Synthèse de contrôleurs pour une relation de conformité

Thierry Jéron, Hervé Marchand, Vlad Rusu, Valérie Tschaen

Synthèse de contrôleurs pour une relation de conformité

T. Jéron — H. Marchand — V. Rusu — V. Tschaen

IRISA/INRIA Rennes, Projet VerTeCs

{Thierry.Jeron,Herve.Marchand,Vlad.Rusu,Valery.Tschaen}@irisa.fr

RÉSUMÉ. *Nous nous intéressons à la combinaison du test de conformité et de la synthèse de contrôleurs ; spécifiquement, au problème de rendre une implémentation d'un système conforme à sa spécification, à l'aide d'un contrôleur calculé automatiquement. Ce dernier peut être vu comme un "patch" qui corrige automatiquement des erreurs, qui autrement auraient dû être détectées par le test et corrigées à la main. Nous traitons ici le cas où toutes les actions de la spécification et de l'implémentation sont observables depuis l'environnement, mais seul un sous-ensemble des actions est contrôlable.*

ABSTRACT. *We are interested here in combining conformance testing and supervisory control techniques. Specifically, we study the problem of controlling an implementation of a system, in order to make it conformant to its specification, by means of an automatically computed supervisor. The latter can be seen as a patch that fixes errors, which otherwise should have been detected and fixed by hand. We study the case where all actions are observable, but only a subset of them is controllable.*

MOTS-CLÉS : *synthèse de contrôleur, systèmes réactifs, conformité*

KEYWORDS: *supervisory control problem, reactive systems, conformance*

1. Introduction

Le test de conformité et la synthèse de contrôleurs sont deux techniques visant à assurer la correction de systèmes réactifs. Le *test de conformité* [TRE 96] consiste à comparer, selon une relation dite de conformité, les comportements observables d'une implémentation d'un système par rapport à ceux décrits par sa spécification. La *synthèse de contrôleurs* [RAM 89] consiste à modifier une implémentation afin de la rendre correcte par rapport à sa spécification, selon des critères appelés objectifs de contrôle. Les deux techniques utilisent des méthodes et des algorithmes inspirés de techniques de *vérification formelle* comme le model checking. Les similitudes entre ces techniques sont nombreuses, et nous allons chercher à les exploiter.

Dans cet article, nous nous intéressons au problème de la synthèse de contrôleurs pour assurer la relation de conformité décrite dans [TRE 96] entre une implémentation d'un système réactif et sa spécification. Cette relation, notée **ioco**, est probablement la plus courante en test de conformité. Étant données une spécification S d'un système et une implémentation IMP non conforme à S, résoudre le problème suivant : calculer un contrôleur $\mathcal{C}$ tel que l'implémentation contrôlée $\mathcal{C}/\text{IMP}$ soit conforme à la spécification S: $\mathcal{C}/\text{IMP}$ **ioco** S. Comme c'est généralement le cas dans la synthèse de contrôleurs, ce problème est plus ou moins complexe selon le degré d'observabilité et de contrôlabilité des actions de l'implémentation.

Nous abordons ici le cas où toutes les actions sont observables, mais seulement un sous-ensemble est contrôlable. Notre algorithme calcule un contrôleur $\mathcal{C}$ optimal, au sens où l'implémentation contrôlée $\mathcal{C}/\text{IMP}$ est le plus grand système, inclus dans l'implémentation non contrôlée IMP, contrôlable et conforme à la spécification S.

L'article est organisé comme suit. Dans la Section 2 nous présentons les modèles que nous utilisons ainsi que les principes de la synthèse de contrôleurs et du test de conformité. Nous traitons ensuite la synthèse de contrôleurs pour assurer la conformité dans la Section 3. L'algorithme principal de synthèse est illustré sur un exemple simple. Pour finir, nous présentons nos conclusions et perspectives en Section 4.

2. Contexte

2.1. *Modèles et notations*

Le modèle que nous utilisons est celui des systèmes de transitions étiquetés, abrégé LTS (pour *Labelled Transition System*) avec quelques adaptations spécifiques au test et au contrôle.

Définition 1 (LTS) *Un LTS est un quadruplet $M = (Q, A, \rightarrow, q_0)$ où Q est un ensemble fini d'états, $q_0 \in Q$ est l'état initial, A est l'alphabet des actions (ou événements) et $\rightarrow \subseteq Q \times A \times Q$ est la relation de transition.* •

Notations: Soit $M = (Q, A, \rightarrow, q_0)$ un LTS. On note :

– $q \xrightarrow{a} q' \triangleq (q, a, q') \in \rightarrow$;

– $q \xrightarrow{\sigma} q'$ avec $\sigma = a_1 \dots a_n \triangleq \exists q_0, \dots, q_n : q = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n = q'$;

– $q \xrightarrow{\sigma} \triangleq \exists q' : q \xrightarrow{\sigma} q'$;

– $\mathcal{L}(q) \triangleq \{\sigma \in A^* \mid q \xrightarrow{\sigma}\}$, l'ensemble de séquences d'actions de M à partir de q;

– $\mathcal{L}(M) \triangleq \mathcal{L}(q_0)$, le comportement global du système M;

– $\Gamma(q) \triangleq \{a \in A \mid q \xrightarrow{a}\}$, l'ensemble des actions possibles en q;

– q after $\sigma \triangleq \{q' \in Q \mid q \xrightarrow{\sigma} q'\}$ (resp. P after $\sigma \triangleq \bigcup_{q \in P} q$ after σ).

Un LTS est *déterministe* lorsque pour tout état q et pour tout événement a, t.q. $q \xrightarrow{a}$, il existe un unique q' t.q. $q \xrightarrow{a} q'$. Un état q est *atteignable* s'il existe une séquence $\sigma \in A^*$ d'actions telle que $q_0 \xrightarrow{\sigma} q$. Un état q est *co-atteignable* pour un ensemble $Q' \subseteq Q$ d'états s'il existe un état $q' \in Q'$ et une séquence $\sigma \in A^*$ d'actions tels que $q \xrightarrow{\sigma} q'$. Un LTS est *atteignable* si tous ses états sont atteignables. Tous les LTS que nous considérons dans cet article sont déterministes et atteignables.

