Contrôle de systèmes symboliques, discrets ou hybrides

Tristan Le Gall* — **Bertrand Jeannet*** — **Hervé Marchand***

* *IRISA, Campus universitaire de Beaulieu*
F35042 Rennes cedex
{tristan.le_gall,bertrand.jeannet,herve.marchand}@irisa.fr

RÉSUMÉ. *Nous abordons le problème de la synthèse de contrôleurs à travers différents modèles allant des systèmes de transitions finis aux systèmes hybrides en nous intéressant à des propriétés de sûreté. Dans ce cadre, nous nous intéressons principalement au problème de synthèse pour un modèle intermédiaire : les systèmes de transitions symboliques. L'analyse des besoins de modélisation nous amène à redéfinir la notion de contrôlabilité en faisant porter le caractère de contrôlabilité non plus sur les événements mais sur les gardes des transitions, puis à définir des algorithmes de synthèse permettant l'usage d'approximations et d'assurer la terminaison des calculs. Nous généralisons par la suite notre méthodologie au contrôle de systèmes hybrides, ce qui donne un cadre unifié du problème de la synthèse pour un ensemble consistant de modèles.*

ABSTRACT. *In this paper, we tackle the safety controller synthesis problem for various models (from finite transition systems to hybrid systems). Within this framework, we are mainly interested in the synthesis problem for an intermediate model: the symbolic transition system. Modelization needs lead us to redefine the concept of controllability by associating it to guards of transitions instead of events. We then define synthesis algorithms based on abstract interpretation techniques so that we can ensure finiteness of the computations. We finally generalize our methodology to the control of hybrid systems, which gives an unified framework to the supervisory control problem for several classes of models.*

MOTS-CLÉS : *systèmes de transition finis, hybrides et symboliques, synthèse de contrôleurs, propriétés de sûreté, interprétation abstraite.*

KEYWORDS: *finite state machine, hybrid and symbolic systems, supervisory control problem, safety properties, abstract interpretation.*

RSTI - TSI – 25/2006. Contrôle temporisé, pages 293 à 319

1. Introduction

Parmi les différentes méthodes de conception et de validation, la synthèse de contrôleur est sans doute l'une des plus séduisantes. Elle permet en effet de raffiner une spécification incomplète de manière à atteindre un certain objectif, typiquement la satisfaction d'une propriété non encore vérifiée sur le système initial. Le raffinement consiste dans ce cadre à contraindre une spécification afin d'atteindre l'objectif fixé. En ce sens, la synthèse procède par élimination de comportements indésirables, et dans son cadre classique ne cherche pas à ajouter ou à inventer de nouveaux comportements au système initial.

La synthèse de contrôleur est un domaine de recherche bien établi, qui se trouve à la frontière de plusieurs disciplines :

1) L'automatique discrète, qui étudie les systèmes à événements discrets ;

2) l'informatique fondamentale, et plus spécifiquement le model-checking ;

3) l'automatique des systèmes régis par des équations différentielles.

Enfin, mentionnons ausi l'approche théorie des jeux du problème (Tomlin *et al.*, 2000).

Les deux premières disciplines ont comme point commun d'utiliser le même type de modèle pour modéliser les systèmes à contrôler et les propriétés à assurer, à savoir des modèles à base d'automates qui définissent des langages formels sur des alphabets d'évènements (Ramadge *et al.*, 1989; Vardi *et al.*, 1986). Une idée fondamentale de cette approche est l'équivalence entre certaines classes de langages formels et leurs représentations à base d'automates (sans oublier leurs représentations à base de formules de logique temporelle (Pnueli, 1977)). Une autre caractéristique est que ces modèles à base d'automates ne sont pas symboliques, au sens où ils n'intègrent pas la notion de variable/donnée et d'opérations algébriques sur ces données.

En revanche, le contrôle de systèmes régis par des équations différentielles a adopté depuis le milieu des années 90 le modèle des systèmes hybrides (Alur *et al.*, 1995). Le terme hybride signifie ici que ces systèmes combinent un comportement discret, modélisant par exemple un changement de mode de fonctionnement, avec un comportement continu, modélisant par exemple la loi de commande d'un bras articulé d'un robot. Les systèmes hybrides sont des modèles symboliques, au sens où ils définissent l'évolution de variables (numériques), évolution qui peut être discrète (par ex., l'incrémentation d'un compteur) ou continue (évolution d'une variable définie par une équation différentielle). Bien qu'on puisse attribuer aux systèmes hybrides une sémantique en terme de langage, il est plus courant d'adopter la vision mathématique classique, dans laquelle on s'intéresse aux propriétés portant sur l'état du système (valeur des variables et des fonctions).

En raison de ces traditions diverses, le problème du contrôle est souvent posé d'une manière sensiblement différente selon que l'on s'intéresse à des systèmes discrets ou à des systèmes hybrides. Par exemple, la notion d'événement contrôlable ou incontrôlable, essentielle dans le contrôle discret, est généralement moins explicitée dans le contrôle hybride. L'objet de cet article est de proposer une cadre unificateur au

contrôle des systèmes discrets et hybrides, cadre qui concerne à la fois les modèles et la spécification des problèmes de contrôle sur ces modèles.

Dans la suite de cette introduction, nous précisons notre définition d'un problème de contrôle et les critères qui interviennent pour classifier différents problèmes de contrôle, avant d'annoncer le plan et de préciser les contributions de cet article.

Problématique du contrôle. Le problème de la synthèse de contrôleur (Ramadge *et al.*, 1987) peut se résumer ainsi : étant donné un système, qui modélise un programme ou un système « réel », comment forcer ce système à respecter ses spécifications, en restreignant son comportement *le moins possible* ? Le critère d'optimalité souligné est important : on veut garder le plus de comportements possibles du système initial. Une première question qui se pose est l'implémentation du système contrôlé :

1) Le contrôle est-il effectué de manière interne ou externe ?

Dans le *contrôle interne*, on modifie le système initial afin d'obtenir le modèle contrôlé. La vision adoptée est celle du raffinement d'une spécification incomplète, afin de la rendre correcte vis-à-vis d'exigences bien définies. Il s'agit typiquement de modifier les transitions du système. Dans le *contrôle externe*, le comportement du système initial est contraint à l'aide d'un superviseur qui a des moyens d'action et d'observation sur le système. Il s'avère en fait que les problèmes sont souvent équivalents, mais il convient de spécifier le point de vue adopté. La communauté systèmes à événements discrets adopte le point de vue du contrôle externe, tandis que la communauté systèmes hybrides utilise fréquemment le contrôle interne.

La vision adoptée par le contrôle externe amène naturellement les questions suivantes :

2) Quels sont les moyens d'observation du superviseur ?

3) Quels sont ses moyens d'actions ?

Lorsqu'on s'intéresse au contrôle interne, le premier critère devient sans objet : on suppose que tout est observable. En contrôle externe en revanche, l'état interne n'est pas directement observable, ainsi que certains événements lorsque l'on considère le contrôle *sous observation partielle* (Cassandras *et al.*, 1999). Enfin, même dans le cas du contrôle interne, le second critère reste pertinent, avec la notion d'événement incontrôlable sur lequel il n'y a pas de moyen d'action. Là encore, les deux communautés que nous avons identifiées diffèrent sur la définition et la modélisation des moyens de contrôle et d'observation.

Bien sûr, pour définir précisément un problème de contrôle selon les critères énoncés ci-dessus, il faut préciser le modèle considéré pour représenter un système, ainsi que le formalisme utilisé pour les propriétés à assurer grâce au contrôle.

Algorithmes et interprétation abstraite. Une fois un problème de contrôle bien posé, il s'agit de calculer le système contrôlé le plus permissif (dans le cas du contrôle interne). Ce calcul peut généralement se ramener à la résolution d'une équation de point-fixe sur le treillis complet des ensembles d'états du système. Le calcul du point fixe est cependant indécidable, à moins que l'espace des états ne soit fini ou ne jouisse de propriétés particulières. Cet obstacle peut toutefois être contourné par l'utilisation

d'approximations, dans le cadre théorique de l'interprétation abstraite. Cela permet d'obtenir une solution correcte, au prix de la perte du caractère optimal du système contrôlé obtenu.

Plan et contribution. Cet article propose d'unifier dans un modèle unique les principales présentations du problème de contrôle apparaissant dans la bibliographie sur le contrôle des systèmes à événements discrets et des systèmes hybrides, et définit des algorithmes adaptés au cas. Nous commencerons par présenter en section 2 les trois modèles de systèmes que nous allons considérer, dans l'ordre croissant de leur expressivité :

– les systèmes de transitions étiquetés (LTS), qui sont des systèmes à événements discrets, qui utilisent les notions d'états, d'événement, et de transition entre états ;

– les systèmes de transitions symboliques (STS), qui sont des systèmes discrets définis à l'aide de variables et d'opérations sur ces variables (tests, affectations) (Jeannet *et al.*, 2005). Le point important ici est l'introduction d'une distinction entre syntaxe et sémantique, et le fait que l'ensemble des états devient potentiellement infini ;

– les systèmes de transitions hybrides (HTS) ; qui étendent les STS par l'ajout d'un comportement continu de certaines variables entre deux transitions discrètes (Alur *et al.*, 1995).

La sémantique de ces systèmes sera définie en terme de traces (ou de trajectoires pour les HTS). Nous ne considérerons explicitement pas le modèle des automates temporisés (Alur *et al.*, 1994), cas particulier des HTS qui a été étudié (Maler *et al.*, 2005).

La section 3 détaille les propriétés que nous considérerons pour le contrôle. Il s'agit essentiellement des propriétés de sûreté, définies en terme d'ensemble de traces ou d'exécutions, ainsi que des propriétés de vivacité particulières comme celle de non blocage. Comme les propriétés de sûreté peuvent se ramener à des propriétés d'interdiction d'états par l'usage d'observateurs, nous nous focaliserons sur les propriétés d'interdiction d'états (ou par dualité, sur les propriétés d'invariance). Nous discutons ensuite le contrôle de ces systèmes selon les critères énoncés dans le paragraphe précédent. Nous rappellerons d'abord le cas bien connu du contrôle des LTS en section 4. Nous aborderons ensuite une modélisation très générale du problème de contrôle dans le cadre des STS en section 5. Nous discuterons aussi des solutions algorithmiques pour la synthèse, et montrerons comment utiliser de manière adaptée la théorie de l'interprétation abstraite pour contourner le caractère indécidable du problème de contrôle dans ce cadre. Enfin, nous aborderons en section 6 le contrôle des systèmes hybrides, en intégrant la modélisation adoptée pour les STS à la notion classique de la contrôlabilité utilisée par la communauté des systèmes hybrides.

2. Les différents types de modèles étudiés

Nous présentons ici les trois modèles d'automates que nous allons utiliser dans la suite : les systèmes de transitions finis en section 2.1, les systèmes de transitions symboliques en section 2.2 et les automates hybrides généralisés en section 2.3.

2.1. *Les systèmes de transitions finis*

Le modèle des systèmes de transitions finis est le formalisme le plus simple : un tel système a un nombre fini d'états, et passe d'un état à un autre lorsque se produit un événement atomique.

