Modélisation à base de contraintes de systèmes dynamiques à événements discrets

Gérard Verfaillie, Cédric Pralet, Michel Lemaître

Modélisation à base de contraintes de systèmes dynamiques à événements discrets

Gérard Verfaillie, Cédric Pralet, Michel Lemaître

ONERA/DCSD, 2 av. Édouard Belin, BP 4025, 31055 Toulouse Cédex 4

{Prenom.Nom}@onera.fr

Résumé

De nombreux cadres dédiés à la modélisation de systèmes dynamiques à événements discrets ont été proposés pour répondre à des tâches de programmation, de simulation, de validation, de suivi de situation ou de décision (automates, réseaux de Petri, chaînes de Markov, logique temporelle, calcul des situations, STRIPS ...). Tous ces cadres présentent des similitudes importantes, mais aucun n'offre la souplesse de cadres plus généraux tels que la logique ou les contraintes.

Dans cet article, nous proposons un cadre générique de modélisation de systèmes dynamiques à événements discrets dont les principaux composants sont des chronogrammes d'état et d'événement et des contraintes sur ces chronogrammes. Bien que n'importe quel type de contrainte puisse être défini sur des chronogrammes, nous nous focalisons sur des contraintes *a priori* utiles : contraintes temporelles pures, contraintes d'état ou d'événement instantané, contraintes de transition instantanée ou non.

Finalement, nous montrons comment le cadre proposé englobe des cadres apparemment différents tels que les automates, les réseaux de Petri ou les cadres classiques utilisés en planification ou en ordonnancement, tout en offrant la très grande souplesse d'une modélisation à base de contraintes.

Abstract

Numerous frameworks dedicated to the modelling of discrete event dynamic systems have been proposed to deal with programming, simulation, validation, situation tracking, or decision tasks (automata, Petri nets, Markov chains, temporal logic, situation calculus, STRIPS ...). All these frameworks present significant similarities, but none offers the flexibility of more generic frameworks such as logics or constraints.

In this article, we propose a generic framework for the modelling of discrete event dynamic systems, whose main components are state and event timelines and constraints on these timelines. Although any kind of constraint can be defined on timelines, we focus on some useful ones : pure temporal constraints, instantaneous state and event constraints, instantaneous and non instantaneous transition constraints.

Finally we show how the proposed framework subsumes existing apparently different frameworks such as automata, Petri nets, or classical frameworks used in planning or scheduling, while offering the great flexibility of a constraint-based modelling.

1 Introduction

Le but de cet article est de proposer un *cadre générique à base de contraintes* pour la modélisation de *systèmes dynamiques à événements discrets*, c'est-à-dire de systèmes dont l'état évolue dans le temps grâce à des *changements instantanés* éventuellement provoqués par des *événements instantanés*.

De nombreux cadres permettent de modéliser de tels systèmes, parmi lesquels on peut citer les *automates*, les *langages synchrones* [4] qui permettent de décrire des automates et la *logique temporelle* [16] qui permet d'exprimer des propriétés sur des automates, mais aussi les *réseaux de Petri*, les *chaînes de Markov* et les *processus décisionnels markoviens* [18] qui permettent toux deux de décrire des évolutions stochastiques, le cadre *STRIPS* [8] et le *calcul des situations* [14] tous deux utilisés en planification et finalement les modèles classiques utilisés en ordonnancement.

Bien que tous ces cadres présentent des similitudes importantes (*instants* de transition discrets, représentation plus ou moins compacte des *états* et des *transitions*), il est difficile de les comparer à moins de les traduire tous dans les cadres les plus basiques et les moins compacts que sont les automates et les chaînes de Markov.

D'un autre côté, bien que la *modélisation à base de*

contraintes soit connue pour combiner *compacité* et *flexibilité* en termes de modélisation avec *efficacité* en termes de résolution, elle reste principalement utilisée pour traiter des problèmes *statiques*, c'est-à-dire des problèmes qui n'impliquent pas le *temps*. On notera quelques exceptions : principalement les problèmes d'*ordonnancement* et dans une certaine mesure les problèmes de *planification* (voir par exemple [11, 20]). Mais, à de rares exceptions près (voir par exemple [7]), la modélisation à base de contraintes n'est pas utilisée pour traiter des problèmes de *validation* de systèmes dynamiques ou des problèmes de *suivi de situation*, tels que les problèmes de *diagnostic de pannes*. Nous pensons qu'une telle situation est principalement due à l'absence d'un cadre générique à base de contraintes, dédié à la modélisation de systèmes dynamiques à événements discrets et utilisable indifféremment pour des tâches de simulation, de validation, de suivi de situation ou de décision.

C'est un cadre de cette nature que nous proposons dans cet article. Il est basé sur l'hypothèse d'un *temps continu* et d'*instants discrets* d'événement et de changement, ainsi que sur la notion de *chronogramme* : *chronogrammes d'état* pour représenter la façon dont l'état du système évolue dans le temps et *chronogrammes d'événement* pour représenter la façon dont les événements surviennent. Ces chronogrammes peuvent être représentés de façon compacte grâce à des *variables* : *variables temporelles* pour représenter les instants de changement ou d'événement et *variables atemporelles* pour représenter les valeurs à ces instants. Du fait de la grande souplesse d'une modélisation à base de contraintes, n'importe quel type de contrainte peut être défini sur les chronogrammes grâce à des contraintes sur les variables temporelles ou atemporelles. Cependant, parmi ces contraintes, les contraintes *temporelles pures*, les contraintes d'*état* ou d'*événement instantané* et les contraintes de *transition instantanée ou non* sont *a priori* très utiles et méritent une attention particulière.

Le cadre proposé dans cet article est inspiré mais différent de travaux menés à la frontière entre planification et ordonnancement, où la notion de *chronogramme* est utilisée pour représenter la façon dont l'état et les ressources évoluent dans le temps et pour raisonner sur le temps, l'état et les ressources [13, 15, 3, 9].

La section 2 introduit les hypothèses de base relatives au *temps*, aux *états* et aux *événements*. La section 3 introduit la représentation à base de *chronogrammes*, tandis que la section 4 définit ce qu'est un *réseau de contraintes sur des chronogrammes* et que la section 5 se focalise sur des types de contrainte *a priori* très utiles. Dans la section 6, nous montrons comment le cadre proposé englobe les automates, les réseaux de Petri et les cadres classiques utilisés en planification et en ordonnancement. La section 7 conclue avec le travail restant à faire et diverses extensions possibles.

Cet article traite essentiellement de questions de *modélisation* et ne dit rien sur les questions *algorithmiques* (propagation de contraintes, recherche, ...) qui seront le sujet de futurs études et articles. Nous faisons ce choix parce que nous pensons que le premier et peut-être le principal obstacle à l'utilisation systématique du raisonnement à base de contraintes dans le contexte des systèmes dynamiques à événements discrets est relatif à la *modélisation*.

