Introduction au contrôle des systèmes temps-réel

Karine ALTISEN[*] — Patricia BOUYER[†] —
Thierry CACHAT[‡] — Franck CASSEZ[♭] — Guillaume GARDEY[♭]

* *VERIMAG – Centre Équation*
2, avenue de Vignates
38610 GIÈRES*, France*

e-mail : `Karine.Altisen@imag.fr`

† *LSV – ENS Cachan & CNRS*
61, avenue du Président Wilson
94235 CACHAN *Cedex, France*

e-mail : `Patricia.Bouyer@lsv.ens-cachan.fr`

‡ *LIAFA – Université Denis Diderot*
2 place Jussieu, Case 7014
75251 PARIS *Cedex 05, France*

e-mail : `Thierry.Cachat@liafa.jussieu.fr`

♭ *IRCCyN*
1, rue de la Noë – BP 92 101
44321 NANTES *Cedex 3, France*

e-mail : `{Franck.Cassez,Guillaume.Gardey}@irccyn.ec-nantes.fr`

Article rédigé dans le cadre du projet CORTOS de l'ACI « Sécurité Informatique ».
Site web : `http://www.lsv.ens-cachan.fr/aci-cortos/`

RÉSUMÉ. *Ce document est une introduction rapide au domaine du contrôle des systèmes temporisés.*

ABSTRACT. *In this paper we give a quick overview of the area of control of real-time systems.*

MOTS-CLÉS : *Contrôle, automates temporisés, jeux temporisés.*

KEYWORDS: *Control, timed automata, timed games.*

1. Introduction

Modélisation et vérification des systèmes temps-réel. La plupart des appareils domestiques (machines à laver, téléphones, téléviseurs, lecteurs CD/DVD) que nous utilisons contiennent désormais de nombreux *logiciels*. Si des erreurs dans ces logiciels ne sont pas dommageables, il n'en va pas de même pour les logiciels qui contrôlent des centrales nucléaires ou des trajectoires (avions, fusées, voitures) : dans ces systèmes, une défaillance logicielle conduit souvent à des dégâts importants ou des pertes en vies humaines. On parle alors de systèmes *critiques*, et il convient donc, lors du développement de telles applications, de s'assurer qu'elles satisfont un certain nombre de *propriétés*, notamment des propriétés de sûreté (comme l'absence de défaillance grave).

Une approche classique de vérification, le *model-checking* [SCH 99], consiste à construire un modèle complet S du comportement du système étudié (par exemple un automate fini), à formaliser la propriété de correction attendue par une formule ψ de logique (temporelle), puis à utiliser un algorithme (de model-checking) pour vérifier que S satisfait (ou non) ψ (ce qui est noté $S \models \psi$). Ces techniques sont maintenant bien connues et utilisées dans le domaine industriel. Cependant, pour certains types d'applications, il est nécessaire de prendre en compte des caractéristiques temporelles autres que le temps *logique* capturé par un modèle du type automate fini. Par exemple, pour des problèmes d'ordonnancement, les durées des tâches doivent être prises en compte explicitement. Il est parfois possible de se ramener à un modèle (en temps) discret mais cela peut s'avérer un facteur limitant : par exemple, si un système a des *constantes* de temps variant de 1 à 10000, considérer une horloge discrète de granularité 1 engendre un nombre d'états prohibitif pour les algorithmes de model-checking ; d'autre part, l'utilisation du temps discret suppose que l'on connaît exactement les durées des diverses opérations à réaliser. Pour prendre en compte une *incertitude* sur ces durées (durée des tâches, des communications), on utilise donc des modèles plus fins que les automates finis, par exemple les *automates temporisés* [ALU 94] où l'on peut utiliser des *horloges* pour spécifier le comportement temporel du système. Les algorithmes de model-checking utilisés sur les structures discrètes (automates finis, logique temporelle) ont été étendus aux modèles temporisés et ont conduit au développement d'outils de vérification dédiés tels que UPPAAL [AMN 01], KRONOS [YOV 97], CMC [LAR 98] ou encore HyTech [HEN 97a].

Le problème de la synthèse de contrôleur. Dans le cas du *model-checking*, on dit que les systèmes considérés sont *fermés* : on travaille sur un modèle complet du système incluant l'environnement à contrôler, les éventuels actionneurs et le contrôleur ; ce système évolue sans influence extérieure. Dans le cadre de la synthèse de contrôleurs, on part d'un système *ouvert*, le but étant de le fermer : si S est un modèle du système ouvert à contrôler, on lui ajoute un contrôleur C, la composition parallèle $(S \parallel C)$ représentant le modèle complet ou *fermé* du système.

Si la propriété de correction à satisfaire est ϕ, les techniques de model-checking permettent de répondre à la question « est-ce que S contrôlé par C ($S \parallel C$) satisfait ϕ ? ». Le problème de model-checking s'écrit alors formellement :

$$\text{Étant donnés } (S \parallel C) \text{ et } \phi, \text{ est-ce que } (S \parallel C) \models \phi\,? \tag{MC}$$

Cette approche nécessite donc d'abord de construire (éventuellement à la main) un contrôleur C, de manière à définir complètement le système, et ensuite à vérifier le système contrôlé $S \parallel C$. Cece peut s'avérer difficile dans le cas de systèmes complexes ou/et avec des propriétés difficiles à mettre en œuvre (comme par exemple des propriétés dépendant du temps). De plus, si la propriété attendue n'est pas vérifiée, on doit modifier le contrôleur et vérifier de nouveau le système de façon itérative jusqu'à obtenir un contrôleur correct[1]. Idéalement, on souhaiterait ne décrire que le système à contrôler et *calculer* (ou *synthétiser*) un contrôleur (s'il en existe un), de manière à ce que la spécification ϕ soit satisfaite. C'est la problématique du *contrôle*, plus générale que celle du model-checking. Le problème du *contrôle* (CP) pour les systèmes ouverts est formellement le suivant :

$$\text{Étant donnés } S \text{ et } \phi, \text{ existe-t-il } C \text{ tel que } (S \parallel C) \models \phi\,? \tag{CP}$$