Définition 2 (Composition parallèle) *Soient $M_i = (Q^{M_i}, A^{M_i}, \rightarrow_{M_i}, q_0^{M_i})$, $i = 1, 2$, deux LTS. La composition parallèle $M_1 \parallel M_2$ de M_1 et M_2 est le LTS $M_{12} = (Q^{M_{12}}, A^{M_{12}}, \rightarrow_{M_{12}}, q_0^{M_{12}})$ tel que :*

– *$Q^{M_{12}} = Q^{M_1} \times Q^{M_2}$, $q_0^{M_{12}} = (q_0^{M_1}, q_0^{M_2})$, $A^{M_{12}} = A^{M_1} \cup A^{M_2}$;*

– *Pour tout état $q = (q_1, q_2) \in Q^{M_{12}}$ et toute action $a \in A^{M_{12}}$:*

- *Si $a \in A^{M_1} \cap A^{M_2}$, alors $(q_1, q_2) \xrightarrow{a}_{M_{12}} (q_1', q_2')$ ssi $q_1 \xrightarrow{a}_{M_1} q_1'$ et $q_2 \xrightarrow{a}_{M_2} q_2'$*
- *Si $a \in A^{M_1} \setminus A^{M_2}$, alors $(q_1, q_2) \xrightarrow{a}_{M_{12}} (q_1', q_2)$ ssi $q_1 \xrightarrow{a}_{M_1} q_1'$,*
- *Si $a \in A^{M_2} \setminus A^{M_1}$, alors $(q_1, q_2) \xrightarrow{a}_{M_{12}} (q_1, q_2')$ ssi $q_2 \xrightarrow{a}_{M_2} q_2'$.* •

2.2. *La synthèse de contrôleurs*

Nous rappelons brièvement la théorie du contrôle des systèmes à événements discrets de Ramadge & Wohnam [RAM 89] pour des systèmes modélisés par des LTS. Il nous faut distinguer les actions sur lesquelles il est possible d'agir (*contrôlables*), et celles sur lesquelles il est impossible d'agir (*incontrôlables*).

Définition 3 (Système à contrôler) *Un système à contrôler est un LTS $(Q, A, \rightarrow, q_0)$ dont l'alphabet A est partitionné en deux sous-ensembles : A_c, l'alphabet des actions contrôlables, et A_{uc}, l'alphabet des actions incontrôlables. De plus, le LTS est muni d'un ensemble $Q_m \subseteq Q$ d'états* finals. •

Dans la théorie de Ramadge & Wohnam, le système est modélisé par un LTS M, et caractérisé par deux langages: le langage de M : $\mathcal{L}(M)$, et le langage *marqué* de M: $\mathcal{L}_m(M) = \{\sigma \in A^* \mid \exists q' \in Q_m : q_0 \xrightarrow{\sigma} q'\}$. Intuitivement, $\mathcal{L}(M)$ représente des séquences possibles d'actions du système, alors que $\mathcal{L}_m(M)$ représente l'ensemble de

ces séquences qui terminent une tâche avec succès. Pour $K \subseteq A^*$ un langage sur l'alphabet A, $\overline{K}$ est la clôture par préfixe du langage K. Un LTS est dit *bloquant* lorsque $\mathcal{L}(M) \neq \overline{\mathcal{L}_m(M)}$. Si M est bloquant, il peut atteindre soit un état q en deadlock (*i.e.* un état q à partir duquel aucune action n'est possible : $\Gamma(q) = \emptyset$), soit un ensemble d'états non marqués de M formant une composante fortement connexe terminale. Si le système atteint cet ensemble, il n'a plus aucun moyen d'atteindre un état marqué.

Il est possible de montrer [CAS 99] que si tous les états de M sont atteignables depuis l'état initial q_0, et co-atteignables pour Q_m, alors le système est non-bloquant.

Le contrôleur. Le but d'un contrôleur $\mathcal{C}$ est d'agir sur l'évolution d'un système modélisé par un LTS M en permettant (ou en interdisant) l'occurrence d'actions contrôlables en fonction du comportement passé du système. Formellement, un contrôleur est une fonction $\mathcal{C}$ allant de $\mathcal{L}(M)$ dans 2^A qui, à une observation $\sigma \in \mathcal{L}(M)$, associe une politique de contrôle $\mathcal{C}(\sigma) \in 2^A$, l'ensemble des événements permis après l'occurrence de cette séquence. Comme tous les événements ne sont pas contrôlables, tous les contrôleurs ne sont pas admissibles. En particulier, un contrôleur ne doit pas interdire des actions incontrôlables de se produire alors qu'elles étaient admissibles dans le système non contrôlé. Ceci est formalisé dans la définition suivante :

Définition 4 (Contrôleur admissible et système contrôlable) *Un contrôleur $\mathcal{C}$ est* admissible *par rapport à un système M si $\mathcal{L}(\mathcal{C}/M) \cdot A_{uc} \cap \mathcal{L}(M) \subseteq \mathcal{L}(\mathcal{C}/M)$. Le système $\mathcal{C}/M$ est alors* contrôlable *par rapport au LTS M et à l'alphabet A_{uc}.* •

La problématique du contrôle est alors la suivante: *Étant donnés un LTS à contrôler M et un LTS S (appelé objectif de contrôle, ou encore spécification) tels que M et S portent sur le même alphabet, synthétiser un contrôleur admissible, non-bloquant $\mathcal{C}$ tel que $\mathcal{L}_m(\mathcal{C}/M) \subseteq \mathcal{L}_m(S)$, et $\mathcal{L}_m(\mathcal{C}/M)$ est le plus permissif,* i.e. *, pour tout contrôleur $\mathcal{C}'$, admissible et non-bloquant tel que $\mathcal{L}_m(\mathcal{C}'/M) \subseteq \mathcal{L}_m(S)$, on a l'inclusion $\mathcal{L}_m(\mathcal{C}'/M) \subseteq \mathcal{L}_m(\mathcal{C}/M)$.*

L'algorithme standard (e.g. [CAS 99]) qui résout ce problème consiste à calculer la composition $M||S$, puis à itérer les deux opérations suivantes jusqu'à stabilisation : *(1)* enlever les états non-atteignables, et ceux qui ne sont pas co-atteignables pour l'ensemble des états finals ; en théorie du contrôle, cette opération est appelée TRIM. *(2)* enlever les états à partir desquels, par une séquence d'actions incontrôlables, une action permise par M, mais pas par S, est tirable (opération appelée CONTROLLABLE).

Algorithme SUPCONT

ENTRÉES: Système à contrôler M$= (Q^{\text{M}}, A^{\text{M}}, \rightarrow_{\text{M}}, q_0^{\text{M}})$ avec $Q_{\text{m}}^{\text{M}} \subseteq Q^{\text{M}}$ les états marqués, et objectif de contrôle S $= (Q^{\text{S}}, A^{\text{S}}, \rightarrow_{\text{S}}, q_0^{\text{S}})$, avec $A^M = A^S$.

SORTIE : $\mathcal{C}/M$ le système contrôlé.