2 Hypothèses de modélisation

2.1 Temps

Nous voulons raisonner sur des instants, sur l'*ordre* entre ces instants, mais aussi sur leurs *positions* dans le temps. Ces positions sont supposées prendre leurs valeurs dans un ensemble *continu*. C'est pourquoi nous utilisons $\mathbb{R}$, avec l'ordre naturel sur les réels, pour modéliser le temps.

2.2 États et changements d'état

États Nous supposons que l'état d'un système peut être modélisé grâce à un ensemble fini de *variables d'état*, représentant chacune un attribut de l'état du système. À chaque variable d'état, est associé un *domaine* de valeurs qui peut être fini ou infini, discret ou continu, symbolique ou numérique. Dans ces conditions, l'état du système à un instant est modélisé par une affectation à chaque variable d'état d'une valeur de son domaine.

On notera que les variables d'état peuvent être utilisées pour représenter aussi bien des attributs *passifs* de l'état (comme, pour un robot, sa position ou son niveau d'énergie disponible) que des attributs *actifs* (comme, toujours pour un robot, le mode d'un instrument d'observation ou le fait que le robot effectue un certain mouvement). En d'autres termes, les variables d'état peuvent être utilisées pour représenter aussi bien ce qu'on appelle habituellement l'état du système (position, niveau d'énergie, ...) que ce qu'on appelle habituellement les actions, quand elles ne sont pas instantanées (une observation, un mouvement, ...).

Changements d'état Nous supposons que l'état d'un système ne peut évoluer que grâce à des *changements instantanés*. Des changements continus ne peuvent donc pas être modélisés de façon précise, mais seulement grossièrement approchés par une séquence de

changements instantanés. Dans ces conditions, un changement de l'état du système est modélisé par un changement simultané de l'affectation d'un sous-ensemble non vide des variables d'état. De plus, nous adoptons la convention que, quand l'affectation d'une variable d'état v change à l'instant t de la valeur val à la valeur val', l'affectation de v est val avant t, t exclu, et val' après t, t inclus.[1]

Les changements d'état peuvent se produire à tout instant, mais nous supposons que les instants auxquels ils se produisent forment un sous-ensemble *discret* de $\mathbb{R}$. En conséquence, l'affectation d'une variable d'état reste constante d'un instant de changement t au prochain instant de changement $t' > t$, c'est-à-dire égale à la valeur qu'elle a prise à t sur l'intervalle semi-ouvert $[t, t'[$ (voir la figure 1).

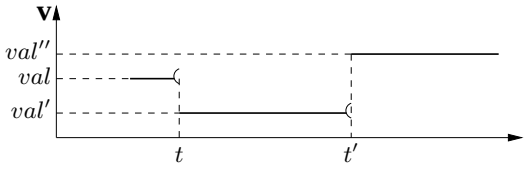

FIG. 1 – Évolution temporelle d'une variable d'état.

2.3 Événements et occurrences d'événement

Événements De la même façon que nous avons supposé que l'état d'un système peut être modélisé grâce à un ensemble fini de variables d'état, nous supposons que les événements qui sont susceptibles de se produire peuvent être modélisés grâce à un ensemble fini de *variables d'événement*. À chaque variable d'événement, est associé un *domaine* de valeurs qui peut être fini ou infini, discret ou continu, symbolique ou numérique. À ce domaine, nous ajoutons systématiquement une valeur *rien* ($\perp$) permettant de représenter l'absence de valeur. Dans ces conditions, à chaque instant, l'ensemble des événements est modélisé par une affectation à chaque variable d'événement d'une valeur de son domaine, éventuellement égale à $\perp$.

Il est par exemple possible d'associer une variable d'événement à chaque type d'événement, avec la valeur $\perp$ désignant l'absence d'événement de ce type et une valeur différente de $\perp$ désignant à la fois la présence d'un événement de ce type et son contenu.

Occurrences d'événement Nous supposons que les événements sont des *phénomènes instantanés*. Dans ces conditions, une occurrence d'événement est modélisée par une affectation simultanée d'une valeur différente de $\perp$ à un sous-ensemble non vide des variables d'événement.

Les événements peuvent se produire à tout instant, mais nous supposons que les instants auxquels ils se produisent forment un sous-ensemble *discret* de $\mathbb{R}$. En conséquence, l'affectation d'une variable d'événement reste égale à $\perp$ entre deux instants successifs d'événement t et $t' > t$, t et t' exclus, c'est-à-dire sur l'intervalle ouvert $]t, t'[$ (voir la figure 2).

FIG. 2 – Évolution temporelle d'une variable d'événement.

2.4 Changements d'état et occurrences d'événement

Aucune hypothèse *a priori* n'est faite sur une quelconque relation de *corrélation* ou de *causalité* entre changements d'état et occurrences d'événement. Changements d'état et occurrences d'événement peuvent être simultanés. Des changements d'état peuvent se produire sans événement et des événements sans changement d'état.

3 Chronogrammes

Dans cette section, nous montrons comment des *chronogrammes* peuvent être utilisés pour représenter de façon compacte l'évolution temporelle de variables d'état ou d'événement.[2]

Définition 1 *Un chronogramme tl est un quintuple* $\langle v, d, I, t_I, v_I \rangle$ *où* v *est une variable d'état ou d'événement,* d *son domaine de valeurs,* I *une séquence d'instants,* t_I *une séquence de variables temporelles et* v_I *une séquence de variables atemporelles.*

Si v *est une variable d'état, on parle de chronogramme d'état. Sinon, on parle de chronogramme d'événement.*

La séquence I *est supposé finie ou dénombrable.*[3] *Soit* $I = [1, \ldots i, \ldots]$, $I^+ = [0, 1, \ldots i, \ldots]$ *et* $I^- = [2, \ldots i, \ldots]$. *La séquence* t_I *associe à chaque instant*

[1] Cette convention, utilisée par exemple dans les *langages synchrones*, est très utile pour modéliser des événements instantanés qui entraînent des changements instantanés au même instant, par exemple une panne qui conduit instantanément le système dans un mode de défaillance.

[2] *Chronogramme* est la traduction choisie pour le terme anglais *timeline*. Selon *Wikipédia*, un chronogramme est une représentation graphique de l'évolution temporelle d'un signal ou d'un état.

[3] Un ensemble est dénombrable si et seulement si il est équipotent à l'ensemble $\mathbb{N}$ des entiers naturels.

$i \in I$ une variable temporelle t_i de domaine $\mathbb{R}$ qui représente la position temporelle de l'instant i. La séquence v_I associe à chaque instant $i \in I^+$ une variable atemporelle v_i de domaine d qui représente la valeur de v à l'instant i.

Les instants sont temporellement ordonnés. Nous imposons donc que $\forall i \in I^-$, $t_{i-1} \leq t_i$ et $(t_i = t_{i-1}) \rightarrow (v_i = v_{i-1})$. De plus, dans le cas d'un chronogramme d'événement, nous imposons que $v_0 = \perp$.