Pour répondre au problème (CP) il est souvent nécessaire de restreindre la classe de modèles dans laquelle on va chercher un contrôleur. Par exemple on peut chercher des contrôleurs qui sont des automates finis (donc à mémoire bornée), ou bien des automates temporisés. Le problème naturel qui se pose après (CP) est celui de la *synthèse de contrôleur* :

$$\text{Si la réponse à (CP) est « oui », peut-on construire un tel contrôleur } C\,? \tag{CSP}$$

Du contrôle aux jeux. Un problème de contrôle peut être formulé simplement dans le cadre de la *théorie des jeux*, discrets [MCN 93, THO 95, ARN 03, RIE 03] ou temporisés [MAL 95, ASA 98, FAë 02, ALF 03]. On peut modéliser un système ouvert par un automate temporisé *de jeu* comme celui de la Figure 1.

Ce système ouvert comporte des transitions *contrôlables* (traits pleins) et *incontrôlables* (traits pointillés). Le type d'une transition est déterminé par son étiquette : on classe les actions de transitions $\{c_1, c_2, c_3, u\}$ en deux groupes, $\Sigma_c = \{c_1, c_2, c_3\}$ pour les actions contrôlables, et $\Sigma_u = \{u\}$ pour les actions incontrôlables. Ainsi une transition est contrôlable (resp. incontrôlable) si son étiquette est dans Σ_c (resp. Σ_u). Les protagonistes du jeu sont le contrôleur (qui détermine quand une action de Σ_c est faite) et l'environnement (qui peut faire des actions de Σ_u). Dans l'exemple précédent, l'objectif (pour le contrôleur) est d'éviter que le système aille dans l'état Bad, et ce, quels que soient les actions (de Σ_u) de l'environnement. On parle dans ce cas d'un objectif de *sûreté*. Un automate temporisé possède des horloges prenant leur valeur dans $\mathbb{R}_{\geq 0}$ comme x dans l'exemple de la Figure 1. Les trajectoires définies par un tel

1. Ce qui peut être très long si un tel contrôleur n'existe pas . . .

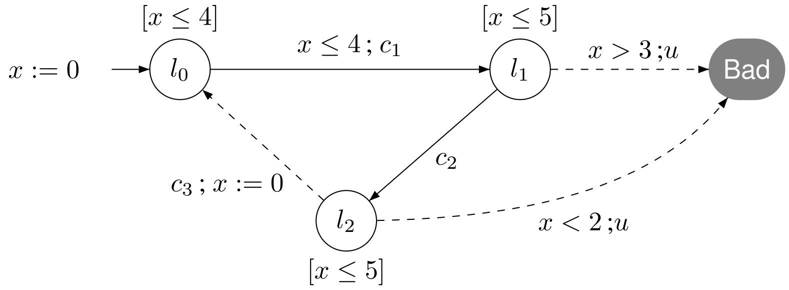

Figure 1. *Un exemple de jeu temporisé – le système ouvert S.*

automate commencent dans l'état initial $(\ell_0, x = 0)$. Le temps peut ensuite s'écouler d'une durée δ ($\in \mathbb{R}_{\geq 0}$) et le système atteint l'état $(\ell_0, x = \delta)$ (les horloges vont à la même vitesse que le temps physique). La *localité* ℓ_0 est contrainte par un invariant $[x \leq 4]$ qui impose que le temps qui s'écoule dans ℓ_0 à partir de $x = 0$ soit inférieur à 4 unités de temps. Enfin, le tir des transitions est lui aussi contraint : c_1 ne peut être tirée que si $x \leq 4$. La spécification de la Figure 1 impose donc qu'on fasse c_1 avant 4 unités de temps. La sémantique d'un automate temporisé est un *système de transitions temporisé*, qui est constitué de deux types d'évolution : *passage du temps* et *action discrète*. Une exécution est une séquence de telles évolutions avec une alternance entre le passage du temps et les actions discrètes. Des exécutions possibles pour l'automate temporisé de la Figure 1 sont par exemple[2] ρ_1 et ρ_2 :

$$\begin{aligned}
\rho_1 :\quad & (\ell_0, 0) \xrightarrow{1.55} (\ell_0, 1.55) \xrightarrow{c_1} (\ell_1, 1.55) \xrightarrow{1.67} (\ell_1, 3.22) \xrightarrow{u} (\textsf{Bad}, 3.22)\\
\rho_2 :\quad & (\ell_0, 0) \xrightarrow{1.1} (\ell_0, 1.1) \xrightarrow{c_1} (\ell_1, 1.1) \xrightarrow{2.1} (\ell_1, 3.2) \xrightarrow{c_2} (\ell_2, 3.2)\\
& \quad \xrightarrow{0.1} (\ell_2, 3.3) \xrightarrow{u} (\ell_0, 0) \cdots
\end{aligned}$$

Dans un système de transitions temporisé il y a tout à la fois une infinité de transitions temporisées (d'étiquette $\delta \in \mathbb{R}_{\geq 0}$) et une infinité d'états. Cependant, il existe une abstraction finie de ces systèmes (*automate des régions* de Alur & Dill [ALU 94]) qui permet de prouver des propriétés de correction. Les états *symboliques* d'une telle abstraction sont des *zones* définies par une localité et des contraintes sur les horloges. Dans l'exemple de la Figure 1, on obtient des états symboliques du type $(\ell_0, 1 \leq x < 3)$ (bien entendu dans le cas où il y plus d'une horloge les zones sont plus compliquées [ALU 94, BOU 04a]).

