# Méthodes et outils pour la vérification symbolique de systèmes temporisés

Sergio Yovine

THESE

présentée par

YOVINE Sergio

pour obtenir le titre de DOCTEUR
de l'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE
*(arrêté ministériel du 30 mars 1992)*

(Spécialité : INFORMATIQUE)

---

# Méthodes et Outils pour la Vérification Symbolique de Systèmes Temporisés

---

Date de soutenance : 19 mai 1993

| | | |
|---|---|---|
| Composition du jury : | Président | J. Mossière |
| | Rapporteurs | R. De Simone |
| | | P. Le Guernic |
| | Examinateurs | G. Berry |
| | | A. Pnueli |
| | | J. Sifakis |

Thèse préparée au sein du Laboratoire de Génie Informatique

# Remerciements

Je tiens à remercier tous ceux qui ont participé et m'ont soutenu tout au long de ce travail.

*Del lado de acá.*

Je remercie les membres du jury de cette thèse : Jacques Mossière, Professeur à l'INPG, Robert de Simone et Paul Le Guernic, Directeurs de Recherches à l'INRIA, Gérard Berry, Maître de Recherches à l'École Nationale Supérieure des Mines de Paris, et Amir Pnueli, Professeur au Weizmann Institute de Rehovot, Israël.

Je remercie Joseph Sifakis, mon directeur de thèse, pour m'avoir accueilli dans son équipe et surtout pour l'intérêt qu'il a toujours porté à mon travail. J'ai eu beaucoup de plaisir à travailler avec lui, Xavier Nicollin et Alfredo Olivero. Nous avons passé ensemble de nombreuses heures de discussion face au tableau. Je suis particulièrement reconnaissant à Xavier, pour la lecture très attentive de ce document, et à Alfredo, pour la mise en œuvre de l'outil KRONOS.

J'ai profité de la bonne ambiance créée au sein de l'équipe SPECTRE, aujourd'hui VERIMAG, dont je garde un très bon souvenir. Je remercie Claire Loiseaux, Ahmed Bouajjani, Rachid Echahed, Laurent Mounier et Riadh Robbana, qui ont accepté de lire et commenter plusieurs parties de ce document, Hubert Garavel, pour l'aide qu'il m'a souvent apportée, et Chantal Costes et Sabine Maury, qui ont résolu tous les problèmes administratifs que je leur ai occasionnés.

J'ai beaucoup apprécié l'aide de Marie-Noëlle, Pedro, Conrado et Juan pendant la préparation de mon exposé. Alejandra m'a toujours apporté son soutien et a fait que les couleurs n'ont pas manqué aux transparents.

*Del lado de allá.*

C'est grâce à Jorge Vidart, directeur de l'Escuela Superior Latinoamericana de Informática (ESLAI), Argentine, que j'ai eu l'opportunité de faire ce travail. Luiz Monteiro, professeur à l'Université de Bahía Blanca, Argentine, a motivé mes premiers pas en informatique.

Je remercie mes parents et mon frère qui, malgré la distance, m'ont toujours donné leur soutien.

*De otros lados.*

Je remercie Thomas A. Henzinger, de l'Université de Cornell, États-Unis, qui a beaucoup contribué aux résultats du Chapitre 3.

# Table des Matières

# Chapitre 1

# Introduction

## 1.1 Vérification automatique de systèmes temps-réel

### 1.1.1 Les systèmes temps-réel

Un système *temps-réel* est un système dont le comportement doit satisfaire des contraintes temporelles strictes. Les systèmes temps-réel sont de plus en plus utilisés dans la vie quotidienne. De plus, ces systèmes font souvent partie des applications critiques dans le domaine de l'avionique, le contrôle de processus industriels, le contrôle de centrales nucléaires, etc.

Les systèmes temps-réel sont en général caractérisés par des interactions complexes avec l'environnement et par des contraintes temporelles strictes dont la violation peut entraîner des conséquences graves. Par conséquent, un propriété importante de ces systèmes est leur fiabilité.

En pratique, le développement des applications temps-réel est caractérisé par l'absence des méthodes formelles et outils de spécification et validation automatiques. Les programmeurs passent souvent d'une spécification informelle à une implémentation dans un langage de programmation de systèmes temps-réel "asynchrones" comme par exemple ADA [ADA83] ou ESTELLE [ISO85].

Le problème principal dans cette approche est que les opérateurs proposés par ces langages pour exprimer les contraintes temporelles n'ont pas une sémantiques précise. En particulier, aucune hypothèse n'est faite sur la façon dont le temps progresse par rapport à l'exécution du programme ou sur la durée d'exécution d'une instruction. Le comportement d'un programme dépend du compilateur et des caractéristiques du matériel.

### 1.1.2 L'approche synchrone

Depuis quelques années, une nouvelle famille de langages appelés *synchrones* [BB91] a été proposée. Ces langages constituent une réponse au besoin d'avoir une sémantique précise du temps pour pouvoir appliquer l'approche formelle au développement des applications temps-réel. Des exemple de ces langages sont LUSTRE [CHPP87, HCRP91, HLR92], ESTEREL [BC85, BCG87, BdS91], STATECHARTS [Har87], ARGOS [Mar90] et SIGNAL [GBBG85, GGBM91].

Le point de vue synchrone consiste à supposer que le temps de réaction d'un programme est négligeable par rapport au temps de réaction de son environnement. Cette abstraction permet la définition d'une sémantique précise et simple.

De plus, les langages synchrones peuvent être implémentés efficacement, en suivant les techniques développées pour ESTEREL [BG88]. Par exemple, des compilateurs pour LUS-

TRE [Ray91, HRR91] et ARGOS [Mar90] ont été mise en œuvre. Le code objet produit par les compilateurs, appelé OC [CPPS90, Ray91], est structuré sous la forme d'un automate dont les transitions correspondent aux réactions du programme.

Une propriété importante de ces automates est que le temps d'exécution maximal d'une instruction peut être mesuré de manière précise, et par conséquent, l'hypothèse de synchronisme peut être vérifiée.

A partir des automates produits par les compilateurs il est possible de générer du code exécutable et appliquer des méthodes de vérification. Une avantage importante de cette approche est que le code vérifié est très proche du code exécuté [Ber89].

## 1.2 Motivations, objectifs et contributions

Les langages synchrones comme ESTEREL [BdS91] et LUSTRE [HLR92] sont particulièrement adaptés pour la programmation de systèmes temporisés dont l'exécution est guidée par les *ticks* d'une horloge et le délai entre deux événements est donné par le nombre de ticks entre eux.

La modélisation des systèmes où les occurrences des événements peuvent être arbitrairement proches dans le temps nécessite un modèle de temps *dense*.

En particulier, l'objectif de ce travail est de montrer que l'approche suivie pour les langages synchrones, i.e., compiler les programmes vers des automates à partir desquels il est possible de vérifier des propriétés, peut être suivie aussi dans le cas où le domaine temporel est dense.

Schématiquement, la vérification formelle de systèmes temporisés nécessite donc de la réunion de quatre éléments :

1. Un langage de *description* de systèmes temps-réel, avec des opérateurs pour exprimer les contraintes temporelles. Ce langage doit comporter une sémantique formelle qui associe une signification précise à la notion de comportement.

2. Un langage de *spécification* pour décrire les aspects qualitatifs et quantitatifs des propriétés temps-réel d'un système.

3. Une relation de *satisfaction* qui définit formellement la comparaison entre le programme à vérifier et sa spécification.

4. Un algorithme de *vérification* pour mettre en œuvre cette comparaison.

Sur le plan pratique, cette approche doit être accompagnée de deux types d'outils :

- un *compilateur*, dont le rôle est de traduire le programme à vérifier en un automate, et
- un *vérificateur*, qui implémente l'algorithme permettant de comparer l'automate à la spécification.

La figure 1.1 illustre l'architecture de cette approche.

### 1.2.1 La compilation

Comme langage de description nous avons choisi *l'algèbre de processus* temporisés ATP, proposée dans [NRSV90, NS90, NS91, NSY91, NSY92, Nic92, NSY93] pour modéliser les systèmes temps-réel.

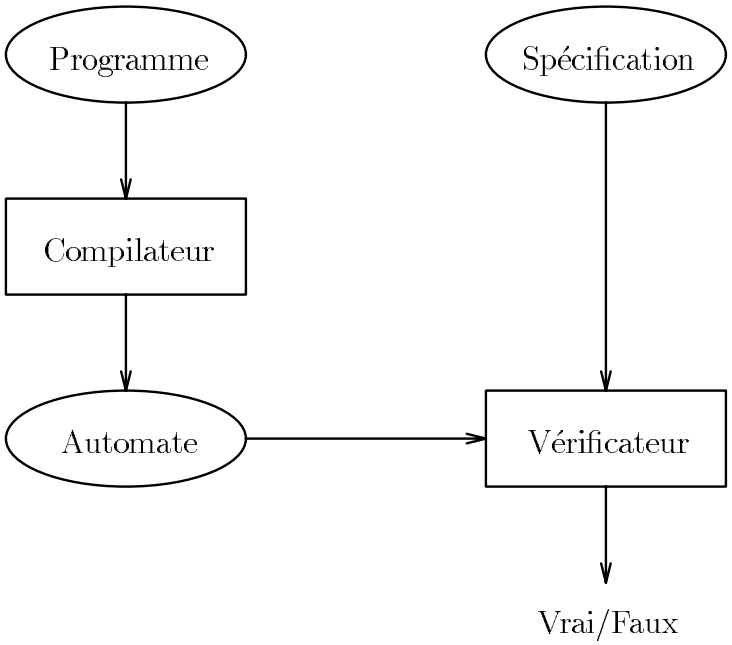



Figure 1.1: Méthode de vérification automatique.

Les algèbres de processus sont des langages de spécification qui offrent dans leur syntaxe des opérateurs qui correspondent aux constructeurs les plus importants existant dans les langages de programmation concrets.

Plusieurs algèbres de processus ont été définies pour étudier les processus communicants, comme par exemple, CCS [Mil80, Mil89], ACP [BK84, BK86] et TCSP [BHR84] (qui est une version algébrique du langage CSP [Hoa78]).

Outre des opérateurs classiques de préfixage, choix non déterministe et composition parallèle trouvés dans les algèbres de processus communicants, ATP comporte des opérateurs pour décrire des contraintes temporelles, ce qui permet d'obtenir des processus dont le comportement dépend du passage du temps.

ATP comporte un nombre restreint d'opérateurs temporels dont le choix est guidé par le souci de simplifier l'étude théorique. Pourtant, avec cet ensemble réduit d'opérateurs de base il est possible de décrire les opérateurs temporels qui apparaissent dans les langages concrets, comme les timeouts, les retards et les watchdogs.

Même si ATP peut être considérée comme un langage "jouet" à cause de l'absence de données, elle satisfait deux propriétés importantes :

1. ATP a une sémantique opérationnelle formelle, ce qui est nécessaire afin d'appliquer des techniques formelles de vérification.

2. Les opérateurs d'ATP permettent d'exprimer de manière naturelle les contraintes temporelles. Cet argument est justifié à l'aide d'exemples.

L'algèbre ATP est compilée vers des *graphes temporisés* [NSY91, NSY92], modèle qui joue un rôle similaire au rôle des automates (OC) dans l'approche synchrone. Ces graphes sont des automates étendus avec un ensemble de variables, appelées horloges, dont les valeurs croissent

uniformément avec le passage du temps, et qui servent à exprimer les contraintes temporelles. Ils sont une variante des automates temporisés définis dans [ACD90, Alu91] et des programmes temporisés à commandes gardées définis dans [HNSY92].

D'une part, les graphes temporisés sont suffisamment expressifs pour modéliser les systèmes temps-réel spécifiés en ATP. D'une autre part, ils sont suffisamment simples pour que la vérification de propriétés temps-réel soit possible. Ces deux aspects justifient pleinement leur choix comme langage objet.

La contribution de ce travail est de donner un algorithme pour construire un graphe temporisé à partir d'un terme de l'algèbre ATP. Une caractéristique importante de l'algorithme est que la taille du graphe construit est indépendante des valeurs des paramètres des constructeurs temporels d'ATP.

### 1.2.2 La vérification

Comme langage de spécification de propriétés nous considérons la *logique temporelle* temps-réel TCTL [ACD90, Alu91, HNSY92], une extension de la logique temporelle CTL [EC82, CES86], permettant d'exprimer des propriétés temps-réel.

L'utilisation des logiques temporelles en tant que formalisme de spécification de propriétés des programmes a été proposé pour la première fois dans [Pnu77], et depuis lors, ce domaine a été largement étudié [MP81, Sif82b, EC82, Lam83, CES86, MP89, Eme90].

Nous abordons dans ce travail le problème du *model-checking* pour TCTL, qui consiste à déterminer les états du modèle d'un graphe temporisé qui satisfont une formule de TCTL. Nous distinguons deux approches pour évaluer l'ensemble caractéristique d'une formule sur un modèle :

**L'approche énumérative** consiste à *construire le modèle* et à évaluer la formule après une énumération *exhaustive* de tous les états du modèle. L'ensemble caractéristique de la formule est donc représenté en extension, i.e., par énumération de ses états.

Cette approche est possible seulement si le modèle est fini. Elle a comme avantage principal la simplification du problème de la comparaison d'ensembles d'états.

En revanche, son inconvénient majeur est la taille du modèle généré qui croît exponentiellement avec le nombre de composantes parallèles du système.

**L'approche symbolique** consiste à représenter les ensembles caractéristiques par des *prédicats* et à évaluer la formule directement sur le programme *sans construire le modèle*.

Cette approche a l'avantage principal d'éviter l'explosion combinatoire due à la construction du modèle. De plus, elle peut aussi s'appliquer aux systèmes qui ont des modèles infinis.

Dans l'approche symbolique, il est crucial de disposer d'une procédure de décision efficace permettant de comparer des prédicats.

Pour les systèmes temps-réel les modèles sont non seulement infinis, mais non dénombrables du à la densité du temps. Le problème de model-checking pour TCTL a été prouvé décidable dans [ACD90, Alu91]. L'algorithme présenté repose sur la construction d'un modèle fini, réduit modulo une relation d'équivalence entre les états. Cette relation garantit que deux états équivalents ne peuvent pas être distingués par les formules de TCTL. L'évaluation de l'ensemble caractéristique est faite par application de la méthode énumérative sur le modèle réduit. L'algorithme utilisé est un algorithme d'étiquetage du modèle similaire à celui développé pour CTL [CES86].

Le principal inconvénient de cet algorithme est que la taille du modèle construit croît exponentiellement, non seulement en fonction du nombre de composantes parallèles du système, mais aussi en fonction du nombre d'horloges ainsi que des valeurs des constantes qui apparaissent dans les contraintes temporelles.

Notre objectif est de montrer que l'approche symbolique peut être appliquée pour contourner ce problème. Dans ce sens, la contribution de ce travail consiste à donner un algorithme de model-checking symbolique pour TCTL.

Pour les logiques temporelles arborescentes comme CTL, la caractérisation par point fixe des opérateurs temporels donne une méthode effective de calcul des ensembles caractéristiques de façon symbolique [Sif82a, Sif82b]. La représentation des ensembles d'états au moyen de graphes binaires de décision (BDD) [Bry86] a permis de mettre efficacement en pratique cette méthode [CBM89, BCD+90, Rat92].

Le problème principal pour appliquer une approche similaire pour TCTL est la définition d'un opérateur "état suivant" approprié, à cause de la continuité du temps.

Nous proposons dans ce travail un opérateur "état suivant" pour un domaine de temps dense, qui est utilisé pour exprimer les opérateurs temporels de TCTL au moyen de points fixes. Ce résultat fournit une méthode d'évaluation symbolique des ensembles caractéristiques des formules de TCTL.

## 1.3 Organisation du document

Nous avons résumé les motivations et les objectifs du travail, ainsi que les solutions proposées aux problèmes abordés. Nous présentons ici l'organisation du document.

Dans le **Chapitre 2** nous définissons la syntaxe et la sémantique des graphes temporisés [NSY91, NSY92]. A la différence des automates temporisés définis dans [ACD90, Alu91], les graphes temporisés ont une sémantique opérationnelle donnée au moyen des systèmes de transitions étiquetées.

Nous présentons dans le **Chapitre 3** la logique TCTL. Plusieurs définitions de TCTL ont été proposées, par exemple dans [ACD90, Alu91, HNSY92]. La version de TCTL que nous étudions dans ce chapitre correspond à celle définie dans [HNSY92]. Nous montrons que comme pour CTL, il est possible de caractériser les opérateurs temporels de TCTL comme des points fixes de fonctionnelles monotones à l'aide de l'opérateur "état suivant".

Dans le **Chapitre 4** nous proposons un algorithme de model-checking symbolique pour TCTL. Cet algorithme repose sur les résultats du chapitre précédent et sur une méthode de représentation des ensembles caractéristiques par des contraintes linéaires qui est à la base d'un algorithme de décision effectif. Cet algorithme est mis en œuvre dans l'évaluateur de KRONOS [NSY92, NOSY92, OY92].

Dans le **Chapitre 5** nous présentons brièvement l'algèbre de processus temporisés ATP. Une étude approfondie d'ATP est réalisée dans [Nic92].

Dans le **Chapitre 6** nous proposons une méthode de traduction d'ATP vers des graphes temporisés [NSY91, NSY92, NSY93]. Cette méthode est *consistante*, c'est-à-dire, le système de transitions d'un terme et celui du graphe correspondant satisfont le même ensemble de formules.

La preuve de consistance est montrée dans l'annexe A.

Le **Chapitre 7** est consacré à la présentation de l'algorithme de compilation mis en œuvre. La méthode de traduction d'ATP vers des graphes temporisés présentée dans le chapitre 6 est théorique et ne peut pas être implémentée directement de façon efficace. Afin de résoudre ce problème nous proposons une technique de compilation basée sur des transformations successives [NSY92, Yov92].

Finalement, dans le **Chapitre 8**, nous montrons à travers des études de cas comment l'approche complète est appliquée pour spécifier et vérifier des applications temps-réel significatives à l'aide des outils développés.

Nous traitons deux exemples. Le premier est une simplification d'un système téléphonique réel, néanmoins il comporte des propriétés temps-réel intéressantes. Le second est le protocole CSMA/CD permettant d'arbitrer l'accès au canal dans un réseau de stations.

# Chapitre 2

# Les systèmes temporisés

Nous étudions les *systèmes temporisés*, i.e., les systèmes dont le comportement est fortement conditionné par le passage du temps. Pour appliquer des méthodes formelles de spécification et vérification de ces systèmes nous avons besoin d'un modèle mathématique de leurs comportements. En outre, afin de pouvoir décrire les systèmes temporisés nous avons besoin aussi d'un langage de description avec des moyens pour exprimer les contraintes temporelles.

## 2.1 Un modèle des systèmes temporisés

Dans cette section nous présentons un formalisme pour modéliser les systèmes temporisés. Ce formalisme est basé sur les systèmes de transitions étiquetées.

### 2.1.1 Les hypothèses de base

Nous considérons qu'un système temporisé est composé en général par plusieurs processus communicants. Un système temporisé peut évoluer d'un état à un autre par l'*exécution d'une action*, qui est en général le résultat d'une communication, ou bien par l'*écoulement continu* du temps.

Le temps est modélisé par une variable globale fictive. Elle est fictive car elle n'est pas une vraie variable du système mais seulement une abstraction qui sert à exprimer les contraintes temporelles. Elle est globale au sens où elle est la même pour toutes les composantes du système.

L'exécution d'un système temporisé est une séquence de pas, où chaque pas a deux étapes :

- La première est une étape de *progression du temps*. Dans cette étape le temps progresse d'une quantité finie ou infinie, mais de la *même* quantité pour *toutes* les composantes du système. Une hypothèse fondamentale du modèle est donc que le temps passe de manière *synchrone* pour tous les processus qui font partie du système.

- Dans la seconde étape, les composantes exécutent, individuellement ou en coopération, une séquence arbitrairement longue mais finie d'actions *instantanées*.

Un nouveau pas commence lorsque la seconde étape termine. Le principe de fonctionnement en deux étapes pour deux processus communicants est illustré par la figure 2.1.

Le fonctionnement décrit combine les coopérations synchrone et asynchrone en deux étapes alternées : dans la première, toutes les composantes se synchronisent pour laisser passer le temps de la même quantité, dans la seconde, les composantes exécutent instantanément et de manière asynchrone une séquence d'actions.

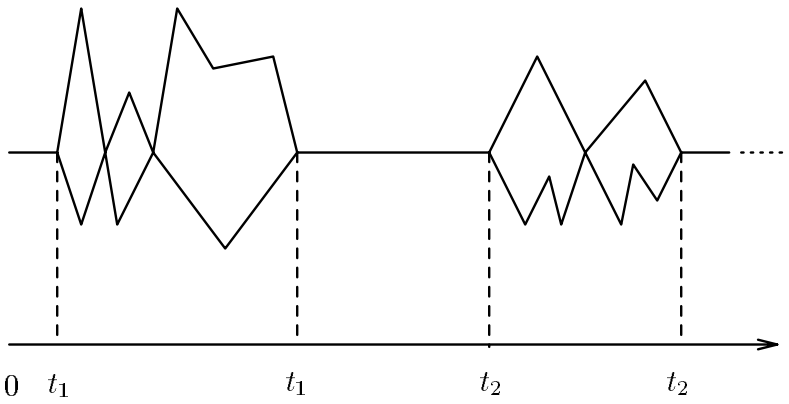



Figure 2.1: Fonctionnement en deux phases.

La plupart des modes de coopération des systèmes parallèles peuvent être obtenus par une simplification de ce modèle. Dans le fonctionnement dit *asynchrone* seulement la seconde étape existe. Dans les langages synchrones comme ESTEREL [BdS91], LUSTRE [HCRP91] et ARGOS [Mar90], un pas correspond implicitement à une unité de temps et seulement l'état atteint à la fin de la seconde l'étape est observé. Cet état est obtenu par la composition des effets produits par toutes les actions exécutées dans cette étape.

Ce mode de fonctionnement en deux étapes s'avère approprié et naturel à la modélisation des systèmes temps-réel. De plus, il introduit la notion de durée d'un pas d'exécution et permet aussi d'associer des durées aux séquences d'actions.

Ce modèle, qui peut être considéré comme peu réaliste pour modéliser les systèmes où l'exécution d'une action prend un temps non nul, s'est pourtant avéré très utile aussi bien d'un point de vue théorique que pratique. En effet, quelques conséquences intéressantes de ces hypothèses sont les suivantes :

- L'hypothèse de l'instantanéïté des actions simplifie le développement théorique. Cette hypothèse a été adoptée par des langages de programmation de systèmes temporisés comme ESTEREL [BC85, BdS91].

- L'occurrence d'une action $a$ dont l'exécution prend un temps $t > 0$ peut être modélisée par deux actions $deb\_a$ et $fin\_a$ plus une contrainte temporelle exprimant le fait que l'action $fin\_a$ doit être exécutée exactement $t$ unités de temps après l'exécution de $deb\_a$.

- Le temps est considéré *abstrait* dans le sens où il est utilisé comme un paramètre pour exprimer les contraintes temporelles imposées sur les instants des occurrences des actions.

- Le temps continu permet de modéliser les systèmes qui se caractérisent pour que deux changements d'état peuvent être arbitrairement proches dans le temps. Il est possible aussi de modéliser les systèmes où tous les changements d'état se produisent selon les *ticks* d'une horloge. Il suffit dans ce cas d'ajouter des contraintes temporelles supplémentaires pour forcer la synchronisation temporelle.

D'après ces hypothèses, nous modélisons le comportement d'un système temporisé par un système de transitions étiquetées dont les étiquettes sont des actions ou des valeurs réelles positives qui représentent l'écoulement du temps.

### 2.1.2 Les modèles

Le comportement d'un système temporisé est modélisé au moyen d'un système de transitions étiquetées. Soit $Lab$ l'ensemble des étiquettes. Un élément quelconque de $Lab$ est noté $\ell$.

**Définition 2.1** Un *système de transitions étiquetées* est un tuple $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ où

| | |
|---|---|
| $\mathcal{Q}$ | est un ensemble d'états, |
| $\rightarrow \subseteq \mathcal{Q} \times Lab \times \mathcal{Q}$ | est une relation de transition entre les états, |
| $q_I \in \mathcal{Q}$ | est l'état initial. |

Nous écrivons $s \xrightarrow{\ell} s'$ pour $\langle s, \ell, s' \rangle \in \rightarrow$. □

Un système de transitions $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ est fini si l'ensemble d'états $\mathcal{Q}$ et la relation de transition $\rightarrow$ sont finis. $T$ est *à branchement fini* si pour tout $s \in S$, l'ensemble $\{\langle \ell, s' \rangle \mid s \xrightarrow{\ell} s'\}$ est fini, c'est-à-dire, si pour tout état $s \in S$, l'ensemble des transitions qui le sont issues est fini.

Soit $Act$ l'ensemble des *actions*. Les actions sont *atomiques* et *instantanées*.

Le modèle d'un système temporisé est un système de transitions $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$. L'exécution d'une action $a$ est représentée par une transition étiquetée par $a$. L'écoulement du temps est modélisé au moyen de transitions étiquetées par des valeurs réelles positives. L'ensemble $Lab$ des étiquettes est donc $Act \cup \mathsf{R}^{>0}$.

Soit $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$. Pour que $T$ soit le modèle d'un système temporisé, nous exigeons que la relation de transition satisfasse les deux propriétés suivantes :

**Déterminisme** Le système évolue de façon déterministe du point de vue de l'écoulement du temps [NS91, Nic92, Wan90], i.e., le système ne peut évoluer, depuis un état donné et en un temps donné, que vers un seul autre état.