Un chronogramme $tl = \langle v, d, I, t_I, v_I \rangle$ représente l'évolution temporelle de la variable v. La séquence I est la séquence des instants auxquels des changements ou des événements peuvent se produire (ils peuvent se produire, mais ne sont pas obligatoires). t_I est la séquence de leurs positions temporelles et v_I la séquence des valeurs de v à ces instants. Le premier instant de la séquence (0) est fictif et n'a pas de position temporelle. Il est utilisé pour représenter la *valeur initiale* de v, égale à $\perp$ dans le cas d'un chronogramme d'événement (pas d'événement à l'instant initial). La figure 3 montre la représentation tabulaire d'un chronogramme.

	0	1	...	i	...
t		t_1	...	t_i	...
v	v_0	v_1	...	v_i	...

FIG. 3 – Représentation tabulaire d'un chronogramme.

Il est important de ne pas confondre la séquence I des *instants* et la séquence t_I de leurs *positions temporelles*. Il est aussi important de ne pas confondre les variables d'*état* et d'*événement* et les variables *temporelles* et *atemporelles* qui apparaissent dans les chronogrammes. Seules ces dernières sont réellement des *variables mathématiques*. Les premières sont des *fonctions du temps*. Quand la confusion sera possible, nous réserverons le terme *variable* aux variables temporelles et atemporelles et utiliserons le terme *chronogramme* pour les variables d'état et d'événement. De plus, quand aucune confusion ne sera possible, nous parlerons indifféremment de v et de tl, ne faisant aucune distinction entre une variable d'état ou d'événement et le chronogramme associé.

Un chronogramme $tl = \langle v, d, I, t_I, v_I \rangle$ est dit *fini* si I est fini. Il est dit complètement affecté si toutes les variables temporelles et atemporelles de t_I et v_I sont affectées. Soit $tl = \langle v, d, I, t_I, v_I \rangle$ un chronogramme fini et complètement affecté, avec $I = [1, \ldots i, \ldots l]$. l est sa *longueur* et l'intervalle fermé $H = [t_1, t_l]$ son *horizon temporel*.

Compte-tenu des hypothèses de la section 2 (une variable d'état reste constante et une variable d'événement reste égale à $\perp$ entre deux instants successifs de changement ou d'événement), il est facile d'associer à tout chronogramme $tl = \langle v, d, I, t_I, v_I \rangle$ fini et complètement affecté la fonction qui associe à tout $t \in H$ (et pas seulement à tout $t \in t_I$) la valeur de v à t (v_i avec $i = max\{i' \in I \,/\, t_{i'} \leq t\}$), mais aussi sa valeur juste avant t_1 (v_0 dans le cas d'un chronogramme d'état et $\perp$ dans le cas d'un chronogramme d'événement) et juste après t_l (v_l dans le cas d'un chronogramme d'état et $\perp$ dans le cas d'un chronogramme d'événement). La figure 4 montre une représentation graphique partielle de cette fonction : fonction *constante par morceaux* dans le cas d'un chronogramme d'état et fonction *dirac multiple* dans le cas d'un chronogramme d'événement.

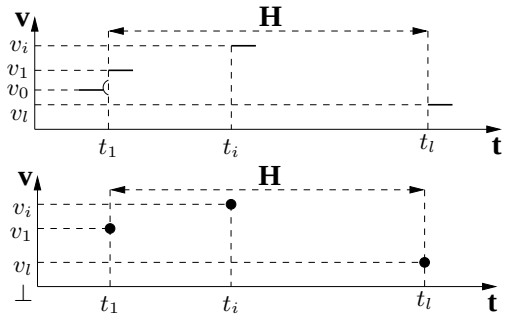

FIG. 4 – Fonctions du temps, associées à un chronogramme fini et complètement affecté de longueur l, dans le cas d'un chronogramme d'état (haut) et d'un chronogramme d'événement (bas).

4 Réseaux de contraintes sur des chronogrammes

4.1 Définition d'un réseau de contraintes

Dans cette section, nous montrons comment des *contraintes* peuvent être définies sur des chronogrammes pour représenter les évolutions combinées des variables d'état et d'événement qui sont *possibles* ou *requises*.

Définition 2 *Un réseau de contraintes sur des chronogrammes (CNT) est une paire $\langle TL, C \rangle$ où :* [4]

- *TL est un ensemble fini de chronogrammes qui partagent tous la la même séquence I d'instants et la même séquence t_I de leurs positions temporelles ;*
- *C est un ensemble fini de contraintes sur les chronogrammes de TL (voir la définition 3).*

On note $V = \{v,\ \langle v, d, I, t_I, v_I \rangle \in TL\}$, $\forall i \in I^+$, $V_i = \{v_i, v \in V\}$, $V_I = [V_i, i \in I^+]$. SV, SV_i et SV_I (respectivement EV, EV_i et EV_I) peuvent être définis de façon similaire en se restreignant aux chrono-

[4] CNT pour *Constraint Network on Timelines*.

grammes d'état (respectivement d'événement). Finalement, on note $Var = t_I \cup V_I$.

t_I est l'ensemble des variables temporelles du CNT, V_I l'ensemble des variables atemporelles et Var l'ensemble des variables, temporelles ou atemporelles. De la même façon que pour les chronogrammes, on peut définir ce qu'est un CNT *fini* et un CNT *complètement affecté*.

Définition 3 *Une contrainte c sur un ensemble TL de chronogrammes est un triplet $\langle qt, sc, df\rangle$ où :*
- *qt est une séquence finie $[q_1, \dots q_j, \dots q_m]$ de quantificateurs, avec $q_j \in \{\forall, \exists\}$;*
- *sc est une fonction qui associe à chaque séquence $[i_1, \dots i_j, \dots i_m] \in I^m$ la portée d'une contrainte basique $sc(i_1, \dots i_m)$, c'est-à-dire une séquence finie de variables de Var ;*
- *df est une fonction qui associe à chaque séquence $[i_1, \dots i_j, \dots i_m] \in I^m$ la définition d'une contrainte basique $df(i_1, \dots i_m)$, c'est-à-dire une fonction booléenne sur le produit cartésien des domaines des variables de $sc(i_1, \dots i_m)$.*

Si $m = 0$, $qt = \emptyset$, sc est la portée d'une contrainte basique et df la définition associée.