Définition 1 *Un LTS est un quadruplet $\mathcal{M} = (Q, Q_o, \Lambda, \rightarrow)$ où Q est un ensemble fini d'états, $Q_o \subseteq Q$ sont les états initiaux, Λ est l'alphabet des événements, et $\rightarrow \subseteq Q \times \Lambda \times Q$ est la relation de transition.*

Un LTS est dit fini si Q et A sont finis, infini sinon. Soit $\mathcal{M} = (Q, Q_o, \Lambda, \rightarrow)$ un LTS. Nous utiliserons les notations $q \xrightarrow{a} q'$ pour $(q, a, q') \in \rightarrow$, $q \xrightarrow{a}$ pour $\exists q' : q \xrightarrow{a} q'$. Une exécution est une séquence $q = q_0 \xrightarrow{\sigma_0} q_1 \xrightarrow{\sigma_1} \cdots \xrightarrow{\sigma_{n-1}} q_n$ avec $q_0 \in Q_o$. Une trace est la projection d'une exécution sur les événements. La relation de transition $\rightarrow$ est étendue à une séquence d'événements $\sigma = \sigma_1 \ldots \sigma_n$: $q \xrightarrow{\sigma} q' \Leftrightarrow \exists q_0, \ldots, q_n : q = q_0 \xrightarrow{\sigma_0} q_1 \xrightarrow{\sigma_1} \ldots q_n = q'$. Un état q est dit *bloquant* si $\neg(\exists a \in \Lambda : q \xrightarrow{a})$. Le système est dit bloquant s'il existe une exécution menant dans un état bloquant. Nous nous restreindrons à des systèmes déterministes, pour lesquels la relation de transition est une fonction de signature $Q \times \Lambda \rightarrow Q$.

La sémantique d'un LTS est définie par son ensemble de trace $\mathcal{L}(\mathcal{M})$.

2.2. *Systèmes de transitions symboliques*

Un système de transitions symbolique (STS) est un système discret étendu avec des variables. La figure 1 en donne un exemple. Ce modèle permet de représenter des systèmes infinis, lorsque les variables prennent leur valeur dans des domaines infinis. Un STS permet typiquement de modéliser un programme non-récursif (ou une procédure) sans allocation dynamique, qui interagit avec son environnement. Les localités de l'automate ne seront pas explicitées dans le formalisme, celles-ci pouvant être codées par l'intermédiaire d'une variable énumérée type *compteur de programme*.

Notations. Pour une variable v, nous notons $\mathcal{D}_v$ le domaine de valeurs associé. Nous étendons cette notation aux vecteurs et aux ensembles de variables : si $\vec{v} = \langle v_1, \ldots, v_n \rangle$ et $V = \{v_1, \ldots, v_n\}$, $\mathcal{D}_{\vec{v}} = \mathcal{D}_V = \mathcal{D}_{v_1} \times \cdots \times \mathcal{D}_{v_n}$. Une condition ou une garde $G(\vec{v})$ portant sur les variables $\vec{v}$ sera manipulée suivant le contexte comme un prédicat ou comme le sous-ensemble de $\mathcal{D}_{\vec{v}}$ des valuations de $\vec{v}$ qui satisfont le prédicat. Nous désignons par $\bar{E}$ le complémentaire d'un ensemble $E \subseteq D$ inclus dans un domaine D. Pour un prédicat $G(\vec{v}, \vec{p})$, la projection $\exists \vec{p} : G(\vec{v}, \vec{p})$ s'écrit en notation ensembliste $Proj_{\mathcal{D}_{\vec{v}}}(G) = \{\vec{\nu} \mid \exists \vec{\pi} \in \mathcal{D}_{\vec{p}} : (\vec{\nu}, \vec{\pi}) \in G\}$. La projection universelle

$\forall \vec{p} : G(\vec{v}, \vec{p})$ s'écrit en notation ensembliste $Proj^{\forall}_{\mathcal{D}_{\vec{v}}}(G) = \{\vec{\nu} \mid \forall \vec{\pi} \in \mathcal{D}_{\vec{p}} : (\vec{\nu}, \vec{\pi}) \in G\}$. On a $Proj^{\forall}_{\mathcal{D}_{\vec{v}}}(G) = \overline{Proj_{\mathcal{D}_{\vec{v}}}(\overline{G})}$. Pour une quelconque fonction $f : E \to F$ et $Y \subseteq F$, on note $f^{-1}(Y)$ l'image réciproque $\{e \in E \mid f(e) \in Y\}$.

Définition 2 (STS, syntaxe) *(Jeannet* et al.*, 2005) Un STS est défini par un n-uplet $\mathcal{M} = (V, \Theta, \Sigma, \Delta)$ où*

– $V = \{v_1, \ldots, v_n\}$ est un ensemble fini de variables. Nous notons $Q \triangleq \mathcal{D}_V$ le domaine potentiellement infini des variables ;

– $\Theta \subseteq Q$ est la condition initiale portant sur les variables V ;

– Σ est l'alphabet des actions. Chaque action $\sigma \in \Sigma$ a une signature $sig(\sigma)$ qui est un vecteur de types $\langle t_1, \ldots, t_k \rangle$ spécifiant les types des paramètres de communications portés par l'action σ ;

– Δ un ensemble fini de transitions symboliques. Chaque transition $\delta = \langle \sigma, \vec{p}, G, A \rangle$, également notée $[\sigma(\vec{p}) : G(\vec{v}, \vec{p})?\ \vec{v}' = A(\vec{v}, \vec{p})]$, est définie par

- une action σ et un vecteur de paramètres[1] $\vec{p} = \langle p_1, \ldots, p_k \rangle$, qui sont des variables locales à la transition, typées conformément à la signature $sig(\sigma)$. Nous notons $\mathcal{D}_\sigma \triangleq \mathcal{D}_{\vec{p}}$;

- une garde $G(\vec{v}, \vec{p}) \subseteq Q \times \mathcal{D}_\sigma$ portant sur les variables $\vec{v}$ et les paramètres $\vec{p}$;

- une affectation de l'ensemble des variables, de la forme $\vec{v}' := A(\vec{v}, \vec{p})$, avec $A : Q \times \mathcal{D}_\sigma \to Q$, définissant l'évolution des variables au cours de l'exécution.

Pour une transition δ, nous notons resp. σ^δ, G^δ et A^δ son action, sa garde et son affectation.

Un état du système est définie par la valeur de chaque variable. Les transitions sont étiquetées par des actions valuées, ainsi que par des gardes et des affectations sur les variables. Par abus de notation, nous employons désormais $\vec{p}$ pour désigner la valeur $\pi \in \mathcal{D}_{\vec{p}}$ du paramètre, et $\vec{v}$ pour désigner la valeur $\nu \in \mathcal{D}_{\vec{v}}$ des variables.

Définition 3 (STS, sémantique) *La sémantique d'un STS $\mathcal{M} = (V, \Theta, \Lambda, \Delta)$ est le LTS $[\![\mathcal{M}]\!] = (Q, Q_o, \Lambda, \to)$ potentiellement infini, où*

– $Q = \mathcal{D}_V$, $Q_0 = \Theta$;

– $\Lambda = \{\langle \sigma, \vec{p} \rangle \mid \sigma \in \Sigma \wedge \vec{p} \in \mathcal{D}_\sigma\}$ est l'alphabet des actions valuées ;

– la relation de transition $\to = \bigcup_{\delta \in \Delta} \to_\delta$ est défini par la règle d'inférence

$$\frac{\delta = [\sigma(\vec{p}) : G(\vec{v}, \vec{p})\ ?\ \vec{v}' = A(\vec{v}, \vec{p})] \in \Delta, G(\vec{v}, \vec{p})\ \wedge\ \vec{v}' = A(\vec{v}, \vec{p})}{\vec{v} \xrightarrow{\langle \sigma, \vec{p} \rangle}_\delta \vec{v}'}$$

Par comparaison avec la définition des LTS donnée plus haut, l'ensemble des états Q et l'alphabet Λ peuvent être infinis et même non dénombrables. Un STS est donc

1. Les paramètres de communication servent essentiellement à représenter la communication de données entre deux systèmes modélisés par des STS.

essentiellement une syntaxe définissant de manière finie un système de transitions infini.

Les notions d'exécutions, de traces, de blocage, de déterminisme, ... d'un STS $\mathcal{M}$ se définissent à l'aide des mêmes notions sur le LTS $[\![\mathcal{M}]\!]$. D'un point de vue symbolique, $\vec{v}$ est bloquant ssi,

$$\forall \delta \in \Delta, \forall \vec{p} : \neg G^{\delta}(\vec{v}, \vec{p}) \qquad [1]$$

i.e., s'il n'existe aucune transition symbolique et aucune valeur des paramètres tel que la garde de la transition soit satisfaisable.

Le caractère déterministe d'un STS $[\![\mathcal{M}]\!]$ peut être difficile à déterminer sur le LTS $[\![M]\!]$, car des affectations syntaxiquement différentes peuvent avoir la même sémantique. Nous introduisons donc la notion suivante. Un STS $\mathcal{M}$ est *structurellement déterministe* si les gardes des transitions portant la même action sont mutuellement exclusives : $\forall \delta_1, \delta_2 \in \Delta \ : \ \sigma^{\delta_1} = \sigma^{\delta_2} \implies G^{\delta_1}(\vec{v}, \vec{p}) \wedge G^{\delta_2}(\vec{v}, \vec{p}) = \text{false}$. Le déterminisme structurel implique le déterminisme. Il est vérifiable sur la syntaxe d'un STS si une procédure de décision pour la satisfaisabilité des formules existe pour la classe de formules considérées.

2.3. *Automates hybrides*

Le modèle des STS présenté dans le paragraphe précédent est étendu ici par l'ajout d'un comportement continu de certaines variables réelles entre deux transitions discrètes. Ce comportement continu est défini à l'aide d'équations (ou d'inéquations) différentielles. On obtient ainsi un automate hybride (Alur *et al.*, 1995). Ce type d'automate est particulièrement bien adapté pour modéliser des systèmes obéissant aux lois de la physique (écoulement d'un fluide, objets en mouvements) sur lesquels il est possible d'agir par des commandes, modélisées par des transitions discrètes : par exemple l'ouverture ou la fermeture d'une vanne commandant la vidange d'une cuve. Plus généralement, il permet de modéliser, par composition d'automates, les interactions entre un contrôleur discret et un environnement physique "continu".

Définition 4 (HTS, syntaxe) *Un automate hybride est défini par un tuple* $(V, \Theta, \Sigma, \Delta, D, H)$ *dans lequel :*

– $(V, \Theta, \Sigma, \Delta)$ *définit un STS, dont on partitionne l'ensemble des variables* V *en* $V_Q \cup V_X$*. Les variables dans* V_Q *sont des variables soumises à un comportement discret, tandis que les variables dans* V_X *sont des variables réelles soumises en outre à un comportement continu. On note* $S \triangleq Q \times X = \mathcal{D}(V_Q) \times \mathcal{D}(V_X)$ *l'espace d'état induit par* V*. On a* $X = \mathbb{R}^n$*, avec* $n = |V_X|$ *le nombre de variables continues.*

– $D : Q \to (X \to X)$ *est une fonction qui associe à chaque état discret une équation différentielle* $\dot{\vec{x}} = D(\vec{q})(\vec{x})$*. La fonction* $D(\vec{q})$ *est supposée satisfaire la condition de Lipschitz pour tout* $\vec{q} \in Q$ *et* $\vec{x} \in X$

– $H : Q \to \wp(X)$ *associe à chaque état discret* q *un* invariant *(topologiquement ouvert) qui représente le domaine de validité de l'équation différentielle définie par*

$D(\vec{q})$. Lorsque $\vec{x} \notin H(\vec{q})$, les variables continues $\vec{x}$ ne peuvent plus évoluer continûment et seule une transition discrète peut être empruntée.
Nous désignerons également par H l'ensemble : $\{\langle \vec{q}, \vec{x} \rangle \mid q \in Q \wedge x \in H(q)\}$.

