Synthèse de contrôleurs et stratégies gagnantes dans les jeux de parité

Aymeric Vincent

Laboratoire Bordelais de Recherche en Informatique
Université Bordeaux I
351, cours de la Libération
F-33405 Talence Cedex
Aymeric.Vincent@labri.u-bordeaux.fr

RÉSUMÉ. *Alors que dans le cadre proposé par Ramadge et Wonham, les auteurs s'intéressent au contrôle de systèmes à événements discrets dont les objectifs portent sur leurs comportements séquentiels, nous proposons ici une méthode générale permettant de synthétiser un contrôleur dont les objectifs sont donnés par des formules du μ-calcul modal, ce qui permet de contraindre aussi les comportements arborescents. Cette méthode repose sur le calcul d'une stratégie gagnante dans un jeu de parité associé au système et à l'objectif de contrôle.*

ABSTRACT. *In the framework of Ramadge and Wonham, the authors focus on the control of discrete event systems whose objectives are specified on their sequential behaviour. We offer here a generic method which allows us to specify these objectives in modal μ-calculus, which can also constrain the computational tree of those behaviours. This method relies on the computation of a winning strategy in a parity game based on the system and the control objective.*

MOTS-CLÉS : *Synthèse de contrôleurs, Ramadge Wonham, μ-calcul modal, jeu de parité, stratégie gagnante*

KEYWORDS: *Controller synthesis, Ramadge Wonham, modal μ-calculus, parity game, winning strategy*

1. Introduction

Cet article présente une méthode permettant de synthétiser un contrôleur dans le cadre proposé par Ramadge et Wonham dans [RAM 89]. Cependant, alors que Ramadge et Wonham proposent des contrôleurs vérifiant seulement des propriétés linéaires (sur une exécution donnée du système), la méthode que nous proposons permet de traiter n'importe quelle propriété exprimable dans une logique linéaire ou arborescente. La logique de spécification utilisée est le μ-calcul modal.

Le cœur de notre méthode est basé sur la relation forte qui existe entre synthétiser un contrôleur et calculer une stratégie gagnante dans un jeu de parité que nous allons expliciter.

Le problème posé est le suivant : étant donné un système de transitions $\mathcal{A}$ sur un alphabet d'événements dont certains peuvent être "évités" (contrôlés), et une spécification φ du comportement souhaité de ce système, on veut construire un système de transitions $\mathcal{C}$ (le contrôleur) tel que lorsqu'on le synchronise avec $\mathcal{A}$, le système produit résultant satisfasse φ.

Dans un premier temps, nous présenterons la notion de contrôlabilité pour les événements. Ensuite nous définirons les systèmes de transitions que nous allons utiliser et nous décrirons une façon informelle de résoudre le problème de la synthèse d'un contrôleur afin de familiariser le lecteur avec le problème.

Puis sera donnée la syntaxe du système d'équations utilisé pour la spécification ainsi qu'une représentation ensembliste plus concise et plus facilement manipulable de ces équations.

La sémantique de ce système sur le système de transitions sera donnée sous la forme d'un jeu de parité qui n'est intuitivement autre que le produit du système de transitions par le système d'équations.

Si une position particulière de ce jeu (que l'on pourrait qualifier de "position initiale") est gagnante, alors il existe une stratégie gagnante pour l'un des deux joueurs. De plus il existe une stratégie gagnante qui ne dépend que de la position courante. Cette stratégie nous fournira directement la fonction de transition du contrôleur.

Enfin, nous présenterons quelques problèmes qui ne sont pas traités dans cet article et dont certains seront résolus dans une publication à venir.

2. Les événements

Les événements qui constituent le comportement du système étudié sont représentés par les éléments d'un alphabet fini Σ.

La notion de contrôlabilité prend un sens lorsque l'on place le système dans un environnement. En un état donné du système, plusieurs événements différents peuvent se produire, et ceux qui sont dictés par l'environnement ne sont pas contrôlables : le système les subit. A l'inverse, certaines actions possibles du système en un état peuvent être évitées : un superviseur peut interdire au système d'effectuer cette action.

