Programmation par contraintes pour la synthèse de contrôleur

Gérard Verfaillie **Cédric Pralet**

Onera - The French Aerospace Lab, F-31055, Toulouse, France

{Gerard.Verfaillie,Cedric.Pralet}@onera.fr

Résumé

Dans ce papier, nous montrons comment le problème de synthèse d'un contrôleur d'un système dynamique satisfaisant certaines propriétés de sûreté, en contexte éventuellement non déterministe et partiellement observable, peut être modélisé comme un pur problème de satisfaction de contraintes, en remplaçant la propriété d'atteignabilité par une propriété dite d'atteignabilité faible. Nous montrons, d'abord sur un exemple jouet illustratif, puis sur un exemple réel de contrôle d'un sous-système d'un satellite, comment des outils standards de programmation par contraintes peuvent être utilisés pour modéliser et résoudre le problème de synthèse de contrôleur. Nous concluons sur les forces et les faiblesses de l'approche proposée.

Abstract

In this paper, we show how the problem of synthesis of a controller for a dynamic system that must satisfy some safety properties, possibly in a non deterministic and partially observable setting, can be modeled as a pure constraint satisfaction problem, by replacing the reachability property by a so-called weak reachability property. We show, first on a toy illustrative example, then on a real-world example of control of a satellite subsystem, how standard constraint programming tools can be used to model and solve the controller synthesis problem. We conclude with the strengths and weaknesses of the proposed approach.

1 Le problème de synthèse de contrôleur

1.1 Une vue informelle

Dans ce papier, nous nous intéressons au contrôle en boucle fermée de systèmes dynamiques (voir la figure 1). Plus précisément, nous nous intéressons à des contrôleurs logiciels implémentés sur des calculateurs digitaux (l'essentiel des contrôleurs modernes), qui n'agissent pas de façon continue sur le système qu'ils contrôlent, mais de façon discrète, par étapes successives.

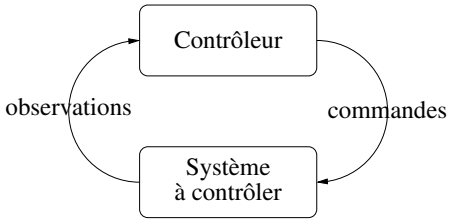

FIGURE 1 – Contrôle en boucle fermée d'un système dynamique

A chaque étape, le contrôleur collecte des informations sur le système à contrôler (observations) et prend de façon réactive une décision de contrôle (commandes) en fonction des informations recueillies. Les observations peuvent venir du système lui-même, d'autres systèmes avec lesquels il est en interaction ou d'opérateurs humains. Elles peuvent contenir des informations synthétiques sur les observations et les commandes passées (mémoire du contrôleur). En sens inverse, les commandes sont adressées au système lui-même ou à d'autres systèmes ou sont des informations à destination des opérateurs.

Nous supposons que l'évolution du système sous l'impact des commandes (transitions) est non déterministe et markovienne : plusieurs transitions sont possibles à partir de l'état et de la commande courants ; la transition effective ne dépend que de l'état et de la commande courants ; elle ne dépend pas des états et commandes précédents. Mais nous ne supposons pas la connaissance d'une distribution de probabilité sur l'ensemble des transitions possibles : seules sont distinguées les transitions possibles et impossibles. De plus, nous ne supposons pas que l'état réel du sys-

tème à chaque étape soit connu du contrôleur : seules des observations sont disponibles. Nous faisons par contre l'hypothèse que l'ensemble des états possibles du système, l'ensemble des observations possibles et l'ensemble des commandes possibles sont discrets et finis.

Nous nous intéressons à des propriétés dites de sûreté, c'est-à-dire à des propriétés qui doivent être satisfaites par toute transition parcourue par le système. Dans un tel contexte, le problème de synthèse de contrôleur consiste à construire hors-ligne (avant la mise en service du contrôleur) ce qu'on appelle une politique, c'est-à-dire une fonction qui associe à chaque ensemble d'observations un ensemble de commandes et qui, en dépit du non déterminisme des transitions, garantisse que toute transition parcourue par le système satisfasse les propriétés de sûreté.

La section 1 introduit le problème de synthèse de contrôleur. La section 2 présente un modèle CSP de ce problème, utilisant la notion d'atteignabilité faible. La section 3 montre comment l'approche proposée peut être appliquée au problème de contrôle d'un sous-système d'un satellite. La section 4 conclut sur les forces et les faiblesses de cette approche.

1.2 Une définition formelle

Le problème de synthèse de contrôleur peut être formellement défini de la façon suivante.

Données Les données du problème sont :

- une séquence finie S de variables d'état du système ;
- une sous-séquence $O \subseteq S$ de variables d'état observables ;
- une séquence finie C de variables de commande ;
- un ensemble $I \subseteq \mathbf{d}(S)$ d'états initiaux possibles du système, supposé non vide ;
- un ensemble $T \subseteq \mathbf{d}(S) \times \mathbf{d}(C) \times \mathbf{d}(S)$ de transitions possibles ;
- un ensemble $P \subseteq \mathbf{d}(S) \times \mathbf{d}(C) \times \mathbf{d}(S)$ de transitions acceptables.

Toute variable x d'état ou de commande est supposée avoir un domaine de valeurs fini, noté $\mathbf{d}(x)$. Si X est une séquence de variables, le produit cartésien des domaines de valeurs des variables de X est noté $\mathbf{d}(X)$. Si X est une séquence de variables et Y une sous-séquence et si A est une affectation de X, $A_{\downarrow Y}$ désigne l'affectation de Y (projection).

Les ensembles I, T et P peuvent être implicitement représentés par des ensembles (finis) de contraintes.

Politique Une politique π est une fonction partielle de $\mathbf{d}(O)$ dans $\mathbf{d}(C)$, partielle parce qu'une politique ne doit pas être définie pour toutes les observations possibles, mais uniquement pour celles qui sont atteignables (associées à au moins un état atteignable). Soit $df_\pi \subseteq \mathbf{d}(O)$ le domaine de définition de π.

