Génération de tests à partir de critères dynamiques de sélection et par abstraction

Fabrice Bouquet Pierre-Christophe Bué Jacques Julliand Pierre-Alain Masson
LIFC, Université de Franche-Comté
16 Route de Gray F-25030 Besançon Cedex France
{bouquet, bue, julliand, masson}@lifc.univ-fcomte.fr

Résumé

Cet article présente une méthode de génération assistée de tests. Elle applique des critères dynamiques de sélection des tests (TP) sur un modèle formel comportemental (M) utilisé auparavant, par exemple par LTG, pour générer des tests fonctionnels à partir de critères statiques de sélection. On peut appliquer à M un critère dynamique de sélection TP mais ceci nécessite de représenter M par un automate. Pour des applications réelles, sa taille en nombre d'états et de transitions est beaucoup trop grande (voir infinie) pour être utilisable. Nous proposons une méthode pour extraire une abstraction de M à partir d'un objectif de test TP. Nous effectuons un produit synchronisé de cette abstraction avec TP afin de cibler les exécutions du système sous test qui satisfont TP. Puis nous générons des tests abstraits symboliques à partir de ce modèle réduit en appliquant les critères de couverture tous les états ou toutes les transitions. Cet ensemble de tests est valué à partir de M, concrétisé puis exécuté sur l'implémentation sous test. Cette méthode est proposée pour compléter la méthode BZ-TT de génération de tests à partir de critères statiques de sélection. L'utilisateur obtient des tests complémentaires en fournissant un critère dynamique de sélection. La méthode réutilise M, la couche de concrétisation des tests et l'infrastructure d'exécution des tests. L'originalité de l'approche est de construire une abstraction du modèle issue automatiquement de l'analyse statique d'un objectif de test formalisant des besoins de test d'une propriété dynamique du système.
Mots-clés : Model-Based Testing, abstraction, objectif de test, génération de tests symboliques

1 Introduction

La génération de tests à partir de modèles [UL06, BJK+05], appelée MBT (Model-Based Testing), est une démarche dont l'utilisation par des équipes de validation de logiciels se généralise. Elle permet d'augmenter la productivité de la phase de test. En particulier, dans un contexte d'externalisation du développement logiciel où une modélisation, fournie à l'équipe de développement comme élément du cahier des charges pour servir de référence pour la phase de recette, est réutilisée pour engendrer les tests. Elle permet également la systématisation de la production des tests qui garantit une meilleure qualité des jeux de tests évaluée par des critères de couverture parfaitement définis. Cette tendance est concrétisée par le développement d'environnements de génération de tests commercialisés comme LTG [JL07] et LTD [BBC+06], SpecExplorer [BLS05], Réactis [Rea08], etc. et d'environnements de génération de tests développés dans le cadre d'activités de recherche comme TGV [JJ05] et STG [JJRZ05], AGATHA [GGL04], BZ-TT [ABC+02].

La démarche MBT repose d'une part sur la définition d'un modèle comportemental (noté M) donnant l'oracle et d'autre part sur le pilotage de l'ingénieur validation. Celui-ci est essentiellement effectué par le choix de *critères de sélection* des tests. On peut distinguer deux catégories principales de critères de sélection, les critères statiques et les critères dynamiques. Les critères statiques sont des critères structurels sur le modèle comme les états et les transitions pour les modèles état-transition. Pour les

modèles pré-post en B, Z ou OCL et également les modèles de type EFSM (Extended Finite State Machine) les critères sont les décisions, les conditions, ou les valeurs des variables d'état. Les critères dynamiques sont des critères sur les enchaînements d'actions ou d'états du modèle. Plusieurs approches ont été explorées pour exprimer ce type de critère. Dans [ABM98] et [HLSU02], le critère dynamique est un enchaînement d'états exprimé par des propriétés en logique temporelle PLTL. Dans [CIVDP07], [JJRZ05], c'est un enchaînement d'actions exprimé par un IOLTS (Input Output Labelled Transition System). Dans [LdBMB04], c'est un enchaînement d'actions exprimé par des expressions régulières. Dans cet article, nous proposons de décrire des critères dynamiques, appelés *objectifs de tests* dénotés par TP (*Test Purpose*), par une expression régulière enchaînant états et actions. Cette expression régulière a pour sémantique un automate dont les états sont décorés par des propriétés d'état, et les transitions étiquetées par des noms d'actions.

La qualité d'une suite de tests engendrée à partir d'un critère dynamique de sélection peut être mesurée par un critère statique de couverture sur l'automate des enchaînements (automate de Büchi associé aux propriétés PLTL, IOLTS ou automate associé aux expressions régulières) tels que le nombre d'états, le nombre de transitions et le nombre de chemins d'exécution couverts. Les critères de couverture sur les chemins sont les plus forts dans une hiérarchie de critères (voir dans [UL06] les critères tels que *all-one-loop-paths*, *all-round-trips*, *all-loop-free-paths*, etc.). Mais la génération de tests en utilisant ces critères est fortement combinatoire et le problème de trouver des techniques utilisables en pratique sur des cas d'étude réelles est loin d'être résolu.

Dans la section 2 nous présentons les spécificités de l'approche MBT à partir de critères dynamiques de sélection et par abstraction relativement à l'approche MBT sans abstraction. Dans la section 3, nous donnons les définitions des notions de modèle comportemental, de critère dynamique de sélection, d'abstraction et de test abstrait symbolique. Dans la section 4, nous définissons la méthode de génération de l'abstraction et sa synchronisation avec un TP. Dans la section 5 nous présentons des résultats d'une expérimentation effectuée sur trois études de cas et quatre TP. Enfin en section 6 nous concluons, nous donnons quelques perspectives et nous comparons notre approche à d'autres approches de génération de tests à partir de critères dynamiques de sélection.

2 Génération de tests à partir d'abstraction

Dans cette section, nous présentons notre démarche de génération de tests à partir de critères dynamiques de sélection et d'abstractions. Nous commençons par un rappel sur les travaux concernant la définition de propriétés qui servent pour la définition de critères dynamiques de sélection.

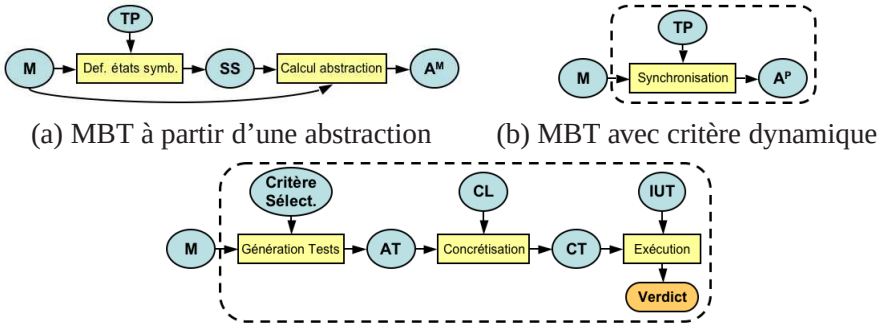
2.1 Problématique et motivations

Dans [JMT08a], nous avons proposé un langage basé sur les expressions régulières, permettant de décrire des TP comme des séquences d'actions à activer et d'états à atteindre, ciblés par ces actions. Dans [JMT08b], nous avons présenté une démarche de génération de tests à partir d'un TP et d'un modèle B, où chaque action est décrite par une opération. Cette démarche consiste à déplier tous les chemins de l'automate associé au TP. Chaque chemin est un test abstrait symbolique. Il est abstrait car il est décrit au niveau d'abstraction du modèle et devra être concrétisé pour être exécuté sur l'implémentation à tester. Il est symbolique car les valeurs des paramètres des opérations ne sont pas définies. Elles sont définies par une étape de valuation utilisant des techniques de résolution de contraintes avec des stratégies de valuation aux limites. Cette démarche a été expérimentée avec succès sur l'application industrielle IAS [GIX04] qui est une norme définissant des services d'Identification, d'Authentification et de Signature électronique embarqués sur des cartes à puces pour développer des applications comme les passeports électroniques. Mais la méthode n'a été appliquée que sur trois TP qui sont des enchaînements séquentiels très simples d'actions et d'états pour tester des propriétés de contrôle d'accès. La méthode

souffre de deux inconvénients. Le premier est lié à l'explosion combinatoire pour établir un chemin à partir d'un TP. Ceci est en partie dû à la richesse de description du langage proposé qui propose de définir des séquences complexes de transitions devant mener à un état cible défini par un prédicat p. De plus, cette construction est effectuée en aveugle, car indépendamment de M. Le second concerne la confiance dans les séquences de tests générées. Celle-ci est basée sur la couverture des tests qui est effectuée par l'évaluation du nombre d'états et de transitions couverts du TP. Comme la sélection des tests est effectuée à partir du TP, les taux de couverture sont en général soit bons soit mauvais, selon que le TP décrit peu ou beaucoup de chemins inexistants dans M. Dans les deux cas, les taux de couverture étant évalués indépendamment du modèle M, ils ne sont pas forcément significatifs sur M. Nous adressons ces deux problèmes dans cet article. Pour cela, nous proposons d'intégrer dans la démarche un modèle A^M qui correspond à une abstraction du modèle comportemental M sur certaines de ses variables d'état.