On voit bien ici se dessiner le problème de la synthèse de contrôleurs : il s'agit de prédire quelles actions du système il faut interdire pour ne pas se retrouver relégué dans un état indésirable au gré des événements incontrôlables produits par l'environnement.

Les notions d'événement contrôlable et d'événement incontrôlable sont clairement mutuellement exclusives. C'est pour cela que l'on peut partitionner Σ en les deux sous-ensembles Σ_u contenant les événements incontrôlables et Σ_c contenant les événements contrôlables.

3. Le système

Le système étudié sera ici modélisé par un système de transitions fini et déterministe. Cela suit le cadre de Ramadge et Wonham dans [RAM 89] qui appellent ce type de modèle des systèmes à événements discrets (DES).

On pose donc la définition suivante :

Un système de transitions est un quadruplet $\langle Q, \delta, q_0, \lambda \rangle$.

Q est l'ensemble fini des états du système, $\delta : Q \times \Sigma \to Q$ est sa fonction (partielle) de transition, q_0 est son état initial et $\lambda : Q \to \mathcal{P}(\textit{Prop})$ associe à chaque état l'ensemble des propriétés élémentaires qu'il satisfait.

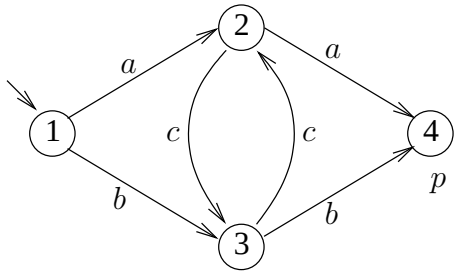
Prop est un ensemble quelconque de symboles de *propriétés élémentaires* des états (par exemple marqué/non marqué dans [RAM 89]).

Dans la suite, on appellera $\mathcal{A} = \langle Q, \delta, q_0, \lambda \rangle$ le système de transitions à contrôler.

Une exécution du système est un mot fini ou infini $w = w_0 w_1 \ldots \in \Sigma^\infty$ tel que $\delta(q_0, w)$ est défini (en étendant δ aux mots par morphisme), i.e. $\exists q_1, q_2, \ldots, q_i, \ldots$ tels que $\forall i \geq 0, \delta(q_i, w_i) = q_{i+1}$

L'ensemble des exécutions du système forme donc un langage préfixe qui peut aussi être vu comme un arbre dont les arcs sont étiquetés par des événements et les nœuds par des états de $\mathcal{A}$.

EXEMPLE. — Soit par exemple le système de transitions suivant sur l'alphabet $\Sigma_c = \{c\}, \Sigma_u = \{a, b\}$.

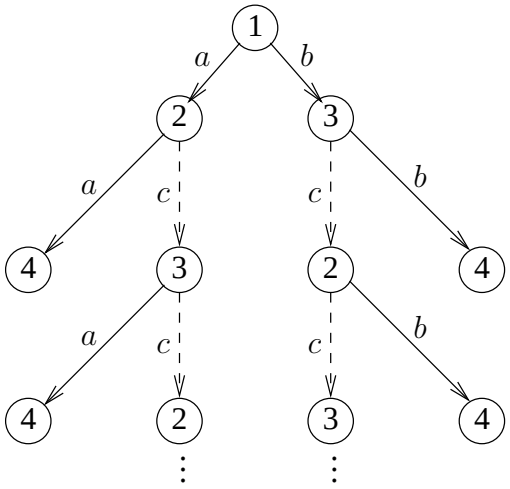


Il est défini par :

– $Q = \{1, 2, 3, 4\}, q_0 = 1$
– $\delta : (1, a) \mapsto 2, (1, b) \mapsto 3, (2, c) \mapsto 3, (3, c) \mapsto 2, (2, a) \mapsto 4, (3, b) \mapsto 4$
– $\lambda : 1 \mapsto \emptyset, 2 \mapsto \emptyset, 3 \mapsto \emptyset, 4 \mapsto \{p\}$

Ici, on se donne un seul symbole p de propriété élémentaire (*Prop* $= \{p\}$), avec lequel on distingue l'état 4.