États atteignables Étant donnée une politique π, l'ensemble r_π des états qui sont atteignables à partir de l'état initial en suivant π (r pour *reachability*) peut être défini de la façon suivante. Si $r_{\pi,k}$ est l'ensemble des états atteignables en au plus de k étapes à partir d'un état initial, on a :

$$\forall s \in \mathbf{d}(S),\ r_{\pi,0}(s) = I(s) \tag{1}$$

$$\begin{aligned} &\forall k, 1 \leq k \leq |\mathbf{d}(S)| - 1, \\ &r_{\pi,k}(s) = r_{\pi,k-1}(s) \vee (\exists s' \in \mathbf{d}(S), \\ &r_{\pi,k-1}(s') \wedge df_\pi(s'_{\downarrow O}) \wedge T(s', \pi(s'_{\downarrow O}), s)) \end{aligned} \tag{2}$$

$$r_\pi(s) = \max_{0 \leq k \leq |\mathbf{d}(S)|-1} r_{\pi,k}(s) \tag{3}$$

L'équation 1 indique qu'un état est atteignable en 0 étape si et seulement si il est un état initial possible. L'équation 2 indique qu'un état est atteignable en k étapes si et seulement si il est atteignable en $k-1$ étapes ou si une transition est possible depuis un état atteignable en $k-1$ étapes. Enfin, l'équation 3 indique qu'un état est atteignable si et seulement si il est atteignable en k étapes, avec $0 \leq k \leq |\mathbf{d}(S)| - 1$. En effet, au pire, tous les états sont atteignables, il existe un seul état initial possible et un seul nouvel état est atteignable chaque fois que k est incrémenté de 1. Il faut souligner que, suivant cette définition, l'ensemble des états atteignables dépend de la politique retenue.

Exigences sur la politique Les exigences sur la politique sont les suivantes :

$$\forall s \in \mathbf{d}(S), \quad r_\pi(s) \rightarrow df_\pi(s_{\downarrow O}) \tag{4}$$

$$\begin{aligned} &r_\pi(s) \rightarrow (\exists s' \in \mathbf{d}(S), \\ &T(s, \pi(s_{\downarrow O}), s')) \end{aligned} \tag{5}$$

$$\begin{aligned} &r_\pi(s) \rightarrow (\forall s' \in \mathbf{d}(S), \\ &T(s, \pi(s_{\downarrow O}), s') \rightarrow P(s, \pi(s_{\downarrow O}), s')) \end{aligned} \tag{6}$$

L'équation 4 spécifie que la politique doit être définie pour tout état atteignable. L'équation 5 spécifie que la politique ne doit pas introduire de blocage : pour tout état atteignable, en suivant la politique, il existe une transition possible. Enfin, l'équation 6 spécifie que la

politique doit être “acceptable” : pour tout état atteignable, en suivant la politique, toute transition possible est acceptable. Il faut souligner que ces exigences ne doivent pas être satisfaites sur tous les états possibles, mais seulement sur ceux qui sont atteignables à partir de l'état initial en suivant la politique retenue.

1.3 Un exemple jouet

À titre d'illustration, considérons l'exemple jouet utilisé dans [17]. On considère un robot capable de se mouvoir sur la grille de la figure 2 où les murs apparaissent en gras. Initialement, le robot se trouve sur l'une des deux places d'abscisse $x = 2$. À chaque étape, le robot se trouve sur une place p qu'il ne connaît pas directement. Il observe seulement les murs présents immédiatement autour de p. À chaque étape, il se déplace vers le nord, le sud, l'est ou l'ouest. Il n'est pas autorisé à rester sur place. En raison de la présence des murs, seuls certains de ces mouvements sont réalisables. Le robot doit éviter la place marquée X $(x = 3, y = 1)$ considérée comme dangereuse.

FIGURE 2 – Problème de contrôle d'un robot sur une grille

Pour modéliser ce problème, on considère six variables d'état : $S = \{x, y, w_N, w_S, w_E, w_W\}$. x et y représentent la position du robot, avec $\mathbf{d}(x) = [1..3]$ et $\mathbf{d}(y) = [1..2]$. w_N, w_S, w_E et w_W sont des variables booléennes qui représentent la présence (ou non) d'un mur au nord, au sud, à l'est et à l'ouest de la place courante. Seules les variables d'état w_N, w_S, w_E et w_W sont observables par le robot : $O = \{w_N, w_S, w_E, w_W\}$. On considère une seule variable de commande m qui représente le mouvement du robot, avec $\mathbf{d}(m) = \{m_N, m_S, m_E, m_W\}$ (quatre mouvements possibles).

L'ensemble I des états initiaux possibles est défini par les cinq contraintes unaires suivantes : $x = 2$, w_N, w_S, $\neg w_E$ et $\neg w_W$.

L'ensemble T des transitions possibles est défini par les contraintes suivantes :

$$w_N \rightarrow (m \neq m_N)$$
$$w_S \rightarrow (m \neq m_S)$$
$$w_E \rightarrow (m \neq m_E)$$
$$w_W \rightarrow (m \neq m_W)$$

$$x' = x + (m = m_E) - (m = m_W)$$
$$y' = y + (m = m_N) - (m = m_S)$$

$$w'_N = (y' = 2 \vee (y' = 1 \wedge x' = 2))$$
$$w'_S = (y' = 1 \vee (y' = 2 \wedge x' = 2))$$
$$w'_E = (x' = 3)$$
$$w'_W = (x' = 1)$$

Les quatre premières contraintes traduisent les mouvements irréalisables du fait de la présence des murs. Les deux suivantes expriment les transitions possibles qui, dans ce cas, sont déterministes : une seule transition possible associée à chaque paire état-commande. Pour toute variable d'état z, z' représente sa valeur à l'étape suivante. Les quatre dernières contraintes traduisent la connaissance disponible sur la topologie de la grille.