Cette propriété s'exprime par :

$$\forall q, q', q'' \in \mathcal{Q}. \forall t \in \mathsf{R}^{>0}.\ q \xrightarrow{t} q' \wedge q \xrightarrow{t} q'' \Rightarrow q' = q''$$

**Additivité** Les transitions temporelles doivent exprimer la continuité du temps. Si le système peut évoluer d'un état $q$ vers un état $q'$ en $t$ unités de temps, et de $q'$ vers $q''$ en $t'$ unités de temps, alors il peut naturellement évoluer de $q$ vers $q''$ en $t + t'$ unités de temps. La réciproque doit être aussi vraie.

Cette propriété appelée *additivité* du temps dans [NSY91, NS91, Nic92] et *continuité temporelle* dans [Wan90] s'exprime par :

$$\forall q, q' \in \mathcal{Q}. \forall t, t' \in \mathsf{R}^{>0}.\ (\exists q'' \in \mathcal{Q}.\ q \xrightarrow{t} q'' \wedge q'' \xrightarrow{t'} q') \iff q \xrightarrow{t+t'} q'$$

**Définition 2.2** Un *système temporisé* $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ est un système de transitions étiquetées sur $Lab = Act \cup \mathsf{R}^{>0}$ satisfaisant les propriétés de déterminisme et d'additivité des transitions temporelles. □

Notons qu'à cause de la propriété d'additivité exigée à la relation de transition et de la densité du domaine temporel, les systèmes temporisés sont en général infinis et aussi à branchement infini.

Nous avons défini un modèle de comportement des systèmes temporisés. Dans la section suivante nous définissons les graphes temporisés comme formalisme de description.

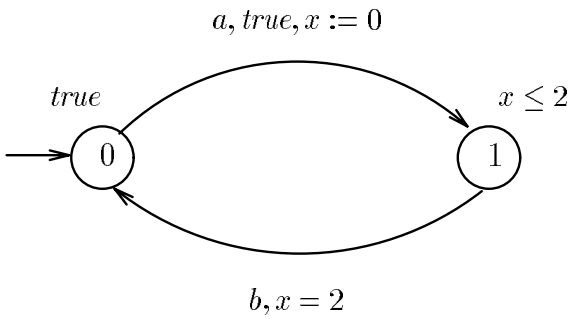



Figure 2.2: Exemple de graphe temporisé avec une horloge.

## 2.2 Les graphes temporisés

Les graphes temporisés [AD90, ACD90, Alu91, NSY91, NSY92] sont des automates étendus avec un ensemble de variables réelles, appelées *horloges*, dont les valeurs croissent uniformément avec le passage du temps.

Une horloge peut être mise à zéro par un arc du graphe. A tout instant, la valeur d'une horloge est égale au temps écoulé depuis la dernière fois qu'elle a été initialisée. Une contrainte portant sur les horloges est associée à tout sommet et à tout arc du graphe. Le système peut rester dans un sommet tant que la contrainte associée au sommet est vérifiée par les valeurs des horloges. Un arc peut être franchi seulement si les valeurs courantes des horloges satisfont la condition associée à l'arc.

### 2.2.1 Exemples introductifs

Avant de définir formellement la syntaxe et la sémantique des graphes temporisés nous considérons quelques exemples.

*Exemple.* Considérons le graphe illustré par la figure 2.2. Le sommet initial du graphe est 0. Le graphe a une seule horloge, appelée $x$. Le système commence au sommet 0 avec l'horloge $x$ initialisée à zéro. La condition *true* attachée au sommet 0 indique que le système peut y rester indéfiniment.

L'arc de 0 vers 1 est étiqueté par l'action $a$. La condition associée à l'arc est *true*. Ceci veut dire que l'arc peut être franchi à tout instant. L'étiquette $x := 0$ sur l'arc correspond à l'*initialisation* de l'horloge $x$ lorsque l'arc est franchi.

Au sommet 1, la valeur de $x$ est égale au temps écoulé depuis l'instant auquel l'arc étiqueté par l'action $a$ a été franchi. La condition $x \leq 2$ associée au sommet 1 indique que le système peut rester dans le sommet tant que la valeur de $x$ est inférieure ou égale à 2. Autrement dit, le système doit quitter le sommet au plus tard 2 unités de temps après l'arrivée. Cette contrainte *oblige* le système à franchir l'arc de 1 vers 0 lorsque l'horloge $x$ atteint la valeur 2.

Le cycle se répète lorsque le système retourne au sommet 0. Ce graphe décrit donc un système temporisé qui exécute de façon cyclique une action $a$ suivie d'une action $b$ exactement 2 unités de temps après. □

Cet exemple montre comment un délai entre l'exécution de deux actions est modélisé par un

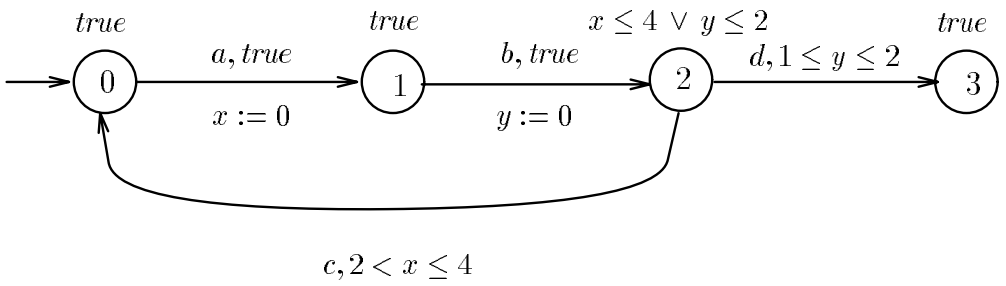



Figure 2.3: Exemple de graphe temporisé avec deux horloges.

graphe temporisé. En général, afin de restreindre le délai entre deux transitions correspondants à deux arcs $e_1$ et $e_2$, il faut initialiser une horloge dans l'arc $e_1$ et associer à l'arc $e_2$ et au sommet dont il est issu, les contraintes temporelles appropriées.

*Exemple.* Considérons le graphe temporisé illustré par la figure 2.3. Le système commence au sommet 0 avec les deux horloges $x$ et $y$ initialisées et il peut y rester indéfiniment car la condition associée est *true*.

Si l'arc de 0 vers 1 est franchi, l'horloge $x$ est initialisée. Donc, l'horloge $x$ mémorise le temps écoulé depuis l'exécution de l'action $a$. Le système peut rester au sommet 1 indéfiniment. Lorsque l'arc de 1 vers 2 est franchi, l'horloge $y$ est initialisée. Donc, au sommet 2 la valeur de $y$ est à tout instant égale au temps écoulé depuis l'exécution de l'action $b$.

La condition $2 < x \leq 4$ sur l'arc étiqueté par $c$ indique que l'action $c$ peut être exécutée entre 2 et 4 unités de temps après l'action $a$. La condition $1 \leq y \leq 2$ sur l'arc dont l'étiquette est $d$ impose un délai minimum d'une unité de temps et un délai maximum de 2 unités de temps entre l'action $b$ et l'action $d$.

La condition $x \leq 4 \vee y \leq 2$ indique que le système peut rester dans le sommet 2 tant que la valeur de $x$ est inférieure ou égale à 4, ou que la valeur de $y$ est inférieure ou égale à 2.

Ce graphe définit un système dont le comportement est le suivant :

- Si l'arc $b$ est franchi plus de 4 unités de temps après l'arc $a$, la condition $x \leq 4$ est fausse lorsque le sommet 2 est atteint. Donc, dans ce cas, l'action $b$ est suivie de l'action $d$ entre 1 et 2 unités de temps après.

- Si l'arc $b$ est franchi lorsque la valeur de $x$ satisfait $2 < x \leq 4$, la condition $x \leq 4$ deviendra fausse avant la condition $y \leq 2$. Dans ce cas, le système peut exécuter $c$ ou $d$, mais il peut toujours laisser passer l'opportunité de franchir l'arc $c$ car la condition associée au sommet lui permet de rester jusqu'à que la valeur de $y$ soit 2.

- Si l'arc $b$ est franchi lorsque la valeur de $x$ est égale à 2, le système ne peut rester dans le sommet qu'au plus 2 unités de temps. A cet instant-là, il doit choisir de façon non déterministe entre franchir l'arc $c$ ou l'arc $d$, mais il doit franchir l'un des deux.

- Si l'arc $b$ est franchi avant 2 unités de temps après l'arc $a$, la condition $y \leq 2$ deviendra fausse avant la condition $x \leq 4$. Dans ce cas, le système peut exécuter $c$ ou $d$, mais il peut toujours laisser passer l'opportunité de franchir l'arc $d$ car la condition associée au sommet lui permet de rester jusqu'à que la valeur de $x$ soit 4.

□

### 2.2.2 Les contraintes temporelles

Nous définissons tout d'abord les contraintes temporelles qui portent sur les horloges et qui sont admissibles comme conditions sur les arcs et les sommets des graphes temporisés.

**Définition 2.3** Soit $\mathcal{H}$ un ensemble fini d'horloges prenant leurs valeurs dans l'ensemble $\mathsf{R}^+$ des réels non négatifs. L'ensemble $\Psi(\mathcal{H})$ des *contraintes temporelles* sur $\mathcal{H}$ est défini par la syntaxe suivante :

$$\psi ::= \mathit{true} \mid x \prec c \mid x - y \prec c \mid \neg\psi \mid \psi \wedge \psi$$

où $x, y \in \mathcal{H}$, $c \in \mathsf{Z}$ et $\prec \in \{<, \leq\}$. □

D'autres conditions, comme par exemple, $x > 3$, $x = 2$, $2 \leq x < y + 5$, $\psi \vee \psi'$ peuvent être définies comme des abréviations.

Une *valuation* $v$ des horloges est une fonction qui associe à chaque $x \in \mathcal{H}$ une valeur dans $\mathsf{R}^+$. L'ensemble des valuations est noté $V$.

Pour chaque $\psi \in \Psi(\mathcal{H})$ et $v \in V$, $\psi(v)$ représente la valeur de la condition $\psi$ évaluée en $v$. La valuation $v$ satisfait une condition $\psi$ si $\psi(v)$ est vrai.

Soient $v \in V$ une valuation des horloges et $H \subseteq \mathcal{H}$ un sous-ensemble d'horloges, $v[H := 0]$ est la valuation définie par :

$$v[H := 0](x) = \begin{cases} 0 & \text{si } x \in H, \\ v(x) & \text{sinon.} \end{cases}$$

Donc, la valuation $v[H := 0]$ est la valuation $v'$ obtenue de $v$ par initialisation des horloges qui appartiennent à $H$.

Soient $v \in V$ une valuation des horloges et $t \in \mathsf{R}^+$, $v + t$ est la valuation $v' \in V$ telle que $v'(x) = v(x) + t$ pour tout $x \in \mathcal{H}$. Donc, la valuation $v + t$ est la valuation obtenue de $v$ en augmentant la valeur des horloges de $t$.

Soit $\psi \in \Psi(\mathcal{H})$ une contrainte temporelle. Pour $H \subseteq \mathcal{H}$, on note $\psi[H := 0]$ la contrainte temporelle $\psi' \in \Psi(\mathcal{H})$ obtenue à partir de $\psi$ dans laquelle, pour tout $x \in H$, 0 est substitué à toutes les occurrences de $x$. Pour $t \in \mathsf{R}^+$, on note $\psi + t$ la contrainte obtenue à partir de $\psi$ dans laquelle, pour tout $x \in \mathcal{H}$, $x + t$ est substitué à toutes les occurrences de $x$. Notons que $\psi + t$ n'appartient pas à $\Psi(\mathcal{H})$.

**Remarque 2.1** Pour toute valuation $v \in V$ et pour toute contrainte temporelle $\psi \in \Psi(\mathcal{H})$,

1. pour tout $H \subseteq \mathcal{H}$, $v$ satisfait $\psi[H := 0]$ si et seulement si $v[H := 0]$ satisfait $\psi$,
2. pour tout $t \in \mathsf{R}^+$, $v$ satisfait $\psi + t$ si et seulement si $v + t$ satisfait $\psi$.

□

### 2.2.3 Syntaxe et sémantique des graphes temporisés

Nous présentons ici la définition formelle des graphes temporisés.

**Définition 2.4** Un *graphe temporisé* est un tuple $\langle S, H, E, s_I, \delta \rangle$ où

- $S$ est un ensemble de sommets,
- $H \subseteq \mathcal{H}$ est un ensemble d'horloges,
- $E \subseteq S \times Act \times \Psi(H) \times \mathbf{2}^H \times S$ est un ensemble d'arcs,
- $s_I \in S$ est le sommet initial,
- $\delta : S \to \Psi(H)$ associe à chaque sommet une condition appelée *condition d'activité* du sommet.

On écrit souvent $\delta_s$ au lieu de $\delta(s)$. On exige

$$\forall s \in S, v \in V, t \in \mathsf{R}^+.\ \delta_s(v+t) \Rightarrow \delta_s(v)$$

et on dit que $\delta_s$ est *fermée en arrière.* □

Soit $G = \langle S, H, E, s_I, \delta \rangle$ un graphe temporisé. $G$ définit un système temporisé dont le comportement est le suivant :

- Un état du système est représenté par un sommet du graphe et une valuation des horloges.
- L'état initial du système est $(s_I, v_I)$ où $s_I$ est le sommet initial du graphe et $v_I$ est telle que $v_I(x) = 0$ pour tout $x \in H$.
- A partir de l'état $(s, v)$, un arc $\langle s, a, \psi, H', s' \rangle \in E$ peut être franchi si les valeurs des horloges satisfont la condition $\psi$, i.e., si $\psi(v)$ est vrai. L'état résultant est $(s', v')$, où $v' = v[H' := 0]$.
- Les valeurs des horloges croissent uniformément avec le passage du temps. A tout instant, la valeur d'une horloge est égale au temps écoulé depuis la dernière fois où l'horloge a été mise à zéro.
- Le système peut séjourner dans un sommet $s$ tant que la condition d'activité du sommet $\delta_s$ est vérifiée par les valeurs des horloges.

Formellement, la sémantique opérationnelle d'un graphe temporisé lui associe un système de transitions étiquetées. Chaque transition représente le fait de franchir un arc ou simplement de laisser passer le temps dans un sommet en y séjournant. Dans le premier cas, la transition est étiquetée par l'action associée à l'arc. Dans l'autre cas, l'étiquette est une valeur réelle positive égale au temps de séjour dans le sommet.

**Définition 2.5** Le modèle d'un graphe temporisé $G$, noté $\mathcal{T}[\![G]\!]$, est le système de transitions étiquetées $\langle \mathcal{Q}, \to, q_I \rangle$ où :

- $\mathcal{Q} = \{(s, v) \mid \delta_s(v)\}$
- $q_I = (s_I, v_I)$
- La relation de transition $\to$ est la plus petite relation définie par les règles suivantes :

$$\frac{\langle s, a, \psi, H', s' \rangle \in E \wedge \psi(v)}{(s, v) \xrightarrow{a} (s', v[H' := 0])} \tag{2.1}$$

$$\frac{\delta_s(v+t)}{(s, v) \xrightarrow{t} (s, v+t)} \tag{2.2}$$

Nous considérons seulement les états qui satisfont la condition d'activité. □

Nous avons la proposition suivante.

**Proposition 2.1** *Pour tout graphe temporisé $G$, le système de transitions étiquetées $\mathcal{T}[\![G]\!] = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ est un système temporisé.*

**Remarque 2.2** La propriété d'additivité temporelle est assurée par la condition de fermeture en arrière exigée à la fonction $\delta$. □

### 2.2.4 Exemples

Nous présentons ici quelques exemples. Par convention, lorsqu'une condition est égale à *true*, elle est omise. L'ensemble des horloges initialisées par un arc est représenté par une liste d'affectations qui est omise lorsqu'elle est vide.

**Exemple 2.1** Le premier exemple consiste à spécifier un système qui reconnaît les clicks simples et les clicks doubles d'une souris à un seul bouton [Sha92]. Cette application est intéressante car le comportement du processus dépend exclusivement des temps relatifs entre les événements.

Il y a deux types d'événements. Le premier consiste à presser le bouton de la souris et le second consiste à le relâcher. Presser le bouton est signalé par l'action $p$ et le relâcher par l'action $r$.

Un *click simple* est défini lorsque le temps écoulé entre presser et relâcher le bouton est inférieur à une certaine constante $t_{cs}$. Un *click double* est défini par deux clicks simples séparés d'au plus $t_{cd}$ unités de temps.

Le graphe temporisé qui spécifie le comportement du système est illustré par la figure 2.4.

Le sommet initial du graphe est 0. Le système commence au sommet 0 avec $x = 0$. Lorsque le signal $p$ arrive, le système franchi l'arc de 0 vers 1 dont l'étiquette est $p$ et initialise la valeur de $x$.

Au sommet 1, la valeur de $x$ représente le temps écoulé depuis la première fois que le bouton a été pressé. Si le signal $r$ arrive avant $t_{cs}$ unités de temps, le système franchi l'arc de 1 vers 2 et initialise la valeur de $x$.

Au sommet 2, la valeur de $x$ représente le temps écoulé depuis l'instant où le bouton a été relâché. La contrainte $x \leq t_{cd}$ indique que le système ne peut pas rester dans le sommet au-delà de $t_{cd}$ unités de temps. Lorsque $x$ atteint la valeur $t_{cd}$, un click simple est reconnu en franchissant l'arc étiqueté par $cs$ de 2 vers le sommet 3. Si un second signal $p$ arrive avant $t_{cd}$ unités de temps, le système franchit l'arc étiqueté par $p$ vers le sommet 4. Au sommet 3 le système attend un nouveau signal $p$ pour recommencer.

Au sommet 4, la valeur de $x$ représente le temps écoulé depuis la seconde fois que le bouton a été pressé. Si le signal $r$ arrive avant $t_{cs}$ unités de temps, le système franchit l'arc $r$ vers le sommet 6 et initialise la valeur de $x$. Par contre, lorsque $x$ atteint la valeur $t_{cs}$ et aucun signal $r$ n'a été reçu, le système signale un click simple et franchit l'arc $cs$ vers le sommet 5, où il attend indéfiniment que le bouton soit relâché pour retourner au sommet initial.

La condition $x = 0$ associée au sommet 6 indique que le système doit le quitter immédiatement en franchissant l'arc $cd$, ce qui correspond à signaler un click double. □

**Exemple 2.2** Nous décrivons ici le contrôleur de température d'un réacteur [JLHM91]. La température du réacteur est captée périodiquement par un capteur. La tâche du contrôleur consiste à réfrigérer le réacteur lorsqu'il reçoit le signal $r$ du capteur.

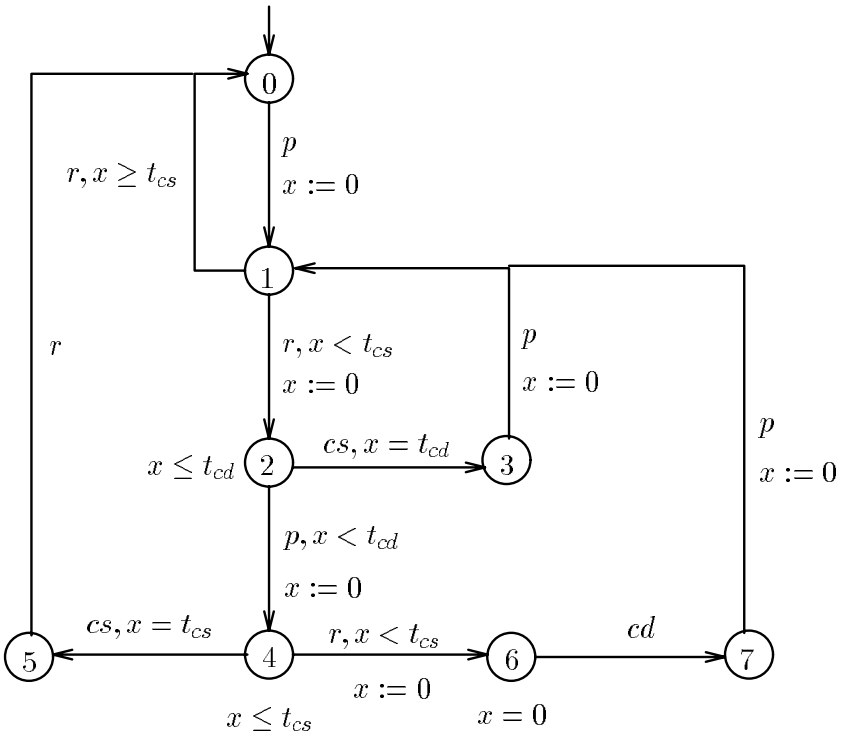



Figure 2.4: Graphe temporisé de la souris.

L'opération de réfrigération est mise en œuvre au moyen de deux barres qui doivent être utilisées en ordre. Lorsque le contrôleur reçoit le signal $r$ du capteur, commence l'opération de mouvement de la barre 1, ce qui prend un certain temps $t_1$. Ensuite, il répète l'opération avec la barre 2, ce qui prend $t_2$ unités de temps.

L'arrivée d'un signal $r$ pendant le temps de réfrigération est considérée comme une erreur. De plus, le capteur n'est pas fiable, i.e., il peut tomber en panne et ne pas envoyer le signal $r$ au contrôleur. Dans ce cas, et pour des raisons de sécurité, si le temps écoulé depuis le dernier signal $r$ reçu est supérieur à $t_{max}$, le contrôleur commence une nouvelle opération de réfrigération.

La figure 2.5 montre le graphe temporisé du contrôleur de température du réacteur.

Le système commence au sommet 0 avec $x$ et $y$ égales à 0. Lorsque le signal $r$ arrive, le contrôleur franchit l'arc vers le sommet 1 et initialise l'horloge $x$. L'horloge $x$ sert à mesurer le temps écoulé depuis la dernière fois qu'un signal $r$ a été reçu. La condition associée au sommet 1 étant $x = 0$, le système doit quitter le sommet immédiatement en franchissant l'arc $b_1$ vers le sommet 2, ce qui correspond à initier le mouvement de la barre 1.

Le contrôleur reste au sommet 2 jusqu'à ce qu'un nouveau signal $r$ arrive ou le mouvement de la barre 1 soit fini.

Dans le premier cas, le système franchit l'arc vers le sommet 5 et initialise $y$. Comme la condition associée au sommet 5 est $y = 0$, le contrôleur doit franchir l'arc $h$ vers 6 immédiatement, ce qui correspond à signaler l'erreur.

Dans l'autre cas, lorsque le mouvement de la barre termine, i.e., lorsque la valeur de $y$ est égale à $t_1$, le contrôleur franchit l'arc $b_2$ vers 3, ce qui correspond à commencer le mouvement de la barre 2.

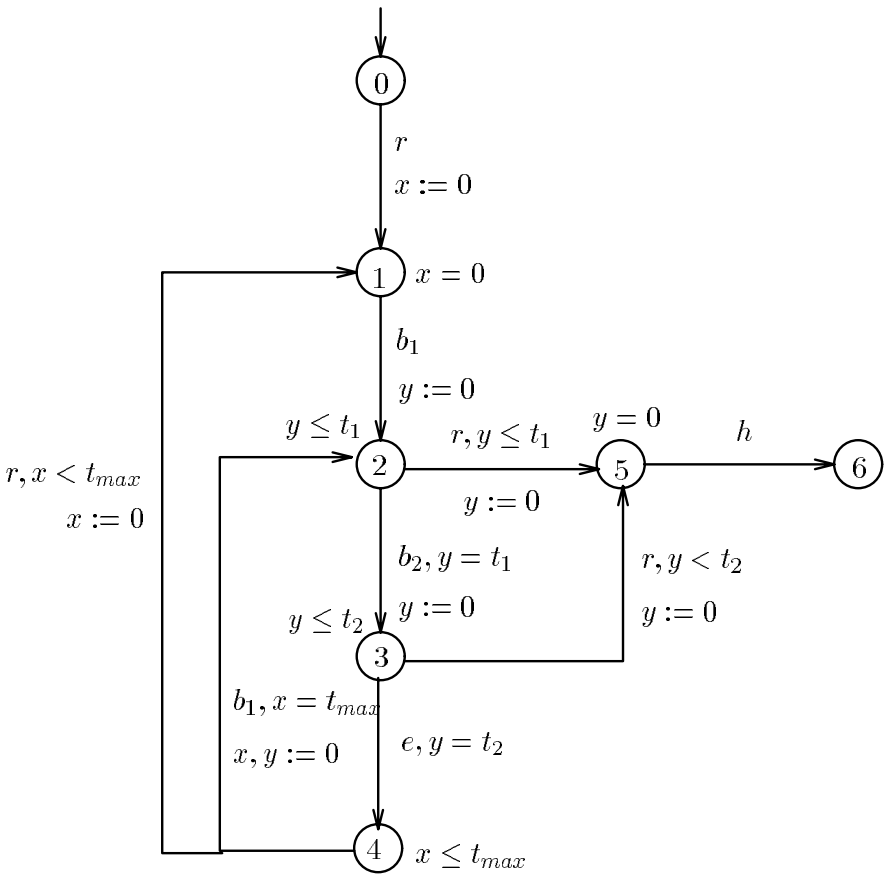



Figure 2.5: Graphe temporisé du contrôleur de température.

Le comportement du contrôleur au sommet 3 est similaire. Ici, la valeur de $y$ est égale au temps écoulé depuis l'action $b_2$. Lorsque $y = t_2$, le système franchit l'arc $e$ vers 4.

Dans 4, si un signal $r$ arrive avant que la valeur de $x$ atteigne $t_{max}$, le contrôleur retourne au sommet 1 et initialise $x$. Lorsque $x = t_{max}$, le système franchit l'arc $b_1$ vers 2 et initialise $x$ et $y$. □

# Chapitre 3

# TCTL : une logique temporelle temps-réel

Nous présentons dans ce chapitre un langage de spécification de systèmes temporisés. La logique temporelle CTL a été introduite dans [EC82] comme formalisme de spécification de propriétés pour les systèmes non temporisés. CTL permet d'exprimer plusieurs propriétés intéressantes. Par exemple, la formule $\forall\Box p$ dit que la propriété $p$ est un invariant du système, i.e., dans tous les états et tout au long de toutes les exécutions, la propriété $p$ est vraie. La formule $\forall\Box\exists\Diamond p$ dit qu'il est toujours possible d'atteindre un état qui satisfait la propriété $p$.