Et on peut le déplier pour obtenir l'arbre de ses exécutions possibles :

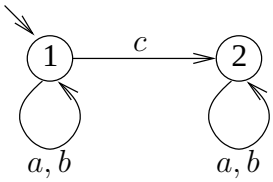


Les arcs correspondant à des événements contrôlables sont en pointillés. □

4. Le système contrôlé

Etant donnée une partition (Σ_c, Σ_u) de Σ en événements contrôlables et incontrôlables, un *contrôleur* est un système de transitions $\mathcal{S} = \langle Q^{\mathcal{S}}, \delta^{\mathcal{S}}, q_0^{\mathcal{S}}, \lambda^{\mathcal{S}} \rangle$ tel que $\forall q \in Q, \forall e \in \Sigma_u, \delta^{\mathcal{S}}(q, e)$ est défini.

EXEMPLE. — Voici un contrôleur S sur l'alphabet $\Sigma_c = \{c\}$, $\Sigma_u = \{a, b\}$.



□

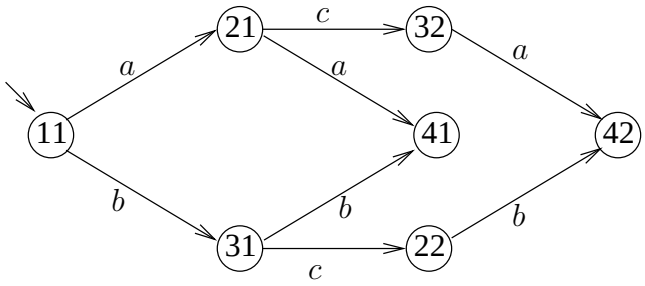
Si $\mathcal{A}$ est un système et $\mathcal{S}$ un contrôleur, le système contrôlé est le produit synchrone $\mathcal{A} \times \mathcal{S}$ défini par :

$Q^{\mathcal{A}\times\mathcal{S}} = Q \times Q^{\mathcal{S}}, q_0^{\mathcal{A}\times\mathcal{S}} = (q_0, q_0^{\mathcal{S}})$,

$\delta^{\mathcal{A}\times\mathcal{S}}((q, q')) = (\delta(q, a), \delta^{\mathcal{S}}(q', a))$ si $\delta(q, a)$ et $\delta^{\mathcal{S}}(q', a)$ sont définis,

$\forall (q, q') \in Q \times Q^{\mathcal{S}}, \lambda^{\mathcal{A}\times\mathcal{S}}((q, q')) = \lambda(q)$

EXEMPLE. — Voici le système A contrôlé par le système S des deux exemples précédents :



□

On peut d'ores et déjà imaginer les grandes lignes d'une méthode pour synthétiser un contrôleur : en effet, l'arbre des exécutions possibles contient des arcs que l'on peut couper à loisir. Ce sont ceux qui correspondent aux événements contrôlables.

Si donc on sait spécifier quels sous-arbres ne satisfont pas les critères que l'on s'est fixé, il suffit de trouver en amont un événement contrôlable et le supprimer, en vérifiant qu'on n'invalide pas une autre propriété souhaitée en le faisant.

Ayant abouti à cet arbre d'exécution élagué, il reste à lui redonner si cela est possible une forme finie, de manière à pouvoir en faire un contrôleur manipulable. On peut pour cela regrouper les nœuds dans des classes d'équivalence telles que tous les nœuds qui appartiennent à la même classe sont racines d'arbres isomorphes.

Dit plus rigoureusement, cela revient en fait à calculer l'automate déterministe minimal correspondant à cet ensemble d'exécutions. Evidemment, cela n'est possible

que si ce langage admet une représentation finie (s'il est régulier). Comme le système de départ est déjà régulier, la restriction va devoir porter sur la spécification des propriétés souhaitées.