L'ensemble P des transitions acceptables est défini par la contrainte binaire $\neg((x' = 3) \wedge (y' = 1))$.

1.4 Méthodes existantes

La méthode de loin la plus utilisée pour construire un contrôleur consiste à le spécifier en utilisant un langage de programmation généraliste ou spécialisé, par exemple un langage de la famille des langages synchrones [10]. Une fois le contrôleur programmé, ses propriétés peuvent être vérifiées, soit expérimentalement par simulation, soit formellement en utilisant des outils de preuve ou de *model-checking* [5].

La synthèse de contrôleur est une approche alternative, qui vise la construction automatique d'un contrôleur à partir des propriétés physiques du système à contrôler et des exigences sur le système contrôlé. Le contrôleur produit est valide par construction et aucune vérification n'est théoriquement à réaliser après construction.

Les premiers travaux en synthèse de contrôleur utilisent les automates et la théorie des langages [19]. Par la suite, de nombreux travaux utilisent les automates pour représenter le système physique et la logique temporelle pour exprimer les exigences [15] (voir par exemple les outils ANZU et RATSY [12, 20] qui supposent tous deux une observabilité complète). Certains travaux font appel au μ-calcul [1]. L'outil MBP (*Model-Based Planner* [14]) utilise des techniques dites

de *symbolic model-checking*, à base de BDD (*Binary Decision Diagrams*), pour synthétiser des contrôleurs qui garantissent qu'un but est atteint en dépit du non déterminisme et de l'observabilité partielle [3]. Des algorithmes de recherche génériques pour la synthèse de contrôleurs à mémoire finie sont proposés dans [4, 17], avec les mêmes hypothèses de non déterminisme et d'observabilité partielle (avec cependant certaines restrictions dans [4]).

On doit souligner la proximité entre le problème de synthèse de contrôleur et les (PO)MDP (*(Partially Observable) Markov Decision Processes* [18]) pour lesquels des algorithmes à base de programmation dynamique sont les plus utilisés. La première différence entre les deux problèmes est que, dans les (PO)MDP, une distribution de probabilité conditionnelle sur les états résultant d'une transition et sur les observations résultant d'un état est supposée être disponible. La seconde différence est que les exigences prennent dans les (PO)MDP la forme d'un critère additif à optimiser.

On doit aussi souligner la différence entre le problème de synthèse de contrôleur et le problème de planification en Intelligence Artificielle [8]. En planification, on s'intéresse à des propriétés d'atteignabilité : un état but doit être atteint et le contrôle est supposé être stoppé une fois le but atteint. En synthèse de contrôleur, on s'intéresse à des propriétés de sûreté qui doivent être satisfaites tout au long de la trajectoire du système et le contrôle est supposé sans fin.

2 Formulation comme un problème de satisfaction de contraintes

2.1 Difficultés

Supposons que l'on associe à chaque observation $o \in \mathbf{d}(O)$ une variable $\pi(o)$ de domaine $\mathbf{d}(C) \cup \{\perp\}$, représentant la commande à appliquer lorsque o est observée. La valeur $\perp$ sert à représenter l'absence de commande. Supposons que l'on associe à chaque état $s \in \mathbf{d}(S)$, les variables suivantes :

— une variable booléenne $r_\pi(s)$ qui représente le fait que s soit atteignable (ou non) ;

— pour tout $k, 0 \leq k \leq |\mathbf{d}(S)| - 1$, une variable booléenne $r_{\pi,k}(s)$ qui représente le fait que s soit atteignable (ou non) en au plus k étapes.

Si l'on remplace $df_\pi(o)$ par $\pi(o) \neq \perp$, les équations 1 à 6 forment un problème de satisfaction de contraintes P (CSP [21]) qui modélise précisément le problème de synthèse de contrôleur. Malheureusement, le nombre de variables de P est prohibitif, essentiellement du fait des variables $r_{\pi,k}(s)$: pour chaque état $s \in \mathbf{d}(S)$, on a $|\mathbf{d}(S)|$ variables $r_{\pi,k}(s)$, d'où au total $|\mathbf{d}(S)|^2$ variables $r_{\pi,k}(s)$. Pour contourner cette difficulté, nous allons utiliser une relaxation de la propriété d'atteignabilité que nous appellons atteignabilité faible.

2.2 Atteignabilité et atteignabilité faible

Nous adoptons la définition suivante de l'atteignabilité faible : une relation wr_π est une relation d'atteignabilité faible associée à une politique π (wr pour *weak reachability*) si et seulement si elle satisfait les deux équations suivantes :

$$\forall s \in \mathbf{d}(S), \quad I(s) \rightarrow wr_\pi(s) \tag{7}$$

$$\forall s, s' \in \mathbf{d}(S), \quad (wr_\pi(s) \wedge df_\pi(s_{\downarrow O}) \wedge T(s, \pi(s_{\downarrow O}), s')) \rightarrow wr_\pi(s') \tag{8}$$

L'équation 7 exprime que, si un état est un état initial possible, il est faiblement atteignable. L'équation 8 exprime que, si un état s est faiblement atteignable et si une transition est possible depuis l'état s vers un autre état s', l'état s' est lui aussi faiblement atteignable. À partir de cette définition de l'atteignabilité faible, les quatre propriétés suivantes peuvent être établies.

Propriété 1 *Soit π une politique. La relation d'atteignabilité associée r_π est unique. Il peut par contre exister plusieurs relations d'atteignabilité faible associées wr_π.*

La relation d'atteignabilité est unique parce qu'elle est définie par les équations 1 à 3 qui sont toutes des égalités. L'existence possible de plusieurs relations d'atteignabilité faible est mise en évidence par l'exemple de la figure 3 où est représenté le graphe d'atteignabilité associée à une politique. Dans ce graphe, les nœuds représentent les états et les arcs des transitions possibles.

