, sans toutefois pouvoir interdire les actions incontrôlables. La **synthèse de contrôleurs** consiste à construire automatiquement un contrôleur à partir d'une spécification du système et des propriétés de correction à assurer.

Enfin, même après un effort conséquent de vérification de la conception, le **test** de l'implémentation réelle est toujours nécessaire. De même le contrôleur produit devra aussi être testé. En effet, la vérification et la synthèse de contrôleurs s'effectuent sur un modèle du système qui, même s'il se veut proche du système réel, n'est jamais identique à celui-ci. L'activité de modélisation impose d'abstraire le comportement du système pour n'en considérer que les aspects essentiels et de faire des hypothèses sur l'environnement d'utilisation. Par conséquent, il se peut que certains comportements, non considérés dans le modèle pour la vérification ou la synthèse de contrôleurs, provoquent des erreurs lors de l'exécution du système réel. La vérification (ou le contrôle), même si elle est exhaustive, n'est donc exhaustive que sur un modèle du système, d'où le besoin de valider le système réel par des tests.

La tendance actuelle dans le monde industriel est plutôt à l'utilisation de méthodes de conception (objet en particulier), qui reposent sur l'utilisation d'une discipline de conception et sur l'outillage de cette discipline jusqu'aux phases de déploiement. Cette discipline limite bien sûr les erreurs de conception ou de codage mais elle ne les supprime pas complètement. L'utilisation conjointe de techniques de vérification permettrait d'aller plus loin dans la qualité du logiciel, mais son introduction dans le monde industriel se heurte encore à des difficultés, en particulier parce que la vérification demande une bonne connaissance des techniques, a un coût qu'il faut amortir par ses résultats sur la fiabilité, et surtout n'appartient pas déjà à la culture des ingénieurs. En revanche, le test fait déjà partie du cycle de développement des logiciels, et son coût est extrêmement important. En effet, il est largement reconnu que près de 30% du coût du logiciel est dû au test dans toutes ses variantes: test unitaire, test d'intégration, test système, test de non régression, etc. Or cette activité est encore peu automatisée.

L'automatisation ou l'assistance à la génération des tests[1] peut donc être rapidement rentable économiquement. Un prérequis à l'automatisation du développement des tests est de disposer du code ou d'un modèle du logiciel qui décrit ses fonctionnalités, et qui sert de référence. Ceci impose donc une corrélation forte avec l'activité de conception du logiciel. Le modèle est utilisé afin d'en extraire des comportements à tester. Malgré cet effort supplémentaire, des études de cas significatives ont apporté la preuve de sa rentabilité économique (voir [DEF+96] par exemple). De plus, le modèle formel nécessaire à la génération de tests peut alors être utilisé pour la simulation, la vérification ou la synthèse de contrôleurs. Nous pensons que l'utilisation de méthodes formelles pour l'automatisation du test peut être un vecteur important d'introduction plus massive de techniques formelles dans tout le cycle de développement des systèmes réactifs.

Notre ambition à moyen terme est de fournir à l'ingénieur logiciel des méthodes et des outils permettant d'automatiser le plus possible la génération des tests et la synthèse de contrôleurs à partir de spécifications formelles. Les moyens que nous mettons en œuvre sont la conception de modèles formels des objets et relations intervenant dans le test et le contrôle, et la conception d'algorithmes de génération de tests et de synthèse de contrôleurs. Nous fondons nos recherches sur les techniques de vérification telles que le model-checking, la preuve, l'analyse statique, la théorie du contrôle des systèmes à événements discrets, et les modèles et logiques qui sous-tendent ces techniques. Ce cadre mathématique est le moyen d'assurer la pertinence et des propriétés essentielles de correction des tests et contrôleurs produits.

Pourquoi nous intéressons nous à la fois à la génération de tests et à la synthèse de contrôleurs? Une première raison est qu'il existe une similitude forte entre les deux problématiques, et par conséquent, les solutions envisagées pour l'une sont potentiellement intéressantes pour l'autre. Les deux problèmes sont en effet très proches: il s'agit dans les deux cas d'extraire des sous-comportement d'une spécification en fonction de propriétés (appelés critères ou objectifs de contrôle ou de test). Il

1. Le terme "génération de test" est parfois utilisé dans l'industrie pour décrire la production de tests exécutables sur une plate-forme de test à partir de tests abstraits ou spécifications de tests. Il s'agit alors essentiellement d'un problème de compilation et de nombreux outils commerciaux résolvent ce problème. En revanche, dans ce document, le terme "génération de tests" signifie la production de tests abstraits, problème nettement plus difficile.

[DEF+96] L. Doldi, V. Encontre, J.-C. Fernandez, T. Jéron, S. Le Bricquir, N. Texier, and M. Phalippou. Assessment of automatic generation methods of conformance test suites in an industrial context. In B. Baumgarten and A. Burkhardt, H.-J.Giessler, editors, *IFIP TC6 9^{th} International Workshop on Testing of Communicating Systems*. Chapman & Hall, September 1996.

s'agit aussi de différencier les comportements contrôlables (sur lesquels le testeur/le contrôleur peut agir), des comportements qui sont seulement observables. Il faut aussi dans les deux cas considérer l'observabilité partielle du système, la compositionnalité, la répartition, la prise en compte simultanée des données et du contrôle. On peut déjà factoriser des efforts au niveau modèle, mais aussi au niveau des algorithmes et des outils. Nous avons déjà constaté cette similitude pour les résultats acquis et nous nous efforçons actuellement de la démontrer précisément dans un cadre commun. Nous espérons ensuite que cette synergie apportera des avancées significatives dans les deux domaines. Une autre raison est la complémentarité des deux techniques. Les contrôleurs implémentés doivent être testés vis à vis des contrôleurs synthétisés. Inversement, la génération de test peut profiter de techniques de synthèse de contrôleurs pour améliorer l'efficacité ou la pertinence de la génération de tests.

D'autre part, même si notre objectif n'est pas directement de s'attaquer à la problématique de la vérification, nous maintenons une veille technologique sur ce thème afin d'en utiliser certains résultats à nos problématiques. Inversement nous pouvons également espérer qu'une partie de nos recherches s'applique à la vérification. En particulier nos recherches sur le traitement symbolique des données, la compositionnalité et la répartition, peuvent être d'intérêt pour un cadre plus général que celui du test et du contrôle.

Un des aspects les plus novateurs de notre recherche est celui du traitement symbolique des données. C'est aussi un des aspects les plus transversaux. En effet il a clairement des applications dans la problématique de notre projet à la fois pour la génération de tests et la synthèse de contrôleurs, mais aussi au delà du projet pour la vérification, et plus largement dans les techniques d'analyse de programmes combinant données et contrôle.

Nous nous proposons de développer des modèles formels adaptés à différents types de systèmes réactifs, concevoir les algorithmes permettant l'automatisation de la génération de tests et leur exécution, la synthèse de contrôleurs, et de prototyper ces algorithmes dans des environnements logiciels. Nous devons aussi nous appuyer sur la réalité industrielle afin de choisir les bons formalismes et proposer des solutions en adéquation avec la pratique industrielle. Nous sommes motivés par la validation de nos travaux à travers la réalisation de prototypes en vue de leur transfert industriel et leur distribution dans le milieu académique. Des études de cas devront être traitées afin de mieux percevoir les réalités industrielles, ressourcer notre imagination, et repousser les limites d'applicabilité de nos outils.

Nos domaines d'application privilégiés sont les systèmes de télécommunication, les cartes à puces et les systèmes de contrôle commande sur lesquels nous avons acquis une certaine expérience et dans lesquels nos travaux passés ont trouvé des applications. Cependant, nous ne souhaitons pas *a priori* restreindre ce domaine d'application à un sous-ensemble trop strict de systèmes réactifs. En effet notre expérience nous a déjà permis de constater que des techniques pensées dans un cadre particulier (télécom par exemple) peuvent s'avérer utiles dans un autre cadre *a priori* différent (hardware [29], cartes à puces [17]).

2.2 Contexte

2.2.1 Le test

Le test est une activité largement répandue dans des domaines extrêmement variés. Du petit matériel électroménager (robot de cuisine, cafetière, etc) au grand système (avion, fusée, centrale nucléaire, etc), tous ces systèmes subissent des tests à divers degrés de développement avant leur mise sur le marché. Les logiciels, qui font d'ailleurs de plus en plus partie de ces systèmes, n'échappent pas à la règle.

Malgré son universalité, l'activité de test s'appuie sur des caractéristiques communes. Il consiste généralement en une stimulation du système sous test et en l'observation de son comportement ou de ses propriétés, suivi d'une comparaison avec une observation d'un système idéal de référence (souvent un comportement ou des propriétés modélisés mathématiquement). Le but premier du test est de détecter des erreurs (de conception ou de réalisation, suivant le niveau d'application du test) qui se caractérisent par des fautes (l'observation de la manifestation d'une erreur). Inversement, à travers le test, on souhaite aussi acquérir une certaine confiance dans la réalisation. C'est le cas lorsque le test ne détecte plus d'erreur. Cette confiance se mesure par la capacité d'un test à détecter des fautes, et est fonction de l'effort de test (en termes de quantité et de qualité). Sachant que le test ne peut pas être exhaustif, et ne fournit donc quasiment jamais de garantie de correction, les contraintes économiques imposent que l'effort de test soit un compromis entre la confiance apportée et son coût.

Nous nous intéressons ici à la fiabilité du logiciel, en particulier réactif, et donc au test de ce type de système. Le test de logiciel a plusieurs facettes, et les techniques de test diffèrent en fonction de plusieurs caractéristiques. D'abord, les techniques de test varient selon le niveau d'observabilité du code, depuis le test *boîte blanche* (dans lequel le code du système testé est connu), jusqu'au test *boîte noire* (dans lequel le code du système testé n'est pas connu et seule l'interface du système est accessible), en passant par le test *boîte grise* (ou le code est inconnu mais certaines informations internes sont disponibles). Les méthodes de test varient aussi suivant les aspects ou les propriétés testés: les fonctionnalités, les performances, la robustesse, la fiabilité, etc. Enfin, on n'emploie pas les mêmes techniques selon que l'on est le développeur, un utilisateur, ou une tierce partie donnant un avis indépendant sur le système testé.

Le test de conformité

Pour notre part, nous nous intéressons principalement au test dit de *conformité*. Le test de conformité est un type de test qui s'intéresse à la correction fonctionnelle d'une implémentation d'un système vis à vis d'une spécification de référence. C'est donc un test de type fonctionnel. C'est aussi un test de type boîte noire, car le code de l'implémentation n'est en général pas connu. Dans le domaine des télécommunications, le test de conformité est souvent réalisé par des laboratoires de test indépendants, sur des systèmes développés par des fournisseurs, afin de fournir un certificat de conformité utile pour des clients éventuels. Dans ce cadre, le fait que le code ne soit pas connu est dû à des raisons de confidentialité évidentes.

En pratique, le test de conformité consiste à faire interagir l'implémentation sous test (*IUT* pour *implementation under test*) avec son environnement. Cet environnement, qui est constitué par un ou plusieurs processus ou utilisateurs, est simulé par un ou un ensemble de testeurs. Ces testeurs exécutent des tests élémentaires appelés *cas de test* qui consistent en une suite d'interactions. Cette suite d'interactions contrôle l'IUT par des stimuli (des envois de messages, des appels de fonctions, de méthodes, etc) et observe ses réactions (messages reçus, résultats d'appels, etc). Les comportements de l'IUT sont contrôlés et observés à travers des interfaces (appelées *PCO* pour *Points de Contrôle et d'Observation* dans le monde télécom). Généralement, chaque *cas de test* a pour objectif le test d'une fonctionnalité précise, décrite dans ce qu'on appelle un *objectif de test*.

Le test de conformité est particulièrement utilisé dans le domaine des protocoles de télécommunication. Dans ce cadre une norme a été décrite, ISO 9646 [ISO] qui en décrit les ingrédients

[ISO] ISO. *Information Processing Systems, Open Systems Interconnection, OSI Conformance Testing Methodology and Framework.* ISO 9646.

principaux. Mais le test de conformité n'est pas confiné à ce cadre, et le même type de test peut être utilisé pour des systèmes réactifs quelconques, et en particulier pour les systèmes critiques.

Le gain potentiel en terme de réduction du coût du test, apporté par l'automatisation de la génération de test de conformité, a suscité un grand intérêt à la fois chez les industriels et les universitaires depuis de nombreuses années. Cependant, bien que la génération automatique de tests connaisse depuis plusieurs années un réel succès industriel dans le domaine du matériel, elle a eu plus de mal à percer dans le domaine du logiciel. Une des raisons, est peut être la trop grande distance entre les techniques universitaires et la pratique industrielle [LL95]. Cette distance tend à diminuer mais un effort reste à faire pour rendre nos techniques plus utilisables sur des cas réels.

Techniques de génération

Essentiellement, deux types de méthodes coexistent. Ces méthodes diffèrent sur le modèle utilisé pour représenter les comportements des spécifications de systèmes et par conséquent sur les techniques mises en œuvre.

La première famille, la plus ancienne, est celle des méthodes fondées sur les automates (voir le *survey* [LY94] par exemple). Elle est issue des méthodes de test de circuits et des problèmes de reconnaissance de machines [Moo56,Gil62]. Sous certaines hypothèses à la fois sur la spécification et l'implémentation, ces techniques permettent de tester exhaustivement l'implémentation pour toutes les fautes dans un certain modèle de faute (caractérisant le type de fautes observables). Les algorithmes cherchent à construire une unique séquence, la plus courte possible, qui permette de montrer l'équivalence entre les comportements de l'implémentation et ceux de la spécification. L'atout de ces méthodes est leur exhaustivité. Mais cette exhaustivité est surtout théorique. En effet, si les hypothèses faites sur la spécification sont parfois acceptables (contrôlabilité, déterminisme, minimalité) elles sont invérifiables sur l'implémentation. De plus les algorithmes de génération sont assez coûteux, ce qui interdit de les utiliser sur des systèmes de taille réelle. Ces techniques semblent donc cantonnées à traiter des spécifications peut complexes, en particulier dans le domaine du matériel.

