Contrôle de systèmes symboliques, discrets ou hybrides

Tristan Le Gall, Bertrand Jeannet, Hervé Marchand

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Contrôle de systèmes symboliques, discrets ou hybrides

Tristan Le Gall, Bertrand Jeannet & Hervé Marchand

N°5474

Janvier 2005

Systèmes communicants

ISSN 0249-6399 ISRN INRIA/RR--5474--FR

Contrôle de systèmes symboliques, discrets ou hybrides

Tristan Le Gall, Bertrand Jeannet & Hervé Marchand

Systèmes communicants
Projet VerTeCs

Rapport de recherche n ° 5474 — Janvier 2005 — 29 pages

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 d'assurer la finitude 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.

Mots-clé : systèmes de transition finis hybrides et symoliques, synthèse de contrôleurs, propriétés de sûreté, interprétation abstraire

(Abstract: pto)

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : 02 99 84 71 00 - International : +33 2 99 84 71 00
Télécopie : 02 99 84 71 71 - International : +33 2 99 84 71 71

Supervisory Control of Discrete or Hybrid Symbolic systems

Abstract: In this paper, we tackle the safety controller synthesis problem for various models ranging from finite transition systems to hybrid systems. Within this framework, we are mainly interested in an intermediate model: the symbolic transition system. Modeling requirements lead us to redefine the concept of controllability by applying it to the guards of symbolic transitions, instead to 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.

Key-words: Finite state machine, hybrid and symbolic systems, supervisory control problem, safety properties, abstract interpretation

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 entre autre 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. enfin, l'automatique des systèmes régis par des équations différentielles.

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'événements [22, 24]. 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 [20]). 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 [2]. 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 que l'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 différentes 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 [21] 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 équivalents, mais il convient de spécifier le point de vue adopté. La communauté systèmes à événement 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 revanche, 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. 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. Ce papier 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 les algorithmes de synthèse correspondants.

Nous commencerons par présenter au §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). 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 numériques entre deux transitions discrètes.

La sémantique de ces systèmes sera définie en terme de traces (ou de trajectoires pour les HTS).

Le §3 détaille les propriétés que nous considérerons pour le contrôle. Il s'agit essentiellement les 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 au §4. Nous aborderons ensuite une modélisation très

générale du problème de contrôle dans le cadre des STS au §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. Dans le §6, nous introduirons une notion de priorité entre les événements dans un modèle STS. Cette notion de priorité sera exploitée au §7 qui abordera le contrôle des systèmes hybrides, systèmes dans lesquelles existe un choix non-déterministe entre les transitions discrètes et l'écoulement du temps. Nous intégrerons la notion de contrôlabilité adoptée pour les STS et celle 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 au §2.1, les systèmes de transitions symboliques au §2.2 et les automates hybrides généralisés au §2.3. Nous discutons ensuite au §3 la sémantique adoptée pour ces automates et les propriétés que nous considérerons sur cette sémantique.

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.*

Soit $\mathcal{M} = (Q,q_o,\Lambda, \rightarrow)$ un LTS. On utilisera 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 $q = q_0 \xrightarrow{\sigma_0} q_1 \xrightarrow{\sigma_1} \cdots \xrightarrow{\sigma_{n-1}} q_n$ démarrant de l'état initial. Une trace est la projection d'une exécution sur les événements. L'ensemble des traces de $\mathcal{M}$ est noté $\mathcal{L}(\mathcal{M}) \subseteq \Lambda^*$. 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 $Q \times \Lambda \rightarrow Q$.

Une opération essentielle sur les LTS est le produit synchrone:

Définition 2 (Produit synchrone) *Soient* $\mathcal{M}^i = (Q^i,q_0^i,\Lambda, \rightarrow_i)$, $i = 1,2$, *deux LTS. Leur produit synchrone* $\mathcal{M}^1 \times \mathcal{M}^2$ *est le LTS* $\mathcal{M} = (Q,q_0,\Lambda, \rightarrow)$ *avec* $Q = Q^1 \times Q^2$, $q_0 = (q_0^1,q_0^2)$, *et* $\rightarrow$ *est la plus petite relation satisfiant la règle suivante:*

$$\frac{q^1 \xrightarrow{a}_1 q'^1 \quad q^2 \xrightarrow{a}_2 q'^2}{(q^1,q^2) \xrightarrow{a} (q'^1,q'^2)} \tag{1}$$

On a $\mathcal{L}(\mathcal{M}) = \mathcal{L}(\mathcal{M}^1) \cap \mathcal{L}(\mathcal{M}^2)$.

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

Remarque 1 *Nous ne distinguons pas événements visibles et événements internes. Aussi les définitions traditionnelles de langage et de trace coïncident ici.*

2.2 Systèmes de transitions symboliques

Un système de transitions symbolique (STS) est un système discret étendu avec des variables. 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 de type *compteur de programme*.

Notations. Pour un type de donnée t ou une variable v (de type t), nous noterons $\mathcal{D}_t$ ou $\mathcal{D}_v$ le domaine de valeurs associé. Nous étendons cette notation aux vecteurs et aux ensembles de variables $\vec{v}$ et V. Une condition ou une garde $G(\vec{v})$ portant sur les variables $\vec{v}$ est normalement un prédicat. Cependant, on emploiera la même notation pour dénoter le sous-ensemble de $\mathcal{D}_{\vec{v}}$ des valuations de $\vec{v}$ qui satisfont le prédicat. On notera $\bar{E}$ 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 3 (STS, syntaxe) *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. On notera $Q \triangleq \mathcal{D}_V$.*
- *$\Theta \subseteq Q$ est la condition (prédicat) 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. On notera $\mathcal{D}_\sigma \triangleq \mathcal{D}_{sig(\sigma)}$.*
- *Δ un ensemble fini de transitions symboliques. Chaque transition $\delta = \langle\sigma,p,G,A\rangle$, également notée $[\sigma(p) : G(\vec{v},\vec{p})?\ \vec{v}' = A(\vec{v},\vec{p})]$, est définie par*
  - *une action σ et un vecteur de paramètres $\vec{p} = \langle p_1,\ldots,p_k\rangle : sig(\sigma)$, qui sont des variables locales à la transition, correctement typées;*
  - *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 désormais étiquetées par des actions valuées, ainsi que par des gardes et des affectations sur les variables.

Exemple 1 *L'exemple suivant montre comment spécifier le calcul de la factorielle à l'aide d'un STS (figure 1). Celui-ci a deux variables entières: x, dans laquelle est stockée la valeur d'entrée, et y, qui accumule les résultats du calcul. L'alphabet des actions est composé de deux actions de communication recevoir?(p) et envoyer!(p) qui prennent chacune un paramètre entier, et d'une action interne, notée τ.*

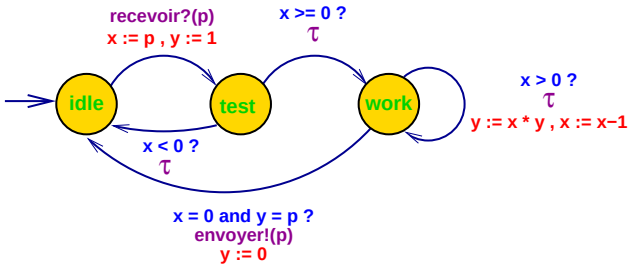

FIG. 1 – *STS calculant la factorielle*

Un STS est en réalité une syntaxe permettant de définir de manière finie un système de transitions infini.

Définition 4 (STS, sémantique) *La sémantique d'un STS $\mathcal{M} = (V,\Theta,\Lambda,\Delta)$ est le LTS $[\![\mathcal{M}]\!] = (Q,Q_o,\Lambda,\to)$, où*

- *$Q = \mathcal{D}_V$, $Q_0 = \Theta$;*
- *$\Lambda = \{\langle\sigma,\vec{\pi}\rangle \mid \sigma \in \Sigma \wedge \vec{\pi} \in \mathcal{D}_\sigma\}$;*

– $\rightarrow = \bigcup_{\delta \in \Delta} \rightarrow_\delta$ *est défini par la règle d'inférence*

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

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,

$$\forall \delta \in \Delta, \forall \vec{p} : \neg G^\delta(\vec{v},\vec{p}) \tag{2}$$

ie., 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 plus facile à vérifier sur la syntaxe d'un STS, à condition qu'une procédure de décision pour la satisfaisabilité des formules existe pour la classe de formules considérées. En contrôle, la notion de déterminisme est importante pour la raison suivante: pour un LTS ou un STS déterministe, la donnée d'un état initial et d'une trace définissent de manière unique une exécution.

La notion de produit synchrone s'étend aux STS.

Définition 5 (Produit synchrone de STS) *Soit $\mathcal{M}_i = (V_i,\Theta_i,\Sigma,\Delta_i)$ deux STS t.q. $V_1 \cap V_2 = \emptyset$. Le produit synchrone $\mathcal{M}_1 \times \mathcal{M}_2$ est le STS $\mathcal{M} = (V,\Theta,\Sigma,\Delta)$ défini par :*

– *$V = V_1 \cup V_2$ et $\Theta = \Theta_1 \wedge \Theta_2$;*

– *Δ est le plus petit ensemble de transitions satisfaisant la règle:*

$$\frac{[\sigma(\vec{p}) \ : \ G_1(\vec{v}_1,\vec{p}) \ ? \ \vec{v}_1' := A_1(\vec{v}_1,\vec{p})] \in \Delta_1 \quad [\sigma(\vec{p}) \ : \ G_2(\vec{v}_2,\vec{p}) \ ? \ \vec{v}_2' := A_2(\vec{v}_2,\vec{p})] \in \Delta_2}{[\sigma(\vec{p}) \ : \ G_1(\vec{v}_1,\vec{p}) \wedge G_2(\vec{v}_2,\vec{p}) \ ? \ \vec{v}_1' := A_1(\vec{v}_1,\vec{p}),\vec{v}_2' := A_2(\vec{v}_2,\vec{p})] \in \Delta}$$

On a la propriété $[\![\mathcal{M}_1 \times \mathcal{M}_2]\!] = [\![\mathcal{M}_1]\!] \times [\![\mathcal{M}_2]\!]$. Par ailleurs, le produit synchrone de deux STS (structurellement) déterministes est (structurellement) déterministe.