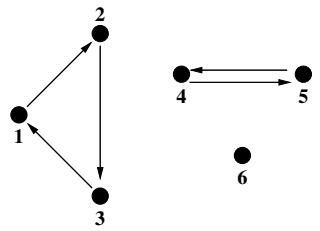

FIGURE 3 – Exemple de graphe d'atteignabilité associée à une politique

Sur cet exemple, si 1 est le seul état initial possible, la relation d'atteignabilité r est définie par l'ensemble d'états $\{1, 2, 3\}$. Par contre, les quatre relations définies respectivement par les ensembles d'états

$\{1,2,3\}$, $\{1,2,3,4,5\}$, $\{1,2,3,6\}$ et $\{1,2,3,4,5,6\}$ satisfont toutes les équations 7 et 8 et sont donc toutes des relations d'atteignabilité faible.

Propriété 2 *Soient π une politique, r_π la relation d'atteignabilité associée et wr_π une relation d'atteignabilité faible associée. r_π est un sous-ensemble de wr_π.*

Pour établir cette propriété, montrons que $\forall s' \in \mathbf{d}(S), r_\pi(s') \rightarrow wr_\pi(s')$.

Si $r_\pi(s')$, il existe k, $0 \leq k \leq |\mathbf{d}(S)| - 1$, tel que $r_{\pi,k}(s')$ (d'après l'équation 3). Montrons par récurrence sur k que $\forall k \geq 0, \forall s' \in \mathbf{d}(S), r_{\pi,k}(s') \rightarrow wr_\pi(s')$.

Pour $k = 0$, si $r_{\pi,0}(s')$, on a $I(s')$ (d'après l'équation 1) et donc $wr_\pi(s')$ (d'après l'équation 7).

Pour $k > 0$, supposons que $\forall s' \in \mathbf{d}(S), r_{\pi,k-1}(s') \rightarrow wr_\pi(s')$. Si $r_{\pi,k}(s')$, on a (d'après l'équation 2) soit $r_{\pi,k-1}(s')$ et donc $wr_\pi(s')$ (d'après l'hypothèse de récurrence), soit $\exists s \in \mathbf{d}(S)$, $r_{\pi,k-1}(s) \wedge df_\pi(s_{\downarrow O}) \wedge T(s, \pi(s_{\downarrow O}), s')$ et donc $\exists s \in \mathbf{d}(S)$, $wr_\pi(s) \wedge df_\pi(s_{\downarrow O}) \wedge T(s, \pi(s_{\downarrow O}), s')$ (toujours d'après l'hypothèse de récurrence) et finalement $wr_\pi(s')$ (d'après l'équation 8). On en déduit que $\forall s' \in \mathbf{d}(S), r_{\pi,k}(s') \rightarrow wr_\pi(s')$.

En conséquence, $\forall s' \in \mathbf{d}(S), r_\pi(s') \rightarrow wr_\pi(s')$.

Propriété 3 *Soit π une politique. La relation d'atteignabilité associée r_π est une relation d'atteignabilité faible associée à π.*

Pour l'établir, il suffit de montrer que r_π satisfait les équations 7 et 8 qui définissent l'atteignabilité faible. Elle les satisfait du fait de la définition de l'atteignabilité (équations 1 à 3).

Propriété 4 *Soit π une politique. La relation d'atteignabilité associée r_π est l'unique plus petite relation d'atteignabilité faible associée à π (plus petite au sens de l'inclusion et donc, dans ce cas, de la cardinalité).*

Cette propriété est la conséquence immédiate des propriétés 1, 2 et 3.

2.3 Formulation proposée

En utilisant la notion d'atteignabilité faible, nous proposons la formulation suivante du problème de synthèse de contrôleur.

Variables Cette formulation utilise deux ensembles de variables :

- pour chaque observation $o \in \mathbf{d}(O)$, une variable $\pi(o)$ de domaine $\mathbf{d}(C) \cup \{\perp\}$, représentant la commande à appliquer lorsque o est observée ;
- pour chaque état $s \in \mathbf{d}(S)$, une variable booléenne $wr_\pi(s)$ qui représente le fait que s soit faiblement atteignable ou non.

Contraintes Les contraintes à satisfaire sont définies par les équations suivantes :

$$\forall s \in \mathbf{d}(S), \quad I(s) \rightarrow wr_\pi(s) \tag{9}$$

$$\forall s, s' \in \mathbf{d}(S), \quad (wr_\pi(s) \wedge T(s, \pi(s_{\downarrow O}), s')) \rightarrow wr_\pi(s') \tag{10}$$

$$\forall s \in \mathbf{d}(S), \quad wr_\pi(s) \rightarrow (\pi(s_{\downarrow O}) \neq \perp) \tag{11}$$

$$wr_\pi(s) \rightarrow (\exists s' \in \mathbf{d}(S), T(s, \pi(s_{\downarrow O}), s')) \tag{12}$$

$$wr_\pi(s) \rightarrow (\forall s' \in \mathbf{d}(S), T(s, \pi(s_{\downarrow O}), s') \rightarrow P(s, \pi(s_{\downarrow O}), s')) \tag{13}$$

Les équations 9 et 10 sont de simples copies des équations 7 et 8 qui définissent l'atteignabilité faible. Les équations 11 à 13 sont des copies des équations 4 à 6 qui définissent les exigences sur la politique, où r_π est remplacé par wr_π. Dans l'équation 11, $df_\pi(o)$ est remplacé par la formulation équivalente $(\pi(o) \neq \perp)$. L'équation 10 est finalement simplifiée pour prendre en compte l'équation 11.

Alors que le CSP précédent P (voir la section 2.1) impliquait $|\mathbf{d}(O)| + |\mathbf{d}(S)| + |\mathbf{d}(S)|^2$ variables, ce CSP P' n'en n'implique plus que $|\mathbf{d}(O)| + |\mathbf{d}(S)|$.

Nous allons montrer que, comme P, P' modélise précisément le problème de synthèse de contrôleur, c'est-à-dire que l'approche consistant à résoudre P' pour résoudre le problème de synthèse de contrôleur est correcte, complète et possiblement optimale, dans un sens que nous allons définir.