La deuxième famille est fondée sur les systèmes de transitions (LTS pour *Labelled transition system*) et est issue des problèmes d'équivalence et de préordre de test dans les algèbres de processus [DNH84,Abr87,Phi87,Bri88]. Au départ dédiées à la formalisation du problème du test, ces théories

[LL95] R. Lai and W. Leung. Industrial and Academic Protocol Testing: the Gap and the Means of Convergence. *Computer Networks and ISDN Systems*, 27:537–547, 1995.

[LY94] Lee and Yannakakis. Testing Finite State Machines: State Identification and Verification. *IEEE Trans Computer*, 43(3):306–320, 1994.

[Moo56] E. F. Moore. Gedanken-experiments on sequential machines. *Automata Studies*, pages 129–153, 1956.

[Gil62] A. Gill. *Introduction to the Theory of Finite State Machine*. New-York, McGraw-Hill, 1962.

[DNH84] R. De Nicola and M. Henessy. Testing Equivalences for Processes. *Theoretical Computer Science*, 34:83–133, 1984.

[Abr87] S. Abramsky. Observational Equivalence as a Testing Equivalence. *Theoretical Computer Science*, 53(3), 1987.

[Phi87] I. Phillips. Refusal testing. *Theoretical Computer Science*, 50(3):241–284, 1987.

[Bri88] E. Brinskma. A Theory for the Derivation of Tests. In *Protocol Secification, Testing and Verification VIII*, pages 63–74. North-Holland, 1988.

ont ensuite évolué vers des algorithmes et des outils de génération de tests [Pha94,Tre95,FJJV96]. Les modèles utilisés permettent de représenter les comportements de systèmes réels, et les hypothèses faites à la fois sur la spécification et sur l'implémentation sont réduites au minimum. Au lieu de chercher à montrer l'équivalence de comportements comme dans les techniques évoquées plus haut, on cherche à montrer une inclusion de (certains) comportements (un préordre ou une relation de conformité). Les modèles et techniques permettent aisément de prendre en compte le non-déterminisme et l'observation partielle. Des algorithmes efficaces existent maintenant [27] et ont permis de traiter des études de cas de taille industrielle [FJJV97], [3].

Notons aussi qu'au sein de l'ISO, un groupe de chercheurs de ces deux communautés ont essayé de présenter ces deux types de techniques dans un cadre unifié. Il en est résulté le document *Formal Methods in Conformance Testing* [IJW96] qui décrit les ingrédients essentiels à la description d'une technique de génération de tests. En particulier, ces ingrédients sont constitués des modèles de spécifications, d'implémentations, de tests, de la définition de la conformité (modèle de faute ou relation de conformité), de la formalisation de l'exécution d'un test sur une implémentation, des verdicts associés, d'un algorithme de génération et des propriétés attendues des tests générés.

Le test structurel

Les travaux que nous menons depuis plusieurs années sur le test de conformité nous amènent aussi à nous intéresser à un autre type de test, le test *structurel*.

Le test structurel est un test de type boîte blanche, où la sélection des tests se base sur la structure du code source de l'implémentation [RW85]. Cette sélection est aussi souvent fondée sur la notion de *critère de couverture*. Un critère de couverture définit la façon dont les constructions de base du code source de l'implémentation sont testées. Par exemple, on cherche à tester au moins une fois chaque branche, les enchaînements de définition/utilisation de variables, à exercer les conditionnelles de différentes façons, etc. Nous souhaitons adapter le concept de critère de couverture au cadre du test de conformité et l'appliquer à la spécification pour aider à la sélection des tests (voir en 3.3).

2.2.2 Le contrôle

La proportion sans cesse croissante de l'informatique dans les systèmes de contrôle commande (concernant la production manufacturière, les systèmes robotiques, le contrôle de trafics de véhicules, la logistique, les réseaux de communication, etc.) amène à nous interroger de manière approfondie

[Pha94] M. Phalippou. *Relations d'Implantations et Hypothèses de Test sur les Automates à Entrées et Sorties.* PhD thesis, Université de Bordeaux, 1994.

[Tre95] J. Tretmans. Testing Labelled Transition Systems with Inputs and Outputs. In *8th International Workshop on Protocols Test Systems*, Evry - France, September 1995.

[FJJV96] J.-C. Fernandez, C. Jard, T. Jéron, and G. Viho. Using on-the-fly verification techniques for the generation of test suites. In A. Alur and T. Henzinger, editors, *Conference on Computer-Aided Verification (CAV '96), New Brunswick, New Jersey, USA*, volume 1102 of *LNCS*. Springer-Verlag, July 1996. Egalement disponible en rapport de recherche Irisa n^o 1036.

[FJJV97] J.-C. Fernandez, C. Jard, T. Jéron, and G. Viho. An experiment in automatic generation of conformance test suites for protocols with verification technology. *Science of Computer Programming*, 29:123–146, 1997. Egalement disponible en rapport de recherche Irisa n^o 1035 et Inria n^o 2923.

[IJW96] ITU-T SG 10/Q.8 ISO/IEC JTC1/SC21 WG7. Information Retrieval, Transfer and Management for OSI; Framework: Formal Methods in Conformance Testing. Committee Draft CD 13245-1, ITU-T proposed recommendation Z 500. ISO - ITU-T, 1996.

[RW85] S. Rapps and E. Weyuker. Selecting software test data using data flow information. *IEEE Transactions on Software Engineering*, 11(4):367–375, 1985.

sur les questions suivantes : dans quelle mesure les logiciels qui sont au cœur de ces systèmes sont-ils fiables? Comment est il possible d'améliorer les techniques et les méthodes permettant d'accroître cette fiabilité? Cette fiabilité devient d'autant plus importante que l'on considère des systèmes de contrôle commande critiques. Jusqu'à présent, le problème de la validité de tels systèmes de contrôle commande s'avérait relativement simple du fait de la faible complexité des applications envisagées. Les problèmes de contrôle pouvaient alors être appréhendés de manière intuitive. Une fois le système de contrôle commande spécifié, les techniques classiques de validation pouvaient être alors utilisées (validation par vérification formelle ou model checking, simulation du système de contrôle commande en fonction d'un jeu de tests donné par l'expert, etc). Le problème devient vite très difficile quand on considère des applications de grande taille ou des applications de grande complexité combinatoire.

La *théorie du contrôle*, traditionnellement utilisée dans le cadre des systèmes à espaces d'états et de signaux continus, permet de considérer une autre approche. Le système de contrôle-commande est élaboré en utilisant des méthodes de construction garantissant *a priori* les propriétés attendues du système global. La phase de validation est ainsi restreinte aux seules propriétés non garanties par construction et, expérimentalement, à l'évaluation des écarts dus aux erreurs ou approximations dans la modélisation.

La même approche a été adoptée dans le domaine des systèmes à événements discrets. Ici deux approches différentes du contrôle coexistent.

- dans la première, le modèle physique est représenté par un langage dont l'alphabet représente les différentes actions possibles; un mot de ce langage représente alors une séquence d'actions correspondant à une trajectoire souhaitée du système. Dans cette approche, l'action du contrôleur a pour effet d'interdire, à différents moments, un certain nombre de lettres de l'alphabet, appelées événements contrôlables [RW89]. Cette approche a notamment été mise en œuvre sur des automates [RW89] et des réseaux de Petri [II87,HKG97].
- l'autre approche [BHWTF93] [9], à notre sens plus proche de la réalité industrielle et qui relève d'avantage de l'approche des processus continus, consiste à fournir sur certains des ports d'entrée du système (dits événements contrôlables), les événements induisant un comportement souhaité du système vis à vis des objectifs de contrôle (on parlera alors de contrôle entrées/ sorties).

Dans les deux cas, le contrôle revient à ajouter à un système non contrôlé des contraintes supplémentaires induisant une réduction du comportement du système à un comportement souhaité.

Ce comportement souhaité est obtenu à partir de la spécification d'un objectif de contrôle (dans les situations concrètes, on est amené a composer plusieurs objectifs pour obtenir le comportement souhaité[RW86]). Là encore, il existe plusieurs formalismes pour représenter ces objectifs. Dans l'approche Ramadge & Wonham, ces objectifs s'exprimaient comme des langages (inclus dans le langage de la spécification). L'idée du contrôle était alors de dégager de ce langage de contrôle, le langage le

[RW89] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE; Special issue on Dynamics of Discrete Event Systems*, 77(1):81–98, 1989.

[II87] A Ichikawa and K. Iraishi. Analysis and control of discrete-events sytems represented by petri nets. In *Discrete Event Sytems: Models and Applications, IIASA conf.*, Sopron, Hungary, august 1987.

[HKG97] L.E. Holloway, B.H. Krogh, and A. Giua. A survey of Petri net methods for controlled discrete event systems. *Discrete Event Dynamic Systems: Theory and Application*, 7:151–190, 1997.

[BHWTF93] S. Balemi, G. J. Hoffmann, H. Wong-Toi, and G. F. Franklin. Supervisory control of a rapid thermal multiprocessor. *IEEE Transactions on Automatic Control*, 38(7):1040–1059, july 1993.

[RW86] P. J. Ramadge and W. M. Wonham. Modular supervisory control of discrete event systems. In A. Bensoussan and J. L. Lions, editors, *Proc. of 7th Int. Conf. Analysis and Optimization of Syst.*, volume 83 of *LNCIS*, pages 202–214, Antibes, France, June 1986. Springer-Verlag , Berlin, Germany.

plus permissif et contrôlable (seuls les événements contrôlables peuvent être interdits par le contrôleur). En d'autres termes, ce langage correspond au comportement le plus permissif que le système peut avoir s'il veut à la fois respecter l'objectif de contrôle et le critère de contrôlabilité. A partir de ce langage, un contrôleur peut alors être vu comme une fonction qui, en fonction de l'histoire du système (i.e, la trajectoire que le système a emprunté jusque là), va envoyer au système l'ensemble des événements contrôlables qui doivent être interdits pour rester dans le cadre du comportement contrôlé. Cette théorie a par la suite été entendue au cadre de l'observation partielle [BGK+90] sur le système à contrôler, et utilisé pour la synthèse de contrôleurs décentralisés [YL00].

Toutefois, même si la spécification d'objectifs de contrôle par le biais de "langages-objectifs" est très générale et permet d'exprimer des propriétés relativement complexes, elle reste assez inhabituelle. Dans la pratique, on préfère exprimer des propriétés comme "*on aura toujours ...*", ou encore "*il arrivera que ...*", propriétés pouvant s'exprimer notamment à l'aide de logiques temporelles telles que CTL. Le problème du contrôle est alors le suivant : étant donné un système et une propriété d'états sur ce système (e.g., invariance, atteignabilité, persistance, récurrence, etc), comment restreindre le comportement du système de manière à ce que celui-ci vérifie cette propriété. Les techniques employées pour calculer ce contrôleur s'appuient alors sur celles développées en vérification, et impliquent le calcul de points fixes appliquées à des transformateurs de prédicats (e.g., pre, $\tilde{pre}$) prenant en compte le caractère incontrôlable de certains événements. Des algorithmes permettant de calculer des contrôleurs à la volée ont également été développés [TA99]. Notons pour finir que des objectifs tels que l'atteignabilité ont été étendus de manière à pouvoir tenir compte de critères quantitatifs. On parlera alors de commande optimale [SL98,PA89] (voir section 3.3 pour plus de détails).

3 Axes de recherche

Les techniques de vérification par modèle (ou model-checking) sont l'une des bases de nos recherches sur la génération de tests et la synthèse de contrôleurs. Ces techniques partent de la donnée d'une spécification dans un langage dont la sémantique permet de modéliser l'ensemble des comportements possibles par un modèle simple comme des automates ou des systèmes de transitions. Ces systèmes de transitions peuvent être représentés de deux manières:

- sous forme explicite ou énumérative par énumération des états et transitions du modèle si celui-ci est fini,
- sous forme implicite ou symbolique par une formule caractérisant l'ensemble des états et des transitions du modèle.

[BGK+90] R. D. Brandt, V. K. Garg, R. Kumar, F. Lin, S. I. Marcus, and W. M. Wonham. Formulas for calculating supremal and normal sublanguages. *Systems and Control Letters*, 15(8):111–117, 1990.

[YL00] T. Yoo and S. Lafortune. A general architecture for decentralized supervisory control of discrete-event systems. In *Proc of 5th Workshop on Discrete Event Systems, WODES 2000*, Ghent, Belgium, august 2000.

[TA99] Stavros Tripakis and Karine Altisen. On-the-fly controller synthesis for discrete and dense-time systems. In Jeanette M. Wing, Jim Woodcock, and Jim Davies, editors, *FM'99—World Congress on Formal Methods, Volume I*, volume 1708 of *Lecture Notes in Computer Science*, pages 233–252, Toulouse, France, septembre 1999. Springer.

[SL98] R. Sengupta and S. Lafortune. An optimal control theory for discrete event systems. *SIAM Journal on Control and Optimization*, 36(2), march 1998.

[PA89] K. M. Passino and P. J. Antsaklis. On the optimal control of discrete event systems. In *Proc. of 28th Conf. Decision and Control*, pages 2713–2718, Tampa, Floride, december 1989.