Cette présentation au travers de l'arbre d'exécution est assez visuelle, mais elle a l'inconvénient de ne pas se traduire directement en algorithme : elle manipule explicitement un objet de taille infinie (l'arbre des exécutions) et le passage à l'automate quotient reste donc difficilement implémentable.

Il reste donc deux problèmes à résoudre : trouver un langage de spécification qui ne permette d'engendrer que des comportements réguliers, et trouver un algorithme qui permettre de passer directement du couple système de transitions – spécification à une représentation finie du contrôleur.

Nous proposons ici d'utiliser comme langage de spécification les formules du μ-calcul modal et nous donnons un algorithme qui permet d'obtenir un contrôleur à représentation finie lorsqu'il en existe un.

5. La spécification

La spécification sera donnée par un système d'équations modales de points fixes. Le pouvoir d'expression est le même que celui d'une formule du μ-calcul modal.

Plus précisément, chaque équation sera sous une forme gardée :

$$x_i \stackrel{\theta_i}{=} \bigvee_{j \in J_i} \left(\bigwedge_{a \in A_{i,j} \subseteq \Sigma} (a) x_{a,i,j} \wedge \bigwedge_{p \in P_{i,j} \subseteq \mathit{Prop}} p \right)$$

pour $i = 0, \ldots, n$, $\theta_i \in \{\mu, \nu\}$, et le système d'équations considéré est la séquence des variables $x_i \in X$, donc ordonnée selon les i croissants.

X contient aussi un autre symbole $x_\top$.

(a) représente soit une modalité existentielle $\langle a \rangle$, soit une modalité universelle $[a]$.

Nous allons noter chaque conjonction par un triplet $r = (B_r, \sigma_r, P_r)$ que nous appellerons *règle*, par analogie avec les automates d'arbres. L'ensemble des règles est noté R.

– $B_r \subseteq \Sigma$ contient les événements qui apparaissent sous la forme existentielle ($\langle a \rangle$) dans la conjonction

– $\sigma_r : \Sigma \to X$ fournit l'équation qui doit etre satisfaite par l'état suivant : $\sigma_r(a) = x_{a,i,j}$ si $a \in A_r = A_{i,j}$, $x_\top$ sinon.

– $P_r \subseteq \mathit{Prop}$ est l'ensemble des propriétés que l'état doit satisfaire.

Si elle n'y est pas déjà, on ajoute à l'ensemble des règles la règle $r_\top = (\emptyset, \sigma_\top, \emptyset)$ avec $\forall e \in \Sigma, \sigma_\top(e) = x_\top$.

On peut représenter un système d'équations modales par un ensemble de règles pour chaque variable ; on se donne pour cela une application $\Delta : X \to \mathcal{P}(R)$ avec

$$\Delta(x_i) = \{(B_{i,j}, \sigma_{i,j}, P_{i,j}) | j \in J_i\}$$

et on associe à chaque variable x_i l'entier $\text{rang}(x_i) = \begin{cases} 2i & \text{si } \theta_i = \nu \\ 2i+1 & \text{si } \theta_i = \mu \end{cases}$

On a aussi $\Delta(x_\top) = \{r_\top\}$ et $\text{rang}(x_\top) = 0$.

EXEMPLE. — Sur notre exemple, on souhaite assurer que l'on finira toujours par atteindre l'état 4.

Le système d'équations spécifiant ce comportement est réduit à une seule équation :

$$x \stackrel{\mu}{=} p \vee ([a]\, x \wedge [b]\, x \wedge [c]\, x)$$

Ce qui peut se traduire dans l'écriture ensembliste par :

$X = \{x, x_\top\}, R = \{r_1, r_2, r_\top\},$

$\Delta(x) = \{r_1, r_2\}, \Delta(x_\top) = \{r_\top\}$

$r_1 = (\emptyset, \sigma_\top, \{p\}),$
$r_2 = (\emptyset, \sigma_2, \emptyset), \quad \forall e \in \Sigma, \sigma_2(e) = x,$
$r_\top = (\emptyset, \sigma_\top, \emptyset), \quad \forall e \in \Sigma, \sigma_\top(e) = x_\top$

□

6. Jeu de parité associé

En reprenant le vocabulaire de [ARN 01], nous allons construire un jeu de parité permettant de construire le contrôleur.