2.4 Correction, complétude et optimalité

Correction La correction de l'approche résulte de la propriété 2. Supposons en effet que P' admette une solution, faite d'une politique π et d'une relation wr_π. D'après les équations 9 et 10, la relation wr_π est une relation d'atteignabilité faible. Soit r_π la relation d'atteignabilité associée à π. On a d'après la propriété 2 : $\forall s \in \mathbf{d}(S), r_\pi(s) \rightarrow wr_\pi(s)$. La relation r_π satisfait donc les exigences associées aux équations 11 à 13. En conséquence, π est une solution du problème de synthèse de contrôleur.

Complétude Quant à la complétude de l'approche, elle résulte de la propriété 3. Supposons en effet que le problème de synthèse de contrôleur admette une solution, sous la forme d'une politique π et d'une relation

d'atteignabilité associée r_π. D'après la propriété 3, r_π est une relation d'atteignabilité faible. π et r_π constituent donc une solution de P'. P' est donc cohérent et un solveur de contraintes complet produira une solution (celle-ci ou une autre).

Optimalité Il reste que, si P' admet une solution, la politique produite π peut être définie pour des observations non atteignables, qui ne sont associées à aucun état atteignable. Pratiquement, cela ne constitue pas un problème puisque ces observations ne seront jamais atteintes en suivant π. Cependant si, pour des raisons de lisibilité ou de compacité, on désire une politique π qui ne soit définie que pour les observations atteignables (associées à au moins un état atteignable), il suffit de transformer le problème de satisfaction de contraintes en un problème d'optimisation sous contraintes où le critère à minimiser est le nombre d'observations pour lesquelles la politique est définie ($\sum_{o \in \mathbf{d}(O)} (\pi(o) \neq \bot)$). Si l'on obtient une solution optimale avec preuve d'optimalité, on a d'après la propriété 4 la garantie que la relation wr_π produite est la relation d'atteignabilité r_π et que la politique π produite n'est définie que pour les observations atteignables. Sinon, on obtient une politique éventuellement définie pour des observations non atteignables, ce qui reste acceptable.

2.5 Modèle OPL

Nous avons utilisé le langage OPL [11] pour exprimer les modèles associés à divers problèmes de synthèse de contrôleur de systèmes dynamiques. N'importe quel autre langage de programmation par contraintes aurait pu être utilisé. Les modèles OPL générés sont optimisés de façon à limiter autant que possible le nombre de contraintes induites. Ils exploitent en particulier le fait que la relation $T(s, c, s')$ (respectivement $P(s, c, s')$) est habituellement une conjonction de relations $T_1(s)$ (respectivement $P_1(s)$), $T_2(s, c)$ (respectivement $P_2(s, c)$) et $T_3(s, c, s')$ (respectivement $P_3(s, c, s')$). La relation $T_1(s)$ (respectivement $P_1(s)$) exprime les états possibles (respectivement acceptables). La relation $T_2(s, c)$ (respectivement $P_2(s, c)$) exprime les commandes possibles (respectivement acceptables). Finalement, la relation $T_3(s, c, s')$ (respectivement $P_3(s, c, s')$) exprime les transitions possibles (respectivement acceptables). Pour le moment, ces modèles sont définis manuellement, ce qui est potentiellement une source d'erreurs. La construction automatique d'un modèle OPL à partir des ensembles de variables S, O et C et des ensembles de contraintes qui définissent les relations I, T et P est cependant envisageable.

Sur l'exemple jouet de la section 1.3, le modèle OPL génère 112 variables et 980 contraintes. Il est résolu par l'outil *CP Optimizer* associé à OPL en moins d'1/100 de seconde. La politique produite est la suivante :

$$\begin{aligned} w_N \wedge w_S \wedge \neg w_E \wedge \neg w_W &: \; m = m_W \\ w_N \wedge \neg w_S \wedge \neg w_E \wedge w_W &: \; m = m_S \\ \neg w_N \wedge w_S \wedge \neg w_E \wedge w_W &: \; m = m_N \end{aligned}$$

Chaque ligne est associée à une observation faiblement atteignable. Sur chaque ligne, l'observation apparaît avant les deux points et la commande associée après. Par exemple, la première ligne spécifie que, si des murs sont observés au nord et au sud, mais ni à l'est, ni à l'ouest, le robot doit se déplacer vers l'ouest. Cette politique consiste à se déplacer vers l'ouest, puis à alterner mouvements vers le nord et vers le sud. Dans ce cas, la politique produite sans optimisation explicite est par chance optimale : elle est définie uniquement pour les observations atteignables.

3 Un exemple réel

Pour valider l'approche proposée, nous l'avons mise en œuvre sur un exemple réel de contrôle d'un sous-système d'un satellite, précédemment introduit dans [16].

Le contexte est un satellite de surveillance de la Terre dont la mission est de détecter des points chauds à la surface de la Terre, dus à des feux de forêt ou à des éruptions volcaniques [6]. Il est équipé d'un instrument de détection à large fauchée et d'un instrument d'observation à fauchée réduite. En cas de détection de point chaud, une alarme doit être adressée au sol via un satellite géostationnaire relais et des observations de la zone doivent être réalisées. Les données d'observation sont ensuite télé-déchargées vers des stations sol lors de passages en visibilité. Dans ce contexte, nous nous intéressons à un équipement appelé DSP (*Digital Signal Processor*) en charge de l'analyse des images produites par l'instrument de détection et de la détection de points chauds dans ces images. Le DSP est composé de trois éléments :

— un analyseur, en charge de l'analyse des images ;
— un circuit, qui fournit le courant nécessaire à l'analyseur ;
— un interrupteur, qui permet d'activer ou désactiver la tension aux bornes du circuit.