Par exemple, pour une variable d'entrée de la spécification, les techniques énumératives énumèrent le domaine de valeur de cette entrée alors que les techniques symboliques représenteront l'ensemble des valeurs possibles par une formule dont les modèles (au sens logique) sont ces valeurs.

Les techniques énumératives ont l'avantage d'être proches des modèles sous-jacents, mais souffrent souvent du problème de l'explosion combinatoire qui résulte de l'énumération. Cette explosion combinatoire peut empêcher de construire entièrement le système de transitions d'une spécification aussitôt que celle-ci manipule des données ou de la composition parallèle, et limiter ainsi l'applicabilité de la vérification.

Durant ces dix dernières années, un des axes de recherche principaux en vérification a été de tenter d'éviter l'explosion combinatoire par différentes techniques comme la vérification à la volée [FJJM93], les techniques symboliques pour traiter les données [McM93], et les techniques de type ordre partiel [God90,Val90,Pel94] ou à base de modèles de vrai parallélisme comme les structures d´événements [Win89] pour traiter la concurrence.

La génération de tests et la synthèse de contrôleurs n'échappent pas à ces problèmes d'explosion combinatoire, et les alternatives aux techniques énumératives pour la vérification sont d'un intérêt certain dans notre problématique. Encore faut-il adapter ces techniques et les améliorer.

Depuis le milieu des années 90, on a vu un essor important de l'utilisation de méthodes formelles pour la génération de tests. Nous avons participé à cet essor depuis le début. Au départ, la plupart des techniques de génération de tests fondées sur les méthodes formelles n'étaient cependant que des réutilisations de techniques de vérification, comme le model-checking. Elles ne répondaient que partiellement au problème car elles manquaient de spécificités par rapport à la problématique du test. Les choses évoluent et la communauté scientifique en prend conscience, comme en témoignent l'émergence de nombreux *workshops* et conférences affichant le thème de l'utilisation de techniques formelles dans le test. La synthèse de contrôleurs est également en plein essor comme en témoignent les nombreuses publications récentes sur le sujet. Notre recherche est à la pointe de cette mouvance, et nous devons maintenant aller plus loin pour rendre ces techniques plus utilisables dans le milieu industriel.

3.1 Techniques énumératives et à la volée

Depuis plusieurs années, nous avons élaboré des techniques énumératives de génération de test et de synthèse de contrôleurs fondées sur des modèles simples de type systèmes de transitions ou automates. Nous avons pu en démontrer l'efficacité, en particulier dans le cas du test par l'utilisation de techniques à la volée, dans plusieurs études de cas et dans différents domaines d'application. Ces techniques représentent encore le meilleur de l'état de l'art, mais nous pouvons encore les améliorer. D'autre part, les modèles simples sont une bonne base pour comparer les techniques de génération de test et de synthèse de contrôleurs.

[FJJM93] J.-C. Fernandez, C. Jard, T. Jéron, and L. Mounier. On-the-fly verification of finite transition systems. *Formal Methods in System Design*, 1:251–273, 1993.

[McM93] K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, Englewood Cliffs, 1993.

[God90] P. Godefroid. Using partial orders to improve automatic verification methods. In *2nd CAV Workshop*, volume 531 of *LNCS*, pages 176–185. Springer Verlag, 1990.

[Val90] A. Valmari. A stubborn attack on state explosion. In *2nd CAV Workshop*, volume 531 of *LNCS*, pages 156–165. Springer Verlag, 1990.

[Pel94] D. Peled. Combining partial order reductions with on-the-fly model-checking. In D. L. Dill, editor, *CAV Workshop*, volume 818 of *LNCS*. Springer Verlag, 1994.

[Win89] G. Winskel. An introduction to event structures. In *LNCS 354*, pages 364–397. Springer Verlag, 1989.

Test à la volée

Nous avons développé ces dernières années des techniques de génération de tests énumératives [FJJV96,JM97],[27]. Ces techniques partent de la donnée d'une spécification formelle, dont la sémantique s'exprime dans un modèle de système de transitions dans lequel on distingue les événements contrôlables, observables ou internes. La formalisation du test, issue principalement des travaux de Tretmans et al. [Tre96], consiste en particulier à définir l'ensemble des implémentations conformes par une relation de conformité.

Pour palier partiellement au problème d'explosion combinatoire, nous avons élaboré des techniques de génération de tests originales fondées sur l'exploration à la volée de systèmes de transitions. Il s'agit essentiellement de contraindre l'exploration du système de transitions de la spécification par un automate qui modélise un objectif de test, et de produire un cas de test pour l'objectif donné, en n'explorant que la partie de la spécification relative à cet objectif. La difficulté réside ici dans l'algorithmique. Celle-ci induit une combinaison à la volée des algorithmes nécessaires à la génération et opérant des transformations du système de transitions de la spécification tout en le composant avec l'objectif.

La technologie mise en œuvre a atteint une certaine maturité. Les algorithmes ont été implémentés dans l'outil TGV (voir section 5) et a donné lieu à un transfert industriel dans l'outil TestComposer de l'environnement ObjectGéode (Telelogic) [18].

Il n'empêche que le sujet n'est pas clos. Les expérimentations effectuées avec TGV nous montrent que des améliorations significatives sont encore possibles. Par exemple l'utilisation conjointe de techniques fondées sur la vérification dite "ordre partiel" serait clairement un plus, encore faut-il les adapter à la problématique du test, en particulier au cadre de l'observation partielle.

D'autre part, nos techniques sont basées sur une relation de conformité qui, bien qu'adaptée au contexte des protocoles, l'est moins pour d'autres types d'applications. La relation, en restant réaliste, pourrait être plus discriminante, en allant au delà d'une relation sur les traces. De plus, la relation de conformité actuelle ne possède pas de bonnes propriétés de compositionnalité qui permettrait d'appréhender de façon plus satisfaisante le problème du test dans un contexte (le système à tester n'est accessible qu'à travers un autre système, supposé correct) ou du test modulaire (par composition des tests de processus élémentaires). Ces problèmes de compositionnalité, bien qu'importants, sont peu abordés dans le littérature.

Contrôle énumératif

En synthèse de contrôleurs, nous avons également utilisé des techniques énumératives. Dans cette approche fondée sur les automates [8, 42], il s'agissait principalement d'utiliser des techniques usuelles de programmation dynamique et des algorithmes issues du problème du voyageur de commerce pour synthétiser des contrôleurs optimaux pilotant un système à travers un ensemble de tâches tout en minimisant un critère de performance portant sur les événements du système à contrôler (avec une observation totale ou partielle de celui-ci) (voir section 3.3).

[FJJV96] J.-C. Fernandez, C. Jard, T. Jéron, and G. Viho. Using on-the-fly verification techniques for the generation of test suites. In A. Alur and T. Henzinger, editors, *Conference on Computer-Aided Verification (CAV '96), New Brunswick, New Jersey, USA*, volume 1102 of *LNCS*. Springer-Verlag, July 1996. Egalement disponible en rapport de recherche Irisa n^o 1036.

[JM97] T. Jéron and P. Morel. Abstraction, τ-réduction et déterminisation à la volée: application à la génération de test. In *CFIP'97, Congrès Francophone sur l'Ingéniérie des Protocoles, Liège, Belgique*. Hermes, September 1997.

[Tre96] J. Tretmans. Test generation with inputs, outputs and repetitive quiescence. *Software - Concepts and Tools*, 17, 1996.

Test et contrôle

C'est également dans ce cadre énumératif que nous avons remarqué que les problèmes, les modèles et les algorithmes de génération de tests et de synthèse de contrôleurs étaient très similaires. Des travaux existent déjà sur la synthèse de contrôleurs à la volée [TA99] qui, bien que très proches de la génération de tests à la volée, ne font pas le lien. Nous avons donc commencé à étudier l'unification, au moins partielle, de ces deux problématiques, traitées pour l'instant par des communautés assez éloignées. Pour cela, il est nécessaire de bien comprendre les vocabulaires spécifiques, d'unifier autant que possible les modèles et de factoriser au maximum les algorithmes. Ensuite, on peut espérer que cette unification permettra de faire des avancées dans les deux domaines. Par exemple, il semble que les algorithmes de génération de tests soient plus élaborés pour certains aspects, comme la prise en compte du non-déterminisme ou la sélection. En revanche, les objectifs de test ont moins de puissance d'expression que les objectifs de contrôle. Certains aspects, comme l'observation partielle, sont traités différemment alors qu'ils pourraient être appréhendés de la même manière. Nous espérons donc une synergie importante entre la génération de tests et la synthèse de contrôleurs qui permettra de faire avancer l'état de l'art dans les deux domaines. Ceci devrait nous conduire assez rapidement à l'implémentation, sur la base de TGV, d'algorithmes de synthèse de contrôleurs.

D'autre part, nous avons des idées sur la combinaison des techniques de génération de test et de synthèse de contrôleurs. Il est clair qu'après synthèse d'un contrôleurs, celui-ci est implémenté et que le test de l'implémentation est une étape nécessaire. Inversement, les techniques de synthèse de contrôleur peuvent aider à la génération de test, par exemple en restreignant les comportements de la spécification.

3.2 Techniques symboliques

Les techniques énumératives de génération de tests et de synthèse de contrôleurs atteignent leur limite quand on s'attaque à des spécifications manipulant beaucoup de données, ou des données sur de grands domaines (voire infinis), ou encore des spécifications paramétrées. Au mieux, on est face à une explosion combinatoire, au pire les systèmes sont infinis.

De plus, les tests et contrôleurs produits par les techniques énumératives instancient les variables d'entrée. Or, les tests de conformité (dits *abstraits* par opposition à *exécutables*) sont souvent des programmes réactifs qui manipulent des variables. Ils sont aussi parfois paramétrés, tout comme les spécifications, afin d'être plus génériques. La situation est identique pour les contrôleurs. Générer ce type de test ou de contrôleur est donc d'un intérêt certain pour se rapprocher des tests ou contrôleurs écrits à la main. Mais les techniques de génération doivent s'appuyer sur des modèles plus riches que les systèmes de transitions. Ces modèles symboliques doivent prendre en compte variables et paramètres, sans les instancier contrairement à ce qu'on fait dans les systèmes de transitions. Dans le cas le plus simple, le problème à résoudre est un problème d'extraction de programme: extraire la partie du système correspondant à un objectif donné. Ce problème est en général indécidable. Viennent s'y ajouter des problèmes inhérents au test et au contrôle de systèmes réactifs: différence entre contrôle et observation, problème d'observabilité partielle, etc.

[TA99] Stavros Tripakis and Karine Altisen. On-the-fly controller synthesis for discrete and dense-time systems. In Jeanette M. Wing, Jim Woodcock, and Jim Davies, editors, *FM'99—World Congress on Formal Methods, Volume I*, volume 1708 of *Lecture Notes in Computer Science*, pages 233–252, Toulouse, France, septembre 1999. Springer.

Synthèse de contrôleurs et BDD

Dans le cadre de la synthèse de contrôleur, nous utilisons déjà des techniques symboliques à base de BDD (Binary Decision Diagram [Bry86]). Issues du model-checking, ces techniques de codage permettent de maîtriser l'explosion combinatoire par le codage efficace de la relation de transition obtenue à partir d'une abstraction booléennes des programmes à contrôler. À partir de cette abstraction, il est alors possible de dériver des contrôleurs en fonction d'objectifs de contrôle qui s'expriment comme des propriétés d'états (e.g., l'invariance, l'atteignabilité, la persistance, la récurrence, etc). Les algorithmes sont basés (tout comme en vérification) sur des calculs de plus grands (plus petits) points fixes et sur des transformateurs de prédicats prenants en compte l'aspect contrôlable de certains événements du système)[9]. Toutefois, même si ces techniques nous ont permis de dériver des contrôleurs pour des systèmes de tailles raisonnables (plusieurs centaines de millions d'états), il reste, à l'heure actuelle, difficile de dériver un programme de contrôle du contrôleur symbolique produit.

Test symbolique

Dans le cadre du test nous avons commencé à étudier le problème de la génération de tests symboliques et avons obtenus des premiers résultats encourageants [39]. En posant quelques restrictions syntaxiques (absence de boucles sur les actions internes par exemple), on sait produire, à partir d'un système de transitions à entrés-sorties symbolique (IOSTS) et d'un objectif de même type, un graphe de test (IOSTS) correct par construction (non-biaisé, non laxiste, etc) et le plus général possible. La construction décrite est syntaxique, c'est à dire faite sans interpréter les opérations sur les variables. Les algorithmes utilisés sont similaires à ceux de TGV mais appliqués sur les graphes de contrôle. Ceux-ci sont déjà implémentés dans le prototype STG (voir section 5).

Le graphe de test symbolique généré n'est cependant pas un test comme aurait pu l'écrire un ingénieur. En effet, il peut contenir des chemins infranchissables si les conditions de franchissement de certaines transitions sont insatisfiables. Bien entendu, le problème de l'exécutabilité des chemins est indécidable dans notre modèle général. Plusieurs possibilités peuvent alors être envisagées. Soit on restreint *a priori* le modèle pour rendre le problème décidable. Le risque est alors de ne plus pouvoir modéliser des cas réels. Cette solution n'est donc envisageable que dans un domaine d'application particulier, ou alors dans un modèle intermédiaire obtenu par abstraction.

