Une application de la théorie des jeux au test de conformité [1]

Solofo Ramangalahy* , **Pascale Le Gall**† et **Thierry Jéron**‡

LETI (CEA – Technologies Avancées) DEIN CEA/Saclay F-91191 Gif-sur-Yvette Cedex
Solofo.Ramangalahy@cea.fr
†*LaMI, Université d'Évry Val d'Essone Cours Monseigneur Romero F-91025 Évry Cedex*
Pascale.Legall@lami.univ-evry.fr
‡*IRISA/INRIA Rennes Campus de Beaulieu F-35042 Rennes Cedex*
Thierry.Jeron@irisa.fr

RÉSUMÉ — Une méthode et un algorithme de génération de tests de conformité sont présentés. Ils sont fondés sur l'interprétation des concepts de la recommandation ITU-T « *Formal Methods in Conformance Testing* » dans le cadre de la théorie des jeux. Partant d'une spécification et d'un objectif de test, un jeu est défini. Une stratégie gagnante pour ce jeu définit un testeur pour une classe de modèles d'implantation et une relation de conformité données. Une conséquence de l'utilisation de cette technique est la diminution de la production de verdicts inconclusifs du fait du chaînage de la réexécution d'un test, un testeur étant représenté par un graphe. Une autre conséquence est de donner à ces tests un meilleur contrôle sur l'implantation.
MOT-CLÉS : *FMCT, génération de tests de conformité, objectifs de test, hypothèses de test, équité, théorie des jeux, stratégies gagnantes.*

ABSTRACT — A test generation method and the associated algorithm are proposed. It is based on the interpretation of testing concepts from the ITU-T recommendation « Formal Methods in Conformance Testing », in a game theoretic setting. Starting from a specification and a test purpose, a game is defined. A winning strategy for this game defines a tester for a given implementation model class and an implementation relation. A consequence of the use of this technique is the reduction of the production of inconclusive verdicts because of the chaining of test reexecution, a tester being defined as a graph. Another consequence is that produced test cases have a better control on implementations.
KEY WORDS : *FMCT, conformance testing, automated test generation, test purposes, test hypotheses, fairness, game theory, winning strategies.*

1. INTRODUCTION

Le test de conformité des protocole a été normalisé dans le document « Conformance Testing Methodology and Framework » [ISO91]. Le test de conformité est le problème de relier la spécification d'un système à

[1] Cet article est une version étendue de [RLGJ99] et de [Ram98]

sa réalisation physique. Celle-ci (dénommée IUT, implantation sous test) n'est observable qu'au travers des interactions qu'elle échange avec son environnement. En effet, il est courant qu'une IUT soit fournie à un client seulement par son code exécutable. Cette caractéristique « boîte noire » fait que le test reste essentiel pour la correction d'un système en complément de méthodes telles que la vérification par preuve ou le *model checking*, qui s'attachent au problème de la correction de la spécification.

Un test décrit une suite d'interactions à échanger entre un testeur placé dans l'environnement et l'IUT afin de décider de sa conformité vis-à-vis de sa spécification. En pratique un test est décrit relativement à un *objectif de test* qui décrit (informellement) quel est le but du test. Pour simplifier, le test peut émettre trois types de verdicts : *fail* quand il juge que l'IUT n'est pas conforme vis à vis de la spécification, *pass* quand l'exécution s'est bien passée et que l'objectif du test est atteint, *inconclusif* lorsque rien d'anormal ne s'est passé mais, que du fait du contrôle limité du testeur sur l'IUT, l'objectif de test n'a pas pu être atteint. Lorsque la spécification est formelle (par exemple écrite en LDS, Estelle ou LOTOS) et qu'on se pose la question de générer automatiquement des tests, la notion de conformité doit elle aussi être formalisée. C'est alors une relation mathématique entre la spécification et un modèle formel m_{IUT} de l'implantation physique IUT. Conformément à « l'hypothèse de test », seule l'existence de m_{IUT} est supposée, pas sa donnée puisqu'on se place dans le contexte du test boîte noire.

Cette présentation du test provient de la recommandation ITU-T « Formal Methods in Conformance Testing » [ITU97][2], qui capitalise le travail réalisé par la norme ISO 9646 [ISO91] en tirant parti de celui effectué lors de la normalisation des langages de spécification formels et des avancées de la recherche (voir [CFP96] pour une présentation de la recommandation et de ses enjeux et [KPT] pour une bibliographie recensant les travaux de recherche à l'origine de la norme).

Le cadre du test de conformité étant posé, il reste à fournir les moyens de sa réalisation. Dans cet article nous proposons un algorithme de génération de test de conformité dans le cadre de FMCT. La plupart des autres techniques de génération de tests produisent des tests sous forme d'arbre ou de séquences. L'originalité de notre méthode est la présentation des cas de tests sous forme de graphe où une suite de tests est codée dans un automate qui représente un « chaînage » des cas de tests individuels[3]. L'intérêt est de ne pas réinitialiser un système pour relancer un même test lorsque l'exécution pourrait conduire à un verdict inconclusif, donnant ainsi un test plus compact, et une suppression de l'émission du verdict inconclusif. Ce résultat découle d'une vision du test en terme de théorie des jeux : un testeur est une stratégie gagnante réalisée par un automate. Une conséquence de l'utilisation de la théorie des jeux est que les tests produits permettent un meilleur contrôle de l'IUT en favorisant le contrôle du testeur sur l'observation des actions de l'IUT. Ce travail peut être en partie comparé aux travaux récents sur l'outil TGV [JM99] qui propose un algorithme fondé sur le model-checking à la volée qui génère aussi des tests sous forme de graphe et qui permet non pas de supprimer mais de réduire les verdicts inconclusifs.

Dans cet article, nous ne traitons pas du problème de la couverture des suites de tests qui, à juste titre, intéresse les praticiens du test. Nous fournissons une réponse au problème de la correction car les tests produits ne peuvent pas rejeter d'IUT conforme. Nous donnons également un résultat d'exhaustivité au sens où, sous certaines hypothèses d'équité, toute IUT non conforme peut être rejetée par un test. Ce résultat, bien que nécessaire à la complétude de la méthode, est purement théorique puisqu'il nécessite d'écrire un objectif qui permette de dériver ce test. De notre point de vue, la couverture doit donc être traitée en amont comme cela est fait par exemple dans l'outil TestComposer de Vérilog [AJG99] bâti à partir de TVéda et TGV. Il s'agit de produire de façon automatique ou semi-automatique des objectifs qui couvrent la spécification pour un critère donné. Cet aspect dépasse le cadre du travail présenté.

La suite de l'article rappelle quelques éléments de la recommandation FMCT et présente le type de jeu utilisé et son lien avec le test de conformité. Ce lien établi, on raisonne en termes de jeux pour calculer

[2]Cette recommandation est en cours de normalisation à l'ISO.

[3]Nous utilisons les termes testeur et cas de test pour désigner un même objet, selon qu'il soit donné sous forme de graphes ou bien d'arbre ou de séquence.

les stratégies gagnantes qui seront les testeurs. Dans la dernière partie, nous étudions la correction et l'exhaustivité des tests générés conformément au cadre fourni par la recommandation FMCT. Un exemple simple sert de fil conducteur à la présentation de la méthode.

2. JEUX ET TEST DE CONFORMITÉ

2.1. Test de conformité

Nous rappelons succinctement quelques éléments de la recommandation FMCT [ITU97]. L'utilisation de la recommandation FMCT passe par le choix des formalismes utilisés pour décrire les différents objets que l'on manipule. Bien que la recommandation soit indépendante de ces formalismes, les automates sont généralement le formalisme choisi car ils sont sous-jacents (dans différentes variantes) à la sémantique des langages comme LDS [Z.188], Estelle [ISO85] ou LOTOS [LOT85].

La conformité est définie comme une relation R entre deux classes de modèles : les modèles d'implantation MODS et les modèles de spécification SPECS. L'*hypothèse de test* suppose qu'une implantation IUT peut être modélisée par un élément m_{IUT} de MODS. Ceci est nécessaire pour définir la conformité. Mais il n'en reste pas moins que l'IUT et donc son modèle supposé sont inconnus. Étant données une spécification $S \in \mathsf{SPECS}$ et une relation d'implantation $R \subseteq \mathsf{MODS} \times \mathsf{SPECS}$, le problème du test de conformité est de vérifier que le modèle m_{IUT} appartient à l'ensemble des modèles d'implantation qui sont conformes à S $(\{m \in \mathsf{MODS} \mid (m, S) \in R\})$. La résolution de ce problème passe par la génération d'un ensemble de cas de tests (appartenant à un ensemble TESTS) modélisés aussi par des automates. L'exécution de ces tests est alors modélisée par le produit parallèle d'un cas de test t et de m_{IUT}. Les traces de ce produit parallèle sont les observations qui peuvent être faites de l'exécution des tests. De ces observations, des verdicts peuvent éventuellement être émis à propos de la conformité.