(i) Soit $i = 0$ et $G_i = M \parallel S = (Q_i, A, \rightarrow_i, q_{i_0}), Q_{\text{m}}^{\text{i}} = (Q_{\text{m}}^{\text{M}} \times Q^{\text{S}}) \cap Q_i$
(ii) $G'_i = (Q'_i, A, \rightarrow_{i'}, q_{i_0}) = \text{TRIM}(G_i)$ et $Q_{\text{m}}^{\text{i'}} = Q_{\text{m}}^{\text{i}} \cap Q'_i$
(On enlève tous les états qui ne sont pas atteignables ou co-atteignables)
(iii) $G_{i+1} = \text{CONTROLLABLE}(G'_i, M, A_{uc})$. i.e.

a) Calcul des états *interdits* :
$F_s^i = \{q = (q_{\text{M}}, q_{\text{S}}) \in Q'_i \mid \exists a \in A_{uc},\ q_{\text{M}} \xrightarrow{a}_{\text{M}}$ et $q \not\xrightarrow{a}_{i'}\}$
b) Calcul des états *faiblement interdits* :
$W_s^i = \{q \in Q_i \mid \exists \sigma \in A_{uc}^*,\ t.q.\ (q \text{ after } \sigma) \cap F_s^i \neq \emptyset\}$
c) $G_{i+1} = G'_i$, t.q. :
$Q_{i+1} = Q'_i \setminus W_s^i$, restriction de $\rightarrow_{i+1}$ aux états Q_{i+1} et $Q_{\text{m}}^{\text{i+1}} = Q_{\text{m}}^{\text{i'}} \cap Q_{i+1}$
(iv) Si $G_{i+1} \neq G'_i$: $i = i + 1$, Goto (ii)
(v) $\mathcal{C}/M = G'_i$

2.3. *Le test de conformité*

Le test de conformité est une technique de validation de systèmes réactifs. Cette technique suppose l'existence d'une spécification formelle S des comportements du système ainsi que d'une implémentation IMP de ce système, un programme exécutable. Son but est de valider l'implémentation IMP par rapport à la spécification S. Typiquement, l'implémentation est une boîte noire : seules ses interfaces sont connues et accessibles. À travers ces interfaces, un autre système, appelé *testeur*, interagit avec l'implémentation par des séquences appelées *cas de test*. Ces séquences consistent à *stimuler* l'implémentation, au moyen, par exemple, d'envois de messages, d'appels de fonctions ou de méthodes, et à observer ses *réponses* (messages reçus, résultats d'appels, etc). Le but de ces expériences est de détecter des différences observables, appelées *non-conformités*, entre implémentation et spécification.

La relation de conformité a été formalisée par [TRE 96], et la méthodologie du test de conformité **ioco** a été largement automatisée, depuis la génération de cas de test à partir d'une spécification, jusqu'à l'exécution de ceux-ci sur une implémentation et l'obtention des verdicts [BEL 99, JéR 02]. Nous n'entrerons pas dans les détails dans cet article. Pour le problème qui nous préoccupe, *i.e.* assurer la conformité d'une implémentation vis-à-vis d'une spécification par la synthèse de contrôleur, nous n'aurons besoin que de quelques définitions : le modèle utilisé et la relation de conformité.

2.3.1. *Les IOLTS*

Le modèle des LTS donné dans la définition 1 n'est pas adapté au test de conformité. Il n'y est fait aucune différence entre entrées (stimuli) et sorties (réponses) d'un système. Or, dans le cadre du test, la distinction est fondamentale.

Définition 5 (IOLTS, Input/Output Labelled Transition System) *Un IOLTS est un LTS $(Q, A, \rightarrow, q_0)$ dont l'alphabet d'actions A est partitionné en deux ensembles : $A = A_? \cup A_!$ avec $A_?$ l'alphabet d'entrées et $A_!$ l'alphabet de sorties.* •

Relativement à ce partitionnement de l'alphabet, on définit l'ensemble des sorties possibles dans un état $q \in Q$ par $Out(q) \triangleq \Gamma(q) \cap A_!$.

Modélisation des blocages. Nous allons revoir la définition du blocage, introduite en section 2.2 pour les LTS à contrôler, afin de l'adapter aux IOLTS. Cette notion

intervient aussi dans la formalisation de la relation de conformité (cf. définition 7 plus loin). Le blocage défini en section 2.2 est caractérisé par l'entrée dans un état où aucune action n'est possible, ou par l'entrée dans une composante fortement connexe terminale d'états non marqués. Pour les IOLTS, cette définition n'est pas adaptée, à la fois parce que les IOLTS n'ont pas la notion d'état marqué, et que les actions sont partitionnées en entrées et sorties. Nous considérons donc deux types de blocages :

– le *deadlock* lorsque le système ne peut plus évoluer (*i.e.* $\Gamma(q) = \emptyset$) ;

– les *blocages de sortie* lorsque le système est bloqué en attente d'une entrée de l'environnement (*i.e.* $\Gamma(q) \subseteq A_?$).

Du point de vue du test, un blocage dans le système ne constitue une erreur que s'il n'est pas conforme au comportement décrit dans la spécification. Dans notre approche, nous ferons donc une distinction entre les *blocages conformes* (*i.e.* prévus par la spécification) et ceux qui ne le sont pas. Seuls ces derniers devront être éliminés. Formellement, nous modélisons les blocages par une nouvelle action observable : δ.

Définition 6 (automate suspendu) *L'*automate suspendu *d'un IOLTS* $(Q, A, \rightarrow, q_0)$ *est l'IOLTS* $(Q, A^\delta, \rightarrow_\delta, q_0)$ *défini comme suit:* $A^\delta = A \cup \{\delta\}$ *avec* $\delta \in A^\delta_!$ *(l'action* δ *est une sortie, observable par l'environnement) et la relation de transition* $\rightarrow_\delta$ *est obtenue à partir de* $\rightarrow$ *par ajout de boucles* $q \xrightarrow{\delta} q$ *pour tous les états* q *de blocage (deadlock ou blocages de sortie). On note* $\delta(M)$ *l'IOLTS suspendu d'un IOLTS M.* •

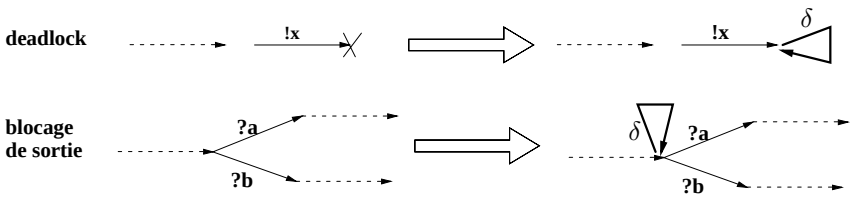

Figure 1. *Calcul des blocages*

Étant donné un IOLTS suspendu $\delta(M)$, il est possible de lui faire correspondre un IOLTS obtenu par l'opération inverse δ^{-1}, consistant à enlever de $\delta(M)$ les transitions étiquetées par l'événement δ et à supprimer l'action δ de l'alphabet du système.