2.2 Démarche de génération

Les abstractions sont définies à partir des informations mises en jeu dans les TP : les variables d'état utilisées dans les propriétés d'état, et les variables d'état modifiées par les opérations activées. Cette abstraction apporte des réponses aux deux problèmes énoncés précédemment. La réponse au premier problème consiste à engendrer des tests à partir de la synchronisation de l'abstraction A^M avec TP. Les tests abstraits symboliques sont beaucoup plus précis que ceux produits dans l'approche [JMT08b]. L'abstraction les a complété en ajoutant aux propriétés d'état d'une part la condition d'activation des opérations, et d'autre part les conditions définissant les états symboliques. Elle a également instancié tous les noms d'opérations du TP par un sous-ensemble des noms d'opérations du modèle M. Enfin, l'évaluation de la couverture des jeux de tests étant effectuée sur la synchronisation de A^M et de TP, elle est beaucoup plus pertinente car elle est mesurée relativement aux exécutions de M qui satisfont TP au lieu de l'être uniquement sur TP.



(a) MBT à partir d'une abstraction (b) MBT avec critère dynamique

(c) MBT avec critère statique

FIG. 1 – Démarche de génération de tests par abstraction

Notre démarche est présentée en 3 vues dans la figure 1. La vue 1(c) présente le processus MBT usuel de sélection de tests à partir de critères structurels. Le processus MBT à partir de critères dynamiques est obtenu en remplaçant l'entrée M de la vue 1(c) par la vue 1(b). Le processus que nous proposons combine les trois vues en remplaçant le modèle M du processus MBT dynamique par la vue 1(a). Ceci permet de remplacer le modèle détaillé M en entrée du processus MBT par une abstraction de M.

– **Figure 1(c).** À partir d'un modèle comportemental M fourni par l'ingénieur validation et de critères statiques de sélection des tests (par exemple *tous les états*, ou *toutes les transitions*), le générateur de tests (LTG dans notre cas) calcule et value un ensemble de tests abstraits AT. Ces tests ont le niveau d'abstraction de M et doivent être concrétisés via une couche de concrétisation CL en un ensemble de tests concrets CT, exécutables sur l'implémentation sous test IUT. Leur

exécution permet de délivrer un verdict de réussite ou d'échec, en comparant les résultats prédits par M à ceux retournés par l'IUT.

– **Figure 1(b).** L'entrée M dans la vue 1(c) est remplacée par un modèle A^{P}. Celui-ci est le résultat de la synchronisation de M avec un critère dynamique de sélection TP. Ainsi les exécutions de A^{P} sont les exécutions de M qui satisfont TP.
– **Figure 1(a).** Là encore, l'entrée M du processus précédent (vue 1(b)), est remplacée par une autre entrée : une abstraction A^{M} du modèle M. L'originalité de notre démarche réside dans le fait que cette abstraction est *calculée* à partir de M et d'un TP, et non pas décrite directement par l'ingénieur validation. L'abstraction est calculée en deux temps : (i) un ensemble SS d'états symboliques de l'abstraction est calculé à partir des informations du TP (voir la partie 4.1). Le but est d'observer M relativement à cette information ; (ii) une abstraction A^{M} est calculée, dont les états symboliques sont ceux définis par SS. En pratique, le modèle M étant défini en B, nous utilisons *GénéSyst* [Sto07] pour calculer l'abstraction. Pour celui-ci, nous définissons l'ensemble d'états symboliques SS par une assertion qui est une disjonction de prédicats, chacun d'entre eux définissant un état symbolique.

2.3 Exemple : un robot de transport de pièces

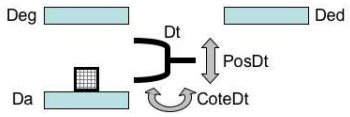


FIG. 2 – Dispositif physique du robot de transport de pièces

Notre approche est illustrée par l'exemple d'un robot de transport de pièces. Le dispositif physique se présente comme l'illustre la figure 2. Les dispositifs transportent trois types de pièces T_1, T_2, T_3, et peuvent n'en comporter aucune, dénoté l. Le dispositif d'arrivée des pièces ($Da \in \{l, T_1, T_2, T_3\}$) est en dessous de deux dispositifs d'évacuation des pièces, un à gauche ($Deg \in \{l, T_1, T_3\}$), l'autre à droite ($Ded \in \{l, T_2, T_3\}$) de la pince ($Dt \in \{l, T_1, T_2, T_3\}$). Dt est une pince qui a une position basse ou haute ($PosDt \in \{b, h\}$) et un sens gauche ou droit ($CoteDt \in \{g, d\}$). Le système est muni de neuf opérations. Ap fait arriver une pièce sur Da. C charge une pièce de Da dans Dt. Dc décharge la pièce dans Dt sur Deg ou Ded selon sa position. M fait monter Dt. D fait descendre Dt. Eg et Ed évacuent respectivement les pièces de Deg et Ded. Enfin Rd et Rg font tourner Dt respectivement à droite et à gauche. Les pièces de type T_1 sont obligatoirement déchargées sur Deg, celles de type T_2 sur Ded et celles de type T_3 sur l'un ou l'autre. Pour limiter les rotations à droite, la pince décharge en priorité les pièces de type T_3 sur Deg. Notre modèle fait une interprétation stricte de la priorité en choisissant de toujours décharger à gauche quand c'est possible même après une rotation à droite. Les rotations ne peuvent avoir lieu qu'en haut. La pince ne décharge que sur des évacuateurs libres.

3 Définitions préliminaires

Cette section présente des définitions formelles des objets sur lesquels s'appuient les étapes de la méthode présentée dans la section 4. Nous définissons les notions d'opération (définition 1) ou d'événement (définition 2) du modèle comportemental M. Ces notions sont utilisées pour définir l'information extraite des opérations activées par l'objectif de test. Puis, nous décrivons (définition 3) la notion de modèle comportemental, soit par une machine abstraite, soit par un système d'événements B. Ensuite, nous définissons la notion d'objectif de test (définition 4), utilisé comme critère dynamique de sélection, par un automate. Enfin, nous définissons la notion d'abstraction du modèle comportemental par un STES

(définition 5, issue de [Sto07]) et la notion de test abstrait symbolique (définition 6) par une exécution d'un STES.

3.1 Modèle comportemental

Les machines abstraites B ont été introduites par J. R. Abrial dans [Abr96a]. Elles permettent de définir des spécifications ouvertes de systèmes par un ensemble d'opérations. L'environnement agit sur le système en appelant des opérations. Intuitivement, une opération a une pré condition et modifie l'état interne des variables d'état par une substitution généralisée au sens B. Une opération est éventuellement munie de paramètres et retourne des résultats. Dans la suite de l'article, les opérations sont définies comme dans la définition 1.

Les systèmes d'événements B ont été introduits dans [Abr96b]. Ils définissent des spécifications fermées de systèmes par un ensemble d'événements. Un événement est une opération B sans paramètre et sans résultat puisque l'environnement est modélisé. Il n'y a donc pas d'appel extérieur des événements, mais ceux-ci se déclenchent automatiquement quand une condition, appelée *garde*, est satisfaite. Les événements n'ont donc plus de pré condition, mais une garde. Ils modifient toujours les variables d'état par une substitution généralisée.