Remarque 2 *Il est possible de définir un modèle de STS et un produit dans lequel chaque automate peut accéder aux valeurs des variables de l'autre automate [18]. Par souci de simplicité, nous avons choisi de garder le modèle simple exposé ci-dessus, mais la possibilité pour un STS d'accéder à l'état interne d'un autre STS est très utile: elle permet d'écrire un observateur décrivant une propriété sur des exécutions et non seulement sur des traces (cf. §3).*

2.3 Automates hybrides généralisés

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

- (V,Θ,Σ,Δ) *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* $S = Q \times X = \mathcal{D}(V_Q) \times \mathcal{D}(V_X)$ *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 de classe* $\mathcal{C}^1$ *pour tout* $\vec{q} \in Q$.
- $H : Q \to \wp(X)$ *associe à chaque état discret* q *un* invariant *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.*

Remarque 3 *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, 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).*

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 7 (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* $\vec{\nu} : [0,T] \to S$, *avec* $\vec{\nu}(t) = \langle \vec{q},\vec{\xi}_{(\vec{q},\vec{x})}(t)\rangle$ *et* $\forall t \in [0,T[$: $\vec{\xi}_{(\vec{q},\vec{x})}(t) \in H(\vec{q})$.

Nous introduisons pour les besoins du §7 l'opérateur suivant, lié à cette définition. Soit un état discret $\vec{q}$ et un ensemble d'états $Y \subseteq S$. $\mathsf{before}(\vec{q},Y)$ désigne l'ensemble des états $\langle \vec{q},\vec{x}\rangle$ qui mènent à Y par un segment de trajectoire.:

$$\mathsf{before}(\vec{q},Y) \;=\; \{\langle \vec{q},\vec{x}\rangle \mid \exists T > 0,\, \vec{\nu}_{(\vec{q},\vec{x})}(T) \in Y\} \tag{3}$$

Par abus de notation, on désignera aussi par H l'ensemble: $\{\langle \vec{q},\vec{x}\rangle \mid x \in H(q)\}$. Soit $Y \subseteq Q \times X$, on définit la *fermeture temporelle* de Y l'ensemble $B(Y)$ des états qui, si on laisse le temps s'écouler quelques instants, mènent dans Y:

$$B(Y) = \{\langle \vec{q},\vec{x}\rangle \mid \exists \varepsilon > 0, \forall 0 < t < \varepsilon,\, \langle \vec{q},\vec{\xi}_{(\vec{q},\vec{x})}(t)\rangle \in Y\} \tag{4}$$

Remarque 4 *Dans la définition de* $B(Y)$, *on ne tient pas compte des invariants. On peut ainsi spécifier l'ensemble des états* $\langle \vec{q},\vec{x}\rangle$ *qui sont dans* H *mais dans lequel la trajectoire calculée fait immédiatement sortir de* H *(d'où la notion de frontière). Si* $\bar{H}$ *désigne le complémentaire de* H, *alors cet ensemble est* $H \cap B(\bar{H})$.

Définition 8 (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,Θ,Σ,Δ) *(cf. définition 4).*

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 considérés dans l'article.

Un HTS est (structurellement) *déterministe* si le STS sous-jacent est (structurellement) déterministe, et si 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, on a 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 $\langle \vec{q},\vec{x} \rangle \in B(\bar{H})$ (aucune évolution continue n'est possible). Un HTS est *non-bloquant* s'il n'existe aucune exécution menant dans un état bloquant.

Définition 9 (Produit synchrone de HTS) *Soit* $\mathcal{M}_i = (V_i,\Theta_i,\Sigma,\Delta_i,D_i,H_i)$ *deux SHS t.q.* $V_1 \cap V_2 = \emptyset$. *Le produit synchrone* $\mathcal{M}_1 \times \mathcal{M}_2$ *est le HTS* $\mathcal{M} = (V,\Theta,\Sigma,\Delta,D,H)$ *défini par :*

- (V,Θ,Σ,Δ) *est le produit des STS sous-jacents à* $\mathcal{M}_1$ *et* $\mathcal{M}_2$;
- $D(\vec{q}_1,\vec{q}_2)(\vec{x}_1,\vec{x}_2) = \langle D_1(\vec{q}_1)(\vec{x}_1),D_2(\vec{q}_2)(\vec{x}_2)\rangle$;
- $H(\vec{q}_1,\vec{q}_2)(\vec{x}_1,\vec{x}_2) = H_1(\vec{q}_1)(\vec{x}_1) \wedge H_2(\vec{q}_2)(\vec{x}_2)$

Dans le produit, les deux HTS composés se synchronisent non seulement par synchronisation d'actions, mais aussi implicitement par synchronisation sur le temps global. La remarque 2 reste pertinente pour les HTS.

3 Propriétés et Observateurs

Nous précisons maintenant les propriétés que nous considérons dans le cadre de la synthèse de contrôleurs. Les notions introduites sont valables pour les trois modèles d'automates qui ont été définis au §2.

3.1 Propriétés de sûreté

Propriétés d'invariance. 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, on peut 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.*

Propriétés de sûreté. 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. Une *propriété de sûreté* P est un ensemble de séquences telle que

$$\rho \notin P \Rightarrow \forall \rho' : \rho \cdot \rho' \notin P.$$

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. Intuitivement, une propriété de sûreté spécifie quelque chose de mauvais ne se produira pas, par opposition aux propriétés de vivacité qui spécifient que quelque chose de bon finira par se produire, dans un futur non borné. L'absence de panne, de défaillance, ou bien la conformité sont des exemples classiques de propriétés de sûreté. Notons qu'une propriété d'invariance est un cas particulier de propriété de sûreté.

Propriétés sur les traces ou sur les exécutions? Une trace, contrairement à l'exécution dont elle est la projection, ne contient plus d'information sur les états de l'automate. Une propriété d'invariance ne peut donc pas être spécifiée par une sémantique de traces. La communauté système à événements discrets qui travaille sur le modèle des LTS s'est principalement intéressé à des propriétés sur les traces, et fait correspondre propriété et langage sur l'alphabet des événements[1]. En revanche, la communauté systèmes hybrides qui travaille sur les HTS considère généralement des propriétés sur les exécutions, et en particulier des propriétés d'invariance.

3.2 Observateurs et réduction d'une propriété de sûreté à une propriété d'invariance.

Il est toujours possible de réduire une propriété de sûreté à une propriété d'invariance par le biais d'*observateurs*.

Observateurs. Étant donné un automate $\mathcal{M}$, un observateur est un automate $\mathcal{O}$ muni d'un état distingué $\mathsf{Violate}$ et qui est *non-intrusif*, c'est-à-dire vérifiant la propriété $\mathcal{L}(\mathcal{M} \times \mathcal{O}) = \mathcal{L}(\mathcal{M})$. *Le langage reconnu par un observateur* $\mathcal{O}$, noté $\mathcal{L}_{\mathsf{Violate}}(\mathcal{O})$, est l'ensemble des traces menant à l'état $\mathsf{Violate}$. On peut remarquer que dans le cadre des LST ou STS, si l'automate est *complet*[2] alors il est non-intrusif. Cette condition n'est toutefois pas suffisante dans le cadre des HTS, pour lesquels il faut également s'assurer que la synchronisation via le temps global ne restreint pas le comportement de $\mathcal{M}$. Une condition suffisante sur $\mathcal{O}$ est typiquement $\forall \vec{q} \in Q : H(\vec{q}) = \text{true}$.

Un observateur code la négation d'une propriété de sûreté $\varphi_{\mathcal{O}}$. En effet le langage reconnu par $\mathcal{O}$, $\mathcal{L}_{\mathsf{Violate}}(\mathcal{O})$, 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.

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 $\mathcal{M} \times \mathcal{O}$:

Proposition 1 *Soit $\mathcal{M}$ un automate, dont l'alphabet (valué dans le cas des STS et HTS) est Λ. Soit $\mathcal{O}$ un observateur pour $\mathcal{M}$ définissant la propriété de sûreté $\varphi_{\mathcal{O}}$. Alors $\mathcal{M}$ satisfait $\varphi_{\mathcal{O}}$ ssi $\mathcal{M} \times \mathcal{O}$ satisfait la propriété d'interdiction d'état $S^{\mathcal{M}} \times \{\mathsf{Violate}^{\mathcal{O}}\}$.*

Remarque 5 *Dans le cas des LTS, on peut associer à toute propriété de sûreté un observateur de cette propriété (qui est un automate régulier). En revanche, dans le cas des STS ou des HTS, il est possible qu'il n'existe aucune représentation syntaxique finie (ie., aucun modèle/programme) générant exactement un ensemble de traces défini en intention.*

Remarque 6 (Propriétés sur les exécutions) *Lorsque l'on a besoin de spécifier des propriétés sur les exécutions de $\mathcal{M}$ (par exemple, des propriétés d'invariance), il faut utiliser des observateurs capables d'examiner l'état interne de $\mathcal{M}$, ce qui alourdit les définitions, cf. remarque 2. À défaut, il est toujours possibles de rajouter à $\mathcal{M}$ une transition étiquetée par une action qui exporte l'état interne de $\mathcal{M}$ (dans le cas symbolique).*

Grâce à la proposition 1, dans la suite de l'article nous ne considérerons plus que des propriétés d'invariance définie par un ensemble d'états interdits. Contrôler le système $\mathcal{M}$ pour assurer un propriété de sûreté modélisée par un observateur $\mathcal{O}$ revient à contrôler $\mathcal{M} \times \mathcal{O}$ et à assurer sur celui-ci la propriété d'interdiction d'états $S^{\mathcal{M}} \times \{\mathsf{Violate}^{\mathcal{O}}\}$.

1. Notons que des approches à base de prédicats d'états ont été développées notamment dans [21, 23].

2. *ie.*, sans blocage, et tel que pour tout état et tout événement, il existe une transition discrète ayant pour origine cet état et pour étiquette cet événement.