Remarque 1 *De même que pour les STS, nous n'avons pas de notion explicite de localité dans les HTS. Il est clair cependant que l'ensemble Q correspond aux localités dans les modèles classiques d'automates hybrides (Alur* et al.*, 1995), ensemble qui peut être ici infini (si par ex. V_Q contient des variables de types entier ou réel). Ce modèle apparaît en fait naturellement si l'on considère des HTS obtenus par composition d'un programme (discret) et d'un environnement physique (continu ou hybride).*

Par ailleurs, le comportement continu des HTS ainsi définis est déterministe. On peut généraliser ce modèle en autorisant des inclusions différentielles *(Alur* et al.*, 1995), ou alors des* entrées continues *comme dans (Tomlin* et al.*, 2000), qui donne une définition très générale des systèmes hybrides, mais où les événements sont remplacés par des entrées synchrones, discrètes ou continues.*

Pour définir la sémantique d'un HTS, il est nécessaire de généraliser la notion d'exécution à celle de *trajectoire*, composée de *segments de trajectoire* séparés par des transitions discrètes. Par le théorème de Cauchy, étant donné un état initial $(\vec{q}, \vec{x})$, il existe une unique solution $\vec{\xi}(t)$ à l'équation $\dot{\vec{x}} = D(\vec{q})(\vec{x})$ avec pour condition initiale $\vec{\xi}(0) = \vec{x}$, que nous noterons $\vec{\xi}_{(\vec{q},\vec{x})}(t)$.

Définition 5 (Segment de trajectoire) *Étant donné un état $(\vec{q}, \vec{x}) \in S$ et un intervalle $[0, T] \subseteq \mathbb{R}$, un segment de trajectoire est une fonction $\vec{\nu}_{(\vec{q})(\vec{x})} : [0, T] \to S$, avec $\vec{\nu}_{(\vec{q})(\vec{x})}(t) = \langle \vec{q}, \vec{\xi}_{(\vec{q})(\vec{x})}(t) \rangle$ et $\forall t \in [0, T] : \vec{\xi}_{(\vec{q})(\vec{x})}(t) \in H(\vec{q})$.*

Définition 6 (Trajectoire) *Étant donné un HTS $(V, \Theta, \Sigma, \Delta, D, H)$, une trajectoire ν est une séquence $\vec{\nu}_0 \xrightarrow{\langle \sigma_0, \vec{\pi}_0 \rangle} \vec{\nu}_1 \xrightarrow{\langle \sigma_1, \vec{\pi}_1 \rangle} \vec{\nu}_2 \ldots$, telle que $\vec{\nu}_0(0) \in \Theta$ est initial et, $\forall i \geq 0$:*

1) $\vec{\nu}_i : [0, T_i] \to S$ est un segment de trajectoire ;

2) $\vec{\nu}_i(T_i) \xrightarrow{\langle \sigma_i, \vec{\pi}_i \rangle} \vec{\nu}_{i+1}(0)$ (ce qui est noté plus simplement $\vec{\nu}_i \xrightarrow{\langle \sigma_i, \vec{\pi}_i \rangle} \vec{\nu}_{i+1}$), où $\to$ est la relation de transition définie par le STS sous-jacent $(V, \Theta, \Sigma, \Delta)$ (cf. définition 3).

Dans la suite, nous parlerons d'*exécution* d'un HTS au lieu de trajectoire, afin de disposer d'un terme identique pour les trois modèles d'automates que nous considérons.

Un HTS est (structurellement) *déterministe* si le STS sous-jacent est (structurellement) déterministe (les évolutions continues sont déterministes). Notons qu'il reste une source d'indéterminisme, lorsqu'il existe un choix entre la progression d'un évolution continue et une transition discrète. En incluant dans les traces les domaines $[0, T_i]$ des segments de trajectoires, nous avons la propriété des systèmes déterministes selon

laquelle un état initial et une trace (incluant les T_i) définissent de manière unique une exécution.

Un état $s = (\vec{q}, \vec{x})$ d'un STS est *bloquant* s'il est bloquant dans le STS sous-jacent et si $H(\vec{q})(\vec{x}) = \text{false}$ (aucune évolution continue n'est possible). Un HTS est *non bloquant* s'il n'existe aucune exécution menant dans un état bloquant.

3. Propriétés et observateurs

Étant donné un automate, une *propriété d'invariance* est définie par un ensemble d'états de l'automate et spécifie qu'aucune exécution de l'automate ne sort de cet ensemble. Par dualité, nous pouvons définir une propriété d'invariance par un ensemble d'états qu'aucune exécution de l'automate ne doit atteindre. Dans la suite, nous utiliserons cette seconde formulation de *propriété d'interdiction d'états*. Plus généralement, une propriété porte sur des séquences, qui peuvent être des traces ou des exécutions. Dans ce cadre, une propriété est définie par un ensemble de séquences/traces. Une *propriété de sûreté* P est un ensemble de séquences clos par préfixe : $\rho \notin P \Rightarrow \forall \rho' : \rho \cdot \rho' \notin P$. En d'autres termes, si une séquence finie viole la propriété P, aucun prolongement de cette séquence ne pourra la satisfaire. Un automate satisfait une propriété P si l'ensemble de ses traces ou de ses exécutions (selon le cas) est inclus dans P. L'absence de panne, de défaillance, ou bien la conformité sont des exemples classiques de propriétés de sûreté.

Il est toujours possible de réduire une propriété de sûreté sur les traces à une propriété d'invariance par le biais d'*observateurs*. Étant donné un automate $\mathcal{M}$, un observateur est un automate $\mathcal{O}$ muni d'un état distingué $\mathsf{Violate}$ et *non-intrusif*, c'est-à-dire vérifiant la propriété $\mathcal{L}(\mathcal{M} \times \mathcal{O}) = \mathcal{L}(\mathcal{M})$, où $\mathcal{L}(\mathcal{M})$ désigne l'ensemble des *comportements* de l'automate $\mathcal{M}$ (définition exacte dépendant du modèle) et $\times$ dénote le produit d'automates, décrivant l'intersection des comportements (voir (Legall *et al.*, 2005) pour la définition du produit sur les différents modèles LTS, STS et HTS). *Le langage reconnu par un observateur* $\mathcal{O}$, noté $\mathcal{L}_{\mathsf{Violate}}(\mathcal{O})$, est l'ensemble des traces menant à l'état $\mathsf{Violate}$.

Un observateur $\mathcal{O}$ code la négation d'une propriété de sûreté $\varphi_{\mathcal{O}}$. En effet le langage reconnu par $\mathcal{O}$ peut être interprété comme l'ensemble des séquences qui violent $\varphi_{\mathcal{O}}$. D'un point de vue opérationnel, $\mathcal{O}$ est joué en parallèle avec le système et observe les événements produits par le système, produisant le verdict $\mathsf{Violate}$ lorsque la propriété est violée. En exploitant la propriété suivante : $\mathcal{M}$ satisfait $\varphi_{\mathcal{O}}$ ssi $\mathcal{M} \times \mathcal{O}$ satisfait la propriété d'interdiction d'état $S^{\mathcal{M}} \times \{\mathsf{Violate}^{\mathcal{O}}\}$, l'usage d'observateurs permet de réduire une propriété de sûreté sur les traces de $\mathcal{M}$ à une propriété d'invariance sur le produit synchrone $\mathcal{M} \times \mathcal{O}$.

Dans la suite de l'article nous ne considérerons plus que des propriétés d'invariance définies par un ensemble d'états interdits. Contrôler le système $\mathcal{M}$ pour assurer une propriété de sûreté modélisée par un observateur $\mathcal{O}$ revient en effet à contrôler $\mathcal{M} \times \mathcal{O}$ et à assurer sur celui-ci la propriété d'interdiction d'états $S^{\mathcal{M}} \times \{\mathsf{Violate}^{\mathcal{O}}\}$.

4. Contrôle des systèmes de transitions finis

Nous rappelons ici brièvement une version simplifiée de la théorie du contrôle de systèmes à événements discrets initiée par (Ramadge *et al.*, 1989), restreinte au problème de l'interdiction d'états (cf. (Ramadge *et al.*, 1987) pour plus de détails). Les notations que nous utiliserons seront assez différentes de celles utilisées par cette communauté, afin de faciliter le lien avec le contrôle des STS et HTS.

Définition du problème de contrôle. D'après nos critères de classification, nous nous situons dans le cadre du contrôle interne et que donc tout est observable (état et événements). En ce qui concerne les moyens de contrôle, nous supposons un alphabet des événements $\Lambda = \Lambda_{uc} \cup \Lambda_c$ partitionné en événements respectivement incontrôlables et contrôlables. Le moyen d'action dont nous disposons est l'interdiction des événements contrôlables.

Nous considérons donc un LTS $\mathcal{M} = (Q, Q_0, \Lambda_{uc} \cup \Lambda_c, \rightarrow)$, et un ensemble d'états $E \subseteq Q$ à interdire, et nous voulons générer un LTS $\mathcal{M}' = (Q, Q'_0, \Lambda_{uc} \cup \Lambda_c, \rightarrow')$ avec $\rightarrow' \subseteq \rightarrow$ tel qu'aucun état dans E n'est accessible. Par ailleurs, $\mathcal{M}'$ doit être le plus permissif possible, c'est-à-dire que $\rightarrow'$ doit être la plus grande relation (si elle existe) assurant la propriété.

Définition constructive du système contrôlé maximal. L'idée est de rendre inaccessible les états qui peuvent mener de manière incontrôlable à un état interdit. Nous commençons donc par calculer le plus petit ensemble $E' \supseteq E$ qui vérifie cette propriété. À cet effet, nous introduisons l'opérateur suivant : pour $Y \subseteq Q$

$$\mathrm{Pre}(Y) = \{q \in Q \mid \exists q' \in Y, \exists a \in \Lambda_{uc} : q \xrightarrow{a} q'\} \qquad [2]$$

$\mathrm{Pre}(Y)$ est l'ensemble des états qui peuvent mener à Y par l'occurrence d'un seul événement incontrôlable. L'ensemble E' des états pouvant mener à E par une séquence quelconque d'événements incontrôlables est défini par le plus petit point-fixe de Pre contenant E (ensemble $\mathrm{Coreach}(E)$ des états coaccessibles depuis E) :

$$E' = \mathrm{Coreach}(E) \triangleq \mathrm{lfp}(\lambda Y \,.\, E \cup \mathrm{Pre}(Y)) \qquad [3]$$

Nous définissons maintenant la fonction auxiliaire $\mathcal{C}$, qui spécifie pour un état q quelles sont ses transitions sortantes à interdire :

$$\mathcal{C}(q) = \begin{cases} \emptyset & \text{si } q \in \mathrm{Coreach}(E) \\ \{(a, q') \in \Lambda \times Q \mid q \xrightarrow{a} q' \wedge q' \in \mathrm{Coreach}(E)\} & \text{sinon} \end{cases}$$

Par définition de $\mathrm{Coreach}$, $\mathcal{C}(q) \subseteq \Lambda_c \times Q$ pour tout q.