Nous utilisons les deux formes de spécifications car le calcul d'abstractions par *GénéSyst* est défini sur des systèmes d'événements, et la génération de tests par LTG sur des machines abstraites. La différence de sémantique entre les deux concepts n'est pas un problème ici car LTG interprète les machines abstraites comme des systèmes d'événements en appelant une opération uniquement si sa pré condition est satisfaite et si l'opération est faisable (au sens du prédicat *fis* en B). De plus LTG engendrant des tests *off line*, il n'accepte que des spécifications déterministes pour calculer l'oracle et, dans le cas déterministe, les sémantiques des machines abstraites et des systèmes d'événements sont identiques.

Pour définir des spécifications B, nous rappelons les notions de prédicats B et de substitutions généralisées B. Les prédicats B sur un ensemble de variables x sont dénotés par $P(x)$, $Q(x)$, $I(x)$, $T(x)$, etc. Dans la suite de l'article, le prédicat $I(x)$ dénote un invariant et $T(p)$ dénote un prédicat de typage des paramètres p. Quand il n'y a pas d'ambiguïtés sur x, nous notons simplement P, Q, I, etc. Nous dénotons S les substitutions généralisées et E, F des expressions B. *Init* dénote la substitution d'initialisation. $y := E$ dénote la substitution élémentaire d'affectation de la variable y par la valeur de l'expression E. Nous notons $(y := E)$ *dans* S le prédicat qui est vrai si il existe une occurrence de la substitution $y := E$ dans la substitution S. Étant donné une substitution S et une post condition Q, nous dénotons $[S]Q$ la plus faible pré condition pour que Q soit satisfait après avoir appliqué la substitution S. Une machine abstraite définissant un modèle comportemental est définie comme dans la Définition 3.

Définition 1 (Opération) *Soit S^o une substitution. Soit r une liste de variables résultat, p une liste de paramètres et $T(p)$ une pré condition sur ces paramètres. Une opération de nom o est définie en B par :*

$$r \leftarrow o(p) =_{def} \mathsf{PRE}\ \ T(p)\ \mathsf{THEN}\ S^o\ \mathsf{END}.$$

Il est possible de transformer systématiquement et de plusieurs manières une machine abstraite en système d'événements (voir [Sto07] par exemple). La principale transformation consiste à supprimer les paramètres des opérations, pour obtenir des événements, en remplaçant les paramètres par des variables locales dont les valeurs sont choisies de manière non déterministe. Cette transformation revient à modéliser l'utilisateur qui appelait les opérations. La transformation d'une opération en événement est définie comme dans la définition 2.

Définition 2 (Événement) *Une opération o de la définition 1 est transformée en un événement o ainsi :*

$$o =_{def} \mathsf{ANY}\ p\ \mathsf{WHERE}\ T(p)\mathsf{THEN\ VAR}\ r\ \mathsf{IN}\ S^o\ \mathsf{END\ END}.$$

Pour engendrer des tests, la méthode appliquée par LTG consiste à décomposer chaque opération de la spécification en un ensemble de *comportements*. L'un des critères statiques de sélection des tests de

LTG consiste à couvrir tous les comportements. Intuitivement, un comportement d'une opération est un chemin de son graphe de flot de contrôle. Chaque comportement est défini par une condition d'activation et un effet. La condition d'activation est la condition nécessaire pour que s'exécute le chemin de contrôle. L'effet spécifie la modification des variables d'états effectuées par le chemin de contrôle. Par exemple, si S_1 et S_2 sont des compositions parallèles de substitutions élémentaires d'affectation, l'opération suivante

$$r \leftarrow o(p) =_{\text{def}} \mathsf{PRE}\ T(p)\mathsf{THEN\ IF}\ C_1\mathsf{THEN}\ S_1\ \mathsf{ELSE}\ S_2\ \mathsf{END\ END}$$

est constituée de deux comportements dont les conditions d'activation sont respectivement $T(p) \wedge C_1$ et $T(p) \wedge \neg C_1$ et dont les effets sont respectivement S_1 et S_2. Par exemple, si S_1 est l'affectation $x := E$, l'effet associé est exprimé par le prédicat $x' = E$. Formellement, un comportement peut se définir par un prédicat avant-après portant sur les variables x, x' où x désigne les variables d'états avant l'exécution du comportement et x' les variables d'état après exécution du comportement. Par exemple, le comportement où $S_1 =_{\text{def}} x := E$ est appliquée est exprimé par le prédicat $T(p) \wedge C_1 \wedge x' = E$. La décomposition des opérations en un ensemble de comportements par mise en forme normale disjonctive est décrite dans la thèse de F. Peureux [Peu02]. Dans la suite de l'article, nous notons *CA*(S) la condition d'activation de la substitution S.

Définition 3 (Modèle comportemental) *Un modèle comportemental* M*, dont l'ensemble des noms d'opérations est noté* O^{M}*, est une machine abstraite B définie par* $\langle C^{\mathsf{M}}, R^{\mathsf{M}}, x^{\mathsf{M}}, I^{\mathsf{M}}, \mathit{Init}^{\mathsf{M}}, \mathit{OP}^{\mathsf{M}}\rangle$ *avec :*

- C^{M} *est un ensemble de noms de constantes (clause* CONSTANTS*),*
- R^{M} *est un prédicat définissant les constantes de* C^{M} *(clause* PROPERTIES*),*
- x^{M} *est une liste de variables d'état,*
- I^{M} *est un prédicat invariant sur* C^{M} *et* x^{M}*,*
- $\mathit{Init}^{\mathsf{M}}$ *est une substitution généralisée définissant l'état initial des variables de* x^{M}*,*
- OP^{M} *est l'ensemble de définitions des opérations de* O^{M}*.*

Dans la suite de l'article, l'ensemble des prédicats d'états sur les variables x^{M} d'un modèle comportemental M est dénoté $\mathit{Pred}^{\mathsf{M}}$. Sur cet ensemble nous notons x *libre_ dans* P un prédicat qui est vrai si il y a au moins une occurrence de la variable x qui est libre dans le prédicat P.

3.2 Objectif de test, Abstraction et Test abstrait symbolique

Dans [JMT08a], nous avons défini un langage pour décrire des objectifs de test qui combinent appels d'opérations et description d'états cibles visés par ces opérations et définis par des prédicats d'état. Par exemple, la figure 3 a pour objectif de tester la propriété de priorité de déchargement des pièces de type T_3 à gauche (sur Deg). Pour cela, nous plaçons le système dans un état où la pince est tournée à droite et contient une pièce de type T_3 et dans lequel l'évacuateur gauche est libre pour vérifier que la pince revient décharger cette pièce à gauche. La sémantique d'un objectif de test est un automate (voir définition 4). Un exemple est donné dans la figure 3. Un objectif de test TP est associé à un modèle comportemental M qui spécifie le système sous test et qui est décrit en B comme dans la définition 3. Nous disons que TP est défini sur M. Un TP est compatible avec un modèle M si d'une part les étiquettes de ses transitions sont des noms d'opérations de M ($\in O^{\mathsf{M}}$) ou bien le symbole \$op, qui désigne n'importe laquelle des opérations, et d'autre part si ses états q sont décorés par des prédicats d'état cibles ($\lambda^{\mathsf{P}}(q) \in \mathit{Pred}^{\mathsf{M}}$). Ces prédicats sont décrits en B et portent sur les variables d'état x^{M} de M. Par exemple, sur la figure 3 l'état q_1 est décoré par le prédicat d'état $Dt = T_3 \wedge Deg \neq l$ qui spécifie que la pince contient une pièce de type T_3 et que l'évacuateur gauche contient une pièce.