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 [22], restreinte au problème de l'interdiction d'états (voir [21] 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. Nous aurons essentiellement une approche état, plutôt qu'une approche langage.

Définition du problème de contrôle D'après nos critères de classification, rappelons que 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 on dispose 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. On commence donc par calculer le plus petit ensemble $E' \supseteq E$ qui vérifie cette propriété. À cet effet, on introduit l'opérateur suivant:

$$\begin{array}{rccl} \mathrm{Pre}_{uc} : & \wp(Q) & \rightarrow & \wp(Q) \\ & X & \mapsto & \{q \in Q \mid \exists q' \in X, \exists a \in \Lambda_{uc} : q \xrightarrow{a} q'\} \end{array} \tag{5}$$

$\mathrm{Pre}_{uc}(X)$ est l'ensemble des états qui peuvent mener à X 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_{uc} contenant E:

$$E' = \mathrm{Coreach}_{uc}(E) \triangleq \mathrm{lfp}(\lambda X \ . \ E \cup \mathrm{Pre}_{uc}(X)) \tag{6}$$

On définit maintenant une fonction de contrôle $\mathcal{C}$, qui spécifie pour un état q quelles sont les transitions sortantes à interdire.

$$\mathcal{C}(q) = \begin{cases} \emptyset & \text{si } q \in \mathrm{Coreach}_{uc}(E) \\ \{(a,q') \in \Lambda \times Q \mid q \xrightarrow{a} q' \wedge q' \in \mathrm{Coreach}_{uc}(E)\} & \text{sinon} \end{cases} \tag{7}$$

Par définition de $\mathrm{Coreach}_{uc}$, $\mathcal{C}(q) \subseteq \Lambda_c \times Q$ 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 $Q'_0 = Q_0 \setminus \mathrm{Coreach}_{uc}(E)$ et

$$\frac{q \xrightarrow{a} q' \quad (a,q') \notin \mathcal{C}(q)}{q \xrightarrow{a}{}' q'} \tag{8}$$

L'existence d'un système contrôlé maximal découle directement de l'existence d'un *plus petit* point-fixe $\mathrm{Coreach}_{uc}(E)$, 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 $Q'_0 \in \overline{\mathrm{Coreach}_{uc}(E)}$, le système contrôlé $\mathcal{M}'$ vérifie la propriété d'interdiction d'états $\mathrm{Coreach}_{uc}(E) \supseteq E$. En effet ses états initiaux vérifie cette propriété, et la fonction $\mathcal{C}$ interdit toutes les transitions de $\mathcal{M}$ faisant sortir de l'ensemble d'états $\overline{\mathrm{Coreach}_{uc}(E)}$. On en déduit la condition de contrôlabilité d'un LTS $\mathcal{M}$ vis-à-vis d'une propriété E.

Définition 10 ($\mathcal{M}$ contrôlable vis-à-vis de la propriété E) *Un LTS $\mathcal{M} = (Q, Q_0, \Sigma_{uc} \cup \Sigma_c, \rightarrow)$ est* contrôlable vis-à-vis de la propriété d'interdiction d'état E *ssi,* $Q_0 \not\subseteq \mathrm{Coreach}_{uc}(E)$.

En d'autre terme, un système sera considéré comme incontrôlable si son état initial peut mener de manière incontrôlable à une violation de la propriété.

Calcul effectif du système contrôlé. Le théorème de Kleene et le fait que $(\wp(Q), \subseteq)$ est un treillis de hauteur finie garantissent que la suite définie par $E_0 = E$, $E_{n+1} = E_n \cup \mathrm{Pre}_{uc}(E_n)$ converge en un nombre fini d'étapes vers l'ensemble $\mathrm{Coreach}_{uc}(E)$, qui lui-même définit le contrôleur $\mathcal{C}$.

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[3]. De manière à assurer cette propriété, la contrainte de non-blocage est directement intégrée dans l'opérateur calculant les nouveaux états à interdire:

$$\mathrm{Pre}^{nb}_{uc}(X) = \mathrm{Pre}_{uc}(X) \cup \left\{ q \;\middle|\; \begin{array}{l} \forall a \in \Lambda_c : q \xrightarrow{a} q' \implies q' \in X \\ \wedge\, \forall a \in \Lambda_{uc} : \neg(q \xrightarrow{a}) \end{array} \right\} \tag{9}$$

$$= \mathrm{Pre}_{uc}(X) \cup \{ q \mid \forall a \in \Lambda : q \xrightarrow{a} q' \implies q' \in X \} \tag{10}$$

L'équation (9) rajoute à $\mathrm{Pre}_{uc}(X)$ les états dont toutes les transitions contrôlables mènent à X et qui n'ont pas de transitions incontrôlables[4]. En effet, dans ce cas, le contrôle consistera à couper toutes les transitions sortantes et le système sera alors en blocage. L'équation (10) simplifie l'expression de $\mathrm{Pre}^{nb}_{uc}(X)$ en remplaçant l'union disjointe par une union non disjointe.

En effectuant le même calcul de point-fixe que précédemment, l'ensemble $\mathrm{Coreach}^{nb}_{uc}(E)$ obtenu avec ce nouvel opérateur vérifie la propriété suivante:

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

Par conséquent, le système obtenu est non-bloquant.

Le contrôle des LTS, avec pour objectif l'interdiction de certains états, tel qu'exposé dans ce paragraphe, 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 paragraphes suivants développent ces notions dans le cadre de systèmes symboliques.

5 Contrôle des systèmes de transitions symboliques

5.1 Moyen de contrôle sur un STS

Dans un STS on distingue les actions (noms d'événements) comme σ des événements valués comme $\sigma(3,5)$. Par ailleurs on dispose 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. On partitionne $\Sigma = \Sigma_{uc} \cup \Sigma_c$, ce qui engendre une partition des actions valuées Λ. De manière cohérente, le contrôle consistera à interdire dans une transition symbolique une action valuée $\sigma(\vec{p})$ quelle que soit la valeur des paramètres, *ie.* 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, on associe 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 $c_\sigma(\vec{p}) = \mathrm{false}$.

3. 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.

4. Les transitions incontrôlables sont prises en compte dans $Pre_{uc}(E)$

De manière cohérente, le contrôle consistera 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 $G(\vec{v},\vec{p}) \wedge \neg c(\vec{p})$, avec $c(\vec{p}) \implies c_\sigma(\vec{p})$.

Exemple 2 *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* Interrupt(n)*, on peut modéliser le fait que le signal peut être inhibé ssi 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 3 *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 7 *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 du §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 spécifier finement dans la spécification initiale ce qui peut-être raffiné (interdit) de ce qui doit rester inchangé. Nous adoptons donc le modèle suivant:

Définition 11 *Un* STS intégrant la notion de contrôlabilité *est STS défini comme dans la définition 3, 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 que*

$$\forall \vec{v} \in Q, \forall \vec{p} \in \mathcal{D}_\sigma : G_{uc}(\vec{v},\vec{p}) \implies \begin{array}{l} \textit{l'événement } \sigma(\vec{p}) \textit{ est incontrôlable} \\ \textit{lorsque le système est dans l'état } v \end{array}$$

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 le cadre ainsi défini, les *moyens d'actions* du contrôle est de renforcer les gardes G des transitions symboliques par des gardes G' satisfaisant

$$G_{uc} \implies G' \implies G \tag{11}$$

5.2 Définition constructive du système contrôlé maximal

Le principe de la synthèse du système contrôlé, exposé au §4 reste identique: on définit le plus petit ensemble d'états à interdire, étant donné les contraintes d'incontrôlabilité, puis on modifie 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 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:

$$\begin{aligned}
\mathrm{Pre}_{uc}(\delta)(X)(\vec{v}) &= \exists\vec{p}\exists\vec{v}' : G_{uc}(\vec{v},\vec{p}) \wedge \vec{v}' = A(\vec{v},\vec{p}) \wedge X(\vec{v}') && (12)\\
\mathrm{Pre}_{uc}(\delta)(X) &= Proj_Q\left(G_{uc} \cap A^{-1}(X)\right) && (13)\\
\mathrm{Pre}_{uc}(X)(\vec{v}) &= \bigvee_{\delta\in\Delta} \mathrm{Pre}_{uc}(\delta)(X)(\vec{v}) && (14)\\
\mathrm{Pre}_{uc}(X) &= \bigcup_{\delta\in\Delta} \mathrm{Pre}_{uc}(\delta)(X) && (15)
\end{aligned}$$

On définit ensuite $\mathrm{Coreach}_{uc}(E) = \mathrm{lfp}(\lambda X.E \cup \mathrm{Pre}_{uc}(X))$.

On introduit la fonction auxiliaire $\mathcal{C}$ qui spécifie les transitions à interdire:

$$\begin{aligned}
\mathcal{C}(\delta)(\vec{v},\vec{p}) &= \neg\mathrm{Coreach}_{uc}(E)(\vec{v}) \wedge \mathrm{Coreach}_{uc}(E)\left(A^\delta(\vec{v},\vec{p})\right) && (16)\\
\mathcal{C}(\delta) &= (A^\delta)^{-1}(\mathrm{Coreach}_{uc}(E)) \setminus (\mathrm{Coreach}_{uc}(E) \times \mathcal{D}_{\sigma^\delta}) && (17)
\end{aligned}$$

Par définition de $\mathrm{Coreach}_{uc}(E)$, 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 pour les LTS, le fait de contrôler le système peut introduire des blocages qui n'existaient pas. En supposant que le système initial $\mathcal{M}$ ne contient pas de blocage, on assure 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 (10) (plutôt qu'à l'équation (9)), il s'agit de rajouter à $\mathrm{Pre}_{uc}(X)$ les états qui mènent nécessairement à X, quelle que soit la transition considérée. Cet ensemble Y se définit ainsi:

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

Ce qui se traduit en formulation ensembliste par

$$Y = \bigcap_{\delta\in\Delta} Proj^\forall_Q\left(\overline{G^\delta} \cup (A^\delta)^{-1}(X)\right) \quad (18)$$

On en déduit l'expression de l'opérateur de pré-condition

$$\mathrm{Pre}^{nb}_{uc}(X) = \mathrm{Pre}_{uc}(X) \cup \bigcap_{\delta\in\Delta} Proj^\forall_Q\left(\overline{G^\delta} \cup (A^\delta)^{-1}(X)\right) \quad (19)$$

5.3 Définition du système contrôlé maximal

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

- $\Theta' = \Theta \setminus \mathrm{Coreach}_{uc}(E)$;
- Δ' est défini à partir de Δ par $\dfrac{(\sigma,\vec{p},G_{uc},G,A) \in \Delta \quad G' = G \wedge \neg\mathcal{C}(\delta)}{(\sigma,\vec{p},G_{uc},G',A) \in \Delta'}$

Il est facile de voir qu'aucune exécution de $\mathcal{M}'$ ne peut atteindre $\mathrm{Coreach}_{uc}(E)$, et donc E. En effet, les états initiaux de $\mathcal{M}'$ ne sont pas dans $\mathrm{Coreach}_{uc}(E)$, et aucune transition dans Δ' ne peut faire passer l'état courant de $\neg\mathrm{Coreach}_{uc}(E)$ à $\mathrm{Coreach}_{uc}(E)$, *cf.* équation (17).