L'ensemble des modèles conformes n'est pas construit explicitement et on ne dispose pas non plus du modèle m_{IUT}. Les expériences de test permettent uniquement l'observation de certaines traces de m_{IUT}. Cela reste suffisant pour détecter la non conformité (le verdict émis est alors *fail*, $(m_{IUT}, S) \notin R$). Du fait de la nature purement « boîte noire » du test, l'émission d'un verdict global de conformité est en pratique rarement possible.

Nous choisissons classiquement comme instances de SPECS, MODS, TESTS les automates à entrées et sorties et comme instance de R l'égalité des sorties possibles après une trace appartenant à la spécification et au modèle d'implantation. La définition des automates à entrées et sorties est la suivante :

Définition 1 (automate à entrées et sorties, IOSM).
Un automate à entrées et sorties est un quadruplet $\langle S, L, T, s_0 \rangle$ où :

- *S est un ensemble fini non vide d'états ;*
- *L est un alphabet fini non vide d'interactions. Les symboles $?, !$ et τ sont des symboles additionnels ne faisant pas partie de L ;*
- *$T \subseteq \Big(S \times ((\{?, !\} \cdot L) \cup \{\tau\}) \times S\Big)$ est la relation de transition. Chaque élément de T est une transition entre un état de départ et un état d'arrivée. Cette transition est soit associée à une action observable (émission ! ou réception ? d'une interaction, soit une transition interne τ).*
- *s_0 est l'état initial de l'automate.*

Exemple 1
Dans l'exemple de la figure 1, $S = \{1, 2, 3, 4, 5\}$ et $L = \{a, b, c, d, e, f, g, h\}$. Cet exemple illustratif peut être vu comme une spécification d'une cabine téléphonique : lorsque l'utilisateur introduit sa carte (`?a`), la cabine peut réagir en indiquant que la carte n'est pas valide – pas assez d'unités, carte endommagée, etc ... – ou bien elle peut l'accepter (`!b` resp. `!c`). Dans ce dernier cas, l'utilisateur de la cabine peut

composer son numéro (`?f`). Si il attend trop longtemps, la carte est restituée (`!e`). Si son correspondant peut être joint, la communication téléphonique peut être établie (`!h`), dans le cas contraire, il faudra qu'il réessaye de composer ce numéro ou un autre (`!g`). L'utilisateur peut raccrocher (`?d`) ou la cabine peut stopper la communication pour insuffisance de crédit ou tout autre problème (second `!b`).

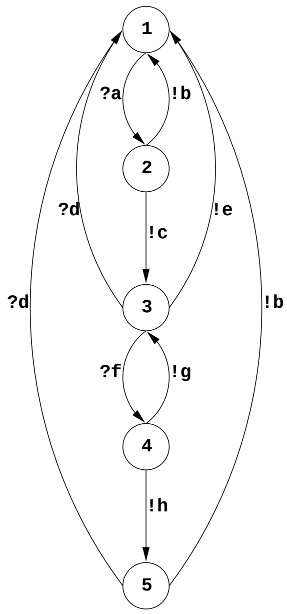

Figure 1. *un automate à entrées et sorties*

Concernant les éléments de MODS, une condition supplémentaire est que toute réception doit être possible dans chaque état ($\forall a \in L, \forall s_1 \in S, \exists s_2 (s_1, ?a, s_2) \in E$). Cette condition signifie qu'une implantation ne peut pas refuser un message externe, même si elle ne réagit pas à ce message ($s_1 = s_2$). Cette condition est appelée complétude dans [Pha94] ou *input-enabling* dans [Tre96]. Nous supposons que les automates de SPECS ne comportent pas d'action interne et sont déterministes au sens usuel de la théorie des automates, cette hypothèse ne diminue pas la généralité de l'approche dans le cadre du test : le pouvoir du test étant limité à l'observation de traces ([Pha94] chap. 2), on peut considérer les automates modulo équivalence de trace. De plus, ceci peut être fait efficacement d'un point de vue pratique [JM97]. Il est aussi nécessaire de savoir si l'absence d'émission est observable pour l'émission d'un verdict en temps fini ([Pha94], chap. 2, [Tre92] p. 117). Nous supposons pour cela que les blocages sont des actions observables des implantations. On peut considérer trois types de blocages : les deadlocks (états sans successeurs), les livelocks (états dans lesquels il existe un cycle d'action interne), les blocages de sorties (les états desquels ne part aucune transition étiquetée par une action interne ou une émission) Enfin, nous supposons que les automates vus en tant que graphes sont fortement connexes. Ceci afin d'éviter l'hypothèse plus restrictive d'un *reset* permettant de ramener un système dans son état initial pour relancer un test.

Moyennant ces hypothèses, l'exécution d'un test ne s'arrête que du fait du testeur.

La relation d'implantation définissant la conformité est la suivante :

Définition 2 (relation d'implantation).
Soient I et S deux IOSMs (le modèle d'implantation et la spécification), $Tr(S)$ dénote l'ensemble des traces de S et $\mathcal{O}(\sigma, S)$ l'ensemble des interactions de sortie possibles pour S après la trace σ. La relation

d'implantation R est définie par :

$$R(I, S) \text{ ssi } \forall \sigma \in Tr(S) : \sigma \in Tr(I) \Rightarrow \mathcal{O}(\sigma, I) = \mathcal{O}(\sigma, S).$$

Il est à noter que d'autres relations d'implantation sont possibles (voir par exemple [Pha94] pour différentes relations). On peut par exemple être plus laxiste et n'exiger que l'inclusion des sorties $\mathcal{O}(\sigma, I) \subseteq \mathcal{O}(\sigma, S)$ au lieu de l'égalité. L'égalité rajoute à l'inclusion le fait qu'une IUT pourra être rejetée par un test si après avoir exécuté plusieurs fois le test, elle a montré tous ses comportements (sous une hypothèse d'équité) et que des sorties prévues de la spécification n'ont pas été constatées sur l'IUT.

Un problème important pour la génération de tests est celui de la sélection. Il s'agit de réduire la taille des suites de test pour diminuer l'effort de test. Un moyen est l'utilisation d'objectifs de test. Il provient de la pratique : les experts écrivent un commentaire dans les cas de tests écrits en notation TTCN (*Tree and Tabular Combined Notation*) [ISO91] afin d'exprimer l'intention qu'ils ont en écrivant un test particulier. L'objectif de test est ici formalisé et utilisé afin de limiter la recherche dans l'espace des états, en se restreignant à un sous-ensemble des comportements du système.

Définition 3 (objectif de test).
Un objectif de test TP associé à une spécification S est un sous-arbre de S.

Pour des raisons de simplicité, l'objectif de test utilisé ici est moins général que celui de [FJJV96]. L'approche se généralise néanmoins à des objectifs de test similaires (voir [Ram99] pour plus de détails).

Nous reprenons la condition de contrôlabilité de [FJJV96] qui exprime qu'il est inutile de rajouter de l'indéterminisme dans le processus de test : lorsque le testeur envoie un message à l'implantation, son choix est unique. Cette condition est donc ajoutée aux objectifs de test.

Exemple 2
L'objectif de test de la figure 2 a pour signification informelle un scénario d'appel téléphonique réussi : l'utilisateur introduit sa carte téléphonique (`?a`), la cabine l'accepte (`!c`), l'utilisateur compose son numéro (`?f`) et peut joindre son correspondant (`!h`). L'arbre est ici réduit à une branche, mais la méthode prend en compte les branchements éventuels.

2.2. Jeux

Nous établissons dans ce paragraphe un lien entre jeux et test de conformité. Pour cette présentation, nous nous restreignons à ne considérer que les jeux entrant dans la catégorie des jeux comme les échecs (dénommés « jeux à deux joueurs à information parfaite »). Nous présentons ces jeux informellement avec l'intuition correspondante dans le cadre du test.

Deux joueurs déplacent alternativement une pièce sur un échiquier (l'implantation et l'environnement s'échangent des messages). La position des différentes pièces sur l'échiquier définit une configuration (la valeur des variables – variables de données, de contrôle, états des files d'attentes ... – définit l'état du système). Bouger une pièce sur l'échiquier suivant les règles conduit à une autre configuration choisie parmi plusieurs possibles (l'échange de message suivant les transitions change l'état du système). Nous pouvons alors voir le jeu comme un graphe biparti[4] fini où les sommets représentent les configurations et les arêtes les différents mouvements possibles. Les sommets ont traditionellement deux couleurs possibles : rouge et noir (le joueur rouge représente l'implantation et le joueur noir l'environnement). Le joueur associé à la couleur du sommet courant choisit la configuration suivante (l'IUT ou le testeur émettent un message, on rappelle que l'interaction est synchrone). Une partie est une séquence infinie de mouvements (le système

[4]Un graphe biparti est un graphe dont l'ensemble des sommets peut être partitionné en deux classes, de sorte que deux sommets de la même classe ne soient jamais adjacents.