Soit on utilise des techniques d'abstraction. On peut utiliser toute la panoplie des techniques d'interprétation abstraite [CC77], qui permettent d'obtenir des résultats sur un modèle abstrait où le problème d'accessibilité peut être décidable, et d'en déduire des résultats sur le modèle concret. Ces techniques fonctionnent bien dans le cadre de la vérification. On peut en effet prouver que des abstractions sont conservatives pour des classes de propriétés intéressantes (inclusion de comportement pour les propriétés de sûreté). Le problème est différent dans le cas du test et on peut facilement montrer que des abstractions conservatives pour la vérification ne le sont pas dans le cas du test. Le problème est en effet différent. Il s'agit d'assurer que les tests produits sur une abstraction (corrects par construction sur cette abstraction) restent corrects sur le système concret. La correction signifie que seules les implémentations non-conformes sont rejetées par le test. Cela revient donc à assurer que les implémentations conformes à la spécification concrète, doivent rester conformes à

[Bry86] R.E. Bryant. Graph-based algorithms for boolean function manipulations. *IEEE Transaction on Computers*, C-45(8):677–691, august 1986.

[CC77] P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction of approximation of fixed points. In *Proceedings of the 4th ACM Symposium on Principles of Programming Languages, Los Angeles*, pages 238–252, New York, NY, 1977. ACM.

l'abstraction. Ceci serait assuré si la relation de conformité était une inclusion, mais ce n'est pas le cas en général. Pour la relation que nous utilisons, il faudrait combiner sur-approximation sur les sorties avec sous-approximations sur les entrées, ou bien changer de relation de conformité.

Enfin, on peut utiliser des techniques de preuve semi-automatiques pour démontrer qu'un chemin est infranchissable.

Toutes ces techniques peuvent d'ailleurs être combinées pour une même application.

Grâce à l'implémentation de STG, nous avons pu commencer à expérimenter nos techniques sur plusieurs études de cas significatives et quelques exemples jouets. Bien que nos solutions soient encore limitées, nous avons déjà des résultats intéressants. Il nous faudra cependant lever certaines limitations imposées sur les modèles (comme l'absence de cycle d'actions internes). Il nous faut également continuer notre travail sur la combinaison des techniques de génération de test avec les techniques de preuve, interprétation abstraite, propagation et résolution de contrainte. Nous devrons ensuite appliquer ces techniques sur d'autres cas de taille réelle.

Des recherches sur l'utilisation de techniques symboliques en génération de test sont menées ailleurs. Mais, à notre connaissance, aucune ne cherche à produire des tests symboliques. Notre recherche est donc tout à fait originale. D'autre part, nous espérons que ces travaux, menés dans le cadre de la génération de tests symboliques seront, à plus long terme, adaptés au cadre de la synthèse de contrôleurs où là non plus, personne ne s'est intéressé à produire des tests symboliques.

3.3 Optimalité, qualité

Un des problèmes récurrents en test et en contrôle est celui de l'optimalité. Cette optimalité se décline sous plusieurs aspects, suivant le référentiel, et suivant qu'on parle de test ou de contrôle.

Optimisation du contrôle sur les systèmes testés

Pour le test, un problème important est celui d'obtenir les tests qui contrôlent le plus possible l'implémentation afin de l'amener dans des comportements que l'on souhaite observer. Nous avons déjà abordé ce problème et exploré une solution par l'utilisation la théorie des jeux [38]. On modélise l'interaction testeur/implémentation par un jeu, et la recherche d'une stratégie optimale pour le testeur induit un test qui optimise le contrôle qu'il a sur l'implémentation à tester. Cette recherche mériterait d'être poursuivie afin de la rendre utilisable, en particulier dans le cadre de la génération à la volée. Elle est aussi proche de techniques de synthèse de contrôleurs fondées sur la théorie des jeux.

Commande optimale

Pour le contrôle, certains objectifs de contrôle (l'atteignabilité par exemple) peuvent se traduire par la question simple : est il possible de "piloter" le système considéré vers un ensemble donné d'états en garantissant certaines propriétés tout au long de la trajectoire? La théorie du contrôle permet de synthétiser le contrôleur le plus permissif pilotant le système de l'ensemble de départ vers l'ensemble d'arrivée (ce qui en soi est un critère d'optimalité qui se rapproche de celui décrit dans le cadre du test : maximum de liberté vs minimum de contrôlabilité). A ce niveau, une deuxième question peut alors se poser : quelle est la meilleur façon d'y arriver, la plus rapide, la moins coûteuse, la plus sécuritaire . . . ? Ce type de problème s'appelle *problème de la commande optimale*. Le champ d'action de la commande optimale est extrêmement vaste. Il couvre le contrôle de tous les systèmes dynamiques où une performance optimale est souhaitée. Le problème intrinsèque de la commande optimale est le choix du critère sur lequel l'optimalité va porter. Dans certains cas, ce critère est directement imposé par le système. Dans d'autres, au contraire, le choix du critère est (totalement) arbitraire et dépend fortement de la personne qui pose le problème. Cette notion a notamment été étudiée dans [SL98] [8] et étendue au problème de l'observation partielle dans [42]. Intuitivement, l'idée de la commande optimale est de calculer un contrôleur qui pilote un système de manière à forcer celui-ci à achever une tâche en minimisant un certain critère de performance. Dans cette approche, des coûts sont affectés à chaque événement. À partir de cette fonction de coûts, il est possible de définir le coût d'une trajectoire comme étant la somme des coûts des événements qui interviennent dans cette trajectoire. Le but du contrôleur est alors de restreindre le comportement du système de manière à ce que celui-ci n'emprunte que des trajectoires de coûts minimaux. D'autres critères d'optimalité portent plus sur les moyens à utiliser pour atteindre un objectif logique, plutôt que sur le but à atteindre [11]. Ces objectifs s'expriment en ordonnant les états du système suivant une certaine relation d'ordre. Dans ce cas, le contrôle revient à choisir, pour un état donné, la commande faisant évoluer le système vers l'état considéré comme étant le meilleur pour la relation d'ordre. Nous parlerons alors d'objectif d'optimalité statique pour les objectifs de contrôle exprimés comme une relation d'ordre entre états, et de commande optimale statique. Les objectifs d'optimalité statiques peuvent également être composés avec des objectifs de contrôle logiques de manière à obtenir potentiellement des lois de contrôle (expression des commandes en fonction des entrées non contrôlables et des états).

Couverture des tests

Un autre aspect d'optimalité du test est celui de la couverture, c'est à dire la mesure de la pertinence d'une suite de tests par rapport aux comportements possibles du système, donc aux fautes détectables. On sait bien sûr que l'exhaustivité n'existe pas dans le domaine du test, puisque l'ensemble des comportements du système est infini alors que l'exécution d'une suite de tests est finie : seulement un ensemble fini de séquences finies est en final exercé. Un moyen de contourner le problème est de faire des hypothèses sur l'implémentation tels que la régularité ou l'uniformité [BGM91]. Une alternative est de définir des critères de couverture [RW85] qui s'expriment justement en une propriété sur un ensemble fini de séquences finies. Dans le cadre du test boîte blanche de programmes séquentiels, plusieurs notions de couverture plus ou moins fines existent telles que la couverture de sommets, de branches, de conditions, etc. Ces notions sont définies sur le code du programme source. Dans le cas du test boîte noire, on peut utiliser le même type de critère sur la spécification qui est le modèle de référence. Bien que ces critères reposent sur des heuristiques, il est communément admis qu'ils donnent de bons résultats en termes de détection de fautes. Certains résultats démontrent même qu'une couverture de 100% d'un certain critère permet de détecter toutes les erreurs d'un modèle de faute [Kuh99]. Il existe des outils commerciaux qui permettent de produire des séquences de test couvrant 100% du critère de couverture, mais ce critère est défini sur la syntaxe du programme en utilisant des heuristiques. Les séquences résultantes ne sont donc pas forcément exécutables. Pour étudier ce problème nous avons défini un sous-modèle d'automates étendu avec arithmétique de Presburger et fonctions non-interprétées, qui permet de décrire des programmes paramétrés assez

[SL98] R. Sengupta and S. Lafortune. An optimal control theory for discrete event systems. *SIAM Journal on Control and Optimization*, 36(2), march 1998.

[BGM91] G. Bernot, M. C. Gaudel, and B. Marre. Software Testing Based on Formal Specification: a Theory and a Tool. *Software Engineering Journal*, pages 387–405, November 1991.

[RW85] S. Rapps and E. Weyuker. Selecting software test data using data flow information. *IEEE Transactions on Software Engineering*, 11(4):367–375, 1985.

[Kuh99] D. Richard Kuhn. Fault classes and error detection capability of specification-based testing. *ACM Transactions on Software Engineering and Methodology*, 8(4):411–424, October 1999.

généraux avec des variables entières et des types de données plus complexes comme des tableaux de taille paramétrée. Nous avons démontré la décidabilité du problème de l'atteignabilité bornée pour ces modèles par simulation symbolique [47] et une implémentation est en cours. Ceci permettrait de séparer les séquences exécutables de celles qui ne le sont pas et ainsi d'affiner la mesure de couverture exacte, voire interagir avec le calcul syntaxique de séquences pour obtenir un meilleur taux de couverture.

3.4 Compositionnalité, répartition

Contrôle hiérarchique

Que ce soit en vérification, génération de tests ou en synthèse de contrôleurs, la méthodologie de spécification appliquée est le plus souvent la suivante: les systèmes à contrôler, tester et/ou à vérifier sont à l'origine décrits de manière hiérarchique (analyse descendante en génie logiciel). Puis la synthèse et/ou la vérification s'appliquent sur le système mis à plat (toute la modularité verticale a été oubliée). Sachant que la complexité des algorithmes de calcul croît exponentiellement avec le nombre d'états des systèmes mis en parallèle et imbriqués, il apparaît crucial de chercher à garder cette hiérarchie lors du calcul des contrôleurs et/ou des testeurs. L'objectif est donc d'étudier des techniques générales de simplification portant à la fois sur le système et sur les algorithmes de synthèse. Dans ce contexte, la prise en compte de la modularité verticale semble être une voie prometteuse: il est possible de spécifier des objectifs de contrôle à différents niveaux selon la "vue" que l'on a du système. (e.g., objectif d'ordonnancement de tâches à haut niveau, objectifs de sécurité propres, plus bas dans la hiérarchie). Le problème est relativement difficile: comment synthétiser des contrôleurs et/ou des testeurs à un certain niveau de la hiérarchie qui restent génériques lorsqu'on les plonge dans un contexte de plus haut niveau?

Cette étude est déjà commencée. Un premier modèle hiérarchique a été décrit. Il s'agit d'une généralisation des structures de Kripke Hiérarchiques développées par [AY98]. À partir de ce modèle, des algorithmes de synthèse de contrôleurs pour des objectifs de contrôle traitant de l'invariance et l'atteignabilité ont également été décrits. Ces algorithmes garantissent que la compositions des sous-systèmes hiérarchiques contrôlés par leurs contrôleurs locaux est équivalent au système global et contrôlé par un contrôleur global. On évite ainsi le problème d'explosion combinatoire dû au calcul du système global, ainsi que la décomposition d'un contrôleur global en un ensemble de contrôleurs locaux.

Test réparti

Il est assez courant que les systèmes réactifs soient répartis pour des raisons intrinsèques à l'architecture sur lesquels ils sont déployés. Tester ou contrôler ce type de système pose alors de nouveaux problèmes.

Dans le cas le plus général du test, à la fois le système sous test et les testeurs constituent des systèmes répartis. Les testeurs doivent communiquer entre eux pour se coordonner et communiquer avec le système. De plus, la communication testeur/testé est en général asynchrone, ce qui diminue le contrôle et déforme l'observation que le testeur a sur le système. Même dans le cas simple où un seul testeur communique de façon asynchrone avec un seul composant, nous sommes déjà face à cette perte de contrôle et d'observation. Nous avons montré dans le cas d'une communication FIFO

[AY98] R. Alur and M. Yannakakis. Model checking of hierarchical state machines. In *Sixth ACM Symposium on the Foundations of Software Engineering*, volume 23, 6 of *Software Engineering Notes*, pages 175–188, New York, 1998. ACM Press.

qu'une instrumentation du système permettait cependant de reconstruire l'ordre d'occurrence des événements sur le système et de regagner ainsi la puissance d'observation d'un test synchrone [22]. Dans le cas d'un système sous test lui même réparti, on ne peut reconstruire un ordre total, mais seulement un ordre partiel codant la causalité et la concurrence, mais seulement au prix d'une instrumentant du système par des horloges logiques. Les observations sont alors des ordres partiels et non plus des traces. Si on considère toujours une relation de conformité fondée sur les traces, il faut considérer comme observations l'ensemble des linéarisation possibles d'un ordre partiel. On peut facilement constater que le test doit être plus permissif que dans le cas centralisé synchrone, car il ne peut plus distinguer certains comportements. Si maintenant le testeur lui-même est réparti, se pose de plus le problème de la coordination des testeurs entre eux afin de contrôler et observer le système. Nous avons traité partiellement ce cas en exhibant un schéma de distribution d'un test séquentiel en tests répartis s'appuyant sur un service de consensus distribué [20]. Un test séquentiel produit à partir d'une spécification du système global incorporant l'environnement asynchrone de communication testeur/testé est alors distribué en un ensemble de tests répartis ayant la même puissance.

Ces premiers résultats et observations nous ont permis de mieux cerner la problématique du test réparti. On voit qu'une instrumentation du système réparti sous test nous permettrait d'observer des ordres partiels. On peut alors envisager de revisiter la théorie du test réparti avec des modèles de vrai parallélisme plutôt que sur des modèles d'entrelacement où la concurrence a disparu. Les observations codant concurrence et causalité, les relations de conformité doivent être fondées sur ces modèles. Pour la génération automatique, partant d'une spécification dans un tel modèle, on peut espérer produire des tests dans le même type de modèle et les répartir.

Cependant, en l'absence d'instrumentation du système, la concurrence est impossible à observer. Ceci n'empèche cependant pas de considérer des spécifications et de générer des tests dans des modèles de vrai parallélisme.