On en déduit la condition de contrôlabilité d'un STS $\mathcal{M}$ vis-à-vis d'une propriété E, traduction symbolique de la définition 10 pour les LTS.

Définition 12 ($\mathcal{M}$ contrôlable vis-à-vis de la propriété E) *Un STS $\mathcal{M} = (V,\Theta,\Sigma,\Delta)$ est* contrôlable vis-à-vis de la propriété d'interdiction d'état E *ssi,* $\Theta \not\subseteq \mathrm{Coreach}_{uc}(E)$.

5.4 Calcul effectif du système contrôlé

Comme pour les LTS, le calcul du STS contrôlé repose sur le calcul par résolution de point-fixe de $\text{Coreach}_{uc}(E)$. 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[5]. Il existe trois approches pour contourner ce problème:

1. Identifier des sous-classes de systèmes infinis pour lesquels ce type de calcul reste décidable (voir [14]);
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) [1, 4, 25, 10]. 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.

Nous suivons dans cet article cette dernière approche. L'observation importante est la suivante: si on remplace dans les équations précédentes $\text{Coreach}_{uc}(E)$ 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 l'on veut 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 $\text{Coreach}_{uc}(E)$. 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.

5.4.1 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. On a 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 [6]. 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: $\text{lfp}(F) \subseteq \gamma(\text{lfp}(F^\sharp))$; *ie.*, 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 chaines infinies strictement croissante. Exiger que A ne contienne pas de telles chaines est trop restrictif. De meilleurs résultats peuvent être obtenus en utilisant un opérateur d'élargissement [7], qui contrairement aux techniques d'accélération permettent de converger (i) en un nombre d'étape *fini*, (ii) vers un *post*-point-fixe y de $F^\sharp$ (*ie.*, tel que $y \sqsupseteq F^\sharp(y) \sqsupseteq \text{lfp}(F^\sharp)$).

Un exemple très classique d'interprétation abstrait est l'analyse de relations linéaires [8], 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.

5. 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).

5.4.2 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 $\mathcal{M} = (V,\Theta,\Sigma,\Delta)$. Supposons que l'on dispose 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, on suppose que la connexion de Galois peut s'étendre à $\wp(Q \times \mathcal{D}_\sigma) \leftrightarrows A_\sigma$.

On transpose les équation du §5.2 comme suit:

$$\mathrm{Pre}^\sharp_{uc}(\delta)(Y \in A) = \alpha\left(Proj_Q(G^\delta_{uc} \cap (A^\delta)^{-1}(\gamma(Y)))\right) \quad (20)$$

$$\mathrm{Pre}^\sharp_{uc}(Y \in A) = \bigsqcup_{\delta\in\Delta} \mathrm{Pre}_u c^\sharp(\delta)(Y) \quad (21)$$

et on résout l'équation $Y = \alpha(E) \sqcup \mathrm{Pre}^\sharp_{uc}(Y)$, en obtenant un post-point-fixe noté $\mathrm{Coreach}^\sharp_{uc}(E)$.[6] Le contrôleur $\mathcal{C}$ et le système contrôlé sont alors définis comme précédemment à partir de $\gamma(\mathrm{Coreach}^\sharp_{uc}(E))$.

Exemple 4 *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 [12, 13], à la différence que les variables booléennes sont codées dans un automate de contrôle dans ces travaux. Par souci de simplicité, [12, 13] 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 pour la partie numérique, il faut utiliser les techniques plus fines de [17, 16].*

5.5 Étude de cas: le Bounded Retransmission Protocol(BRP)

5.5.1 Présentation du BRP

Le Bounded Retransmission Protocol (BRP) 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). Le fichier à transmettre est découpé en N 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(Ack) à travers le canal L. Les 3 bits d'information (f,l,b) indiquent si la frame est la première ($f = 1$), la dernière ($l = 1$) et b 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.

Plus précisément, une transmission s'effectue selon le scénario suivant. 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 $rmax$ met fin à la transmission. Le receveur R attend les frames, tant que S essaie de lui en envoyer. Dès qu'il reçoit une frame, il envoie un message d'acquittement et compare b avec son bit de référence. Si les 2 sont égaux, ca veut dire que le message reçu est le même que le précédent (il est alors ignoré). Si c'est un nouveau message, on change le bit de référence. Chaque processus a une variable d'état qui traduit, de son point de vue, l'état de la transmission. Pour S, soit on ignore tout (état $\bot$), soit la transmission est réussie (état OK) ou ratée (état NOK). Il se peut aussi que S ait transmis toutes les frames mais n'ait pas reçu de message d'acquittement pour la dernière (état DK). Pour R, soit on ignore tout (état $\bot$), soit la transmission est amorcée (état FST), en cours (état INC), achevée (état OK) ou interrompue (état NOK).

6. Pour l'équation (20), il est possible d'abstraire de manière plus compositionnelle, mais au détriment de la précision.

5.5.2 Modélisation du BRP en STS

Pour simplifier les figures, on s'autorise à passer des variables comme paramètres de communication. Si x est une variable, [recevoir?$(x) : \ldots$] signifie [recevoir?$(p) : \ldots$? $x' := p, \ldots$], et [envoyer!$(x) : \ldots$] signifie [envoyer!$(p) : p = x \wedge \ldots$? $\ldots$]. Le nom des actions est de la forme $X \to Y(msg)$ où X est l'envoyeur, Y le destinataire et msg le message transmis.

L'ioSTS représentant l'émetteur S est donné par la figure 2. Les canaux de communication K et L sont modélisés par deux automates (figure 3). Le fait d'avoir une erreur de transmission est modélisé par un message *time-out* TO envoyé à S. Enfin, on modélise le receveur R (figure 4). Tel que le BRP a été modélisé, il y a des messages échangés directement entre S et R: $S \to R(End)$ et $R \to S(End)$. Ce ne sont pas de vrais messages, cela représente juste le fait que, s'il n'y a aucun message échangé pendant une longue durée, la transmission est finie et on se retrouve dans l'état initial.

5.5.3 Application de l'algorithme de contrôle des STS

Sur cet exemple, on va utiliser les techniques de synthèse de contrôleurs pour faire du raffinement de spécification. Le problème qui nous intéresse est le suivant: *quelles doivent être les contraintes sur l'environnement pour que le protocole n'échoue jamais?*

Une fois les quatre sous-systèmes modélisés, on en fait le produit mixte (synchronisation seulement sur les actions communes) pour obtenir au final un STS modélisant le fonctionnement global du protocole. Il faut maintenant déterminer quelles sont les transitions contrôlables et incontrôlables. Étant donné le problème de contrôle 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.

Tel que le BRP est défini, on peut voir que la transmission est un échec si on se trouve dans un état tel que $ES = NOK$. Le problème du contrôle est donc de déterminer quelles doivent être les conditions sur les transitions de K et L de manière à interdire l'ensemble des états E du système global où la variable $ES = NOK$. Il est clair que la nécessité d'interdire ces transitions sera lié au nombre d'erreurs de transmission. Aussi on rajoute dans les automates de K et de L deux variables k et l pour compter le nombre d'erreurs de transmission de chaque canal.

Pour calculer l'ensemble des états interdits E et ceux qui mènent de manière incontrôlables à E, on utilise le logiciel NBAC [17, 19] qui est un logiciel de vérification des systèmes à variables booléennes et numériques, faisant de l'interprétation abstraite en utilisant les BDD [5] et les polyèdres convexes [8, 12].

L'analyse avec NBAC permet d'identifier deux mauvaises situations ($rmax$ est une constante fixée par le protocole, qui donne le nombre maximal de retransmission avant échec):

- $k \geq rmax$: le receveur est dans la localité *new*, aucune trame ne lui est parvenue car toutes ont été bloquées par le canal K.
- $k + l \geq rmax$: le receveur est dans la localité *idle*, il y a eu une ou plusieurs frames effectivement reçues, mais la transmission a été interrompue à cause de perte de message sur les canaux K et L.

On peut donc raffiner le système en "imposant" un nombre limité d'erreurs. On restreint les gardes des transitions $K \to S!(TO)$ et $L \to S!(TO)$ en mettant la condition $k + l < rmax$. On obtient ainsi un système qui transmet à coup sûr les messages. Si l'on veut maintenant extraire l'environnement du système composé, il faut rendre interne toutes les actions non utilisées par K et L. On obtient alors l'environnement contraint maximal qui assure que le protocole n'échoue jamais.