Le système contrôlé maximal $\mathcal{M}' = (Q, Q'_0, \Lambda_{uc} \cup \Lambda_c, \rightarrow')$ est alors défini à partir de $\mathcal{M}$ par $Q'_0 = Q_0 \setminus \mathrm{Coreach}(E)$ et

$$\frac{q \xrightarrow{a} q' \quad (a, q') \notin \mathcal{C}(q)}{q \xrightarrow{a}{}' q'} \qquad [4]$$

L'existence d'un système contrôlé maximal découle directement de l'existence d'un *plus petit* point-fixe $\mathrm{Coreach}(E)$, qui découle du théorème de Tarski et de la continuité (impliquant la croissance) des fonctions impliquées dans le treillis complet des ensembles d'états $(\wp(Q), \subseteq)$.

Comme $Q'_0 \subseteq \overline{\mathrm{Coreach}(E)}$, le système contrôlé $\mathcal{M}'$ vérifie la propriété d'interdiction d'états $\mathrm{Coreach}(E) \supseteq E$. En effet ses états initiaux vérifient cette propriété, et la fonction $\mathcal{C}$ interdit toutes les transitions de $\mathcal{M}$ faisant sortir de l'ensemble d'états $\overline{\mathrm{Coreach}(E)}$.

Calcul effectif du système contrôlé. Le théorème de Kleene permet d'obtenir le point-fixe $\mathrm{Coreach}(E)$ ([3]) en calculant la limite de la suite définie par $E_0 = E$ et $E_{n+1} = E_n \cup \mathrm{Pre}(E_n)$. Comme $(\wp(Q), \subseteq)$ est un treillis de hauteur finie, cette suite converge en un nombre fini d'étapes vers l'ensemble $\mathrm{Coreach}(E)$, qui lui-même définit le contrôleur $\mathcal{C}$. Ce type de calcul peut se faire de manière énumérée (Emerson *et al.*, 1986) ou symbolique (Burch *et al.*, 1992) et est à la base du *model-checking*.

Garantir le non blocage. En appliquant l'algorithme précédent, le système contrôlé obtenu vérifie la propriété d'interdiction, mais il peut être bloquant, alors même que le système de départ ne l'était pas[2]. De manière à assurer cette propriété, la contrainte de non blocage est directement intégrée dans l'opérateur calculant les nouveaux états à interdire :

$$\mathrm{Pre}^{nb}(Y) = \mathrm{Pre}(Y) \cup \left\{ q \;\middle|\; \begin{array}{c} \forall a \in \Lambda_c : q \xrightarrow{a} q' \implies q' \in Y \\ \wedge\, \forall a \in \Lambda_{uc} : \neg(q \xrightarrow{a}) \end{array} \right\} \quad [5]$$

$$= \mathrm{Pre}(Y) \cup \{ q \mid \forall a \in \Lambda : q \xrightarrow{a} q' \implies q' \in Y \} \quad [6]$$

L'équation [5] rajoute à $\mathrm{Pre}(Y)$ les états dont toutes les transitions contrôlables mènent à Y et qui n'ont pas de transitions incontrôlables[3]. En effet, dans ce cas, le contrôle consistera à couper toutes les transitions sortantes et le système sera alors en blocage. L'équation [6] simplifie l'expression de $\mathrm{Pre}^{nb}(Y)$ en remplaçant l'union disjointe par une union non disjointe. En effectuant le même calcul de point-fixe que précédemment, l'ensemble $\mathrm{Coreach}^{nb}(E)$ obtenu avec ce nouvel opérateur vérifie :

$$\forall q \notin \mathrm{Coreach}^{nb}(E), \exists a \in \Lambda, \exists q' \notin \mathrm{Coreach}^{nb}(E),\ q \xrightarrow{a} q'$$

Par conséquent, le système obtenu est non bloquant.

Le contrôle des LTS, avec pour objectif l'interdiction de certains états, tel qu'exposé dans cette section, définit les concepts et les méthodes fondamentales du contrôle. En particulier, le recours à des calculs de point-fixe est universel, comme de manière plus générale en vérification de modèle et en interprétation abstraite. Les deux sections suivants développent ces notions dans le cadre de systèmes symboliques.

2. Ceci constitue une hypothèse de départ. Si le système initial est bloquant, il suffit de rajouter à E l'ensemble des états en blocage avant contrôle et de les interdire.
3. Les transitions incontrôlables sont prises en compte dans $\mathrm{Pre}(E)$.

5. Contrôle des systèmes de transitions symboliques

5.1. *Moyen de contrôle sur un STS*

Dans un STS, nous distinguons les actions (noms d'événements) comme σ des événements valués comme $\sigma(3,5)$. Par ailleurs nous disposons de prédicats et d'opérations algébriques sur les variables. Faut-il alors raffiner la définition de la contrôlabilité utilisée pour les LTS ? Nous examinons quelques pistes et proposons un modèle.

Partition globale entre actions contrôlables et non contrôlables : il s'agit de la transposition du modèle utilisé pour les LTS. Nous partitionnons $\Sigma = \Sigma_{uc} \cup \Sigma_c$, ce qui engendre une partition des actions valuées Λ. Le contrôle consistera alors à interdire dans une transition symbolique une action valuée $\sigma(\vec{p})$ quelle que soit la valeur des paramètres, *i.e.* à interdire purement et simplement la transition symbolique.

Une généralisation naturelle, et cohérente avec le cadre symbolique, est de partitionner à la place les actions valuées, c'est-à-dire que le caractère contrôlable d'un action $\sigma(\vec{p})$ dépend de la valeur de ses paramètres $\vec{p}$. Pour ce faire, nous associons à chaque action $\sigma(\vec{p})$ une condition $c_\sigma(\vec{p})$ qui indique quand elle est contrôlable et peut être interdite. Par exemple, si l'action σ est totalement incontrôlable, on aura $c_\sigma(\vec{p}) = \mathrm{false}$. Le contrôle consistera alors à interdire dans une transition symbolique δ une action valuée $\sigma(\vec{p})$ pour certaine valeurs des paramètres. Ceci peut se faire en renforçant la garde $G(\vec{v},\vec{p})$ de la transition δ par une nouvelle garde $G(\vec{v},\vec{p}) \wedge c(\vec{p})$, avec $c(\vec{p}) \implies c_\sigma(\vec{p})$.

Exemple 1 *Les interruptions d'un système matériel ont généralement une priorité qui leur est attribuée, et il existe des mécanismes pour inhiber les interruptions de basse priorité. Si on considère un signal* $\mathsf{Interrupt}(n)$*, on peut modéliser le fait que le signal peut être inhibé si sa priorité est inférieure à* N*, en posant* $c_{\mathsf{Interrupt}}(n) = (n \leq N)$.

Partition locale entre actions contrôlables et non contrôlables : l'idée ici est de faire dépendre le caractère contrôlable d'une action non seulement de ses paramètres, mais aussi de l'état du système. C'est-à-dire qu'on remplace la condition de contrôlabilité $c_\sigma(\vec{p})$ par $c_\sigma(\vec{v},\vec{p})$.

Exemple 2 *Supposons un système de pilotage d'un avion, disposant d'un module de pilotage automatique, et une action* TurnLeft*. Lorsque le pilote automatique est branché (état particulier du système), il est logique de considérer que* TurnLeft *est incontrôlable, tandis que s'il est débranché, l'opérateur contrôle cet événement.*

Remarque 2 *Dans le cas des LTS, la notion de partition locale a également un sens. Elle consiste à faire dépendre le caractère contrôlable d'une transition non seulement de l'événement associé, mais aussi de l'état d'origine. De plus, en remplaçant l'alphabet* Λ *par* $Q \times \Lambda$*, on peut se ramener au cas d'une partition globale. Cependant si ce codage permet d'établir des équivalences théoriques, en pratique il n'est ni satisfaisant (changement d'alphabet, redéfinition du produit synchrone ...), ni même nécessaire (on peut aisément modifier les équations de la section 4 pour prendre en compte l'aspect local de la contrôlabilité des événements).*

Dans le cadre symbolique, la notion de partition locale est séduisante par sa généralité et le fait qu'elle correspond à certains besoins de modélisation. En particulier, lorsque le contrôle est utilisé pour raffiner une spécification, on peut indiquer dans la spécification initiale ce qui peut-être raffiné (interdit) de ce qui doit rester inchangé. Nous adoptons donc le modèle suivant :

Définition 7 *Un* STS intégrant la notion de contrôlabilité *est STS défini comme dans la définition 2, sauf pour les transitions* $\delta = \langle \sigma, p, G, G_{uc}, A \rangle$ *qui ont pour composante supplémentaire une* garde d'incontrôlabilité $G_{uc}(\vec{v}, \vec{p}) \subseteq G(\vec{v}, \vec{p})$*, qui spécifie*

$$\forall \vec{v} \in Q, \forall \vec{p} \in \mathcal{D}_\sigma : G_{uc}(\vec{v}, \vec{p}) \implies \textit{l'événement } \sigma(\vec{p}) \textit{ est incontrôlable}$$

En comparaison avec ce que nous avions esquissé ci-dessus avec la condition de contrôlabilité $c_\sigma(\vec{v}, \vec{p})$, le caractère contrôlable de σ dépend non seulement de l'état courant et des paramètres $\vec{p}$, mais aussi de la transition symbolique considérée, c'est-à-dire aussi de son affectation A. Toutefois, dans le cas d'un système déterministe, les deux formulations sont équivalentes. D'un point de vue « langage de programmation », celle-ci nous apparaît plus pratique.

Dans ce cadre, les *moyens d'actions* du contrôle sont de renforcer les gardes G des transitions symboliques par des gardes G' satisfaisant $G_{uc} \implies G' \implies G$.

5.2. *Définition constructive du système contrôlé maximal*

Le principe de la synthèse du système contrôlé, exposé en section 4 reste identique : nous définissons le plus petit ensemble d'états à interdire, étant donné les contraintes d'incontrôlabilité, puis nous modifions le système initial en conséquence. La nouveauté majeure est qu'il faut manipuler les états à partir de la description syntaxique du STS à contrôler.