Une *contrainte basique* est une contrainte CSP classique, définie par sa *portée* sc, qui est une séquence finie de variables, et sa *définition* df, qui est une fonction booléenne sur le produit cartésien des domaines des variables de sc [19]. La *quantification* qt est utilisée pour spécifier en une contrainte non basique un ensemble éventuellement infini de contraintes basiques, en itérant sur I qui est éventuellement infini. Les fonctions sc et df sont utilisées pour associer une contrainte basique, c'est à dire une portée $sc(i_1, \dots i_m)$ et une définition $df(i_1, \dots i_m)$, à toute séquence $[i_1, \dots i_m] \in I^m$. Les *portées* peuvent être spécifiées en extension quand I est fini ou $m = 0$. Autrement, elles doivent être spécifiées en intension. Les *définitions* peuvent être spécifiées en extension quand I est fini ou $m = 0$ et quand les domaines des variables impliquées sont finis. Autrement, elles doivent être spécifiées en intension. Pour toute séquence $[i_1, \dots i_m] \in I^m$ telle qu'aucune contrainte basique n'est requise, on peut maintenir l'homogénéité de représentation en spécifiant $sc(i_1, \dots i_m) = \emptyset$ (portée nulle) et $df(i_1, \dots i_m) \equiv true$ (fonction associant $true$ à l'affectation vide).

Pour prendre un exemple très simple, considérons un système représenté par une variable d'état v dont la valeur change à chaque instant. Nous voulons exprimer que $\forall i \in I$, $v_i \neq v_{i-1}$. La contrainte non basique associée est $c = \langle qt, sc, df\rangle$ où $qt = [\forall]$ (séquence de quantificateurs réduite au seul quantificateur $\forall$) et $\forall i \in I$, $sc(i) = [v_i, v_{i-1}]$ (portée limitée aux variables v_i et v_{i-1}) et $df(i) \equiv (v_i \neq v_{i-1})$ (définition donnée par la relation de différence entre les deux variables).

En dépit de la présence de quantificateurs, il est important de ne pas confondre ce cadre avec celui des *CSP quantifiés* (QCSP [5]). Ici, la quantification est associée aux *indices des variables* et utilisée pour spécifier de façon compacte des ensembles éventuellement infinis de contraintes, alors que la quantification est associée aux *valeurs des variables* dans le cadre QCSP.

4.2 Définition de la satisfaction de contraintes

Considérons un CNT fini $\langle TL, C\rangle$ et une affectation complète A de ce CNT, c'est-à-dire de toutes les variables de Var. On peut définir de façon récursive ce qu'est la *satisfaction* par A d'une contrainte $c \in C$.

Définition 4 *Une affectation complète A d'un CNT fini $\langle TL, C\rangle$ satisfait une contrainte $c \in C$, $c = \langle qt, sc, df\rangle$ si et seulement si elle satisfait le quadruplet $\langle \emptyset, qt, sc, df\rangle$. Une affectation complète A d'un CNT fini satisfait un quadruplet $\langle Is, qt, sc, df\rangle$, où Is est une séquence d'éléments de I, si et seulement si :*
- *si $qt = \emptyset$: $(df(Is))(A_{\downarrow sc(Is)}) = true$*
- *si $qt = [q] \cup qt'$:*
  - *si $q = \forall$: $\forall i \in I$, A satisfait $\langle Is \cup [i], qt', sc, df\rangle$;*
  - *si $q = \exists$: $\exists i \in I$ tel que A satisfait $\langle Is \cup [i], qt', sc, df\rangle$.*

Dans le premier cas (séquence vide de quantificateurs), le quadruplet spécifie une contrainte basique et la satisfaction de contrainte est définie comme habituellement dans le cadre CSP. Le second cas (séquence non vide de quantificateurs) se décompose en deux sous-cas suivant la nature du premier quantificateur de la séquence : $\forall$ ou $\exists$. On notera qu'un quantificateur universel produit une *conjonction* de contraintes, alors qu'un quantificateur existentiel produit une *disjonction*.

Nous disons qu'une affectation complète A d'un CNT fini $\langle TL, C\rangle$ est *cohérente* si et seulement si elle satisfait toutes les contraintes de C.

4.3 Complexité de la vérification de contraintes

Si toutes les variables ont des domaines finis de *taille maximale* md, si toutes les contraintes basiques induites par les contraintes non basiques sont d'*arité maximale* ma, si toutes les contraintes non basiques ont des séquences de quantificateurs de *longueur maximale* ml et si le CNT est de *longueur maximale* l, alors la *complexité temporelle* de la vérification de la satisfaction d'une contrainte par une affectation complète est $O(l^{ml} \cdot c(ma, md))$, si nous notons $c(ma, md)$ la

complexité temporelle de la vérification de la satisfaction d'une contrainte basique d'arité maximale ma sur des domaines de taille maximale md. Sans surprise, cette complexité croit de façon exponentielle avec la longueur maximale ml des séquences de quantificateurs utilisées dans la spécification des contraintes.

5 Types de contrainte utiles

La section 4.1 a introduit une façon générique de spécifier des contraintes sur des chronogrammes. Mais il peut être intéressant de se focaliser sur des types de contrainte spécifiques *a priori* très utiles pour modéliser et raisonner sur des systèmes dynamiques à événements discrets. Dans cette section, nous considérons des contraintes *temporelles pures*, d'*état instantané*, d'*événement instantané*, de *transition instantanée* et de *transition non instantanée*.

5.1 Contraintes temporelles pures

Des *contraintes temporelles pures*, impliquant seulement des *variables temporelles*, sont utiles pour contraindre les positions temporelles des instants dans les chronogrammes.

Une contrainte temporelle pure est définie comme une contrainte dont les portées $sc(i_1, \dots i_m)$ ne contiennent que des variables de t_I : $\forall [i_1, \dots i_m] \in I^m$, $sc(i_1, \dots i_m) \subseteq t_I$.

Une restriction supplémentaire intéressante consisterait à limiter à 2 l'arité des contraintes basiques et à imposer que leurs définitions soient de la forme $df(i_1, i_2) \equiv ((t_{i_1} - t_{i_2}) \in [lb, ub])$ dans le cas de contraintes binaires et $df(i) \equiv (t_i \in [lb, ub])$ dans le cas de contraintes unaires, se limitant ainsi à des *contraintes temporelles simples* [6].

On notera la présence de contraintes temporelles simples implicites dans chaque chronogramme, imposant que $\forall i \in I^-, t_{i-1} \leq t_i$. Ces contraintes peuvent être représentées par une seule contrainte non basique $c = \langle qt, sc, df \rangle$, où $qt = [\forall]$ et, $\forall i \in I^-$, $sc(i) = [t_{i-1}, t_i]$ et $df(i) \equiv (t_{i-1} \leq t_i)$.[5]

5.2 Contraintes d'état instantané

Les *contraintes d'état instantané* n'impliquent que des *variables atemporelles d'état* au même instant, auxquelles est éventuellement ajoutée la *variable temporelle* à cet instant. Elles sont utiles pour décrire l'ensemble des combinaisons de valeur des variables d'état au même instant qui sont possibles ou requises, ensemble éventuellement dépendant de la position temporelle de cet instant.

[5]Pour être complet, $sc(1) = \emptyset$ et $df(1) \equiv true$, pour spécifier qu'il n'y a pas de contrainte basique associée à $i = 1$.

Une contrainte d'état instantané est définie comme une contrainte telle que la longueur de la séquence de quantificateurs est limitée à 1 et $\forall i \in I$, $sc(i) \subseteq SV_i \cup \{t_i\}$.