Définition 4 (Objectif de test) *Un objectif de test sur un modèle* M *est un quintuplet* $\langle Q^{\mathsf{P}}, q_0^{\mathsf{P}}, T^{\mathsf{P}}, \lambda^{\mathsf{P}}, Q_f^{\mathsf{P}}\rangle$ *où* Q^{P} *est un ensemble fini d'états,* $q_0^{\mathsf{P}} \in Q^{\mathsf{P}}$ *est un état initial,* $Q_f^{\mathsf{P}} \subseteq Q^{\mathsf{P}}$ *est un ensemble d'états terminaux,* $T^{\mathsf{P}} \subseteq Q^{\mathsf{P}} \times O^{\mathsf{M}} \cup \{\$\mathsf{op}\} \times Q^{\mathsf{P}}$ *est l'ensemble des transitions étiquetées et* $\lambda^{\mathsf{P}} \in Q^{\mathsf{P}} \rightarrow Pred^{\mathsf{M}}$ *est une fonction totale qui associe un prédicat d'état, dénoté* $\lambda^{\mathsf{P}}(q)$ *à chaque état* q *de* Q^{P}*.*

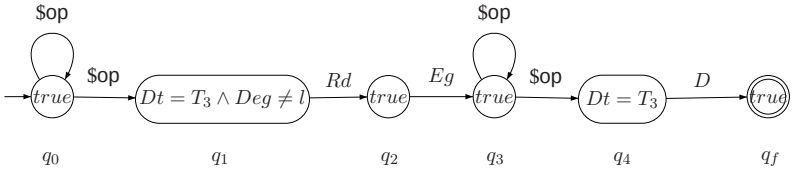


FIG. 3 – Un objectif de test pour le robot de transport de pièces

L'abstraction d'un modèle M est un automate dont l'alphabet est celui des noms d'opérations de M (c'est à dire O^{M}), et dont l'ensemble des variables x^{A} est un sous-ensemble de x^{M}. Notons que l'ensemble des prédicats d'états sur x^{A}, dénoté $Pred^{\mathsf{A}}$, est un sous-ensemble de $Pred^{\mathsf{M}}$. Chaque état symbolique est décoré par le prédicat d'état qui le définit. La relation de transition est un ensemble de transitions étiquetées par deux prédicats d'état et par un nom d'opération. La conjonction des deux prédicats, appelés condition de déclenchabilité et condition d'atteignabilité, définit la condition d'activation de l'opération. La définition 5 d'une abstraction est celle qui est décrite dans [Sto07] à laquelle nous avons ajouté un ensemble d'états terminaux afin de définir des tests abstraits symboliques finis. Dans l'abstraction d'un modèle M, tous les états de Q^{A} sont terminaux. Par contre, dans le STES qui est le résultat de la synchronisation d'un STES et d'un TP, les états terminaux sont les états composés d'un état terminal de chacun.

Définition 5 (Abstraction - Système de Transition Étiqueté Symbolique) *Une abstraction A^M d'un modèle M est un STES défini par le n-uplet $\langle x^A, O^M, Q^A, q_0^A, Def^A, L^A, T^A, Q_f^A\rangle$ où :*

- *$x^A(\subseteq x^M)$ est un ensemble de variables,*
- *Q^A est l'ensemble des identificateurs des états symboliques,*
- *$q_0^A(\in Q^A)$ est l'état initial,*
- *$Def^A(\in Q^A \rightarrow Pred^A)$ associe à chaque état symbolique le prédicat d'état qui le définit,*
- *$L^A(\subseteq Pred^A \times Pred^A \times O^M)$ est un ensemble d'étiquettes des transitions,*
- *$T^A(\subseteq Q^A \times L^A \times Q^A)$ est une relation de transition,*
- *$Q_f^A(\subseteq Q^A)$ est un ensemble d'états terminaux.*

Un test abstrait symbolique est une exécution d'un STES. Il est défini dans la définition 6. C'est une séquence de couples formés d'une étiquette et d'un état cible et qui débute par l'état initial du STES. Notons que l'étiquette indique l'opération appliquée par la transition et sa condition de déclenchement. Notons également que chaque état est décoré par un prédicat le définissant.

Définition 6 (Test abstrait symbolique) *Un test abstrait symbolique σ est une exécution finie d'un STES $\langle x^A, O^M, Q^A, q_0^A, Def^A, L^A, T^A, Q_f^A\rangle$. C'est une séquence $\sigma = q_0^A, (l_1, q_1), \ldots, (l_n, q_n)$ telle que, pour i de 0 à n-1, $q_i \xrightarrow{l_{i+1}} q_{i+1} \in T^A$ et $q_n \in Q_f^A$.*

Sur la base de ces définitions, nous allons maintenant définir la construction de notre abstraction.

4 Génération d'une abstraction et synchronisation avec un modèle comportemental

Nous utilisons *GénéSyst* pour engendrer une abstraction d'un modèle comportemental M. Pour calculer une abstraction, *GénéSyst* a besoin d'une définition de l'ensemble d'états symboliques. Cet ensemble peut être défini par une assertion ($\in Pred^{\mathsf{M}}$) de la forme $P_1 \vee P_2 \vee \ldots \vee P_n$ où chaque élément

P_i de la disjonction définit un état symbolique. Notons que nous n'utilisons que des formes disjonctives définissant un ensemble d'états disjoints deux à deux. Dans cette section, nous présentons la méthode de définition de cette assertion puis nous donnons la définition de la synchronisation de l'abstraction A^{M} avec un objectif de test TP. C'est à partir de cette synchronisation qu'est effectuée la génération de tests abstraits qui passe par l'animation de tests abstraits symboliques (voir définition 6).

4.1 Définition de l'abstraction

Dans cette section, nous présentons une méthode de définition d'une abstraction d'un modèle comportemental M pour engendrer des tests à partir d'un objectif de test TP compatible avec M. L'ensemble d'états symboliques de l'abstraction est défini à partir des prédicats d'état et des opérations apparaissant dans l'objectif de test.

Étant donné un modèle comportemental M et un objectif de test compatible TP, pour définir l'ensemble d'états symboliques, nous proposons une démarche en trois étapes :

- extraction des noms des variables de x^{M} utilisées dans les propriétés d'état de TP et modifiées par les opérations appelées dans TP,
- partitionnement des domaines de ces variables en sous-domaines en fonction de l'utilisation des variables dans les prédicats d'états et de leur modification par les opérations du TP,
- définition de l'assertion disjonctive définissant l'ensemble d'états symboliques et construction de l'abstraction.

4.1.1 Extraction des noms de variables

Étant donné un modèle comportemental M et un objectif de test compatible $\mathsf{TP} = \langle Q^{\mathsf{P}}, q_0^{\mathsf{P}}, T^{\mathsf{P}}, \lambda^{\mathsf{P}}, Q_{\mathrm{f}}^{\mathsf{P}} \rangle$, l'ensemble des noms d'opérations O^{P} de M utilisés dans TP est défini ainsi :

$$O^{\mathsf{P}} =_{\text{def}} \{o \mid o \in O^{\mathsf{M}} \wedge \exists q \in Q^{\mathsf{P}} \cdot \exists q' \in Q^{\mathsf{P}} \cdot (q \xrightarrow{o} q' \in T^{\mathsf{P}})\}.$$

L'ensemble $x^{\mathsf{P}}(o)$ des variables modifiées par l'opération o est défini ainsi :

$$x^{\mathsf{P}}(o) =_{\text{def}} \{y \mid y \in x^{\mathsf{M}} \wedge (y := E) \textit{ dans } S^o\}.$$

Notons que toute substitution multiple $y_1, y_2 := E_1, E_2$ est considérée comme une substitution parallèle de la forme $y_1 := E_1 \parallel y_2 := E_2$. Alors, l'ensemble x^{A} des variables de l'abstraction issues d'un objectif de test TP compatible avec M est défini ainsi :

$$x^{\mathsf{A}} =_{\text{def}} \{y \mid y \in x^{\mathsf{M}} \wedge \exists q \in Q^{\mathsf{P}} \cdot (y \textit{ libre\_ dans } \lambda^{\mathsf{P}}(q))\} \cup (\bigcap_{o \in O^{\mathsf{P}}} x^{\mathsf{P}}(o)).$$

Cet ensemble est constitué de toutes les variables de M utilisées dans les prédicats d'états de TP et des variables modifiées par toutes les opérations de TP. Concernant les variables modifiées par les opérations, étant donné les exemples traités, nous avons choisi de limiter leur nombre en ne prenant que celles qui sont modifiées par toutes les opérations utilisées dans TP afin de limiter l'explosion du nombre d'états symboliques sachant que le nombre d'obligations de preuve pour générer l'abstraction est proportionnel au nombre d'événements et au carré du nombre d'états symboliques du modèle.