Cette recherche sera menée en collaboration avec Claude Jard du futur projet TRISKELL. Elle trouve son application dans le test de composants logiciels auquel s'intéresse ce projet. On pourra utiliser des Message Sequence Charts ou des diagrammes de séquence UML comme modèles des tests. D'un point de vue théorique, une collaboration avec le futur projet SYCOMORE sur la synthèse de code réparti sera aussi fort utile.

La plupart des problèmes du test réparti se retrouvent dans le cadre du contrôle réparti. Sauf qu'on peut raisonnablement se placer dans la situation qu'un contrôleur réparti est local et interagit de façon synchrone avec le système contrôlé.

3.5 Expressivité des objectifs de contrôle et de test

Que ce soit en synthèse de contrôleurs ou en génération de tests, la sélection se fonde sur un même type d'objet: un objectif. La puissance d'expression de ces objectifs de contrôle et de test a donc une grande influence sur les deux activités. L'expressivité des objectifs dépend aussi des modèles de spécification sous-jacents : systèmes de transitions, modèles symboliques, modèles de vrai parallélisme.

Dans le cadre du test, les objectifs de test que nous avons considérés expriment seulement des propriétés d'atteignabilité. Le choix de ce type de propriété a été guidé par la vision des objectifs comme abstraction des tests à produire et par le fait que les tests ne permettent de capturer que ce type de propriété. Or si l'on considère les objectifs de test non plus comme des abstractions de tests, mais plutôt comme des propriétés servant à la sélection de tests, on peut envisager de décrire d'autres types de propriétés. En effet, il est usuel en model-checking de définir la sémantique d'une formule par les états qui satisfont cette formule. Rien n'empêche donc de caractériser un ensemble d'états par une propriété d'invariance ou de vivacité. Le test ne permet pas de tester l'invariance (il peut seulement le tester négativement) ni la vivacité (ni positivement, ni négativement). Le test généré pour une propriété d'invariance ou de vivacité permet de tester des propriétés sur des préfixes finis de longueur arbitraire. Les logiques utiles dans ce contexte doivent permettre d'exprimer la différence entre événements contrôlables et événements incontrôlables (observables ou internes). C'est le cas par exemple de la logique ATL [AHK98]. Pour le contrôle, les résultats sont plus avancés de ce point de vue car les propriétés des objectifs de contrôle couvrent un spectre plus large. Cependant, il n'y a pas encore de logique reconnue permettant d'exprimer au moins les objectifs de contrôle usuels et pour lesquels la construction effective d'un contrôleur est possible. Des recherches sont actuellement menées par S. Pinchinat (SYCOMORE) pour répondre à la question *a priori* plus simple mais déjà difficile de la décidabilité de l'existence d'un contrôleur pour des logiques différenciant contrôle et observation. Nul doute que les résultats de ces recherches seront utiles pour la synthèse de contrôleurs et la génération de tests.

L'expression des objectifs doit aussi être en accord avec les modèles de description des spécifications. En effet, si on considère des modèles de vrai parallélisme pour décrire les spécifications, les tests ou les contrôleurs, on a envie d'exprimer des objectifs avec le même type de formalisme. Le même problème se pose dans le cadre symbolique où nous exprimons les objectifs de test par des automates étendus, qui sont traités symboliquement.

4 Programme de recherche à trois ans

Combinaison test et contrôle (axe 1)

Participants : *Thierry Jéron, Hervé Marchand, X doctorant*

L'objectif est de contribuer à la théorie et à la pratique du test et du contrôle en se fondant a priori sur les modèles de systèmes de transitions. Il s'agit d'une part de réexaminer les fondements en tentant de factoriser ce qui est factorisable pour le test et le contrôle. On s'attachera en particulier à examiner de nouvelles relations de conformité plus discriminantes, effectivement testables et ayant de bonnes propriétés de compositionnalité (axe 4). Ce cadre permettra également d'entamer notre réflexion sur l'expressivité des objectifs de tests (axe 5). Il s'agit d'autre part de modifier en conséquences les techniques de génération de test et de synthèse de contrôleurs. On examinera aussi l'utilisation de techniques à la volée pour la synthèse de contrôleurs et des techniques de type ordre partiel conjointement à la génération de tests à la volée.

Test symbolique (axe 2)

Participants : *Vlad Rusu, Elena Zinovieva, Duncan Clarke, Thierry Jéron*

L'objectif est de contribuer à la théorie sur les techniques symboliques dans le test et de les amener à un niveau de maturité tel qu'elles soient utilisables pratiquement sur des systèmes de taille réelle. Cet objectif nécessite d'incorporer à la génération de test, des techniques élaborées de propagation et résolution de contraintes, des calculs d'invariants, d'interprétation abstraite, de preuve.

[AHK98] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Lecture Notes in Computer Science*, 1536:23–60, 1998.

Il s'agit aussi d'étudier le problème de l'exécution des tests symboliques et sa complémentarité avec la génération (dans quelle mesure les problèmes non résolus à la génération peuvent l'être à l'exécution).

Ces recherches seront valorisées principalement dans le cadre des applications cartes à puces.

A plus long terme, nous essaierons, grâce aux résultats obtenus dans la combinaison du test et du contrôle, d'adapter les techniques symboliques à la synthèse de contrôleurs.

Vérification déductive et test symbolique (axe 2)

Participants : *Vlad Rusu, Duncan Clarke*

Il s'agit d'optimiser les techniques de vérification déductives (e.g. preuve d'invariants) en automatisant la génération et la preuve d'invariants inductifs dans un contexte [61]. Un prototypage de premiers résultats est en cours. Ce travail initié dans le cadre de notre veille sur les techniques de vérification devrait trouver des débouchés dans cadre de la génération de test. En effet, les invariants constituent potentiellement des objectifs de test intéressants pour une phase ultérieure de génération de tests symboliques.

Couverture des tests et simulation symbolique (axe 3)

Participants : *Vlad Rusu, Elena Zinoviewa, Thierry Jéron*

Il s'agit ici d´étudier la simulation symbolique et ses résultats théoriques pour les appliquer à la génération de tests, en particulier pour affiner les mesures de couverture de tests et améliorer les techniques syntaxiques de couverture. Les notions de couverture de code séquentiels devront au passage être adaptées à la couverture de spécifications pour le test de conformité et donc prendre en compte les relations de conformité considérées.

Contrôle hiérarchique et commande optimale (axes 3 et 4)

Participants : *Hervé Marchand, Y (doctorant)*

Il s'agit ici d'étendre les résultats obtenus pour le contrôle hiérarchique sur un premier modèle hiérarchique et des objectifs de contrôle de type invariance et atteignabilité. Il s'agit donc d'étendre le modèle et les objectifs de contrôle, en particulier vers des objectifs qualitatifs pour faire de la commande optimale hiérarchique.

Test réparti et vrai parallélisme (axe 4)

Participants : *Thierry Jéron,* Triskell, Sycomore

Il s'agit d'étudier l'apport des modèles de vrai parallélisme pour la génération de tests de systèmes répartis.

Dans l'hypothèse d'une instrumentation du système sous test par des horloges logiques, cas où la concurrence peut être observée, on pourra envisager de revisiter la théorie du test, ceci afin de tenir compte de la concurrence à la fois dans les relations de conformité et le type de tests produits. Sans cette hypothèse permettant l'observabilité de la concurrence, ces modèles pourront toutefois être utilisés pour des relations de conformité fondées sur les traces. Dans ce cas, ces modèles permettront à la fois d'éviter l'explosion combinatoire lors de la génération et d'obtenir des tests dans des modèles de vrai parallélisme, par exemple exprimés par des MSC ou des diagrammes de séquence en UML. Afin d'être utilisables sur des systèmes réels, ces techniques devront prendre en compte l'observation partielle et être adaptées à la génération à la volée.

5 Logiciels et prototypes

Nos recherches se placent dans le contexte du génie logiciel. Dans ce contexte, il est important, pour que les techniques que nous promouvons puissent être transférées dans le milieu industriel, de les démontrer par des outils. Ces outils doivent être suffisamment évolués pour leur permettre de passer à l'échelle. Nous présentons ici les outils et prototypes du projet. Ces outils sont à des stades différents de leur cycle de vie, et par conséquent, ne nécessitent pas la même quantité de travail. TGV pour la génération de tests, a déjà été transféré dans un outil commercial, mais nous en maintenons une version académique à des fins expérimentales et à titre de démonstrateur dans plusieurs contrats industriels. SIGALI pour la synthèse de contrôleurs, est aussi un prototype abouti qui sert de support expérimental. STG est par contre un outil en cours de développement et sur lequel nous comptons, dans les années à venir, démontrer notre savoir faire en techniques symboliques, à la fois sur la génération de test et la synthèse de contrôleurs.

Ces outils ont un intérêt pour d'autres projets de l'Irisa: TGV et STG pour TRISKELL dans sa connexion avec UMLAUT, SIGALI pour ESPRESSO dans l'environnement Signal, à plus long terme STG pour SYCOMORE et ESPRESSO pour BDL. Inversement, UMLAUT nous sert à valoriser TGV et probablement STG à l'avenir. Nous espérons maintenir et développer cette interaction forte entre nos outils et factoriser les efforts de développements relatifs. Le recrutement actuel d'un ingénieur Génie Logiciel à l'Irisa permettra certainement d'y contribuer.

5.1 TGV

TGV (Test Generation with Verification technology) est un prototype de générateur automatique de tests de conformité à partir de spécifications (SDL, LOTOS, UML). Son originalité tient dans son algorithmique à la volée. TGV est issu d'un projet commun avec le laboratoire Verimag Grenoble. Il utilise des bibliothèques de la boîte à outils CADP de Vérimag et de l'Inria Rhône-Alpes. Une première version de TGV a été déposé par l'Inria à l'Agence de Protection des Programmes en 97.

Les premiers développements de TGV datent de 95 et ont vu le jour lors d'un contrat financé par le STEI regroupant Vérilog, Cap Sesa Région Rennes, le Cnet Lannion et le Celar de Bruz, Vérimag et Pampa. La première version développée par Vérimag et Pampa était utilisable sur les graphes d'états de spécifications SDL produits par Géode.

Depuis cette première version, TGV a été constamment amélioré du point de vue algorithmique mais aussi par l'interfaçage avec de nouveaux environnements. TGV génère maintenant des tests à la volée (sans construire complètement le graphe d'état de la spécification), à la fois sur des spécifications SDL grâce à sa connexion à ObjectGéode, sur des spécifications Lotos grâce à sa connexion à l'environnement CADP et sur des spécifications UML grâce à sa connexion à l'outil UMLAUT du projet TRISKELL. TGV peut aussi fonctionner sur des graphes explicites décrits au format Aldébaran ou dans le format compressé BCG. En sortie, TGV construit un cas de test dans le format Aldébaran ou BCG ou dans le langage TTCN (*Tree and Tabular Combined Notation*).

TGV a été conçu de façon à être relativement indépendant des langages de spécification. Cette indépendance est obtenue par son architecture en couches par l'intermédiaire d'API (*Application Programming Interface*) fournissant les fonctions de parcours de systèmes de transitions. Ces différentes couches utilisent aussi des bibliothèques de stockage de graphes de CADP. La première couche, la seule qui soit spécifique au langage d'entrée, permet de connecter TGV soit à l'API du simulateur ObjectGéode pour SDL, soit à l'API fournie par le compilateur Lotos Caesar de la boîte à outils CADP, soit à une API similaire fournie par UMLAUT, soit encore aux API de parcours de graphes explicites.

En 99, TGV a bénéficié d'une mise à jour en vue de sa distribution. L'architecture de TGV a été repensée complètement pour mieux correspondre à sa découpe fonctionnelle et permettre une plus grande modularité. Les fonctionnalités ont été améliorées, en particulier pour permettre une plus grande expressivité des objectifs de tests, des fichiers de masquage et de renommage, en généralisant l'utilisation des expressions régulières. Le calcul des temporisateurs a été simplifié et incorporé au cœur de TGV. Le calcul des postambules a été affiné par la notion d'état stable. Un module appelé VTS, et pouvant se substituer au module principal de TGV, permet aussi de vérifier et corriger à la volée des cas de tests. Ceci a servi à valider TGV par la vérification des cas de tests produits et sert à la correction de tests manuels.

TGV est distribué gratuitement à des fins de recherche dans la boîte à outils CADP. Il fonctionne sur les systèmes Solaris, Linux et Windows NT. Il a été et est toujours utilisé par les membres du projet sur plusieurs études de cas et aussi par quelques utilisateurs extérieurs.

Une opération de transfert industriel de TGV avec la société Telelogic (Vérilog à l´époque) et le soutien de France Télécom a été réalisée en 98-99 [18]. L'objet en était en particulier l'implantation des algorithmes de TGV et son architecture dans l'outil ObjectGéode. Une bonne part des développements relatifs à TGV ont été effectués dans le projet Pampa. La version commerciale est vendue sous le nom Test Composer.

L'outil TGV a maintenant atteint un niveau de reconnaissance certain dans la communauté industrielle et académique internationale. Nous l'utilisons dans le cadre de plusieurs contrats, soit tel quel avec ses diverses interfaces langage, soit pour l'adapter à un contexte particulier d'étude. TGV servira aussi encore pendant quelques temps de démonstrateur d'idées pour y développer de nouveaux algorithmes fondés sur les systèmes de transition, pour le test et la synthèse de contrôleur.