Par exemple, considérons un robot équipé de deux instruments d'observation qui ne peuvent pas être simultanément actifs. Cette exigence peut être modélisée en utilisant deux chronogrammes d'état $is1$ et $is2$, de domaines booléens, représentant l'activité courante de chaque instrument, avec $true$ associée à activité, et une contrainte d'état instantané non basique $c = \langle qt, sc, df \rangle$, où $qt = [\forall]$ et, $\forall i \in I$, $sc(i) = [is1_i, is2_i]$ et $df(i) \equiv \neg(is1_i \wedge is2_i)$. Cette contrainte spécifie que $\forall i \in I$, $\neg(is1_i \wedge is2_i)$.

Pour prendre un autre exemple, supposons que nous voulions que le robot soit à une position donnée lo_G avant un instant donné t_G. Cette exigence peut être modélisée en utilisant un chronogramme d'état lo représentant la position courante du robot et une contrainte d'état instantané non basique $c = \langle qt, sc, df \rangle$, où $qt = [\exists]$ et, $\forall i \in I$, $sc(i) = [lo_i, t_i]$ et $df(i) \equiv ((lo_i = lo_G) \wedge (t_i \leq t_G))$. Cette contrainte spécifie que $\exists i \in I$ tel que $((lo_i = lo_G) \wedge (t_i \leq t_G))$.

5.3 Contraintes d'événement instantané

Les *contraintes d'événement instantané* n'impliquent que des *variables atemporelles d'événement* au même instant, auxquelles sont éventuellement ajoutées la *variable temporelle* à cet instant et des *variables atemporelles d'état* à l'instant précédent. Elles sont utiles pour décrire l'ensemble des combinaisons de valeur des variables d'événement au même instant qui sont possibles ou requises, ensemble éventuellement dépendant de la position temporelle de cet instant et des combinaisons de valeur des variables d'état juste avant cet instant, de façon à représenter par exemple des *préconditions d'action*.

Une contrainte d'événement instantané est définie comme une contrainte telle que la longueur de la séquence de quantificateurs est limitée à 1 et $\forall i \in I$, $sc(i) \subseteq EV_i \cup \{t_i\} \cup SV_{i-1}$.

Par exemple, considérons un robot qui a à sa disposition un ensemble fini A d'actions qui ne peuvent pas être déclenchées simultanément. De plus, supposons que chaque action $a \in A$ requiert un niveau $e(a)$ d'énergie pour être déclenchée. Cette exigence peut être modélisée en utilisant un chronogramme d'événement ca représentant l'action déclenchée courante, avec un domaine égal à $A \cup \{\perp\}$ ($\perp$ si aucune action n'est déclenchée), un chronogramme d'état ce représentant le niveau courant d'énergie et une contrainte d'événement instantané non basique $c = \langle qt, sc, df \rangle$, où $qt = [\forall]$ et, $\forall i \in I$, $sc(i) = [ca_i, ce_{i-1}]$ et $df(i) \equiv$

$((ca_i \neq \perp) \rightarrow (ce_{i-1} \geq e(ca_i)))$. Cette contrainte spécifie que $\forall i \in I$, $((ca_i \neq \perp) \rightarrow (ce_{i-1} \geq e(ca_i)))$.

5.4 Contraintes de transition instantanée

Les *contraintes de transition instantanée* n'impliquent que des *variables atemporelles d'état ou d'événement* au même instant, auxquelles sont éventuellement ajoutées la *variable temporelle* à cet instant et des *variables atemporelles d'état* à l'instant précédent. Elles sont utiles pour décrire l'ensemble des combinaisons de valeur des variables d'état et d'événement au même instant qui sont possibles ou requises, ensemble éventuellement dépendant de la position temporelle de cet instant et des combinaisons de valeur des variables d'état juste avant cet instant, de façon à représenter par exemple des *effets d'actions instantanées*.

Une contrainte de transition instantanée est définie comme une contrainte telle que la longueur de la séquence de quantificateurs est limitée à 1 et $\forall i \in I$, $sc(i) \subseteq V_i \cup \{t_i\} \cup SV_{i-1}$.

Par exemple, considérons un interrupteur dont la position (ouverte ou fermée) peut changer en cas d'impulsion. Supposons que cet interrupteur peut défaillir en restant bloqué à la position qu'il avait avant défaillance. Ces faits peuvent être modélisés en utilisant trois chronogrammes de domaines booléens et une contrainte de transition instantanée non basique. Un premier chronogramme d'état *sp* représente la position courante de l'interrupteur (ouverte ou fermée) avec *true* associé à ouverte. Un second chronogramme d'état *st* représente l'état courant de l'interrupteur (bloqué ou non) avec *true* associé à bloqué. Un troisième chronogramme d'événement *im* représente l'impulsion courante (présente ou non) avec *true* associé à présente et $false$ à absente ($\perp = false$). La contrainte $c = \langle qt, sc, df \rangle$ est telle que $qt = [\forall]$ et, $\forall i \in I$, $sc(i) = [sp_i, sp_{i-1}, st_i, im_i]$ et $df(i) \equiv ((sp_i \neq sp_{i-1}) \leftrightarrow (\neg st_i \wedge im_i))$. Elle spécifie que $\forall i \in I$, $((sp_i \neq sp_{i-1}) \leftrightarrow (\neg st_i \wedge im_i))$, exprimant que la position de l'interrupteur change si et seulement si l'interrupteur n'est pas bloqué et une impulsion se produit.

5.5 Contraintes de transition non instantanée

Les *contraintes de transition non instantanée* sont un peu plus complexes. Elles impliquent des *variables atemporelles d'état ou d'événement* entre deux instants i_1 et i_2, i_1 et i_2 inclus, auxquelles sont éventuellement ajoutées les *variables temporelles* aux instants i_1 et i_2 et des *variables atemporelles d'état* à l'instant $i_1 - 1$. Elles sont utiles pour décrire l'ensemble des combinaisons de valeur des variables d'état et d'événement qui sont possibles ou requises entre deux instants, ensemble éventuellement dépendant de la position temporelle des deux instants et des combinaisons de valeur des variables d'état juste avant le premier instant, de façon à représenter par exemple des *effets d'actions non instantanées*.

Une contrainte de transition non instantanée est définie comme une contrainte telle que la longueur de la séquence de quantificateurs est limitée à 2 et $\forall [i_1, i_2] \in I^2$ tels que $i_1 < i_2$, $sc(i_1, i_2) \subseteq \cup_{i_1 \leq i \leq i_2} V_i \cup \{t_{i_1}\} \cup \{t_{i_2}\} \cup SV_{i_1 - 1}$.