Sur l'exemple de TP de la figure 3, l'ensemble de variables considéré est constitué de deux variables, Dt et Deg qui apparaissent dans les prédicats d'état. Aucune variable n'est modifiée par les trois opérations utilisées :

$$x^{\mathsf{A}} =_{\text{def}} \{Deg, Dt\}.$$

4.1.2 Partitionnement des domaines des variables

Le partitionnement des domaines des variables utilisées par un objectif de test TP a pour but d'observer les états symboliques apparaissant dans TP et les effets des opérations utilisées dans TP. Pour définir les sous-domaines de chaque variable x de x^{A}, nous définissons d'une part l'ensemble des sous-domaines issus des propriétés d'états cibles de TP et d'autre part les sous-domaines des états cibles des opérations activées par TP.

L'ensemble des sous-domaines de la variable x, issus des propriétés définissant les états cibles de TP, dénoté $\mathsf{SDE}(x)$, est l'union pour tous les prédicats d'états où x apparaît libre, des ensembles de valeurs de x, notées x', pour lesquelles il existe au moins une valeur des autres variables d'état du modèle M qui satisfont le prédicat d'état cible $\lambda^{\mathsf{P}}(q)$. Soit $y^{\mathsf{M}} = x^{\mathsf{M}} - \{x\}$, l'ensemble des variables d'états de M à l'exception de x, l'ensemble des sous-domaines sont définis ainsi :

$$\mathsf{SDE}(x) =_{\text{def}} \bigcup_{\{q \mid x \textit{ libre\_ dans } \lambda^{\mathsf{P}}(q)\}} \{x' \mid \exists y^{\mathsf{M}} \cdot ([x := x']\lambda^{\mathsf{P}}(q))\}.$$

Par exemple, pour la variable Deg, on obtient le sous-domaine suivant défini par un prédicat d'état :

$$\mathsf{SDE}(Deg) =_{\text{def}} \{Deg \neq l\}.$$

L'ensemble des sous-domaines de la variable x ($x \in x^{\mathsf{A}}$) issus des états cibles des opérations de TP, est défini comme l'union des sous-domaines atteints par chaque action de substitution d'affectation de la forme $x := E$ apparaissant dans les opérations de O^{P}. Notons A_x^{P} l'ensemble des couples affectation $x := E$ et condition d'activation de $x := E$ apparaissant dans les opérations utilisées dans TP.

$$A_x^{\mathsf{P}} =_{\text{def}} \{x := E, \mathit{CA}(x := E) \mid \exists o \cdot (o =_{\text{def}} r \leftarrow o(p) = \mathsf{PRE}\ T(p)\ \mathsf{THEN}\ S^o\ \mathsf{END} \wedge o \in O^{\mathsf{P}} \wedge (x := E)\ \textit{dans}\ S^o)\}.$$

Par exemple, pour la variable Deg, le couple $Deg := Dt, PosDt = h \wedge ((Deg = l \wedge CoteDt = g \wedge Dt \in \{T1, T3\}) \vee (Ded = l \wedge CoteDt = d \wedge (Dt = T2 \vee (Dt = T3 \wedge Deg \neq l))))$ est issu de l'opération D.

L'ensemble des sous-domaines d'une variable x engendré par les opérations, noté $\mathsf{SDO}(x)$, est l'union pour tous les couples de A_x^{P} des sous-domaines cibles de $x := E$ dans le contexte de sa condition d'activation. Le sous-domaine cible de l'affectation $x := E$, qui est activée si la condition $\mathit{CA}(x := E)$ est vraie, est l'ensemble des valeurs x' vérifiant l'invariant et égales à la valeur de E obtenue à partir des valeurs des variables de x^{M} telles qu'il existe au moins une valeur de ces variables qui vérifie l'invariant, les conditions sur les constantes et la condition d'activation de $x := E$. Si $\mathsf{SDO}(x)$ ne recouvre pas totalement le domaine de x défini dans I^{M}, nous ajoutons le sous-domaine complémentaire de l'ensemble des sous-domaines $\mathsf{SDO}(x)$ pour définir l'ensemble des sous-domaines de x, noté $\mathsf{SD}(x)$.

$$\mathsf{SDO}(x) =_{\text{def}} \bigcup_{\{(x:=E, \mathit{CA}(x:=E)) \in A_x^{\mathsf{P}}\}} \{x' \mid \exists x^{\mathsf{M}} \cdot ([x := x']I^{\mathsf{M}} \wedge x' = E \wedge I^{\mathsf{M}} \wedge R^{\mathsf{M}} \wedge \mathit{CA}(x := E)\}.$$

$$\mathsf{SD}(x) =_{\text{def}} \mathsf{SDO}(x) \cup \{x' \mid \exists x^{\mathsf{M}} \cdot ([x := x']I^{\mathsf{M}} \wedge I^{\mathsf{M}} \wedge R^{\mathsf{M}} \wedge \forall D \in \mathsf{SDO}(x) \cdot (x' \notin D)\}.$$

Par exemple, $\mathsf{SDO}(Deg)$ est composé de deux sous-domaines, $Deg = l$ et $Deg \in \{T_1, T_3\}$. Le sous-domaine $Deg = T_2$ complète $\mathsf{SDO}(Deg)$ pour obtenir $\mathsf{SD}(Deg)$. Notons qu'une variable x peut à la fois apparaître dans les prédicats cibles et être modifiée par les opérations. Dans ce cas, l'ensemble des sous-domaines de x est l'union de $\mathsf{SD}(x)$ et de $\mathsf{SDE}(x)$. Notons enfin que si les sous-domaines ne sont pas disjoints deux à deux, nous les décomposons en sous-domaines disjoints afin d'éviter d'obtenir une abstraction non déterministe. Par exemple, l'union disjointe des sous-domaines de $\mathsf{SD}(Deg)$ et de $\mathsf{SDO}(Deg)$ donne l'ensemble des sous-domaines présentés dans la ligne 2 de la figure 4.

Les prédicats définissant les sous-domaines sont obtenus par résolution de contraintes dans la mesure où les domaines des variables du modèle comportemental pour le test ont tous été définis par des ensembles finis.

4.1.3 Définition de l'ensemble d'états symboliques

L'ensemble d'états symboliques est défini par une assertion qui est une disjonction de prédicats, chacun définissant un état symbolique. Nous l'obtenons en effectuant le produit cartésien des ensembles de sous-domaines de chaque variable.

Soit $x^{\mathsf{P}} = \{x_1, x_2, \ldots, x_n\}$, l'ensemble des variables extraites d'un TP pour engendrer une abstraction. Soit $\mathsf{SD}(x_i) = \{D_i^1, D_i^2, \ldots, D_i^{m_i}\}$, l'ensemble des sous-domaines de la variable x_i. L'assertion $\mathcal{A}$ définissant l'ensemble d'états symboliques est la suivante :

$$\mathcal{A} =_{\text{def}} \bigvee_{\{(k_1,\ldots,k_n) \mid k_j \in 1..m_j \wedge j \in 1..n\}} \bigwedge_{i \in 1..n} (x_i \in D_i^{k_i})$$

Sur l'exemple, nous obtenons l'assertion représentée dans la figure 4 qui définit 9 états symboliques dont seulement 6 sont atteignables car $Deg = T_2$ est interdit par les règles de déchargement.

Variable	*Sous-domaines*
`Deg`	$\{l\}$, $\{T_1,T_3\}$, $\{T_2\}$
`Dt`	$\{l\}$, $\{T_1,T_2\}$, $\{T_3\}$

FIG. 4 – Sous-domaines des variables d'abstraction du robot de transport de pièces pour l'objectif de test de la figure 3

4.2 Synchronisation d'une abstraction et d'un objectif de test

Pour générer les tests nous synchronisons une abstraction A^{M} définie par un STES avec un objectif de test TP compatible. Une abstraction $\mathsf{A}^{\mathsf{M}} = \langle x^{\mathsf{A}}, O^{\mathsf{M}}, Q^{\mathsf{A}}, q_0^{\mathsf{A}}, \mathit{Def}^{\mathsf{A}}, L^{\mathsf{A}}, T^{\mathsf{A}}, Q_{\mathrm{f}}^{\mathsf{A}} \rangle$ d'un modèle M et un TP sont compatibles si :

- x^{A} est l'ensemble des noms de variables observées sur TP pour extraire l'abstraction,
- O^{M} est l'ensemble des noms d'opérations de M qui étiquettent les transitions TP et de A^{M},
- Q^{A} est un ensemble d'identificateurs de chaque état symbolique définis à partir de TP,
- $\mathit{Def}^{\mathsf{A}}$ associe à chaque élément de Q^{A} un élément du prédicat disjonctif $\mathcal{A}$ défini à partir de TP.