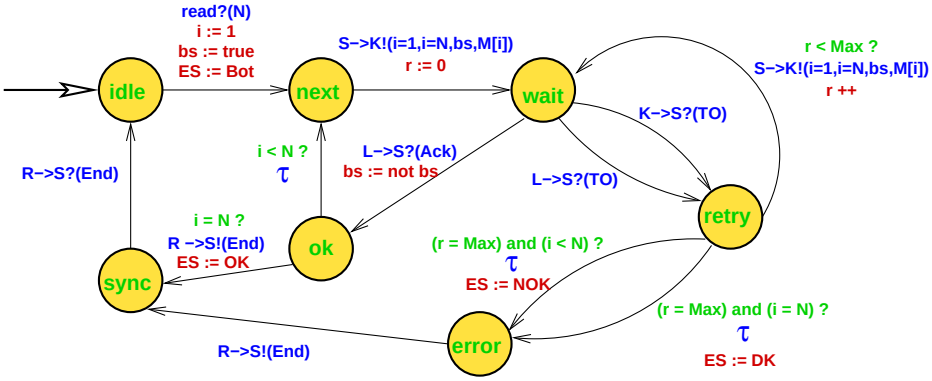

FIG. 2 – *L'émetteur S*

FIG. 3 – *Les canaux K et L*

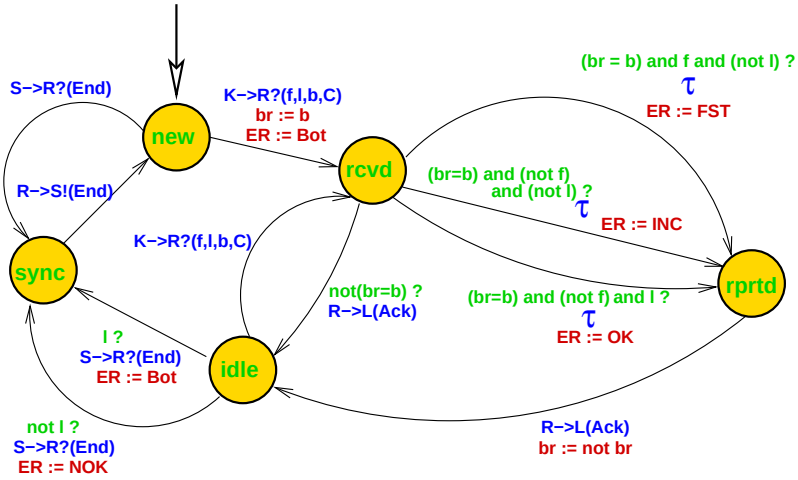

FIG. 4 – *Le receveur R*

6 Contrôle des systèmes symboliques avec notion de priorité entre les événements

6.1 Moyen de contrôle sur un STS avec priorité

Certains auteurs ont introduit, pour faire du contrôle sur des modèles proches des STS, une notion de priorité entre les événements [11]. Nous allons reprendre ce concept et l'adapter au modèle des STS.

Cette notion de priorité, comme celle d'événement contrôlable ou incontrôlable, n'a pas de sémantique intrinsèque; son rôle est de raffiner la notion de contrôlabilité. Intuitivement, on va s'autoriser à interdire des événements *incontrôlables* (et non prioritaires) lorsque des événements *prioritaires* (et contrôlables) sont possibles. Ceci revient à considérer un événement prioritaire comme une interruption qui peut interrompre l'exécution standard du système (et qui permet ici d'ignorer un événement incontrôlable émis par l'environnement, par exemple). Lorsque ce cas se présente, on dira qu'on oblige l'activation d'un événement prioritaire (en inhibant les événements non prioritaires). La condition supplémentaire que, parmi les événements prioritaires, seuls ceux qui sont *contrôlables* peuvent inhiber un événement incontrôlable, provient de l'intuition que dans la synthèse on a la maîtrise des événements contrôlables, mais pas des incontrôlables. Ce choix, qui est affaire de modélisation, sera conforté dans l'application de cette notion de priorité aux systèmes hybrides, au §7.

Exemple 5 *Si t est une variable et que l'on a deux transitions, l'une incontrôlable et menant dans un état interdit, avec des gardes $G^1 = \{t \leq 5\}$ et $G^1_{uc} = G^1$, l'autre contrôlable et menant dans un "bon état", avec des gardes $G^2 = \{t \leq 3\}$ et $G^2_{uc} = \emptyset$. Sans notion de priorité, il faut interdire les états $\{t \leq 5\}$.*

Si maintenant on considère la seconde transition comme prioritaire, on s'autorise à remplacer G^1 par $(G^1)' = \{3 < t \leq 5\}$, ce qui revient à inhiber la transition 1 lorsque la transition 2 est activable. Ce faisant, il ne faut plus interdire que les états $\{3 < t \leq 5\}$.

Nous mettons en évidence dans la suite et au §6.3 que cette notion de priorité n'est pas traduisible dans le modèle de contrôlabilité classique dans le cas général, même en modifiant les gardes d'incontrôlabilité

Nous formalisons maintenant le contrôle de système avec priorités.

Définition 13 *Un* STS intégrant les notions de contrôlabilité et de priorité entre les événements *est STS défini comme dans la définition 11, avec en plus une partition de l'ensemble des actions: $\Sigma = \Sigma_p \cup \Sigma_{np}$. Cette partition induit une partition sur l'ensemble des transitions; ainsi on définit $\Delta_p = \{\delta \in \Delta \mid \sigma^\delta \in \Sigma_p\}$.*

Remarque 8 *Le caractère prioritaire d'une transition ne dépend pas des valeurs des paramètres. On remarque également que l'on a uniquement deux niveaux de priorité; cependant on pourrait généraliser le modèle en attribuant à chaque action un niveau de priorité. Enfin, un événement peut être à la fois incontrôlable et prioritaire.*

Dans le cadre ainsi défini, les *moyens d'actions* du contrôle est de renforcer les gardes G^δ des transitions symboliques par des gardes $(G^\delta)' \subseteq G^\delta$ satisfaisant:

pour une transition prioritaire $\delta \in \Delta_p$**:** $\quad G^\delta_{uc} \subseteq (G^\delta)'$

cette condition est identique à la condition (11).

pour une transition non-prioritaire $\delta \in \Delta_{np}$**:**

$$\forall \langle \vec{v}, \vec{p} \rangle \in G^\delta_{uc} \setminus (G^\delta)', \exists \delta' \in \Delta_p, \exists \vec{p'} \in \mathcal{D}_{\sigma^{\delta'}}, \langle \vec{v}, \vec{p'} \rangle \in (G^{\delta'})' \setminus G^{\delta'}_{uc} \tag{22}$$

Cette condition indique que, dans l'état $\vec{v}$, l'événement $\sigma^\delta(\vec{p})$ peut être interdit alors qu'il est incontrôlable (hypothèse $\langle \vec{v}, \vec{p} \rangle \in G^\delta_{uc} \setminus (G^\delta)'$) seulement s'il existe un événement prioritaire et contrôlable $\sigma^{\delta'}(\vec{p'})$ qui est activable(dans l'état $\vec{v}$) dans le système contrôlé.

Cette notion de priorité n'est pas traduisible dans le modèle classique de contrôlabilité dans la mesure où la contrainte sur la garde synthétisée $(G^\delta)'$ dans l'équation (22) dépend d'autre gardes synthétisées $(G^{\delta'})'$. En ce sens, la notion de contrôlabilité est dynamique car elle dépend du système contrôlé, tandis que dans l'équation (11), elle ne dépend que du système initial.

Pour résumer, les événements se classifient ainsi:

1. Parmi les événements contrôlables,
   (a) les prioritaires peuvent inhiber des événements non-prioritaires;
   (b) les non-prioritaires ne le peuvent pas.
2. Parmi les événements incontrôlables,
   (a) les prioritaires ne peuvent pas être inhibés;
   (b) les non-prioritaires peuvent être inhibés par des contrôlables prioritaires

6.2 Définition constructive du système contrôlé maximal

Soit un STS déterministe $\mathcal{M} = (V,\Theta,\Sigma,\Delta,\Sigma_p)$ et une propriété d'interdiction d'état décrite par un prédicat $E(\vec{v})$. On se place immédiatement dans le cas où on cherche à garantir le non-blocage.

On cherche à définir une fonction de contrôle $\mathcal{C} : \Delta \to \biguplus_{\sigma\in\Sigma} \wp(Q \times \mathcal{D}_\sigma)$ qui permet de définir les transitions qui doivent être interdites. Comme précédemment, on va utiliser un opérateur de transformation ensembliste qui permet d'identifier les états menant de manière inévitable à un état vérifiant le prédicat E. Par souci de simplicité, on n'adoptera qu'une version ensembliste. Soit $X \subseteq Q$ un ensemble d'états, $\bar{X}$ son complémentaire, on définit :

$$\mathrm{Pre}^p_{uc}(X) = \bigcup_{\delta\in\Delta_p} Proj_Q\left(G^\delta_{uc} \cap (A^\delta)^{-1}(X)\right) \tag{23}$$

$$\mathrm{Pre}^{np}_{uc}(X) = \bigcup_{\delta\in\Delta_{np}} Proj_Q\left(G^\delta_{uc} \cap (A^\delta)^{-1}(X)\right) \setminus \bigcup_{\delta\in\Delta_p} Proj_Q\left((G^\delta \setminus G^\delta_{uc}) \cap (A^\delta)^{-1}(\bar{X})\right) \tag{24}$$

$$\mathrm{Pre}_{nb}(X) = \bigcup_{\delta\in\Delta} Proj^\forall_Q\left(\overline{G^\delta} \cup (A^\delta)^{-1}(X)\right) \tag{25}$$

Pour les événements incontrôlables *prioritaires*, on se ramène à l'expression classique, équation (23). Pour les événements incontrôlables *non prioritaires*, s'il existe une transition contrôlable et prioritaire qui mène à un état de $\bar{X}$, on poura inhiber cette transition incontrôlable pour que le système respecte toujours la propriété souhaitée et donc on ne considère pas l'état en question comme à interdire, équation (24). Enfin, Pre_{nb} garde sa définition classique, équation (25).