Dans ce chapitre nous étudions la logique TCTL [ACD90, Alu91, HNSY92] qui est une extension de CTL permettant d'exprimer des propriétés temps-réel. Une façon simple d'introduire le temps explicitement dans la syntaxe consiste à borner la portée dans le temps des opérateurs temporels. Cette approche a été suivie dans [ACD90]. Par exemple, nous pouvons écrire $\exists\Diamond_{<2}p$ pour exprimer qu'il existe une exécution où la propriété $p$ est vraie avant 2 unités de temps. D'autres notations ont été proposées dans [Alu91, HNSY92]. La version de TCTL que nous étudions dans ce chapitre correspond à celle définie dans [HNSY92].

Nous présentons tout d'abord dans la section 3.1 les modèles sur lesquels nous allons interpréter les formules de la logique. Ces modèles sont basés sur les systèmes de transitions étiquetées définis dans 2.1. Dans la section 3.2 nous définissons la syntaxe et la sémantique de TCTL.

Les opérateurs temporels de CTL sont exprimables en termes de points fixes de fonctionnelles monotones à l'aide de l'opérateur "état suivant"noté $\bigcirc$. Par exemple, l'ensemble des états qui satisfont la formule $p_1\,\exists\mathcal{U}\,p_2$ de CTL est le plus petit point fixe de la fonctionnelle monotone $F(X) = Q_2 \vee Q_1 \wedge \bigcirc X$ où $Q_1$ et $Q_2$ sont les ensembles des états qui satisfont $p_1$ et $p_2$ respectivement.

A cause de la continuité du temps, le problème principal pour suivre une approche similaire dans le cas de TCTL est définir un opérateur "état suivant", car il n'y a aucune granularité pour la progression du temps. Dans la section 3.3 nous montrons que les opérateurs temporels de TCTL peuvent être aussi exprimés en termes de points fixes à l'aide d'un opérateur "état suivant" défini de façon appropriée.

Finalement, ces caractérisations donnent un moyen d'évaluation effective des formules qui est le noyau de l'algorithme de model-checking présenté dans le chapitre 4.

## 3.1 Les séquences d'états

Jusqu'à présent nous avons modélisé le comportement d'un système temporisé par un système de transitions. Comme les formules de la logique TCTL expriment des propriétés sur les séquences d'états, dans cette section nous définissons les exécutions d'un système temporisé, i.e., les séquences d'états produites par le système de transitions.

### 3.1.1 Les états

Soit $G = \langle S, H, E, s_I, \delta \rangle$ un graphe temporisé, $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ le modèle de $G$, et $Pro$ un ensemble fini de *propositions* dont un élément est noté $p$.

Nous considérons une fonction $\nu : S \rightarrow \mathbf{2}^{Pro}$ qui associe à chaque sommet du graphe $G$ l'ensemble des propositions vraies dans le sommet.

Un état $(s, v) \in \mathcal{Q}$ est noté $q$. On définit :

- Pour tout $x \in \mathcal{H}$, $q(x) = v(x)$.
- Pour tout $p \in Pro$,

$$q(p) = \begin{cases} \mathit{true} & \text{si } p \in \nu(s), \\ \mathit{false} & \text{sinon.} \end{cases}$$

Nous utilisons la notation suivante :

- Pour tout $H' \subseteq \mathcal{H}$, $q[H' := 0]$ est l'état $(s, v[H' := 0])$.
- Pour tout $t \in \mathsf{R}^+$, $q + t$ est l'état $(s, v + t)$. Notons que pour toute proposition $p \in Pro$, $q(p) = (q + t)(p)$.

**Définition 3.1** L'ensemble $\Phi(\mathcal{H})$ des *prédicats d'état* est défini par la grammaire suivante :

$$\phi ::= p \mid x \prec c \mid x - y \prec c \mid \neg\phi \mid \phi \wedge \phi$$

où $p \in Pro$, $x, y \in \mathcal{H}$, $c \in \mathsf{Z}$ et $\prec \in \{<, \leq\}$. □

Un prédicat d'état est interprété de manière standard comme un prédicat sur l'ensemble des états. Pour tout prédicat d'état $\phi \in \Phi$ et pour tout état $q \in \mathcal{Q}$, $\phi(q)$ représente la valeur de $\phi$ évaluée en $q$. L'état $q$ satisfait le prédicat d'état $\phi$ si $\phi(q)$ est vrai.

Soit $\phi \in \Phi(\mathcal{H})$ un prédicat d'état. Pour $H' \subseteq \mathcal{H}$, on note $\phi[H' := 0]$ le prédicat d'état $\phi' \in \Phi(\mathcal{H})$ obtenu à partir de $\phi$ dans lequel, pour tout $x \in H'$, 0 est substitué à toutes les occurrences de $x$. Pour $t \in \mathsf{R}^+$, on note $\phi + t$ le prédicat obtenu à partir de $\phi$ dans lequel, pour tout $x \in \mathcal{H}$, $x + t$ est substitué à toutes les occurrences de $x$. Notons que $\phi + t$ n'appartient pas à $\Phi(\mathcal{H})$.

**Remarque 3.1** Pour tout état $q \in \mathcal{Q}$ et pour tout prédicat d'état $\phi \in \Phi$,

1. pour tout $H' \subseteq \mathcal{H}$, $q$ satisfait $\phi[H' := 0]$ si et seulement si $q[H' := 0]$ satisfait $\phi$,
2. pour tout $t \in \mathsf{R}^+$, $q$ satisfait $\phi + t$ si et seulement si $q + t$ satisfait $\phi$.

□

Par la suite, nous supposons qu'un système temporisé $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ est le modèle d'un graphe temporisé $G$.

### 3.1.2 Les séquences

Contrairement au cas des systèmes non temporisés, une exécution d'un système temporisé ne peut pas être représentée par une séquence discrète d'états. En effet, une transition $q \xrightarrow{t} q'$ modélise le passage par tous les états $q + t'$ pour $t' \leq t$. Il n'est pas non plus possible de modéliser une exécution par une application de $\mathsf{R}^+$ dans $\mathcal{Q}$ puisque plusieurs états peuvent être visités *au même instant*. Afin de modéliser les exécutions d'un système temporisé nous définissons les notions de pas et de séquences de pas.

Tout d'abord, nous introduisons la notation suivante. Soit $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$.

- Pour tout $q, q' \in \mathcal{Q}$, $q \overset{0}{\mapsto} q'$ si $q = q'$ ou s'il existe une action $a \in Act$ telle que $q \xrightarrow{a} q'$.
- Pour tout $q, q' \in \mathcal{Q}$ et $t > 0$, $q \overset{t}{\mapsto} q'$ si $q \xrightarrow{t} q'$.

Notons que si $q \overset{t}{\mapsto} q'$ et $t > 0$ alors $q' = q + t$. Une transition $q \overset{0}{\mapsto} q'$ est appelée *instantanée*. Donc, une transition instantanée consiste à rester dans le même état sans laisser passer le temps ou à exécuter une action.

**Définition 3.2** Soit $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ un système temporisé. Pour $q, q' \in \mathcal{Q}$ et $t \in \mathsf{R}^+$, on définit :

$$q \overset{t}{\triangleright} q' \quad \text{si} \quad q \overset{t}{\mapsto} q + t \text{ et } q + t \overset{0}{\mapsto} q'.$$

On dit que $q \overset{t}{\triangleright} q'$ est un *pas* de $q$ à $q'$ de durée $t$. Il existe un pas de $q$ à $q'$, noté $q \triangleright q'$, s'il existe $t \in \mathsf{R}^+$ tel que $q \overset{t}{\triangleright} q'$. □

Un pas de durée $t$ consiste à laisser passer un temps $t$ et ensuite faire une transition instantanée. Tout état a une transition instantanée vers lui-même, par conséquent pour tout $q, q' \in \mathcal{Q}$ et $t \in \mathsf{R}^+$, si $q \overset{t}{\mapsto} q'$ alors $q \overset{t}{\triangleright} q'$. L'inverse n'est pas vrai.

**Définition 3.3** Soit $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ un système temporisé. Une *séquence de pas*

$$\sigma \;=\; q_0 \overset{t_0}{\triangleright} q_1 \overset{t_1}{\triangleright} \ldots$$

est une suite infinie de pas. L'ensemble des séquences de pas de $T$ est noté $\Sigma_T$.

Pour tout $q \in \mathcal{Q}$, on définit :

$$\Sigma_T(q) \;=\; \{q_0 \overset{t_0}{\triangleright} q_1 \overset{t_1}{\triangleright} \ldots \mid q_0 = q\}$$

Donc, $\Sigma_T(q) \subseteq \Sigma_T$ est l'ensemble des séquences de pas de $T$ qui ont $q$ comme état initial. Par la suite une séquence de pas est appelée simplement séquence. □

Un état peut être visité plusieurs fois par la même séquence. Il est donc nécessaire de distinguer entre l'état et l'occurrence de l'état dans la séquence. Une occurrence est définie par un état et un indice. Toute séquence définit un ordre en deux niveaux entre les occurrences. Le premier niveau est donné par les indices des occurrences. Le second niveau est déterminé par le temps à l'intérieur de l'intervalle définie par un pas.

**Définition 3.4** Soit $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ un système temporisé. Pour toute séquence

$$\sigma \quad = \quad q_0 \overset{t_0}{\triangleright} q_1 \overset{t_1}{\triangleright} \ldots$$

de $\Sigma_T$, le *support* de $\sigma$ est une paire $(\chi(\sigma), \preceq)$ où :

$$\chi(\sigma) = \bigcup_{i \in \mathsf{N}} \chi_i(\sigma)$$

où pour tout $i \in \mathsf{N}$,

$$\chi_i(\sigma) = \{(q_i + t, i) \mid t \leq t_i\}$$

et $\preceq$ est un ordre total défini par :

$$\forall i, j \in \mathsf{N}, t, t' \in \mathsf{R}^+. \ (q_i + t, i) \preceq (q_j + t', j) \iff i < j \vee i = j \wedge t \leq t'$$

□

**Définition 3.5** Soit $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ un système temporisé. Une séquence $\sigma \in \Sigma_T$ *diverge* si pour tout $t \in \mathsf{R}^+$, il existe $i \in \mathsf{N}$ tel que $t < \sum_{j \leq i} t_j$. L'ensemble des séquences divergentes de $T$ est noté $\Delta_T$. □

Les séquences divergentes sont celles où le temps peut progresser au-delà de tout instant. Le comportement d'un système temporisé doit garantir que l'écoulement du temps ne soit jamais bloqué. Pourtant, l'ensemble des séquences d'un système temporisé contient aussi des séquences dites de "Zenon" où le temps converge. Ceci motive la définition suivante.

**Définition 3.6** Soit $T$ un système temporisé. $T$ est dit *bien temporisé* si tout préfixe fini d'une séquence de $T$ est préfixe d'une séquence divergente de $T$. □

Nous illustrons à présent quelques exemples de systèmes qui ne sont pas bien temporisés.

**Exemple 3.1** Considérons le graphe temporisé illustré par la figure 3.1. Le système temporisé définit par ce graphe n'est pas bien temporisé. En effet, il existe une séquence de pas qui commence au sommet 0 avec $x = 0$, où le système alterne entre les sommets 0 et 1 jusqu'à un instant où la valeur de $x$ est supérieure à 3 et, par conséquent, la transition de 0 vers 2 n'est plus possible. La séquence est alors inévitablement convergente car la condition d'activité des sommets 0 et 1 impose la condition $x \leq 3$, et donc, l'instant 3 ne peut pas être dépassé.

Par contre, le graphe illustré par la figure 3.2 est bien temporisé. Notons que les deux graphes ont le même ensemble de séquences divergentes. □

**Exemple 3.2** Considérons à présent le graphe illustré par la figure 3.3. Le système temporisé définit par ce graphe n'est pas bien temporisé. Par exemple, le système peut séjourner au sommet 1 tant que la valeur de $z$ est inférieure à 8. Aucune contrainte n'est imposée aux valeurs de $x$ et $y$, et par conséquent, elles peuvent augmenter au-delà des bornes 5 et 7, respectivement. Dans ce cas, le système reste "bloqué", ne pouvant pas franchir l'arc de 1 vers 0.

Comme dans l'exemple précédent, il est possible de renforcer les conditions d'activité des sommets pour obtenir un système bien temporisé (cf. figure 3.4). □

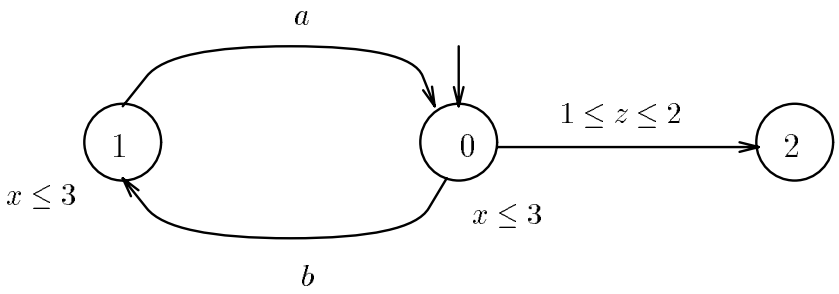



Figure 3.1: Exemple de système qui n'est pas bien temporisé.

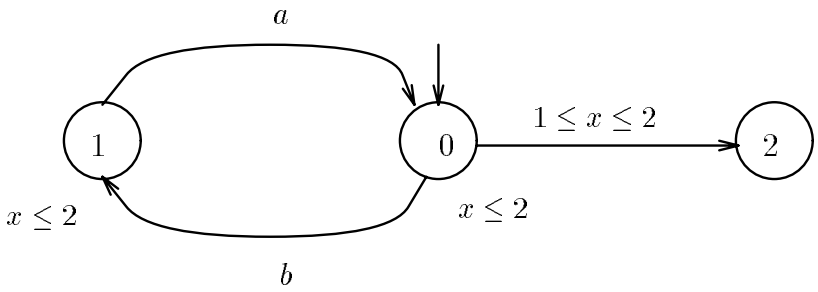



Figure 3.2: Système bien temporisé.

## 3.2 La logique TCTL

TCTL est une extension de la logique temporelle arborescente CTL [EC82, CES86].

### 3.2.1 Syntaxe et sémantique

Les formules de TCTL sont construites à partir des propositions et de contraintes sur les horloges au moyen d'opérateurs logiques, d'un opérateur d'initialisation d'horloge noté $z.$, et de deux opérateurs temporels notés $\exists\mathcal{U}$ et $\forall\mathcal{U}$.

Intuitivement, un état $q$ satisfait $\varphi_1 \exists\mathcal{U} \varphi_2$ s'il existe une séquence à partir de $q$ pour laquelle la formule $\varphi_1$ est continuellement vraie jusqu'à un état qui satisfait $\varphi_2$.

Un état $q$ satisfait $\varphi_1 \forall\mathcal{U} \varphi_2$ si pour toute séquence à partir de $q$, $\varphi_1$ est continuellement vraie jusqu'à un état où $\varphi_2$ est vraie.

L'opérateur d'initialisation $z.$ sert à introduire une horloge auxiliaire $z \in \mathcal{H}$ qui permet de mesurer le temps écoulé à partir de l'état présent. Dans la formule $z.\varphi$, les occurrences de $z$ dans $\varphi$ sont liées par l'opérateur $z.$ Lorsque la formule $z.\varphi$ est interprétée sur un système temporisé $T$, nous exigeons que $z$ ne soit pas une horloge de $T$, c'est-à-dire, si $T$ est le modèle d'un graphe temporisé $G$ dont l'ensemble d'horloges est $H \subseteq \mathcal{H}$, alors $z \notin H$.

**Définition 3.7** Les formules de TCTL son définies par la syntaxe suivante :

$$\varphi ::= p \mid x \prec c \mid x - y \prec c \mid \neg\varphi \mid \varphi \vee \varphi \mid z.\varphi \mid \varphi \exists\mathcal{U} \varphi \mid \varphi \forall\mathcal{U} \varphi$$

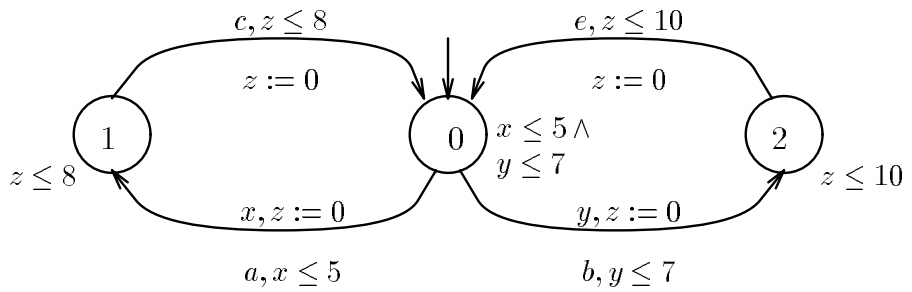



Figure 3.3: Exemple de système qui n'est pas bien temporisé.

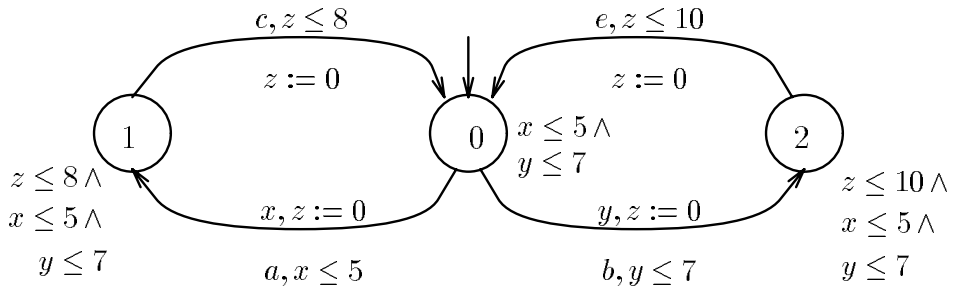



Figure 3.4: Système bien temporisé.

où $p \in Pro$, $x, y, z \in \mathcal{H}$, $c \in \mathsf{Z}$ et $\prec \in \{<, \leq\}$. □

Les formules de TCTL sont interprétées sur les états d'un système temporisé.

**Définition 3.8** Pour un système temporisé $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$, un ensemble de séquences $\Sigma \subseteq \Sigma_T$, un état $q \in \mathcal{Q}$ et une formule $\varphi$ de TCTL, la relation de satisfaction $q \models_\Sigma \varphi$ est définie par induction sur $\varphi$ de la façon suivante :

$q \models_\Sigma p$ ssi $q(p) = true$
$q \models_\Sigma x \prec c$ ssi $q(x) \prec c$
$q \models_\Sigma x - y \prec c$ ssi $q(x) - q(y) \prec c$
$q \models_\Sigma \neg\varphi$ ssi $q \not\models_\Sigma \varphi$
$q \models_\Sigma \varphi_1 \vee \varphi_2$ ssi $q \models_\Sigma \varphi_1$ ou $q \models_\Sigma \varphi_2$
$q \models_\Sigma z.\varphi$ ssi $q[z := 0] \models_\Sigma \varphi$
$q \models_\Sigma \varphi_1 \exists\mathcal{U} \varphi_2$ ssi $\exists \sigma \in \Sigma(q).\ \exists (q', i) \in \chi(\sigma).\ q' \models_\Sigma \varphi_2 \wedge \forall (q'', j) \preceq (q', i).\ q'' \models_\Sigma \varphi_1 \vee \varphi_2$
$q \models_\Sigma \varphi_1 \forall\mathcal{U} \varphi_2$ ssi $\forall \sigma \in \Sigma(q).\ \exists (q', i) \in \chi(\sigma).\ q' \models_\Sigma \varphi_2 \wedge \forall (q'', j) \preceq (q', i).\ q'' \models_\Sigma \varphi_1 \vee \varphi_2$

L'*ensemble caractéristique* d'une formule $\varphi$ relativement à l'ensemble de séquences $\Sigma$, noté $[\![\varphi]\!]_\Sigma$, est :

$$[\![\varphi]\!]_\Sigma = \{q \in \mathcal{Q} \mid q \models_\Sigma \varphi\}$$

i.e., l'ensemble des états qui satisfont $\varphi$ relativement à $\Sigma$. □

**Remarque 3.2** Dans la définition de la relation de satisfaction pour les formules $\varphi_1 \exists\mathcal{U} \varphi_2$ et $\varphi_1 \forall\mathcal{U} \varphi_2$, nous exigeons que la formule $\varphi_1 \vee \varphi_2$ soit vérifiée le long de la séquence, au lieu d'exiger que seulement $\varphi_1$ soit vérifiée comme dans le cas de CTL. Ceci est nécessaire pour éviter des "anomalies" dues à la densité du temps.

Considérons par exemple la formule $(x \leq 2) \exists\mathcal{U} (x > 2)$. Il paraît raisonnable qu'un état $q$ qui satisfait $x \leq 2$, possédant un successeur temporel $q + t$ qui satisfait $x + t > 2$, satisfasse la propriété. Pourtant, si seul la condition $x \leq 2$ est exigée, la formule ne serait pas satisfaite par $q$. En effet, entre $q$ et $q + t$ il y a une infinité d'états qui satisfont $x > 2$. Autrement dit, il n'existe pas un "premier" état qui satisfait $x > 2$ dont "tous" les prédécesseurs satisfont $x \leq 2$. □

*Exemple.* Considérons, par exemple, la formule :

$$z.((z < 2 \vee p_1) \exists\mathcal{U} (z = 5 \wedge p_2))$$

En appliquant la définition ci-dessus on a que $q$ satisfait la formule s'il existe une séquence $\sigma \in \Sigma(q[z := 0])$ telle que :

1. Il existe $i \geq 0$ et $(q', i) \in \chi(\sigma)$, tels que $q'(z) = 5$ et $q'(p_2) = \mathit{true}$, où $q'(z)$ est le temps $t'$ écoulé depuis le début de l'séquence. Par conséquent, la proposition $p_2$ est satisfaite exactement 5 unités de temps après le début de la séquence.

2. De plus, tout état $(q'', j)$ qui précède à $(q', i)$, satisfait $q''(z) < 2$ ou $q''(p_1) = \mathit{true}$ ou $q''(z) = 5$ et $q''(p_2) = \mathit{true}$, où $q''(z)$ est le temps $t''$ écoulé depuis le début de la séquence. De plus, $t'' \leq t'$. Par conséquent, la proposition $p_1$ est vraie pour $2 \leq t'' \leq 5$. En effet, puisque les transitions temporelles ne changent pas les valeurs des propositions, la proposition $p_1$ doit être satisfaite aussi à l'instant 5.

Donc, la formule exprime la propriété suivante : il existe une séquence où la proposition $p_2$ est vraie à l'instant 5 et la proposition $p_1$ est continuellement satisfaite dans l'intervalle de temps $[2, 5]$. □

### 3.2.2 Abréviations

Les opérateurs temporels additionnels comme $\exists\Diamond$, $\exists\Box$, $\forall\Diamond$ et $\forall\Box$ sont définis comme des abréviations à partir des opérateurs temporels $\exists\mathcal{U}$ et $\forall\mathcal{U}$ de la façon suivante :

- $\exists\Diamond\varphi$ pour $\mathit{true} \exists\mathcal{U} \varphi$,
- $\forall\Diamond\varphi$ pour $\mathit{true} \forall\mathcal{U} \varphi$,
- $\exists\Box\varphi$ pour $\neg\forall\Diamond\neg\varphi$,
- $\forall\Box\varphi$ pour $\neg\exists\Diamond\neg\varphi$.

Un exemple caractéristique de propriété temps-réel est celle de *réponse bornée* : un état qui satisfait $p_1$ est toujours suivi par un état qui satisfait $p_2$ avant $c$ unités de temps. Cette propriété s'exprime en TCTL par :

$$\forall\Box(p_1 \Rightarrow z.\forall\Diamond(p_2 \wedge z < c))$$

où l'opérateur d'initialisation d'horloge est introduit afin d'exprimer la contrainte "avant $c$ unités de temps à partir d'un état qui satisfait $p_1$."

Dans TCTL, les opérateurs temporels bornés [ACD90] sont définis comme des abréviations. Soient # un symbole de relation dans l'ensemble $\{<, \leq, >, \geq, =\}$, $c \in \mathsf{N}$, $\varphi_1$ et $\varphi_2$ deux formules de TCTL et $z$ une horloge qui n'apparaît dans aucune des deux formules. On définit :

$$\varphi_1 \, \exists\mathcal{U}_{\# \, c} \, \varphi_2 \quad \text{et} \quad \varphi_1 \, \forall\mathcal{U}_{\# \, c} \, \varphi_2$$

comme des abréviations de :

$$z.(\varphi_1 \, \exists\mathcal{U} \, (\varphi_2 \wedge z \# c)) \quad \text{et} \quad z.(\varphi_1 \, \forall\mathcal{U} \, (\varphi_2 \wedge z \# c))$$

Par exemple, la propriété décrite ci-dessus s'exprime en terme des opérateurs bornés par :

$$\forall\Box(p_1 \Rightarrow \forall\Diamond_{<c} p_2)$$

D'autres propriétés peuvent être spécifiées au moyen des opérateurs temporels bornés. Par exemple :

- La formule $\exists\Diamond_{>3} p$ dit qu'il existe une exécution où la proposition $p$ est satisfaite au moins une fois après l'instant 3.
- La formule $\forall\Box_{\leq 5} p$ spécifie que $p$ est un invariant jusqu'à l'instant 5.
- La formule $\forall\Diamond_{\leq 3} p$ dit qu'il est inévitable d'atteindre un état où $p$ est vraie dans au plus 3 unités de temps.