Par exemple, considérons un robot qui a à sa disposition un ensemble fini A d'actions qui ne peuvent pas se dérouler simultanément. De plus, supposons que chaque action $a \in A$ a une durée qui n'est pas connue précisément, mais appartient à un intervalle $[dmin(a), dmax(a)]$, et qu'une action a ne peut pas être immédiatement suivie par la même action a. Ces faits peuvent être modélisés en utilisant un chronogramme d'état *ca* représentant l'action courante, avec un domaine égal à A auquel on ajoute une valeur spéciale représentant l'absence d'action courante, et une contrainte de transition non instantanée et non basique $c = \langle qt, sc, df \rangle$, où $qt = [\forall, \exists]$ et, $\forall [i_1, i_2] \in I^2$ tels que $i_1 < i_2$, $sc(i_1, i_2) = [ca_i, (i_1 - 1) \leq i \leq i_2] \cup [t_{i_1}, t_{i_2}]$ et $df(i_1, i_2) \equiv (((ca_{i_1 - 1} \neq a) \wedge (ca_{i_1} = a)) \rightarrow ((\wedge_{i_1 < i < i_2} (ca_i = a)) \wedge (ca_{i_2} \neq a) \wedge (dmin(a) \leq (t_{i_2} - t_{i_1}) \leq dmax(a))))$, alors que $\forall [i_1, i_2] \in I^2$ tels que $i_2 \leq i_1$, $sc(i_1, i_2) = \emptyset$ et $df(i_1, i_2) \equiv false$. Il est en effet facile de montrer à partir de la définition 4 que cette contrainte spécifie que $\forall i_1 \in I$, $(((ca_{i_1 - 1} \neq a) \wedge (ca_{i_1} = a)) \rightarrow (\exists i_2 \in I, ((i_1 < i_2) \wedge (\wedge_{i_1 < i < i_2} (ca_i = a)) \wedge (ca_{i_2} \neq a) \wedge (dmin(a) \leq (t_{i_2} - t_{i_1}) \leq dmax(a)))))$.

Si les actions de A sont productrices d'une ressource, telle que la mémoire à bord en cas d'actions de télédéchargement de données, et si nous supposons que chaque action $a \in A$ produit $r(a)$ et que cette production n'est effective qu'à la fin de a, l'ensemble de ces faits peut être modélisé en utilisant un chronogramme d'état *cr* représentant le niveau courant de la ressource et en ajoutant $cr_{i_2 - 1}$ et cr_{i_2} aux portées $sc(i_1, i_2)$ et la condition $(cr_{i_2} = cr_{i_2 - 1} + r(a))$ à la partie droite des définitions $df(i_1, i_2)$.

6 Cadres englobés

Dans cette section, nous montrons comment le cadre proposé englobe des cadres existants tels que les *automates*, les *réseaux de Petri* ou les cadres classiques utilisés en *planification* ou en *ordonnancement*.

6.1 Automates

Un *automate* est habituellement défini par un quadruplet $\langle S, E, T, s_0 \rangle$ où :

1. S est un ensemble fini d'*états* ;
2. E est un ensemble fini d'*étiquettes de transition* ;
3. $T \subseteq S \times E \times S$ est un ensemble de *transitions* ;
4. $s_0 \in S$ est l'*état initial*.

Un automate spécifie des transitions possibles : une transition $e \in E$ est possible de l'état $s \in S$ vers l'état $s' \in S$ si et seulement si $\langle s, e, s' \rangle \in T$. Il est facile de montrer qu'un automate $\langle S, E, T, s_0 \rangle$ est équivalent à un CNT $\langle TL, C \rangle$ où :

1. TL est constitué de deux chronogrammes : un chronogramme d'état cs de domaine S et un chronogramme d'événement e de domaine $E \cup \{\bot\}$;
2. C est constitué de deux contraintes :
   (a) une contrainte d'état instantané basique $c_0 = \langle qt_0, sc_0, df_0 \rangle$, avec $qt_0 = \emptyset$, $sc_0 = \{cs_0\}$ et $df_0 \equiv (cs_0 = s_0)$, spécifie l'état initial ;
   (b) une contrainte de transition instantanée non basique $c = \langle qt, sc, df \rangle$, avec $qt = [\forall]$, $\forall i \in I$, $sc(i) = [cs_{i-1}, e_i, cs_i]$ et $df(i) \equiv (\langle cs_{i-1}, e_i, cs_i \rangle \in T)$, spécifie les transitions possibles.

De plus, les CNT apparaissent comme une façon compacte de spécifier des *produits synchronisés d'automates* [2].

6.2 Réseaux de Petri

Un *réseau de Petri* est habituellement défini par un quadruplet $\langle P, T, Ip, Op \rangle$ où :

1. P est un ensemble fini de *places* ;
2. T est un ensemble fini de *transitions* ;
3. Ip est une *fonction d'entrée* de $P \times T$ dans $\mathbb{N}$, qui associe un entier positif (éventuellement nul) à chaque place $p \in P$ et chaque transition $t \in T$;
4. Op est une *fonction de sortie* similaire.

Un *marquage* m (qui peut être vu comme un état) est une fonction de P dans $\mathbb{N}$, qui associe un entier $m(p)$ à chaque place $p \in P$. Pour être déclenchée à partir du marquage m, une transition $t \in T$ doit satisfaire la condition suivante : $\forall p \in P$, $m(p) \geq Ip(p,t)$. Si une transition $t \in T$ est déclenchée à partir du marquage m, le résultat est un marquage m' où $\forall p \in P$, $m'(p) = m(p) - Ip(p,t) + Op(p,t)$. Comme avec les automates, il est facile de montrer qu'un réseau de Petri $\langle P, T, Ip, Op \rangle$ est équivalent à un CNT $\langle TL, C \rangle$ où :

1. un chronogramme d'état m_p de domaine $\mathbb{N}$ est associé à chaque place $p \in P$, à quoi il est nécessaire d'ajouter un chronogramme d'événement e de domaine $T \cup \{\bot\}$;[6]
2. C est constitué de deux ensembles de contraintes :
   (a) une contrainte d'événement instantané non basique $c^E_{p,t} = \langle qt^E_{p,t}, sc^E_{p,t}, df^E_{p,t} \rangle$ est associée à chaque place $p \in P$ et chaque transition $t \in T$, avec $qt^E_{p,t} = [\forall]$, $\forall i \in I$, $sc^E_{p,t}(i) = [e_i, m_{p,i-1}]$ et $df^E_{p,t}(i) \equiv ((e_i = t) \rightarrow (m_{p,i-1} \geq Ip(p,t)))$, pour spécifier les préconditions des transitions ;
   (b) une contrainte de transition instantanée non basique $c^T_{p,t} = \langle qt^T_{p,t}, sc^T_{p,t}, df^T_{p,t} \rangle$ est associée à chaque place $p \in P$ et chaque transition $t \in T$, avec $qt^T_{p,t} = [\forall]$, $\forall i \in I$, $sc^T_{p,t}(i) = [e_i, m_{p,i-1}, m_{p,i}]$ et $df^T_{p,t}(i) \equiv ((e_i = t) \rightarrow (m_{p,i} = m_{p,i-1} - Ip(p,t) + Op(p,t)))$, pour spécifier les effets des transitions.