Modèles de la spécification et de l'implémentation. Nous considérons les IOLTS suivants S$= (Q^{\text{S}}, A^{\text{S}}, \rightarrow_{\text{S}}, q_0^{\text{S}})$, la spécification et IMP$= (Q^{\text{IMP}}, A^{\text{IMP}}, \rightarrow_{\text{IMP}}, q_0^{\text{IMP}})$ l'implémentation que nous chercherons à contrôler. Les alphabets d'entrées et de sorties des deux IOLTS sont les mêmes, *i.e.* $A^{\text{S}}_? = A^{\text{IMP}}_?$ et $A^{\text{S}}_! = A^{\text{IMP}}_!$; on dira que les deux IOLTS sont *compatibles*. Notons qu'en faisant l'hypothèse de *connaître le modèle* IOLTS IMP de l'implémentation, nous sommes plus exigeants que la théorie du test de conformité qui se contente de supposer que l'implémentation est *modélisable* par un IOLTS.

2.3.2. *Relation de conformité :* **ioco**.

La relation de conformité que nous considérons est celle définie dans [TRE 96]. C'est la relation qui est la plus utilisée en test de conformité, dans les outils TorX [BEL 99], TGV [JéR 02] ainsi que dans des outils industriels tels que TestComposer de Telelogic.

Définition 7 (relation de conformité ioco) *Soit* S *et* IMP *deux IOLTS compatibles :*

$$\text{IMP } \textbf{ioco } \text{S} \iff \forall \sigma \in \mathcal{L}(\delta(\text{S})), Out(\delta(\text{IMP}) \textit{ after } \sigma) \subseteq Out(\delta(\text{S}) \textit{ after } \sigma)$$

En clair, une implémentation IMP est conforme à sa spécification S si après toute séquence σ de S, incluant éventuellement des blocages, les sorties (et donc les blocages) de IMP sont inclus dans ceux prévus par S.

Exemple 1 *La relation* **ioco** *est illustrée par la figure 2. Les deux premières implémentations* IMP_1 *et* IMP_2 *sont conformes à la spécification* S *:* IMP_1 **ioco** S *car on peut vérifier qu'en tout état, les sorties de* IMP_1 *sont incluses dans celles de* S. **ioco** *permet donc de restreindre les sorties de l'implémentation par rapport à celles prévues dans la spécification.* IMP_2 **ioco** S *car, après la séquence* $!x.!z$, IMP_2 *permet une entrée supplémentaire,* $?b$*, par rapport à* S *mais* **ioco** *l'autorise.* **ioco** *permet donc de considérer des spécifications partielles. Par contre* $\neg(\text{IMP}_3$ **ioco** $S)$ *puisque la sortie* $!y$ *après la séquence* $!x.!z.!x$ *n'est pas autorisée dans la spécification. De même* $\neg(\text{IMP}_4$ **ioco** $S)$ *car le blocage de sortie de* IMP_4 *après la séquence* $!x.!z.!x$ *n'existe pas dans la spécification.*

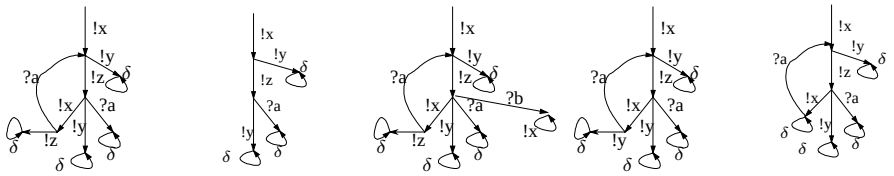

Figure 2. ioco *par l'exemple*

3. Synthèse de contrôleurs pour la conformité

3.1. *Contrôle des IOLTS et problématique du contrôle*

Les IOLTS distinguent les entrées et les sorties. Afin de faire de la synthèse de contrôleurs, il nous faut encore définir la partition des actions en contrôlables et incontrôlables (cf. définition 3). Deux points de vue sont envisageables :

1) *Contrôler les entrées du système*. Cela consiste à scruter les événements envoyés au système par l'environnement, et à éliminer ceux qui peuvent mener à un comportement indésirable. Dans ce cas, le contrôle se situe au niveau de l'environnement ;

2) *Contrôler les sorties du système*. C'est le point de vue dual du précédent. Ici on contrôle, non pas les événements envoyés par l'environnement, mais les réactions du système à ces événements. Dans ce cas, le contrôle est au niveau du système.

Ce second point de vue nous parait plus réaliste. En effet, s'il semble raisonnable de pouvoir modifier le comportement d'un système, prétendre pouvoir contrôler l'environnement dans lequel sera placé ce système est plus discutable. Nous faisons donc le choix de nous placer au niveau du système et de contrôler une partie de ses émissions.

Définition 8 (IOLTS à contrôler) *Un* IOLTS à contrôler *est un IOLTS* $(Q, A = A_? \cup A_!, \rightarrow, q_0)$ *et deux ensembles d'actions* A_c, A_{uc}*, qui vérifient :*

- $A_c \cup A_{uc} = A$, $A_c \cap A_{uc} = \emptyset$ *;*
- $A_? \subseteq A_{uc}$ (i.e. *les entrées du système sont incontrôlables) ;*
- $A_c \subseteq A_!$ (i.e. *un sous-ensemble des sorties du système est contrôlable).* •

Basé sur cette définition, le problème de contrôle que nous posons est le suivant:

(**ioco-control**) *Étant donnés deux IOLTS compatibles : l'implémentation* IMP *et la spécification* S*, calculer un contrôleur admissible* $\mathcal{C}$ *tel que* $\mathcal{C}/\text{IMP}$ **ioco** S *et* $\mathcal{C}/\text{IMP}$ *est le plus permissif, i.e., pour tout contrôleur admissible* $\mathcal{C}'$*,* $\mathcal{L}(\mathcal{C}'/\text{IMP}) \subseteq \mathcal{L}(\text{IMP})$ *et* $\mathcal{C}'/\text{IMP}$ **ioco** S *impliquent* $\mathcal{L}(\mathcal{C}'/\text{IMP}) \subseteq \mathcal{L}(\mathcal{C}/\text{IMP})$.

Les différences entre ce problème de contrôle et celui présenté en section 2.2 tiennent dans l'absence d'états marqués et dans la notion de blocage. Ainsi, *les blocages de l'implémentation prévus dans la spécification devront être conservés*.

Enfin, pour nous ramener à une définition plus simple de la relation de conformité **ioco**, nous supposerons que les spécifications que nous considérons sont *complètes en entrées* (*i.e.* $\forall \sigma \in \mathcal{L}(\text{S}), \forall a \in A_? : \sigma.a \in \mathcal{L}(\text{S})$). Cette hypothèse n'est pas restrictive dans la mesure où il est toujours possible de compléter en entrées une spécification sans changer l'ensemble des implémentations conformes (cf. proposition 1).

Définition 9 (Complétion en entrée) *Soit un IOLTS* $\text{S} = (Q, A, \rightarrow, q_0)$*, l'IOLTS de* S *complété en entrée correspond à* $Comp(\text{S}) = (Q_c, A, \rightarrow_c, q_0)$ *avec :*

- $Q_c = Q \cup \{sink\}$ *;*
- $\rightarrow_c = \rightarrow \cup \{sink \xrightarrow{a} sink \mid a \in A\} \cup \{q \xrightarrow{a} sink \mid a \in A_?, \neg q \xrightarrow{a}\}$.