Ces abréviations permettent d'exprimer certaines propriétés plus succinctement.

La syntaxe que nous avons adoptée pour TCTL est plus expressive que la notation avec des opérateurs bornés. Par exemple, la formule :

$$x.\exists\Diamond(p_1 \wedge \exists\Diamond(p_2 \wedge z.\exists\Diamond(p_3 \wedge z < x + 2)))$$

dit qu'il existe une exécution avec un état $q_1$ qui satisfait la proposition $p_1$, suivi d'un état $q_2$ qui satisfait $p_2$, suivit d'un état $q_3$ qui satisfait $p_3$, avant 2 unités de temps depuis $q_1$. Cette propriété ne peut pas être décrite au moyen des seuls opérateurs bornés.

**Remarque 3.3** La version de TCTL définie ici diffère légèrement de la définition donnée dans [Alu91]. Cette différence réside dans l'opérateur d'initialisation. Nous préférons cet opérateur au quantificateur "freeze" qui apparaît dans la définition de TCTL dans [Alu91], car la notion de remise à 0 est plus adaptée à la structure de nos modèles. Dans l'approche "freeze", la formule $z.\varphi$ affecte à $z$ le temps courant. De toutes façons, il est montré dans [Alu91] que les deux formalismes sont équivalents. □

## 3.3 Caractérisation par point fixe

Dans cette section nous présentons une caractérisation par point fixe des opérateurs temporels de TCTL. Cette caractérisation repose essentiellement sur la définition d'un opérateur "état suivant" approprié.

Etant donné que les treillis des prédicats unaires sur l'ensemble $\mathcal{Q}$ avec les opérateurs logiques $\vee$, $\wedge$ et $\neg$ est isomorphe à $\mathbf{2}^{\mathcal{Q}}$ avec l'union, l'intersection et la complémentation, par commodité dans la présentation les deux structures sont souvent identifiées.

De plus, étant donné que les formules sont interprétées sur l'ensemble des états d'un modèle et qu'une formule caractérise l'ensemble des états qui la satisfont, nous identifions par la suite une formule et son ensemble caractéristique et nous interprétons les opérateurs de la logique comme des opérateurs sur les ensembles d'états. De même, afin de simplifier les preuves, les propositions sont énoncées pour des ensembles caractéristiques au lieu des formules, sauf dans les cas où nous voulons dire explicitement que les résultats sont valides pour une certaine classe de formules.

### 3.3.1 L'opérateur "état suivant"

Il est bien connu que les opérateurs temporels de CTL sont exprimables comme des points fixes de fonctionnelles définies au moyen de l'opérateur "état suivant" noté $\bigcirc$.

Pour un ensemble d'états $Q$, $\bigcirc Q$ est l'ensemble des états possédant une transition vers un état de $Q$. Cette définition capte la notion de "pas" lorsque les formules sont interprétées sur un ensemble de séquences d'états discrètes.

A l'aide de l'opérateur $\bigcirc$, l'ensemble des états qui satisfont de la formule :

$$\varphi_1 \,\exists\mathcal{U}\, \varphi_2$$

de CTL, est caractérisé par le plus petit point fixe de la fonctionnelle monotone $F(X)$ définie par :

$$F(X) = Q_2 \vee (Q_1 \wedge \bigcirc X)$$

où $Q_i$ est l'ensemble caractéristique de $\varphi_i$, pour $i = 1, 2$. Autrement dit, l'ensemble des états qui satisfont la formule $\varphi_1 \,\exists\mathcal{U}\, \varphi_2$ est l'ensemble $Q$ tel que $Q$ est le plus petit point fixe de $F$, c'est-à-dire, $Q$ satisfait $Q = F(Q)$ et pour tout ensemble d'états $Q'$ tel que $Q' = F(Q')$, on a $Q \subseteq Q'$.

Par la suite, le plus petit point fixe d'une fonctionnelle $F$ défini sur les ensembles d'états $\mathbf{2}^{\mathcal{Q}}$ est noté $\mu X.F(X)$.

Dans notre cas, pour suivre une approche similaire il faut définir un opérateur "état suivant" approprié. Puisque le temps est dense, cet opérateur ne doit pas forcer le temps à avancer de plus qu'une fraction infinitésimale. Autrement dit, cet opérateur ne doit pas imposé une granularité déterminée à l'écoulement du temps. De plus, son itération doit permettre au temps de progresser au-delà de tout instant.

L'exemple suivant montre qu'il est nécessaire de disposer d'un opérateur "état suivant" *binaire* pour exprimer une propriété comme $\varphi_1 \,\exists\mathcal{U}\, \varphi_2$, car il faut assurer que $\varphi_1$ est *continuellement* satisfaite jusqu'à ce que $\varphi_2$ soit vraie.

*Exemple.* Considérons la formule $(x \leq 2 \vee x \geq 4) \,\exists\mathcal{U}\, (x \geq 5)$. Soit $Q_1$ l'ensemble caractéristique de $x \leq 2 \vee x \geq 4$ et $Q_2$ l'ensemble caractéristique de $x \geq 5$. Supposons que nous avons défini un opérateur "état suivant" $\bigcirc$ unaire, et supposons que tous les états qui satisfont $x < 5$ ont un successeur temporel qui satisfait $x \geq 5$. Donc, l'ensemble caractéristique de $x < 5$ est contenu dans $\bigcirc Q_2$. Par conséquent, $Q_1 \wedge \bigcirc Q_2$ contient tous les états qui satisfont $x \leq 2$. Pourtant, il n'est pas possible d'aller d'un état qui satisfait $x \leq 2$ à un état qui satisfait $x \geq 5$ sans passer par un état qui n'appartient pas à $Q_1$. □

**Définition 3.9** Soit $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ un système temporisé et soit $Q_1, Q_2 \subseteq \mathcal{Q}$. Pour tout $q \in \mathcal{Q}$,

$$Q_1 \triangleright Q_2$$

est l'ensemble des états tels que $q \in Q_1 \triangleright Q_2$ s'il existe $q' \in Q_2$ et $t \in \mathsf{R}^+$ tels que $q \overset{t}{\triangleright} q'$, et pour tout $t' \leq t$, $q + t' \in Q_1 \vee Q_2$. □

Pour $Q_1$ et $Q_2$, $Q_1 \triangleright Q_2$ caractérise l'ensemble des états pour lesquels il existe un état accessible en un pas qui est dans $Q_2$ et tous les états intermédiaires sont dans $Q_1$ ou $Q_2$.

**Remarque 3.4** Dans la définition de l'opérateur $\triangleright$ nous exigeons $Q_1 \vee Q_2$ pour éviter des anomalies dues à la densité du temps (cf. remarque 3.2). □

**Proposition 3.1** *Pour tout $Q_1, Q_2, Q_1', Q_2' \subseteq \mathcal{Q}$ tels que, pour $i = 1, 2$, $Q_i \subseteq Q_i'$, on a :*

$$Q_1 \triangleright Q_2 \subseteq Q_1' \triangleright Q_2'$$

**Preuve.** Soit $q \in Q_1 \triangleright Q_2$. Alors, il existe $q' \in Q_2$ et $t \in \mathsf{R}^+$ tels que $q \overset{t}{\triangleright} q'$ et pour tout $t' \leq t$, $q + t' \in Q_1 \vee Q_2$. Donc, $q' \in Q_2'$ et pour tout $t' \leq t$, $q + t' \in Q_1' \vee Q_2'$. Par conséquent, $q \in Q_1' \triangleright Q_2'$. ■

### 3.3.2 Caractérisation de l'opérateur $\exists\mathcal{U}$

La proposition suivante établit le fait que pour un système bien temporisé et deux ensembles d'états $Q_1$ et $Q_2$, la signification de $Q_1 \,\exists\mathcal{U}\, Q_2$ sur l'ensemble de toutes les séquences est la même que sur l'ensemble des séquences divergentes.

**Proposition 3.2** *Soient $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ un système bien temporisé et $Q_1, Q_2 \subseteq \mathcal{Q}$. Alors,*

$$[\![Q_1 \,\exists\mathcal{U}\, Q_2]\!]_{\Sigma_T} = [\![Q_1 \,\exists\mathcal{U}\, Q_2]\!]_{\Delta_T}$$

**Preuve.** Evidemment, $[\![Q_1 \,\exists\mathcal{U}\, Q_2]\!]_{\Delta_T} \subseteq [\![Q_1 \,\exists\mathcal{U}\, Q_2]\!]_{\Sigma_T}$ car $\Delta_T \subseteq \Sigma_T$.

Soit $q \in [\![Q_1 \,\exists\mathcal{U}\, Q_2]\!]_{\Sigma_T}$. Alors, il existe une séquence $\sigma = q \overset{t_0}{\triangleright} \ldots q_i \overset{t_i}{\triangleright} \ldots \in \Sigma_T$ telle que $q_i \in Q_2$, et pour tout $(q'', j) \preceq (q_i, i)$, $q'' \in Q_1 \vee Q_2$. Comme $T$ est bien temporisé, il existe une séquence $\sigma' \in \Delta_T$ telle que $q \overset{t_0}{\triangleright} \ldots q_i \overset{t_i}{\triangleright}$ est préfixe de $\sigma'$. Donc, $q \in [\![Q_1 \,\exists\mathcal{U}\, Q_2]\!]_{\Delta_T}$. ■

La proposition 3.2 est vraie pour deux ensembles d'états $Q_1$ et $Q_2$ quelconques. Ce résultat peut être étendu pour des formules $\varphi_1$ et $\varphi_2$ de TCTL telles que l'ensemble caractéristique de $\varphi_i$ relativement à $\Sigma_T$ est égal à son ensemble caractéristique relativement à $\Delta_T$.

**Proposition 3.3** *Soient $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ un système bien temporisé et $\varphi_1$ et $\varphi_2$ deux formules de* TCTL *telles que $[\![\varphi_i]\!]_{\Sigma_T} = [\![\varphi_i]\!]_{\Delta_T}$. Alors,*

$$[\![\varphi_1 \,\exists\mathcal{U}\, \varphi_2]\!]_{\Sigma_T} = [\![\varphi_1 \,\exists\mathcal{U}\, \varphi_2]\!]_{\Delta_T}$$

Autrement dit, pour tout système bien temporisé $T$ et toute formule $\varphi$ du fragment de TCTL dont les formules sont construites à partir des prédicats d'état, des opérateurs logiques et de l'opérateur $\exists\mathcal{U}$, nous avons $[\![\varphi]\!]_{\Delta_T} = [\![\varphi]\!]_{\Sigma_T}$.

**Théorème 3.1** *Soient $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ un système bien temporisé et $\varphi$ une formule de* TCTL *telle que l'opérateur $\forall\mathcal{U}$ n'apparaît dans aucune sous-formule. Alors,*

$$[\![\varphi]\!]_{\Sigma_T} = [\![\varphi]\!]_{\Delta_T}$$

La proposition suivante montre que pour tout système temporisé $T$, la propriété $Q_1 \exists\mathcal{U} Q_2$ évaluée sur l'ensemble de séquences $\Sigma_T$ est exprimable comme $\mu X.(Q_2 \vee Q_1 \triangleright X)$.

**Proposition 3.4** *Soient $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ un système temporisé et $Q_1, Q_2 \subseteq \mathcal{Q}$. Alors,*

$$[\![Q_1 \exists\mathcal{U} Q_2]\!]_{\Sigma_T} = \mu X.(Q_2 \vee Q_1 \triangleright X)$$

**Preuve.** Soit $Q = [\![Q_1 \exists\mathcal{U} Q_2]\!]_{\Sigma_T}$.

Tout d'abord, nous montrons que $Q$ est un point fixe de la fonctionnelle $F$ définit par :

$$F(X) = Q_2 \vee Q_1 \triangleright X$$

Ensuite nous montrons que $Q$ est le plus petit point fixe de $F$.

**1.** Nous montrons $Q = Q_2 \vee Q_1 \triangleright Q$.

$\subseteq$ Soit $q \in Q$. Il existe donc une séquence

$$\sigma = q_0 \overset{t_0}{\triangleright} q_1 \overset{t_1}{\triangleright} \ldots \in \Sigma_T$$

avec $q = q_0$ et $(q', i) \in \chi(\sigma)$ tel que :

- $q' \in Q_2$, et
- pour tout $(q'', j) \preceq (q', i)$, $q'' \in Q_1 \vee Q_2$.

Comme $Q_2 \subseteq Q$ on a :

- $q' \in Q$, et
- pour tout $(q'', j) \preceq (q', i)$, $q'' \in Q_1 \vee Q$.

Or, $(q', i) \in \chi_i(\sigma)$. Donc, il existe $t \leq t_i$ tel que $q' = q_i + t$, et par conséquent, $q_i \in Q_1 \triangleright Q$. Par le même raisonnement on déduit que pour tout $j \leq i$, $q_j \in Q_1 \triangleright Q$.

Donc, $q \in Q_1 \triangleright Q$ et par conséquent $q \in Q_2 \vee Q_1 \triangleright Q$.

$\supseteq$ Soit $q \in Q_2 \vee Q_1 \triangleright Q$.

Si $q \in Q_2$ alors $q \in Q$.

Supposons $q \in Q_1 \triangleright Q$. Il existe $q_0 \in Q$ et $t \geq 0$ tels que :

- $q \overset{t}{\triangleright} q_0$, et
- pour tout $t' \leq t$, $q + t \in Q_1 \vee Q$.

Comme $Q \subseteq Q_1 \vee Q_2$ on a :

- $q_0 \in Q_1 \vee Q_2$, et

– pour tout $t' \leq t$, $q_0 + t \in Q_1 \vee Q_2$.

Comme $q_0 \in Q$, il existe une séquence

$$\sigma = q_0 \overset{t_0}{\triangleright} q_1 \overset{t_1}{\triangleright} \ldots \in \Sigma_T$$

et $(q', i) \in \chi(\sigma)$ tel que :

– $q' \in Q_2$, et
– pour tout $(q'', j) \preceq (q', i)$ $q'' \in Q_1 \vee Q_2$.

Donc, la séquence

$$\sigma' = q \overset{t}{\triangleright} q_0 \overset{t_0}{\triangleright} q_1 \overset{t_1}{\triangleright} \ldots \in \Sigma_T$$

satisfait la définition de $Q_1 \exists\mathcal{U} Q_2$, et par conséquent, $q \in Q$.

**2.** Soit $B \subseteq \mathcal{Q}$ tel que $B = Q_2 \vee Q_1 \triangleright B$. Il faut montrer $Q \subseteq B$.
Soit $q \in Q$. Il existe donc une séquence

$$\sigma = q_0 \overset{t_0}{\triangleright} q_1 \overset{t_1}{\triangleright} \ldots \in \Sigma_T$$

avec $q = q_0$ et $(q', i) \in \chi(\sigma)$ tel que :

- $q' \in Q_2$, et
- pour tout $(q'', j) \preceq (q', i)$, $q'' \in Q_1 \vee Q_2$.

Puisque $Q_2 \subseteq B$, on a :

- $q' \in B$, et
- pour tout $(q'', j) \preceq (q', i)$, $q'' \in Q_1 \vee B$.

Or, $(q', i) \in \chi_i(\sigma)$. Donc, il existe $t \leq t_i$ tel que $q' = q_i + t$, et par conséquent, $q_i \overset{t}{\triangleright} q'$. On a donc, $q_i \in Q_1 \triangleright B$ et $q_i \in B$. Par le même raisonnement on déduit que pour tout $j \leq i$, $q_j \in B$.
Donc, $q \in B$. ■

La proposition 3.4 donne donc une caractérisation de l'opérateur $\exists\mathcal{U}$ pour TCTL similaire à celle déjà connue pour le même opérateur dans le cas de CTL. Le théorème 3.2 suit des propositions 3.3 et 3.4.

**Théorème 3.2** *Soient $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ un système bien temporisé et $\varphi_1$ et $\varphi_2$ deux formules de* TCTL *avec $[\![\varphi_i]\!]_{\Delta_T} = Q_i$ pour $i = 1, 2$. Alors,*

$$[\![\varphi_1 \exists\mathcal{U} \varphi_2]\!]_{\Delta_T} = \mu X.(Q_2 \vee Q_1 \triangleright X)$$

### 3.3.3 Caractérisation de l'opérateur $\forall\mathcal{U}$

Nous disposons d'une caractérisation en termes de point fixe de l'opérateur $\exists\mathcal{U}$. Nous voulons à présent caractériser également l'opérateur $\forall\mathcal{U}$. Tout d'abord, nous constatons qu'à la différence de la formule $\varphi_1 \exists\mathcal{U} \varphi_2$, la formule $\varphi_1 \forall\mathcal{U} \varphi_2$ n'a pas la même signification relativement à l'ensemble de toutes les séquences et à l'ensemble de séquences divergentes, même dans le cas des systèmes bien temporisés. En effet, la proposition suivante montre que la formule $\varphi_1 \forall\mathcal{U} \varphi_2$ n'a le sens attendu que lorsqu'elle est évaluée sur les séquences divergentes.

**Proposition 3.5** *Soient $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ un système temporisé et $Q_1, Q_2 \subseteq \mathcal{Q}$. Alors,*

$$[\![Q_1 \forall\mathcal{U} Q_2]\!]_{\Sigma_T} = Q_2$$

**Preuve.** Soit $Q = [\![Q_1 \forall\mathcal{U} Q_2]\!]_{\Sigma_T}$.

- $\subseteq$ Soit $q \in Q$. La séquence $\sigma = q \overset{0}{\triangleright} q \overset{0}{\triangleright} \ldots \in \Sigma_T(q)$. Il existe donc $(q', i) \in \chi(\sigma)$ tel que $q' \in Q_2$. Mais $q'$ est nécessairement $q$. Par conséquent, $q \in Q_2$.
- $\supseteq$ Soit $q \in Q_2$. Toute séquence $\sigma \in \Sigma_T(q)$ est telle que $(q, 0) \in \chi(\sigma)$ et $q \in Q_2$. Donc, $q \in Q$.

■

Dans le cas de CTL, la formule :

$$\varphi_1 \forall\mathcal{U} \varphi_2$$

est caractérisée au moyen de l'opérateur dual de $\bigcirc$ comme le plus petit point fixe suivant :

$$\mu X.(Q_2 \vee Q_1 \wedge \bigcirc X \wedge \neg \bigcirc \neg X)$$

où $Q_i$ est l'ensemble caractéristique de $\varphi_i$, pour $i = 1, 2$.

Nous constatons que, dans notre cas, l'opérateur dual de $\triangleright$ n'est d'aucune utilité. En effet, à cause de la densité du temps, un état peut avoir un successeur qui satisfait $\varphi$, alors que tous les états intermédiaires satisfont $\neg\varphi$.

*Exemple.* Soit $q$ un état qui satisfait $true \triangleright x = 2$ et dont tous les successeurs sont des successeurs temporels. Donc, pour aller de $q$ à l'état $q + t$ qui satisfait $x = 2$, tous les états intermédiaires satisfont $x < 2$. Autrement dit, l'état $q$ atteint *inévitablement en un pas* un état qui satisfait $x = 2$, mais *tout au long du pas* sauf à la fin, la propriété $x < 2$ est vraie. Dans ce cas, l'état $q$ n'appartient pas à $\neg(true \triangleright x \neq 2)$, pourtant $q$ satisfait $\forall\Diamond x = 2$. □

Il est cependant possible de caractériser l'opérateur $\forall\mathcal{U}$ en termes de point fixe. Nous montrons tout d'abord que l'*inévitabilité bornée* s'exprime en termes de *possibilité* si on se restreint aux séquences divergentes d'un système bien temporisé. Nous montrons ensuite comment caractériser l'inévitabilité en termes de point fixe et de l'inévitabilité bornée.

Afin de prouver ces résultats nous introduisons la définition suivante.

**Définition 3.10** Un *état symbolique* est un couple $(s, \psi)$ où $s \in S$ est un sommet de $G$ et $\psi \in \Psi(\mathcal{H})$ est une contrainte temporelle. □

L'état symbolique $(s, \psi)$ représente l'ensemble des états $(s, v) \in \mathcal{Q}$ tels que $v$ satisfait $\psi$. On définit :

$$[\![(s, \psi)]\!] = \{(s, v) \in \mathcal{Q} \mid v \text{ satisfait } \psi\}$$

Nous avons le lemme suivant.

**Lemme 3.1** *Soient $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ un système temporisé, et $q, q' \in \mathcal{Q}$ tels que $q \triangleright q'$. Pour tout état symbolique $(s, \psi)$, la valeur de $\psi$ change un nombre fini de fois dans le pas de $q$ à $q'$.*

**Preuve.** Par induction sur la structure de $\psi$. ■

L'union des états symboliques $\bigcup_{s \in S}(s, \psi_s)$ représente l'ensemble des états $(s, v)$ tels que $v$ satisfait $\psi_s$. On a :

$$[\![\bigcup_{s \in S}(s, \psi_s)]\!] = \bigcup_{s \in S} [\![(s, \psi_s)]\!]$$

Par la suite, nous supposons qu'un *ensemble caractéristique* $Q \subseteq \mathcal{Q}$ est représenté par une union d'états symboliques, i.e., $Q = \bigcup_{s \in S} [\![(s, \psi_s)]\!]$. Imposer cette condition n'est pas restrictif, car comme nous le montrons dans le chapitre 4, l'ensemble caractéristique d'une formule de TCTL, relativement à l'ensemble de séquences divergentes d'un graphe bien temporisé, est représentable comme une union d'états symboliques (cf. section 4.4).

### L'inévitabilité bornée

Afin de simplifier la présentation nous considérons en premier lieu l'opérateur $\forall\Diamond$. La proposition suivante montre que, pour un système bien temporisé $T$, la propriété :

*il est inévitable de satisfaire $\varphi$ avant $c$ unités de temps*

évaluée sur l'ensemble des séquences divergentes de $T$, est équivalente à la propriété :

*il n'est pas possible de dépasser l'instant $c$ sans avoir satisfait $\varphi$ avant*

évaluée sur l'ensemble de toutes les séquences de $T$.

**Proposition 3.6** *Soient $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ un système bien temporisé, $Q \subseteq \mathcal{Q}$ et $c \in \mathbb{N}$, $c > 0$. Alors,*

$$[\![z.\forall\Diamond(Q \wedge z \leq c)]\!]_{\Delta_T} = [\![\neg z.(\neg Q\, \exists\mathcal{U}\, z > c)]\!]_{\Sigma_T}$$

**Preuve.** Soient $Q_1 = [\![z.\forall\Diamond(Q \wedge z \leq c)]\!]_{\Delta_T}$ et $Q_2 = [\![\neg z.(\neg Q\, \exists\mathcal{U}\, z > c)]\!]_{\Sigma_T}$.

$\subseteq$ Soit $q \in Q_1$. Sans perte de généralité on peut supposer $q(z) = 0$. Pour toute séquence $\sigma = q \overset{t_0}{\triangleright} q_1 \overset{t_1}{\triangleright} \ldots \in \Delta_T$,

$$\text{il existe } (q', i) \in \chi(\sigma) \text{ tel que } q' \in Q \text{ et } q'(z) \leq c. \tag{3.1}$$

Supposons que $q \notin Q_2$. Donc, il existe une séquence $\bar{\sigma} = q \overset{\bar{t}_0}{\triangleright} \bar{q}_1 \overset{\bar{t}_1}{\triangleright} \ldots \in \Sigma_T$ et $(\bar{q}', k) \in \chi(\bar{\sigma})$ tel que :

- $\bar{q}'(z) > c$, et

– pour tout $(\bar{q}'', j) \preceq (\bar{q}', k)$, $\bar{q}''(z) > c$ ou $\bar{q}'' \notin Q$.

On a donc :

$$\text{pour tout } (\bar{q}'', j) \preceq (\bar{q}', k), \text{ si } \bar{q}''(z) \leq c \text{ alors } \bar{q}'' \notin Q. \tag{3.2}$$

La séquence $\bar{\sigma}$ n'est pas forcement divergente. Or, $T$ est bien temporisé. Donc, il existe $\hat{\sigma} \in \Delta_T$ telle que $q \overset{\bar{t}_0}{\triangleright} \bar{q}_1 \ldots \bar{q}_i \overset{\bar{t}_i}{\triangleright}$ est un préfixe de $\hat{\sigma}$.

De 3.1 on déduit :

$$\text{il existe } (q', i) \in \chi(\hat{\sigma}) \text{ tel que } q' \in Q \text{ et } q'(z) \leq c. \tag{3.3}$$

On a deux cas :

1. Si $(q', i) \preceq (\bar{q}', k)$ alors par 3.2 on a $q' \notin Q$, ce qui contredit 3.3.
2. Si $(\bar{q}', k) \preceq (q', i)$ alors $q'(z) > c$, ce qui contredit 3.3.

Dans les deux cas, on aboutit à une contradiction.

Par conséquent, $q \in Q_2$.

$\supseteq$ Soit $q \in Q_2$. Sans perte de généralité on peut supposer $q(z) = 0$. Alors, pour toute séquence $\sigma \in \Sigma_T(q)$,

$$\begin{array}{l}\text{pour tout } (q', i) \in \chi(\sigma), \\ \quad \text{si } q'(z) > c \text{ alors il existe } (q'', j) \preceq (q', i), \text{ tel que } q''(z) \leq c \text{ et } q'' \in Q.\end{array} \tag{3.4}$$

Supposons que $q \notin Q_1$. Alors, il existe une séquence divergente $\bar{\sigma} \in \Delta_T(q)$ telle que

$$\text{pour tout } (q', i) \in \chi(\bar{\sigma}), \text{ si } q'(z) \leq c \text{ alors } q' \notin Q. \tag{3.5}$$

Or, $\bar{\sigma}$ est divergente. Donc :

$$\text{il existe } (q', i) \in \chi(\bar{\sigma}) \text{ tel que } q'(z) > c. \tag{3.6}$$

A partir de 3.4 et 3.6 on déduit :

$$\text{il existe } (q'', j) \preceq (q', i), \text{ tel que } q''(z) \leq c \text{ et } q'' \in Q. \tag{3.7}$$

Mais 3.7 contredit 3.5. On aboutit à une contradiction.