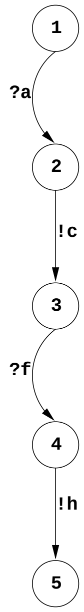

Figure 2. *un objectif de test pour l'IOSM de la figure 1*

est un système réactif dont les exécutions sont classiquement vues comme des mots infinis). Une partie est gagnante pour le joueur noir si elle appartient à un ensemble prédéfini de parties gagnantes (un test est réussi si on a pu observer un comportement spécifié par l'objectif de test ou si on a détecté une erreur).

Cette vision intuitive d'un système réactif comme un jeu se retrouve dans l'étude de problèmes connexes comme celui du *deadlock* [Dev77], de la synthèse de programmes concurrents [NYY92] ou du *model checking* [Sti96].

Les définitions en termes de jeux correspondant à cette description intuitive sont :

Définition 4 (Jeu).
Un jeu est un graphe biparti dirigé fini $G = (V, V_B, V_R, E)$ où :

- *V est l'ensemble des sommets partitionés en sommets noirs (V_B) et rouges (V_R) ;*
- *$E \subseteq V \times V$ est un ensemble d'arêtes tel que $\forall v \in V. \exists v' \in V.(v, v') \in E$.*

Une partie est un mot infini de V^∞, $w = s_1 s_2 \dots$ tel que $\forall i, (s_i, s_{i+1}) \in E$ ($s_1 \in V_B$ par convention).
Un ensemble de parties gagnantes étant fixé, une partie est gagnante pour le joueur noir si elle appartient à l'ensemble des parties gagnantes (sinon, elle est gagnante pour le joueur rouge).
Une stratégie pour le joueur noir est une fonction $f : (V_B \cdot V_R)^* \cdot V_B \to V$. Une stratégie pour le joueur noir est gagnante si il gagne en suivant cette stratégie (ie $\forall k, s_{2k+1} = f(s_1 \dots s_{2k})$ et il existe un indice k tel que s_{2k+1} appartient aux configurations gagnantes).

Donnons maintenant l'interprétation du test en termes de jeu.

Définition 5 (Jeu associé à une spécification et un objectif de test).
Soit $\mathcal{S} = \langle S, L, T, s_0 \rangle$ une spécification et TP un objectif de test. Le jeu $G = (V, V_B, V_R, E)$ associé à S et TP est le plus petit graphe biparti tel que :

1. *$V \supseteq S \cup \{\perp\}$, $V_R \supseteq \{s \in S | \forall (s, t, s') \in T, \exists a \in L.t = !a\}$, $V_B = V \backslash V_R$;*
2. *$\forall v \in V_R : (v, \perp) \in E$;*

3. *$\forall (v, a, v') \in T : (v, v') \in E$ si v et v' ne sont pas de même couleur, sinon un nouveau nœud intermédiaire de couleur opposé est intercalé ;*
4. *les parties gagnantes sont les parties dans $s_0 V^* \perp V^\omega$ et $s_0 V^* w V^* s_0 V^\omega$, où w est une branche de TP.*

L'état $\perp$ représente un état où la non conformité a été détectée. L'ensemble des sommets rouges V_R contient l'ensemble des états où seule une sortie est possible. Ce choix reflète la convention qu'un testeur aura la priorité s'il se présente un choix entre une émission et une réception dans l'exécution parallèle du testeur et de l'implantation. De tout état de V_R, on peut atteindre l'état $\perp$. Intuitivement, cela signifie que sur émission de l'IUT, le testeur peut déclarer la non-conformité. Les parties gagnantes sont celles où la non-conformité est détectée en atteignant l'état $\perp$ et celles où une branche de l'objectif a été parcourue et on est revenu dans l'état initial.

Exemple 3

Le jeu associé à l'automate de l'exemple 1 a) est représenté figure 3. Les sommets noirs sont représentés par des cercles, les rouges par des carrés. Les sommets additionnels 6, 7, 8 et 9 sont utilisés pour obtenir un graphe biparti, et le sommet 10, en plus de l'hypothèse de forte connexité sur S, assure que les parties sont toujours infinies. L'objectif de test est celui de la figure 2. Les parties gagnantes sont donc les éléments de $1V^* \perp V^\omega$ et $1V^* 12345 V^* 1 V^\omega$.

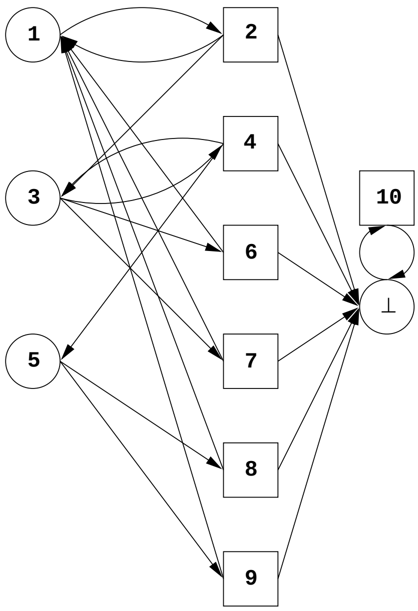

Figure 3. *jeu associé à l'automate de la figure 1*

La formalisation du test en termes de jeu nous permet désormais de raisonner uniquement en termes de jeux.

3. CONSTRUCTION DES STRATÉGIES GAGNANTES

Nous avons défini précédement un jeu entre spécification et objectif de test. Reste maintenant à définir les stratégies gagnantes pour ce jeu qui vont définir les testeurs. Les stratégies gagnantes permettent au joueur noir – l'environnement – de choisir les messages à envoyer pour, soit satisfaire l'objectif de test soit détecter la non conformité ($\perp$).

Dans un premier temps, nous allons simplifier le jeu pour permettre une expression simple des stratégies gagnantes pour revenir ensuite au jeu original.

3.1. Jeu d'atteignabilité

Nous résolvons dans un premier temps le problème de la construction d'une stratégie gagnante pour un jeu simple que nous appellerons le jeu d'atteignabilité pour le sommet t, où les parties gagnantes sont celles de $s_0V^*tV^\omega,\ t \in V$. Cette construction est une extension d'une construction classique pour le jeu appelé *guarantee game* dans [Tho94], dont notre présentation s'inspire. Le but pour le joueur noir est d'atteindre le sommet t, quels que soient les mouvements choisis par le joueur rouge. Pour cela, nous construisons une suite croissante d'ensembles de sommets $\mathcal{W}_i^j$.

La construction de cette stratégie est le cœur de la méthode présentée ici avec la transformation du problème de test en un problème de jeu du paragraphe précédent. Nous reprenons l'exemple du jeu d'échec pour en expliquer l'intuition. Lorsqu'un joueur (l'IUT ou le testeur) conçoit sa stratégie, il ne prend en compte qu'*une* possibilité lorsqu'il joue (il contrôle par une émission), mais il doit prendre en compte *tous* les coups possibles de son adversaire en réponse au sien (il observe et doit considérer toutes les réceptions possibles pour choisir sa prochaine émission). Ceci se retrouve dans l'alternation de quantifications $\exists$ et $\forall$ dans la définition des ensembles $\mathcal{W}_{i+1}^j$. Une caractéristique particulière du jeu présenté ici est que seul le testeur recherche une stratégie gagnante. L'adversaire (l'implantation) jouera selon une stratégie particulière (« équitable » dans un sens défini plus loin) qui fait que le testeur peut toujours trouver une stratégie gagnante. Ceci explique que l'on continue à construire la stratégie pour le joueur noir en $\mathcal{W}_0^{j+1}$, bien qu'une stratégie optimale de l'adversaire serait gagnante à ce point.