L'opérateur dont on va chercher à calculer le plus petit point-fixe est:

$$\mathrm{Pre}^{nb}_{uc}(X) = \mathrm{Pre}^p_{uc}(X) \cup \mathrm{Pre}^{np}_{uc}(X) \cup \mathrm{Pre}_{nb}(X) \tag{26}$$

Cet opérateur est bien croissant, car l'expression décroissante $\overline{X}$ n'apparaît qu'en second membre d'une différence dans l'équation (24). On définit ensuite $\mathrm{Coreach}_{uc}(E) = \mathrm{lfp}(\lambda X.E \cup \mathrm{Pre}_{uc}(X))$.

Le contrôleur $\mathcal{C}$ est définit comme précédemment: pour toute transition $\delta = \langle\sigma,G,G_{uc},A\rangle$ on a

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

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

- $\Theta' = \Theta \setminus \mathrm{Coreach}_{uc}(E)$;
- Δ' est défini à partir de Δ par $\dfrac{(\sigma,G,G_{uc},A) \in \Delta \quad G' = G \wedge \neg\mathcal{C}(\delta)}{(\sigma,G',G_{uc},A) \in \Delta'}$

Il est facile de voir qu'aucune exécution de $\mathcal{M}'$ ne peut atteindre $\text{Coreach}_{uc}(E)$, et donc E. En effet, les états initiaux de $\mathcal{M}'$ ne sont pas dans $\text{Coreach}_{uc}(E)$, et aucune transition dans Δ' ne peut faire passer l'état courant de $\neg\text{Coreach}_{uc}(E)$ à $\text{Coreach}_{uc}(E)$. On définit ensuite le système contrôlé par $(G^\delta)' = G^\delta \setminus \mathcal{C}(\delta)$.

Vérifions que ce renforcement des gardes correspond bien aux moyens de contrôle définis au §6.1.

- pour une transition prioritaire, il faut vérifier que $G_{uc} \cap \mathcal{C}(\delta) = \emptyset$. Supposons qu'il existe $\langle\vec{v},\vec{p}\rangle \in G_{uc} \cap \mathcal{C}(\delta)$. Alors, par définition de $\mathcal{C}(\delta)$, $\langle\vec{v},\vec{p}\rangle \in G_{uc} \cap (A^\delta)^{-1}(\text{Coreach}_{uc}(E))$, et par définition de Pre^p_{uc}, $\vec{v} \in \text{Coreach}_{uc}(E)$. Mais par définition de $\mathcal{C}(\delta)$, on a aussi $\vec{v} \notin \text{Coreach}_{uc}(E)$. La contradiction nous permet de conclure que $\nexists\langle\vec{v},\vec{p}\rangle \in G_{uc} \cap \mathcal{C}(\delta)$.
- pour une transition non-prioritaire $\delta \in \Delta_{np}$, il faut vérifier la propriété (22). Soit $\langle\vec{v},\vec{p}\rangle \in G^\delta_{uc} \setminus (G^\delta)'$, ce qui implique $\langle\vec{v},\vec{p}\rangle \in G^\delta_{uc} \cap \mathcal{C}(\delta)$. Donc

$$\langle\vec{v},\vec{p}\rangle \in G^\delta_{uc} \wedge A(\vec{v},\vec{p}) \in \text{Coreach}_{uc}(E) \tag{27}$$

Or $\vec{v} \notin \text{Coreach}_{uc}(E)$, en particulier, d'après l'équation (26), on a $\vec{v} \notin \text{Pre}^{np}_{uc}(\text{Coreach}_{uc}(E))$. D'après l'équation (24), on en déduit qu'il existe $\delta' \in \Delta_p$ et $\vec{p'} \in \mathcal{D}_{\sigma^{\delta'}}$ tels que $\langle\vec{v},\vec{p'}\rangle \in (G^{\delta'} \setminus G^{\delta'}_{uc}) \wedge A^{\delta'}(\vec{v},\vec{p'}) \notin \text{Coreach}_{uc}(E)$. Donc $\langle\vec{v},\vec{p}\rangle$ ne sera pas interdit dans la transition δ et $\langle\vec{v},\vec{p}\rangle \in (G^\delta)'$.

◇

6.3 Transformation des STS avec priorité vers un modèle sans priorité

Le calcul, pour un ensemble d'états X, de $\text{Pre}^{nb}_{uc}(X)$ est relativement plus compliqué dans le cas des STS avec priorité entre les événements. On peut se demander à quelle condition on peut se passer de cette notion de priorité.

Le problème qu'on se pose est le suivant: étant donné un STS $\mathcal{M}_1$ tel que les actions sont soit contrôlables, soit incontrôlables(Σ_{uc}), et soit prioritaires(Σ_p), soit non-prioritaires, et un ensemble d'états interdits E, existe-t-il un moyen de le transformer en un STS classique $\mathcal{M}_2$ (ayant le même ensemble d'états) tel que, si on applique l'algorithme défini en §5 sur $\mathcal{M}_2$, on calcule un ensemble d'états $\text{Coreach}_{uc}(E)$ identique à celui calculé par l'algorithme défini en §6 appliqué sur $\mathcal{M}_1$?

Transformation "naturelle". En comparant les définitions des "moyens de contrôle" sur les deux modèles, il semble naturel de considérer la transformation suivante: on transforme un automate avec priorité $\mathcal{M}_1 = (V,\Theta,\Sigma,\Delta^1,\Sigma_{uc},\Sigma_p)$ en un automate sans priorité $\mathcal{M}_2 = (V,\Theta,\Sigma,\Delta^2)$ en suivant les règles :

$$\frac{[\sigma(\vec{p}) : G(\vec{v},\vec{p})?\vec{v'} := A(\vec{v},\vec{p})] \in \Delta^1_{uc} \ \wedge\ G_{uc} = G \setminus \bigcup_{\delta\in\Delta_p} G^\delta}{[\sigma(\vec{p}) : G(\vec{v},\vec{p}),G_{uc}(\vec{v},\vec{p})?\vec{v'} := A(\vec{v},\vec{p})] \in \Delta^2}$$

et

$$\frac{[\sigma(\vec{p}) : G(\vec{v},\vec{p})?\vec{v'} := A(\vec{v},\vec{p})] \notin \Delta^1_{uc} \ \wedge\ G_{uc} = \emptyset}{[\sigma(\vec{p}) : G(\vec{v},\vec{p}),G_{uc}(\vec{v},\vec{p})?\vec{v'} := A(\vec{v},\vec{p})] \in \Delta^2}$$

Exemple 6 *Sur notre exemple (figure 5), la transition* 1*, étiquetée par* $a()$*, est incontrôlable (*$G = G_{uc}$*), tandis que la transition* 2*, étiquetée par* $b()$*, est contrôlable (*$G_{uc} = \emptyset$*) et prioritaire. On transforme le système en supprimant les priorité, et en modifiant les transitions; la première transition aura ses gardes* $G_{c,1} = \{t \leq 5\}$ *et* $G_{uc,1} = \{3 < t \leq 5\}$ *tandis que la seconde aura des gardes* $G_{c,2} = \{t \leq 3\}$ *et* $G_{uc,2} = \emptyset$*.*

Le problème est que cette transformation est *statique*. Or les moyens de contrôle sur un STS avec priorité sont dynamiques (*cf.* §6.1). Cette transformation qui permet de masquer la notion de priorité ne peut être effectuée que si on a des hypothèses très fortes sur le système.

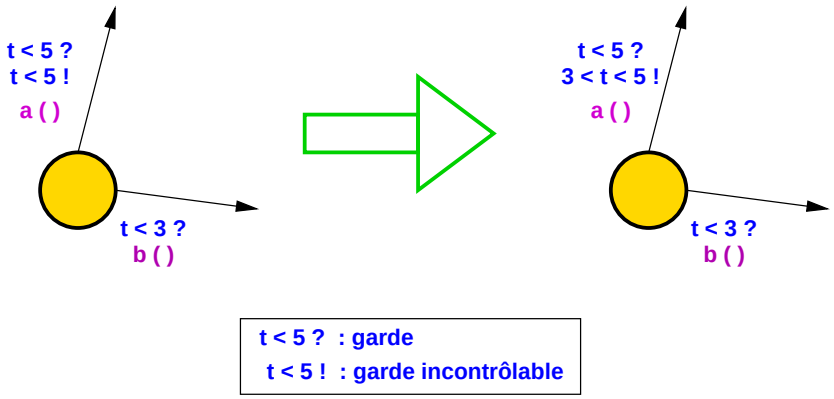

FIG. 5 – *Transformation vers un modèle sans priorités*

Calculs de $\mathrm{Coreach}_{uc}(E)$**.** Sur le premier automate, on peut calculer $E_1^* = \mathrm{lfp}(\lambda X.E \cup \mathrm{Pre}_1(X))$, avec $\mathrm{Pre}_1 = \mathrm{Pre}_{uc}^{nb}(X))$ défini par l'équation (26). Sur le second, on calcule $E_2^* = \mathrm{lfp}(\lambda X.E \cup \mathrm{Pre}_2(X))$, avec $\mathrm{Pre}_2 = \mathrm{Pre}_{uc}^{nb}(X))$ défini par l'équation (19).

Théorème 1 *Si un automate avec priorité* $\mathcal{M}_1 = (V,\Theta,\Sigma,\Delta^1,\Sigma_{uc},\Sigma_p)$ *vérifie les deux conditions:*

- *toutes les transitions prioritaires sont contrôlables*
- *pour tout état, il y a au plus une seule transition non-prioritaire tirable*

et qu'on applique la transformation pour obtenir un automate $\mathcal{M}_2 = (V,\Theta,\Sigma,\Delta^2)$*, alors* $E_1^* = E_2^*$.