6.3 Problèmes de planification

Les problèmes de planification peuvent être très différents les uns des autres et, malgré de nombreux efforts, il n'existe pas de cadre unique permettant de les couvrir tous [10]. C'est pourquoi nous nous focalisons sur le plus classique d'entre eux : le cadre *STRIPS* [8] où un problème de planification est défini par un quadruplet $\langle F, A, Is, G \rangle$ où :

1. F est un ensemble fini de variables booléennes, appelées *fluents* ;
2. A est un ensemble fini d'*actions*, où chaque action $a \in A$ est définie par un triplet $\langle p_a, e^-_a, e^+_a \rangle$ où p_a, e^-_a et e^+_a sont les *préconditions*, les *effets négatifs* et les *effets positifs* de a, avec $p_a, e^-_a, e^+_a \subseteq F$ et $e^-_a \cap e^+_a = \emptyset$;
3. $Is \subseteq F$ est l'ensemble des fluents qui sont vrais dans l'*état initial* ;
4. G est un ensemble fini de conditions logiques sur F, appelées *buts*.

Un état s est défini par une fonction de F dans $\mathbb{B}$, qui associe une valeur booléenne $s(f)$ à chaque fluent $f \in F$. Les conditions suivantes doivent être satisfaites par les états et les transitions :

1. dans l'état initial s_0, $(s_0)(f)$ si et seulement si $f \in Is$;
2. une action $a \in A$ peut être déclenchée dans un état s si et seulement si $\forall f \in p_a$, $s(f)$;

[6] Dans un réseau de Petri, deux transitions ne peuvent pas être déclenchées au même instant.

3. si une action $a \in A$ est déclenchée dans un état s, le résultat est un état s' où :
   (a) $\forall f \in e_a^-$, $\neg s'(f)$;
   (b) $\forall f \in e_a^+$, $s'(f)$;
   (c) $\forall f \in F - (e_a^- \cup e_a^+)$, $s'(f) = s(f)$.

La requête habituellement associée à un problème de planification est de produire un plan, c'est-à-dire une séquence d'actions dont l'exécution permet au système d'aller de l'état initial vers un état satisfaisant les conditions buts. Il est facile de montrer qu'un problème de planification $\langle F, A, Is, G\rangle$ est équivalent à un CNT $\langle TL, C\rangle$ où :

1. un chronogramme d'état s_f de domaine booléen est associé à chaque fluent $f \in F$, à quoi il est nécessaire d'ajouter un chronogramme d'événement *act* de domaine $A \cup \{\bot\}$;
2. C est constitué de six ensembles de contraintes :
   (a) une contrainte d'état instantané basique $c_f^{Is} = \langle qt_f^{Is}, sc_f^{Is}, df_f^{Is}\rangle$ est associée à chaque fluent $f \in F$, avec $qt_f^{Is} = \emptyset$, $sc_f^{Is} = [s_{f,0}]$ et $df_f^{Is} \equiv (s_{f,0} \leftrightarrow f \in Is)$, pour spécifier l'état initial ;
   (b) une contrainte d'événement instantané non basique $c_{f,a}^P = \langle qt_{f,a}^P, sc_{f,a}^P, df_{f,a}^P\rangle$ est associée à chaque action $a \in A$ et chaque fluent $f \in p_a$, avec $qt_{f,a}^P = [\forall]$, $\forall i \in I$, $sc_{f,a}^P(i) = [act_i, s_{f,i-1}]$ et $df_{f,a}^P(i) \equiv ((act_i = a) \rightarrow s_{f,i-1})$, pour spécifier les préconditions des actions ;
   (c) une contrainte de transition instantanée non basique $c_{f,a}^{E-} = \langle qt_{f,a}^{E-}, sc_{f,a}^{E-}, df_{f,a}^{E-}\rangle$ (respectivement $c_{f,a}^{E+} = \langle qt_{f,a}^{E+}, sc_{f,a}^{E+}, df_{f,a}^{E+}\rangle$) est associée à chaque action $a \in A$ et chaque fluent $f \in e_a^-$ (respectivement $f \in e_a^+$), avec $qt_{f,a}^{E-} = qt_{f,a}^{E+} = [\forall]$, $\forall i \in I$, $sc_{f,a}^{E-}(i) = sc_{f,a}^{E+}(i) = [act_i, s_{f,i}]$, and $df_{f,a}^{E-}(i) \equiv ((act_i = a) \rightarrow \neg s_{f,i})$ (respectivement $df_{f,a}^{E+}(i) \equiv ((act_i = a) \rightarrow s_{f,i})$), pour spécifier les effets négatifs (respectivement positifs) des actions ;
   (d) une contrainte de transition instantanée non basique $c_{f,a}^N = \langle qt_{f,a}^N, sc_{f,a}^N, df_{f,a}^N\rangle$ est associée à chaque action $a \in A$ et chaque fluent $f \in F - (e_a^- \cup e_a^+)$, avec $qt_{f,a}^N = [\forall]$, $\forall i \in I$, $sc_{f,a}^N(i) = [act_i, s_{f,i-1}, s_{f,i}]$ et $df_{f,a}^N(i) \equiv ((act_i = a) \rightarrow (s_{f,i} = s_{f,i-1}))$, pour spécifier l'absence d'effet des actions ;
   (e) une contrainte d'état instantanée non basique $c^G = \langle qt^G, sc^G, df^G\rangle$, avec $qt^G = [\exists]$, $\forall i \in I$, $sc^G(i) = \cup_{g \in G}(sc(g))_i$ (si $sc(g)$ est l'ensemble des fluents impliqués dans g) et $df^G(i) \equiv (\wedge_{g \in G}(g))$, spécifie les conditions but.

6.4 Problèmes d'ordonnancement

Dans le domaine de l'ordonnancement, il n'existe pas de cadre de référence similaire au cadre *STRIPS* utilisé en planification. Il existe simplement des problèmes qui peuvent être très différents les uns des autres. Nous nous focalisons ici sur le problème d'ordonnancement de type *job-shop*, qui est l'un des problèmes d'ordonnancement les plus classiques, défini par un ensemble fini T de tâches avec, associées à chaque tâche $t \in T$, une durée du_t, une date de début au plus tôt es_t et une date de fin au plus tard le_t. On suppose que toutes les tâches requièrent une même ressource non partageable (deux tâches ne peuvent pas utiliser cette ressource au même instant) et qu'elles doivent toutes être réalisées. Soit $nt = |T|$ le nombre de tâches. Ce problème peut être modélisé par un CNT *fini* $\langle TL, C\rangle$ de longueur $l = 2 \cdot nt$ où :

1. un chronogramme d'état *ct* de domaine $T \cup \{0\}$ représente la tâche active courante, avec 0 représentant l'absence de tâche active courante (aucun chronogramme d'événement n'est nécessaire) ;
2. une contrainte de transition non instantanée et non basique $c_t = \langle qt_t, sc_t, df_t\rangle$ est associée à chaque tâche $t \in T$, avec $qt_t = [\exists]$, $\forall i \in I$, $sc_t(i) = [ct_i, t_i, t_{i+1}]$ et $df_t(i) \equiv ((ct_i = t) \wedge (es_t \leq t_i \leq t_{i+1} \leq le_t) \wedge ((t_{i+1} - t_i) = du_t))$.