Soit un STS structurellement déterministe $\mathcal{M} = (V, \Theta, \Sigma, \Delta)$ et une propriété d'interdiction d'état décrit par un prédicat $E(\vec{v})$. En adoptant alternativement une formulation logique et une formulation ensembliste, on a les définitions suivantes :

$$\mathrm{Pre}(\delta)(Y)(\vec{v}) = \exists \vec{p} \exists \vec{v}' : G_{uc}(\vec{v}, \vec{p}) \wedge \vec{v}' = A(\vec{v}, \vec{p}) \wedge Y(\vec{v}') \quad [7]$$

$$\mathrm{Pre}(\delta)(Y) = \mathit{Proj}_Q\left(G_{uc} \cap A^{-1}(Y)\right) \quad [8]$$

$$\mathrm{Pre}(Y)(\vec{v}) = \bigvee_{\delta \in \Delta} \mathrm{Pre}(\delta)(Y)(\vec{v}) \quad [9]$$

$$\mathrm{Pre}(Y) = \bigcup_{\delta \in \Delta} \mathrm{Pre}(\delta)(Y) \quad [10]$$

Nous définissons ensuite $\mathrm{Coreach}(E) = \mathrm{lfp}(\lambda Y.E \cup \mathrm{Pre}(Y))$. Nous introduisons la fonction auxiliaire $\mathcal{C}$ qui spécifie les transitions à interdire :

$$\mathcal{C}(\delta)(\vec{v}, \vec{p}) = \neg \mathrm{Coreach}(E)(\vec{v}) \wedge \mathrm{Coreach}(E)\left(A^\delta(\vec{v}, \vec{p})\right) \quad [11]$$

$$\mathcal{C}(\delta) = (A^\delta)^{-1}(\mathrm{Coreach}(E)) \setminus (\mathrm{Coreach}(E) \times \mathcal{D}_{\sigma^\delta}) \quad [12]$$

Par définition de $\mathrm{Coreach}(E)$, on a, pour toute transition δ, $\mathcal{C}(\delta) \Rightarrow \neg G^{\delta}_{uc}$: on n'interdit aucune transition incontrôlable.

Garantir le non blocage. Comme en section 4, contrôler le système peut introduire des blocages qui n'existaient pas. En supposant que le système initial $\mathcal{M}$ ne contient pas de blocage, nous assurons la propriété de non blocage sur le système contrôlé $\mathcal{M}'$ en procédant de manière similaire au cas des LTS. En se référant à l'équation [6] (plutôt qu'à l'équation [5]), il s'agit de rajouter à $\mathrm{Pre}(Y)$ les états qui mènent nécessairement à Y, quelle que soit la transition considérée. Cet ensemble Z se définit ainsi :

$$\begin{aligned} \vec{v} \in Z \quad &\Leftrightarrow \quad \forall \delta \in \Delta,\, \forall \vec{p} \,:\, G^{\delta}(\vec{v},\vec{p}) \Rightarrow A^{\delta}(\vec{v},\vec{p}) \in Y \\ &\Leftrightarrow \quad \bigwedge_{\delta \in \Delta} \forall \vec{p} \,:\, G^{\delta}(\vec{v},\vec{p}) \Rightarrow \left((A^{\delta})^{-1}(Y)\right)(\vec{v},\vec{p}) \end{aligned}$$

Ce qui se traduit en formulation ensembliste par :

$$Z \;=\; \bigcap_{\delta \in \Delta} \mathit{Proj}^{\forall}_{Q}\left(\overline{G^{\delta}} \cup (A^{\delta})^{-1}(Y)\right) \qquad [13]$$

Nous en déduisons l'expression de l'opérateur de précondition :

$$\mathrm{Pre}^{nb}(Y) \quad = \quad \mathrm{Pre}(Y) \cup \bigcap_{\delta \in \Delta} \mathit{Proj}^{\forall}_{Q}\left(\overline{G^{\delta}} \cup (A^{\delta})^{-1}(Y)\right) \qquad [14]$$

Système contrôlé maximal. Le système contrôlé $\mathcal{M}' = (V, \Theta', \Sigma, \Delta')$ est alors défini par :

– $\Theta' = \Theta \setminus \mathrm{Coreach}(E)$;

– Δ' est défini à partir de Δ par $\dfrac{\langle \sigma, \vec{p}, G_{uc}, G, A\rangle \in \Delta \quad G' = G \wedge \neg\mathcal{C}(\delta)}{\langle \sigma, \vec{p}, G_{uc}, G', A\rangle \in \Delta'}$

Il est facile de voir qu'aucune exécution de $\mathcal{M}'$ ne peut atteindre $\mathrm{Coreach}(E)$, et donc E. En effet, les états initiaux de $\mathcal{M}'$ ne sont pas dans $\mathrm{Coreach}(E)$, et aucune transition dans Δ' ne peut faire passer l'état courant de $\neg\mathrm{Coreach}(E)$ à $\mathrm{Coreach}(E)$, cf. équation [12].

5.3. *Calcul effectif du système contrôlé*

Comme pour les LTS, le calcul du STS contrôlé repose sur le calcul par résolution de point-fixe de $\mathrm{Coreach}(E)$. Cependant, lorsque des variables ont un domaine de valeur infini, l'espace d'états induits est lui aussi infini, et le point-fixe n'est plus calculable dans le cas général[4]. Il existe trois approches pour contourner ce problème :

4. En rendant tous les événements incontrôlables, on se ramène à un calcul de coaccessibilité, qui est indécidable pour une machine à deux compteurs (indécidabilité de la terminaison).

1) identifier des sous-classes de systèmes infinis pour lesquels ce type de calcul reste décidable (voir (Henzinger *et al.*, 2002)) ;

2) utiliser des techniques d'accélération qui permettent d'obtenir le point-fixe exact en cas de convergence (et lorsque le point-fixe est représentable) (Abdulla *et al.*, 1998; Bouajjani, 2001; Wolper *et al.*, 1998; Finkel *et al.*, 2002). Il est à noter que même quand le point-fixe est représentable, la convergence n'est pas nécessairement assurée ;

3) ou enfin, se contenter d'une (sur)approximation du point-fixe.

Remarquons que lorsqu'on applique l'approche 1 *dans le cadre particulier de la synthèse* (et sans garantir le non blocage), une condition suffisante pour la calculabilité de $\mathrm{Coreach}(E)$ est que les séquences d'actions incontrôlables dans le LTS $[\![M]\!]$ soient de taille bornée — ce qui assure la convergence du calcul itératif en un nombre borné d'étapes, et que les formules utilisées pour décrire les ensembles et les expressions appartiennent à une logique décidable – ce qui permet de détecter la convergence.

Nous suivons dans cet article cette dernière approche, en notant que si remplace dans les équations précédentes $\mathrm{Coreach}(E)$ par un sur-ensemble, la propriété que les états interdits de E sont inatteignables reste vérifiée. En revanche, la caractéristique que l'on perd est la permissivité maximale du contrôleur : il se peut que des exécutions ne menant jamais à E de manière incontrôlable soient interdites dans le système contrôlé. Si nous voulons assurer en outre le non blocage, la sur-approximation nous conduit aussi à interdire des états qui ne seraient pas en blocage.

Il reste à expliquer comment calculer un sur-ensemble de $\mathrm{Coreach}(E)$. L'interprétation abstraite, qui est une théorie du calcul approché de point-fixe appliquée à l'analyse de programme, offre un cadre théorique et propose des méthodes adaptées à cette question.

5.3.1. *Principe de l'interprétation abstraite*

Étant donné une équation de point-fixe $x = F(x), x \in \wp(S)$ sur le treillis complet des parties de l'espace des états S, avec F croissante, l'interprétation abstraite propose de calculer une approximation du plus petit point-fixe, en résolvant les deux problèmes suivants :

– **Représentation et manipulation des ensembles d'états.** Nous avons besoin de représenter, de manipuler et de comparer des ensembles d'états, ce qui pose des problèmes lorsque S est infini. Une première approximation consiste à remplacer les valeurs concrètes $x \in \wp(S)$ par des valeurs abstraites $y \in A$ plus simples. Les treillis concrets $(\wp(S), \subseteq)$ et abstraits $(A, \sqsubseteq)$ sont reliés par une connexion de Galois $\wp(S) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} A$ qui assure la correction de la méthode (Cousot *et al.*, 1977). L'équation de point-fixe est transposée dans A: $y = F^\sharp(y), y \in A$, avec $F^\sharp \sqsupseteq \alpha \circ F \circ \gamma$. Sous les hypothèses mentionnées, le résultat fondamentale est le suivant : $\mathrm{lfp}(F) \subseteq \gamma(\mathrm{lfp}(F^\sharp))$; *i.e.*, la concrétisation du point-fixe de l'équation abstraite est une surapproximation du point-fixe de l'équation concrète.

– **Calcul approché du point-fixe abstrait.** La résolution itérative de l'équation

$y = F^\sharp(y), y \in A$ peut ne pas converger, lorsque A contient des chaînes infinies strictement croissantes. Exiger que A ne contienne pas de telles chaînes est trop restrictif. De meilleurs résultats peuvent être obtenus en utilisant un opérateur d'élargissement (Cousot *et al.*, 1992), qui contrairement aux techniques d'accélération permettent de converger (i) en un nombre *fini* d'étapes, (ii) vers un *post*-point-fixe y de $F^\sharp$ (*i.e.*, tel que $y \sqsupseteq F^\sharp(y) \sqsupseteq \mathrm{lfp}(F^\sharp)$). Typiquement, un "bon" élargissement doit capturer et extrapoler les régularités les plus fréquentes dans le treillis considéré.

Un exemple très classique d'interprétation abstraite est l'analyse de relations linéaires (Cousot *et al.*, 1978; Halbwachs *et al.*, 1997), dans laquelle l'espace d'états induit par n variables est $S = \mathbb{R}^n$ et le treillis concret $\wp(S)$ est abstrait par le treillis $\mathsf{Pol}[n]$ des polyèdres convexes de dimension n. Dans cette abstraction, un ensemble de valeurs des variables est représenté de manière approchée par le plus petit polyèdre convexe le contenant. L'élargissement standard sur ce treillis consiste *grosso-modo*, dans $P = P_1 \nabla P_2$, à supprimer les contraintes de P_1 qui ne sont pas satisfaites par P_2.

5.3.2. *Application au calcul du contrôleur*

On cherche généralement à dériver l'équation de point-fixe de manière compositionnelle, à partir de la syntaxe du système à analyser, ici notre STS $\mathcal{M} = (V, \Theta, \Sigma, \Delta)$. Si nous disposons d'un treillis abstrait $A(\sqsubseteq, \sqcup, \sqcap, \top, \bot)$ pour notre espace d'état A, avec $\wp(Q) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} A$. Formellement, nous supposons que la connexion de Galois peut s'étendre à $\wp(Q \times \mathcal{D}_\sigma) \leftrightarrows A_\sigma$.

Nous transposons les équation du section5.2 comme suit :

$$\mathrm{Pre}^\sharp(\delta)(Y \in A) = \alpha\left(Proj_Q(G^\delta_{uc} \cap (A^\delta)^{-1}(\gamma(Y)))\right) \quad [15]$$

$$\mathrm{Pre}^\sharp(Y \in A) = \bigsqcup_{\delta \in \Delta} \mathrm{Pre}^\sharp(\delta)(Y) \quad [16]$$

et on résout l'équation $Y = \alpha(E) \sqcup \mathrm{Pre}^\sharp(Y)$, en obtenant un post-point-fixe noté $\mathrm{Coreach}^\sharp(E)$.[5] Le contrôleur $\mathcal{C}$ et le système contrôlé sont alors définis comme précédemment à partir de $\gamma(\mathrm{Coreach}^\sharp(E))$.

Treillis utilisés. Dans la pratique, il faut choisir un treillis adapté au type des paramètres et variables, et qui induise une précision suffisante du calcul pour que le système contrôlé résultant reste raisonnablement permissif.