La suite $\mathcal{W}_i^j$ est formellement définie récursivement en partant du but et en propageant en arrière le niveau de contrôle du testeur :

1. $$\mathcal{W}_0^0 = \{t\}$$
   où t est le sommet à atteindre.
2. $$\mathcal{W}_{i+1}^j = \mathcal{W}_i^j \cup \{p \in V_B \mid \exists q \in \mathcal{W}_i^j : (p,q) \in E\}$$
   le testeur garde le même niveau de contrôle en restant dans la sous-suite $\mathcal{W}_i^j$ avec j fixe. Il le réalise soit quand il joue en choisissant une entrée de S, soit quand l'IUT joue dans le cas où n'importe quelle sortie de S mène à un état dans l'ensemble $\mathcal{W}_i^j$).
3. Pour j fixé, on note $k_j = \min\{i \mid \mathcal{W}_{i+1}^j = \mathcal{W}_i^j\}$ et on définit :
   $$\begin{aligned} \mathcal{W}_0^{j+1} = \quad & \mathcal{W}_{k_j}^j \cup \{p \in V_R \mid \forall q \in V_B : (p,q) \in E \Rightarrow q \in \mathcal{W}_{k_j}^j\} \\ & \cup \{p \in V_R \backslash \mathcal{W}_{k_j}^j \mid \exists q, r \in V_B : (p,q),(p,r) \in E \wedge q \in \mathcal{W}_{k_j}^j\} \\ & \cup \{p \in V_R \backslash \mathcal{W}_{k_j}^j \mid \exists q, r \in V_B : (p,q),(p,r) \in E \wedge q \in \mathcal{W}_{k_j}^j \wedge r \notin \mathcal{W}_{k_j}^j\} \end{aligned}$$
   Ainsi, $\mathcal{W}_{k_j}^j$ est le point fixe de la sous-suite croissante $\mathcal{W}_i^j$ pour j fixe. Le testeur perd un peu plus de contrôle en passant à une nouvelle sous-suite $(\mathcal{W}_i^{j+1})_i$. Cette sous-suite démarre à partir des états où S permet par une émission d'aller soit dans un état de $\mathcal{W}_{k_j}^j$, soit d'en sortir.

Ces ensembles forment une suite $\mathcal{W}_0^0, \ldots, \mathcal{W}_{k_0}^0, \mathcal{W}_0^1, \ldots, \mathcal{W}_{k_1}^1, \ldots, \mathcal{W}_0^l, \ldots, \mathcal{W}_{k_l}^l$. L'indice l est le premier indice pour lequel l'un des ensembles $\mathcal{W}_i^l$ (pour $i \in [0, k_l]$) coïncide avec l'ensemble des sommets du jeu (voir lemme ci-dessous). Ils sont utilisés pour définir un ordre sur les sommets par leur rang. Le rang d'un sommet p est

$$rank(p) = \left(\min\{i | p \in \mathcal{W}_i^j\}, \min\{j \mid p \in \mathcal{W}_i^j\}\right).$$

La stratégie pour le joueur noir est alors de choisir un sommet rouge inférieur par rapport à l'ordre lexicographique gauche [5]. Cette stratégie a pour but d'augmenter pas à pas le contrôle du testeur sur l'IUT pour l'amener dans l'état à atteindre t.

Le lemme suivant montre que cette stratégie est bien définie :

Lemme 1:
Le rang est défini sur tous les sommets du jeu et il est défini de manière unique :

1. $\lim_{i,j} \mathcal{W}_i^j = V \backslash \{\perp\}$ *(V est l'ensemble des sommets du jeu) ;*
2. *Les ensembles* $\{\mathcal{W}_{i+1}^j \backslash \mathcal{W}_i^j \mid i \in [0, k_j], j \in [0, l]\}$ *et* $\{\mathcal{W}_{k_i}^i \mid i \in [0, l]\}$ *forment une partition de* $V \backslash \{\perp\}$.

PREUVE:

1. Par construction du jeu, le jeu vu en tant que graphe hérite de la forte connexité du graphe initial (l'automate à la base de la définition du jeu). Plus précisément, $V \backslash \{\perp\}$ est fortement connexe.
2. Soit s_n un élément de $V \backslash \{\perp\}$. $V \backslash \{\perp\}$ étant une composante fortement connexe de G, il existe un chemin $\omega = s_n s_{n-1} \dots s_1 s_0$ de s_n à s_0. Chacun des s_i appartient à un des ensembles $\mathcal{W}_l^k$: s_n appartient à $\mathcal{W}_0^0$. Si $s_i \in \mathcal{W}_m^k$ et $s_{i+1} \in V_B$ alors $s_{i+1} \in \mathcal{W}_{m+1}^k$ ou bien $s_{i+1} \in V_R$ et toutes ses arêtes sortantes vont dans $\mathcal{W}_m^k$; sinon il appartient à $\mathcal{W}_0^{k+1}$.
3. Les ensembles sont disjoints par construction et couvrent l'ensemble des sommets de S $\{\perp\}$. ■

Bien que l'implantation soit vue comme une boîte noire, plusieurs hypothèses sont utiles à l'amélioration de la génération de tests. La notion d'hypothèse de test vient à l'origine de [Ber91] dans le contexte des spécifications algébriques et a été étudiée ensuite dans [Pha94] pour le test de conformité dans le cadre des automates à entrées et sorties. L'effet des hypothèses de test est de réduire l'ensemble des implantations conformes à une spécification. Dans le contexte habituel des jeux, le problème est de trouver les ensembles gagnants de chaque joueur (l'ensemble des sommets à partir desquels le joueur a une stratégie gagnante). Ici nous voulons voir le joueur noir gagner, éventuellement en explicitant des contraintes additionnelles sur les mouvements du joueur rouge (ce qui revient à mettre à plat les conditions de testabilité, i.e. les conditions à la fois sur la spécification et sur l'implantation susceptibles de garantir la possibilité d'établir la non conformité de l'IUT par le test). Nous supposerons que le joueur rouge joue équitablement. Intuitivement, cela signifie que si une implantation peut exhiber des comportements différents de façon non-déterministe elle finira par tous les exhiber. Dans ce cas, l'environnement (qui lui ne joue pas équitablement) aura toujours une stratégie gagnante.

Définition 6 (parties équitables).
Soient $\pi = s_1 \dots s_n \dots$ *une partie, et* $Inf(\pi) = \{s \mid s$ *apparaît infiniment dans* $\pi\}$. *Soit*

$$\{(r_i, B_i = \{b_{i_1}, \dots, b_{i_j}\}) \mid (r_i, b_{i_k}) \in E\} \subset V_R \times \mathcal{P}(V_B)$$

l'ensemble des couples composés d'un sommet rouge r_i *et de l'ensemble des sommets noirs* b_{i_k} *adjacents à* r_i.

La partie π *est équitable si* $\forall i : r_i \in Inf(\pi) \Rightarrow B_i \subset Inf(\pi)$.

[5] $i < i' \Rightarrow (i,j) < (i',j)$ et $\forall i, j < j' \Rightarrow (i,j) < (i,j')$

Cette notion d'équité est standard. Elle stipule que si l'implantation a infiniment souvent le choix entre plusieurs actions d'émission, elle exécutera infiniment souvent chacune des actions. Une stratégie équitable est une stratégie qui ne peut pas conduire à des parties non équitables si le joueur joue selon cette stratégie. Moyennant cette hypothèse, nous avons le résultat suivant :

Proposition 1:
La stratégie « faire décroître le rang » est une stratégie gagnante pour le joueur noir pour le jeu de l'atteignabilité, dans le cas où le joueur rouge joue suivant une stratégie équitable.

PREUVE:
Par contradiction. Soit π une partie gagnante pour le joueur rouge. Soient $r_1, \ldots, r_n$ les éléments minimaux de $Inf(\pi) \cap V_R$ par rapport à leur rang suivant l'ordre lexicographique gauche. Ces sommets rouges appartiennent à un ensemble $\mathcal{W}_0^i$. Par définition de l'équité, chacun des r_i étant connecté à au moins un b de $\mathcal{W}_{k_{i-1}}^{i-1}$, il y aura au moins un b dans $Inf(\pi) \cap \mathcal{W}_{k_{i-1}}^{i-1}$. Il y a donc un sommet rouge r (celui choisi par la stratégie du joueur noir en b) tel que $r \in Inf(\pi)$. Ceci contredit la minimalité de l'ensemble des r_i. ■

En terme de test, la proposition précédente signifie que si l'IUT joue équitablement, le testeur peut contrôler l'IUT pour l'amener vers l'état t, en tenant compte du non-déterminisme possible de l'IUT en réponse aux stimuli du testeur.

Dans la suite, nous noterons $\mathcal{W}_i^j(t)$ l'ensemble des sommets de rang (i, j) relativement au jeu d'atteignabilité pour le sommet t.

3.2. Jeu de conformité

Nous examinons maintenant le cas général. Nous rappelons que les parties gagnantes sont les éléments de $s_0V^*\bot V^\omega$ et $s_0V^*wV^*s_0V^\omega$ (pour une branche w de l'objectif de test). La stratégie est construite à partir de stratégies définies dans le paragraphe précédent. Comme nous allons en utiliser plusieurs, nous ajoutons comme paramètre aux ensembles $\mathcal{W}_i^j$ le but à atteindre. Nous divisons le jeu en trois parties correspondant au préambule du test, au corps du test et au postambule.