Pour l'exemple de la Figure 1, si le contrôleur veut éviter l'état Bad, il doit imposer des restrictions sur les dates de tir des transitions contrôlables : par exemple, il ne doit pas autoriser l'action c_1 à partir de ℓ_0 si $x > 3$; il ne doit pas non plus attendre trop longtemps dans ℓ_1 s'il arrive dans cette localité avec $x \leq 3$ car dès que $x > 3$ l'environnement peut tirer une transition incontrôlable amenant le système dans Bad.

2. On note (ℓ, v) un état du système au lieu de $(\ell, x = v)$.

2. Sémantique des jeux temporisés

Lors du déroulement du jeu, il est nécessaire de définir la marche à suivre, et aussi les coups possibles pour chaque joueur. Dans le cas des jeux *discrets*, les coups possibles sont des *actions discrètes*. Chacun des joueurs possède un ensemble d'actions qu'il peut jouer, et qui peuvent varier suivant l'état dans lequel se trouve le système. Dans l'exemple de la Figure 1, c_1 n'est tirable que dans l'état $(\ell_0, x \leq 4)$, u dans $(\ell_1, x > 3), (\ell_2, x \geq 0)$. Dans le cas des jeux temporisés [MAL 95, ASA 98], les joueurs disposent d'une action supplémentaire qui est l'« attente » : un joueur peut décider d'attendre avant de faire une action discrète si celle-ci ne doit pas être jouée trop tôt. Dans les deux cas, la légalité et le résultat d'une action sont déterminés par les *règles du jeu*.

2.1. *Règles du jeu*

Dans la théorie des jeux discrets [THO 95] (automates finis, à pile, *etc.*), les règles du jeu définissent l'espace d'états du système : par exemple un jeu *à tour* à deux joueurs impose que les coups des joueurs alternent ; le prochain état du système est déterminé par un joueur à la fois, celui dont c'est le tour de jouer. Un jeu discret à deux joueurs est dit *concurrent* s'il impose que les deux joueurs choisissent simultanément et indépendamment leurs actions : le prochain état du système est alors la résultante de ces choix simultanés. Dans le cas des jeux temporisés, la notion de jeu à tour n'a pas beaucoup de sens[3] : l'intérêt du temps (quantitatif) dans le modèle est de décrire des joueurs dont les comportements dépendent du temps, et leurs actions sont donc fonctions des délais écoulés. On distingue deux *sémantiques* classiques pour les jeux temporisés :

– les jeux à *observation continue* [MAL 95, ASA 98]. Dans cette sémantique chaque joueur observe continûment l'état du système pour déterminer son action, soit attendre, soit faire une action discrète ; dans l'exemple de la Figure 1, dans la localité ℓ_1, le contrôleur peut attendre tant qu'il veut ou tirer la transition c_2 : son choix se fait en fonction de l'état courant du système dont il a une connaissance exacte à tout moment. Il faut comprendre que si par exemple, le système est encore dans ℓ_1 à la date $x = 4$, il se peut que l'environnement tire u conduisant le système à Bad. Si à cette même date le contrôleur choisit aussi de tirer c_2 le système va de façon non-déterministe dans ℓ_2 ou dans Bad. Par conséquent, si le contrôleur veut à tout pris empêcher le système d'atteindre Bad, le contrôleur doit tirer c_2 avant la date $x = 3$ (inclue). En ce sens, le contrôleur peut anticiper les actions de l'environnement : l'environnement ne peut pas le *surprendre* car s'il peut faire une action incontrôlable (mauvaise) dans δ unités de temps, le contrôleur peut observer tous les états intermédiaires et décider de faire une action discrète avant δ.

3. Sauf dans le cas du contrôle d'un environnement continu par un contrôleur digital comme dans [HEN 99a].

– les jeux avec *surprise* [ALF 03]. Dans cette sémantique, le jeu est *symétrique* : chaque joueur doit proposer indépendamment un coup qui est un couple (durée, action), où l'action peut être une action de contrôle ou l'action « ne rien faire ». Le couple (δ, a) correspond au coup « je fais l'action a dans δ unités de temps » ; une fois les propositions faites, *e.g.* (δ_1, a_1) pour le contrôleur et (δ_2, a_2) pour l'environnement, c'est celle de durée la plus courte qui est sélectionnée. Si $\delta_1 < \delta_2$ le prochain état du jeu est déterminé par le résultat de a_1, et si $\delta_1 = \delta_2$ il y a deux prochains états possibles : celui obtenu après a_1 et celui obtenu après a_2. Contrairement à la sémantique de l'*observation continue*, le contrôleur ne peut plus intervenir une fois qu'il a choisi de laisser passer δ_1 unités de temps.

2.2. *Stratégies*

Dans le cadre de la théorie des jeux, le problème de contrôle (CP) devient : « Existe-t-il une *stratégie* telle que le contrôleur gagne le jeu ? » (la condition de gain est discutée dans la partie 2.3). La notion de stratégie est celle communément utilisée dans les jeux (comme les échecs par exemple). Une stratégie indique le coup à jouer, en fonction de l'historique des coups joués par les joueurs et des états rencontrés depuis le début du jeu. Dans le cas des jeux temporisés sous la sémantique à *observation continue*, une stratégie est une fonction partielle[4] de l'historique des coups (incluant les temps écoulés) dans l'ensemble $\{\lambda\} \cup \Sigma_c$ où λ est l'action spéciale « ne rien faire ». Dans le cas de la Figure 1, on peut définir partiellement la stratégie f par exemple par : $f(\rho.\ell_0, x < 2) = \lambda$, $f(\rho.\ell_0, x = 2) = c_1$, où la notation $\rho.\ell_i$ signifie « pour tous les historiques se terminant en ℓ_i ». Cette stratégie indique d'attendre à partir de tout historique se terminant dans l'état $(\ell_0, x < 2)$, et de faire l'action c_1 quand l'historique se termine par $(\ell_0, x = 2)$. Il est donc impossible d'obtenir des trajectoires se terminant en $(\ell_0, x > 2)$ car pour cela, il faudrait que la stratégie autorise l'action λ d'attente pour un historique se terminant en $(\ell_0, x = 2)$. Une telle stratégie f qui ne dépend pas de toute l'histoire mais seulement du dernier état du système est appelée *stratégie positionnelle* ou encore *sans mémoire*. Bien entendu, une stratégie doit prescrire un coup valide. Ainsi la fonction $f'(\rho.\ell_0, x > 4) = c_1$ n'est pas une stratégie car elle prescrit un coup qui n'est pas dans le jeu donné par l'exemple de la Figure 1. Dans le cas des jeux *avec surprise*, une stratégie est une fonction de l'historique dans l'ensemble des couples de $\mathbb{R}_{\geq 0} \times (\{\lambda\} \cup \Sigma_c)$. Si dans un jeu *avec surprise*, on impose que les coups soient de la forme $(0, c)$ avec $c \in \Sigma_c$ ou (δ, λ) avec $\delta \in \mathbb{R}_{\geq 0}$, on obtient des stratégies « sans surprise » où l'état du système est observable après tout délai écoulé et avant toute action discrète.