Supposons que les variables du STS $\mathcal{M}$ et les paramètres des actions sont des booléennes ou des réels. L'espace d'états est alors de la forme $Q = \mathbb{B}^n \times \mathbb{R}^p$. Le treillis $\wp(Q) \simeq \mathbb{B}^n \to \wp(\mathbb{R}^p)$ peut être abstrait par le treillis $\mathbb{B}^n \to \mathsf{Pol}[p]$ des fonctions qui à chaque valuation des variables booléennes associe un polyèdre convexe sur les variables réelles. Il s'agit en fait de la méthode utilisée dans (Halbwachs *et al.*, 1997),(Henzinger *et al.*, 1997), à la différence que les variables booléennes

5. Pour l'équation [15], il est possible d'abstraire de manière plus compositionnelle, mais au détriment de la précision.

sont codées dans un automate de contrôle dans ces travaux. Par souci de simplicité, (Halbwachs *et al.*, 1997),(Henzinger *et al.*, 1997) supposent que les conditions numériques sont des conjonctions de contraintes linéaires et que les affectations sont des fonctions affines. Si l'on veut utiliser des formules plus générales (tout en restant dans le cadre linéaire), et si l'on veut combiner plus finement les types booléens et numériques, on peut utiliser les techniques plus sophistiquées de partitionnement dynamique (Jeannet, 2003; Jeannet, 2002).

D'autres treillis que celui des polyèdres convexes existent pour les variables numériques, en général moins précis ou incomparables : les intervalles (Cousot *et al.*, 1976), peu précis, les octogones (Miné,, 2001) qui offrent un bon compromis précision/efficacité, ou les congruences (Granger, 1989), incomparables avec les polyèdres.

5.4. *Applications*

5.4.1. *Exemple de calcul de point-fixe*

Afin d'illustrer la section précédente, et les techniques d'approximations, voici un exemple simple de STS (figure 1). Ce STS, qui est structurellement déterministe, a deux variables x et y de type entier et un compteur de programme implicite dénotant les localités INIT, RUN et ERR. Après une phase d'initialisation, x et y peuvent décroître et on retourne dans l'état initial si y est pair. Les transitions ont des gardes soit totalement contrôlables ($G_{uc} = \emptyset$), soit totalement incontrôlables ($G_{uc} = G$). Les gardes contrôlables seront représentées avec un "?" (e.g. INIT → RUN) et les gardes incontrôlables avec un "!" (e.g. RUN → ERR).

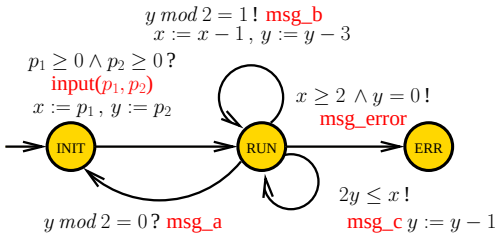

Figure 1. *Exemple d'automate à contrôler*

Nous considérons la propriété d'interdiction d'état $\mathbb{Z}^2 \times \{\mathrm{ERR}\}$. L'ensemble exact des valeurs de x et y qui définissent, dans la localité RUN, l'ensemble des états menant de manière incontrôlable à un cas d'erreur est :

$$\{(x,y) \mid \left\{ \begin{array}{ll} 2y \leq x \wedge x \geq 2 & \text{si } y \text{ est pair} \\ 2y \leq x + 5 \wedge x \geq 2 & \text{si } y \text{ est impair} \end{array} \right\}$$

Or l'enveloppe convexe de cet ensemble contient des états qui sont considérés comme bons : par exemple l'état $x = 3 \wedge y = 2$. Notre méthode d'approximation, basée sur les polyèdres convexes, ne pourra donc pas donner le résultat exact. Nous allons détailler les étapes du calcul pour bien montrer les approximations faites. Nous

notons δ_1 et δ_2 les deux transitions RUN $\rightarrow$ RUN. Au départ, nous avons un ensemble d'états interdits P_0 défini par la conjonction de contraintes $\{y = 0, x \geq 2\}$ (figure 2).

En utilisant l'équation [15], nous pouvons calculer l'ensemble $P_1 = \mathrm{Pre}^\sharp(P_0) = \{0 \leq y \leq 3, x \geq 2, y \leq 2x - 3\}$ (figure 2). P_1 est obtenu en prenant l'enveloppe convexe des trois demi-droites représentant P_0, $\mathrm{Pre}^\sharp(\delta_1)(P_0)$ et $\mathrm{Pre}^\sharp(\delta_2)(P_0)$. De même nous pouvons calculer $P_2 = \mathrm{Pre}^\sharp(P_1) = \{0 \leq y \leq 5, x \geq 2, y \leq 2x-3\}$. $P3$, qui est obtenu par élargissement: $P3 = P_1 \nabla P_2 = \{y \geq 0, x \geq 2, y \leq 2x - 3\}$, est un post-point-fixe (figure 2). Nous avons donc réussi à calculer un post-point-fixe en deux étapes, au prix d'approximations faites lors des calculs de l'enveloppe convexe et de l'élargissement.

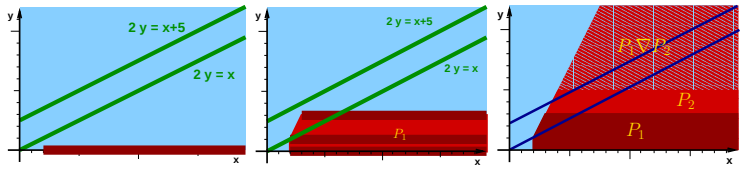

Figure 2. *Ensemble d'états « mauvais » et étapes de calcul du point-fixe*

5.4.2. *Étude de cas : le BRP*

Le *Bounded Retransmission Protocol* (BRP) (Helmink *et al.*, 1994) est un protocole de transmission de fichiers entre un émetteur S et un receveur$R$ à travers deux canaux K et L non fiables (les messages peuvent ne pas être transmis). R, S, K et L sont modélisés par des STS. Le fichier à transmettre est découpé en morceaux (*Chunks*). S envoie à R des *frames* composées d'un chunk et de 3 bits d'information, à travers le canal K. R envoie à S un message d'acquittement à travers le canal L. Les 3 bits d'information indiquent si la frame est la première, la dernière et le dernier est un bit alterné qui change d'une frame à l'autre. Ainsi en comparant le bit d'une frame par rapport à un bit référence, le receveur pourra savoir s'il a déjà reçu cette frame. L'émetteur S essaie d'envoyer une frame. S'il reçoit un message d'acquittement, il passe à la frame suivante (en changeant le bit alterné), sinon il réessaie, tant qu'on n'a pas eu un certain nombre $rmax$ d'échecs consécutifs. Le dépassement de $rmax$ provoque la fin de la transmission : on dit alors que le protocole a échoué. La spécification du système complet en STS consiste en 4 automates, manipulant 10 variables booléennes ou énumérées, et 3 variables entières.

Nous avons utilisé les techniques de synthèse de contrôleurs pour faire du raffinement de spécification. Le problème auquel nous nous sommes intéressés est le suivant : *quelles doivent être les contraintes sur l'environnement pour que le protocole n'échoue jamais ?*

Il faut tout d'abord déterminer quelles sont les transitions contrôlables et incontrôlables. Étant donné le problème posé, il apparaît normal de supposer que l'environnement, modélisé par les canaux K et L, est modifiable, tandis que les deux processus

S et R ne le sont pas. Ainsi, les seules transitions contrôlables sont donc celles où un message est émis par K ou L. De plus, comme la synthèse de contrôleur "n'invente jamais", mais procède par restriction, nous avons équipé les transitions contrôlables de compteurs comptant les pertes de messages.

Pour calculer l'ensemble des états qui mènent de manière incontrôlables à un cas d'échec, nous avons utilisé le logiciel NBAC (Jeannet, 2003; NBA, n.d.) qui est un logiciel de vérification des systèmes à variables booléennes et numériques, faisant de l'interprétation abstraite en utilisant les BDD (Bryant, 1986) et les polyèdres convexes (Cousot *et al.*, 1978; Halbwachs *et al.*, 1997). L'analyse a permis de résoudre le problème en trouvant une relation entre $rmax$ et le nombre de messages perdus par les canaux K et L, qui se révèle être précise. Il est à noter que la relation symbolique obtenue est valable pour une constante $rmax$ *symbolique* et non nécessairement bornée. L'usage du modèle des LTS, en comparaison, ne permet pas de synthétiser des systèmes paramétrés. La version complète de cette étude de cas est disponible dans un rapport de recherche (Legall *et al.*, 2005).

6. Contrôle des systèmes hybrides

6.1. *Moyen de contrôle sur un HTS*

Nous avons discuté et défini en section 5.1 les moyens de contrôle que nous considérons sur les STS. La partie discrète d'un HTS étant un STS, nous conservons le modèle défini pour les STS, qui associe à chaque transition discrète $\delta \in \Delta$ une garde d'incontrôlabilité $G^{\delta}_{uc} \subseteq G^{\delta}$.

Nous nous focalisons donc sur les deux nouveautés introduites par le modèle des HTS, qui sont :

1) l'existence d'évolutions continues induites par le passage du temps ;

2) l'interaction entre transitions discrètes et passage du temps.

Dans la vision classique du contrôle, on considère comme contrôlable ce que l'on peut modifier. Mais un segment de trajectoire du système ne peux pas être classé directement comme contrôlable ou non :

– La dynamique continue du système ne peut pas être modifiée ; elle modélise en principe un environnement physique auquel il s'agit plutôt de s'adapter. En outre, dans notre modèle, cette dynamique est déterministe.

– Suspendre l'écoulement du temps n'a pas non plus de sens.

Nous considérons donc que les évolutions continues ne sont pas *directement* contrôlables. Nous nous plaçons donc dans le cadre de la synthèse de "switching controllers" (Asarin *et al.*, 2000), qui n'agissent que sur les événements discrets.[6] En revanche, un moyen *indirect* de contrôle du comportement continu est de jouer sur l'interac-

6. Remarquons toutefois que dans certains modèles (Tomlin *et al.*, 2000; Maler, 2002), le comportement continu est partiellement contrôlable. Mais ces modèles, qui n'ont pas de notion

tion entre transitions discrètes et passage du temps. En effet, nous pouvons éviter un état "mauvais" par anticipation, en forçant un changement de mode (une transition discrète) au bon moment. Par exemple, le conducteur d'une voiture anticipe le déplacement de son véhicule pour freiner à temps ; il ne peut pas modifier les lois de la physique qui régissent le déplacement de la voiture, mais il peut passer du mode "accélération" au mode"freinage". Ceci suppose la possibilité de forcer un changement de mode, c'est-à-dire dans notre cadre de donner la priorité aux transitions discrètes sur l'écoulement du temps.

Dans le cadre des HTS, nous disposons d'un moyen de contrôle sur l'écoulement du temps : l'invariant $H(\vec{q})$ associé à la partie discrète d'un état. Forcer une transition discrète consiste alors à restreindre la "garde temporelle" $H(\vec{q})$ pour que la seule possibilité soit de prendre la transition discrète contrôlable.