Donc, $q \in Q_1$.

Par conséquent, $Q_1 = Q_2$. ■

Soit $\varphi$ une formule de TCTL qui caractérise le même ensemble d'états relativement à $\Sigma_T$ et $\Delta_T$. La proposition 3.6 comporte deux résultats. Tout d'abord, elle construit une formule $\varphi'$ en termes de l'opérateur $\exists\mathcal{U}$ dont l'ensemble caractéristique est le même que celui de la formule $\forall\Diamond_{\leq c}\varphi$ relativement aux séquences divergentes. De plus, elle établie le fait que si $T$ est un système bien temporisé, l'ensemble caractéristique de $\varphi'$ est le même, soit calculé sur toutes les séquences, soit calculé sur les séquences divergentes seulement.

A partir de la proposition 3.6 et du théorème 3.2 nous avons le théorème suivant.

**Théorème 3.3** *Soient $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ un système bien temporisé, $c \in \mathbb{N}$ tel que $c > 0$, et $\varphi$ une formule de* TCTL *avec $[\![\varphi]\!]_{\Delta_T} = Q$. Alors,*

$$[\![z.\forall\Diamond(\varphi \wedge z \leq c)]\!]_{\Delta_T} = \neg z.\mu X.(z > c \vee \neg Q \triangleright X)$$

Nous avons donc caractérisé les propriétés d'inévitabilité bornée exprimées au moyen de l'opérateur $\forall\Diamond_{\leq}$. En suivant les mêmes idées il est possible de caractériser l'opérateur plus général $\forall\mathcal{U}_{\leq}$. La proposition 3.7 montre que, pour un système bien temporisé $T$, la propriété :

> *il est inévitable de satisfaire $\varphi_1$ avant $c$ unités de temps, $\varphi_1 \vee \varphi_2$ étant auparavant continûment satisfaite*

évaluée sur l'ensemble des séquences divergentes de $T$, est équivalente à la propriété :

> *il n'est pas possible de ne satisfaire ni $\varphi_1$ ni $\varphi_2$ ou de dépasser l'instant $c$ sans avoir satisfait $\varphi_2$ avant l'instant $c$*

évaluée sur l'ensemble de toutes les séquences de $T$.

**Proposition 3.7** *Soient $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ un système bien temporisé, $Q_1, Q_2 \subseteq \mathcal{Q}$ deux ensembles caractéristiques et $c \in \mathbb{N}$ tel que $c > 0$. Alors,*

$$[\![z.(Q_1 \forall\mathcal{U} (Q_2 \wedge z \leq c))]\!]_{\Delta_T} = [\![\neg z.(\neg Q_2 \exists\mathcal{U} (\neg(Q_1 \vee Q_2) \vee z > c))]\!]_{\Sigma_T}$$

**Preuve.** Soit $Q = [\![z.(Q_1 \forall\mathcal{U} (Q_2 \wedge z \leq c))]\!]_{\Delta_T}$ et $\hat{Q} = [\![\neg z.\neg Q_2 \exists\mathcal{U} (\neg(Q_1 \vee Q_2) \vee z > c)]\!]_{\Sigma_T}$.

- $\subseteq$ Suit de la même manière que $\subseteq$ dans la preuve de la proposition 3.6.

- $\supseteq$ Soit $q \in \hat{Q}$. Sans perte de généralité on peut supposer $q(z) = 0$. Alors, pour toute séquence $\sigma \in \Sigma_T(q)$, pour tout $(q', i) \in \chi(\sigma)$, $q' \in Q_1 \vee Q_2$ et $q'(z) \leq c$, ou il existe $(q'', j) \preceq (q', i)$, tel que $q'' \in Q_2$ et $q''(z) \leq c$.

  C'est-a-dire :

  pour tout $(q', i) \in \chi(\sigma)$,
  si $q' \notin Q_1 \vee Q_2$ ou $q'(z) > c$,
  alors il existe $(q'', j) \preceq (q', i)$, tel que $q'' \in Q_2$ et $q''(z) \leq c$. (3.8)

  Supposons que $q \notin Q$. Alors, il existe une séquence divergente $\bar{\sigma} \in \Delta_T(q[z := 0])$ telle que :

  pour tout $(q', i) \in \chi(\bar{\sigma})$,
  si $q' \in Q_2$ et $q'(z) \leq c$,
  alors il existe $(q'', j) \preceq (q', i)$, tel que $q'' \notin Q_1 \vee Q_2$ et $q''(z) \leq c$. (3.9)

  Or, $\bar{\sigma}$ est divergente. Donc :

  il existe $(q_0, i_0) \in \chi(\bar{\sigma})$ tel que $q_0(z) > c$. (3.10)

  De 3.8 et 3.10 on déduit :

  il existe $(q_1, i_1) \preceq (q_0, i_0)$ tel que $q_1 \in Q_2$ et $q_1(z) \leq c$. (3.11)

De 3.9 et 3.11 on déduit :

pour tout $(q', i) \preceq (q_1, i_1)$,
si $q' \in Q_2$ et $q'(z) \leq c$,
alors il existe $(q'', j) \preceq (q', i)$, tel que $q'' \notin Q_1 \vee Q_2$ et $q''(z) \leq c$. (3.12)

De 3.8 et 3.11 on déduit :

pour tout $(q', i) \preceq (q_1, i_1)$,
si $q' \notin Q_1 \vee Q_2$ et $q'(z) \leq c$,
alors il existe $(q'', j) \preceq (q', i)$, tel que $q'' \in Q_2$ et $q''(z) \leq c$. (3.13)

D'après le lemme 3.1, on peut partitionner l'intervalle de temps $[0, q_1(z)]$ en $n \geq 1$ intervalles $I_1, \ldots, I_n$, tels que pour tout $k = 1, \ldots, n$, tous les états $q'$ tels que $(q', i) \preceq (q_1, i_1)$ et $q'(z) \in I_k$ sont, soit dans $Q_2$, soit dans $Q_1 \wedge \neg Q_2$, soit dans $\neg(Q_1 \vee Q_2)$.

De 3.11 et 3.12 on déduit qu'il existe un $k$ tel que tous les états $q'$ tels que $(q', i) \in I_k$ sont tels que $q' \notin Q_1 \vee Q_2$. Soit $k$ le premier indice tel que $I_k$ satisfait cette propriété. Donc :

pour tout $l < k$, pour tout $(q'', j) \in I_l$ on a $q'' \in Q_1 \vee Q_2$. (3.14)

Soit $(q', i) \in I_k$. De 3.13 on déduit :

il existe $(q'', j) \preceq (q', i)$ tel que $q'' \in Q_2$ et $q''(z) \in I_h$ avec $h < k$. (3.15)

De 3.11 on déduit :

il existe $(q''', m) \preceq (q'', j)$ tel que $q''' \notin Q_1 \vee Q_2$ et $q'''(z) \in I_l$ avec $l < h$. (3.16)

Ce qui contredit 3.14. On aboutit à une contradiction.

Donc, $q \in Q$.

Par conséquent, $Q = \hat{Q}$. ■

A partir de la proposition 3.7 et du théorème 3.2 nous avons le théorème suivant.

**Théorème 3.4** *Soient $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ un système bien temporisé, $c \in \mathbb{N}$ tel que $c > 0$, et $\varphi_1$ et $\varphi_2$ deux formules de* TCTL *avec $[\![\varphi_i]\!]_{\Delta_T} = Q_i$, pour $i = 1, 2$. Alors,*

$$[\![z.\varphi_1 \forall\mathcal{U} (\varphi_2 \wedge z \leq c)]\!]_{\Delta_T} = \neg z.\mu X.(\neg(Q_1 \wedge Q_2) \vee z > c \vee \neg Q_2 \triangleright X)$$

### L'inévitabilité non bornée

Nous avons donc caractérisé l'inévitabilité bornée en termes de point fixe pour les systèmes bien temporisés. Nous montrons à présent comment caractériser l'inévitabilité en termes de point fixe et de l'inévitabilité bornée. Comme nous l'avons fait précédemment, nous présentons tout d'abord le cas particulier de l'opérateur $\forall\Diamond$ pour lequel la preuve est plus simple. Nous traitons ensuite le cas plus général.

**Proposition 3.8** *Soient $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ un système temporisé, $Q_1 \subseteq \mathcal{Q}$ un ensemble caractéristique et $c \in \mathbb{N}$ tel que $c > 0$. Alors,*

$$[\![\forall\Diamond Q_1]\!]_{\Delta_T} = \mu X.(Q_1 \vee z.\forall\Diamond(X \wedge z \leq c))$$

**Preuve.** Soit $Q = [\![\forall\Diamond Q_1]\!]_{\Delta_T}$.

Tout d'abord, on prouve que $Q$ est un point fixe de la fonctionnelle $F$ définie par :

$$F(X) = Q_1 \vee z.\forall\Diamond(X \wedge z \leq c)$$

et ensuite que $Q$ est le plus petit.

**1.** Soit $\hat{Q} = Q_1 \vee z.\forall\Diamond(Q \wedge z \leq c)$. Il faut montrer $Q = \hat{Q}$.

- $\subseteq$ Supposons $q \notin \hat{Q}$. Sans perte de généralité on peut supposer $q(z) = 0$. Donc, il existe $\sigma \in \Delta_T(q)$, telle que pour tout $(q', i) \in \chi(\sigma)$, si $q'(z) \leq c$ alors $q' \notin Q$. Par conséquent, $q \notin Q$. Donc, $Q \subseteq \hat{Q}$.

- $\supseteq$ Soit $q \in \hat{Q}$. Sans perte de généralité on peut supposer $q(z) = 0$.

  Si $q \in Q_1$ alors $q \in Q$. Supposons que $q \in z.\forall\Diamond(Q \wedge z \leq c)$. Donc, pour tout $\sigma \in \Delta_T(q)$, il existe $(q', i) \in \chi(\sigma)$ tel que $q' \in Q$ et $q'(z) \leq c$.

  Puisque $q' \in Q$, il existe $(q'', j) \in \chi(\sigma)$ tel que $(q', i) \preceq (q'', j)$ et $q'' \in Q_1$. Par conséquent $q \in Q$.

  Donc, $\hat{Q} \subseteq Q$.

**2.** Soit $B = Q_1 \vee z.\forall\Diamond(B \wedge z \leq c)$.

Supposons $q \notin B$. Sans perte de généralité on peut supposer $q(z) = 0$. Donc, $q \notin Q_1$ et $q \notin z.\forall\Diamond(B \wedge z \leq c)$.

Par conséquent, il existe $\sigma \in \Delta_T(q)$, telle que pour tout $(q', i) \in \chi(\sigma)$, si $q'(z) \leq c$ alors $q' \notin B$.

Or, $\sigma$ est divergente. Donc, il existe $(q', i) \in \chi(\sigma)$ tel que $q'(z) = c$ et pour tout $(q'', j) \preceq (q', i)$, $q'' \notin B$. Par conséquent, pour tout $(q'', j) \preceq (q', i)$, on a $q'' \notin Q_1$.

On peut reproduire pour $q'$ le même raisonnement que celui qu'on a effectué pour $q$. On construit ainsi une séquence divergente $\hat{\sigma} \in \Delta_T(q)$ telle que pour tout $(\hat{q}, i) \in \chi(\hat{\sigma})$, $\hat{q} \notin Q_1$. La séquence $\hat{\sigma}$ est divergente puisque à chaque pas on construit un préfixe où le temps avance $c$ unités de temps. Par conséquent, $q \notin Q$. ■

A partir de la proposition 3.8 et du théorème 3.3 on a le théorème suivant.

**Théorème 3.5** *Soient $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ un système bien temporisé, $c \in \mathbb{N}$ tel que $c > 0$, et $\varphi$ une formule de* TCTL *avec $[\![\varphi]\!]_{\Delta_T} = Q$. Alors,*

$$[\![\forall\Diamond\varphi]\!]_{\Delta_T} = \mu X.(Q \vee \neg z.\mu Y.(z > c \vee \neg X \triangleright Y))$$

La proposition 3.9 donne une caractérisation de l'inévitabilité dans le cas général. La preuve suit de la même manière que celle de la proposition 3.8.

**Proposition 3.9** *Soient $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ un système temporisé, $Q_1, Q_2 \subseteq \mathcal{Q}$ deux ensembles caractéristiques et $c \in \mathbb{N}$ tel que $c > 0$. Alors,*

$$[\![Q_1 \forall\mathcal{U}\, Q_2]\!]_{\Delta_T} = \mu X.(Q_2 \vee z.(Q_1 \forall\mathcal{U}\, (X \wedge z \leq c)))$$

A partir de la proposition 3.9 et du théorème 3.4 nous avons le théorème suivant.

**Théorème 3.6** *Soient $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ un système bien temporisé, $c \in \mathbb{N}$ tel que $c > 0$, et $\varphi_1$ et $\varphi_2$ deux formules de* TCTL *avec $[\![\varphi_i]\!]_{\Delta_T} = Q_i$, pour $i = 1, 2$. Alors,*

$$[\![\varphi_1 \forall\mathcal{U}\, \varphi_2]\!]_{\Delta_T} = \mu X.(Q_2 \vee \neg z.\mu Y.(\neg(Q_1 \vee X) \vee z > c \vee \neg X \triangleright Y))$$

### 3.3.4 Les systèmes bien temporisés

Etant donné qu'il est nécessaire que dans un système temporisé la progression du temps ne puisse pas être bloquée, il est donc important de détecter si un système temporisé $T$ est bien temporisé.

D'après la définition 3.6, un système $T$ n'est pas bien temporisé s'il existe une séquence dans $\Sigma_T$ dont un préfixe n'est préfixe d'aucune séquence divergente. Il doit exister alors un état à partir duquel le temps dans aucune séquence ne dépasse une unité de temps. La proposition suivante caractérise les systèmes bien temporisés.

**Proposition 3.10** *Soient $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ un système temporisé et $c \in \mathbb{N}$ tel que $c > 0$. $T$ est bien temporisé si et seulement si $\mathcal{Q} \subseteq z.\exists\Diamond z = c$.*

**Preuve.**

$\Rightarrow$ Soit $T$ un système bien temporisé. Soit $q \in \mathcal{Q}$. Sans perte de généralité on peut supposer $q(z) = 0$. Donc, pour toute séquence $\sigma \in \Sigma_T(q)$, tout préfixe de $\sigma$ est préfixe d'une séquence divergente $\bar{\sigma} \in \Delta_T(q)$. Donc, il existe $(\bar{q}, i) \in \chi(\bar{\sigma})$ tel que $\bar{q}(z) = c$. Par conséquent, $q \in z.\exists\Diamond z = c$.

$\Leftarrow$ Supposons que $T$ ne soit pas bien temporisé.

D'après la définition 3.6, il existe un état $q_0$ et une séquence $\sigma \in \Sigma_T(q_0)$ dont un préfixe $\sigma^1 = q_0 \triangleright \ldots \triangleright q_1$ n'est préfixe d'aucune séquence divergente.

Supposons que $q_1 \in z.\exists\Diamond z = c$. Sans perte de généralité on peut supposer $q_1(z) = 0$. Alors, il existe une séquence $\bar{\sigma} \in \Sigma_T(q_1)$ et $(q_2, k) \in \chi(\bar{\sigma})$ tel que $q_2(z) = c$.

Soit $\sigma^2 = q_1 \triangleright \ldots \triangleright q_2$. On a qu'il n'existe aucune séquence divergente ayant $\sigma^2$ comme préfixe, car sinon $\sigma^1\sigma^2$ et $\sigma^1$ seraient préfixes d'une séquence divergente.

On peut reproduire le même raisonnement pour $q_2$. On construirait ainsi une séquence divergente infinie $\sigma^1\sigma^2 \ldots \in \Delta_T$; la divergence étant assurée car pour chaque $\sigma^i$ le temps avance une unité de temps. On aboutit une contradiction.

Donc, $q_1 \notin z.\exists\Diamond z = c$.

Par conséquent, $\mathcal{Q} \not\subseteq z.\exists\Diamond z = c$.

■

## 3.4 Conclusions du chapitre

Les résultats de ce chapitre constituent les fondements de notre approche. En effet, nous avons défini un opérateur "état suivant" pour les systèmes temporisés à partir duquel il est possible de caractériser les propriétés de *possibilité* et d'*inévitabilité*.

Nous avons montré que pour les modèles des systèmes temps-réel, l'opérateur $\exists\mathcal{U}$ s'exprime comme un point fixe, essentiellement de la même façon que dans le cas des modèles discrets.

Le deuxième résultat montre que vérifier une propriété de *vivacité bornée*, sur les systèmes bien temporisés, se réduit à la vérification d'une propriété de *sûreté*. Autrement dit, $p$ est inévitable pendant l'exécution d'un système avant un certain temps $t$ s'il n'est pas possible que $\neg p$ soit vérifiée continûment pendant l'exécution durant plus de $t$ unités de temps.

Le troisième résultat montre que les propriétés de *vivacité* peuvent être évaluées en termes des propriétés de vivacité bornée : $p$ est inévitable s'il est inévitable dans un temps fini.

Les résultats de ce chapitre constituent le noyau de l'algorithme de model-checking symbolique pour TCTL que nous décrivons dans le chapitre suivant.

# Chapitre 4

# Model-checking symbolique

Nous abordons dans ce chapitre le problème du *model-checking* pour TCTL qui consiste à calculer les états du modèle d'un graphe temporisé qui satisfont une formule de TCTL. Nous proposons un algorithme de model-checking symbolique qui calcule l'ensemble caractéristique d'une formule de TCTL, relativement à l'ensemble des séquences divergentes d'un système bien temporisé.

Comme nous l'avons déjà remarqué dans le chapitre 1, le principal problème de l'approche symbolique concerne la procédure de décision. En effet, dans ce cas, décider si un ensemble d'états est inclus dans un autre consiste à décider si un prédicat en implique un autre. Afin de résoudre ce problème, nous proposons une représentation des ensembles caractéristiques des formules qui est à la base d'une procédure de décision mise en œuvre de manière efficace.

L'algorithme de model-checking repose sur :

- les résultats du chapitre 3 qui concernent la caractérisation des opérateurs temporels au moyen des points fixes, et
- une méthode de représentation des ensembles caractéristiques des prédicats d'état qui fournit un algorithme de décision.

La méthode de représentation des prédicats d'état est présentée en deux étapes. La première étape consiste à donner une représentation des prédicats d'état au moyen de contraintes temporelles. Donc, le problème de décider si un prédicat d'état en implique un autre est réduit au problème de décider si une contrainte temporelle en implique une autre. La seconde étape consiste à donner une méthode de décision pour les contraintes temporelles. Celle-ci est basée sur une représentation des contraintes temporelles au moyen d'objets appelés *matrices de bornes* [MB83, Dil89] dont les calculs sont effectués de façon simple. Par conséquent, le calcul d'une opération sur des prédicats d'état s'effectue sur leur représentation. Le résultat obtenu représente aussi un prédicat d'état.

Les résultats de ce chapitre constituent le noyau de l'évaluateur mis en œuvre dans l'outil KRONOS [NSY92, NOSY92, OY92].

## 4.1 Représentation des prédicats d'état

Nous présentons dans cette section une méthode de représentation des ensembles caractéristiques des prédicats d'états. Soient $G = \langle S, H, E, s_I, \delta \rangle$ un graphe temporisé et $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ le modèle de $G$. $G$ et $T$ sont fixés dans cette section.

### 4.1.1 Les états symboliques

Pour $s, s' \in S$ et $\psi, \psi' \in \Psi(\mathcal{H})$, on dit que deux états symboliques $(s, \psi)$ et $(s', \psi')$ sont *équivalents*, si $[\![(s, \psi)]\!] = [\![(s', \psi')]\!]$, i.e., si $s = s'$ et $\psi$ et $\psi'$ sont équivalentes.

Soient $(s, \psi)$ un état symbolique et $(s, v) \in [\![(s, \psi)]\!]$. Puisque $(s, v) \in \mathcal{Q}$, $v$ satisfait aussi la condition d'activité $\delta_s$ associée au sommet $s$. Par conséquent, on a $[\![(s, \psi)]\!] = [\![(s, \psi \wedge \delta_s)]\!]$. On dit qu'un état symbolique $(s, \psi)$ est *normalisé* si $\psi$ implique $\delta_s$, i.e., $\psi$ est équivalent à $\psi \wedge \delta_s$.

On dit que $\bigcup_{s \in S}(s, \psi_s)$ est normalisé si $(s, \psi_s)$ est normalisé pour tout $s \in S$. On définit les opérations suivantes :

$$\begin{aligned}
\bigcup_{s \in S}(s, \psi_s) \wedge \bigcup_{s \in S}(s, \psi'_s) &= \bigcup_{s \in S}(s, \psi_s \wedge \psi'_s) \\
\bigcup_{s \in S}(s, \psi_s) \vee \bigcup_{s \in S}(s, \psi'_s) &= \bigcup_{s \in S}(s, \psi_s \vee \psi'_s) \\
\neg \bigcup_{s \in S}(s, \psi_s) &= \bigcup_{s \in S}(s, \neg \psi_s) \\
(\bigcup_{s \in S}(s, \psi_s))[x := 0] &= \bigcup_{s \in S}(s, \psi_s[x := 0])
\end{aligned}$$

### 4.1.2 La fonction de représentation

Nous montrons à présent comment représenter l'ensemble caractéristique d'un prédicat d'état par un ensemble d'états symboliques. Considérons tout d'abord l'exemple suivant.

**Exemple 4.1** La figure 4.1 illustre le graphe du contrôleur de température présenté dans l'exemple 2.2, où les valeurs des paramètres $t_1$, $t_2$ et $t_{max}$ sont 10, 20 et 35, respectivement, et les sommets sont étiquetés par des propositions. La proposition $r$ indique que le contrôleur a reçu l'ordre de réfrigérer et la proposition $b$ indique que le contrôleur a commencé le mouvement des barres.

Considérons, par exemple, le prédicat d'état $r \wedge 35 < z$. Son ensemble caractéristique est l'ensemble des états $(s, v)$ tels que $r \in \nu(s)$ et $35 < v(z)$. Les sommets étiquetés par la proposition $r$ sont 1 et 5, on a donc :

$$\begin{aligned}
[\![r \wedge 35 < z]\!] = \; & \{(s, v) \mid s = 1 \wedge v(x) = 0 \wedge 35 < v(z)\} \\
& \cup \{(s, v) \mid s = 5 \wedge v(y) = 0 \wedge 35 < v(z)\}
\end{aligned}$$

Par conséquent, l'ensemble caractéristique de $r \wedge 35 < z$ est représenté par l'ensemble d'états symboliques suivant :

$$(1, x = 0 \wedge 35 < z) \cup (5, y = 0 \wedge 35 < z)$$

Notons que cet ensemble d'états symboliques est normalisé. □

On définit une fonction de représentation $\mathcal{D}$ qui associe à tout prédicat d'état $\phi \in \Phi(\mathcal{H})$ un ensemble d'états symboliques. $\mathcal{D}$ est définie par induction sur la structure de $\phi$, de la façon suivante :

$$\begin{aligned}
\mathcal{D}(p) &= \bigcup_{s \in S, p \in \nu(s)} (s, \delta_s) \cup \bigcup_{s \in S, p \notin \nu(s)} (s, \mathit{false}) \\
\mathcal{D}(x \prec c) &= \bigcup_{s \in S} (s, \delta_s \wedge x \prec c)
\end{aligned}$$

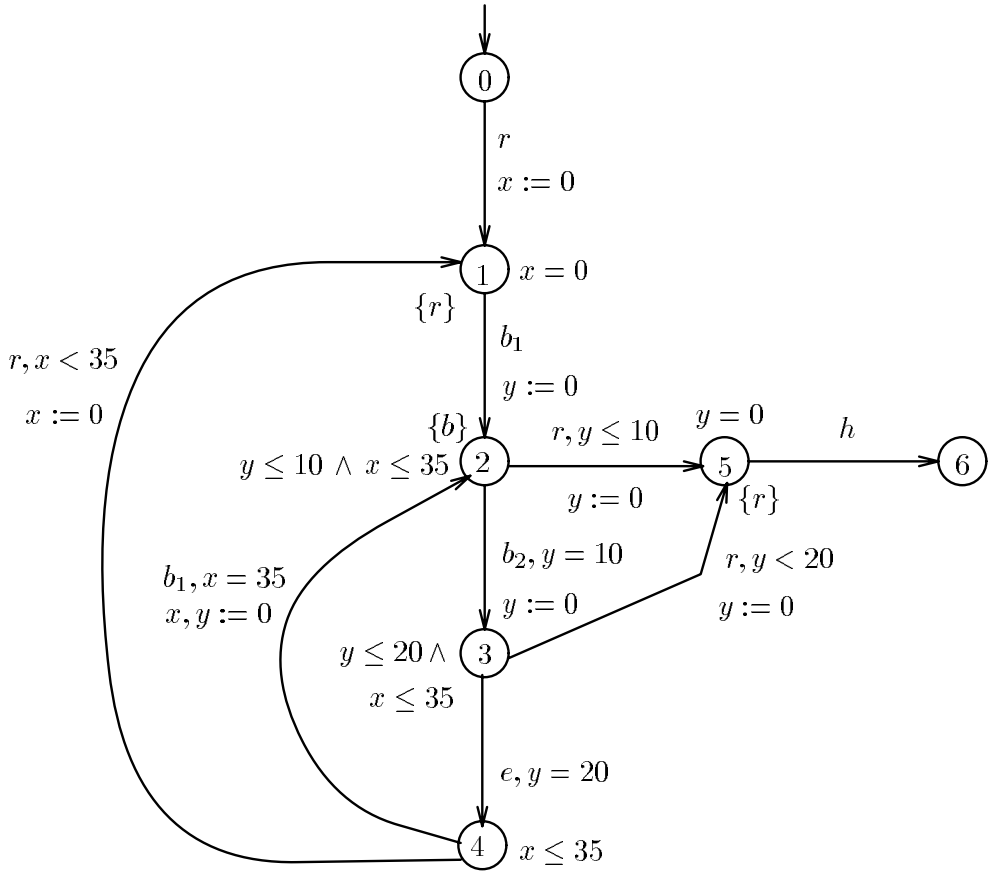



Figure 4.1: Graphe du contrôleur de température.

$$
\begin{aligned}
\mathcal{D}(x - y \prec c) &= \bigcup_{s \in S} (s, \delta_s \wedge x - y \prec c) \\
\mathcal{D}(\neg \phi_1) &= \neg \mathcal{D}(\phi_1) \wedge \bigcup_{s \in S} (s, \delta_s) \\
\mathcal{D}(\phi_1 \wedge \phi_2) &= \mathcal{D}(\phi_1) \wedge \mathcal{D}(\phi_2)
\end{aligned}
$$

Notons que $\mathcal{D}(\phi)$ est normalisé. Nous avons la proposition suivante.