---

4. Il se peut en effet qu'un historique se termine par un état où aucun coup du contrôleur n'est possible ; dans ce cas c'est l'environnement seul qui peut jouer et on ne peut pas définir de stratégie pour le contrôleur.

2.3. *Objectifs de contrôle*

Dans la section 1, le problème de contrôle (CP) a comme paramètre la propriété souhaitée ϕ. Les propriétés les plus simples sont les propriétés (définies sur les états) de *sûreté* et d'*atteignabilité*. Dans le premier cas, ϕ désigne un ensemble d'états sûrs du système, et le problème de contrôle consiste alors à trouver une stratégie de manière à maintenir le jeu dans les états de ϕ. Dans le second cas, ϕ correspond à un ensemble d'états dans lequel on veut amener le système : le problème de contrôle consiste à trouver une stratégie de manière à forcer le jeu à aller dans un état de ϕ. De façon plus générale, on peut définir des objectifs de contrôle qui sont des formules de logiques temporelles (par ex. LTL) [PNU 89]. On peut ainsi spécifier des objectifs comme « amener infiniment souvent le jeu dans ϕ_1 sans jamais passer par ϕ_2 ». De tels objectifs sont dits ω-réguliers [THO 95].

2.4. *Propriétés des stratégies*

On définit maintenant de manière plus formelle ([MAL 95, ASA 98]) les problèmes de contrôle donnés sous la forme de jeux temporisés ayant une sémantique à *observation continue* et pour des objectifs de sûreté. Supposons que le jeu temporisé soit G (*e.g.* l'automate temporisé de la Figure 1), et l'ensemble des états atteignables dans G (sans aucun contrôle) est *Reach*(G). Soit ϕ l'ensemble des états sûrs. Une stratégie, pour gagner, va devoir restreindre l'ensemble des états atteignables du système pour que seuls des états sûrs soit atteignables. Remarquons tout d'abord qu'une stratégie, même si elle est déterministe, n'engendre pas une unique exécution. En effet, une stratégie ne donne que des restrictions sur les actions contrôlables (et ne restreint donc pas les actions incontrôlables). Par exemple la stratégie $f(\rho.\ell_2, x < 1) = \lambda$, et $f(\rho.\ell_2, x = 1) = c_3$ engendre les exécutions $(\ell_2, x = \nu < 1) \xrightarrow{\delta} (\ell_2, x = \nu + \delta < 1)$ et aussi $(\ell_2, x = \nu < 1) \xrightarrow{u} (\mathsf{Bad}, x = \nu)$. De même à partir de l'état $(\ell_2, x = 1)$, il n'y a pas de priorité à l'action contrôlable c_2, et les deux exécutions $(\ell_2, x = 1) \xrightarrow{c_2} (\ell_0, x = 1)$ et $(\ell_2, x = 1) \xrightarrow{u} (\mathsf{Bad}, x = 1)$ sont possibles. Le jeu G contrôlé avec la stratégie f, noté $f(G)$, produit comme résultat un ensemble d'exécutions qui forme un sous-ensemble de l'ensemble des exécutions du système ouvert G. En terme d'états atteignables, on a *Reach*$(f(G)) \subseteq$ *Reach*(G). Le problème de contrôle de sûreté devient dans ce cas : « Existe-t-il une stratégie f, telle que *Reach*$(f(G)) \subseteq \phi$? ». Une telle stratégie est alors dite *gagnante*. La stratégie triviale f_λ qui consiste pour le contrôleur à ne rien faire (toujours dire λ) peut être une stratégie gagnante : il suffit de vérifier par model-checking que *Reach*$(f_\lambda(G)) \subseteq \phi$. Mais cette solution n'est pas satisfaisante car le jeu risque de se bloquer du fait de l'immobilisme du contrôleur. On souhaite évidemment éviter cela et que le contrôleur soit non-bloquant. On cherche donc des stratégies qui ont cette propriété de non-blocage [MAL 95, ALF 02, D'S 02]. De la même manière, il se peut qu'une stratégie soit gagnante parce qu'elle induit des comportements dits *Zenon*, c'est-à-dire des comportements dans lesquels un nombre infini d'actions discrètes sont effectués en un temps fini. Sur l'exemple de la Figure 2, si l'on veut éviter Bad et donc rester dans $\phi = \{(\ell_0, x, y \geq 0)\}$, on peut définir la stratégie suivante : soit ρ_n l'historique $(\ell_0, x = 0) \xrightarrow{(\delta_0, c)} \cdots \xrightarrow{(\delta_n, c)} (\ell_0, x = \sum_{i=0}^{n} \delta_i)$, où la transition $\xrightarrow{(\delta_i, c)}$ consiste à attendre δ_i unités de temps et ensuite à faire c. On définit la stratégie $f(\rho_n) = (\frac{1}{2} \cdot (1 - \sum_{i=0}^{n} \delta_i), c)$.