Brièvement, un jeu de parité est un graphe dont les sommets appelés *positions* appartiennent chacun à un des deux joueurs (Adam ou Eve). A chaque sommet on associe un *rang* qui est un nombre entier positif ou nul.

Une *partie* se déroule en laissant chaque joueur pousser un jeton le long des arcs du graphe de jeu, en partant d'une position initiale.

La partie est gagnée par un joueur si l'autre ne peut plus jouer. Si elle est infinie, le rang maximal rencontré sur les positions parcourues par le jeton désigne le vainqueur : Eve si le rang maximal est pair, et Adam s'il est impair.

Chaque joueur peut adopter une *stratégie*, qui est une application décrivant la position vers laquelle le joueur déplacera le jeton en une position donnée. Une stratégie

est dite gagnante en une position (elle aussi dite gagnante) si toute partie démarrant en cette position et jouée par ce joueur en respectant sa stratégie est gagnée quelle que soit la stratégie de l'adversaire.

Ici, nous adopterons le point de vue d'Eve : quand il sera question de partie, de position ou de stratégie gagnante, cela signifiera qu'elle est gagnante pour Eve.

Les positions d'Eve sont dans $\mathit{Pos}_E = (Q \times X) \cup (\Sigma \times Q \times R) \cup \{\bot\}$ et celles d'Adam sont dans $\mathit{Pos}_A = (Q \times R) \cup \{\top\}$. L'ensemble des positions est noté *Pos*.

Le jeton se situe initialement en (q_0, x_0).

Déplacements d'Adam

Si le jeton arrive en $\top$, Adam ne peut plus jouer.

Sur une position (q, r), si l'état q ne satisfait pas l'une des propriétés élémentaires de P_r, Adam peut se déplacer vers $\bot$. Sinon, Adam choisit une lettre, c'est-à-dire peut déplacer le jeton vers toutes les positions (a, q, r) pour tout $a \in \Sigma$.

Le rang des positions d'Adam n'a pas d'importance et on le fixe à 0 : toutes les positions d'Adam sont transitoires, et ne déterminent en rien l'issue d'une partie infinie.

Déplacements d'Eve

Si le jeton arrive en $\bot$, Eve ne peut plus jouer. Le rang de $\bot$ n'a pas d'importance, on le fixe à 1.

Les déplacements possibles d'Eve sont les suivants :

choix d'une règle :
$(q, x) \longrightarrow (q, r)$ $\quad \forall r \in \Delta(x)$
si $\delta(q, a)$ est défini :
$(a, q, r) \longrightarrow (\delta(q, a), \sigma_r(a))$
$(a, q, r) \longrightarrow \top$ $\quad$ si $a \in \Sigma_c \setminus B_r$
si $\delta(q, a)$ n'est pas défini :
$(a, q, r) \longrightarrow \bot$ $\quad$ si $a \in B_r$
$(a, q, r) \longrightarrow \top$ $\quad$ si $a \notin B_r$

Le rang des (a, q, r) n'a pas d'importance, on le fixe à 1. Par contre, $\text{rang}((q, x)) = \text{rang}(x)$.

En (q, x), Eve choisit une règle. En (a, q, r), Adam a choisi de jouer la lettre a et Eve peut :

– chercher à s'assurer que l'équation dans la portée de la modalité doit être vérifiée dans l'état successeur du système de transitions

– décider qu'elle a gagné si la modalité est satisfaite même en l'absence de transition (elle n'appartient pas à B_r) et que l'événement correspondant est contrôlable. Ce choix indique que cette transition doit être interdite par le contrôleur

– constater que la transition correspondant à la lettre choisie par Adam n'existe pas et donc perdre si la modalité est existentielle (la lettre appartient à B_r) ou gagner si la modalité est universelle.