7 Ce qui reste à faire

Le principal résultat de cet article est la proposition d'un cadre qui, pour la première fois, permet de modéliser des systèmes dynamiques à événements discrets (depuis les automates et les réseaux de Petri jusqu'aux problèmes de planification et d'ordonnancement) de façon uniforme en utilisant la même notion basique de *contrainte*.

Sur cette base, la première tâche est de mieux évaluer la capacité de modélisation du cadre proposé en considérant la possibilité d'englober d'autres cadres tels que la *logique temporelle* [16], le *calcul des situations* [14] ou les *automates temporisés* [1], mais aussi des problèmes variés issus du monde réel.

Quand les CNT sont finis (et donc l'ensemble des variables impliquées), le cadre proposé permet de poser de façon uniforme des problèmes de *suivi de situation*, de *validation* ou de *décision* comme des CSP ou des QCSP et de les résoudre en utilisant n'importe quel solveur de CSP ou de QCSP : CSP pour les problèmes

de suivi de situation, de validation et de décision optimiste, QCSP pour les problèmes de décision pessimistes.[7] Dans un tel contexte, il serait utile de développer ou d'adapter des mécanismes de *propagation de contraintes* adaptés aux contraintes les plus utiles que nous avons identifiées, qui peuvent être vues comme des *contraintes globales*. Au delà, il serait nécessaire d'explorer les différentes façons de répondre à des requêtes sur des CNT *infinis*, comme cela est possible avec les automates, les réseaux de Petri ou les problèmes de planification.

À propos des extensions possibles, une première extension consisterait à passer des contraintes *dures*, utilisées pour modéliser les faits possibles ainsi que les exigences fortes, à des contraintes *souples*, permettant de représenter et de raisonner sur des degrés de *plausibilité* et d'*utilité*, comme cela est fait dans [17]. Ceci permettrait de capturer des cadres tels que les *processus décisionnels markoviens* [18] ou la *planification probabiliste* [12]. Une seconde extension, orthogonale à la précédente, consisterait à relaxer l'hypothèse qu'une variable d'état reste *constante* entre deux instants successifs d'un chronogramme et à considérer des évolutions *linéaires*, *monotones* ou autres. Finalement, une troisième extension consisterait à relaxer l'hypothèse d'un *ordre total* entre instants dans un CNT.

Références

[1] R. Alur and D. Dill. A Theory of Timed Automata. *Journal of Theoretical Computer Science*, 126(2) :183–235, 1994.

[2] A. Arnold and M. Nivat. Comportements de processus. In *Actes du Colloque AFCET "Les Mathématiques de l'Informatique"*, pages 35–68, 1982.

[3] R. Barták. Dynamic Constraint Models for Complex Production Environments. In *Proc. of the Joint ERCIM/Compulog-Net Workshop*, Cyprus, 1999.

[4] A. Benveniste, P. Caspi, S. Edwards, N. Halbwachs, P. Le Guernic, and R. de Simone. The Synchronous Languages Twelve Years Later. *Proc. of the IEEE*, 91(1) :64–83, 2003.

[5] F. Börner, A. Bulatov, P. Jeavons, and A. Krokhin. Quantified Constraints : Agorithms and Complexity. In *Proc. of CSL-03*, pages 244–258, Vienna, Austria, 2003.

[6] R. Dechter, I. Meiry, and J. Pearl. Temporal Constraint Networks. *Artificial Intelligence*, 49 :61–95, 1991.

[7] G. Delzanno and A. Podelski. Constraint-based Deductive Model Checking. *Software Tools for Technology Transfer*, 3(3) :250–270, 2001.

[8] R. Fikes and N. Nilsson. STRIPS : a New Approach to the Application of Theorem Proving. *Artificial Intelligence*, 2 :189–208, 1971.

[9] J. Frank and A. Jónsson. Constraint-Based Attribute and Interval Planning. *Constraints*, 8(4) :339–364, 2003.

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

[11] H. Kautz and B. Selman. Planning as Satisfiability. In *Proc. of ECAI-92*, pages 359–363, Vienna, Austria, 1992.

[12] N. Kushmerick, S. Hanks, and D. Weld. An Algorithm for Probabilistic Planning. *Artificial Intelligence*, 76 :239–286, 1995.

[13] P. Laborie and M. Ghallab. IxTeT : an Integrated Approach for Plan Generation and Scheduling. In *Proc. of ETFA-95*, pages 485–495, Paris, France, 1995.

[14] H. Levesque, R. Reiter, Y. Lesperance, F. Lin, and R. Scherl. GOLOG : A Logic Programming Language for Dynamic Domains. *Journal of Logic Programming*, 31(1-3) :59–83, 1997.

[15] N. Muscettola. HSTS : Integrating Planning and Scheduling. In M. Zweden and M. Fox, editors, *Intelligent Scheduling*, pages 169–212. Morgan Kaufmann, 1994.

[16] A. Pnueli. The Temporal Logic of Programs. In *Proc. of FOCS-77*, pages 46–57, Providence, RI, USA, 1977.

[17] C. Pralet, G. Verfaillie, and T. Schiex. Decision with Uncertainties, Feasibilities, and Utilities : Towards a Unified Algebraic Framework. In *Proc. of ECAI-06*, pages 427–431, Riva del Garda, Italy, 2006.

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

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

[20] P. van Beek and X. Chen. CPlan : A Constraint Programming Approach to Planning. In *Proc. of AAAI-99*, pages 585–590, Orlando, FL, USA, 1999.

[7] Résoudre un problème de *suivi de situation* revient à trouver toutes les solutions du CSP associé au CNT, auquel on a ajouté les observations. Résoudre un problème de *validation* revient à prouver que le CSP associé au CNT, auquel on a ajouté la négation des propriétés à vérifier, est incohérent. En ce qui concerne les problèmes de décision, il est nécessaire de distinguer variables *contrôlables* et variables *incontrôlables*. Résoudre un problème de *décision optimiste* revient à trouver une solution du CSP associé au CNT, auquel on a ajouté les objectifs, et à la projeter sur les variables contrôlables. Résoudre un problème de *décision pessimiste* revient à trouver une solution du QCSP associé au CNT, auquel on a ajouté les objectifs, avec les variables contrôlables quantifiées existentiellement et les variables incontrôlables quantifiées universellement. Quand les domaines de certaines variables temporelles ou atemporelles sont continus, les CSP/QCSP résultants sont des problèmes hybrides discrets/continus.