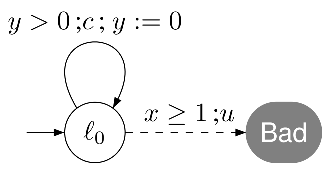

Figure 2. *Contrôle Zénon*

Cette stratégie est bien non bloquante si on part de $0 < \delta_0 < 1$. Elle produit une seule exécution limite (infinie) qui ne passe jamais par Bad. Néanmoins elle empêche le temps de dépasser 1 unité de temps, produisant ainsi une exécution *Zénon*. En pratique, cela impose que le contrôleur réagisse de plus en plus vite et fasse des actions c séparées par une durée de plus en plus courte, tendant vers zéro. Ceci n'est pas réaliste et des études ont été menées pour construire des stratégies *non-Zenon* [CAS 02, ALF 03].

2.5. *États gagnants*

Le problème de contrôle se réduit au problème du calcul des états *gagnants* du jeu. Ces états sont définis sémantiquement comme étant ceux à partir desquels il existe une stratégie gagnante. On note $\mathcal{W}$ les états gagnants du jeu. Si on sait calculer l'ensemble de ces états, on peut déterminer facilement s'il existe une stratégie permettant de gagner le jeu : il suffit de déterminer si l'état initial fait partie de $\mathcal{W}$. Une bonne propriété des jeux de sûreté à *observation continue* est qu'ils sont *déterminés*, ce qui signifie que tout état s du système est soit gagnant pour le contrôleur, soit gagnant pour l'environnement. De plus, dans le cas où l'état initial est gagnant, il existe une stratégie globale f, sans mémoire, gagnante (*Cf.* Théorème 2 dans la partie 3.3). Cette propriété de jeu déterminé, permet par exemple de réduire le calcul des états gagnants pour un jeu d'atteignabilité « le contrôleur doit forcer ϕ » à un jeu *inversé* où on calcule les états gagnants du jeu de sûreté « l'environnement doit maintenir le système dans $\neg\phi$ ».

Dans le cas général des jeux *avec surprise*, on n'a plus ces propriétés car les jeux *avec surprise* ne sont pas déterminés. D'autre part, les jeux avec surprise sont plus compliqués que les jeux à observation continue : par exemple, il existe des jeux temporisés *avec surprise* pour lesquels il faut des stratégies à mémoire non bornée pour gagner [ALF 03], alors que les stratégies sans mémoire suffisent pour gagner les jeux *à observation continue* (Théorème 2).

3. Algorithmes pour la synthèse de contrôleur

Les algorithmes pour la synthèse de contrôleur pour les automates temporisés de jeu sont donnés dans [MAL 95, ASA 98, ALF 01, ALF 03]. On donne ici l'exemple d'un algorithme permettant de calculer les états gagnants d'un jeu de sûreté dans le cadre défini dans [MAL 95]. Le résultat de cet algorithme sur l'exemple de la Figure 1 est donné à la fin de la section.

3.1. *Prédécesseurs contrôlables*

Soit *CPre*$(X) = \{s \,|\, \exists c \in \Sigma_c \,|\, s \xrightarrow{c} s' \wedge s' \in X\}$ et *UPre*$(X) = \{s \,|\, \exists u \in \Sigma_u \,|\, s \xrightarrow{u} s' \wedge s' \in X\}$. Ces deux ensembles correspondent aux états à partir desquels il existe une transition contrôlable (resp. incontrôlable) dont le but est dans X.

La Figure 3 décrit les conditions pour qu'un état s soit un *prédécesseur contrôlable* d'un ensemble d'états X : il faut que le contrôleur puisse *forcer* le système à aller dans un état de X, en choisissant de laisser passer du temps (δ) puis de faire une action contrôlable c, et ce sans que l'environnent puisse l'amener à l'extérieur de X (noté $\overline{X}$) à partir d'un état intermédiaire s_t rencontré en allant de s à s'.

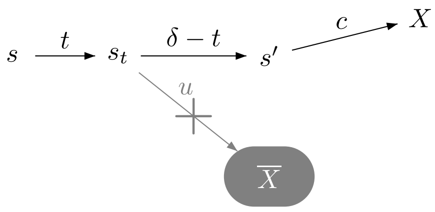

Figure 3. *Prédécesseurs contrôlables*

Un état s est donc un prédecesseur contrôlable de X ssi :

1) $\exists \delta \geq 0, s \xrightarrow{\delta} s' \wedge s' \in$ *CPre*(X)

2) pour tout $0 \leq t \leq \delta$ tel que $s \xrightarrow{t} s_t$, on a $s_t \notin$ *UPre*$(\overline{X})$

L'ensemble des prédécesseurs contrôlables de X est noté $\pi(X)$.

Dans l'exemple de la Figure 1, l'état $(\ell_1, x = 1)$ est un prédécesseur contrôlable de $(\ell_2, x = 2)$: on peut laisser passer du temps de $(\ell_1, x = 1)$ jusqu'à $(\ell_1, x = 2)$ puis tirer c_2 de ℓ_1 à ℓ_2. Lors du passage du temps aucune transition incontrôlable n'est tirable.

3.2. *Opérateurs symboliques*

Comme nous l'avons indiqué précédemment, la sémantique d'un automate temporisé est un système de transitions temporisé contenant une infinité de transitions et d'états. Pour analyser de tels automates, il est nécessaire de regrouper les états en *états symboliques* qui sont des (unions finies de) couples (ℓ, Z) où ℓ est une localité et Z une contrainte convexe sur les horloges. Les opérateurs *CPre*, *UPre* et π ont deux propriétés importantes. Si X est une union de zones, alors :

$\mathbf{P_1}$: *CPre*(X), *UPre*(X) et $\pi(X)$ sont des unions de zones,

$\mathbf{P_2}$: *CPre*(X), *UPre*(X) et $\pi(X)$ sont effectivement calculables.