Le produit synchrone entre un TP et une abstraction A^{M} d'un modèle M compatibles est décrit définition 7. Son ensemble d'états est un sous-ensemble du produit cartésien des ensembles d'états. On ne retient que les couples d'états symboliques où celui de l'abstraction est un sous-ensemble de celui du TP. Notons que les états symboliques de l'abstraction sont par construction tous des sous-ensembles d'états symboliques du TP. L'état initial est le couple formé des états initiaux des deux. Le prédicat d'état associé aux états est celui de l'état symbolique de l'abstraction. Les transitions synchronisent une transition de chacun appliquant la même opération. Notons que $\$\mathsf{op}$ se synchronise avec toutes les opérations. Les conditions de déclenchabilité p_1 et d'atteignabilité p_2 sont celles de l'abstraction. Enfin, les états terminaux sont les couples d'états terminaux. Le produit synchronisé pour l'exemple est représenté dans la figure 5. Ce graphe est composé de deux sous-graphes reliés par la séquence d'opérations Rd suivi de Eg. Chaque sous-graphe instancie les exécutions abstraites par $\$\mathsf{op}^+$ dans TP de la figure 3. Ce graphe est un cas particulier qui ne contient qu'un seul état d'acceptation postérieur à l'opération de déchargement (D), dont l'état résultant correspond à un seul état symbolique de l'abstraction.

Définition 7 (Produit synchrone d'un TP et d'un STES compatibles) *Le produit synchrone entre un objectif de test* $\mathsf{TP} = \langle Q^{\mathsf{P}}, q_0^{\mathsf{P}}, T^{\mathsf{P}}, \lambda^{\mathsf{P}}, Q_f^{\mathsf{P}} \rangle$ *et un STES* $\mathsf{A}^{\mathsf{M}} = \langle x^{\mathsf{A}}, O^{\mathsf{M}}, Q^{\mathsf{A}}, q_0^{\mathsf{A}}, \mathit{Def}^{\mathsf{A}}, L^{\mathsf{A}}, T^{\mathsf{A}}, Q_f^{\mathsf{A}} \rangle$ *abstraction d'un modèle M, compatibles, est un STES* $\langle x^{\mathsf{A}}, O^{\mathsf{M}}, Q^S, q_0^S, \mathit{Def}^S, L^{\mathsf{A}}, T^S, Q_f^S \rangle$ *où :*

- $Q^S (\subseteq Q^{\mathsf{P}} \times Q^{\mathsf{A}})$ *est l'ensemble d'états symboliques tels que chaque état* $(q^{\mathsf{P}}, q^{\mathsf{A}})$ *vérifie la condition* $\mathit{Def}^{\mathsf{A}}(q^{\mathsf{A}}) \Rightarrow \mathit{Def}^{\mathsf{P}}(q^{\mathsf{P}})$,

Modèle B					Objectif de test			
Nom	#Cpts.	#Variables d'états	#Lignes B	#États (borne sup.)	#États	#Trans.	#Op.	#Préd. états
QuiDonc	20	3	216	18	4	4	2	1
Robot	10	6	107	384	5	6	3	2
DeMoney	42	8	530	10^{21}	2	4	2	0
					4	5	2	2
IAS	12	9	1032	∞	12	13	11	8

TAB. 1 – Présentation des différents modèles B et TP utilisés

- $q_0^S = (q_0^P, q_0^A)$,
- $Def^S = \{(q^P, q^A) \mapsto Def^A(q^A) \mid (q^P, q^A) \in Q^S\}$,
- $T^S = \{(q^P, q^A) \xrightarrow{(p_1,p_2,o)} (q'^P, q'^A) \mid (q^P \xrightarrow{o} q'^P \in T^P \vee q^P \xrightarrow{\$op} q'^P \in T^P) \wedge q^A \xrightarrow{(p_1,p_2,o)} q'^A \in T^A \wedge Def^A(q'^A) \Rightarrow Def^P(q'^P)\}$,
- $Q_f^S = \{(q^P, q^A) \mid q^P \in Q_f^P \wedge q^A \in Q_f^A\}$.

On calcule un ensemble de tests abstraits symboliques (voir définition 6) à partir du graphe représentant le produit synchrone entre l'abstraction du modèle M et un TP.

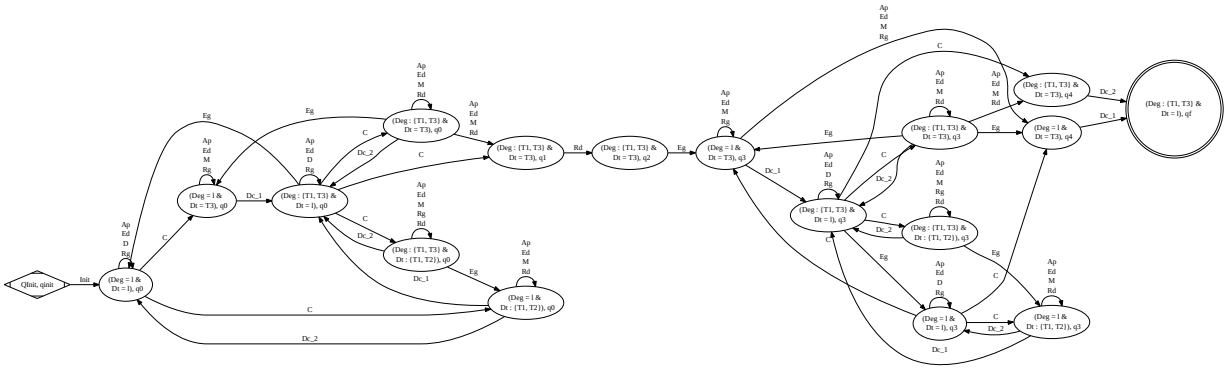


FIG. 5 – Le produit synchrone effectué entre l'abstraction du robot et l'objectif de test de la figure 3

5 Résultats expérimentaux

Nous avons calculé des tests à partir du produit synchrone (voir définition 7) en appliquant un algorithme de postier chinois [Thi03] pour couvrir toute les transitions de ce graphe. Nous avons expérimenté la démarche sur quatre exemples de taille croissante : le système d'annuaire inversé Qui-Donc [UL06], le robot transporteur de pièces, Demoney [MM02] et IAS [GIX04]. Pour les exemples du robot, de Qui-Donc et d'IAS, nous avons généré des tests pour un seul objectif de test. Pour IAS, celui-ci est un enchaînement de dix ou onze opérations visant huit prédicats d'état différents. Pour Demoney, nous avons utilisé deux objectifs de test, l'un étant un enchaînement d'opérations sans prédicat d'état, l'autre étant un enchaînement avec prédicat d'état et opération non définie ($op), comme celui du Qui-Donc, et également du robot, présenté figure 3.

Le tableau 1 présente les différents modèles utilisés pour les expérimentations. Il est constitué de deux parties. La partie « modèle B » donne des mesures sur le modèle comportemental : le nombre d'opérations, le nombre de variables d'état, le nombre de lignes de B, une borne supérieure du nombre d'états. La partie « objectif de test » indique pour TP son nombre d'états, de transitions, d'opérations différentes étiquetant ses transitions, et de prédicats d'états différents de `true`.