REMARQUE. —
Toute position de la forme $(q, x_\top)$ est gagnante pour Eve puisque toutes les positions rencontrées à partir de celle-ci seront de rang 0, ce qui assure la victoire à Eve dans le cas d'une partie infinie. Si une partie est finie, c'est forcément qu'elle se termine sur la position $\top$, qui est aussi gagnante pour Eve (en effet, $B_{r_\top} = \emptyset$).

7. Synthèse du contrôleur

A partir du jeu décrit précédemment, nous calculons l'ensemble des positions gagnantes pour Eve. Si (q_0, x_0) appartient à cet ensemble, nous pouvons synthétiser un contrôleur $\mathcal{C} = \langle Q^{\mathcal{C}}, \delta^{\mathcal{C}}, q_0^{\mathcal{C}}, \lambda^{\mathcal{C}} \rangle$ en utilisant une stratégie ε gagnante en (q_0, x_0) d'Eve, ne dépendant que de la position courante. Une telle stratégie existe toujours si (q_0, x_0) est gagnante. On peut retrouver ce théorème bien connu de la théorie des jeux de parité dans [ARN 01, Th. 4.3.8, p.92]

$Q^{\mathcal{C}} = Q \times X,\ q_0^{\mathcal{C}} = (q_0, x_0),\ \lambda^{\mathcal{C}}((q, x)) = \lambda(q),$

$\delta^{\mathcal{C}}((q, x), a) = \varepsilon((a, q, r))$ si $\varepsilon((a, q, r)) \neq \top$, où $(q, r) = \varepsilon((q, x))$.

L'état qui correspondrait à la position $\perp$ n'est pas accessible à partir de (q_0, x_0) puisque la stratégie d'Eve est gagnante en (q_0, x_0).

De plus, afin de satisfaire la définition de contrôleur, il faut compléter $\mathcal{C}$ en le contrôleur proprement dit $\mathcal{C}^+$, défini comme suit :

$Q^{\mathcal{C}^+} = Q^{\mathcal{C}},\ q_0^{\mathcal{C}^+} = q_0^{\mathcal{C}},\ \lambda^{\mathcal{C}^+} = \lambda^{\mathcal{C}}$ et :

$$\forall a \in \Sigma_u, \delta^{\mathcal{C}^+}((q, x), a) = \begin{cases} \delta^{\mathcal{C}}((q, x), a) & \text{si } \delta^{\mathcal{C}}((q, x), a) \text{ est défini} \\ (q, x) & \text{sinon} \end{cases}$$

Nous allons maintenant construire le produit synchrone de $\mathcal{A}$ et de $\mathcal{C}^+$ afin de s'assurer qu'il vérifie bien le système d'équations.

$Q^{\mathcal{A}\times\mathcal{C}^+} = Q \times (Q \times X),$

$q_0^{\mathcal{A}\times\mathcal{C}^+} = (q_0, q_0^{\mathcal{C}^+}) = (q_0, (q_0, x_0)),$

$\lambda^{\mathcal{A}\times\mathcal{C}^+}((q, (q, x))) = \lambda(q),$

Si $\delta(q, a)$ et $\delta^{\mathcal{C}^+}(q^{\mathcal{C}^+}, a)$ sont définis, et avec $(q, r) = \varepsilon((q, x))$,

$$\begin{aligned}\delta^{\mathcal{A}\times\mathcal{C}^+}((q,q^{\mathcal{C}^+}),a) &= (\delta(q,a),\delta^{\mathcal{C}}(q^{\mathcal{C}},a))\\ &= (\delta(q,a),(\delta(q,a),\sigma_r(a)))\end{aligned}$$