5.2 Sigali

Sigali est un système de calcul formel interactif spécialisé dans l'analyse symbolique de systèmes de transitions modélisés par une relation de transition implicite (et à un niveau inférieur par un BDD). Il est destiné à la vérification des propriétés statiques et dynamiques de programmes Signal. Il permet également la synthèse de contrôleurs de systèmes à événements discrets. L'adjonction de primitives de création et de manipulation de fonctions à valeurs entières autorise les calculs de commande optimale. Sigali est étroitement couplé avec l'environnement Signal permettant ainsi un prototypage rapide du système en Signal, le calcul du contrôleur par l'intermédiaire de Sigali et un retour en Signal en vue de simuler le système contrôlé permettant ainsi de visualiser rapidement le résultat de la synthèse [9].

Sigali est un logiciel déposé à l'APP sous le numéro
IDDN.FR.001.370006.S.P.1999.000.10600. Comme pour l'environnement Signal, Sigali peut être obtenu après signature d'une convention de mise à disposition.

5.3 STG

STG (Symbolic Test Generation) est un prototype de générateur de tests symboliques. Cet outil nous sert pour l'instant de plate-forme pour implémenter les résultats de nos recherches sur le test symbolique et représente la génération future de TGV. Nous comptons également utiliser ou adapter certaines parties de STG pour la synthèse de contrôleur. STG est donc pour nous l'outil sur lequel nous mettrons le plus d'effort de développement à moyen terme.

STG est implémenté en C++ et fait appel à l'outil de dessin de graphes DOTTY et, actuellement, au solveur de contraintes Omega (http://www.cs.umd.edu/projects/omega). Il prend en entrée des spécifications et des objectifs de test symboliques décrits dans une extension du langage IF de Vérimag. Sous certaines hypothèses sur la spécification, STG permet de calculer un *graphe de test* par manipulation syntaxique, résolution de contraintes locales et extraction de sous-graphe à la façon de TGV. Ce graphe de test n'est pas tout à fait un test, mais représente un ensemble de tests correspondant à l'objectif. La production d'un vrai test nécessitera de supprimer les transitions non tirables et d'améliorer la sélection des tests en utilisant des outils sémantiques tels que l'interprétation abstraite, la preuve, la propagation et résolution de contrainte, le calcul d'invariants (voir 3.2).

STG permet aussi l'exécution des tests sur une implémentation. Actuellement, un test exécutable en C++ est produit à partir de sa représentation interne et peut être lié à une implémentation C++ du système à tester (développée par ailleurs). Différentes stratégies sont utilisables pour résoudre le non-déterminisme dans le code exécutable du test. Les paramètres de l'implémentation et du test doivent être fixés à l'exécution pour les rendre exécutables. Lors de l'exécution, la résolution de contraintes (Oméga) est utilisée pour résoudre les contraintes liées aux choix d'émission du testeur.

Nous avons déjà expérimenté STG sur quelques exemples significatifs dont une spécification partielle d'un porte-monnaie électronique de taille conséquente. Les graphes de test produits par manipulation syntaxique et extraction sont déjà extrêmement intéressants et fournissent déjà une aide certaine à la génération de tests. Ces études de cas nous montrent aussi le chemin à faire pour arriver à produire des tests plus proches de ceux qu'on voudrait écrire manuellement.

Le prototype STG s'inscrit dans une méthodologie plus large intégrant les techniques symboliques dans le test et appelé TwiST (*Testing with Symbolic Techniques*). Ceci inclut notre recherche sur la simulation symbolique et la génération/preuve d'invariants inductifs pour lesquels nous avons également commencé à développer un prototype.

6 Domaines d'application

Télécom

Le domaine des télécommunications est pour nous un domaine privilégié pour l'application des techniques de génération de tests. C'est un domaine dans lequel le processus de test est assez bien défini comme le montre l'existence de la norme ISO 9646 [ISO] qui définit entre autres le langage de description de tests TTCN. L'aspect réparti y a une importance croissante. De plus c'est un domaine où les méthodes formelles ont une place importante puisque l'ISO et ITU ont normalisé les trois langages de spécification formelle Estelle, Lotos et SDL. Nombre d'outils de vérification et de test fondés sur ces langages y sont utilisés et peuvent bénéficier des avancées de nos recherches.

La plupart de nos recherches passées sur le test ont été valorisées dans ce contexte. Nul doute que nos recherches futures y verront des débouchés.

[ISO] ISO. *Information Processing Systems, Open Systems Interconnection, OSI Conformance Testing Methodology and Framework*. ISO 9646.

Logiciel embarqué

De plus en plus de logiciel est maintenant embarqué dans des systèmes de natures différentes. Ce logiciel embarqué est souvent aussi réparti pour satisfaire des contraintes d'efficacité et de fiabilité. On le voit par exemple dans l'automobile, où les différents composants logiciels ressemblent de plus en plus à des protocoles de communication.

A l'opposé en termes de taille de système, les cartes à puces possèdent aussi une partie logicielle de plus en plus importante. Bien que ces logiciels soient plus orientés vers les aspects sécurité, la fiabilité de ces logiciels est aussi importante.

Systèmes de contrôle-commande

Outre les logiciels embarqués, les systèmes de contrôle-commande s'avèrent être l'un des domaines d'applications privilégiés pour la synthèse de contrôleur et le test. Les systèmes en question contrôlent en général des installations ou des machines coûteuses (usine de production manufacturière, système robotique, etc), qui sont pour la plupart en interaction avec un environnement (incluant du personnel humain). Ce sont donc des systèmes à sécurité critique. On a donc aussi des préoccupations de sûreté de la programmation, et de méthodes de validation et vérification, fondées sur des modèles formels. Dans cette optique, les apports tant du point de vue de la synthèse de contrôleurs (qui permet de dériver des contrôleurs sûrs à partir de spécifications) que du test (qui permet notamment de tester la validité de l'implémentation du contrôleur obtenu) sont primordiales pour ce type d'applications.

7 Insertion dans la communauté scientifique

Communauté scientifique nationale

Inria

Dans le programme 1C, nos relations privilégiées sont bien sûr avec les nouveaux projets émanant de nos projets d'origine, avec lesquels nous souhaitons vivement préserver des interactions fortes en mettant en avant notre complémentarité pour apporter des réponses communes ainsi qu'en factorisant nos développement logiciels.

Le futur projet TRISKELL (J.-M. Jezequel) s'attaque à la construction fiable de composants logiciels. TRISKELL est ancré dans le logiciel objet pour les télécommunications et a le désir, en particulier, de fournir des outils de vérification et de test dans ce domaine. Notre collaboration a été fructueuse dans le passé dans le cadre du projet Pampa puisqu'elle a déjà permis la connexion du logiciel UMLAUT de manipulation de modèles UML avec notre outil de génération de tests TGV [25]. Nous avons aussi collaboré à la résolution du problème d'ordonnancement de tests d'intégration de composants objets [6], qui mènera prochainement à son transfert dans ObjectTearing. Dans ces deux cas notre apport s'est situé au niveau modèle, algorithmique et outil et celui de TRISKELL au niveau modèle de composants objets, et transformation de modèles UML. Nous continuons cette collaboration à travers le projet RNTL Cote (voir section 8) qui a en particulier pour but de fournir des outils d'aide à la vérification et au test de composants logiciels. Nous espérons donc maintenir ces liens forts et cette complémentarité. En particulier, nous espérons que nos travaux sur les techniques symboliques pour le test permettront de traiter symboliquement des composants objets. Pour cela, il est nécessaire de modifier le niveau d'interconnexion de nos outils. L'interface entre TGV et UMLAUT se situe actuellement au niveau système de transitions. Pour pouvoir utiliser notre prototype STG à partir de modèles UML, il faudra qu'UMLAUT fournisse une interface au niveau d'entrée de STG, c'est à dire dans un format d'automates étendus communicants comme IF. Un autre point de collaboration est celui du test réparti. Les composants logiciels télécom sont par nature des objets répartis et la génération de tests pour de tels composants devra s'appuyer sur des techniques formelles permettant la modélisation de la concurrence et sa prise en compte dans la génération de tests répartis (voir 3.4).

Le futur projet SYCOMORE (B. Caillaud)[2] s'intéresse aux spécifications partielles et en particulier à la synthèse de contrôleurs répartis. Leurs travaux sur les modèles du parallélismes et les algorithmes de synthèse seront donc utiles pour la problématique de la génération de tests ou la synthèse de contrôleurs répartis. D'autre part, leurs travaux, en particulier ceux communs avec le futur projet ESPRESSO (J.-P. Talpin) sur BDL [CTJ+00], va les amener à considérer des spécifications manipulant des données qu'ils devront aussi manipuler de manière symbolique. Ils seront donc potentiellement intéressés par nos travaux sur ce thème.

Un problème qui pourrait unir les efforts de VERTECS, TRISKELL et SYCOMORE est celui de la relation qui existe entre les contrats (pré-post conditions dans UML) et les pré-ordres de test. Le problème est celui de la substituabilité d'une classe par une sous-classe pour que celle-ci satisfasse les mêmes contrats. Pour résoudre le problème, il faut définir formellement ce qu'est la substituabilité souhaitée en fonction du niveau d'observation, ce qui ressemble fort à un pré-ordre de test ou une relation de conformité. Il faut ensuite vérifier statiquement que les sous-classes satisfont cette relation pour pouvoir les substituer.

Le futur projet ESPRESSO s'intéresse à l'ingénierie des systèmes embarqués en promouvant l'approche synchrone. Dans ce cadre, ESPRESSO s'intéresse en particulier à l'analyse de programmes synchrones par vérification et interprétation abstraite. La synthèse de contrôleurs et la génération de tests sont des approches complémentaires à cette analyse. Les langages SIGNAL et BDL et leurs outils constituent pour nous des point d'entrée pour nos techniques. L'outil SIGALI en particulier (voir section 5, intégré à l'environnement SIGNAL et conçu par H. Marchand, permet la vérification et la synthèse de contrôleurs pour des programmes SIGNAL. Nous avons aussi déjà évoqué plus haut l'utilité de techniques symboliques pour BDL. Enfin, les travaux sur l'interprétation abstraite menés dans ESPRESSO sont intéressant pour la synthèse de contrôleurs et la génération de tests.

D'autres projets du programme 1C ont aussi des thèmes de recherche proches et complémentaires des nôtres.

Nous avons derrière nous une collaboration durable avec le projet Vasy de l'Inria Grenoble. D'abord en termes d'outils, puisque TGV utilise la boîte à outils CADP. Nous avons collaboré jusqu'à cette année sur le test et la vérification de spécifications Lotos de protocoles de cohérence de cache dans le cadre de l'action Vasy-FormalFame du GIE Bull-Inria. Nous avons entamé une action FormalCard dans ce même cadre sur la vérification et le test d'applications de cartes à puces avec CP8 (ex-Bull, maintenant Schlumberger). Nous avons aussi collaboré avec le projet Vasy dans l'action coopérative Verdon sur la prise en compte des données dans la vérification et le test. Notre collaboration avec le projet Vasy va donc continuer, en particulier à travers l'action FormalCard.

Nous avons aussi eu des relations avec l'ex-projet Meije de Sophia dans le cadre de l'AC Verdon et sur les langages synchrones dans le projet SYRF. Certains thèmes de recherche du projet Tick

2. Le nom de ce projet est susceptible d'être modifié

[CTJ+00] Benoit Caillaud, Jean-Pierre Talpin, Jean-Marc Jézéquel, Albert Benveniste, and Claude Jard. BDL: A semantics backbone for UML dynamic diagrams. Technical Report RR-4003, Irisa/INRIA Rennes, September 2000. http://www.inria.fr/RRRT/RR-4003.html.

qui en émane sont voisins des nôtres. En particulier les traitements symboliques pour la vérification de spécification dans les langages synchrones, la programmation en Esterel de contrôleurs.

Nous avons des relations avec l'action TRIO de Nancy à travers l'action de développement AEE (Architectures Electroniques Embarquées). Bien qu'orientées plus résolument que nous vers le temps-réel, leur recherche a un point commun autour du test. Leurs travaux sur la génération de tests à partir d'automates temporisés sont en effet très proches des nôtres.

Nous ne nous sommes pas encore penchés réellement sur la problématique du test de code mobile, mais c'est un aspect qui est potentiellement riche et d'avenir. Les travaux des projets Moscova de Rocquencourt et Mimosa de Sophia sur la vérification de ce type de codes sont donc d'éventuelles bases pour le test. De plus, nous sommes aussi clients des travaux sur l'interprétation abstraite de programmes comme ceux menés dans Moscova.

Des complémentarités existent aussi avec des projets d'autres programmes.

Nous avons entamé une collaboration avec le projet Armor et la société Alitec (Laval) sur la combinaison de modèles Markoviens et de systèmes de transitions pour la génération de tests, en particulier pour des protocoles de l'UMTS. Les premiers résultats ont été encourageants. Il faudrait certainement aller plus loin dans cette voie en puisant aussi dans les travaux sur la vérification probabiliste. Armor ayant pour domaine d'application les télécoms, il fait parfois face à des problèmes de test de protocoles qui pourraient être traités formellement par nos techniques et outils. L'arrivée dans Armor de César Viho, issu de Pampa, devrait contribuer à établir cette coopération.

Quelques autres projets des programmes 1B et 1C conçoivent ou expérimentent des protocoles ou applications réparties (Hipercom, Mascotte, Planète, Résédas, Sirac, Solidor, Apache) et pourraient être potentiellement intéressés par nos techniques de génération de tests de protocoles.

Nous sommes aussi proches d'axes de recherche de projets du thème 2. En particulier, nous sommes clients de techniques de preuve de programme, de résolution de contraintes, axes de recherche se retrouvant dans les projets Contraintes, Logical, Certilab et Prothéo, de techniques d'analyse statique (Lande par exemple). Lande est d'ailleurs un des projets les plus proches du nôtre par ses activités sur le test (outil Casting), la vérification, l'analyse statique et le débogage. Nous collaborons d'ailleurs dans le projet RNTL COTE et envisageons l'utilisation conjointe de Casting et TGV. A plus long terme, une connexion entre STG et Casting pourrait être envisagée.