1. **préambule :** partant de l'état initial, le but est d'atteindre la racine tp_0 de l'objectif de test TP (on reconnaît alors le recours au jeu d'atteignabilité précédemment défini avec pour sommet à atteindre le sommet tp_0) ;
2. **corps du test :** une fois que tp_0 a été atteint, la stratégie pour le joueur noir est de rester sur une branche de l'objectif de test ou, si le joueur rouge décide de jouer à l'extérieur de l'objectif de test, de revenir à tp_0 (il s'agit ici d'une combinaison d'un jeu trivial "atteindre un sommet en un coup" avec le jeu d'atteignabilité du sommet initial tp_0, le second n'étant invoqué que lorsque le gain du premier n'est pas réalisé) ;
3. **postambule :** une fois qu'une feuille de l'objectif de test a été atteinte, la stratégie pour le joueur noir est de revenir à l'état initial s_0 (là encore, il s'agit d'un jeu d'atteignabilité).

Comme nous venons de le présenter intuitivement, le jeu de conformité peut être décomposé en plusieurs jeux d'atteignabilité. Afin de repérer les différents jeux d'atteignabilité mis en jeu, nous allons annoter les sommets parcourus par des identificateurs $(preamble, test\ body, postamble, end)$ stipulant quelle partie du jeu est concernée.

Plus précisément, la stratégie est définie par un graphe dirigé $\mathsf{G} = (\mathsf{Q}, \mathsf{E})$, où :

- $\mathsf{Q} = V \times \{preamble, test\ body, postamble, end\}$;
- E est défini par les arêtes suivantes :
  - Pour les sommets noirs ($q \in V_B$, $q' \in V$ et $(q, q') \in E$), les arêtes correspondent aux actions de faire decroître le rang et de changer de stratégie lorsque l'on atteint tp_0 ou une feuille de TP :

- $(q, preamble) \to (q', preamble)$ si $q \in \mathcal{W}_i^j(tp_0)$, $q' \in \mathcal{W}_{i-1}^j(tp_0)$, et $q' \neq tp_0$

  $(q, preamble) \to (tp_0, test\ body)$ if $(q, tp_0) \in E$
- $(q, test\ body) \to (q', test\ body)$ si (q, q') est une arête de TP

  $(q, test\ body) \to (q', postamble)$ si q' est une feuille de TP
- $(q, postamble) \to (q', postamble)$ si $q \in \mathcal{W}_i^j(s_0)$, $q' \in \mathcal{W}_{i-1}^j(s_0)$

  $(q, postamble) \to \begin{cases} (s_0, preamble) \\ (s_0, end) \end{cases}$ si $(q, s_0) \in E$

- Pour les sommets rouges ($q \in V_R$ et $(q, q') \in E$), les arêtes correspondent aux changement de stratégie et à la détection de la non conformité :
  - $(q, preamble) \to (q', preamble)$

    $(q, preamble) \to (tp_0, test\ body)$ si $(q, tp_0) \in E$
  - $(q, test\ body) \to (q', test\ body)$

    $(q, test\ body) \to (q', postamble)$ si q' est une feuille de TP

    $(q, test\ body) \to (q', preamble)$ si $\exists q''.(q, q'') \in TP$
  - $(q, postamble) \to (q', postamble)$

    $(q, postamble) \to \begin{cases} (s_0, preamble) \\ (s_0, end) \end{cases}$ si $(q, s_0) \in E$
  - $\left.\begin{array}{l} (q, preamble) \\ (q, test\ body) \\ (q, postamble) \end{array}\right\} \to (\perp, end)$

Cette stratégie définit le testeur associé à l'objectif de test TP. Le graphe est complété en un IOSM par ajout d'étiquettes sur les arêtes correspondant à celles de la spécification. Les états intermédiaires rajoutés pour disposer d'un graphe biparti sont supprimés. Les arêtes conduisant à $\perp$ correspondent à toutes les sorties qui ne sont pas encore utilisées à un sommet [6].

Exemple 4

La figure 4 montre un testeur pour la spécification de l'exemple 1, l'objectif de test $?a!c?f!h$ de la figure 2 et la relation d'implantation R. Le non-déterminisme pour les états 2, 4 et 5 est à noter. C'est uniquement une facilité de notation : pour émettre un verdict et à cause du non-déterminisme, le même test devra être conduit plusieurs fois[7]. Après avoir exécuté plusieurs fois le test, l'expérience est stoppée et le verdict émis. Le choix non-déterministe à l'état 5 est donc résolu par d'abord un retour à l'état 1 puis un arrêt sur le second nœud 1. Le choix non-déterministe dans l'état 2 (resp. 4) est résolu en retournant d'abord à l'état 1 (resp. 3) puis en stoppant sur $\perp$ en émettant un verdict *fail*. Ce non-déterminisme peut être levé en développant plusieurs copies de ce graphe ou en ajoutant une mémoire au testeur.

Ce paragraphe a montré comment, à partir de la construction basique d'une stratégie gagnante pour un jeu d'atteignabilité, on pouvait étendre la construction pour des jeux ayant des parties gagnantes définies par un objectif de test.

Pour des raisons de simplicité, l'objectif de test a été défini comme un sous-arbre de la spécification. C'est une restriction par rapport à la définition utilisée dans [FJJV96] où l'objectif de test décrit un enchaînement d'actions qui ne sont pas nécessairement consécutives. On peut cependant étendre la démarche pour prendre en compte ce cas en calculant une stratégie d'atteignabilité entre les actions non consécutives.

[6] ceci correspond à l'action *?otherwise* produisant le verdict *fail* dans TTCN

[7] le « plusieurs fois » sera discuté dans le paragraphe 5.

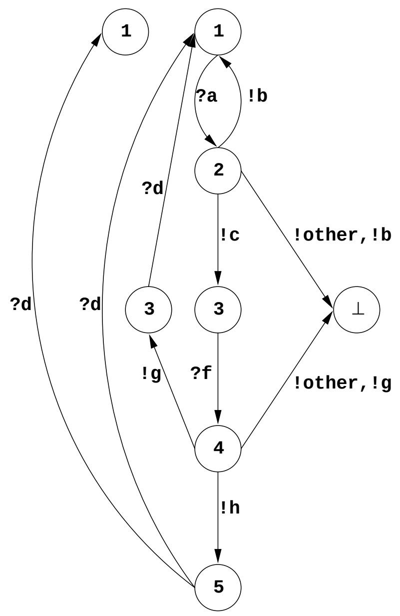

Figure 4. *cas de test pour l'objectif de test de la figure 2*

4. QUELQUES ASPECTS DE COMPLEXITÉ

Dans ce paragraphe, nous examinons quelques aspects de complexité en terme de taille de la stratégie (donc du testeur) et en terme de temps d'exécution par rapport au nombre de sommets du jeu.

Proposition 2:
La taille du testeur, définie par la stratégie $\mathsf{G} = (\mathsf{Q}, \mathsf{E})$ *pour le jeu de conformité est linéaire par rapport à la taille du jeu et peut être construite en temps linéaire.*

PREUVE:

1. La stratégie est construite à partir de la stratégie définie dans le paragraphe 3.1. Chacune de ces stratégies étant un sous-graphe du jeu, la taille finale de la stratégie est linéaire.
2. chaque stratégie est construite en n pas. Comme les ensembles construits dans le paragraphe 3.1 forment une partition de l'ensemble des sommets, il y a moins de pas que de sommets. Le temps pris pour construire la stratégie est linéaire. ■

Proposition 3:
Soit $f : \mathbb{N} \to \mathbb{N}$ *une fonction entière. Il existe un jeu d'atteignabilité et une stratégie équitable pour le joueur rouge tels que la longueur d'une partie est supérieure à* $\prod_{i=1}^{n} f(i)$ *où* n *est la taille du jeu.*

PREUVE:

Le jeu est défini par $V_B = b_0, \dots b_{n+1}$, $V_R = r_1, \dots, r_n$, $E = \{(b_i, r_i) \mid i \in [0,n]\} \cup \{(r_i, b_{i+1}) \mid i \in [0,n]\} \cup \{(r_i, b_0) \mid i \in [0,n]\}$ (cf. figure 5). Les parties gagnantes sont les éléments de $b_0 V^* b_{n+1} V^\omega$ (la longueur est comptée jusqu'à ce que le jeton atteigne b_{n+1}). La stratégie pour le joueur rouge au sommet r_i est de jouer successivement $f(i)$ fois b_0 puis b_{i+1}. ■