Calcul des états gagnants W^*

X_i	ℓ_0	ℓ_1	ℓ_2
0	$0 \leq x$	$0 \leq x$	$0 \leq x$
1	$0 \leq x \leq 4$	$0 \leq x \leq 3$	$2 \leq x \leq 5$
2	$0 \leq x \leq 3$	–	–

Stratégie la plus permissive f^*

	ℓ_0	ℓ_1	ℓ_2
λ	$x < 3$	$x < 3$	$x < 5$
c	$x \leq 3$	$2 \leq x$	$x \leq 5$

Tableau 1. *Calculs symboliques pour l'exemple de la Figure 1.*

Pour l'automate temporisé de la Figure1, on a *CPre*$(\ell_1, x \leq 3) = (\ell_0, x \leq 3)$ et si $Z = (\ell_0, x \ \leq 4) \cup (\ell_1, x \ \geq 0) \cup (\ell_2, x \geq 0)$ alors $\pi(Z) = Z'$ avec $Z' = (\ell_0, x \leq 3) \cup (\ell_1, 2 \leq x \leq 3) \cup (\ell_2, x \geq 2)$.

3.3. *Calcul symbolique des états gagnants*

On peut montrer [ASA 98] que dans le cas d'un automate temporisé de jeu, et d'un objectif de sûreté ϕ convexe, l'ensemble des états gagnants $\mathcal{W}$, défini dans la partie 2.5, est le plus grand point fixe de la fonction $h(X) = \phi \cap \pi(X)$. Il faut bien noter que $\mathcal{W}$ est l'ensemble des états gagnants, donc maximal ; donc un état s est gagnant ssi $s \in \mathcal{W}$. Si ϕ est défini comme une union de zones, alors le calcul itératif défini par $X_0 = \phi$, $X_{i+1} = h(X_i) = \phi \cap \pi(X_i)$ est tel que tous les X_i sont des unions de zones[5], et s'arrête en un nombre fini d'itérations. Lorsque $X_{i_0+1} = X_{i_0}$, le plus grand point fixe W^* de h est atteint et $W^* = X_{i_0}$. Comme on peut décider si l'état initial d'un automate appartient à un état symbolique, on obtient donc le thèorème suivant :

Théorème 1 ([ASA 98, HEN 99b]) *Les problèmes de contrôle (CP) de sûreté et d'atteignabilité sont décidables pour les automates temporisés et EXPTIME-complet.*

Dans le cas de notre exemple de la Figure 1, on obtient itérativement les ensembles[6] du Tableau 1. Pour les propriétés plus compliquées que des propriétés de sûreté, le calcul des états gagnants est plus complexe [MAL 95, ASA 98, ALF 01, ALF 03].

3.4. *Synthèse de stratégies gagnantes*

Pour ce qui est du problème de synthèse de stratégie, dans le cas d'un objectif de sûreté, on peut définir la stratégie *maximale* ou *la plus permissive*. En toute rigueur, cette stratégie la plus permissive n'est pas une stratégie au sens de la partie 2.2 car il faudrait qu'elle associe une action de $\Sigma_c \cup \{\lambda\}$ à une exécution. La stratégie la

5. L'intersection de zones est une zone.
6. Le domaine gagnant est donné pour chaque localité ℓ_i dans la colonne correspondante.

plus permissive f^* associe à chaque exécution ρ un sous-ensemble non vide de $\Sigma_c \cup \{\lambda\}$. Toute (sous-) stratégie (au sens de la partie 2.2) f (non bloquante) telle que $f(\rho) \in f^*(\rho)$ est une stratégie gagnante, et f^* est maximale pour cette propriété (c'est pourquoi on la qualifie de stratégie la plus permissive). Concernant la synthèse de stratégies on a le théorème suivant :

Théorème 2 ([ASA 98]) *Si G est un automate temporisé tel que l'état initial de G est gagnant pour l'objectif de sûreté ϕ, alors il existe une stratégie (maximale) f^* gagnante et sans mémoire.*

Pour calculer cette stratégie on commence par renforcer les gardes des transitions de telle manière qu'en partant d'un état gagnant s, si on franchit une transition contrôlable dans l'automate de jeu avec les gardes renforcées, alors on atteint un état gagnant. Les renforcements des gardes sont donc les plus grandes *préconditions* permettant d'atteindre des états gagnants. Dans notre cas on doit renforcer la garde de c_1 en ajoutant la contrainte $x \leq 3$, celle de c_2 en ajoutant la contrainte $x \geq 2$ (qui indique qu'on ne doit pas tirer trop tôt c_2). La stratégie la plus permissive est ensuite donnée par : attendre pour tous les états gagnants s tels qu'il existe $\delta > 0$ et $s \xrightarrow{\delta} s'$ et s' est gagnant ; faire une transition contrôlable c_i si la garde renforcée le permet. Dans le cas de l'exemple de la Figure 1 la stratégie la plus permissive est donnée dans le tableau 1. Il faut bien noter que pour obtenir une stratégie *non-Zénon*, il ne faut pas choisir indéfiniment λ dans ℓ_0 (ou ℓ_1, ℓ_2), mais bien tirer c_1 à un certain point. De cette manière, on construit un automate temporisé C qui représente la stratégie la plus permissive f^*, et tel que $G \parallel C$ satisfait la propriété de sûreté ϕ. Dans le cas de jeux d'atteignabilité, le calcul d'une stratégie gagnante peut nécessiter plus de travail [BOU 04b].