Le tableau 2 présente les résultats expérimentaux sur la construction d'abstraction à partir d'informations d'extraites d'un objectif de test. Il se compose de deux parties. La partie « génération de l'abstrac-

Modèle B	Génération de l'abstraction						Produit synchrone		
	#Var.	#Part. par var.	#États	#Trans.	#PO	Tps	#États	#Trans.	Tps
QuiDonc	3	2.67	7	57	1242	9 min	11	66	43 ms
Robot	2	3	6	37	430	9 min	17	94	77 ms
DeMoney	1	4	4	176	964	12 min	9	404	55 ms
	3	3.33	13	229	9481	75 min	21	337	207 ms
IAS	5	2.6	96	7704	131569	34 h	209	1437	88s

TAB. 2 – Résultats expérimentaux sur la construction d'abstraction

Modèle	Génération de tests LTG		Couverture des tests LTG sur le produit synchrone			Génération de tests à partir des abstractions			
	#Tests	Longueur moyenne	Tests non exécutables	Couverture des états	Couverture des trans.	#Tests	Longueur moyenne	Écart-type	Tps
QuiDonc	20	4.15	1	6/11 (54.5 %)	19/66 (28.8 %)	6	19.17	19.0	
Robot	13	5.46	0	8/17 (47.1 %)	37/94 (39.4 %)	11	11.27	2.89	
DeMoney	65	2.76	1	4/9 (44.4 %)	40/404 (9.9 %)	46	9.78	8.0	< 1s
			6	7/21 (33.3 %)	37/337 (10.9 %)	32	8.53	2.91	
IAS	54	3.22	46	5/209 (2.4 %)	4/1437 (0.3 %)	1116	9.72	<1	31 s

TAB. 3 – Résultats expérimentaux de génération de test à partir de critères statiques

tion » donne le nombre de variables d'état utilisées pour la génération de l'abstraction, le nombre moyen de sous-domaines par variable, le nombre d'états symboliques atteignables, de transitions et d'obligations de preuve (PO) engendrés par *GénéSyst*. Elle donne également le temps de calcul de l'abstraction avec un Pentium 2.8 GHz muni de 1 Go de mémoire vive. La partie « produit synchrone » présente le nombre d'états et de transitions du produit synchrone effectué entre un TP et l'abstraction. Elle présente également le temps nécessaire à ce calcul.

Le tableau 3 expose les résultats de la génération de tests par LTG en appliquant les critères statiques de sélection « tous les comportements », couverture C/DC des décisions et conditions. On y voit également les résultats de la génération de test par l'algorithme du postier chinois, sur les graphes des produits synchrones présentés dans le tableau 2. La partie « génération de tests LTG » donne le nombre de tests et leur longueur moyenne. La partie « couverture des tests LTG sur le produit synchrone » donne les mesures de la couverture des états et des transitions des tests LTG sur le produit synchrone correspondant. Elle présente également le nombre de tests LTG ne correspondant à aucun chemin de ce même produit synchrone. La partie « génération de tests à partir des abstractions » présente le nombre de tests abstraits symboliques générés par le postier chinois sur le produit synchrone, ainsi que leur longueur moyenne et l'écart-type associé. Elle présente également le temps nécessaire à cette génération.

Le tableau 3 met en évidence que le nombre de tests produits par LTG est proportionnel au nombre de comportements du modèle. La longueur moyenne des tests est petite, car le générateur trouve en général le plus court chemin pour activer chaque comportement.

Le tableau 2 montre que la technique d'abstraction à partir d'objectifs de test permet d'obtenir une vue d'un système de grande taille, sous la forme d'automates de taille utilisable en pratique. Les tableaux 2 et 3 montrent que les temps de génération du produit synchrone et des tests sont tout à fait acceptables même pour le modèle de grande taille d'IAS. Dans le cas de l'application réelle IAS (possédant une centaine d'états symboliques), la méthode est difficilement utilisable à cause du temps de génération de l'abstraction. Celle-ci étant effectuée par *GénéSyst*, il semble possible d'optimiser cette phase, par exemple en réduisant le nombre d'obligations de preuve - qui est de l'ordre de $n^2 \times p$ où n est le nombre d'états symboliques et p le nombre d'opérations - et en améliorant la connexion avec le prouveur. Le nombre d'états de l'abstraction n est borné par d^v où v est le nombre de variables d'états du modèle pris en compte dans la génération de l'abstraction, et d est le nombre moyen de sous-domaines par variable. Notons que l'exemple IAS peut être considéré comme un exemple de grande taille non pas à cause de la taille du modèle B mais à cause de la taille de son abstraction en nombre d'états symboliques.

Les expériences montrent que les tests produits à partir des abstractions sont complémentaires des tests produits par LTG à partir de critères statiques de sélection. D'une part, certains tests LTG ne sont pas produits par notre méthode. D'autre part, les tests LTG ne couvrent que 30 à 60% des états du produit

synchronisé et 10 à 40% des transitions. Notons toutefois que ces ratios sont beaucoup plus petits pour le cas d'IAS. Ceci s'explique par le fait que le TP associé est un enchaînement d'opérations nommés. Les tests générés par LTG sont des enchaînements d'opérations menant à une cible, et ne correspondent qu'à 8 fragments de chemins du produit synchrone sur 54. Enfin, la longueur moyenne des tests LTG est de 2 à 5 fois plus petite. Ce facteur est une sous-approximation puisque nous avons pu constater par animation manuelle de quelques exemples que les tests symboliques engendreront des tests abstraits 2 à 3 fois plus longs. Une comparaison plus précise entre les tests symboliques et les tests valués fournis par LTG n'a pas pu se faire en raison de la différence d'abstraction entre les deux ensembles. Pour les comparer, il faudra valuer systématiquement les tests symboliques. Il sera peut être nécessaire de fusionner la phase de génération de tests symboliques avec la phase de valuation des tests.

Les tests générés par la méthode de dépliage de l'objectif de test présentée dans [JMT08b] sur IAS sont de même longueur que les tests produits à partir des abstractions. Ceci s'explique par le fait que l'objectif de test utilisé consiste en une succession d'appels d'opérations, sans cycle. Le produit synchrone nous donne des informations supplémentaires vis à vis du dépliage du TP. Le nombre de tests obtenus à partir de l'abstraction est beaucoup plus grand car il couvre les transitions du modèle. Ainsi, nous améliorons la méthode [JMT08b] car nous obtenons des tests qui couvrent le modèle et le TP au lieu de couvrir uniquement le TP.

6 Conclusion, travaux connexes et perspectives

Dans cet article, nous avons présenté la construction d'une abstraction A^M d'un modèle M effectuée sur certaines des variables d'état utilisées et modifiées par un objectif de test TP. A^M et TP sont synchronisés pour obtenir un modèle A^P constitué de l'ensemble des séquences de A^M qui satisfont TP. Les tests abstraits symboliques sont un ensemble de ces séquences couvrant les états et/ou les transitions de la synchronisation A^P. Comme A^M est une abstraction qui contient plus d'exécutions que M, mais dont tous les états sont atteignables et toutes les transitions sont franchissables, il est faisable d'engendrer des séquences de tests couvrant tous les états et toutes les transitions de A^P. Ces séquences sont obtenues par la phase de génération de tests et de valuation à partir du modèle M.

De nombreux travaux existent pour la génération de tests à partir d'abstractions. Notre originalité par rapport à ces travaux est que nous calculons automatiquement l'abstraction fournie en entrée du processus, là où elle est en général décrite manuellement dans les travaux existants.

Notons cependant que, comme dans notre approche, l'outil AGATHA [RGLG03, GGL04] calcule une abstraction à partir d'un modèle donné en entrée. L'abstraction est une arborescence symbolique d'exécutions, à partir de laquelle l'outil peut calculer par résolution de contraintes des tests en instanciant les exécutions symboliques décrites par l'abstraction. AGATHA ne considère pas d'objectifs de tests donnés par l'utilisateur. Mais, comme décrit dans [TGLG05], ils peuvent être extraits de l'abstraction sous la forme d'IOSTS, exploitables par exemple par STG [JJRZ05]. Les modèles en entrée d'AGATHA sont des automates de contrôle, alors que nos modèles sont des spécifications B. Notre approche est conceptuellement différente dans le sens où, à l'inverse de [TGLG05], c'est l'objectif de test qui est en entrée du processus, et qui fournit les informations à partir desquelles est calculée notre abstraction.