Preuve: on va considérer la définition ensembliste du plus petit point fixe, $E_i^* = \bigcup_{n\in\mathbb{N}} \mathrm{Pre}_i^n(E)$ $(i = 1,2)$ et montrer par récurrence que:

$$\forall N \in \mathbb{N}, \bigcup_{n\leq N} \mathrm{Pre}_1^n(E) = \bigcup_{n\leq N} \mathrm{Pre}_2^n(E)$$

Il suffit donc de montrer que pour un ensemble d'états X, on a $\mathrm{Pre}_1(X) = \mathrm{Pre}_2(X)$. Soit $v \in pre_1(X)$. D'après l'équation (26), et en utilisant les deux hypothèses faites, on a que $v \in pre_1(X) \Rightarrow v \in pre_{nb}(X)$ et donc $v \in pre_2(X)$ d'après l'équation (19). Réciproquement si $v \in pre_1(X)$, alors soit $v \in pre_{nb}(X)$, soit $v \in pre_{uc}(X)$. Dans ce dernier cas, celà signifie que $\exists \delta \in \Delta_{uc}, v \in Proj(G^\delta \cap A^{\delta^{-1}}(X)) \bigwedge \forall \delta' \in \Delta_c^p, v \notin Proj(G^\delta \cap A^{\delta^{-1}}(X))$ et donc $\in pre_{uc}^{np}(X)$.

6.4 Différences entre événements forçables et événements prioritaires

Nous avons choisi de présenter un modèle à deux niveaux de priorité. Plutôt que de parler de priorité, on aurait pu aussi utiliser la notion d'événements *forçables* [27], qui permettent aussi de "restreindre le caractère incontrôlable" de certaines transitions.

Prenons un STS classique (défini au §5), dans lequel certaines actions (ensemble $\Sigma_f \subseteq \Sigma$) sont *forçables*; le contrôleur a deux moyens d'action: restreindre les gardes (de manière contrôlable) ou obliger le système à prendre une transition forçable. Formellement les moyens de contrôle sur un tel modèle sont de renforcer les gardes G^δ des transitions par des gardes $(G^\delta)' \subseteq G^\delta$ satisfaisant:

$$\forall \langle \vec{v},\vec{p} \rangle \in G_{uc} \setminus G', \exists \delta' \in \Delta, \exists \vec{p'} \in \mathcal{D}_{\sigma^{\delta'}}, \sigma^{\delta'} \in \Sigma_f \wedge \langle \vec{v},\vec{p'} \rangle \in (G^{\delta'})' \tag{28}$$

Cette définition ressemble (mais n'est pas identique) à celle définie par l'équation (22). La principale différence est que toute garde G_{uc} est susceptible d'être restreinte tandis que dans le modèle avec priorité, seule celles des actions non-prioritaires pouvaient l'être. De plus on ne tient pas compte dans l'équation (28) de la garde $G_{uc}^{\delta'}$.

Remarque 9 *Dans [27], un événement peut être incontrôlable et forçable. Toutefois on peut s'interroger sur la compatibilité de ces deux notions et imposer la condition* $\forall \delta \in \Delta, \sigma^\delta \in \Sigma_f \Rightarrow G_{uc}^\delta = \emptyset$.

Pour illustrer la différence entre les deux notions, on va considérer un système simple.

Exemple 7 *Soit le système avec une variable x dont le domaine est $Q = \{0,1\}$:*

$$\mathcal{M} = (V := \{x\}, \Theta := \{x = 1\}, \Sigma := \{a(), b()\}, \Delta := \{[a() : \text{true}?x' := 0], [b() : \text{true}?x' := 1]\})$$

On veut interdire l'état $\{x = 0\}$, sachant que $G^1_{uc} = Q$. À quelle condition l'état $\{x = 1\}$ est-il bon?

1. *Dans le cas où on a des actions forçables: l'état $\{x = 1\}$ est bon si b est forçable (aucune condition sur G^2_{uc} et sur la nature de l'action a): le contrôleur forcera le système à prendre la seconde transition, aussi on aura toujours $\{x = 1\}$.*
2. *Dans le cas où on a des actions prioritaires: l'état $\{x = 1\}$ est bon si les trois conditions suivantes sont vérifiées:*
   - *l'action a n'est pas prioritaire*
   - *l'action b est prioritaire*
   - *$G^2_{uc} \subseteq \{x = 0\}$*

Ce petit exemple montre bien que la notion de contôlabilité n'est pas la même dans les deux modèles. On ne peut donc pas passer automatiquement d'un système ayant des actions prioritaires ou non à un système ayant des actions forçables.

7 Contrôle des systèmes hybrides

7.1 Moyen de contrôle sur un HTS

Nous avons discuté et défini au §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.

En revanche, un moyen indirect de contrôle est de jouer sur l'interaction entre transitions discrètes et passage du temps. En effet, on peut 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, on dispose 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.

On va donc utiliser l'analogie suivante avec ce qui a été vu en §6:

- Les transition discrètes seront considérées comme prioritaires, qu'elles soient contrôlables ou non;
- l'écoulement du temps sera considéré comme une transition non-prioritaire et incontrôlable.

Il n'y aura donc pas de priorité entre transitions discrètes, comme au §4, mais l'écoulement du temps pourra être inhibé par l'activation d'une transition discrète contrôlable.

Formellement, et par analogie au §6.1, les moyens d'actions du contrôle sont:

- de renforcer les gardes G des transitions discrètes δ par des gardes G' en respectant la contrainte $G_{uc} \subseteq G' \subseteq G$;
- de renforcer l'invariant global H par H' en respectant les contraintes

$$H' \subseteq H \tag{29}$$

$$\forall \vec{q} \in Q, \forall \vec{x} \in X,\ \vec{x} \in H(\vec{q}) \setminus H'(\vec{q}) \implies \exists \delta \in \Delta, (\vec{q},\vec{x}) \in Proj_S(G^{\delta'} \setminus G^{\delta}_{uc}) \tag{30}$$

avec la contrainte globale de ne pas introduire de blocage.

Ceci est en fait la notion de contrôle des systèmes hybrides considérée (plus implicitement) dans [3]. On peut donc considérer que le contrôle des systèmes hybrides est un cas particulier de la notion de contrôle des STS avec priorité. En revanche, comparé à [3], l'ajout dans notre modèle d'événements discrets incontrôlables complique l'interaction entre transitions discrètes et évolutions continues.

7.2 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. On suppose que $\mathcal{M}$ est non bloquant. On veut obtenir un HTS contrôlé $\mathcal{M}' = (V,\Theta,\Sigma,\Delta',D,H')$, lui aussi non bloquant, tel que les exécutions de ce nouvel automate ne passent pas par un état de E. On veut 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 $\mathcal{M}'$, on va déterminer comme aux §4 et §5 l'ensemble $\mathrm{Coreach}_{uc}(E)$ des états qui mènent de 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_{uc}, ainsi que l'interaction entre transitions discrètes et évolutions continues.

Rappels. On rapelle les définitions de $\mathsf{before}(\vec{q},Y)$ et de $B(Y)$ (*cf.* §2.3). Pour $\vec{q} \in Q$ et $Y \subseteq Q \times X$ on définit:

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

$$\mathsf{before}(\vec{q},Y) = \{\langle \vec{q},\vec{x}\rangle \mid \exists T > 0,\ \vec{\nu}_{(\vec{q},\vec{x})}(T) \in Y\} \tag{31}$$

- la fermeture temporelle de Y:

$$B(Y) = \{\langle \vec{q},\vec{x}\rangle \mid \exists \varepsilon > 0, \forall 0 < t < \varepsilon,\ \langle \vec{q},\vec{\xi}_{(\vec{q},\vec{x})}(t)\rangle \in Y\} \tag{32}$$

Opérateur esc**.** Étant donné $\delta \in \Delta$ et $Y \subseteq S$, $\mathsf{esc}(\delta)(Y)$ désigne l'ensemble des états qui peuvent s'échapper par une transition contrôlable vers un état $\langle \vec{q}',\vec{x}'\rangle \in \overline{Y}$ en prenant la transition δ. On a:

$$\mathsf{esc}(\delta)(Y) = Proj_S((G^{\delta} \setminus G^{\delta}_{uc}) \cap (A^{\delta})^{-1}(\overline{Y})) \tag{33}$$

On exclut les états qui ne peuvent s'échapper que par un événement incontrôlable.

Opérateur $\mathrm{Pre}^{\nearrow}_{uc}$**.** Étant donné un état discret $\vec{q}$, on cherche 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 être considérés comme bons car ils mènent à $\overline{Y}$ par la transition discrète δ; il s'agit du sous-ensemble de $A \cap B$ sur la figure 6.

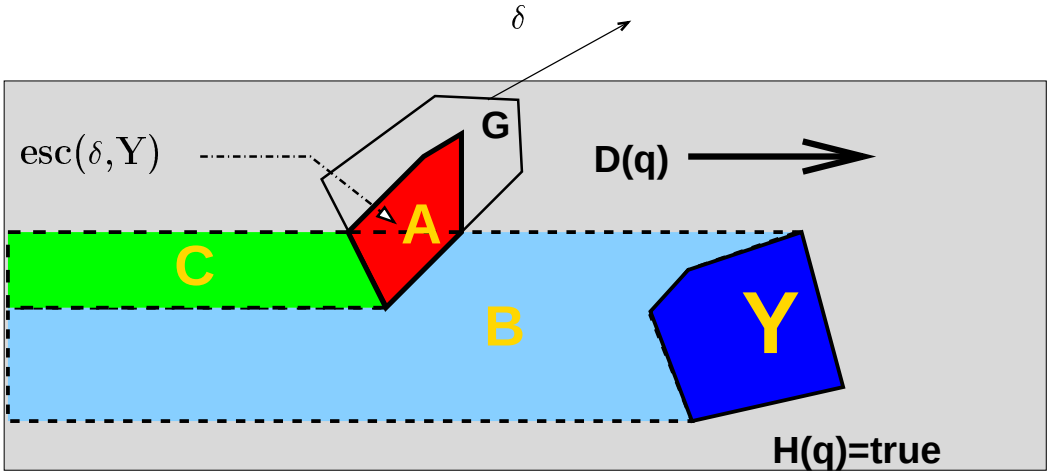

FIG. 6 – $\mathrm{Pre}^{\nearrow}_{uc}(\vec{q},Y)$

– $\mathsf{before}(\vec{q},(\mathsf{before}(\vec{q},Y)\cap \mathsf{esc}(\delta)(Y))$ 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 6.