Remarque 1 *Les deadlocks de* S *ont été remplacés dans* $Comp(\text{S})$ *par des blocages de sortie. On peut aussi montrer que* $\delta(Comp(\text{S})) = Comp(\delta(\text{S}))$. ◇

Lemme 1 (Équivalence de la conformité à S **et à** $Comp(\text{S})$**)** *Soient* IMP *et* S *deux IOLTS. On a :*

IMP **ioco** S $\iff$ IMP **ioco** *Comp(*S*).*

Preuve. ($\Leftarrow$) Supposons $\neg$(IMP **ioco** S). Par définition de **ioco** (définition 7) cela signifie qu'il existe une trace $\sigma \in \mathcal{L}(\delta(\text{S}))$ et $a \in Out(\delta(\text{IMP}) \text{ after } \sigma)$, tels que

l'action $a \notin Out(\delta(\text{S}) \text{ after } \sigma)$. Par définition de $Comp$, nous avons $\mathcal{L}(\delta(\text{S})) \subseteq \mathcal{L}(Comp(\delta(S))) = \mathcal{L}(\delta(Comp(S)))$. Donc $\sigma \in \mathcal{L}(\delta(Comp(S)))$.

Mais $\sigma a \notin \mathcal{L}(\delta(Comp(S)))$, car $a \in A_!$, $a \notin Out(\delta(\text{S}) \text{ after } \sigma)$, et la complétion en entrée n'ajoute à S que des événements appartenant à $A_?$. Par conséquent, $\neg(\text{IMP}$ **ioco** $Comp(\text{S}))$.

($\Rightarrow$) Supposons que $\neg(\text{IMP}$ **ioco** $Comp(\text{S}))$, donc il existe une séquence $\sigma \in \mathcal{L}(\delta(Comp(\text{S})))$, et $a \in Out(\delta(\text{IMP}) \text{ after } \sigma)$ avec $a \notin Out(\delta(Comp(\text{S})) \text{ after } \sigma)$.

– Si $\sigma \in \mathcal{L}(\delta(\text{S}))$, alors $a \notin Out(\delta(Comp(\text{S})) \text{ after } \sigma)$ implique $a \notin Out(\delta(\text{S}) \text{ after } \sigma)$ parce que $\mathcal{L}(\delta(\text{S})) \subseteq \mathcal{L}(\delta(Comp(\text{S})))$. Par conséquent, dans ce cas on a bien $\neg(\text{IMP}$ **ioco** $\text{S})$.

– Autrement, $\sigma \in \mathcal{L}(\delta(Comp(\text{S})) \setminus \mathcal{L}(\delta(\text{S}))$, donc σ est de la forme $\sigma_1 b \sigma_2$ avec $\sigma_1 \in \mathcal{L}(\delta(\text{S}))$ et $b \in A_?$. Dans ce cas, $\sigma_1 b \sigma_2 a \in \mathcal{L}(\delta(Comp(\text{S})))$, par conséquent, on a $a \in Out(\delta(Comp(\text{S}))after\sigma)$, ce qui contredit notre hypothèse de départ. ◇

Dans la suite, nous considérerons toujours que nous travaillons sur une spécification complète en entrée. Sous cette hypothèse, la relation **ioco** peut s'écrire ainsi :

Lemme 2 (Réduction de la conformité à l'inclusion de séquences) *Soient* IMP *et* S *deux IOLTS (avec* S *complet en entrée). On a :*

IMP **ioco** $\text{S} \iff \mathcal{L}(\delta(\text{IMP})) \subseteq \mathcal{L}(\delta(\text{S}))$.

Preuve ($\Rightarrow$) Supposons que IMP **ioco** S. Soit $\sigma \in \mathcal{L}(\delta(\text{IMP}))$ telle que $\sigma \notin \mathcal{L}(\delta(\text{S}))$. Alors, on peut décomposer la séquence σ en $\sigma = \sigma_1\sigma_2$, tel que σ_1 soit le préfixe maximal de σ appartenant à $\mathcal{L}(\delta(\text{S}))$. Soit a la première action de σ_2.

– Si $a \in A_?$, comme $\delta(\text{S})$ est complet en entrée, $\sigma_1 a \in \mathcal{L}(\delta(\text{S}))$, et σ_1 n'est pas maximale ;

– Si $a \in A_! \cup \{\delta\}$, alors comme $\sigma_1 \in \mathcal{L}(\delta(\text{IMP}))$, on a $a \in Out(\delta(\text{IMP}) \text{ after } \sigma_1)$. Or, IMP **ioco** S, on a aussi $a \in Out(\delta(\text{S}) \text{ after } \sigma_1)$, i.e., $\sigma_1 a \in \mathcal{L}(\delta(\text{S}))$, et σ_1 n'est pas maximale.

($\Leftarrow$) Cette implication est triviale. ◇

Remarque 2 *Il est maintenant facile de voir que, si* IMP_1 **ioco** S *et* IMP_2 **ioco** S*, alors* $(\text{IMP}_1 \cup \text{IMP}_2)$ **ioco** S.[1]

3.2. *Algorithme de synthèse*

Nous avons défini le modèle avec lequel nous travaillons, les IOLTS, et nous avons posé le problème auquel nous nous intéressons : contrôler une implémentation pour la rendre conforme, au sens de la relation de conformité **ioco**, à une spécification. Nous allons maintenant présenter les algorithmes permettant de traiter notre problème.

1. L'opération $\cup$ entre IOLTS est similaire à l'union d'automates, i.e., $A \cup B$ est un IOLTS tel que $\mathcal{L}(A \cup B) = \mathcal{L}(A) \cup \mathcal{L}(B)$.

3.2.1. *Calcul de l'implémentation la plus permissive conforme par rapport à* S.

Soient IMP et S deux IOLTS jouant le rôle d'implémentation et de spécification d'un système. Nous allons d'abord construire, *sans tenir compte des problèmes de contrôlabilité*, un IOLTS G^{ioco} modélisant un sous comportement de IMP ($\mathcal{L}(G^{ioco}) \subseteq \mathcal{L}(\text{IMP})$), tel que G^{ioco} **ioco** S et G^{ioco} soit le plus permissif (au sens de l'inclusion de langages). De manière à rendre IMP conforme à S, nous sommes amenés à faire le produit entre ces deux IOLTS, afin de conserver seulement les comportements admis à la fois par IMP et par S (cela permet en particulier de régler le problème des sorties non prévues par S). Il faut également tenir compte des blocages (les seuls admissibles sont ceux de la spécification) :

Lemme 3 *Soit G^{ioco} l'IOLTS le plus permissif assurant la conformité et t.q. :* $\mathcal{L}(G^{ioco}) \subseteq \mathcal{L}(\text{IMP})$. *Alors,* $\mathcal{L}(\delta(G^{ioco})) \subseteq \mathcal{L}(\text{IMP} \parallel \delta(S))$.