Comme dans [JJRZ05] et [CIVDP07], notre approche utilise un modèle symbolique dont elle effectue le produit synchronisé avec un objectif de test. Nous calculons un modèle symbolique adapté à l'objectif de test, alors qu'il est donné par l'utilisateur sous la forme d'un IOSTS dans [JJRZ05], et d'un IOLTS dans [CIVDP07]. Nous utilisons des méthodes d'évaluation symbolique pour calculer l'abstraction. Dans [JJRZ05], les auteurs utilisent de l'interprétation abstraite pour réduire la taille du produit synchronisé, en éliminant les états ne conduisant pas à des tests pertinents (ensemble d'états atteignables) par des techniques de sur-approximation. Dans [CIVDP07], l'abstraction est un IOLTS avec des opérations d'entrée paramétrées, ce qui est également le cas dans nos modèles. Toutes les approches que nous venons de présenter utilisent des techniques de résolution de contraintes [BLP04] pour

effectuer la valuation des paramètres.

Par rapport aux autres approches de génération de tests à partir d'objectifs de test dynamiques, notre méthode prend en entrée des objectifs de test décrivant à la fois des opérations à activer et des états cibles à atteindre. Dans [ABM98] et [HLSU02], les objectifs de tests sont des propriétés PLTL décrivant des enchaînements d'états. Dans [JJRZ05, CIVDP07], les objectifs de test sont des enchaînements d'appels d'opérations exprimés par des IOSTS ou IOLTS.

Notre approche est tributaire de l'automatisation des preuves. En cas de défaut de preuve, il est possible de recourir à des prouveurs interactifs. Dans ce cas, il peut être également envisageable d'engendrer un modèle par approximation. Mais en pratique, comme les modèles pour le test manipulent des variables à domaine fini, les preuves sont décidables. C'est ce que nous avons pu constater sur les exemples traités. L'autre difficulté pratique que nous avons mise en évidence sur les exemples de l'expérimentation est le coût en temps de construction de l'abstraction. Cette question devra être évaluée plus finement en fonction des optimisations envisageables. L'enjeu est de développer une méthode par abstraction où l'amélioration des performances, évaluée en espace et en qualité des tests, obtenue par l'utilisation d'une abstraction pour la génération de tests soit plus importante que la perte de temps due au calcul de l'abstraction, même si cela reste du temps machine.

Enfin, est-ce que notre solution répond aux deux problèmes évoqués dans la section problématique ? A^P apporte trois informations complémentaires pour guider la recherche : les prédicats définissant les états symboliques, les conditions d'activation des transitions symboliques et des contraintes sur les opérations utilisables dans les séquences de transitions permettant de passer d'un état à un autre. Cet avantage est susceptible d'apporter une réponse satisfaisante au problème d'explosion combinatoire de la génération des tests comme le montrent nos premiers résultats expérimentaux.

De plus, la méthode fournit davantage d'éléments pour l'évaluation de la qualité de suites de tests. Les critères de couverture portant sur le nombre d'états et le nombre de transitions du produit synchronisé sont pertinents. L'abstraction exhibe des états symboliques atteignables du modèle qui sont également des états de l'objectif de test. Les transitions sont la synchronisation des transitions du modèle et des transitions de l'objectif de test.

Ces travaux doivent être poursuivis et approfondis afin d'évaluer l'apport en pratique de la démarche. D'une part, il est nécessaire d'évaluer les performances de la phase de génération de tests abstraits à partir de tests symboliques en utilisant les technologies standard de recherche de préambule et de résolution de contraintes de LTG par exemple. D'autre part, il est nécessaire d'évaluer et d'optimiser la phase de génération d'une abstraction.

Références

[ABC+02] F. Ambert, F. Bouquet, S. Chemin, S. Guenaud, B. Legeard, F. Peureux, N. Vacelet, and M. Utting. BZ-TT : A tool-set for test generation from Z and B using constraint logic programming. In *FATES'02*, pages 105–120, 2002.

[ABM98] P.E. Ammann, P.E. Black, and W. Majurski. Using model checking to generate tests from specifications. In *ICFEM'98*, pages 46–54. IEEE Computer Society Press, 1998.

[Abr96a] J.-R. Abrial. *The B Book*. Cambridge Univ. Press, 1996.

[Abr96b] J.-R. Abrial. Extending B without changing it (for developing distributed systems). In *1st B Conference*, pages 169–190, 1996.

[BBC+06] E. Bernard, F. Bouquet, A. Charbonnier, B. Legeard, F. Peureux, M. Utting, and E. Torreborre. Model-based testing from UML models. In *MBT'06*, volume P-94 of *LNI, Lecture Notes in Informatics*, pages 223–230, 2006.

[BJK+05] M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, and A. Pretschner, editors. *Model-Based Testing of Reactive Systems*, volume 3472 of *LNCS*. Springer, 2005.

[BLP04] F. Bouquet, B. Legeard, and F. Peureux. CLPS-B : A constraint solver to animate a B specification. *STTT journal, Software Tools for Technology Transfer*, 6(2) :143–157, 2004.

[BLS05] M. Barnett, K.R.M. Leino, and W. Schulte. The spec# programming system : An overview. In *CASSIS'04*, volume 3362 of *LNCS*, pages 49–69. Springer, 2005.

[CIVDP07] J. Calamé, N. Ioustinova, and J. Van De Pol. Automatic model-based generation of parameterized test cases using data abstraction. *ENTCS*, 191 :25–48, 2007.

[GGL04] J.-P. Gallois, C. Gaston, and A. Lapitre. AGATHA, un outil de simulation symbolique. In *AFADL'04, session outils*, Besançon, France, 2004.

[GIX04] GIXEL. *Common IAS Platform for eAdministration*, 1.01 premium edition, 2004.

[HLSU02] H.S. Hong, I. Lee, O. Sokolsky, and H. Ural. A temporal logic based theory of test coverage and generation. In *TACAS'02*, pages 327–341. Springer, 2002.

[JJ05] C. Jard and T. Jéron. TGV : theory, principles and algorithms. *STTT, Int. Journal on Software Tools for Technology Transfert*, 7(1) :297–315, 2005.

[JJRZ05] B. Jeannet, T. Jéron, V. Rusu, and E. Zinovieva. Symbolic test selection based on approximate analysis. In *TACAS'05*, volume 3440 of *LNCS*, 2005.

[JL07] E. Jaffuel and B. Legeard. LEIRIOS Test Generator : Automated test generation from B models. In *B'2007, Tool Session*, volume 4355 of *LNCS*, pages 277–280. Springer, 2007.

[JMT08a] J. Julliand, P.-A. Masson, and R. Tissot. Generating security tests in addition to functional tests. In *AST'08*, pages 41–44. ACM Press, 2008.

[JMT08b] J. Julliand, P.-A. Masson, and R. Tissot. Generating tests from B specifications and test purposes. In *ABZ'08*, volume 5328 of *LNCS*, pages 139–152. Springer, 2008.

[LdBMB04] Y. Ledru, L. du Bousquet, O. Maur, and P. Bontron. Filtering TOBIAS combinatorial test suites. In *FASE'04*, volume 2984 of *LNCS*, pages 281–294. Springer, 2004.

[MM02] R. Marlet and C. Mesnil. Demoney : A demonstrative electronic purse – card specification. Technical Report SECSAFE-TL-007, SecSafe project / Trusted Logic S.A., 2002.

[Peu02] F. Peureux. *Génération de tests aux limites à partir de spécifications B en Programmation Logique avec Contraintes Ensemblistes*. PhD thesis, Univ. de Franche-Comté, 2002.

[Rea08] Reactive systems. *Model-Based Testing and Validation with Reactis*, 2008. http ://www.reactive-systems.com.

[RGLG03] N. Rapin, C. Gaston, A. Lapitre, and J.-P. Gallois. Behavioral unfolding of formal specifications based on communicating extended automata. In *ATVA'03, Automated Technology for Verification and Analysis*, 2003.

[Sto07] N. Stouls. *Systèmes de transitions symboliques et hiérarchiques pour la conception et la validation de modèles B raffinés*. PhD thesis, INP Grenoble, 2007.

[TGLG05] A. Touil, C. Gaston, and P. Le Gall. Automatic generation of symbolic test purposes. In *MoDeVa'05, Model design and Validation*, Montego Bay, Jamaica, October 2005.

[Thi03] H.W. Thimbleby. The directed chinese postman problem. *Software : Practice and Experience*, 33(11) :1081–1096, 2003.

[UL06] M. Utting and B. Legeard. *Practical Model-Based Testing*. Elsevier Science, 2006.