Nous collaborons avec le projet Aïda sur l'aspect diagnostic et supervision de réseaux. Le problème général abordé par Aïda est celui de l'observation, alors que pour nous l'observation se combine avec le contrôle dans le cadre du test. Bien que les travaux d'Aïda soient issus de l'intelligence artificielle, certains modèles et certaines techniques qu'ils utilisent pour la construction de diagnostiqueurs sont très proches de ceux que nous utilisons pour la génération de tests et la synthèse de contrôleurs. L'aspect réparti et la manipulation symbolique pour la construction de diagnostiqueurs commence aussi à être abordée dans ce projet.

Nous avons des relations avec le projet BIP de INRIA Rhône-Alpes (Thème 4a). Notre relation porte sur l'utilisation concrète des techniques de synthèse de contrôleurs à événements discrets dans le cadre de la programmation de systèmes robotiques ou de contrôle-commande. Plus spécifiquement, nous étudions comment dériver des objectifs de contrôle un "langage" de spécification et/ou de séquencement de tâches robotiques. Dans une autre approche, on s'intéresse à des objectifs plus "internes", où on étudie des structures de tâches comportant plusieurs modes d'exécution. On veut les utiliser pour définir des missions de contrôle-commande, et utiliser la synthèse de contrôleur pour gérer le passage entre ces modes. D'autres aspects portants sur l'utilisation (et plus particulièrement sur l'exécution) du contrôleur obtenu sont également à l'étude.

Le projet ADP s'intéresse aux systèmes répartis et à l'algorithmique répartie. Or, dans le cadre du test réparti, nous sommes face à un système réparti et il est logique d'y retrouver des problématiques chères au projet ADP comme les problèmes de causalité et de synchronisation. La distribution de tests à partir de tests séquentiels nous a déjà amené à utiliser un service de consensus afin de résoudre le problème du choix d'actions non-locales. Dans le cadre plus restreint du test séquentiel mais asynchrone nous avons également utilisé un mécanisme d'estampillage des événements de l'implémentation proche des horloges logiques et permettant de reconstituer l'ordre des événements de l'implémentation. Le problème est voisin mais plus complexe que celui de l'observation répartie sur lequel les projets Pampa et ADP avaient déjà collaboré dans le passé. La combinaison des problèmes de sûreté de fonctionnement menés dans ADP et de génération de test pourrait être aussi étudiée, en particulier dans le cadre de l'AEE (voir 8).

Autres laboratoires de recherche français

Nous avons à notre actif une collaboration durable avec le laboratoire Verimag de Grenoble. Depuis une dizaine d'années, nous avons collaboré d'abord sur la vérification puis sur la génération de tests. Nous avons participé ensemble au transfert de technologie de vérification dans l'outil Véda de Vérilog, ancêtre d'ObjectGéode qui en reprend les principes. Nous avons développé conjointement le prototype TGV et participé à son transfert dans l'outil ObjectGéode. Cette collaboration continue dans le projet européen IST Agedis sur le test de logiciel réparti et le test symbolique. Nous devrions aussi développer nos collaborations sur la synthèse de contrôleurs et le test de programmes Lustre.

Nous avons des liens avec le LSR/IMAG à travers le projet RNTL COTE sur le test de composants objets. Le LSR développe aussi un outil génération de tests de programmes Lustre dont certains aspects ne sont pas très éloignés de TGV. Depuis peu, le LSR s'intéresse également à la combinaison des techniques de synthèse de contrôleurs et de génération de tests. L'accueil en Post-Doc de Lydie du Bousquet en 2000 dans Pampa et son retour au LSR nous permettent de consolider ces relations autour du test.

Nous avons entretenu des relations avec plusieurs laboratoires français (Vérimag, le LaBri à Bordeaux, le LSV à Cachan, le LAAS à Toulouse, le LRI à Orsay, le CNET) autour de la vérification et du test de systèmes critiques pendant l'action Forma entre 97 et 99. Cette action est maintenant terminée mais les liens tissés sont une base solide pour de futures collaborations.

Nous participons depuis 99 au groupe de travail Test et Objets piloté par le LRI, le LAAS et l'Irisa. Ce groupe de travail et d'échange nous permet de rencontrer des acteurs académiques (LRI, LAAS, INT Evry, LSR, Université d'Evry, Onera-CERT, LIENS, UBS Vannes) et industriels (France Télécom R & D, Gemplus, Thomson-CSF)

Nous avons des relations avec le groupe d'Ahmed Bouajjani au LIAFA (Paris 7) qui s'intéresse aux aspects symboliques pour la vérification et développent l'outil d'analyse de systèmes infinis symboliques TReX qui permet entre autres le calcul d'invariants. Leurs techniques sont potentiellement intéressante pour le test et le contrôle symboliques.

Communauté internationale

Nous avons des relations privilégiées avec le laboratoire de recherche SRI (Stanford Research Institute) aux USA, en particulier autour du système de preuve PVS. Vlad Rusu y a effectué son post-doc et a fait plusieurs séjours de longue durée depuis qui ont donné lieu à plusieurs publications [40, 15]. Thierry Jéron y a également effectué un court séjour. Nous sommes principalement utilisateurs de PVS mais les travaux que nous effectuons sur le test, en particulier symbolique intéresse beaucoup le SRI. Une document de collaboration officiel entre le SRI et l'Irisa a été signé par le CNRS.

Nous participons depuis 1999 avec le groupe de Ed Brinksma de l'université de Twente (Hollande) au programme d'action intégré Van Gogh. Notre collaboration se situe dans le domaine de la génération de tests. Plusieurs visites à Twente et à Rennes ont eu lieu ainsi qu'une publication commune sur l'application de TGV à une étude de cas [14]. Cette collaboration nous a permis, grâce à une invitation à une Journée Test chez Phillips (Eindhoven), de rentrer en contact avec la communauté universitaire et industrielle du test en Hollande.

Nous entretenons des relations suivies avec le groupe d'Alexandre Petrenko au CRIM (Montréal), groupe qui travaille aussi sur la génération de tests.

Nous avons également des interactions sur la synthèse de contrôleurs (et plus précisément sur la commande optimale) avec le groupe de "Discrete Event Systems" de Stéphane Lafortune à l'université du Michigan au USA, où Hervé Marchand a effectué son Post-Doc. Des visites ont eu lieu dans les deux sens ainsi que des publications communes. Cette collaboration pourrait s'étendre à des thématiques plus spécifiques telles que les techniques symboliques et l'utilisation conjointe des techniques de diagnostic et de synthèse de contrôleurs.

8 Relations industrielles

Agedis (fin 2000-2003) est un projet européen IST. Ce projet vise à fournir des méthodes et outils pour l'automatisation du test de logiciels, et en particulier pour les logiciels répartis, principalement du domaine télécom. TGV est vu comme base à la réalisation de l'outil de génération de tests qui sera fourni par le projet. Nous y collaborons avec IBM Haifa (Israël) et IBM Hursley (G.-B.), France Télécom R& D (Lannion), IntraSoft (Grèce), Imbus (Allemagne), l'université d'Oxford et Vérimag.

Le projet RNTL Cote (fin 2000-2002) est un projet pré-compétitif dont le but est fournir des méthodes et des outils facilitant le test, la vérification et la certification de composants logiciels. Nous participons à ce projet aux cotés de France Télécom, Gemplus, Softeam et le LSR/IMAG et les projets de l'Irisa TRISKELL et Lande. Là aussi, TGV occupe une place centrale pour la réalisation d'un outil de génération de tests pour UML.

L'AEE (Architectures Electroniques Embarquées) (1999-2001) est une Action de Développement soutenue par le ministère de l'industrie. Son objectif est de concevoir et valider un processus rapide et sûr pour la définition de l'Architecture Système et le développement des logiciels associés, embarqués Pour notre part, nous y étudions l'apport du test en relation avec la sûreté de fonctionnement et les modes de fonctionnement. Cette action nous permet d'avoir des relations avec plusieurs constructeurs automobiles (PSA, Renault) et équipementiers automobile (Siemens, Sagem, Valéo) et EADS (Aérospatiale).

FormalCard (2001-?) est une action démarrée dans le cadre du GIE Bull-Inria Dyade avec CP8 et prolongée suite au rachat de CP8 par Schlumberger. Nous y étudions le test de logiciel de cartes à puces sur lequel nous mettons en avant notre travail sur le test symbolique. C'est dans ce cadre que nous avons élaboré une étude de cas d'un porte monnaie électronique. Nous collaborons avec le projet Vasy (Inria Rhônes-Alpes) qui fournit le langage de spécification.

Autres contacts

Nos relations avec France Télécom R& D à Lannion (groupe de Rolland Groz) datent d'une bonne dizaine d'année. Celles-ci sont maintenant principalement autour du test, en particulier à travers le projet RNTL COTE et le projet IST Agedis ainsi que le groupe de travail Test et Objets.

Nous avions des relations suivies avec Vérilog et celles-ci se poursuivent avec Telelogic qui l'a rachetée. Plusieurs contrats nous ont unis, dont deux concernant le transfert de technologie, l'un ancien sur les techniques de vérification dans l'outil Veda, l'autre plus récent sur la génération de tests dans ObjectGéode. De plus Telelogic nous prête gracieusement ses outils à des fins de recherche et a identifié Pampa comme partenaire universitaire privilégié.

Nous avons des contacts avec la société Valiosys qui travaille sur l'outil de vérification LPV basé sur la programmation linéaire. Valiosys s'intéresse aussi à l'analyse statique et l'abstraction pour la vérification, ce qui se rapproche de nos préoccupations pour le test. Nous étudions en particulier la possibilité de réaliser une plate-forme intégrant leur outil LPV et Invest (Verimag), TReX (LIAFA) et STG.

VERTECS et TRISKELL ont de nouveaux contacts avec le centre de recherche d'Alcatel sur le test d'un modèle UML générique de contrôle d'appel.

9 Animation de la communauté scientifique (2000-2001)

T. Jéron fait partie de la commission de spécialistes de l'ENS Cachan.
T. Jéron est élu au conseil de laboratoire de l'Irisa et a été responsable du séminaire de l'IRISA de 97 à 2000.

T. Jéron a fait partie du comité de programme de LFM2000 (Fifth Nasa Langley Formal Methods Workshop, juin 2000, Williamsburg, Virginie, USA) et fera partie de celui de FATES'01 (Formal Approaches to Testing of Software, Workshop satellite de CONCUR'01 août 2001, Aalborg, Denmark).

H. Marchand est membre du comité de programme du symposium SCODES'2001 affilié à CAV.

T. Jéron a été invité à donner des séminaires à l'Université Juan Carlos de Madrid en mars 2000, dans le cadre de la revue finale du projet LTR Vires à Autrans en juin 2000. Il a fait un tutoriel dans le cadre de l'Ecole CEA-EDF-INRIA *Méthodes Formelles : Spécifications et vérification de systèmes dynamiques* en juillet 2000.

Hervé Marchand a donné des séminaires invités à l'université du Michigan lors de ses différentes visites, à l'Inria Rhône-Alpes dans le cadre de l'action Tolère, et à Movep'00 pour une session spéciale de l'action Mars.

V. Rusu a donné en 2000-2001 des séminaires invités à l'université de Twente, au SRI, à la NASA, à Cadence Berkeley, à Verimag et au LIAFA.

Les membres du projet, sensibles à l'importance de l'enseignement dans la promotion des méthodes formelles, participent à l'enseignement, principalement en troisième cycle:

Ifsic Université de Rennes 1: Dans la filière génie logiciel du DEA d'informatique à l'IFSIC, T. Jéron enseigne la calculabilité et la complexité, V. Rusu donne des conférences sur le prouveur PVS. En 1999-2000, H. Marchand a enseigné la synthèse de contrôleurs par des techniques symboliques basées sur l'utilisation des BDD et son intégration dans l'environnement Signal développé par l'équipe EP-ATR.

En 3ème année de DIIC, T. Jéron est responsable d'un module sur la validation de protocoles et enseigne le model-checking et le test. Ce cours commun avec C. Jard a donné lieu à publication [4].

Enst Rennes et Brest: T. Jéron enseigne le model-checking en dernière année de l'Enst à Rennes et donne un cours sur le test en deuxième année à Brest.

Bibliographie du projet depuis 1998

Thèses et habilitations à diriger des recherches

[1] P. Morel, *Une algorithmique efficace pour la génération automatique de tests de conformité*, thèse de doctorat, Université de Rennes I, France, février 2000.

Articles

[2] H. Ben-Abdallah, J. Y. Choi, D. Clarke, Y. S. Kim, I. Lee, « A Process Algebraic Approach to the Schedulability Analysis of Real-Time Systems », *Real-Time Systems 15*, 3, November 1998, p. 189–219.

[3] 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 », *Journal of Science of Computer Programming, special issue on Formal Methods in Industry 36*, 1, janvier 2000, p. 27–52.

[4] C. Jard, T. Jéron, « An educational case study in protocol verification and distributed observation », *Journal of Computer Science Education, ECASP Special Issue 10*, 3, 2000.

[5] T. Jéron, C. Jard, C. Viho, B. Caillaud, H. Kahlouche, P. Morel, J.-C. Fernandez, A. Kerbrat, M. Bozga, « Génération automatique de tests pour les protocoles: l'exemple de l'approche formelle de TGV », *Revue de l'Electricité et de l'Eléctronique (REE) 3*, mars 1999.

[6] Y. Le Traon, T. Jéron, J.-M. Jézéquel, P. Morel, « Efficient OO Integration and Regression Testing », *IEEE Transactions on Reliability 49*, 1, mars 2000, p. 12–25.