Proposition. $\mathcal{A}\times\mathcal{C}^+ = \mathcal{C}$

Preuve. On remarque d'abord que $\mathcal{A}\times\mathcal{C}^+ = \mathcal{A}\times\mathcal{C}$. En effet, pour que cela soit vrai, il suffit de montrer que si $\delta(q,a)$ est défini et si $\delta^{\mathcal{C}^+}((q,x),a)$ est défini alors $\delta^{\mathcal{C}}((q,x),a)$ est défini. Or, si $\delta(q,a)$ est défini et si $\delta^{\mathcal{C}}((q,x),a)$ ne l'est pas c'est que $\varepsilon((a,q,r)) = \top$ (avec $(q,r) = \varepsilon((q,x))$) et donc que $a \in \Sigma_c$. Mais alors $\delta^{\mathcal{C}^+}((q,x),a) = \delta^{\mathcal{C}}((q,x),a)$.

On remarque ensuite que tous les états accessibles de $\mathcal{A}\times\mathcal{C}^+$ sont de la forme $(q,(q,x))$, que l'ont peut donc abréger sans perte d'information en (q,x). Nous choisirons donc de poser :

$Q^{\mathcal{A}\times\mathcal{C}} = Q\times X,\ q_0^{\mathcal{A}\times\mathcal{C}} = (q_0,x_0),\ \lambda^{\mathcal{A}\times\mathcal{C}} = \lambda^{\mathcal{A}\times\mathcal{C}^+}$,

$\delta^{\mathcal{A}\times\mathcal{C}}((q,x),a) = (\delta(q,a),\sigma_r(a))$ si $\delta^{\mathcal{C}}(q^{\mathcal{C}},a)$ est défini, où $(q,r) = \varepsilon((q,x))$.

Ces égalités montrent directement que $\mathcal{A}\times\mathcal{C} = \mathcal{C}$. □

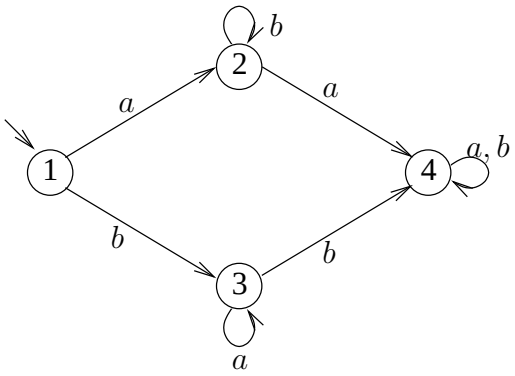
Théorème.

$\mathcal{A}\times\mathcal{C}^+$ satisfait le système d'équations.

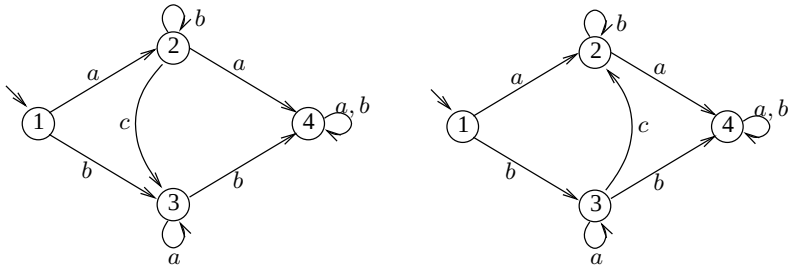
Preuve. La proposition précédente montre que $\mathcal{A}\times\mathcal{C}^+ = \mathcal{C}$. Comme $\mathcal{C}$ satisfait le système d'équations par construction, $\mathcal{A}\times\mathcal{C}^+$ aussi. □

EXEMPLE. — Le jeu qui permet de trouver les différents contrôleurs possibles (un par stratégie positionnelle gagnante) est trop volumineux pour être reproduit ici. Cependant, voici les trois contrôleurs qu'il permet d'obtenir :

le premier supprime toutes les transitions contrôlables, ce qui permet effectivement d'atteindre l'état 4.



Les deux autres contrôleurs suppriment une des deux transitions contrôlables afin de casser la boucle du milieu. En effet, c'est cette boucle qui risquait d'empêcher d'atteindre l'état 4.