4. Pour aller plus loin

Les problèmes d'implementation. La synthèse de contrôleur visant à répondre à un besoin réel, il est naturel de s'interroger sur les problèmes soulevés par la réalisation effective du contrôleur. La démarche présentée précédemment suppose un cadre théorique parfait : synchronisation parfaite, communications instantanées, temps dense, horloges infiniment précises et réactions infiniment rapides. Cependant, un système réel est caractérisé par des retards, des horloges digitales, des vitesses de transmission non-nulles et un delai minimum entre deux réactions du système. Par conséquent, la construction d'un contrôleur implémentable à partir du contrôleur idéal est un problème dépendant de plusieurs paramètres. Il est donc nécessaire de mettre au point une démarche pour s'assurer que le contrôleur obtenu soit effectivement *implémentable* selon des critères prédéfinis. Nous référons le lecteur à l'article « Implémentabilité des automates temporisés » dans ce même volume pour un aperçu plus détaillé des problèmes d'implémentation.

Observation partielle. Nous avons ici supposé que le système à contrôler était totalement observable : à tout moment le contrôleur peut connaître l'état du système, les valeurs des différentes horloges et les actions. Dans un système plus réaliste, le contrôleur accède aux variables du système par des capteurs et le système peut posséder ses propres variables, non communiquées au contrôleur. Largement étudié pour les systèmes à événements discrets, le problème du contrôle sous observation partielle se pose également dans un cadre temporisé. Différents niveaux d'observation partielle ont été introduits allant de la non-observabilité d'actions, à la non-observabilité d'horloges ou d'états. Une synthèse de ces problèmes peut être trouvée dans l'article « Observation partielle des systèmes temporisés » dans ce même volume.

Algorithmes, modèles et propriétés. Concernant les algorithmes de synthèse de contrôleurs, des travaux récents ont proposé des implémentations efficaces de l'algorithme à point fixe exposé dans cet article [ALT 99, ALT 02, CAS 05]. Les algorithmes de synthèse de contrôleurs ont été étendus au cas des systèmes *hybrides* [HEN 99a, HEN 97b, WON 97], qui sont une généralisation des automates temporisés où les horloges peuvent évoluer à différentes vitesses. Finalement, dans le cas de jeux d'atteignabilité pour des automates temporisés se pose le problème du temps optimal pour atteindre un état gagnant. Ce problème a été résolu dans [ASA 99]. Des critères plus généraux d'« optimalité » sont considérés dans [ALU 04, BOU 04c, BRI 05]. Pour considérer des propriétés plus compliquées, il est possible d'exprimer les objectifs de contrôle ω-réguliers [MAL 95, ALF 01] ou encore à l'aide de logiques modales temporisés [BOU 05].

5. Bibliographie

[ALF 01] DE ALFARO L., HENZINGER T. A., MAJUMDAR R., « Symbolic Algorithms for Infinite-State Games », *Proc. 12th International Conference on Concurrency Theory (CONCUR'01)*, vol. 2154 de *Lecture Notes in Computer Science*, Springer, 2001, p. 536–550.

[ALF 02] DE ALFARO L., HENZINGER T. A., STOELINGA M., « Timed Interfaces », *Proc. 2nd International Workshop on Embedded Software (EMSOFT'02)*, vol. 2491 de *Lecture Notes in Computer Science*, Springer, 2002, p. 108–122.

[ALF 03] DE ALFARO L. D., FAËLLA M., HENZINGER T. A., MAJUMDAR R., STOELINGA M., « The Element of Surprise in Timed Games », *Proc. 14th International Conference on Concurrency Theory (CONCUR'2003)*, vol. 2761 de *Lecture Notes in Computer Science*, Springer, 2003, p. 142–156.

[ALT 99] ALTISEN K., TRIPAKIS S., « On-the-Fly Controller Synthesis for Discrete and Dense-Time Systems », *World Congress on Formal Methods (FM'99)*, vol. 1708 de *Lecture Notes in Computer Science*, Springer, 1999, p. 233–252.

[ALT 02] ALTISEN K., TRIPAKIS S., « Tools for Controller Synthesis of Timed Systems », *Proc. 2nd Workshop on Real-Time Tools (RT-TOOLS'02)*, 2002, Proc. published as Technical Report 2002-025, Uppsala University, Sweden.

[ALU 94] ALUR R., DILL D., « A Theory of Timed Automata », *Theoretical Computer Science (TCS)*, vol. 126, n° 2, 1994, p. 183–235, Elsevier Science.

[ALU 04] ALUR R., BERNADSKY M., MADHUSUDAN P., « Optimal Reachability in Weighted Timed Games », *Proc. 31st International Colloquium on Automata, Languages and Programming (ICALP'04)*, vol. 3142 de *Lecture Notes in Computer Science*, Springer, 2004, p. 122–133.

[AMN 01] AMNELL T., BEHRMANN G., BENGTSSON J., D'ARGENIO P. R., DAVID A., FEHNKER A., HUNE T., JEANNET B., LARSEN K. G., MÖLLER O., PETTERSSON P., WEISE C., YI W., « UPPAAL – Now, Next, and Future », *Proc. Modelling and Verification of Parallel Processes (*MOVEP2k*)*, vol. 2067 de *Lecture Notes in Computer Science*, Springer, 2001, p. 99–124.

[ARN 03] ARNOLD A., VINCENT A., WALUKIEWICZ I., « Games for Synthesis of Controllers with Partial Observation », *Theoretical Computer Science*, vol. 1, n° 303, 2003, p. 7–34.

[ASA 98] ASARIN E., MALER O., PNUELI A., SIFAKIS J., « Controller Synthesis for Timed Automata », *Proc. IFAC Symposium on System Structure and Control*, Elsevier Science, 1998, p. 469–474.

[ASA 99] ASARIN E., MALER O., « As Soon as Possible : Time Optimal Control for Timed Automata », *Proc. 2nd International Workshop on Hybrid Systems : Computation and Control (HSCC'99)*, vol. 1569 de *Lecture Notes in Computer Science*, Springer, 1999, p. 19–30.