Preuve : Remarquons d'abord que G^{ioco} existe : c'est l'union de tous les IOLTS qui sont conformes à S et inclus dans IMP ; par la remarque 2, la conformité est préservée par l'union. Soit alors $\Delta(\text{IMP})$ l'IOLTS obtenu à partir de IMP par ajout d'une boucle δ sur chaque état de IMP. Par définition de $\parallel$, on a alors $\text{IMP} \parallel \delta(S) = \Delta(\text{IMP}) \parallel \delta(S)$. Comme $\mathcal{L}(G^{ioco}) \subseteq \mathcal{L}(\text{IMP})$, on a $\mathcal{L}(\delta(G^{ioco})) \subseteq \mathcal{L}(\Delta(\text{IMP}))$. De plus, G^{ioco} **ioco** S. Donc, d'après le lemme 2, $\mathcal{L}(\delta(G^{ioco})) \subseteq \mathcal{L}(\delta(S))$.

On en déduit que $\mathcal{L}(\delta(G^{ioco})) \subseteq \mathcal{L}(\Delta(\text{IMP})) \cap \mathcal{L}(\delta(S)) = \mathcal{L}(\text{IMP} \parallel \delta(S))$. ◇

Lorsque nous effectuons le produit entre IMP et δ(S), il nous faut donc ne garder que les blocages prévus par la spécification. Pour cela, nous introduisons l'opération CUT qui consiste à éliminer les blocages dans un IOLTS.

Définition 10 *Soit $G = (Q, A^{\delta}, \rightarrow, q_0)$, un IOLTS, alors* CUT$(G)$ *est le résultat du point fixe suivant:*

$$\begin{cases} G_o & = G \\ G_{i+1} & = (Q_{i+1}, A^{\delta}, \rightarrow_{i+1}, q_0), \text{ avec} \quad Q_{i+1} = Q_i \setminus \{q \in Q_i \mid \Gamma(q) \subseteq A_?\} \\ & \qquad\qquad\qquad\quad \rightarrow_{i+1} = \rightarrow_i \setminus \{q \xrightarrow{a} q' | q' \in Q_i \setminus Q_{i+1}, a \in A\} \end{cases}$$

D'où l'on peut déduire immédiatement :

Lemme 4 $\delta(\delta^{-1}(\text{CUT}(G))) \subseteq \text{CUT}(G)$.

Intuitivement, l'opération CUT permet d'enlever récursivement tous les états d'un IOLTS qui sont des blocages dans lesquels l'événement δ n'est pas tirable, *i.e.* soit des blocages de sortie ($\Gamma(q) \subseteq A_?$), soit des deadlock.

Le lemme 4 découle du fait que les seuls blocages restant dans $\delta^{-1}(\text{CUT}(G))$ sont ceux de CUT(G) dans lesquels un δ est tirable.

Il est maintenant possible de montrer le résultat suivant :

Proposition 1 *Soient* IMP, *l'implémentation, et* S, *la spécification, deux IOLTS. Soit* $G = \text{IMP} \parallel \delta(\text{S})$ *(cf. définition 2), alors*

1) $\delta^{-1}(\text{CUT}(G))$ **ioco** S

2) $\delta^{-1}(\text{CUT}(G))$ *est l'IOLTS le plus permissif qui soit conforme à* S[2].

Preuve de la proposition 1 : Soit $G' = \text{CUT}(G)$. D'après le lemme 4, $\delta(\delta^{-1}(G')) \subseteq G'$. Par conséquent, l'inclusion suivante est vérifiée : $\mathcal{L}(\delta(\delta^{-1}(G))) \subseteq \mathcal{L}(G')$. Comme de plus, $\mathcal{L}(G') \subseteq \mathcal{L}(G) \subseteq \mathcal{L}(\delta(\text{S}))$, on en déduit d'après la le lemme 1 que $\delta^{-1}(G')$ **ioco** S.

Montrons maintenant le point *2.* Pour cela, on raisonne par récurrence sur le point fixe de l'algorithme CUT (cf. définition 10). D'après le lemme 3, $\mathcal{L}(\delta(G^{ioco})) \subseteq \mathcal{L}(G_0)$. Supposons maintenant que $\mathcal{L}(\delta(G^{ioco})) \subseteq \mathcal{L}(G_i)$. On cherche à montrer que $\mathcal{L}(\delta(G^{ioco})) \subseteq \mathcal{L}(G_{i+1})$. Soit $q \in Q_i \setminus Q_{i+1}$, et une séquence arbitraire σ telle que G_i after $\sigma = \{q\}$. Comme $q \in Q_i \setminus Q_{i+1}$, la définition 10 implique que $\Gamma(G_i \text{ after } \sigma) \subseteq A_?$. Supposons que $\sigma \in \mathcal{L}(\delta(G^{ioco}))$ alors, comme $\mathcal{L}(\delta(G^{ioco})) \subseteq \mathcal{L}(G_i)$, on en déduit que $\Gamma(\delta(G^{ioco}) \text{ after } \sigma) \subseteq A_?$. Ce qui est absurde, car il y aurait alors dans G^{ioco} un blocage de sortie qui n'aurait pas été tagué par l'opération δ. On en déduit donc que $\sigma \notin \mathcal{L}(\delta(G^{ioco}))$.

En d'autres termes, $\mathcal{L}(\delta(G^{ioco})) \subseteq \mathcal{L}(G_i) \setminus \{\sigma A^* | q_o \xrightarrow{\sigma}_i q, q \in Q_i \setminus Q_{i+1}\} = \mathcal{L}(G_{i+1})$. En particulier, une fois le point-fixe atteint, $\mathcal{L}(\delta(G^{ioco})) \subseteq \mathcal{L}(\text{CUT}(G))$ ou encore $\mathcal{L}(G^{ioco}) \subseteq \mathcal{L}(\delta^{-1}(\text{CUT}(G)))$. D'où $\delta^{-1}(\text{CUT}(G))$ est bien le plus permissif pour **ioco** ◇

La propriété 1 montre que $\delta^{-1}(\text{CUT}(G))$, avec $G = \text{IMP} \parallel \delta(\text{S})$ est l'IOLTS le plus permissif qui soit **ioco** à S. Toutefois, il n'est pas garanti que cette nouvelle implémentation soit *contrôlable* par rapport à IMP. En effet, l'opération CUT peut mener à couper des transitions incontrôlables. Le résultat n'est donc plus forcément contrôlable. Ce problème est traité dans la section suivante.