On peut remarquer que les contrôleurs donnés ici ne sont pas optimaux dans le sens où il serait possible d'autoriser un nombre quelconque mais borné de c consécutifs pour satisfaire la spécification. Comme il est évident qu'il n'est *pas possible* d'assurer une contrainte aussi permissive, il paraît assez naturel de laisser la personne qui écrit la spécification expliciter le nombre de transitions contrôlables qu'elle souhaite autoriser en allongeant le système d'équations. □

8. Calcul de stratégies gagnantes et complexité

La méthode qui est proposée ici repose sur le calcul d'une stratégie positionnelle (qui ne dépend que de la position courante) dans un jeu de parité déterminé par le problème de contrôle donné. La construction du contrôleur à partir d'une telle stratégie est immédiate.

Le calcul d'une stratégie gagnante est un problème central de la théorie des jeux, et une solution effective à ce problème est nécessaire au bon fonctionnement de notre construction. On sait qu'il est dans $NP \cap$ co-NP et les algorithmes actuellement disponibles sont de complexité n^d dans le pire des cas, où n est le produit de la taille du système de transitions avec la taille du système d'équations de la spécification et d est le nombre d'alternances (le nombre de passages de μ à ν ou de ν à μ) dans le système d'équations.

On peut noter aussi l'existence de l'algorithme proposé par Vöge et Jurdziński dans [VöG 00] dont la complexité n'est pas connue, mais qui est expérimentalement le plus performant.

Quoiqu'il en soit, le nombre d'alternances manipulées en pratique est souvent faible (une ou deux). Les propriétés de la logique temporelle linéaire par exemple, qui permet de spécifier les comportements vérifiés par Ramadge et Wonham, sont toutes exprimables avec une seule alternance. Il est de plus communément admis que à l'heure actuelle il est extrêmement difficile d'écrire des spécifications ayant plus de deux alternances et un tel besoin ne s'est pas fait sentir.

C'est d'ailleurs grâce au fait que la propriété donnée en exemple n'a qu'une alternance que nous avons pu calculer le jeu de parité, mais surtout les trois contrôleurs à la main.

9. Conclusion

Cet article présente le lien fort qui existe entre la synthèse de contrôleur et le calcul de stratégies gagnantes dans un jeu. Il ouvre les portes à plusieurs améliorations qui ne sont pas traitées dans cet article.

On peut distinguer deux axes principaux :

– l'exemple proposé soulève déjà le problème de l'optimalité du contrôleur généré. Il serait intéressant de déterminer les critères sous lesquels on peut ou ne peut pas générer un contrôleur optimal, c'est-à-dire le plus permissif. En particulier, on constate que le contrôleur $\mathcal{S}$ donné en exemple n'est pas trouvé par notre méthode alors qu'il est plus permissif.

– la seule notion présentée ici est la notion de contrôlabilité. Cependant, la notion d'observabilité, toujours dans [RAM 89], est extrêmement importante pour modéliser des situations réelles. Il est possible de la faire entrer dans le cadre présenté ici au prix d'une astuce de codage dans le système de transitions.

La méthode présentée ici est aussi sans doute propice à la résolution de problèmes de contrôle distribué, où plusieurs contrôleurs participent au contrôle d'un système de transitions. Une généralisation possible est aussi de permettre aux événements de *devenir* contrôlables ou inobservables au fur et à mesure que le système de transitions s'exécute.

10. Bibliographie

[ARN 01] ARNOLD A., NIWIŃSKI D., *Rudiments of μ-calculus*, Studies in Logic and the Foundations of Mathematics, North-Holland, 2001.

[RAM 89] RAMADGE P., WONHAM W. M., « The Control of Discrete Event Systems », *Proceedings of the IEEE*, vol. 77, 1989, p. 79–98.

[VöG 00] VÖGE J., JURDZIŃSKI M., « A Discrete Strategy Improvement Algorithm for Solving Parity Games », *CAV : International Conference on Computer Aided Verification*, 2000.