Cette dernière propriété montre que le temps d'exécution d'un test, bien que fini, peut être arbitrairement long. Cette durée provient du nombre l apparaissant dans les ensembles définissant la stratégie pour le jeu

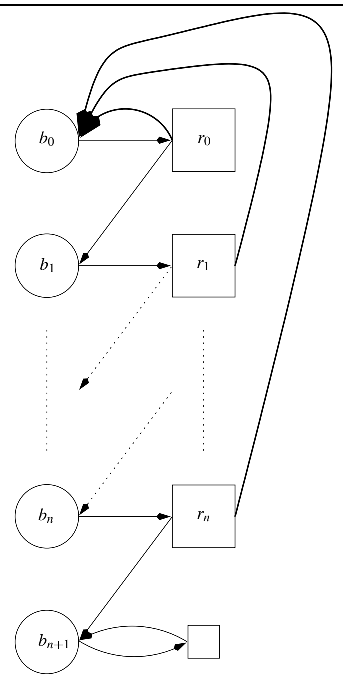

Figure 5. *Jeu de la preuve*

d'atteignabilité (paragraphe 3.1). Ce nombre est une mesure du degré de non-déterminisme de la spécification par rapport à un objectif de test. Pour $l = 1$ le temps d'exécution est linéaire. La complexité provenant du non-déterminisme de la spécification est souvent implicite dans les approches où les testeurs sont des arbres au lieu de graphes : un test doit être appliqué un certain nombre de fois pour obtenir un verdict non inconclusif. Nous avons montré ici que ce nombre pouvait être arbitrairement grand sans hypothèses supplémentaires sur l'implantation. Dans le paragraphe suivant, nous étudierons certaines de ces hypothèses qui réduisent la longueur d'une partie.

5. PROPRIÉTÉS DES TESTEURS GÉNÉRÉS

Ayant raisonné jusqu'ici en termes de jeux, nous revenons à l'étude des testeurs générés du point de vue purement test. Nous rappelons quelques aspects de la recommandation FMCT concernant l'exécution des tests.

Ici, l'exécution des tests consiste à exécuter en parallèle le testeur et l'implantation. Pendant cette exécution des observations peuvent être faites et des verdicts émis. Les observations pouvant être faites sont les suivantes :

- l'état $\perp$ a été atteint. On peut immédiatement émettre un verdict **fail** exprimant la non conformité ;
- étant parti de s_0, une branche de l'objectif de test a été parcourue et l'on est retourné sur s_0. Le verdict **pass** ne peut pas être émis immédiatement car du fait du non-déterminisme, un verdict **fail** aurait peut-être aussi pu être émis. L'émission du verdict **pass** est retardée jusqu'à ce que l'on puisse décider qu'un verdict **fail** n'aurait pas pu être émis. C'est l'hypothèse d'équité bornée qui intervient ici ;

– étant dans le corps de l'objectif de test, on observe une trace admise par la spécification mais qui ne fait pas partie de l'objectif de test. Souvent un verdict inconclusif est émis à ce stade. Ici, nous continuons l'exécution de test afin d'émettre un autre verdict **pass** ou **fail**.

Pour les résultats énoncés ci-dessous, nous utilisons les notations d'algèbre de processus. Pour chaque IOSM, nous associons un terme de l'algèbre des processus (cf. [Tre96] pour les notations complètes). L'exécution d'un test sur une IUT est définie par la composition parallèle dénotée $\|$ et définie par :

$$\forall a \in \{?,!\} \times L, \frac{t_1 \xrightarrow{a} t_1', t_2 \xrightarrow{a} t_2'}{t_1 \| t_2 \xrightarrow{a} t_1' \| t_2'}$$

.

Du fait de l'hypothèse de complétude sur les actions d'entrée, l'exécution parallèle ne peut bloquer que sur $\perp$ ou s_0.

5.1. Correction

La correction, telle que définie dans FMCT, signifie que si une implantation est rejetée par le cas de test (état $\perp$) alors elle n'est pas conforme à la spécification. Nous montrons que les tests produits par notre méthode sont corrects :

Proposition 4 (correction):
Soient S, I, TP, T_{TP} une spécification, un modèle d'implantation, un objectif de test et un testeur associé à S et TP. T_{TP} est correct :

$$\exists \sigma \in V^+ : (t_{TP} \| i \xrightarrow{\sigma} \perp \| i') \Rightarrow (I, S) \notin R.$$

(où i et i' sont des sommets arbitraires de I)

Preuve:
Supposons que $t_{TP} \| i \xrightarrow{\sigma} \perp \| i'$. Soit σ' le mot minimal (par rapport à l'ordre de préfixe) tel que l'on ait : $t_{TP} \| i \xrightarrow{\sigma} \perp \| i'$, et $a \in V$ tel que $\sigma' = \sigma'' \cdot !a$. Par construction du jeu et minimalité de σ', σ'' appartient à $Tr(S)$; par définition de $\|$ il appartient aussi à $Tr(I)$ (de même que $\sigma'' \cdot !a$). Par définition de $\perp$ dans le jeu, $\sigma'' \cdot !a$ n'appartient pas à $Tr(S)$. Nous avons prouvé que $\mathcal{O}(\sigma'', I) \not\subseteq \mathcal{O}(\sigma'', S)$. ■

La propriété de correction est nécessaire pour une suite de test mais elle n'est pas suffisante. En effet, un testeur rendant toujours le verdict **pass** est correct, ce qui bien sûr n'est pas très satisfaisant. Il faudrait dans l'absolu que toute implantation non conforme soit rejetée par un test. Cette propriété d'exhaustivité est bien évidemment impossible à obtenir. Par contre, en utilisant des hypothèses d'équité sur l'IUT, on peut obtenir un résultat plus faible. C'est ce que nous examinons aux deux paragraphes suivants.

5.2. Hypothèses d'équité

Un problème dû à la nature boîte noire du test et au non-déterminisme (observable) des implantations est de savoir si toutes les sorties possibles ont été observées après une suite d'actions. Dans l'exemple de la figure 6, le testeur veut observer la trace $?a!c$. Ayant observé $(?a!b)^*$ lors de l'exécution parallèle avec l'un des modèles d'implantation, il se peut qu'il s'agisse du premier modèle ou du second qui retarde l'émission de $!c$. Ayant observé la trace $(?a(!b|!c))^*$, il est possible qu'il s'agisse du modèle 2 ou du modèle 3.

Deux notions d'hypothèse d'équité seront utilisées dans le résultat concernant l'exhaustivité :

– l'équité : comme dans le cas des parties équitables que nous avons utilisé pour les jeux, l'équité implique que si le second modèle est exécuté en parallèle avec le testeur, $!c$ sera finalement observé. L'équité discrimine donc les modèles 1 et 2.

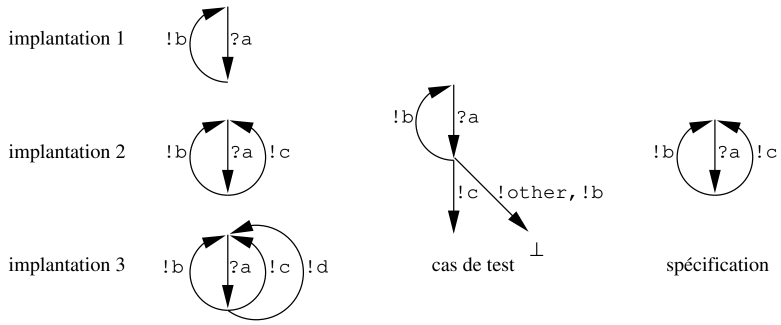

Figure 6. *Hypothèses d'équité*

- l'équité bornée : il existe un entier n connu a priori et dépendant de la longueur de la trace à observer tel qu'ayant observé $(?a(!b|!c))^n$, on est assuré de ne pas observer $!d$ ensuite. L'équité bornée discrimine les modèles 2 et 3.

Contrairement au cas de l'équité (simple) où un nombre fini mais non borné est suffisant pour décider si une interaction que l'on souhaite observer a lieu, l'équité bornée est nécessaire pour décider si une interaction non prévue a lieu. Les définitions découlent des exemples ($\lhd$ est la relation de sous mot).