Le module de contrôle du DSP reçoit des requêtes de mise ON ou OFF en provenance du module de contrôle de la fonction de détection. À chaque étape, il produit différentes sorties (voir la figure 4) :

— une commande de l'interrupteur du DSP ;

— des signaux vers le module de contrôle de la détection, indiquant si le DSP fonctionne correctement ou non ;
— un signal vers le module de contrôle de l'alarme, indiquant si, oui ou non, un point chaud a été détecté.

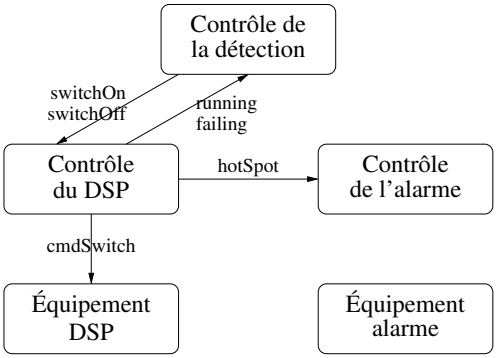

FIGURE 4 – Entrées et sorties du module en charge de contrôler le DSP

Chacun des trois éléments constituant le DSP est susceptible de tomber en panne. Si l'analyseur n'est pas en panne et reçoit du courant, la détection de points chauds est supposée fonctionner correctement. S'il est en panne ou ne reçoit pas de courant, la détection ne fonctionne pas. Si le circuit est en panne, l'analyseur ne reçoit pas de courant. Si l'interrupteur est en panne, la tension aux bornes du circuit n'est pas forcément conforme à la commande de l'interrupteur.

Informellement, les propriétés de sûreté de plus haut niveau qui doivent être satisfaites à chaque étape sont les suivantes : en absence de pannes, si le DSP est ON, la détection doit être correcte (détection de point chaud en cas de point chaud ; pas de détection, sinon) ; s'il est OFF, il ne doit rien détecter.

Variables d'état Pour modéliser ce problème, nous utilisons les variables d'état suivantes (ensemble S) :
— *switchOn* $\in \{0,1\}$: présence ou non d'une requête de mise ON : 1 pour présence ;
— *switchOff* $\in \{0,1\}$: présence ou non d'une requête de mise OFF : 1 pour présence ;
— *switched* $\in \{0,1\}$: dernière requête de mise ON ou OFF reçue par le contrôleur (mémoire du contrôleur) : 1 pour ON, 0 pour OFF ;
— *tension* $\in \{0,1\}$: présence ou non de tension aux bornes du circuit : 1 pour présence ;
— *current* $\in \{0,1\}$: présence ou non de courant dans le circuit : 1 pour présence ;
— *faultAnalyzer* $\in \{0,1\}$: panne ou non de l'analyseur : 1 pour panne ;
— *faultCircuit* $\in \{0,1\}$: panne ou non du circuit : 1 pour panne ;
— *faultSwitch* $\in \{0,1\}$: panne ou non de l'interrupteur : 1 pour panne ;
— *inputIm* $\in \{NOIM, NORM, HOT\}$: nature de l'image en entrée de l'analyseur : *NOIM* en cas d'absence d'image, *NORM* pour une image normale sans point chaud, *HOT* pour une image avec point chaud ; pour le contrôle du DSP, on ne s'intéresse pas au contenu des images, par exemple, à la position géographique des points chauds ; on s'intéresse uniquement à la présence ou non de points chauds ;
— *resultAnal* $\in \{NOIM, NORM, HOT\}$: résultat de l'analyse.

Parmi ces variables, seules les variables *switchOn*, *switchOff*, *switched*, *tension*, *current* et *resultAnal* sont observables par le contrôleur (ensemble O). Les pannes, ainsi que la nature réelle de l'image en entrée de l'analyseur, ne sont pas connues du contrôleur.

Variables de commande Nous utilisons par ailleurs les variables de commande suivantes (ensemble C) :
— *cmdSwitch* $\in \{0,1\}$: commande de l'interrupteur : 1 pour ON, 0 pour OFF ;
— *cmdMemory* $\in \{0,1\}$: mise à jour de la mémoire du contrôleur : 1 pour ON, 0 pour OFF ;
— *running* $\in \{0,1\}$: information de bon fonctionnement du DSP : 1 pour correct ;
— *failing* $\in \{0,1\}$: information de mauvais fonctionnement du DSP : 1 pour incorrect ;
— *hotSpot* $\in \{0,1\}$: information de détection de point chaud : 1 pour détection.

Graphe de dépendance Pour structurer l'écriture du modèle (relations I, T et P), il peut être utile de construire le graphe qui représente les dépendances entre variables d'état et de commande. La figure 5 montre ce graphe dans un cadre de représentation proche du langage graphique utilisé par l'outil SCADE [7]. Les boites marquées "pre" représentent l'accès à la valeur précédente d'une variable d'état. On y voit par exemple que la tension courante dépend de la tension précédente, de la commande de l'interrupteur et de son état (panne ou non).

États initiaux possibles Les états initiaux possibles (ensemble I) sont définies par les contraintes suivantes sur les variables d'état : $\neg switchOn$, $\neg switchOff$, $\neg switched$, $\neg current$, $\neg tension$, $(resultAnal = NOIM)$, $(inputIm = NOIM)$. Des pannes sont donc possibles dans l'état initial.