Ces considérations nous mènent à la définition suivante : les moyens d'actions du contrôle sont :

– de renforcer les gardes G des transitions discrètes δ par des gardes G' satisfaisant
$G_{uc} \Rightarrow G' \Rightarrow G$

– de renforcer l'invariant global H par H' en satisfaisant

$$H' \subseteq H \quad [17]$$

$$\forall \langle \vec{q}, \vec{x} \rangle \in S, \;\; \vec{x} \in H(\vec{q}) \setminus H'(\vec{q}) \Rightarrow \exists \delta \in \Delta, (\vec{q}, \vec{x}) \in Proj_S((G^\delta)' \setminus G^\delta_{uc}) \quad [18]$$

L'équation [18] signifie que l'on ne peut bloquer l'évolution continue dans un état $(\vec{q}, \vec{x})$ que s'il existe une transition discrète contrôlable autorisée dans le système final. Cette notion de contrôlabilité permet de faire une analogie avec le contrôle de STS pour lesquels on définit des événements prioritaires (Legall *et al.*, 2005).

Ceci est en fait la notion de contrôle des systèmes hybrides considérée (plus implicitement) dans (Asarin *et al.*, 2000). Nous pouvons donc considérer que le contrôle des systèmes hybrides est une extension de la notion de contrôle des STS. En revanche, comparé à (Asarin *et al.*, 2000), l'ajout dans notre modèle d'événements incontrôlables complique l'interaction entre transitions discrètes et évolutions continues.

6.2. *Définition constructive du système contrôlé maximal*

Soit un HTS $\mathcal{M} = (V, \Theta, \Sigma, \Delta, D, H)$ et $E \subseteq S$ un ensemble d'états à interdire. Nous supposons que $\mathcal{M}$ est non bloquant. Nous voulons obtenir un HTS contrôlé $\mathcal{M}' = (V, \Theta, \Sigma, \Delta', D, H')$ non bloquant, tel que les exécutions de ce nouvel automate ne passent pas par un état de E. Nous voulons de plus que l'automate contrôlé soit le plus permissif possible : toute exécution de $\mathcal{M}$ qui ne passe pas par un état de E doit être aussi une exécution de $\mathcal{M}'$. Pour calculer $\mathcal{M}'$, nous allons déterminer comme aux section4 et section5 l'ensemble $\mathrm{Coreach}(E)$ des états qui mènent de

d'événement, ne permettent pas d'analyser aussi finement l'interaction entre comportement discret et continu dans le cadre du contrôle.

manière incontrôlable à un état de E, étant donné les moyens de contrôle que l'on s'est donné. Il faut donc prendre en compte les évolutions continues dans l'opérateur de pré-condition Pre, ainsi que l'interaction entre transitions discrètes et évolutions continues.

Pour $\vec{q} \in Q$, $Y \subseteq Q \times X$ et $\delta \in \Delta$, nous définissons :

– l'ensemble des états qui peuvent mener à Y par un segment de trajectoire :

$$\mathsf{before}(\vec{q}, Y) \;=\; \{\langle \vec{q}, \vec{x}\rangle \mid \exists T > 0,\ \vec{\nu}_{(\vec{q},\vec{x})}(T) \in Y\} \qquad [19]$$

– l'ensemble des états qui peuvent s'échapper par une transition contrôlable vers $\overline{Y}$ en prenant la transition δ :

$$\mathsf{esc}(\delta)(Y) \;=\; \mathit{Proj}_S((G^\delta \setminus G^\delta_{uc}) \cap (A^\delta)^{-1}(\overline{Y})) \qquad [20]$$

Opérateur $\mathrm{Pre}^{\nearrow}$. Étant donné un état discret $\vec{q}$, nous cherchons maintenant à définir l'ensemble des états qui mènent à Y par passage du temps sans qu'il soit possible de s'échapper de manière contrôlable vers un état $\langle \vec{q}', \vec{x}'\rangle \in \overline{Y}$. Nous considérons successivement les ensembles d'états suivants :

– $\mathsf{before}(\vec{q}, Y) \cap \mathsf{esc}(\delta)(Y)$ est l'ensemble des états qui peuvent mener à Y par un segment de trajectoire, mais qui peuvent s'échapper vers $\overline{Y}$ par la transition discrète δ (Y étant considéré à éviter) ; il s'agit du sous-ensemble de $A \cap B$ sur la figure 3 ;

– $\mathsf{before}(\vec{q}, (\mathsf{before}(\vec{q}, Y) \cap \mathsf{esc}(\delta)(Y))$ est l'ensemble des états qui mènent par un segment de trajectoire aux états "bons" définis ci-dessus ; il s'agit de l'ensemble C sur la figure 3.

Nous en déduisons donc une formule donnant l'ensemble des états qui mènent à Y par passage du temps, sans qu'il soit possible de forcer une transition contrôlable pour l'éviter (ensemble $B \setminus C$ sur la figure 3) :

$$\mathrm{Pre}^{\nearrow}(\vec{q}, Y) = \mathsf{before}(\vec{q}, Y) \setminus \bigcup_{\delta \in \Delta} \mathsf{before}(\vec{q}, \mathsf{before}(\vec{q}, Y) \cap \mathsf{esc}(\delta)(Y)) \qquad [21]$$

$$\mathrm{Pre}^{\nearrow}(Y) = \bigcup_{\vec{q} \in Q} \mathrm{Pre}^{\nearrow}(\vec{q}, Y) \qquad [22]$$

Opérateur Pre^{nb}. Puisque le système contrôlé doit être non bloquant, il nous faut un opérateur qui élimine les états bloquants :

$$\mathrm{Pre}^{nb}(Y) = \bigcap_{\delta \in \Delta} \mathit{Proj}_S^{\forall}(\overline{G^\delta} \cup (A^\delta)^{-1}(Y)) \setminus H \qquad [23]$$

Un état est bloquant dans un HTS s'il est bloquant dans le STS sous-jacent (équation [13]) et si l'évolution continue est bloquée.

Opérateur Pre. On obtient donc l'opérateur Pre global :

$$\mathrm{Pre}(Y) \;=\; \mathrm{Pre}^{\nearrow}(Y) \cup \mathrm{Pre}^{\Delta}(Y) \cup \mathrm{Pre}^{nb}(Y) \qquad [24]$$

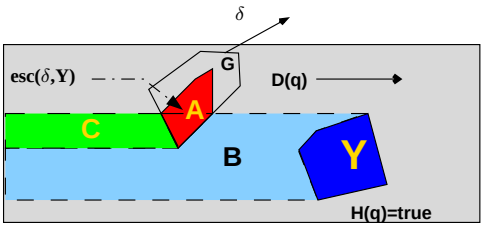

Figure 3. $\mathrm{Pre}^{\nearrow}(\vec{q}, Y)$

où Pre^{Δ} est l'opérateur défini par l'équation [10].

Système contrôlé maximal $\mathcal{M}'$. L'ensemble des états interdits E à éviter est $\mathrm{Coreach}(Y) = \mathrm{lfp}(\lambda Y.E \cup \mathrm{Pre}(Y))$.

Le système contrôlé $\mathcal{M}' = (V, \Theta', \Sigma, \Delta', D, H')$ est défini par :

– $\Theta' = \Theta \setminus \mathrm{Coreach}(E)$;

– Δ' est défini à partir de Δ, en utilisant la fonction $\mathcal{C}(\delta)$ (équation [12]), par

$$\frac{(\sigma, \vec{p}, G_{uc}, G, A) \in \Delta \quad G' = G \wedge \neg\mathcal{C}(\delta)}{(\sigma, \vec{p}, G_{uc}, G', A) \in \Delta'}$$

– $H' = (H \setminus \mathrm{Coreach}(E))^{\circ}$, qui est le plus grand ouvert contenu dans $H \setminus \mathrm{Coreach}(E)$ (pour le produit de la topologie discrète sur Q et euclidienne sur $X = \mathbb{R}^n$).

L'automate contrôlé respecte la propriété de sûreté et est non bloquant : c'est une conséquence directe des propriétés de $\mathrm{Coreach}(E)$. De plus on peut montrer que ce système est le plus permissif en utilisant le fait que $\mathrm{Coreach}(E)$ est le plus petit point-fixe de l'opérateur Pre contenant E.

Remarque 3 *Nous ne nous sommes pas préoccupés ici du problème des "cycles de Zénon" (l'automate a un cycle de transitions discrètes qui peuvent ainsi bloquer l'écoulement du temps). En effet, nous considèrons les HTS comme une extension des STS, autrement dit des systèmes qui peuvent (et non doivent) avoir un comportement continu. On peut toutefois montrer que si l'automate de départ n'a pas de cycle de Zénon, alors l'automate contrôlé n'en a pas non plus.*

6.3. *Intérêt de cette approche*

Le problème de la synthèse de contrôleurs pour des systèmes hybrides a déjà été étudié, y compris avec des modèles intégrant une notion d'événements discrets incontrôlables (Wong-Toi, 1997). Cependant, le modèle des HTS est un peu plus général que ceux déjà étudiés (paramètres de communication, affectations des variables).

Notre démarche est originale d'un point de vue conceptuel : au lieu de partir des systèmes hybrides et de rajouter la notion d'événements incontrôlables, nous sommes partis d'un modèle discret symbolique que nous avons étendu pour traiter le cas des systèmes hybrides. Ce glissement du monde des systèmes discrets au monde des systèmes hybrides présente deux intérêts :

1) les différents notions et méthodes sont introduites progressivement ;

2) les notions de contrôle définies sur les différents modèles sont cohérentes. En particulier, lorsqu'un HTS se réduit à un STS, c'est-à-dire lorsque le temps ne peut jamais s'écouler ($H = \mathrm{false}$), la définition de Pre dans le cadre des HTS, équation [24], se réduit a la définition de Pre dans le cadre des STS, équation [10].

6.4. *Calcul effectif du système contrôlé*

Comme pour les STS, nous ne pouvons pas calculer de manière exacte $\mathrm{Coreach}(E)$. Comme nous avons exprimé cet ensemble comme point-fixe d'opérateurs ensemblistes, on peut travailler dans un treillis abstrait comme au section5.3, en ajoutant les opérations abstraites correspondant aux opérateurs before, esc et $\mathrm{Pre}^{\nearrow}$. Toutefois, ces opérateurs font intervenir des différences ensemblistes, qui ne sont pas nécessairement facile à abstraire de manière satisfaisante, notamment dans le treillis des polyèdres convexes (nécessité de calculer des sous-approximations). Ils n'ont pas encore été définis et implémentés dans l'outil NBAC.

Une vaste littérature existe pour le contrôle de systèmes hybrides. Les travaux issus de la communauté informatique adoptent généralement comme nous un modèle avec notion d'événement et étudient la synthèse de "switching controllers", en utilisant des calculs de point-fixes. Pour la sous-classe des automates temporisés, le problème peut être résolu exactement et a été implémenté (Altisen *et al.*, 2002). L'outil HYTECH (Henzinger *et al.*, 1995) traite la classe des automates dont les gardes et invariants correspondent à des unions finies de polyèdres convexes, et dont les (in)équations différentielles sont définies par des conjonctions de contraintes linéaires sur les dérivées (ne faisant donc pas intervenir les fonctions elles-mêmes). HYTECH n'effectue que des calculs exacts, en manipulant des unions de polyèdres, ce qui fait que la convergence des calculs n'est pas garantie. d/dt (Dang, 2000) traite des systèmes avec des équations différentielles linéaires, plus générales, en calculant des sous- et sur-approximations d'ensembles numériques représentés par des unions d'hypercubes. Ceci permet de manipuler aisément des ensembles non convexes et de calculer des (approximations de) différences ensemblistes. L'inconvénient principal est celui de l'approximation de fonctions par des fonctions en escalier : la complexité croît très rapidement avec le pas de l'escalier, surtout en dimension supérieure.