**Proposition 4.1** *Pour tout prédicat d'état $\phi \in \Phi(\mathcal{H})$, $[\![\phi]\!] = [\![\mathcal{D}(\phi)]\!]$.*

**Preuve.** Par induction sur la structure de $\phi$. ■

### 4.1.3 La méthode de décision

Représenter les ensembles caractéristiques des prédicats d'état par des ensembles d'états symboliques fournit une méthode effective pour décider si $[\![\phi_1]\!] \subseteq [\![\phi_2]\!]$, pour $\phi_1, \phi_2 \in \Phi(\mathcal{H})$.

**Proposition 4.2** *Soient* $\phi_1, \phi_2 \in \Phi(\mathcal{H})$ *tels que* $\mathcal{D}(\phi_i) = \bigcup_{s \in S}(s, \psi_s^i)$, *pour* $i = 1, 2$. *On a :*

$$[\![\phi_1]\!] \subseteq [\![\phi_2]\!] \textit{ ssi pour tout } s \in S, [\![\psi_s^1]\!] \subseteq [\![\psi_s^2]\!]$$

La proposition 4.2 fournit un algorithme de décision pour les prédicat d'état qui repose sur un algorithme de décision pour les contraintes temporelles.

## 4.2 Représentation des contraintes temporelles

Nous présentons dans cette section une méthode de représentation des contraintes temporelles. Considérons par exemple la contrainte :

$$1 \leq x \leq 4 \wedge z \leq 5 \wedge z - x \leq 0 \wedge x - z \leq 2$$

Elle peut être représentée au moyen de la matrice suivante :

$$\begin{pmatrix} & \mathbf{0} & x & z \\ \mathbf{0} & (0,\leq) & (-1,\leq) & (0,\leq) \\ x & (4,\leq) & (0,\leq) & (2,\leq) \\ z & (5,\leq) & (0,\leq) & (0,\leq) \end{pmatrix}$$

Chaque entrée de la matrice représente la borne supérieure de la différence entre deux horloges. Par exemple, l'entrée $M_{xz}$ est égale à $(2, \leq)$ ce qui représente la contrainte $x - z \leq 2$. Il en va de même pour l'entrée $M_{zx}$.

Les entrées $M_{\mathbf{0}x}$ et $M_{x\mathbf{0}}$ représentent les bornes inférieures et supérieures de l'horloge $x$. $\mathbf{0}$ est une "horloge" fictive dont la valeur est toujours 0. Ceci permet de représenter la borne inférieure de $x$ comme la différence $\mathbf{0} - x \leq -1$ et la borne supérieure de $x$ comme la différence $x - \mathbf{0} \leq 4$. Il en va de même pour $z$. Comme les valeurs des horloges sont toujours positives, la borne inférieure de $z$ est $(0, \leq)$. Les éléments de la diagonale sont toujours égaux à $(0, \leq)$.

La matrice $M$ est appelée *matrice de bornes* [MB83, Dil89].

### 4.2.1 Les bornes

L'ensemble des bornes, noté $\mathcal{B}$, est défini par :

$$\mathcal{B} = \mathsf{Z} \times \{<, \leq\} \cup \{(-\infty, <), (\infty, <)\}$$

Soit $c \in \mathsf{N}$, on note $\mathcal{B}_c$ le sous-ensemble des bornes $(c', \prec)$ tels que la valeur absolue de $c'$, notée $\text{abs}(c')$, satisfait $\text{abs}(c') \leq c$, ou bien $c' \in \{-\infty, \infty\}$.

Les symboles de relation $<$ et $\leq$ sont totalement ordonnés : $<$ est supposé strictement plus petit que $\leq$. L'ordre total entre les symboles de relation et l'ordre total $\leq$ sur les entiers augmentés de $\{-\infty, \infty\}$ de façon naturelle, induit un ordre total $\sqsubseteq$ sur l'ensemble $\mathcal{B}$ des bornes.

Pour tout $(c_1, \prec_1), (c_2, \prec_2) \in \mathcal{B}$, on définit :

$$(c_1, \prec_1) \sqsubseteq (c_2, \prec_2)$$

si $c_1 < c_2$, ou si $c_1 = c_2$, si $\prec_1$ est inférieure ou égal à $\prec_2$. Notons que la borne $(\infty, <)$ satisfait pour tout $b \in \mathcal{B}$, $b \sqsubseteq (\infty, <)$.

Pour toute borne $(c, \prec) \in \mathcal{B}$, $\neg(c, <)$ est la borne $(-c, \leq)$ et $\neg(c, \leq)$ est la borne $(-c, <)$. De plus, $-(c, \prec)$ est la borne $(-c, \prec)$.

En lui ajoutant quelques opérations, l'ensemble $\mathcal{B}$ des bornes a une structure de *demi-anneau fermé* [AHU74]. Un demi-anneau fermé est un tuple $\langle A, \oplus, \otimes, \mathbf{n}, \mathbf{e}\rangle$ où $A$ est un ensemble d'éléments, $\oplus$ et $\otimes$ sont deux opérations binaires sur $A$, et $\mathbf{n}, \mathbf{e} \in A$, satisfaisant les propriétés suivantes :

1. $\langle A, \oplus, \mathbf{n}\rangle$ est un *monoïde*, i.e., $A$ est *fermé* pour $\oplus$, $\oplus$ est *associative*, et $\mathbf{n}$ est l'élément *neutre* pour $\oplus$ ($a \oplus \mathbf{n} = a$). Egalement, $\langle A, \otimes, \mathbf{e}\rangle$ est un monoïde et $\mathbf{e}$ est l'élément neutre pour $\otimes$. De plus, $\mathbf{n}$ est l'élément *absorbant* pour $\otimes$ ($a \otimes \mathbf{n} = \mathbf{n}$).
2. $\oplus$ est *commutative* et *idempotente* ($a \oplus a = a$).
3. $\otimes$ est *distributive* par rapport à $\oplus$.
4. $\bigoplus_{i\in\mathsf{N}} a_i$ existe et est unique.
5. $(\bigoplus_{i\in\mathsf{N}} a_i) \otimes \left(\bigoplus_{j\in\mathsf{N}} b_j\right) = \bigoplus_{i,j\in\mathsf{N}}(a_i \otimes b_j) = \bigoplus_{i\in\mathsf{N}} \left(\bigoplus_{j\in\mathsf{N}}(a_i \otimes b_j)\right)$.

On définit sur $\mathcal{B}$ les opérations suivantes.

- Pour tout $(c_1, \prec_1), (c_2, \prec_2) \in \mathcal{B}$, on définit :

  $$(c_1, \prec_1) \otimes (c_2, \prec_2) = (c_1 + c_2, \min(\prec_1, \prec_2))$$

  où $+$ est la somme des entiers pour $c_1, c_2 \in \mathsf{Z}$ et pour $c \in \mathsf{Z}$ on a :

  $$\begin{aligned} c + \infty &= \infty \\ c + (-\infty) &= -\infty \\ \infty + \infty &= \infty \\ \infty + (-\infty) &= \infty \\ -\infty + (-\infty) &= -\infty \end{aligned}$$

  Notons que :

  - la borne $(0, \leq)$ est l'élément *neutre* pour l'opération $\otimes$, i.e., $(0, \leq)$ satisfait pour tout $b \in \mathcal{B}$, $(0, \leq) \otimes b = b$,
  - la borne $(\infty, <)$ est l'élément *absorbant* pour l'opération $\otimes$, i.e., $(\infty, <)$ satisfait pour tout $b \in \mathcal{B}$, $(\infty, <) \otimes b = (\infty, <)$.

- Pour tout $b_1, b_2 \in \mathcal{B}$, on définit :

  $$b_1 \oplus b_2 = \min(b_1, b_2)$$

  Notons que la borne $(\infty, <)$ est l'élément *neutre* pour l'opération $\oplus$, i.e., satisfait pour tout $b \in \mathcal{B}$, $(\infty, <) \oplus b = b$.

- De plus, on définit :

  $$\begin{aligned} \mathbf{n} &= (\infty, <) \\ \mathbf{e} &= (0, \leq) \end{aligned}$$

On peut vérifier que la structure $\langle \mathcal{B}, \oplus, \otimes, \mathbf{n}, \mathbf{e}\rangle$ est un demi-anneau fermé.

### 4.2.2 Les matrices de bornes

Tout d'abord on définit $\mathcal{H}^\sharp = \mathcal{H} \cup \{\mathbf{0}\}$. Soit $M$ une *matrice carrée de bornes* sur $\mathcal{H}^\sharp$. L'*ensemble caractéristique* de $M$, noté $[\![M]\!]$, est l'ensemble des valuations $v \in V$ telles que pour tout $x, y \in \mathcal{H}^\sharp$, $v(x) - v(y) \prec c$ où $M_{xy} = (c, \prec)$. Afin de simplifier la notation, par la suite on écrit $v(x) - v(y) \prec M_{xy}$.

L'ordre total $\sqsubseteq$ défini sur les bornes peut être étendu de manière naturelle aux matrices de bornes. Pour tout paire de matrices $M$ et $M'$,

$$M \sqsubseteq M'$$

si pour tout $x, y \in \mathcal{H}^\sharp$,

$$M_{xy} \sqsubseteq M'_{xy}$$

Par conséquent, si $M \sqsubseteq M'$ alors $[\![M]\!] \subseteq [\![M']\!]$.

Les opérations $\otimes$ et $\oplus$ peuvent également être étendues aux matrices de bornes de la façon suivante :

- Soient $M$ et $M'$ deux matrices de bornes. $M \otimes M'$ est la matrice $M''$ telle que pour tout $x, y \in \mathcal{H}^\sharp$,
$$M''_{xy} = \bigoplus_{z \in \mathcal{H}^\sharp} M_{xz} \otimes M'_{zy}$$
- Soient $M$ et $M'$ deux matrices de bornes. $M \oplus M'$ est la matrice $M''$ telle que pour tout $x, y \in \mathcal{H}^\sharp$,
$$M''_{xy} = M_{xy} \oplus M'_{xy}$$
- De plus, on définit les matrices $\mathbf{N}$ et $\mathbf{E}$ de la façon suivante. Pour tout $x, y \in \mathcal{H}^\sharp$,
$$\begin{array}{rcl} \mathbf{E}_{xy} & = & \begin{cases} \mathbf{e} & \text{si } x = y, \\ \mathbf{n} & \text{sinon.} \end{cases} \\ \mathbf{N}_{xy} & = & \mathbf{n} \end{array}$$
Notons que :
  - $\mathbf{E}$ est l'élément *neutre* pour $\otimes$, i.e., $\mathbf{E} \otimes M = M$ pour toute matrice $M$,
  - $\mathbf{N}$ est l'élément *absorbant* pour $\otimes$, i.e., $\mathbf{N} \otimes M = \mathbf{N}$, et l'élément *neutre* pour $\oplus$, i.e., $\mathbf{N} \oplus M = M$, pour toute matrice $M$.

Soit $\mathcal{M}$ l'ensemble des matrices de bornes définit par :

$$\mathcal{M} = \{M \mid \forall x \in \mathcal{H}^\sharp.\ M_{xx} \sqsubseteq (0, \leq)\} \cup \{\mathbf{N}\}$$

On peut vérifier que la structure $\langle \mathcal{M}, \oplus, \otimes, \mathbf{N}, \mathbf{E} \rangle$ est un demi-anneau fermé. Soit $M \in \mathcal{M}$ une matrice de bornes. On définit :

$$M^* = \bigoplus_{k \in \mathbb{N}} M^k$$

où $M^0 = \mathbf{E}$ et $M^{k+1} = M^k \otimes M$. Notons que pour tout $x \in \mathcal{H}^\sharp$, $M^*_{xx} \sqsubseteq \mathbf{e}$ car $M^0 = \mathbf{E}$. La structure $\langle \mathcal{M}, \oplus, \otimes, *, \mathbf{N}, \mathbf{E} \rangle$ est une *algèbre régulière* [Dil89].

**Remarque 4.1** Soit $M \in \mathcal{M}$, $M \neq \mathbf{N}$, tel que $[\![M]\!] \neq \emptyset$. On a $(0, \leq) \sqsubseteq M_{xx}$, pour tout $x \in \mathcal{H}^\sharp$. En effet, si nous supposons le contraire, puisque $[\![M]\!] \neq \emptyset$, il existe une valuation $v \in [\![M]\!]$ et donc $v$ est telle que $v(x) - v(x) < 0$, ce qui est une contradiction. Par conséquent, $(0, \leq) = M_{xx}$. □

**Exemple 4.2** Soit $\psi_1$ et $\psi_2$ les contraintes :

$$\begin{aligned} \psi_1 &= (2 < x \wedge z \leq 1 \wedge x - y \leq 3) \\ \psi_2 &= (x \leq 3 \wedge y \leq 5 \wedge y - z < 1) \end{aligned}$$

Soit $M_1$ et $M_2$ les matrices correspondantes à $\psi_1$ et $\psi_2$, respectivement. $M_1 \oplus M_2$ est :

$$\begin{pmatrix} & \mathbf{0} & x & y & z \\ \mathbf{0} & (0,\leq) & (-2,<) & (0,\leq) & (0,\leq) \\ x & (\infty,<) & (0,\leq) & (3,\leq) & (\infty,<) \\ y & (\infty,<) & (\infty,<) & (0,\leq) & (\infty,<) \\ z & (1,\leq) & (\infty,<) & (\infty,<) & (0,\leq) \end{pmatrix} \oplus \begin{pmatrix} & \mathbf{0} & x & y & z \\ \mathbf{0} & (0,\leq) & (0,\leq) & (0,\leq) & (0,\leq) \\ x & (3,\leq) & (0,\leq) & (\infty,<) & (\infty,<) \\ y & (5,\leq) & (\infty,<) & (0,\leq) & (1,<) \\ z & (\infty,<) & (\infty,<) & (\infty,<) & (0,\leq) \end{pmatrix}$$

c'est-à-dire :

$$\begin{pmatrix} & \mathbf{0} & x & y & z \\ \mathbf{0} & (0,\leq) & (-2,<) & (0,\leq) & (0,\leq) \\ x & (3,\leq) & (0,\leq) & (3,\leq) & (\infty,<) \\ y & (5,\leq) & (\infty,<) & (0,\leq) & (1,<) \\ z & (1,\leq) & (\infty,<) & (\infty,<) & (0,\leq) \end{pmatrix}$$

qui correspond à la contrainte $\psi_1 \wedge \psi_2$. □

En effet, comme l'illustre l'exemple, l'opération $\oplus$ sur les matrices de bornes coïncide avec l'intersection des ensembles caractéristiques. Autrement dit, si $\psi$ et $\psi'$ sont deux contraintes représentées par les matrices $M$ et $M'$, alors la contrainte $\psi \wedge \psi'$ est représentée par la matrice $M \oplus M'$. Nous avons donc la proposition suivante.

**Lemme 4.1** *Pour tout* $M, M' \in \mathcal{M}$, $[\![M \oplus M']\!] = [\![M]\!] \cap [\![M']\!]$.

**Preuve.** Soit $M''$ la matrice $M \oplus M'$.

- $\subseteq$ D'après la définition de $\oplus$, on a $M'' \sqsubseteq M$ et $M'' \sqsubseteq M'$. Donc, $[\![M'']\!] \subseteq [\![M]\!]$ et $[\![M'']\!] \subseteq [\![M']\!]$. Par conséquent, $[\![M'']\!] \subseteq [\![M]\!] \cap [\![M']\!]$.
- $\subseteq$ Soit $v \in [\![M]\!] \cap [\![M']\!]$. On a, pour tout $x, y \in \mathcal{H}^\sharp$, $v(x) - v(y) \prec M_{xy}$ et $v(x) - v(y) \prec M'_{xy}$. Donc, $v(x) - v(y) \prec M_{xy} \oplus M'_{xy}$. Par conséquent, $v \in [\![M'']\!]$.

Par conséquent, $[\![M \oplus M']\!] = [\![M]\!] \cap [\![M']\!]$. ■

**Exemple 4.3** Soit $\psi$ la contrainte temporelle :

$$1 \leq x \leq 4 \wedge 1 \leq z \leq 5 \wedge z - x \leq 0 \wedge x - z \leq 2 \tag{4.1}$$

La matrice $M$ de $\psi$ est :

$$\begin{pmatrix} & \mathbf{0} & x & z \\ \mathbf{0} & (0,\leq) & (-1,\leq) & (-1,\leq) \\ x & (4,\leq) & (0,\leq) & (2,\leq) \\ z & (5,\leq) & (0,\leq) & (0,\leq) \end{pmatrix}$$

$M \otimes M$ est la matrice $M'$ :

$$\begin{pmatrix} & \mathbf{0} & x & z \\ \mathbf{0} & (0,\leq) & (-1,\leq) & (-1,\leq) \\ x & (4,\leq) & (0,\leq) & (2,\leq) \\ z & (4,\leq) & (0,\leq) & (0,\leq) \end{pmatrix}$$

dont la contrainte associée est :

$$1 \leq x \leq 4 \wedge 1 \leq z \leq 4 \wedge z - x \leq 0 \wedge x - z \leq 2 \tag{4.2}$$

Les contraintes 4.1 et 4.2 sont équivalentes. En effet, 4.2 peut être obtenue de 4.1 de la façon suivante. De :

$$z - x \leq 0 \tag{4.3}$$
$$x \leq 4 \tag{4.4}$$

on déduit

$$z \leq 4 \tag{4.5}$$

La contrainte 4.3 correspond à :

$$M_{zx} = (0,\leq)$$

et la contrainte 4.4 correspond à :

$$M_{x\mathbf{0}} = (4,\leq)$$

La contrainte 4.5 correspond au calcul :

$$M_{zx} \otimes M_{x\mathbf{0}} = (4,\leq) \tag{4.6}$$

De plus, la contrainte $z \leq 5$ correspond à :

$$M_{z\mathbf{0}} \otimes M_{\mathbf{00}} = (5,\leq) \tag{4.7}$$

De 4.6 et 4.7, par application de l'opération $\oplus$, on obtient alors :

$$M'_{z\mathbf{0}} = (4,\leq)$$

Donc, appliquer l'opération $\otimes$ à la matrice $M$ revient à déduire la contrainte implicite $z \leq 4$ de la contrainte 4.1. □

Cet exemple montre que plusieurs contraintes différentes peuvent représenter le même ensemble de valuations à cause des *contraintes implicites*. Ces contraintes peuvent être détectées à l'aide de l'opération $\otimes$. Mais, comme le montre l'exemple suivant, appliquer une seule fois l'opération $\otimes$ ne suffit pas pour détecter *toutes* les contraintes implicites.

**Exemple 4.4** Soit $M$ la matrice de bornes de la contrainte temporelle :

$$\psi \;=\; 2 \leq x \wedge y < 5 \wedge x - y \leq 1 \wedge y - z \leq -4$$

Donc, $M$ est la matrice :

$$\begin{pmatrix} & \mathbf{0} & x & y & z \\ \mathbf{0} & (0,\leq) & (-2,\leq) & (0,\leq) & (0,\leq) \\ x & (\infty,<) & (0,\leq) & (1,\leq) & (\infty,<) \\ y & (5,<) & (\infty,<) & (0,\leq) & (-4,\leq) \\ z & (\infty,<) & (\infty,<) & (\infty,<) & (0,\leq) \end{pmatrix}$$

Donc, $M^2 = M \otimes M$ est :

$$\begin{pmatrix} & \mathbf{0} & x & y & z \\ \mathbf{0} & (0,\leq) & (-2,\leq) & (-1,\leq) & (0,\leq) \\ x & (6,<) & (0,\leq) & (1,\leq) & (-3,\leq) \\ y & (5,<) & (3,<) & (0,\leq) & (-4,\leq) \\ z & (\infty,<) & (\infty,<) & (\infty,<) & (0,\leq) \end{pmatrix}$$

Notons que $M^2_{\mathbf{0}z} = (0,\leq)$ ce qui correspond à $0 \leq z$. Pourtant, à partir de $\psi$ on peut déduire $5 \leq z$. En effet, à partir de :

$$\begin{aligned} y - z &\leq -4 \\ x - y &\leq 1 \end{aligned}$$

on déduit :

$$x - z \;\leq\; -3$$

et, puisque

$$-x \;\leq\; -2$$

on déduit

$$-z \;\leq\; -5$$

Soit $M^3 = M^2 \otimes M$, $M^3$ est la matrice :

$$\begin{pmatrix} & \mathbf{0} & x & y & z \\ \mathbf{0} & (0,\leq) & (-2,\leq) & (-1,\leq) & (-5,\leq) \\ x & (6,<) & (0,\leq) & (1,\leq) & (-3,\leq) \\ y & (5,<) & (3,<) & (0,\leq) & (-4,\leq) \\ z & (\infty,<) & (\infty,<) & (\infty,<) & (0,\leq) \end{pmatrix}$$

$M^3$ est telle que $M^3_{\mathbf{0}z} = (-5,\leq)$. Donc, deux applications de $\otimes$ sont nécessaires pour détecter la contrainte implicite $5 \leq z$.

De plus, $M^3$ satisfait $M^3 = M^3 \otimes M$. $M^3$ peut donc être considérée comme une représentation *canonique* de $M^3$ ne contenant pas de contraintes implicites. □

En effet, toute matrice peut être mise sous forme canonique comme illustré dans l'exemple. Tout d'abord, il faut choisir une matrice pour représenter de manière unique l'ensemble vide, Pour ceci, nous choisissons la matrice $\mathbf{O}$ telle que pour tout $x, y \in \mathcal{H}^\sharp$,

$$\mathbf{O}_{xy} = (-\infty, <)$$

Pour tout $M \in \mathcal{M}$, la forme canonique de $M$ est la matrice définie par :

$$\mathbf{cf}(M) = \begin{cases} \mathbf{O} & \text{si } [\![M]\!] = \emptyset, \\ M^* & \text{sinon.} \end{cases}$$

**Remarque 4.2** Soit $M' = \mathbf{cf}(M)$. Pour tout $x, y, z \in \mathcal{H}^\sharp$, $M'$ satisfait $M'_{xy} \sqsubseteq M'_{xz} \otimes M'_{zy}$. □

L'opération $\otimes$ satisfait la propriété suivante.

**Lemme 4.2** *Pour toute matrice $M \in \mathcal{M}$, si $[\![M]\!] \neq \emptyset$ alors $[\![M \otimes M]\!] = [\![M]\!]$.*

**Preuve.** Si $M = \mathbf{N}$, alors $M \otimes M = M$ et par conséquent $[\![M \otimes M]\!] = [\![M]\!]$.

Supposons $M \neq \mathbf{N}$. Soit $M'$ la matrice $M \otimes M$. Or, pour tout $x, y \in \mathcal{H}^\sharp$, on a $M'_{xy} = \bigoplus_{z \in \mathcal{H}^\sharp} M_{xz} \otimes M_{zy}$.

- $\subseteq$ Or, d'après la remarque 4.1, $M_{xx} = (0, \leq)$ pour tout $x \in \mathcal{H}^\sharp$. Donc, $M_{xx} \otimes M_{xy} = M_{xy}$. Par conséquent, $M'_{xy} \sqsubseteq M_{xy}$ et donc $M' \sqsubseteq M$. On a donc $[\![M']\!] \subseteq [\![M]\!]$.
- $\supseteq$ Soit $v \in [\![M]\!]$. Or, pour tout $x, y \in \mathcal{H}^\sharp$, $v(x) - v(y) \prec M_{xy}$. Or, pour tout $z \in \mathcal{H}^\sharp$, $v(x) - v(z) \prec M_{xz}$ et $v(z) - v(y) \prec M_{zy}$. Donc, pour tout $z \in \mathcal{H}^\sharp$, $v(x) - v(y) \prec M_{xz} \otimes M_{zy}$. Ce qui donne, $v(x) - v(y) \prec \bigoplus_{z \in \mathcal{H}^\sharp} M_{xz} \otimes M_{zy}$. Par conséquent, $v(x) - v(y) \prec M'_{xy}$ et $v \in [\![M']\!]$. On a donc $[\![M]\!] \subseteq [\![M']\!]$.

Par conséquent, $[\![M \otimes M]\!] = [\![M]\!]$. ■

**Corollaire 4.1** *Pour toute matrice $M$, si $[\![M]\!] \neq \emptyset$ alors pour tout $k \geq 1$, $[\![M^k]\!] = [\![M]\!]$.*

Le lemme suivant dit que les matrices $M$ et $\mathbf{cf}(M)$ sont équivalentes.

**Lemme 4.3** *Pour toute matrice $M$, $\mathbf{cf}(M)$ satisfait $[\![M]\!] = [\![\mathbf{cf}(M)]\!]$.*

**Preuve.** Si $[\![M]\!] = \emptyset$, alors $\mathbf{cf}(M) = \mathbf{O}$. $\mathbf{O}$ satisfait $[\![\mathbf{O}]\!] = \emptyset$. Donc, $[\![M]\!] = [\![\mathbf{O}]\!]$.