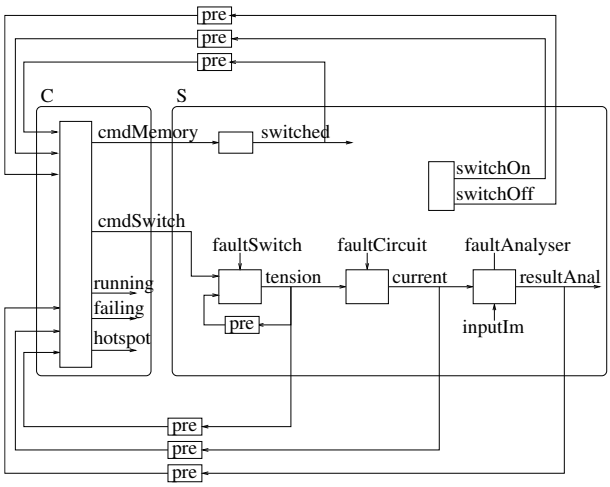

FIGURE 5 – Graphe de dépendances entre variables d'état et de commande

Transitions possibles Les transitions possibles (ensemble T) sont définies par les contraintes suivantes sur les variables d'état et de commande :

Contraintes sur les modules de plus haut niveau :
$\neg(switchOn \wedge switchOff)$

Contraintes sur la mémoire du contrôleur :
$switched' = cmdMemory$

Contraintes sur l'interrupteur :
$\neg tension \wedge \neg cmdSwitch \rightarrow \neg tension'$
$tension \wedge cmdSwitch \rightarrow tension'$
$\neg faultSwitch \rightarrow (tension' = cmdSwitch)$

Contraintes sur le circuit :
$current = (tension \wedge \neg faultCircuit)$

Contraintes sur l'analyseur :
$(\neg current \vee faultAnalyzer)$
$\quad \rightarrow (resultAnal = NOIM)$
$(current \wedge \neg faultAnalyzer)$
$\quad \rightarrow (resultAnal = inputIm)$

Rappelons que, pour chaque variable d'état x, x' représente sa valeur à l'étape suivante. Les contraintes sur les modules de plus haut niveau traduisent l'hypothèse qu'il ne peut pas y avoir simultanément une requête de mise ON et une autre de mise OFF. Les contraintes sur la mémoire du contrôleur expriment que la mémoire est toujours correctement mise à jour. Les contraintes sur l'interrupteur expriment qu'en l'absence de panne de l'interrupteur, la tension est conforme à la commande de l'interrupteur. Celles sur le circuit expriment qu'il y a du courant si et seulement si il y a de la tension et pas de panne du circuit. Enfin, celles sur l'analyseur expriment qu'en l'absence de panne de l'analyseur, le résultat de l'analyse est conforme à la nature de l'image en entrée.

Transitions acceptables Les transitions acceptables (ensemble P) sont définies par les contraintes suivantes sur les variables d'état et de commande :

Exigences sur la mise à jour de la mémoire du contrôleur :
$switchOn \rightarrow cmdMemory$
$switchOff \rightarrow \neg cmdMemory$
$(\neg switchOn \wedge \neg switchOff)$
$\quad \rightarrow (cmdMemory = switched)$

Exigences sur la détection de point chaud :
$hotSpot = (running \wedge (resultAnal = HOT))$

Exigences sur la détection de pannes :
$running = (switched \wedge (resultAnal \neq NOIM))$
$failing = (switched \wedge (resultAnal = NOIM)) \vee$
$\quad (tension \neq switched)) \vee (current \neq tension))$
$\neg(running \wedge failing)$
$switched \rightarrow (running \vee failing)$

Exigences de plus haut niveau :
$((inputIm = HOT) \wedge running) \rightarrow hotSpot$
$hotSpot \rightarrow (inputIm = HOT)$
$(\neg faultSwitch \wedge \neg faultCircuit' \wedge \neg faultAnalyser'$
$\quad \wedge switched') \rightarrow (resultAnal' = inputIm')$
$(\neg faultSwitch \wedge \neg switched')$
$\quad \rightarrow (resultAnal' = NOIM)$

Par exemple, les contraintes sur la détection de point chaud imposent qu'un signal de détection est émis si et seulement si le DSP fonctionne correctement et si le résultat de l'analyse indique la présence de point chaud. Les contraintes sur la détection de pannes imposent que le contrôleur du DSP n'indique pas simultanément qu'il est en bon et en mauvais fonctionnement

et que, si le DSP est ON, il indique l'un ou l'autre. Les exigences de plus haut niveau imposent que, si l'image en entrée contient un point chaud et si le DSP fonctionne correctement, alors un signal de détection est émis. Inversement, elles imposent que si un signal de détection est émis, l'image en entrée contient effectivement un point chaud. De plus, elles imposent qu'en l'absence de panne, si le DSP est ON, le résultat de l'analyse est conforme à la nature de l'image en entrée. Par contre, elles imposent qu'en l'absence de panne de l'interrupteur, si le DSP est OFF, il n'y a pas de résultat d'analyse.

Le modèle OPL associé génère 2356 variables et 512175 contraintes. Il est résolu par *CP Optimizer* en 3,39 secondes grâce à la propagation initiale qui affecte 2338 variables (résultats obtenus sur une station de travail SUN Ultra 45 fonctionnant sous Unix et utilisant un processeur de 1.6 GHz et 1 GB de RAM). Comme avec l'exemple jouet de la section 1.3, la politique produite sans optimisation explicite est par chance optimale : elle est définie uniquement pour les observations atteignables.

4 Forces et faiblesses de l'approche proposée

Dans ce papier, nous avons proposé une formulation du problème de synthèse de contrôleur pour un système dynamique comme un pur problème de satisfaction (optimisation) de contraintes. C'est à notre connaissance la première fois qu'une telle formulation est proposée. Au delà des applications classiques de la programmation par contraintes à la planification et à l'ordonnancement, ce travail ouvre la voie à l'application de la programmation par contraintes au vaste domaine du contrôle de systèmes dynamiques à évènements discrets. Il faut souligner que la formulation proposée peut être utilisée pour synthétiser ou pour vérifier des contrôleurs : la vérification est plus simple que la synthèse parce que la politique est connue en cas de vérification, alors qu'elle est inconnue en cas de synthèse.

L'approche proposée est à rapprocher de la formulation du problème de synthèse d'une politique optimale pour un MDP (*Markov Decision Process* [18]) comme un pur problème de programmation linéaire. En effet, dans cette formulation, une variable est associée à chaque état, représentant la valeur optimale qu'il est possible d'obtenir à partir de cet état. Les équations d'optimalité de Bellman [2] se traduisent par un ensemble de contraintes linéaires : une par paire état-commande. Le critère à minimiser est la somme des variables. L'atteignabilité n'est cependant pas pris en compte : tous les états sont supposés atteignables à toute étape. L'approche que nous avons proposée peut être vue comme l'équivalent "logique" de cette approche de résolution des MDP.