[BOU 04a] BOUYER P., « Forward Analysis of Updatable Timed Automata », *Formal Methods in System Design*, vol. 24, n° 3, 2004, p. 281–320, Elsevier Science.

[BOU 04b] BOUYER P., CASSEZ F., FLEURY E., LARSEN K. G., « Optimal Strategies in Priced Timed Game Automata », BRICS Reports Series n° RS-04-0, 2004, BRICS, Denmark, Aalborg, Denmark.

[BOU 04c] BOUYER P., CASSEZ F., FLEURY E., LARSEN K. G., « Optimal Strategies in Priced Timed Game Automata », *Proc. 24th Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS'2004)*, vol. 3328 de *Lecture Notes in Computer Science*, Springer, 2004, p. 148–160.

[BOU 05] BOUYER P., CASSEZ F., LAROUSSINIE F., « Modal Logics for Timed Control », *Proc. 16th International Conference on Concurrency Theory (CONCUR'2005)*, Lecture Notes in Computer Science, Springer, 2005, À paraître.

[BRI 05] BRIHAYE T., BRUYÈRE V., RASKIN J.-F., « On Optimal Timed Strategies », Technical Report n° 2005.48, 2005, Centre Fédéré en Vérification, Belgique.

[CAS 02] CASSEZ F., HENZINGER T. A., RASKIN J.-F., « A Comparison of Control Problems for Timed and Hybrid Systems », *Proc. 5th International Workshop on Hybrid Systems : Computation and Control (HSCC'02)*, vol. 2289 de *LNCS*, Springer, 2002, p. 134–148.

[CAS 05] CASSEZ F., DAVID A., FLEURY E., LARSEN K. G., LIME D., « Efficient On-The-Fly Algorithms for the Analysis of Timed Games », *Proc. 16th International Conference on Concurrency Theory (CONCUR'2005)*, Lecture Notes in Computer Science, Springer, 2005, À paraître.

[D'S 02] D'SOUZA D., MADHUSUDAN P., « Timed Control Synthesis for External Specifications », *Proc. 19th International Symposium on Theoretical Aspects of Computer Science (STACS'02)*, vol. 2285 de *Lecture Notes in Computer Science*, Springer, 2002, p. 571–582.

[FAë 02] FAËLLA M., LA TORRE S., MURANO A., « Dense Real-Time Games », *Proc. 17th IEEE Symposium on Logic in Computer Science (LICS'02)*, IEEE Computer Society Press, 2002, p. 167–176.

[HEN 97a] HENZINGER T. A., HO P.-H., WONG-TOI H., « HYTECH : A Model-Checker for Hybrid Systems », *Journal on Software Tools for Technology Transfer (STTT)*, vol. 1, n° 1–2, 1997, p. 110–122, Springer.

[HEN 97b] HENZINGER T. A., KOPKE P. W., « Discrete-Time Control for Rectangular Hybrid Automata », *Proc. 24th International Colloquium on Automata, Languages, and Programming (ICALP'97)*, vol. 1256 de *Lecture Notes in Computer Science*, Springer, 1997, p. 582–593.

[HEN 99a] HENZINGER T. A., HOROWITZ B., MAJUMDAR R., « Rectangular Hybrid Games », *Proc. 10th International Conference on Concurrency Theory (CONCUR'99)*, vol. 1664 de *Lecture Notes in Computer Science*, Springer, 1999, p. 320–335.

[HEN 99b] HENZINGER T. A., KOPKE P. W., « Discrete-Time Control for Rectangular Hybrid Automata », *Theoretical Computer Science*, vol. 221, 1999, p. 369–392.

[LAR 98] LAROUSSINIE F., LARSEN K. G., « CMC : A Tool for Compositional Model-Checking of Real-Time Systems », *Proc. IFIP Joint International Conference on Formal Description Techniques & Protocol Specification, Testing, and Verification (FORTE-PSTV'98)*, Kluwer Academic, 1998, p. 439–456.

[MAL 95] MALER O., PNUELI A., SIFAKIS J., « On the Synthesis of Discrete Controllers for Timed Systems », *Proc. 12th Annual Symposium on Theoretical Aspects of Computer Science (STACS'95)*, vol. 900 de *Lecture Notes in Computer Science*, Springer, 1995, p. 229–242.

[MCN 93] MCNAUGHTON R., « Infinite Games Played on Finite Graphs », *Annals of Pure and Applied Logic*, vol. 65, n° 2, 1993, p. 149–184, Elsevier Science.

[PNU 89] PNUELI A., ROSNER R., « On the Synthesis of a Reactive Module », *Proc. 16th ACM Symposium on Principles of Programming Languages (POPL'89)*, ACM, 1989, p. 179–190.

[RIE 03] RIEDWEG S., PINCHINAT S., « Quantified Mu-Calculus for Control Synthesis », *Proc. 28th International Symposium on Mathematical Foundations of Computer Science (MFCS'03)*, vol. 2747 de *Lecture Notes in Computer Science*, Springer, 2003, p. 642–651.

[SCH 99] SCHNOEBELEN P., BÉRARD B., BIDOIT M., LAROUSSINIE F., PETIT A., *Vérification de logiciels : Techniques et outils de model-checking*, Vuibert, 1999.

[THO 95] THOMAS W., « On the Synthesis of Strategies in Infinite Games », *Proc. 12th Annual Symposium on Theoretical Aspects of Computer Science (STACS'95)*, vol. 900, Springer, 1995, p. 1–13, Invited talk.

[WON 97] WONG-TOI H., « The Synthesis of Controllers for Linear Hybrid Automata », *Proc. 36th IEEE Conference on Decision and Control*, IEEE Computer Society Press, 1997, p. 4607–4612.

[YOV 97] YOVINE S., « KRONOS : A Verification Tool for Real-Time Systems », *Journal of Software Tools for Technology Transfer (STTT)*, vol. 1, n° 1–2, 1997, p. 123–133, Springer.