Si $[\![M]\!] \neq \emptyset$, alors $\mathbf{cf}(M) = M^*$. Or, par lemme 4.1 $[\![\bigoplus_{k \in \mathbb{N}} M^k]\!] = \bigcap_{k \in \mathbb{N}} [\![M^k]\!]$. D'après le corollaire 4.1, $[\![M^k]\!] = [\![M]\!]$ pour tout $k \geq 1$. Donc, $[\![\mathbf{cf}(M)]\!] = [\![\mathbf{E}]\!] \cap [\![M]\!]$. Par conséquent, $[\![\mathbf{cf}(M)]\!] = [\![M]\!]$. ■

Le lemme suivant montre que si $[\![M]\!] \neq \emptyset$, $M^*$ est la plus petite matrice qui a le même ensemble caractéristique que $M$.

**Lemme 4.4** *Pour tout $M, M'$, si $[\![M]\!] = [\![M']\!] \neq \emptyset$ et $M = M^*$ alors $M \sqsubseteq M'$.*

**Preuve.** (cf. [Dil89]) La preuve suit de la manière suivante : on suppose le contraire et on trouve une valuation $v$ telle que $v \in [\![M]\!]$ mais $v \notin [\![M']\!]$, ce qui contredit l'hypothèse que $M$ et $M'$ sont équivalents.

(1) Supposons que $M \not\sqsubseteq M'$. Alors, ils existent $x, y \in \mathcal{H}^\sharp$ tels que $M'_{xy} \sqsubset M_{xy}$. Or, $M_{xx} = M'_{xx} = (0, \leq)$, et donc $x$ et $y$ doivent être différentes.

Supposons que $M_{xy} = (c_{xy}, \prec_{xy})$, $M_{yx} = (c_{yx}, \prec_{yx})$, $M'_{xy} = (c'_{xy}, \prec'_{xy})$.

Posons $t = [\max(c'_{xy}, -c_{yx}) + c_{xy}]/2$.

Or, $(0, \leq) \sqsubseteq (c_{xy}, \prec_{xy}) + (c_{yx}, \prec_{yx})$. Donc, on a deux cas :

- Soit $-c_{yx} < c_{xy}$. Donc on a $\max(c'_{xy}, -c_{yx}) < t < c_{xy}$.
- Soit $-c_{yx} = c_{xy}$ et $\leq = \prec_{yx} = \prec_{xy}$. Or, $c'_{xy} < c_{xy}$. Donc, $\max(c'_{xy}, -c_{yx}) = -c_{yx} = c_{xy}$. Par conséquent, on a $t = c_{xy}$.

Dans les deux cas on a $t \prec M_{xy}$ et $-t \prec M_{yx}$, mais par contre, $t \not\prec M'_{xy}$.

(2) La valeur $t$ peut être utilisée à présent pour construire une valuation $v$ telle que $v \in [\![M]\!]$ mais $v \notin [\![M']\!]$. La valuation $v$ est construite de la façon suivante.

Pour $x$, $v(x) = t$ et pour $\mathbf{0}$ et $y$, $v(\mathbf{0}) = v(y) = 0$. Supposons que $v$ est définie pour $\mathcal{H}_1^\sharp \subseteq \mathcal{H}^\sharp$ et satisfait pour tout $z, w \in \mathcal{H}_1^\sharp$, $v(z) - v(w) \prec M_{zw}$. Soit $r \notin \mathcal{H}_1^\sharp$. Il faut trouver une valuation appropriée pour $v(r)$, qui doit satisfaire :

- $v(r) - v(z) \prec M_{rz}$, et
- $v(w) - v(r) \prec M_{wr}$.

Donc, une telle valeur existe si et seulement si $(v(w) - v(z), \leq) \sqsubseteq M_{wr} \otimes M_{rz}$. Or, cette inéquation est valide puisque $v(w) - v(z) \prec M_{wz}$ et, d'après la remarque 4.2, $M_{wz} \sqsubseteq M_{wr} \otimes M_{rz}$.

Par conséquent, la valuation $v$ ainsi construite satisfait $v \in [\![M]\!]$ mais $v \notin [\![M']\!]$. On aboutit donc à une contradiction. Par conséquent, $M \sqsubseteq M'$. ■

La proposition 4.3 dit que toutes les matrices équivalentes ont la même forme canonique. La proposition 4.3 suit des lemmes 4.3 et 4.4.

**Proposition 4.3** *Pour tout $M, M'$,*

$$[\![M]\!] = [\![M']\!] \quad \textit{ssi} \quad \mathbf{cf}(M) = \mathbf{cf}(M')$$

**Preuve.** Si $\mathbf{cf}(M) = \mathbf{cf}(M')$ alors par lemme 4.3 on a $[\![M]\!] = [\![M']\!]$.

Si $[\![M]\!] = [\![M']\!] = \emptyset$ alors $\mathbf{cf}(M) = \mathbf{cf}(M') = \mathbf{O}$ par définition. Si $[\![M]\!] = [\![M']\!] \neq \emptyset$ alors par lemme 4.3 on a $[\![\mathbf{cf}(M)]\!] = [\![\mathbf{cf}(M')]\!]$. Or, $\mathbf{cf}(M)^* = \mathbf{cf}(M)$. Donc, par lemme 4.4, $\mathbf{cf}(M) \sqsubseteq \mathbf{cf}(M')$. Par symétrie, $\mathbf{cf}(M') \sqsubseteq \mathbf{cf}(M)$. Par conséquent, $\mathbf{cf}(M) = \mathbf{cf}(M')$. ■

Pour toute matrice $M$, la matrice $\mathbf{cf}(M)$ peut être calculée par l'algorithme de Floyd-Warshall [AHU74]. Ceci permet aussi de déterminer si l'ensemble caractéristique de $M$ est vide. En effet, si la matrice obtenue par application de l'algorithme de Floyd-Warshall est telle qu'un

élément de la diagonale est inférieur à $(0, \leq)$, d'après la remarque 4.1, l'ensemble caractéristique de $M$ est vide.

Nous avons donc un moyen effectif de déterminer si deux matrices sont équivalentes. La complexité de l'algorithme de Floyd-Warshall est $O(|\mathcal{H}^\sharp|^3)$.

### 4.2.3 La fonction de représentation des contraintes temporelles

Nous donnons ici une représentation des contraintes temporelles au moyen d'ensembles de matrices, ce qui permet ensuite de décider si deux contraintes $\psi_1$ et $\psi_2$ sont équivalentes. Tout d'abord, nous introduisons quelques définitions.

#### Ensemble caractéristique et forme canonique

Soit $K \subseteq \mathcal{M}$ un ensemble de matrices. Les définitions d'ensemble caractéristique et de l'opération **cf** sont étendues de façon naturelle :

$$\begin{aligned} [\![K]\!] &= \bigcup_{M \in K} [\![M]\!] \\ \mathbf{cf}(K) &= \{\mathbf{cf}(M) \mid M \in K \wedge \mathbf{cf}(M) \neq \mathbf{O}\} \end{aligned}$$

Pour $K \subseteq \mathcal{M}$, $\hat{K}$ est l'ensemble de matrices qui satisfait $[\![K]\!] = [\![\hat{K}]\!]$ et pour tout $K' \subseteq \mathcal{M}$ tel que $[\![K]\!] = [\![K']\!]$ on a $K' \subseteq \hat{K}$. C'est-à-dire, $\hat{K}$ est le plus grand ensemble de matrices équivalent à $K$.

#### La matrice universelle

On définit la matrice $\mathbf{U} \subseteq \mathcal{M}$ telle que pour tout $x, y \in \mathcal{H}^\sharp$,

$$\mathbf{U}_{xy} = \begin{cases} (0, \leq) & \text{si } x = y \text{ ou } x = \mathbf{0}, \\ (\infty, <) & \text{sinon.} \end{cases}$$

Donc, $[\![\mathbf{U}]\!]$ est l'ensemble de toutes les valuations. Autrement dit, la matrice $\mathbf{U}$ représente la contrainte *true*.

#### L'opérateur d'initialisation d'horloge

On définit sur les matrices de bornes l'opération d'initialisation d'horloge. Pour ce faire, on définit tout d'abord l'opération suivante. Soient $M \in \mathcal{M}$, $x \in \mathcal{H}$ et $M' = \mathbf{cf}(M)$. $M \downarrow x$ est la matrice $\mathbf{O}$ si $M' = \mathbf{O}$, sinon $\mathbf{cf}(M'')$ où $M''$ est la matrice :

$$M''_{yz} = \begin{cases} (0, \leq) & \text{si } y = x \text{ et } z = \mathbf{0}, \\ M'_{yz} & \text{sinon.} \end{cases}$$

La matrice $M \downarrow x$ satisfait la propriété suivante.

**Lemme 4.5** *Pour toute valuation $v \in V$, $v \in [\![M \downarrow x]\!]$ ssi $v(x) = 0$ et $v \in [\![M]\!]$.*

**Preuve.** Si $M' = \mathbf{O}$, alors $[\![M]\!] = [\![M \downarrow x]\!] = \emptyset$ et donc la propriété est valide.

Supposons que $M' \neq \mathbf{O}$, ce qui est équivalent à $[\![M]\!] \neq \emptyset$.

$\Rightarrow$ Soit $v \in [\![M \downarrow x]\!]$. Par définition $v \in [\![\mathbf{cf}(M'')]\!]$. Donc, d'après la proposition 4.3, $v \in [\![M'']\!]$ et $v(y) - v(z) \prec M''_{yz}$.

Il faut distinguer deux cas :

1. Soit $yz \neq x\mathbf{0}$. Donc, par construction, $v(y) - v(z) \prec M'_{yz}$. Or, $M' = \mathbf{cf}(M)$. Donc, d'après la proposition 4.3, $v(y) - v(z) \prec M_{yz}$.
2. Soit $yz = x\mathbf{0}$. D'après le lemme 4.4, $\mathbf{cf}(M'') \sqsubseteq M''$ et donc $v(x) - v(\mathbf{0}) \prec M''_{x\mathbf{0}}$. Or, par construction, $M''_{x\mathbf{0}} = (0, \leq)$. Donc, $v(x) - v(\mathbf{0}) \leq 0$, et par conséquent, $v(x) = 0$. De plus, $v(\mathbf{0}) - v(x) = 0 \prec M'_{\mathbf{0}x}$. Or, $[\![M']\!] \neq \emptyset$, donc $(0, \leq) \sqsubseteq M'_{\mathbf{0}x}$. Or, $M' = \mathbf{cf}(M)$, donc, d'après le lemme 4.4, $M'_{x\mathbf{0}} \sqsubseteq M_{x\mathbf{0}}$. Par conséquent, $(0, \leq) \sqsubseteq M_{x\mathbf{0}}$ et donc $v(x) - v(\mathbf{0}) = 0 \prec M_{x\mathbf{0}}$.

On a donc $v(x) = 0$ et $v \in [\![M]\!]$.

$\Leftarrow$ Soit $v \in [\![M]\!]$ telle que $v(x) = 0$. Or, $M' = \mathbf{cf}(M)$ et donc, d'après la proposition 4.3, $v \in [\![M']\!]$. Par construction, $v \in [\![M'']\!]$. Donc, d'après la proposition 4.3, $v \in [\![\mathbf{cf}(M'')]\!]$. Par conséquent, $v \in [\![M \downarrow x]\!]$.

■

On définit à présent l'opération d'initialisation d'horloge. Soient $M \in \mathcal{M}$, $x \in \mathcal{H}$ et $M' = M \downarrow x$. $M[x := 0]$ est la matrice $\mathbf{O}$ si $M' = \mathbf{O}$, sinon $M''$ où :

$$M''_{yz} = \begin{cases} (\infty, <) & \text{si } y = x \text{ ou } z = \mathbf{0}, \\ M'_{yz} & \text{sinon.} \end{cases}$$

La matrice $M[x := 0]$ satisfait la propriété suivante.

**Lemme 4.6** *Pour toute valuation $v \in V$, $v \in [\![M[x := 0]]\!]$ ssi $v[x := 0] \in [\![M]\!]$.*

**Preuve.** ($\Rightarrow$) Soient $v \in [\![M[x := 0]]\!]$ et $v' = v[x := 0]$. Donc, par construction, $v' \in [\![M \downarrow x]\!]$. D'après le lemme 4.5 on a $v' \in [\![M]\!]$.

($\Leftarrow$) Soient $v, v' \in V$ tels que $v' = v[x := 0] \in [\![M]\!]$. D'après le lemme 4.5 on a $v' \in [\![M \downarrow x]\!]$. Donc, par construction, $v' \in [\![M[x := 0]]\!]$. ■

L'opération d'initialisation d'horloge est étendue aux ensembles de matrices de la façon suivante. Soient $K \subseteq \mathcal{M}$ et $x \in \mathcal{H}$, $K[x := 0]$ est :

$$K[x := 0] \quad = \quad \bigcup_{M \in K} M[x := 0]$$

Le corollaire 4.2 est une conséquence du lemme 4.6.

**Corollaire 4.2** *Pour toute valuation $v \in V$, $v \in [\![K[x := 0]]\!]$ ssi $v[x := 0] \in [\![K]\!]$.*

### Le complément d'une matrice

Soient $M, M' \in \mathcal{M}$, tels que $M' = \mathbf{cf}(M)$. $\neg M$ est l'ensemble $\{\mathbf{U}\}$ si $M' = \mathbf{O}$, sinon $\bigcup_{x \neq y \in \mathcal{H}^\sharp} M^{xy}$ où pour tout $z, w \in \mathcal{H}^\sharp$,

$$M^{xy}_{zw} = \begin{cases} \neg M'_{yx} & \text{si } z = x \text{ et } w = y, \\ \mathbf{U}_{zw} & \text{sinon.} \end{cases}$$

Donc, $\neg M$ représente le complément de $M$. En effet, nous avons la propriété suivante.

**Lemme 4.7** *Pour toute valuation $v \in V$, $v \in [\![\neg M]\!]$ ssi $v \notin [\![M]\!]$.*

**Preuve.** Supposons $\mathbf{cf}(M) = \mathbf{O}$. Donc, $[\![M]\!] = \emptyset$. Par construction, $[\![\neg M]\!] = V$. Par conséquent, la propriété est valide.

Soit $M' = \mathbf{cf}(M) \neq \mathbf{O}$.

- $\Rightarrow$ Soit $v \in [\![\neg M]\!]$. Par construction, ils existent $x \neq y \in \mathcal{H}^\sharp$ tels que $v \in [\![M^{xy}]\!]$. Donc, $v(x) - v(y) \prec \neg M'_{yx}$, et par conséquent $v(y) - v(x) \not\prec M'_{yx}$. Donc, $v \notin [\![M']\!]$ et d'après la proposition 4.3, on a $v \notin [\![M]\!]$.
- $\Leftarrow$ Soit $v \notin [\![M]\!]$. D'après la proposition 4.3, $v \notin [\![M']\!]$. Donc, ils existent $x \neq y \in \mathcal{H}^\sharp$, tels que $v(y) - v(x) \not\prec M'_{yx}$. Par conséquent, $v(x) - v(y) \prec \neg M'_{yx}$ et $v \in [\![M^{yx}]\!]$.

■

### Conjonction, disjonction et négation

Soit $K_1, K_2 \subseteq \mathcal{M}$. On définit :

$$\begin{aligned} K_1 \vee K_2 &= K_1 \cup K_2 \\ K_1 \wedge K_2 &= \{M_1 \oplus M_2 \mid M_i \in K_i, i = 1, 2\} \\ \neg K &= \bigwedge_{M_1 \in K_1} \neg M_1 \end{aligned}$$

Le lemme suivant est une conséquence des lemmes précédents.

**Lemme 4.8** *Soient $K_1, K_2 \subseteq \mathcal{M}$ et $v \in V$.*

1. *$v \in [\![K_1 \vee K_2]\!]$ ssi $v \in [\![K_1]\!] \cup [\![K_2]\!]$.*
2. *$v \in [\![K_1 \wedge K_2]\!]$ ssi $v \in [\![K_1]\!] \cap [\![K_2]\!]$.*
3. *$v \in [\![\neg K_1]\!]$ ssi $v \notin [\![K_1]\!]$.*

La proposition 4.4 suit du lemme 4.8.

**Proposition 4.4** *Pour tout $K_1, K_2 \subseteq \mathcal{M}$,*

$$[\![K_1]\!] \subseteq [\![K_2]\!] \quad \textit{ssi} \quad \mathbf{cf}(K_1 \wedge \neg K_2) = \emptyset$$

Soit $c \in \mathbb{N}$. On note $\mathcal{M}_c$ le sous-ensemble des matrices de bornes définies sur $\mathcal{B}_c$. La proposition 4.5 suit des lemmes précédents.

**Proposition 4.5** *Pour tout $c \in \mathbb{N}$, $\mathcal{M}_c$ satisfait les propriétés suivantes :*

1. *$\mathcal{M}_c$ est fini.*
2. *Pour tout $K_1, K_2 \in \mathcal{M}_c$ on a : $K_1 \vee K_2 \subseteq \mathcal{M}_c$, $K_1 \wedge K_2 \subseteq \mathcal{M}_c$, $\neg K_1 \subseteq \mathcal{M}_c$, $K_1[x := 0] \subseteq \mathcal{M}_c$.*

Pour toute matrice $K \subseteq \mathcal{M}_c$, on note $\hat{K}^c$ l'ensemble de matrices $\hat{K} \cap \mathcal{M}_c$, c'est-à-dire, $\hat{K}^c$ est le plus grand ensemble de matrices équivalent à $K$ qui est inclus dans $\mathcal{M}_c$.

### La fonction de représentation

Pour toute contrainte temporelle $\psi \in \Psi(\mathcal{H})$ on définit $\mathcal{R}(\psi) \subseteq \mathcal{M}$ par induction sur la structure de $\psi$ de la façon suivante.

Pour la contrainte *true* :

$$\mathcal{R}(true) = \{\mathbf{U}\}$$

Pour la contrainte $x \prec c$, $\mathcal{R}(x \prec c) = \{\mathbf{cf}(M)\}$ où $M$ est tel que pour tout $y, z \in \mathcal{H}^\sharp$,

$$M_{yz} = \begin{cases} (c, \prec) & \text{si } y = x \text{ et } z = \mathbf{0}, \\ \mathbf{U}_{yz} & \text{sinon.} \end{cases}$$

Pour la contrainte $x - y \prec c$, $\mathcal{R}(x - y \prec c) = \{\mathbf{cf}(M)\}$ où $M$ est tel que pour tout $z, w \in \mathcal{H}^\sharp$,

$$M_{zw} = \begin{cases} (c, \prec) & \text{si } z = x \text{ et } w = y, \\ \mathbf{U}_{zw} & \text{sinon.} \end{cases}$$

Pour la contrainte $\neg\psi_1$ :

$$\mathcal{R}(\neg\psi_1) = \mathbf{cf}(\neg\mathcal{R}(\psi_1))$$

Pour la contrainte $\psi_1 \wedge \psi_2 \in \Psi(\mathcal{H})$ :

$$\mathcal{R}(\psi_1 \wedge \psi_2) = \mathbf{cf}(\mathcal{R}(\psi_1) \wedge \mathcal{R}(\psi_2))$$

Pour la contrainte $\psi_1 \vee \psi_2 \in \Psi(\mathcal{H})$ :

$$\mathcal{R}(\psi_1 \vee \psi_2) = \mathcal{R}(\psi_1) \vee \mathcal{R}(\psi_2)$$

La proposition 4.6 suit des lemmes et propositions précédents.

**Proposition 4.6** *Pour toute contrainte temporelle $\psi \in \Psi(\mathcal{H})$, $[\![\psi]\!] = [\![\mathcal{R}(\psi)]\!]$.*

La fonction de représentation peut être étendue à l'opérateur d'initialisation d'horloge. Pour $\psi_1 \in \Psi(\mathcal{H})$ et $x \in \mathcal{H}$, on définit :

$$\mathcal{R}(\psi_1[x := 0]) = \mathcal{R}(\psi_1[x := 0])$$

D'après le corollaire 4.2 on a la proposition suivante.

**Proposition 4.7** *Pour toute contrainte temporelle $\psi \in \Psi(\mathcal{H})$ et $x \in \mathcal{H}$, $[\![\psi[x := 0]]\!] = [\![\mathcal{R}(\psi[x := 0])]\!]$.*

La proposition suivante est une conséquence de la proposition 4.5.

**Proposition 4.8** *Soit $\psi_1, \psi_2 \in \Psi(\mathcal{H})$ et soit $c \in \mathbb{N}$ une constante telle que, pour $i = 1, 2$, $\mathcal{R}(\psi_i) \subseteq \mathcal{M}_c$. On a :*

1. $\mathcal{R}(\psi_1 \vee \psi_2) \subseteq \mathcal{M}_c$
2. $\mathcal{R}(\psi_1 \wedge \psi_2) \subseteq \mathcal{M}_c$
3. $\mathcal{R}(\psi_1[x := 0]) \subseteq \mathcal{M}_c$
4. $\mathcal{R}(\neg\psi_1) \subseteq \mathcal{M}_c$

La représentation des contraintes temporelles au moyen des unions de matrices de bornes donne une méthode effective pour décider si l'ensemble caractéristique de $\psi_1$ est inclus dans l'ensemble caractéristique de $\psi_2$. En effet, la proposition suivante suit des propositions 4.4 et 4.6.

**Proposition 4.9** *Pour tout $\psi_1, \psi_2 \in \Psi$,*

$$[\![\psi_1]\!] \subseteq [\![\psi_2]\!] \quad \textit{ssi} \quad \mathbf{cf}(\mathcal{R}(\psi_1) \wedge \neg\mathcal{R}(\psi_2)) = \emptyset$$

Finalement, a partir de $\mathcal{R}(\psi)$ il est possible de construire une contrainte temporelle $\psi'$ équivalente à $\psi$. La contrainte $\psi'$ est obtenue comme la disjonction des contraintes correspondantes à chacune des matrices de $\mathcal{R}(\psi)$. En effet, soit $\psi'$ la contrainte :

$$\bigvee_{M \in \mathcal{R}(\psi)} \psi_M$$

où $\psi_M$ est la contrainte :

$$\bigwedge_{x \in \mathcal{H}} -M_{\mathbf{0}x} \prec x \prec M_{x\mathbf{0}} \wedge \bigwedge_{x \neq y \in \mathcal{H}} x - y \prec M_{xy}$$

Nous avons $[\![\psi]\!] = [\![\psi']\!]$.

### 4.2.4 La fonction de représentation des prédicats d'état

La fonction de représentation définie pour les contraintes temporelles peut être étendue naturellement pour les prédicats d'état. Soit $\phi \in \Phi(\mathcal{H})$ tel que $\mathcal{D}(\phi) = \bigcup_{s \in S}(s, \psi_s)$. On note $\mathcal{R}(\mathcal{D}(\phi))$ l'ensemble des matrices utilisées dans la représentation de $\phi$ :

$$\mathcal{R}(\mathcal{D}(\phi)) = \bigvee_{s \in S} \mathcal{R}(\psi_s)$$

La proposition suivante est une conséquence de la proposition 4.8.

**Proposition 4.10** *Soit $\phi_1, \phi_2 \in \Phi(\mathcal{H})$ et soit $c \in \mathbb{N}$ une constante telle que, pour $i = 1, 2$, $\mathcal{R}(\mathcal{D}(\phi_i)) \subseteq \mathcal{M}_c$. On a :*

1. $\mathcal{R}(\mathcal{D}(\phi_1 \vee \phi_2)) \subseteq \mathcal{M}_c$
2. $\mathcal{R}(\mathcal{D}(\phi_1 \wedge \phi_2)) \subseteq \mathcal{M}_c$
3. $\mathcal{R}(\mathcal{D}(\phi_1[x := 0])) \subseteq \mathcal{M}_c$
4. $\mathcal{R}(\mathcal{D}(\neg\phi_1)) \subseteq \mathcal{M}_c$

## 4.3 Le calcul de l'opérateur ▷

Nous présentons dans cette section une méthode effective de calcul de l'opération $Q_1 \triangleright Q_2$ où $Q_1$ et $Q_2$ sont des ensembles caractéristiques de deux prédicats d'état. La méthode est basée sur la représentation de $Q_1$ et $Q_2$ par des ensembles d'états symboliques de la forme $\bigcup_{s\in S}(s, \psi_s^i)$, pour $i = 1, 2$.

Soient $G = \langle S, H, E, s_I, \delta\rangle$ un graphe temporisé et $T = \langle Q, \rightarrow, q_I\rangle$ le modèle de $G$. $G$ et $T$ sont fixés dans cette section. Soient $Q_1$ et $Q_2$ deux prédicats d'état. D'après la définition 3.9, calculer $Q_1 \triangleright Q_2$ consiste à déterminer :

- tous les *prédécesseurs instantanés* des états de $Q_2$, i.e., tous les états de $Q_2$ plus tous ceux qui peuvent atteindre un état de $Q_2$ par une transition qui correspond à franchir un arc du graphe $G$, et
- tous les *prédécesseurs temporels* de ces derniers, i.e., tous les états qui peuvent atteindre un prédécesseur instantané d'un état de $Q_2$, par une transition temporelle, telle que tous les états intermédiaires sont dans $Q_1$ ou $Q_2$.

### 4.3.1 Les prédécesseurs instantanés

Soit $s_1, s_2 \in S$ deux sommets de $G$, $e = \langle s_1, a, \psi, H', s_2\rangle \in E$ un arc de $G$ de $s_1$ vers $s_2$. Soit $(s_2, \psi_2)$ un état symbolique. On définit :

$$pre_e(s_2, \psi_2) = (s_1, \psi \wedge \delta_{s_1} \wedge (\psi_2 \wedge \delta_{s_2})[H' := 0])$$

c'est-à-dire, $pre_e(s_2, \psi_2)$ est l'état symbolique qui caractérise les états prédécesseurs des états caractérisés par $(s_2, \psi_2)$, par une transition instantanée à travers l'arc $e$. Notons que $pre_e(s_2, \psi_2)$ est normalisé.