Le principal avantage de l'approche proposée est qu'elle permet, pour la résolution, de se reposer entièrement sur les outils génériques existants de programmation par contraintes. Quand les modèles de programmation par contraintes seront construits automatiquement à partir de la définition du problème (ensembles de variables S, O et C et ensembles de contraintes définissant les relations I, T et P), seule la définition du problème devra être modifiée pour passer d'un système dynamique à un autre.

Son défaut principal est le nombre de variables et de contraintes générées (au moins une variable par état et une contrainte par paire d'états). Comme le nombre d'états possibles est une fonction exponentielle du nombre de variables d'état (à ne pas confondre avec les variables CSP), l'approche ne pourra être mise en œuvre que sur les problèmes impliquant un nombre relativement faible de variables d'état (de l'ordre de la dizaine, comme dans l'exemple du DSP). On notera cependant que les algorithmes de synthèse de contrôleurs développés dans le domaine de l'automatique classique [9] produisent des résultats pour des systèmes dynamiques continus ayant un nombre de variables d'état du même ordre. D'autres approches à base de contraintes, plus économes en termes de variables et de contraintes, devraient cependant être explorées. L'approche précédemment proposée dans [13] est beaucoup plus économe en termes de variables et de contraintes. Malheureusement, elle est correcte, mais incomplète : le problème généré est sur-contraint par rapport au problème originel.

Finalement, dans ce papier, nous nous sommes focalisés sur la modélisation et nous ne nous sommes pas vraiment intéressés à l'efficacité de la résolution. Des comparaisons expérimentales en termes d'efficacité avec des algorithmes génériques dédiés à la synthèse de contrôleur devraient être menées.

Remerciements

Ce travail a été mené dans le cadre du projet français CNES-ONERA AGATA (Architecture Générique pour l'Autonomie : Tests et Applications ; voir `http://www.agata.fr`) dont l'objectif est de développer des techniques permettant d'accroître l'autonomie des systèmes spatiaux. Nous tenons à remercier Michel Lemaître pour ses précieuses contributions.

Références

[1] A. Arnold, A. Vincent, and I. Walukiewicz. Games for Synthesis of Controllers with Partial Information. *Theoretical Computer Science*, 303(1) :7–34, 2003.

[2] R. Bellman. *Dynamic Programming.* Princeton University Press, 1957.

[3] P. Bertoli, A. Cimatti, M. Roveri, and P. Traverso. Planning in Nondeterministic Domains under Partial Observability via Symbolic Model Checking. In *Proc. of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 473–478, Seattle, WA, USA, 2001.

[4] B. Bonet, H. Palacios, and H. Geffner. Automatic Derivation of Memoryless Policies and Finite-State Controllers Using Classical Planners. In *Proc. of the 19th International Conference on Automated Planning and Scheduling (ICAPS-09)*, Thessaloniki, Greece, 2009.

[5] E. Clarke, O. Grumberg, and D. Peled. *Model Checking.* MIT Press, 1999.

[6] S. Damiani, G. Verfaillie, and M.-C. Charmeau. An Earth Watching Satellite Constellation : How to Manage a Team of Watching Agents with Limited Communications. In *Proc. of the 4th Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-05)*, pages 455–462, Utrecht, The Netherlands, 2005.

[7] Esterel Technologies. SCADE. http://www.esterel-technologies.com/.

[8] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning : Theory and Practice.* Morgan Kaufmann, 2004.

[9] F. Golnaraghi and B. Kuo. *Automatic Control Systems.* John Wiley & Sons, 2009.

[10] N. Halbwachs. *Synchronous Programming of Reactive Systems.* Kluwer, 1993.

[11] IBM ILOG. Cplex optimization studio. http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/.

[12] B. Jobstmann, S. Galler, M. Weiglhofer, and R. Bloem. Anzu : A Tool for Property Synthesis. In *Proc. of the 19th International Conference on Computer Aided Verification (CAV-07)*, pages 258–262, Berlin, Germany, 2007.

[13] M. Lemaître, G. Verfaillie, C. Pralet, and G. Infantes. Synthèse de contrôleur simplement valide dans le cadre de la programmation par contraintes. In *Actes des Journées Françaises sur la Planification, la Décision et l'Apprentissage pour la Conduite de Systèmes (JFPDA-10)*, Besançon, France, 2010.

[14] MBP Team. Model Based Planner. http://sra.itc.it/tools/mbp/.

[15] A. Pnueli and R. Rosner. On the Synthesis of a Reactive Module. In *Proc. of the ACM Symposium on Principles of Programming Languages (POPL-89)*, pages 179–190, 1989.

[16] C. Pralet, M. Lemaître, G. Verfaillie, and G. Infantes. Synthesizing Controllers for Autonomous Systems : A Constraint-based Approach. In *Proc. of the 10th International Symposium on Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS-10)*, Sapporo, Japan, 2010.

[17] C. Pralet, G. Verfaillie, M. Lemaître, and G. Infantes. Constraint-based Controller Synthesis in Non-deterministic and Partially Observable Domains. In *Proc. of the 19th European Conference on Artificial Intelligence (ECAI-10)*, pages 681–686, Lisbon, Portugal, 2010.

[18] M. Puterman. *Markov Decision Processes, Discrete Stochastic Dynamic Programming.* John Wiley & Sons, 1994.

[19] P. Ramadge and W. Wonham. The Control of Discrete Event Systems. *Proc. of the IEEE*, 77(1) :81–98, 1989.

[20] RATSY Team. RATSY : Requirements Analysis Tool with Synthesis. http://rat.fbk.eu/ratsy/.

[21] F. Rossi, P. Van Beek, and T. Walsh, editors. *Handbook of Constraint Programming.* Elsevier, 2006.