Définition 7.
Les hypothèses suivantes peuvent être faites sur les modèles d'implantation :

- *équité : un modèle d'implantation I d'une spécification S satisfait une hypothèse d'équité si :*
$$\forall \sigma \in Tr(I) \cap Tr(S) : \forall a \in \mathcal{O}(\sigma, I) : \exists n \in \mathbb{N} : (\exists t : \exists \omega = \omega_1, \dots, \omega_n : t\|i \xrightarrow{\omega} t'\|i' \wedge \bigwedge_i \sigma \lhd \omega_i) \Rightarrow \exists j : \sigma \cdot !a \lhd \omega_j.$$
- *équité bornée : un modèle d'implantation I d'une spécification S satisfait une hypothèse d'équité bornée si :*
$$\forall \sigma \in Tr(I) \cap Tr(S) : \exists n \in \mathbb{N} : \forall a \in \mathcal{O}(\sigma, S) : \big(\exists t : \exists \omega = \omega_1, \dots, \omega_n : (t\|i \xrightarrow{\omega} t'\|i' \wedge \bigwedge_i (\sigma \lhd \omega_i \wedge \sigma \cdot !a \not\lhd \omega_i)\big) \Rightarrow \sigma \cdot !a \notin Tr(I).$$

5.3. Exhaustivité

L'exhaustivité signifie qu'une implantation passant avec succès une suite de test est conforme à la spécification. La propriété que nous prouvons ici est plus faible. Nous prouvons qu'il est toujours possible de détecter une implantation non conforme sous réserve de choisir judicieusement l'objectif de test. Ce résultat donne une limite à l'automatisation de la génération de test : la précision de l'algorithme dépend du choix de l'objectif de tests et ce choix repose sur l'intuition des experts (humains) de test.

Proposition 5 (complétude réfutationnelle):
Soient S et I une spécification et un modèle d'implantation. Si I satisfait une hypothèse d'équité bornée, alors :

$$(I, S) \notin R \Rightarrow \exists TP \exists \sigma \in V^+ : t_{TP}\|i \xrightarrow{\sigma} \perp\|i'.$$

PREUVE:
Supposons $(I, S) \notin R$. Deux cas sont possibles :

- $\mathcal{O}(\sigma, I) \not\subseteq \mathcal{O}(\sigma, S)$. Dans ce cas, il existe $a \in L$ tel que, pour un σ, $\sigma\cdot!a$ appartient aux traces de I mais pas à celles de S. Soit T le testeur associé à S et à l'objectif de test σ. Soit r tel que $t \xrightarrow{\sigma} r$. Par construction du testeur, il existe σ_0 tel que $t||i \xrightarrow{\sigma_0} r||i_0$ et σ_1 tel que $t||i \xrightarrow{\sigma_0\sigma_1} t||i_1$. En réitérant ce processus on construit une séquence qui passe par r aussi souvent nécessaire. Par hypothèse d'équité, comme l'implantation peut effectuer la transition $!a$, il existe n tel que $t||i \xrightarrow{\sigma_0\ldots\sigma_{2n}} r||i_{2n} \xrightarrow{!a} \perp||i_{2n+1}$.
- $\mathcal{O}(\sigma, S) \not\subseteq \mathcal{O}(\sigma, I)$. Dans ce cas, il existe $a \in L$ tel que $\sigma\cdot!a$ appartient a l'ensemble des traces de S mais pas à celles de I. Soit T le testeur associé à S et à l'objectif de test $\sigma!a$. De plus nous utilisons le n de l'hypothèse d'équité bornée pour le testeur. À chaque point de choix où l'implantation peut choisir de quitter l'objectif de test, nous admettons seulement n essais puis le testeur passe dans l'état $\perp$. Si après n essais, le produit parallèle ne s'est pas bloqué en $\perp$, le testeur stoppe et émet le verdict **pass**. Comme le produit parallèle ne peut se bloquer que sur $\perp$ et qu'il finira par se bloquer, il le fera sur $\perp$. ∎

L'équité bornée n'est utilisée que pour l'inclusion des sorties de la spécification dans celle de l'implantation. En affaiblissant l'hypothèse d'équité bornée en une hypothèse d'équité, on retrouve le résultat de la proposition 3.3 de [FJJV96] pour une relation d'implantation plus faible qui ne considère justement que l'inclusion des sorties. D'un point de vue pratique, l'hypothèse d'équité bornée est celle qui est applicable car elle fixe *a priori* la longueur de l'exécution d'un test.

5.4. Comparaison et travaux relatifs

Nous avons présenté un nouvel algorithme de génération de test de conformité. Cet algorithme est comparable à ceux de TGV [FJJV96, JM99]. Bien qu'ils proviennent de domaines a priori éloignés (model-checking et théorie des jeux), tous les deux sont essentiellement des algorithmes de graphes. Notre algorithme met l'accent sur les points suivants :

- Suppression du verdict inconclusif : dans [FJJV96, JM99], il est implicite qu'un test doit être réexécuté lorsqu'il a conduit à un verdict inconclusif. Dans notre approche, la réexécution du test est codé explicitement dans le cas de test. Cela pourrait conduire à un testeur non exécutable (dans le sens de [Pha94] où, du fait de boucles, l'émission d'un verdict en temps fini pourrait ne pas être possible). On utilise donc l'hypothèse (raisonnable) d'équité sur les implantations pour assurer théoriquement la terminaison de l'exécution du test. D'un point de vue pratique des hypothèses plus fortes (équité bornée par exemple) peuvent être utilisées pour réduire encore la longueur du test. La réexécution d'un test suppose souvent implicitement l'existence d'un reset bien implanté. Nous avons pris une hypothèse plus faible, la forte connexité de la spécification. Cela peut conduire à des tests plus courts, car revenir par l'état initial n'est pas toujours le plus court chemin pour retourner à un état donné. Ceci est particulièrement important lorsque l'on traite le non-déterminisme (dans le sens du test) car le chemin pris est celui comportant le moins de point de choix non-déterministes qui, comme nous l'avons vu dans la proposition 3, peuvent augmenter arbitrairement la longueur d'un test.
- Émission de verdicts : dans [FJJV96] deux applications du même test peuvent produire des verdicts différents. Moyennant les hypothèses d'équité, les cas de tests que nous produisons ne peuvent pas produire un verdict **pass** et un verdict **fail**. Ce problème est résolu théoriquement par la définition du verdict **pass** dans [Tre96] : un verdict **pass** n'est émis que si *toutes* (une infinité) les exécutions possibles d'un test conduisent au verdict **pass**. Ceci ne peut être observé en pratique. L'utilisation explicite d'hypothèses de test réduit ce nombre à un nombre fini.
- Une relation d'implantation plus forte : la relation d'implantation de [FJJV96] est fondée sur l'inclusion des sorties , alors que celle utilisée ici utilise l'égalité.

Ces extensions reposent d'une part sur une prise en compte plus fine de l'équité et d'autre part sur la notion de stratégie qui utilise un automate pour être implantée. Les cas de tests sont alors donnés sous forme de graphes et non plus d'arbres comme c'est souvent le cas, sauf dans [JM99].

Les travaux relatifs dans le cadre de la théorie des jeux se placent hors du cadre du test de conformité. [ACY95] (le seul à notre connaissance) étudie la complexité d'un problème de test (*distinguishing test sequences*) qui n'est pas un problème de test de conformité dans le sens de FMCT car la relation de conformité testée n'est pas donnée. La notion de jeux intervient cependant souvent dans les études relatives à des problèmes de corrections de systèmes réactifs : problème du *deadlock* [Dev77], plus généralement celui de la synthèse [NYY92] (notre vision intuitive du test comme un jeu s'inspire de ces travaux) et [PR90], enfin plus récemment celui du *model checking* [Sti96, SS98, AHK97].

6. CONCLUSION

Dans cet article, nous avons présenté une connection entre test de conformité et théorie des jeux et démontré quels apports peuvent être obtenus grâce à cette technique. La notion de maximisation de gain, présente dans tous les problèmes de jeux, s'exprime naturellement dans le cadre du test de conformité : les interactions choisies par le testeur doivent être optimales pour conduire à la réalisation d'un objectif de test. L'optimalité s'exprime suivant différents critères. Ici, l'optimalité se définit par rapport aux choix que peut faire l'implantation pour « contrecarrer » le déroulement du test. Minimiser les possibilités d'émission de l'IUT, c'est maximiser ses chances de réaliser l'objectif de test. Ce n'est pas le critère habituel de minimalité des séquences de test qui en général considère la notion de longueur du test.

Cette connection s'étend à d'autres aspects de la recommandation FMCT. Citons trois exemples :

- *Architectures de test et jeux multi-joueurs.* Nous avons pris ici l'architecture de test la plus simple. Un jeu multi-joueurs modéliserait des architectures à plusieurs testeurs. La difficulté introduite provient de la notion de coopération : les testeurs doivent jouer ensemble pour maximiser leurs chances contre l'implantation. Ce type de jeux (dit « jeux coopératifs ») est plus difficile à étudier que les jeux non coopératifs, comme le test distribué est plus difficile à mettre en œuvre que le test simple.
- *La notion d'information incomplète et les problèmes d'observabilité et de testabilité.* La notion de jeux à information incomplète raffine la modélisation de l'implantation vue comme boite noire en prenant en compte les actions non observables de l'implantation. C'est une modélisation plus fidèle du du processus de test.
- *Jeux probabilistes et couverture.* L'introduction de probabilités permet de maximiser les chances de détecter une erreur en fonction du comportement probable de l'implantation et de pouvoir donner une mesure de couverture de la spécification.