On en déduit 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 6):

$$\mathrm{Pre}^{\nearrow}_{uc}(\vec{q},Y) = \mathsf{before}(\vec{q},Y) \setminus \bigcup_{\delta\in\Delta} \mathsf{before}(\vec{q},\mathsf{before}(\vec{q},Y)\cap \mathsf{esc}(\delta)(Y)) \quad (34)$$

$$\mathrm{Pre}^{\nearrow}_{uc}(Y) = \bigcup_{\vec{q}\in Q} \mathrm{Pre}^{\nearrow}_{uc}(\vec{q},Y) \quad (35)$$

Par analogie au §6, $pre^{\nearrow}_{uc}(Y)$ correspond au $\mathrm{Pre}^{np}_{uc}(Y)$ de l'équation (24): la seule "transition" non prioritaire est l'écoulement du temps.

Opérateur $\mathrm{Pre}^{\Delta}_{uc}$.

$$\mathrm{Pre}^{\Delta}_{uc}(Y) = \bigcup_{\delta\in\Delta} Proj_Q\left(G^{\delta}_{uc}\cap (A^{\delta})^{-1}(Y)\right) \quad (36)$$

Il s'agit très exactement de l'opérateur Pre^{p}_{uc} de l'équation (23) du §6, car on a ici $\Delta_p=\Delta$.

Opérateur Pre^{nb}_{uc}. Puisque le système contrôlé doit être non bloquant, il nous faut un opérateur qui élimine les états bloquants. Un état est bloquant dans un HTS s'il est bloquant dans le STS sous-jacent (équation (18)) et si l'évolution continue est bloquée.

$$\mathrm{Pre}^{nb}_{uc}(Y) = \bigcap_{\delta\in\Delta} Proj^{\forall}_S(\overline{G^{\delta}}\cup (A^{\delta})^{-1}(Y)) \bigcap B(Y\cup\bar{H}) \quad (37)$$

L'ensemble $B(Y\cup\bar{H})$ représente les états tels que toute évolution temporelle est bloquée, soit par Y, soit parcequ'on sort de l'ensemble H. Cette définition est donc la transposition de l'équation 25 du §6, dans le modèle des HTS.

Opérateur Pre_{uc}. On obtient donc l'opérateur Pre_{uc} global:

$$\mathrm{Pre}_{uc}(Y) = \mathrm{Pre}^{\nearrow}_{uc}(Y)\cup \mathrm{Pre}^{\Delta}_{uc}(Y)\cup \mathrm{Pre}^{nb}_{uc}(Y) \quad (38)$$

où $\mathrm{Pre}^{\Delta}_{uc}$ est l'opérateur défini par l'équation (15).

Système contrôlé maximal $\mathcal{M}'$. L'ensemble des états interdits E à éviter est donc $\mathrm{Coreach}_{uc}(E) = \mathrm{lfp}(\lambda Y.E \cup \mathrm{Pre}_{uc}(Y))$. De même que précédemment, on va définir une fonction de contrôle qui, intuitivement, interdit les transitions menant à $\mathrm{Coreach}_{uc}(E)$.

Lorsqu'on respreint les invariants $H(\vec{q})$, il faut veiller à respecter la contrainte de contrôlabilité exprimée en §7.1. C'est pourquoi on ne "bloquera" l'évolution temporelle que dans les états de l'ensemble $\overline{\mathrm{Coreach}_{uc}(E)} \cap B(\mathrm{Coreach}_{uc}(E))$, car on sait (par le calcul du point fixe) que dans ces états il est possible de prendre une transition discrète menant à $\overline{\mathrm{Coreach}_{uc}(E)}$.

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

- $\Theta' = \Theta \setminus \mathrm{Coreach}_{uc}(E)$;
- Δ' est défini à partir de Δ, en utilisant la fonction $\mathcal{C}(\delta)$ définie par l'équation (17), par

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

- $H' = H \setminus \left(\overline{\mathrm{Coreach}_{uc}(E)} \cap B(\mathrm{Coreach}_{uc}(E))\right)$

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

Remarque 10 *On ne s'est pas préoccupé 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, on considère 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.*

7.3 Calcul effectif du système contrôlé

Comme pour les STS, on ne peut pas calculer de manière exacte $\mathrm{Coreach}_{uc}(E)$. Comme nous avons exprimé cet ensemble comme point-fixe d'opérateurs ensemblistes, on peut travailler dans un treillis abstrait comme au §5.4, en ajoutant les opérations abstraites correspondant aux opérateurs before, esc et $\mathrm{Pre}^{\nearrow}_{uc}$.

Pour déterminer le système contrôlé maximal, nous n'avons pas besoin de faire d'hypothèse sur la nature des gardes ou la forme de l'équation différentielle. Toutefois, pour avoir des approximations efficaces, il est préférable d'avoir des gardes et des invariants sous forme d'union finie de polyèdres convexes. C'est avec ces hypothèses que travaillent les outils de vérification (et de synthèse de contrôleurs) des systèmes hybrides comme Hytech [15] ou d/dt [9].

7.4 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 [26]. Cependant, le modèle des HTS est un 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. Bien que les formalismes diffèrent, les notions de contrôle définies sur chacun d'eux sont compatibles. 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_{uc} dans le cadre des HTS, équation (38), se réduit a la définition de Pre_{uc} dans le cadre des STS, équation (15).

2. cette unification des modèles justifie *a posteriori* les choix faits pour le contrôle des STS, et clarifie la notion de forçage d'événements dans les HTS.

8 Conclusion

Dans ce papier, nous avons abordé 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, 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, la synthèse de contrôleurs pour des systèmes intermédiaires n'a pas été un problème fortement considéré, alors même que de nombreux systèmes peuvent être modélisés par des systèmes de transitions symboliques.

Dans cette optique, nous avons regardé comment appliquer les techniques de synthèses 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 montrer 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.

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 forçage d'événements et surtout le critère de contrôlabilité de celui-ci, 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 de manière unifiée pour ces différents modèles.

Références

[1] P. Aziz Abdulla, A. Bouajjani, and B. Jonsson. On-the-fly analysis of systems with unbounded, lossy fifo channels. In *Computer Aided Verification, CAV'98*, volume 1427 of *LNCS*, July 1998.

[2] R. Alur, C. Courcoubetis, N. Halbwachs, T. Henzinger, P. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science B*, 138:3–34, January 1995.

[3] E. Asarin, O. Bournez, T. Dang, O. Maler, and A. Pnueli. Effective synthesis of switching controllers for linear systems. *Proceedings of the IEEE, Special Issue "Hybrid System: Theory & Application*, 88:1011–1025, 2000.

[4] A. Bouajjani. Languages, rewriting systems, and verification of infinite-state systems. In *Int. Colloqium on Automata, Languages and Programming, ICALP'01*, volume 2076 of *LNCS*, 2001.

[5] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–692, 1986.

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

[7] P. Cousot and R. Cousot. Comparing the Galois connection and widening/narrowing approaches to abstract interpretation. In M. Bruynooghe and M. Wirsing, editors, *PLILP'92*, volume 631 of *LNCS*, Leuven (Belgium), January 1992.

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

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

[10] A. Finkel and J. Leroux. How to compose Presburger-accelerations: Applications to broadcast protocols. In *Foundations of Software Technology and Theoretical Computer Science, FSTTCS'02*, volume 2556 of *LNCS*, pages 145–156, 2002.

[11] G. Goessler and J. Sifakis. Priority systems. In *Proc of FMCO'03*, pages 314–329, Leiden, The Netherlands, November 2003.

[12] N. Halbwachs, Y.E. Proy, and P. Roumanoff. Verification of real-time systems using linear relation analysis. *Formal Methods in System Design*, 11(2), August 1997.

[13] T. Henzinger, P.-H. Ho, and H. Wong-Toï. HyTech: A Model Checker for Hybrid Systems. In *Computer Aided Verification, CAV'97*, number 1254 in LNCS, June 1997.

[14] T. Henzinger, R. Manjumdar, and J-F. Raskin. A classification of symbolic transition systems. Accepted for publication in ACM TOCL, 2002.

[15] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. A user guide to hytech. In *Proceedings of the First International Workshop on Tools and Algorithms for Construction and Analysis of Systems*, pages 41–71. Springer-Verlag, 1995.

[16] B. Jeannet. Representing and approximating transfer functions in abstract interpretation of hetereogeneous datatypes. In *Static Analysis Symposium, SAS'02*, volume 2477 of *LNCS*, Madrid (Spain)), September 2002.

[17] B. Jeannet. Dynamic partitioning in linear relation analysis. application to the verification of reactive systems. *Formal Methods in System Design*, 23(1):5–37, July 2003.

[18] T. Jéron. *Contribution à la génération automatique de tests pour les systèmes réactifs*. Habilitation à diriger les recherches, Université de Rennes 1, March 2004.

[19] The NBac verification tool. `http://www.irisa.fr/prive/bjeannet/nbac/nbac.html`.

[20] A. Pnueli. The temporal logic of programs. In *Proc. of the 18th IEEE Symposium Foundations of Computer Science, FOCS'77*, pages 46–57, 1977.

[21] P. J. Ramadge and W. M. Wonham. Modular feedback logic for discrete event systems. *SIAM J. Control Optim.*, 25(5):1202–1218, September 1987.

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

[23] T. Ushio. On controllable predicates and languages in discrete-event systems. In *Proc. of the* 28^{th} *Conference on Decision and Control*, pages 123–124, Tampa, Floride, Decembre 1989.

[24] M. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Science*, 32(2), April 1986.

[25] Pierre Wolper and Bernard Boigelot. Verifying systems with infinite but regular state spaces. In *Computer Aided Verification, CAV'98*, volume 1427 of *LNCS*, July 1998.

[26] H. Wong-Toi. The synthesis of controllers for linear hybrid automata. In *Proc of the 36th IEEE Conference of Decision and Control*, pages 4607–4612, San Diego, CA, Decembre 1997.

[27] Chen Y.-L. and L. Feng. Safety control of discrete event systems using finate state machines with parameters. *ACC01-IEEE1665*, 2001.

Table des matières

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399