3.2.2. *Algorithme général prenant en compte la contrôlabilité*

Notons dans un premier temps que le fait d'extraire de $\delta^{-1}(\text{CUT}(G))$, la plus grande partie *contrôlable* n'est pas suffisant pour assurer le résultat car cela peut faire perdre le caractère *conforme* du résultat. Cela peut en effet induire de nouveaux blocages dans l'implémentation, blocages non prévus par la spécification. En fait, il est possible de montrer que les propriétés de conformité et de contrôlabilité sont antagonistes (le fait d'assurer l'une peut invalider la seconde). Ainsi, dans le but d'obtenir une implémentation qui soit contrôlable par rapport à IMP et **ioco** par rapport à S, il est nécessaire d'itérer le calcul rendant l'implémentation **ioco** et celui la rendant contrôlable. Ceci nous amène à l'algorithme suivant :

2. $\forall G' \; t.q. \; G'$ **ioco** $\text{S}, \mathcal{L}(G') \subseteq \mathcal{L}(\delta^{-1}(\text{CUT}(G)))$

Algorithme PG-IOCO-CONTRÔLABLE

ENTRÉES: IMP$=(Q^{\text{imp}}, A^{\text{imp}}, \rightarrow_{\text{imp}}, q_0^{\text{imp}})$ et S$=(Q^{\text{s}}, A^{\text{s}}, \rightarrow_{\text{s}}, q_0^{\text{s}})$

SORTIE : $\mathcal{C}/$IMP l'implémentation contrôlée

(i) Calcul de l'IOLTS suspendu $\delta(\text{S})$ avec $\delta \in A^{\delta}_{!} \cap A^{\delta}_{uc}$
(ii) Soit $i = 0$ et $G_i = \text{IMP} \parallel \delta(\text{S}) = (Q_i, A^{\delta}, \rightarrow_i, q_{i_0})$
(iii) $G'_i = \text{CUT}(G_i)$
(iv) $G_{i+1} = \text{CONTROLLABLE}(G'_i, \delta(\text{IMP}), A^{\delta}_{uc})$. (c.f. algorithme SUPCONT)
(v) Si $G_{i+1} \neq G'_i$ Goto (iii) sinon $\mathcal{C}/\text{IMP} = \delta^{-1}(G'_i)$.

Le résultat principal de cet article est que l'algorithme PG-IOCO-CONTRÔLABLE calcule bien le contrôleur attendu :

Proposition 2 *$\mathcal{C}/$IMP est l'IOLTS le plus permissif tel que $\mathcal{C}/$IMP est conforme par rapport à* S *et contrôlable par rapport à* IMP *et* A_{uc}.

Avant de prouver cette proposition, nous avons besoin du lemme suivant:

Lemme 5 *Avec les notations de l'algorithme* PG-IOCO-CONTRÔLABLE,

1) $\forall i, \delta^{-1}(G'_i)$ **ioco** S.

2) $\forall i > 0, \delta^{-1}(G_i)$ *est contrôlable par rapport à* IMP *et* A_{uc}.

3) $\forall i, \delta^{-1}(G_i) \parallel \delta(S) = G_i$

Preuve: La preuve du point *1.* est similaire à celle du point *1.* de la proposition 1. La preuve du point *2.* vient du fait que G_i est contrôlable par rapport à $\delta(\text{IMP})$ et A^{δ}_{uc} et que l'opération $\delta^{-1}(.)$ préserve la contrôlabilité. La preuve du point *3.* se fait par induction sur i :

– Cas de base : $i = 0$. Par définition, $G_0 = \text{IMP} \parallel \delta(\text{S})$. On a également, par définition de $\parallel$ et de δ, $\delta^{-1}(\text{IMP} \parallel \delta(\text{S})) = \text{IMP} \parallel \text{S}$ (car IMP $\parallel$ S ne contient pas de δ). D'où, $\delta^{-1}(G_0) \parallel \delta(\text{S}) = (\text{IMP} \parallel \text{S}) \parallel \delta(\text{S}) = \text{IMP} \parallel (\text{S} \parallel \delta(\text{S})) = \text{IMP} \parallel \delta(\text{S}) = G_0$.

– Hypothèse de récurrence : $\delta^{-1}(G_i) \parallel \delta(S) = G_i$ pour $i > 0$.

– Prouvons $\delta^{-1}(G_{i+1}) \parallel \delta(S) = G_{i+1}$. Par définition de CUT et de CONTROLLABLE, $G_{i+1} \subseteq G_i$, et donc (1) $\delta^{-1}(G_{i+1}) \parallel G_i = G_{i+1}$ [3]. De plus, par hypothèse de récurrence, $\delta^{-1}(G_i) \parallel \delta(S) = G_i$, on en déduit donc (2) $\delta^{-1}(G_{i+1}) \parallel G_i = \delta^{-1}(G_{i+1}) \parallel \delta^{-1}(G_i) \parallel \delta(S)$. Comme on a également, par construction, $G_{i+1} \subseteq G_i$, et donc $\delta^{-1}(G_{i+1}) \parallel \delta^{-1}(G_i) = \delta^{-1}(G_{i+1})$, on déduit de (1) et de (2) $\delta^{-1}(G_{i+1}) \parallel \delta(S) = G_{i+1}$.

On a donc bien $\forall i,\ \delta^{-1}(G_i) \parallel \delta(S) = G_i$. ◇

Preuve de la proposition 2 Le fait que $\mathcal{C}/$IMP soit **ioco** par rapport à S et contrôlable par rapport à IMP et A_{uc} provient directement des points *1.* et *2.* du lemme 5 une fois

3. Plus généralement : Soit M un IOLTS, $\delta(M)$ son IOLTS suspendu et $M' \subseteq \delta(M)$, t.q. (Hyp): ($\forall q \in Q_{M'} | q \xrightarrow{\delta}_M q$ *implique* $q \xrightarrow{\delta}_{M'} q$), alors $\delta^{-1}(M') \parallel \delta(M) = M'$.

le point-fixe atteint. Montrons par récurrence sur i que $\mathcal{C}/\text{IMP}$ est le plus permissif. On note G_c^{ioco} l'IOLTS t.q. 1) $\mathcal{L}(G_c^{ioco}) \subseteq \mathcal{L}(\text{IMP})$ et G_c^{ioco} contrôlable par rapport à IMP et A_{uc} et 2) G_c^{ioco} est le plus permissif pour **ioco**. Il est facile de voir que $\mathcal{L}(G_c^{ioco}) \subseteq \mathcal{L}(G^{ioco})$, avec G^{ioco} défini comme en Section 3.2.1. D'après la proposition 1, $\mathcal{L}(\delta(G_c^{ioco})) \subseteq \mathcal{L}(G'_0) = \mathcal{L}(\text{CUT}(G_0))$. Par définition, G_1 est l'IOLTS le plus permissif assurant la contrôlabilité de G'_0. Donc tous les autres IOLTS contrôlables ayant un comportement dans celui de G'_0 ont un comportement inclus dans celui de G_1, d'où $\mathcal{L}(\delta(G_c^{ioco})) \subseteq \mathcal{L}(G_1)$.