Les travaux issus de la communauté automatique utilisent les outils plus traditionnels de l'analyse mathématique, en utilisant des approximations au sens topologique du terme et en généralisant les méthodes pour le contrôle des systèmes continus. (Tomlin *et al.*, 2000) présente une synthèse intéressante. Par ailleurs, certains aspects

sont importants pour les applications pratiques, telle la stabilité et la robustesse à des perturbations (Moor *et al.*, 2001).

7. Conclusion

Dans ce papier, nous avons présenté le problème de la synthèse de contrôleurs à travers différents modèles allant des systèmes de transitions finis (LTS) aux systèmes hybrides (HTS) en nous intéressant à des propriétés de sûreté. Après avoir discuté les différentes approches au problème du contrôle, qui varie selon les communautés de recherche concernées, nous avons adopté une vision raffinement de spécification (ou contrôle interne) par opposition au contrôle externe (ou en boucle fermée) traditionnellement utilisée dans la communauté automatique discrète. Si les *safety controllers*, dans le cas des systèmes de transitions finis, et les *switching controllers*, dans le cas automates hybrides, ont été largement étudiés durant les 20 dernières années, la synthèse de contrôleurs pour des systèmes intermédiaires du type STS n'a pas été un problème très considéré, alors même que de nombreux systèmes peuvent être modélisés par des STS et que leur contrôle est sensiblement plus simple que celui des systèmes hybrides.

Dans cette optique, nous avons regardé comment appliquer les techniques de synthèse sur des systèmes de transitions finis au cadre des systèmes de transitions symboliques (STS), modèle intermédiaire entre celui des LTS et celui des HTS. L'analyse des besoins de modélisation nous ont amené à faire porter le critère de contrôlabilité sur les gardes des transitions symboliques. Nous avons résolu le problème de la synthèse dans le cadre ainsi défini, puis montré comment utiliser l'interprétation abstraite pour obtenir un algorithme effectif de synthèse, permettant d'obtenir un système contrôlé non maximal. L'exemple du BRP a permis de donner une application pratique à cette théorie, à savoir la synthèse d'un environnement assurant la bonne transmission des messages par le protocole.

Nous avons enfin abordé le problème de la synthèse pour le modèle plus général des HTS, en réutilisant les techniques et la notion de contrôlabilité définies pour les STS. Nous avons ainsi formulé précisément la notion de priorité d'événements contrôlables sur l'écoulement du temps, et surtout le critère de contrôlabilité du temps, puis résolu le problème de la synthèse sur ces bases.

L'ensemble de ces résultats fournit un cadre théorique unifié pour la résolution du problème de synthèse de contrôleurs pour un ensemble consistant de modèles de systèmes (portant sur des propriétés de sûreté). Le futur développement d'un logiciel basé sur ces algorithmes, et utilisant les approximations données par l'outil de vérification NBAC permettra de faire de la synthèse de contrôleurs pour ces modèles.

8. Bibliographie

Abdulla P. A., Bouajjani A., Jonsson B., « On-the-fly analysis of systems with unbounded, lossy fifo channels », *Computer Aided Verification, CAV'98, LNCS* 1427, July, 1998.

Altisen K., Tripakis S., « Tools for Controller Synthesis of Timed Systems. », *2nd Workshop on Real-Time Tools (RT-TOOLS'2002)*, July, 2002.

Alur R., Courcoubetis C., Halbwachs N., Henzinger T., Ho P., Nicollin X., Olivero A., Sifakis J., Yovine S., « The Algorithmic Analysis of Hybrid Systems », *Theoretical Computer Science B*, vol. 138, p. 3-34, January, 1995.

Alur R., Dill D., « A theory of timed automata », *Theoretical Computer Science*, 1994.

Asarin E., Bournez O., Dang T., Maler O., Pnueli A., « Effective Synthesis of Switching Controllers for Linear Systems », *Proceedings of the IEEE, Special Issue "Hybrid System: Theory & Application*, vol. 88, p. 1011-1025, 2000.

Bouajjani A., « Languages, Rewriting Systems, and Verification of Infinite-State Systems », *Int. Colloqium on Automata, Languages and Programming, ICALP'01, LNCS* 2076, 2001.

Bryant R. E., « Graph-based algorithms for boolean function manipulation », *IEEE Transactions on Computers*, vol. C-35, n° 8, p. 677-692, 1986.

Burch J. R., Clarke E. M., McMillan K. L., Dill D. L., Hwang L. J., « Symbolic Model Checking: 10^{20} states and beyond », *Information and Computation*, 1992.

Cassandras C., Lafortune S., *Introduction to Discrete Event Systems*, Kluwer Academic, 1999.

Cousot P., Cousot R., « Static determination of dynamic properties of programs », *2nd Int. Symp. on Programming*, Dunod, Paris, 1976.

Cousot P., Cousot R., « Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints », *4th ACM Symposium on Principles of Programming Languages, POPL'77*, Los Angeles, January, 1977.

Cousot P., Cousot R., « Comparing the Galois connection and widening/narrowing approaches to abstract interpretation », *in* , M. Bruynooghe, , M. Wirsing (eds), *PLILP'92*, vol. 631 of *LNCS*, Leuven (Belgium), January, 1992.

Cousot P., Halbwachs N., « Automatic discovery of linear restraints among variables of a program », *5th ACM Symposium on Principles of Programming Languages, POPL'78*, Tucson (Arizona), January, 1978.

Dang T., Verification and Synthesis of Hybrid Systems, PhD thesis, Verimag, Institut National Polytechnique de Grenoble, 2000.

Emerson E., Lei C.-L., « Efficient model checking in fragments ot the propositional μ-calculus », *Proc. of the* 1^{st} *LICS*, 1986.

Finkel A., Leroux J., « How To Compose Presburger-Accelerations: Applications to Broadcast Protocols », *Foundations of Software Technology and Theoretical Computer Science, FSTTCS'02*, vol. 2556 of *LNCS*, p. 145-156, 2002.

Granger P., « Static analysis of arithmetical congruences », *International Journal on Computer Mathematics*, vol. 30, p. 165-190, 1989.

Halbwachs N., Proy Y., Roumanoff P., « Verification of real-time systems using linear relation analysis », *Formal Methods in System Design*, August, 1997.

Helmink L., Sellink M., Vaandrager F., « Proof-checking a data link protocol », *Proc. International Workshop TYPES'93*, vol. 806 of *LNCS*, 1994.

Henzinger T. A., Ho P.-H., Wong-Toi H., « A User Guide to HyTech », *Proceedings of the First International Workshop on Tools and Algorithms for Construction and Analysis of Systems*, Springer-Verlag, p. 41-71, 1995.

Henzinger T., Ho P.-H., Wong-Toï H., « HYTECH: A Model Checker for Hybrid Systems », *Computer Aided Verification, CAV'97*, number 1254 in *LNCS*, June, 1997.

Henzinger T., Manjumdar R., Raskin J.-F., « A Classification of Symbolic Transition Systems », 2002, Accepted for publication in ACM TOCL.

Jeannet B., « Representing and Approximating Transfer Functions in Abstract Interpretation of Hetereogeneous Datatypes », *Static Analysis Symposium, SAS'02*, LNCS 2477, 2002.

Jeannet B., « Dynamic Partitioning In Linear Relation Analysis. Application To The Verification Of Reactive Systems », *Formal Methods in System Design*, vol. 23, n° 1, p. 5-37, July, 2003.

Jeannet B., Jéron T., Rusu V., Zinovieva E., « Symbolic Test Selection based on Approximate Analysis », *11th Int. Conference on Tools and Algorithms for tthe Construction and Analysis of Systems*, vol. 3440 of *LNCS*, Edinburgh (UK), April, 2005.

Legall T., Jeannet B., Marchand H., Controle de systèmes symboliques, discrets ou hybrides, Publication interne n° PI-1683, IRISA, Campus de Beaulieu,Rennes,France, January, 2005.

Maler O., « Control from Computer Science », *Annual Reviews in Control*, 2002.

Maler O., A.Pnueli, J.Sifakis, « On the Synthesis of Discrete Controllers for Timed Systems », *Symposium on Theoretical Aspects of Computer Science (STACS'95), LNCS* 900, 2005.

Miné A., « The Octagon Abstract Domain », *AST'01 in Working Conference on Reverse Engineering 2001*, IEEE CS Press, October, 2001.

Moor T., Davoren J. M., « Robust Controller Synthesis for Hybrid Systems Using Modal Logic », *Hybrid Systems: Computation and Control (HSCC 2001)*, vol. 2034 of *LNCS*, 2001.

NBA, « The NBac verification tool », n.d. `http://www.irisa.fr/prive/bjeannet/nbac/nbac.html`.

Pnueli A., « The Temporal Logic of Programs », *Proc. of the 18th IEEE Symposium Foundations of Computer Science, FOCS'77*, p. 46-57, 1977.

Ramadge P. J., Wonham W. M., « Modular Feedback Logic for Discrete Event Systems », *SIAM J. Control Optim.*, vol. 25, n° 5, p. 1202-1218, September, 1987.

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, p. 81-98, 1989.

Tomlin C., Lygeros J., Sastry S., « A Game Theoretic Approach to Controller Design for Hybrid Systems », *Proc. of the IEEE*, July, 2000.

Vardi M., Wolper P., « Automata-Theoretic Techniques for Modal Logics of Programs », *Journal of Computer and System Science*, April, 1986.

Wolper P., Boigelot B., « Verifying Systems with Infinite but Regular State Spaces », *Computer Aided Verification, CAV'98*, vol. 1427 of *LNCS*, July, 1998.

Wong-Toi H., « The synthesis of controllers for linear hybrid automata », *Proc of the 36th IEEE Conference of Decision and Control*, San Diego, CA, p. 4607-4612, Decembre, 1997.

Article reçu le 25 janvier 2005
Version révisée le 16 juin 2005

Tristan Le Gall *est en doctorat à l'IRISA, dans l'équipe VerTeCs. Il s'intéresse à la vérification de systèmes communicants avec des techniques d'interprétation abstraite.*

Bertrand Jeannet *a obtenu sa thèse de doctorat en informatique en 2000 à l'Institut National Polytechnique de Grenoble. Il a passé un an à l'université d'Aalborg (Danemark), avant d'obtenir un poste de chargé de recherche à l'INRIA-Rennes, dans l'équipe VerTeCs. Ses principaux thèmes de recherche sont la vérification de programmes, l'interprétation abstraite, et leurs applications au test.*

Hervé Marchand *a obtenu sa thèse de doctorat en informatique en 1997 à l'université de Rennes 1. Il a séjourné un an, comme boursier post-doctoral, à l'université du Michigan en 1998, avant de rejoindre l'INRIA à Rennes en tant que chargé de recherche. Il est actuellement rattaché à l'équipe VerTeCs de l'IRISA. Ses domaines de recherche concernent le contrôle de systèmes à événements discrets, la théorie du test ainsi que le diagnostic.*