Ces parallèles permettent l'étude théorique de la décidabilité (quels sont les limites de ce que l'on peut tester ?) et de la complexité (quel est l'effort maximal à fournir dans le pire des cas ?). De façon pratique et comme nous l'avons vu dans cet article, la construction effective (et efficace) de stratégies gagnantes donne une procédure de génération de tests. Pour conclure, nous pensons que cette méthode peut être étendue pour offrir un un cadre uniforme au test et au *model checking.* Des logiques comme la *game logic* [AHK97], prenant en compte l'alternation caractéristique de la procédure de construction de stratégies, peuvent être utilisées aussi pour la formulation des objectifs de test, la procédure de *model checking* devenant aussi une procédure de génération de test pour une sous-classe de formules. À plus long terme un outil utilisant ce cadre permettrait au spécifieur/vérificateur/testeur l'économie de deux formalismes différents pour des tâches somme toute relativement proches.

Remerciements

Le premier auteur a reçu le soutien d'EDF, Direction des Études et Recherches. Il remercie les professeurs GANZINGER et BERNOT pour leur accueil au sein de leur équipe. Le second auteur a reçu le soutien des groupes de travail ESPRIT-IV 22704 ASPIRE et 23531 FIREworks. La collaboration entre les auteurs a eu lieu dans le cadre de l'action DSP\STTC/CNRS/MENRT FORMA « Maîtrise des systèmes complexes réactifs et sûrs » (`http://www.imag.fr/FORMA`).

7. BIBLIOGRAPHIE

[ACY95] Rajeev Alur, Costas Courcoubetis, and Mihalis Yannakakis. Distinguishing tests for nondeterministic and probabilistic machines. In *Proceedings of the 27th ACM Symposium on Theory of Computing*, pages 363–372, 1995.

[AHK97] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science*, Floride, Octobre 1997.

[AJG99] Kerbrat. Alain, Thierry Jéron, and Roland Groz. Automated test generation from sdl specifications. In *Proceedings of the 9th SDL Forum'99 (Montréal, Canada)*, june 1999. À paraître.

[Ber91] Gilles Bernot. Testing against formal specifications : a theoritical view. In S. Abramsky and T.S.E. Maibaum, editors, *TAPSOFT'91 Proceedings of the International Conference on Theory and Practice of Software Development. Advances in Distributed Computing (ADC) and Colloquium on Computing Paradigms for Software Development (CCPSD)*, volume 494 of *Lecture Notes in Computer Science*, pages 99–119, Brighton, UK, April 1991. Springer.

[CFP96] Ana R. Cavalli, Jean-Philippe Favreau, and Marc Phalippou. Standardization of formal methods in conformance testing of communication protocols. *Computer Network and ISDN Systems*, 29 :3–14, 1996.

[Dev77] R. Devillers. Game interpretation of the deadlock avoidance problem. *Communications of the ACM*, 20(10) :741–745, 1977.

[FJJV96] Jean-Claude Fernandez, Claude Jard, Thierry Jéron, and César Viho. Using on-the-fly verification techniques for the generation of test suites. In Rajeev Alur and Thomas A. Henzinger (Eds), editors, *8th International Conference on Computer-Aided Verification (CAV'96)*, volume 1102 of *LNCS*, pages 348–359, New Brunswick,NJ,USA, July/August 1996.

[ISO85] ISO. Estelle : a Formal Description Technique Based on an Extended Transition Model. Technical Report TC97-SC21, ISO, 1985.

[ISO91] ISO. Information Technology, Open Systems Interconnection—Conformance Testing Methodology and Framework. International Standard IS-9646, 1991.

[ITU97] Itu-t recommendation z.500 : Framework on formal methods in conformance testing, May 1997. Aussi ISO/IEC DIS 13245-1.

[JM97] Thierry Jéron and Pierre Morel. Abstraction, τ-réduction et déterminisation à la volée : application à la génération de test. In *Actes du sixième Colloque Francophone sur l'Ingéniérie des Protocoles, CFIP'97*, pages 255–270. Hermes, 1997.

[JM99] Thierry Jéron and Pierre Morel. Test generation derived from model-checking. In *Proceedings of the 11th International Conference on Comuter Aided Verification, CAV'99*, Lecture Notes in Computer Science, Trento, Italy, Juillet 1999. Springer. À paraître.

[KPT] Finn Kristoffersen, Marc Phalippou, and Jan Tretmans. Formal methods in conformance testing : A brief bibliography. Tutorial on Formal Methods in Conformance Testing, PSTV'95.

[LOT85] A formal description technique based on the temporal ordering of obs ervational behaviour, March 1985.

[NYY92] Anil Nerode, Alexander Yakhnis, and Vladimir Yakhnis. Concurrent programs as strategies in games. In Y.N. Moschovakis, editor, *Logic from computer science*, volume 21 of *Mathematical sciences research institute publications*. Springer, 1992.

[Pha94] Marc Phalippou. *Relations d'implantation et hypothèses de test sur des automates à entrées sorties.* Thèse de doctorat, Université de Bordeaux I, Septembre 1994.

[PR90] Amir Pnueli and Roni Rosner. Distibuted reactive systems are hard to synthesize. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science, FOCS'90*, pages 746–757, 1990.

[Ram98] Solofo Ramangalahy. Strategies for conformance testing. Research Report MPI-I-98-2-010, Max-Planck-Institut für Informatik, Im Stadtwald, D-66123 Saarbrücken, Germany, May 1998. `http://data.mpi-sb.mpg.de/internet/reports.nsf/NumberView/98-2-010`.

[Ram99] Solofo Ramangalahy. Test de conformité pour des spécifications à base d'automates : une approche par la théorie des jeux, Thèse prévue pour Septembre 1999.

[RLGJ99] Solofo Ramangalahy, Pascale Le Gall, and Thierry Jéron. Une application de la théorie des jeux au test de conformité. In *Actes du huitième Colloque Francophone sur l'Ingéniérie des Protocoles, CFIP'99*, pages 165–179. Hermes, 1999.

[SS98] Perdita Stevens and Colin Stirling. Practical model-checking using games. In B. Steffens, editor, *Proceedings of the 4th International Conference Tools and Algorithms for the Construction and Analysis of Systems, (TACAS)*, number 1384 in Lecture Notes in Computer Science, Lisbon, Portugal, March 28 - April 4 1998.

[Sti96] Colin Stirling. Games and modal mu-calculus. In *Proceedings of 2nd International Workshop on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'96*, volume 1055 of *Lecture Notes in Computer Science*, pages 298–312, Passau, Germany, March 1996.

[Tho94] Wolfgang Thomas. Finite-state strategies in regular infinite games. In P.S. Thiagarajan, editor, *14th Conf. on Foundations of Software Technology and Theorical Computer Science (FST & TCS)*, volume 880 of *Lecture Notes in Computer Science*, pages 149–158, Madras, India, December 1994. Invited Paper.

[Tre92] Jan Tretmans. *A formal Approach to Conformance Testing.* Proefschrift (thèse de doctorat), Universiteit Twente, December 1992.

[Tre96] Jan Tretmans. Test generation with inputs, outputs and repetitive quiescence. *Software – concepts and tools*, 17 :103–120, 1996.

[Z.188] Specification and Description Language SDL, 1988.

∞

Biographies

Solofo Ramangalahy est doctorant sous contrat CTCI CEA/EDF dans le Groupe Logiciel pour la Sureté de Procédés du LETI à Saclay et dans l'équipe LAMI de l'Université d'Evry. Ses centres d'intérêts concernent l'application de la théorie des jeux au test de conformité.

Pascale Le Gall est Maître de Conférences au Laboratoir des Méthodes Informatiquees (CNRS EP738) à l'Université d'Évry. Ses sujets d'intérêt concernent les méthodes formelles et leurs applications, et plus particulièrement la définition d'un cadre pour des spécifications hiérarachiques hétérogènes et les méthodes de test à partir des spécifications formelles.

Thierry Jéron est Chargé de Recherche INRIA dans le projet PAMPA de l'IRISA (Rennes). Son principal sujet d'intérêt porte sur la génération de suites de test pour les protocoles. Il a participé à l'élaboration de l'outil TGV.