[7] E. Marchand, E. Rutten, H. Marchand, F. Chaumette, « Specifying and verifying active vision-based robotic systems with the Signal environment », *Int. Journal of Robotics Research 17*, 4, April 1998, p. 418–432.

[8] H. Marchand, O. Boivineau, L. S., « On the Synthesis of Optimal Schedulers in Discrete Event Control Problems with Multiple Goals », *SIAM Journal on Control and Optimization 39*, 2, 2000, p. 512–532.

[9] H. Marchand, P. Bournai, M. Le Borgne, P. Le Guernic, « Synthesis of Discrete-Event Controllers based on the Signal Environment », *Discrete Event Dynamic System : Theory and Applications 10*, 4, October 2000, p. 347–368.

[10] H. Marchand, E. Rutten, L. M., M. Samaan, « Formal Verification of Programs specified with Signal: Application to a power transformer Station Controller », *Science of Computer Programming*, 2001, To appear.

[11] H. Marchand, M. Samaan, « Incremental Design of a Power Transformer Station Controller using Controller Synthesis Methodology », *IEEE Transaction on Software Engineering 26*, 8, August 2000, p. 729–741.

[12] S. Ramangalahy, P. Le Gall, T. Jéron, « Une application de la théorie des jeux au test de conformité », *Revue Electronique sur les Réseaux et l'Informatique Répartie (RERIR) 9*, mai 2000, p. 3–23.

[13] O. Roux, V. Rusu, F. Cassez, « Hybrid verifications of reactive programs », *Formal Aspects of Computing 11*, 4, 1999, p. 448–471.

Communications à des manifestations scientifiques

[14] A. Belinfante, L. Du Bousquet, S. Ramangalahy, S. Simon, C. Viho, R. De Vries, « Formal test automation: the conference protocol with TGV/TorX », *in : IFIP TC6/WG6.1 13th International*

Conference on Testing of Communicating Systems, TestCom 2000, Ottawa, Ontario, Canada, Kluwer Academic, p. 221–228, août 2000.

[15] S. Bensalem, V. Ganesh, Y. Lakhnech, C. Muñoz, S. Owre, H. Ruess, J. Rushby, V. Rusu, H. Saïdi, N. Shankar, E. Singerman, A. Tiwari, « An Overview of SAL », *in: LFM 2000: Fifth NASA Langley Formal Methods Workshop*, C. M. Holloway (réd.), NASA Langley Research Center, p. 187–196, Hampton, VA, juin 2000. Proceedings available at http://shemesh.larc.nasa.gov/fm/Lfm2000/Proc/.

[16] D. Clarke, T. Jéron, V. Rusu, E. Zinovieva, « Automated Test and Oracle Generation for Smart-Card Applications », *in: e-Smart 2001, International Conference on Research in Smart Cards*, 2001. to appear in LNCS.

[17] L. Du Bousquet, H. Martin, « Automatic test generation for Java-Card applets », *in: 4th Workshop on Tools for System Design and Verification*, juillet 2000.

[18] R. Groz, T. Jéron, A. Kerbrat, « Automated Test Generation from SDL specifications », *in: SDL'99 The Next Millenium, 9th SDL Forum, Montréal, Québec*, R. Dssouli, G. von Bochmann, Y. Lahav (réd.), Elsevier, p. 135–152, juin 1999.

[19] T. Henzinger, V. Rusu, « Reachability verification for hybrid automata », *in: International Workshop on Hybrid Systems: Computation and Control, LNCS 1386*, Springer Verlag, p. 190–204, 1998.

[20] C. Jard, T. Jéron, H. Kahlouche, C. Viho, « Towards automatic distribution of testers for distributed conformance testing », *in: FORTE/PSTV'98, Paris, France*, Chapman & Hall, novembre 1998.

[21] C. Jard, T. Jéron, P. Morel, « Verification of Test Suites », *in: TestCom 2000, IFIP TC 6 / WG 6.1, The IFIP 13th International Conference on Testing of Communicating Systems, Ottawa, Ontario, Canada*, Kluwer Academic Publishers, août 2000.

[22] C. Jard, T. Jéron, L. Tanguy, C. Viho, « Remote testing can be as powerful as local testing », *in: Formal methods for protocol engineering and distributed systems, FORTE XII/ PSTV XIX' 99, Beijing, China*, J. Wu, S. Chanson, Q. Gao (réd.), Kluwer Academic Publishers, p. 25–40, octobre 1999.

[23] C. Jard, T. Jéron, « Verification and distributed observation of the alternating bit protocol », *in: FORTE/PSTV'98, ECASP: Special Session on Educational Case Studies in Protocols*, Paris, France, novembre 1998.

[24] C. Jard, T. Jéron, H. Kahlouche, C. Viho, « Towards Automatic Distribution of Testers for Distributed Conformance Testing », *in: IFIP TC6/WG6.1 Joint International Conference FORTE/PSTV'98, Formal Description Techniques (FORTE XI), Protocol Specification, Testing, and Verification (PSTV XVIII)*, Paris, November 1998.

[25] T. Jéron, J.-M. Jézéquel, A. Le Guennec, « Validation and Test Generation for Object-Oriented Distributed Software », *in: IEEE Proc. Parallel and Distributed Software Engineering, PDSE'98, Kyoto, Japan*, avril 1998.

[26] T. Jéron, J.-M. Jézéquel, Y. Le Traon, P. Morel, « Efficient Strategies for Integration and Regression Testing of OO Systems », *in: 10th IEEE International Symposium on Software Reliability Engineering, ISSRE'99, Boca Raton, Florida*, p. 260–269, novembre 1999.

[27] T. Jéron, P. Morel, « Test generation derived from model-checking », *in: CAV'99, Trento, Italy*, N. Halbwachs, D. Peled (réd.), *LNCS*, *1633*, Springer-Verlag, p. 108–122, juillet 1999.

[28] H. Kahlouche, C. Viho, M. Zendri, « An Industrial Experiment in Automatic Generation of Executable Test Suites for a Cache Coherency Protocol », *in: IFIP TC6 11^{th} International Workshop on Testing of Communicating Systems*, A. Petrenko, N. Yevtushenko (réd.), Chapman & Hall, September 1998.

[29] H. Kahlouche, C. Viho, M. Zendri, « Hardware Testing using a Communication Protocol Conformance Testing Tool », *in: Tools and Algorithms for the Construction and Analysis of Systems (TACAS'99)*, W. R. Cleaveland (réd.), *Lecture Notes in Computer Science*, *1579*, Springer Verlag, p. 315–329, March 1999.

[30] H. Marchand, O. Boivineau, L. S., « On the Synthesis of Optimal Schedulers in Discrete Event Control Problems with Multiple Goals », *in: 1998 IEEE International Conf. On Systems, Man, And Cybernetics*, p. 734–739, San Diego, California, USA, October 1998.

[31] H. Marchand, P. Bournai, M. Le Borgne, P. Le Guernic, « A Design Environment for Discrete-Event Controllers based on the Signal Language », *in: 1998 IEEE International Conf. On Systems, Man, And Cybernetics*, p. 770–775, San Diego, California, USA, October 1998.

[32] H. Marchand, M. Le Borgne, « On the Optimal Control of Polynomial Dynamical Systems over Z/pZ », *in: 4th International Workshop on Discrete Event Systems*, p. 385–390, Cagliari, Italy, August 1998.

[33] H. Marchand, M. Le Borgne, « Partial Order Control of Discrete Event Systems modeled as Polynomial Dynamical Systems », *in: 1998 IEEE International Conference On Control Applications*, Trieste, Italia, Septembrer 1998.

[34] H. Marchand, S. Pinchinat, « Supervisory Control Problem using Symbolic Bisimulation Techniques », *in: 2000 American Control Conference*, p. 4067–4071, Chicago, Illinois, USA, June 2000.

[35] H. Marchand, M. Samaan, « Incremental Design of a Power Transformer Station Controller using Controller Synthesis Methodology », *in: World Congress on Formal Methods (FM'99)*, *Lecture Notes in Computer Science*, *1709*, Springer Verlag, p. 1605–1624, Toulouse, France, September 1999.

[36] H. Marchand, « Supervisory Control of hierarchical Finite states Machines », *in: Workshop of Synthesis of Concurrent Systems.*, Newcastle upon Tyne, June 2001.

[37] S. Pinchinat, H. Marchand, « Symbolic Abstractions of Automata », *in: Proc of 5th Workshop on Discrete Event Systems, WODES 2000*, p. 39–48, Ghent, Belgium, August 2000.

[38] S. Ramangalahy, P. Le Gall, T. Jéron, « Une application de la théorie des jeux au test de conformité », *in: Colloque Francophone sur l'Ingénierie des Protocoles, CFIP 99, Nancy, France*, A. Schaff (réd.), Hermès, avril 1999.

[39] V. Rusu, L. du Bousquet, T. Jéron, « An Approach to Symbolic Test Generation », *in: Integrated Formal Methods (IFM'00), Dagstuhl, Allemagne*, *LNCS*, Springer Verlag, Novembre 2000.

[40] V. Rusu, E. Singerman, « On proving safety properties by integrating static analysis, theorem proving and abstraction », *in: Tools and Algorithms for the Construction and Analysis of Systems (TACAS'99)*, *LNCS*, *1579*, p. 178–192, 1999. An improved version appeared as INRIA research report 3726, available at http://www.inria.fr/rrrt/rr-3726.html.

Rapports de recherche

[41] H. Marchand, O. Boivineau, L. S., « On the Synthesis of Optimal Schedulers in Discrete Event Control Problems with Multiple Goals », Research Report n ° CGR-98-10, Control Group, College of Engineering, Univeristy of Michigan, USA, July 1998.

[42] H. Marchand, O. Boivineau, L. S., « Optimal control of discrete event systems under partial observation », Research Report n ° CGR-00-10, Control Group, College of Engineering, Univeristy of Michigan, USA, September 2000.

[43] H. Marchand, O. Boivineau, L. S., « Optimal control of discrete event systems under partial observation », Research Report n ° 1359, IRISA, November 2000.

[44] H. Marchand, M. Le Borgne, « The Supervisory Control Problem of Discrete Event Systems using polynomial Methods », Research Report n ° 1271, Irisa, October 1999.

[45] S. Pinchinat, H. Marchand, M. Le Borgne, « Symbolic Abstractions of Automata and their application to the Supervisory Control Problem », Research Report n ° 1279, Irisa, November 1999.

[46] V. Rusu, E. Singerman, « Interactive abstractions: proving safety property by integrating static analysis, theorem proving, and abstraction », rapport de recherche n ° 1256, IRISA, 2000, Also submitted to the *Formal Methods in Systems Design* journal.

[47] V. Rusu, « Analyzing Automata with Presburger Arithmetic and Uninterpreted Function Symbols », rapport de recherche n ° 4100, INRIA, 2001, Available at http://www.inria.fr/rrrt/rr-4100.html.

Divers

[48] A. Benveniste, P. Bournai, T. Gautier, M. Le Borgne, P. Le Guernic, H. Marchand, « The Signal declarative synchronous language : controller synthesis & systems/architecture design. », Submitted to the 40th IEEE Conference on Decision and Control, 2001.

[49] D. Clarke, T. Jéron, V. Rusu, E. Zinovieva, « STG: A Tool for Generating Symbolic Test Programs and Oracles from Operational Specifications », Submitted to 8th European Software Engineering Conference (ESEC), 2001.

[50] « Etude de cas SSCOP », Action Forma, Maîtrise des systèmes réactifs et sûrs, Programme DSP-STTC/CNRS/MENRT, Bilan de la première année, Opération 1, janvier 1998.

[51] C. Jard, T. Jéron, J.-C. Fernandez, « Etat de l'art, outils existants et spécification de l'outil », Marché France-Télécom - Vérilog - Irisa - Verimag, Génération automatique de tests, août 1998.

[52] C. Jard, T. Jéron, « Test Generation for Reactive Systems », FM'99: World Congress on Formal Methods in the development of computing systems, Toulouse, France, Industrial Tutorial: Testing & Formal Methods, septembre 1999.

[53] T. Jéron, H. Kahlouche, P. Morel, « Document de spécification des algorithmes de TGV », Marché France-Télécom - Vérilog - Irisa - Verimag, Génération automatique de tests, 1999.

[54] T. Jéron, « TGV: principles, tool architecture and algorithms », Dagstuhl Seminar on Test Automation for Reactive Systems - Theory and Practice, septembre 1998.

[55] T. Jéron, « Efficient automatic generation of conformance test suites for protocols and reactive systems », Workshop on Formal Methods for Safety Critical Systems, Munich, Germany, mars 1999.

[56] T. Jéron, « Testing Reactive Systems using Model-Based Verification Techniques », Invited presentation, Dutch Testing Day, Philips, Eindhoven, novembre 1999.

[57] H. Marchand, O. Boivineau, S. Lafortune, « Optimal control of discrete event systems under partial observation », Submitted to the Journal of Automatica, 2001.

[58] H. Marchand, O. Boivineau, S. Lafortune, « Optimal control of discrete event systems under partial observation », Submitted to the 40th IEEE Conference on Decision and Control, 2001.

[59] V. Rusu, E. Zinovieva, « Analyzing Automata with Presburger Arithmetic and Uninterpreted Function Symbols », Submitted to the Workshop on Verification of Parameterized Systems (VEPAS'01), 2001.

[60] V. Rusu, « Verifying a Sliding-Window Protocol using PVS », Submitted to Forte'01, 2001.

[61] V. Rusu, « Verifying that Invariants are Context-Inductive », Submitted to Computer Science Logic, 2001.