Supposons maintenant que $\mathcal{L}(\delta(G_c^{ioco})) \subseteq \mathcal{L}(G_i)$ et montrons que $\mathcal{L}(\delta(G_c^{ioco})) \subseteq \mathcal{L}(G_{i+1})$.

Soit $G'_i = \text{CUT}(G_i)$. D'après la proposition 1 appliquée à $\delta^{-1}(G_i)$, jouant le rôle de l'implémentation, et à S, $\delta^{-1}(\text{CUT}(\delta^{-1}(G_i) \parallel \delta(S)))$ **ioco** S. Or, d'après le lemme 5, point *3.*, $\delta^{-1}(G_i) \parallel \delta(S) = G_i$ et donc $\delta^{-1}(G'_i)$ **ioco** S. La même proposition nous dit que $\delta^{-1}(G'_i)$ est le plus grand (i.e. le plus permissif) **ioco** qui soit inclus dans $\delta^{-1}(G_i)$. En particulier, il contient tous les autres comportements **ioco** par rapport à S. Donc, comme $\mathcal{L}(\delta(G_c^{ioco})) \subseteq \mathcal{L}(G_i)$, $\mathcal{L}(\delta(G_c^{ioco})) \subseteq \mathcal{L}(G'_i)$.

Soit $G_{i+1} = \text{CONTROLLABLE}(G'_i, \delta(\text{IMP}), A_{uc}^{\delta})$. Alors $\mathcal{L}(G_{i+1})$ contient tous les langages contrôlables inclus dans $\mathcal{L}(G'_i)$ et, en particulier, $\mathcal{L}(\delta(G_c^{ioco})) \subseteq \mathcal{L}(G_{i+1})$.

Finalement, pour j t.q. $G_j = G_{j+1}$, $\mathcal{L}(\delta(G_c^{ioco})) \subseteq \mathcal{L}(G_j)$ et donc $\mathcal{L}(G_c^{ioco}) \subseteq \mathcal{L}(\delta^{-1}(G_j)) = \mathcal{L}(\mathcal{C}/\text{IMP})$. ◇

Exemple 2 *Considérons les deux IOLTS* IMP *et* S[4] *suivants.* IMP *n'est pas* **ioco** *par rapport à* S, *puisque la sortie !y après la séquence !x!z!x n'est pas autorisée par la spécification. On suppose dans cet exemple que* $A_c = \{!y, !z\}$.

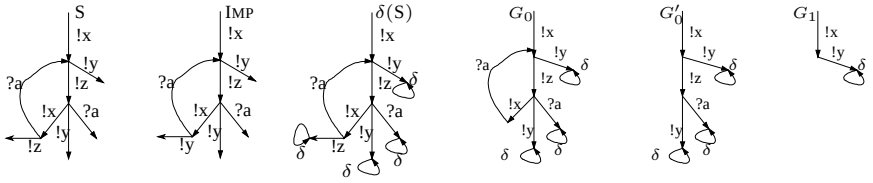

Figure 3. *Exemple de synthèse*

Soit $G_0 = \text{IMP} \parallel \delta(\text{S})$. *Il est facile de voir que* $\delta^{-1}(G_0)$ *n'est pas* **ioco** *par rapport à* S. *Il existe en particulier un blocage de sortie après la séquence !x!z!x qui n'apparaît pas dans* S. *L'opération* $G'_0 = \text{CUT}(G_0)$ *a pour effet de faire disparaître ce blocage. Toutefois, cette opération n'a pas tenu compte du fait que !x était incontrôlable. Soit* $G_1 = \text{CONTROLLABLE}(G'_0, \text{IMP}, A_{uc})$, *le plus grand contrôlable IOLTS résultant. Il est alors facile de vérifier que* $\delta^{-1}(G_1)$ *est à la fois contrôlable par rapport à* IMP *et* **ioco** *par rapport à* S.

4. Pour ne pas alourdir inutilement l'exemple, la spécification complétée en entrées n'est pas représentée.

4. Conclusion et perspectives

Cet article présente un algorithme de synthèse de contrôleurs pour rendre une implémentation d'un système conforme à sa spécification. La différence avec le test de conformité classique est que nous rendons l'implémentation conforme par le contrôle ; et la différence avec la synthèse de contrôleurs classique réside dans le critère de contrôle, qui est la conformité par rapport à une spécification. Une autre originalité par rapport à d'autres approches en synthèse de contrôleurs réside dans le traitement des blocages : en contrôle non-bloquant classique, le système contrôlé n'a pas le droit de s'arrêter dans un état non final (i.e., non marqué) ; dans notre approche, cet arrêt est permis dès lors qu'il est autorisé par la spécification (de même les composantes fortement connexe terminales et non marquées sont admises). Une autre possibilité est d'introduire la notion d'état marqué dans la spécification, et d'autoriser le système contrôlé à s'arrêter uniquement dans les états marqués ; pour cela il suffit de remplacer dans l'algorithme PG-Ioco-Contrôlable (cf. Section 3.2.2), l'opération CONTROLLABLE par l'algorithme complet de synthèse non-bloquante SUPCONT (cf. Section 2.2).

Cette première tentative de rapprochement entre le domaine du test de conformité et celui du contrôle nous ouvre donc de nombreuses perspectives. Nous travaillons sur l'extension de notre approche aux systèmes partiellement observables. Cette extension amène un nouveau type de blocage provenant de boucles d'événements internes au système. Le contrôle de ces boucles n'est pas trivial dès lors que l'on souhaite garder une notion de maximalité du système contrôlé. Nous envisageons également d'étudier un moyen de générer des cas de test dans le même temps que le calcul d'un contrôleur, afin d'obtenir des tests aussi pertinents que possibles sur le contrôle effectif de l'implémentation. Enfin, nous prévoyons d'étendre notre approche à d'autres relations de conformité et d´effectuer des études de cas afin de confirmer la pertinence de notre approche.

5. Bibliographie

[BEL 99] BELINFANTE A., FEENSTRA J., DE VRIES R., TRETMANS J., GOGA N., FEIJS L., MAUW S., « Formal test automation: a simple experiment. », International Workshop on the Testing of Communicating Systems (IWTCS'99), 1999, p. 179-196.

[CAS 99] CASSANDRAS C., LAFORTUNE S., *Introduction to Discrete Event Systems*, Kluwer Academic Publishers, 1999.

[JéR 02] JÉRON T., « TGV: théorie, principes et algorithmes », *Techniques et Sciences Informatiques*, , 2002.

[RAM 89] RAMADGE P. J., WONHAM W. M., « The Control of Discrete Event Systems », *Proceedings of the IEEE; Special issue on Dynamics of Discrete Event Systems*, vol. 77, n° 1, 1989, p. 81–98.

[TRE 96] TRETMANS J., « Test Generation with Inputs, Outputs and Repetitive Quiescence », *Software—Concepts and Tools*, vol. 17, n° 3, 1996, p. 103–120, Springer-Verlag.