**Exemple 4.5** Considérons le graphe temporisé du contrôleur illustré par la figure 4.1. L'état symbolique qui caractérise les prédécesseurs instantanés de l'état symbolique $(5, y = 0 \wedge 35 < z)$ par l'arc de 2 vers 5, est calculé comme suit :

$$\begin{aligned} &pre_{25}(y = 0 \wedge 35 < z) = \\ &\quad = (y \leq 10 \wedge (y = 0 \wedge 35 < z)[y := 0]) \\ &\quad = (y \leq 10 \wedge (0 = 0 \wedge 35 < z)) \\ &\quad = (y \leq 10 \wedge 35 < z) \end{aligned}$$

L'ensemble de valuations caractérisé par la contrainte $pre_{25}(y = 0 \wedge 35 < z)$ est illustré par la figure 4.2 □

Pour un état symbolique $(s, \psi)$, on définit :

$$pre(s, \psi) = \bigcup_{e\in E} pre_e(s, \psi)$$

c'est-à-dire, $pre(s, \psi)$ est l'ensemble des états symboliques qui caractérisent les prédécesseurs instantanés des états caractérisés par $(s, \psi)$, par une transition à travers un arc $e$ du graphe $G$.

**Exemple 4.6** Pour l'état symbolique $(5, y = 0 \wedge 35 < z)$, on a :

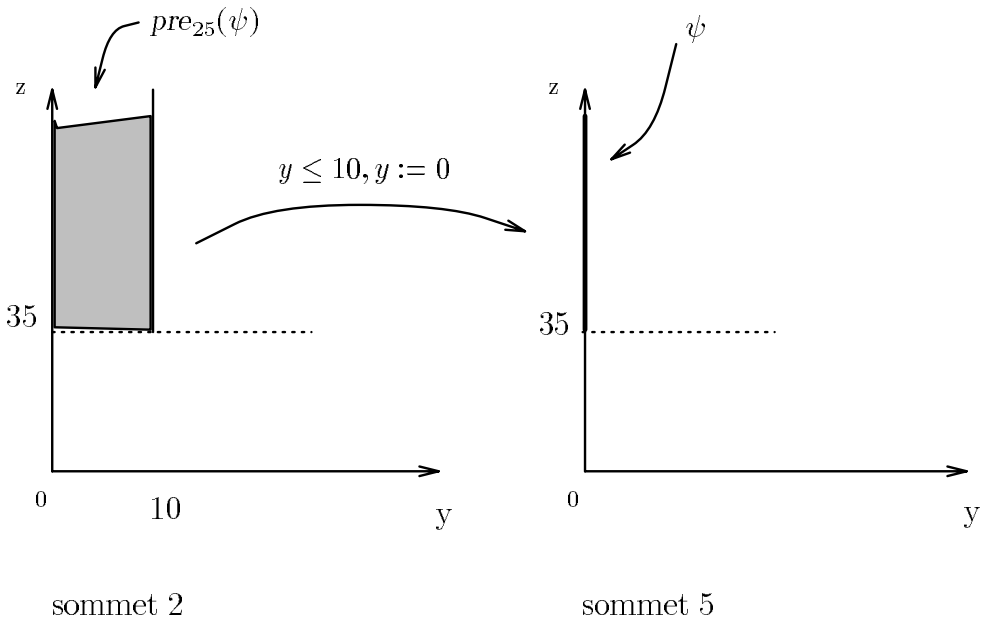



Figure 4.2: $pre_{25}(y = 0 \wedge 35 < z)$.

- Pour l'arc de 2 vers 5 :

$$pre_{25}(5, y = 0 \wedge 35 < z) = (2, y \leq 10 \wedge 35 < z)$$

- Pour l'arc de 3 vers 5 :

$$pre_{35}(5, y = 0 \wedge 35 < z) = (3, y < 20 \wedge 35 < z)$$

Par conséquent, $pre(5, y = 0 \wedge 35 < z)$ est :

$$(2, y \leq 10 \wedge 35 < z) \cup (3, y < 20 \wedge 35 < z)$$

□

**Lemme 4.9** *Pour tout état symbolique* $(s, \psi)$,

$$[\![pre(s, \psi)]\!] = \{(s', v') \mid \exists v \in V, a \in Act.\ v \textit{ satisfait } \psi \textit{ et } (s', v') \xrightarrow{a} (s, v).\}$$

**Preuve.**

$\subseteq$ Soit $(s', v') \in [\![pre(s, \psi)]\!]$. Donc, il existe $e = \langle s', a, \psi', H', s\rangle \in E$ tel que $(s', v') \in [\![pre_e(s, \psi)]\!]$. Or, par définition, $pre_e(s, \psi) = (s, \psi' \wedge \delta_{s'} \wedge (\psi \wedge \delta_{s_2})[H' := 0])$. Par conséquent, $v'$ satisfait $\psi' \wedge \delta_{s'} \wedge (\psi \wedge \delta_{s_2})[H' := 0]$. Soit $v = v'[H' := 0]$, donc $v$ satisfait $\psi$ et $\delta_{s_2}$. Par conséquent, d'après la sémantique des graphes temporisés, on a $(s', v') \xrightarrow{a} (s, v)$.

$\supseteq$ Soient $(s', v')$ et $v$ tels que $v$ satisfait $\psi$ et $(s', v') \xrightarrow{a} (s, v)$. Donc, il existe $e = \langle s', \alpha, \psi', H', s\rangle \in E$ tel que $v'$ satisfait $\psi'$ et $v = v'[H' := 0]$. De plus, $v'$ satisfait $\delta_{s'}$ et, puisque $v$ satisfait $\psi \wedge \delta_s$, on a $v'$ satisfait $(\psi \wedge \delta_s)[H' := 0]$. Donc, $(s', v') \in [\![pre_e(s, \psi)]\!]$ et par conséquent, $(s', v') \in [\![pre(s, \psi)]\!]$.

■

### 4.3.2 Les prédécesseurs temporels

Soient $(s, \psi_1)$ et $(s, \psi_2)$ deux états symboliques. On définit :

$$(s, \psi_1) \triangleright_t (s, \psi_2) = (s, \exists t \in \mathsf{R}^+.((\psi_2 \wedge \delta_s) + t \wedge \forall t' \leq t.(\psi_1 \vee \psi_2) + t'))$$

c'est-à-dire, $(s, \psi_1) \triangleright_t (s, \psi_2)$ caractérise les états prédécesseurs des états caractérisés par $(s, \psi_2)$, par une transition temporelle telle que toutes les valuations intermédiaires satisfont $\psi_1 \vee \psi_2$. Notons qu'il n'est pas nécessaire d'imposer la condition $\delta_s$ pour les instants $t' \leq t$ car $\delta_s$ est fermée en arrière (cf. définition 2.4) et par conséquent, si $\delta_s$ est satisfaite à l'instant $t$, elle est aussi satisfaite à tout instant $t' \leq t$.

La proposition suivante assure que $(s, \psi_1) \triangleright_t (s, \psi_2)$ est en effet un état symbolique.

**Lemme 4.10** *Soient $\psi_1, \psi_2 \in \Psi(\mathcal{H})$. Il existe $\psi \in \Psi(\mathcal{H})$ tel que*

$$[\![\psi]\!] = [\![\exists t \in \mathsf{R}^+.((\psi_2 \wedge \delta_s) + t \wedge \forall t' \leq t.(\psi_1 \vee \psi_2) + t'))]\!]$$

La preuve du lemme 4.10 est donnée dans la section 4.3.4. Celle-ci consiste à montrer que les quantificateurs peuvent être éliminés et que le résultat obtenu est une contrainte temporelle $\psi \in \Psi(\mathcal{H})$. Nous illustrons les idées essentielles avec un exemple.

**Exemple 4.7** Soit $\psi_1$ et $\psi_2$ les contraintes temporelles suivantes :

$$\begin{aligned} \psi_1 &= (0 < x \vee 0 < y) \\ \psi_2 &= (y \leq 10 \wedge 35 < z) \end{aligned}$$

Pour le sommet 2 du graphe du contrôleur de température illustré par la figure 4.1, l'état symbolique $(2, \psi_1) \triangleright_t (2, \psi_2)$ est calculée de la façon suivante. En appliquant la définition ci-dessus, on obtient :

$$\begin{aligned} &\exists t \in \mathsf{R}^+.((y + t \leq 10 \wedge 35 < z + t) \wedge \\ &\quad \forall t' \leq t.((0 < x + t' \vee 0 < y + t') \vee (y + t' \leq 10 \wedge 35 < z + t'))) \end{aligned} \tag{4.8}$$

Le quantificateur universel peut être dualisé en un quantificateur existentiel. On obtient donc :

$$\neg\exists t' \in \mathsf{R}^+.(t' \leq t \wedge \neg(0 < x + t' \vee 0 < y + t') \wedge \neg(y + t' \leq 10 \wedge 35 < z + t'))$$

qui est équivalent à :

$$\begin{aligned} &\neg\exists t' \in \mathsf{R}^+.(t' \leq t \wedge x + t' \leq 0 \wedge 10 < y + t' \leq 0) \\ &\wedge \\ &\neg\exists t' \in \mathsf{R}^+.(t' \leq t \wedge x + t' \leq 0 \wedge y + t' \leq 0 \wedge z + t' \leq 35) \end{aligned}$$

qui est équivalent à :

$$\neg\exists t' \in \mathsf{R}^+.(t' \leq t \wedge x + t' \leq 0 \wedge y + t' \leq 0 \wedge z + t' \leq 35) \tag{4.9}$$

Il est possible d'éliminer le quantificateur. De 4.9 on déduit :

$$0 \leq t' \leq t \wedge t' \leq -x \wedge t' \leq -y \wedge t' \leq 35 - z$$

d'où on déduit :

$$x \leq 0 \wedge y \leq 0 \wedge z \leq 35$$

dont la négation est :

$$0 < x \vee 0 < y \vee 35 < z \tag{4.10}$$

Donc, de 4.8 et 4.10 on obtient :

$$\exists t \in \mathsf{R}^+.(y + t \leq 10 \wedge 35 < z + t \wedge (0 < x \vee 0 < y \vee 35 < z))$$

et le quantificateur est éliminé par la même procédure. Le résultat obtenu est alors :

$$(y \leq 10 \wedge y - z < -25 \wedge (0 < x \vee 0 < y \vee 35 < z))$$

L'état symbolique $(2, \psi_1) \triangleright_t (2, \psi_2)$ est illustré par la figure 4.3. □

Le lemme 4.11 suit immédiatement de la définition de l'opérateur $\triangleright_t$.

**Lemme 4.11** *Soient* $(s, \psi_1)$ *et* $(s, \psi_2)$ *deux états symboliques.*

$$[\![(s, \psi_1) \triangleright_t (s, \psi_2)]\!] = \{(s, v) \mid \exists t \in \mathsf{R}^+.\ (s, v) \xrightarrow{t} (s, v+t) \wedge v+t \in [\![\psi_2]\!] \wedge \forall t' \leq t.\ v+t' \in [\![\psi_1 \vee \psi_2]\!]\}$$

### 4.3.3 Les prédécesseurs en un pas

A partir des opérateurs *pre* et $\triangleright_t$ il est possible à présent de calculer l'ensemble caractéristique de $[\![\phi_1]\!] \triangleright [\![\phi_2]\!]$ pour deux prédicats d'état $\phi_1$ et $\phi_2$.

Soient $\phi_1, \phi_2 \in \Phi(\mathcal{H})$ tels que $\mathcal{D}(\phi_i) = \bigcup_{s \in S}(s, \psi_s^i)$, pour $i = 1, 2$, et $P_s$ l'ensemble des états symboliques défini par :

$$P_s = \{(s, \psi \wedge \psi_s^1) \mid \exists (s', \psi_{s'}^2).\ (s, \psi) \in pre(s', \psi_{s'}^2)\}$$

c'est-à-dire, $P_s$ caractérise les états $(s, v)$ qui satisfont $\phi_1$ à partir desquels il est possible d'atteindre un état $(s', v')$ qui satisfait $\phi_2$ en franchissant un arc $e$ du graphe. Soit $\xi_s$ la contrainte temporelle définie par :

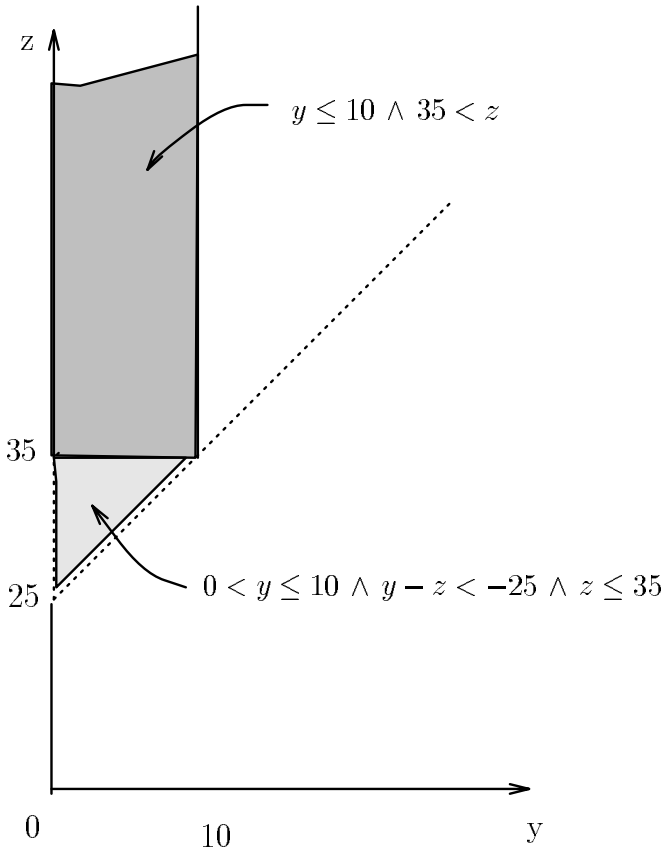
$$\xi_s = \bigvee_{(s,\psi) \in P_s} \psi$$

On définit :

$$\bigcup_{s \in S}(s, \psi_s^1) \blacktriangleright \bigcup_{s \in S}(s, \psi_s^2) = \bigcup_{s \in S}(s, \psi_s^1) \triangleright_t (s, \xi_s \vee \psi_s^2)$$

Le lemme suivant est une conséquence des lemmes 4.9 et 4.10.

**Lemme 4.12** *Soient* $\phi_1, \phi_2 \in \Phi(\mathcal{H})$. $\mathcal{D}(\phi_1) \blacktriangleright \mathcal{D}(\phi_2)$ *est un ensemble d'états symboliques de la forme* $\bigcup_{s \in S}(s, \psi)$.




sommet 2

Figure 4.3: $(2, \psi_1) \triangleright_t (2, \psi_2)$.

**Proposition 4.11** *Soient* $\phi_1, \phi_2 \in \Phi(\mathcal{H})$.

$$[\![\phi_1]\!] \triangleright [\![\phi_2]\!] = [\![\mathcal{D}(\phi_1) \blacktriangleright \mathcal{D}(\phi_2)]\!]$$

**Preuve.** Pour $i = 1, 2$ soit $\mathcal{D}(\phi_i) = \bigcup_{s \in S}(s, \psi^i_s)$.

$\subseteq$ Soit $(s, v) \in [\![\phi_1]\!] \triangleright [\![\phi_2]\!]$. Ils existent $(s', v') \in [\![\phi_2]\!]$ et $t \in \mathsf{R}^+$, tels que $(s, v) \overset{t}{\triangleright} (s', v')$ et pour tout $t' \leq t$, $(s, v + t') \in [\![\phi_1]\!] \cup [\![\phi_2]\!]$.

Par conséquent, d'après le lemme 4.9 on a $(s', v') \in [\![(s', \psi^2_{s'})]\!]$ et $(s, v + t) \in [\![pre(s', \psi^2_{s'})]\!]$.

De plus, $(s, v + t) \in [\![(s, \psi^1_s \vee \psi^2_s)]\!]$, et donc $(s, v + t) \in [\![(s, \xi_s \vee \psi^2_s)]\!]$. Par conséquent, d'après le lemme 4.11 on a $(s, v) \in [\![(\triangleright_t s, \xi_s \vee \psi^2_s)]\!]$.

On a donc, $(s, v) \in [\![\mathcal{D}(\phi_1) \blacktriangleright \mathcal{D}(\phi_2)]\!]$.

$\supseteq$ Soit $(s, v) \in [\![\mathcal{D}(\phi_1) \blacktriangleright \mathcal{D}(\phi_2)]\!]$. Par définition, on a $(s, v) \in [\![(s, \psi^1_s) \triangleright_t (s, \xi_s \vee \psi^2_s)]\!]$.

D'après le lemme 4.11, il existe $t \in \mathsf{R}^+$ tel que $(s,v) \xrightarrow{t} (s,v+t)$ et $v+t \in [\![\xi_s \vee \psi_s^2]\!]$ et pour tout $t' \leq t$, $v+t' \in [\![\psi_s^1]\!] \cup [\![\xi_s \vee \psi_s^2]\!]$. Or, $[\![\xi_s]\!] \subseteq [\![\psi_s^1]\!]$. Donc, pour tout $t' \leq t$, $v+t' \in [\![\psi_s^1]\!] \cup [\![\psi_s^2]\!]$.

On distingue deux cas :

- Soit $v+t \in [\![\psi_s^2]\!]$. Donc, on a $(s,v+t) \xrightarrow{0} (s,v+t)$, et par conséquent, $(s,v) \overset{t}{\triangleright} (s,v+t)$.
- Soit $v+t \in [\![\xi_s]\!]$. Donc, d'après le lemme 4.9, il existe $(s',v')$ tel que $(s,v+t) \xrightarrow{0} (s',v')$. Par conséquent, $(s,v) \overset{t}{\triangleright} (s',v')$.

Donc, $(s,v) \in [\![\phi_1]\!] \triangleright [\![\phi_2]\!]$.

■

**Exemple 4.8** Considérons le graphe temporisé du contrôleur de température illustré par la figure 4.1. Soit $Q_1, Q_2$ tels que :

$$\begin{aligned} Q_1 &= (0,\mathit{true}) \cup (1, x=0) \cup (2, x>0 \vee y>0) \cup \\ &\quad (3, y<20) \cup (4, y \leq 35) \cup (5, y=0) \cup (6,\mathit{true}) \\ Q_2 &= (5, y=0 \wedge 35<z) \end{aligned}$$

On a :

$$\mathit{pre}(Q_2) = (2, y \leq 10 \wedge 35<z) \cup (3, y<20 \wedge 35<z)$$

et

$$\begin{aligned} &(2, x>0 \vee y>0) \triangleright_t (2, y \leq 10 \wedge 35<z) = \\ &\quad = (2, y \leq 10 \wedge y-z<-25 \wedge (x>0 \vee y>0 \vee 35<z)) \\ &(3, y<20) \triangleright_t (3, y<20 \wedge 35<z) = \\ &\quad = (3, y<20 \wedge y-z<-15) \\ &(5, y=0) \triangleright_t (5, y=0 \wedge 35<z) = \\ &\quad = (5, y=0 \wedge 35<z) \end{aligned}$$

Par conséquent :

$$\begin{aligned} &Q_1 \blacktriangleright Q_2 = \\ &\quad = (2, y \leq 10 \wedge y-z<-25 \wedge (x>0 \vee y>0 \vee 35<z)) \\ &\quad \cup (3, y<20 \wedge y-z<-15) \\ &\quad \cup (5, y=0 \wedge 35<z) \end{aligned}$$

□

### 4.3.4 Elimination des quantificateurs existentiels

Nous présentons ici la preuve du lemme 4.10. La preuve consiste à montrer qu'il est possible d'éliminer les quantificateurs du prédicat :

$$\exists t \in \mathsf{R}^+.((\psi_2 \wedge \delta_s) + t \wedge \forall t' \leq t.(\psi_1 \vee \psi_2) + t') \tag{4.11}$$

et que le résultat est une contrainte temporelle.

Il est possible de dualiser le quantificateur universel par un quantificateur existentiel. On a donc :

$$\exists t \in \mathsf{R}^+.((\psi_2 \wedge \delta_s) + t \wedge \neg\exists t' \in \mathsf{R}^+.(t' \leq t \wedge \neg(\psi_1 \vee \psi_2) + t')) \tag{4.12}$$

D'après le résultat de la section 4.2.4, les contraintes $\neg(\psi_1 \vee \psi_2)$ et $\psi_2 \wedge \delta_s$ peuvent être remplacées par des formules équivalentes $\bigvee_{i\in I} M_i$ et $\bigvee_{j\in J} M_j$, respectivement. On a donc :

$$\exists t \in \mathsf{R}^+.((\bigvee_{j\in J} M_j) + t \wedge \neg\exists t' \in \mathsf{R}^+.(\bigvee_{i\in I} t' \leq t \wedge M_i + t')) \tag{4.13}$$

qui est équivalent à:

$$\exists t \in \mathsf{R}^+.(\bigvee_{j\in J}(M_j + t) \wedge \neg \bigvee_{i\in I} \exists t' \in \mathsf{R}^+.(t' \leq t \wedge M_i + t')) \tag{4.14}$$

Nous montrons à présent le lemme suivant.

**Lemme 4.13** *Le résultat de l'élimination du quantificateur existentiel d'une formule de la forme :*

$$\exists t' \in \mathsf{R}^+.(t' \leq t \wedge M + t')$$

*est une contrainte de la forme :*

$$M_1 \wedge (M_2 + t)$$

**Preuve.** Soit $M$ la contrainte :

$$\bigwedge_{x\in\mathcal{H}} -M_{\mathbf{0}x} \prec x \prec M_{x\mathbf{0}} \wedge \bigwedge_{x\neq y\in\mathcal{H}} x - y \prec M_{xy}$$

Nous supposons que $M$ est sous forme canonique. Donc, $M_{xy} \sqsubseteq M_{x\mathbf{0}} \otimes M_{\mathbf{0}y}$.

$$\begin{aligned}
&\exists t' \in \mathsf{R}^+.t' \leq t \wedge M + t' = \\
&= \exists t' \in \mathsf{R}^+.(t' \leq t \wedge \bigwedge_{x\in\mathcal{H}} -M_{\mathbf{0}x} \prec x + t' \prec M_{x\mathbf{0}} \wedge \bigwedge_{x\neq y\in\mathcal{H}} x - y \prec M_{xy}) && (4.15)\\
&= \exists t' \in \mathsf{R}^+.(t' \leq t \wedge \bigwedge_{x\in\mathcal{H}} -M_{\mathbf{0}x} - x \prec t' \prec M_{x\mathbf{0}} - x \wedge \bigwedge_{x\neq y\in\mathcal{H}} x - y \prec M_{xy}) && (4.16)
\end{aligned}$$

De 4.16 on obtient les inéquations suivantes :

$$\text{pour tout } (x \neq y) \in \mathcal{H},\ -M_{\mathbf{0}y} - y \prec M_{x\mathbf{0}} - x, \text{ et pour tout } x \in \mathcal{H},\ 0 \prec x. \tag{4.17}$$

De 4.16 et 4.17 on obtient :

$$\bigwedge_{x\in\mathcal{H}} -M_{\mathbf{0}x} \prec x + t \wedge \bigwedge_{x\in\mathcal{H}} x \prec M_{x\mathbf{0}} \wedge \bigwedge_{x\neq y\in\mathcal{H}} x - y \prec M_{xy} \tag{4.18}$$

Les formules 4.15 et 4.18 sont équivalentes. Soit $v \in V$ une valuation d'horloges.

1. Supposons que $v$ satisfait 4.15. Alors, pour tout $x \in \mathcal{H}$,

$$-M_{\mathbf{0}x} \prec v(x) + t' \prec M_{x\mathbf{0}}$$

Comme $0 \leq t' \leq t$ on obtient :

$$-M_{\mathbf{0}x} \prec v(x) + t \quad \text{et} \quad v(x) \prec M_{x\mathbf{0}}$$

Donc, $v$ satisfait 4.18.

2. Supposons que $v$ satisfait 4.18. Alors,

   (a) Si pour tout $x \in \mathcal{H}$, $-M_{\mathbf{0}x} \prec v(x)$, alors $v$ satisfait 4.15 pour $t' = 0$.

   (b) S'il existe $x \in \mathcal{H}$ tel que $v(x) < -M_{\mathbf{0}x}$, alors $v$ satisfait 4.15 pour $t' = -M_{\mathbf{0}x} - v(x) + \epsilon_x$, où $\epsilon_x$ est tel que $0 \leq \epsilon_x \prec M_{y\mathbf{0}} + M_{\mathbf{0}x} - M_{yx}$.

Finalement, la contrainte 4.18 est de la forme $M_1 \wedge (M_2 + t)$. ■

Donc, à partir de 4.14 et du lemme 4.13 on déduit :

$$\exists t \in \mathsf{R}^+.(\bigvee_{j\in J}(M_j + t) \wedge \neg \bigvee_{i\in I}(M_i^1 \wedge (M_i^2 + t))) \tag{4.19}$$

Cette formule peut être mise sous la forme :

$$\exists t \in \mathsf{R}^+.(\bigvee_{j\in J}(M_j + t) \wedge \bigvee_{k\in K}(M_k^1 \wedge (M_k^2 + t)) \tag{4.20}$$

qui peut être aussi mise sous la forme :

$$\bigvee_{l\in L} \exists t \in \mathsf{R}^+.(M_l^1 \wedge (M_l^2 + t)) \tag{4.21}$$

Il est alors possible d'éliminer les quantificateurs comme il a été montré dans le lemme 4.13. Ceci prouve le lemme 4.10.

### 4.3.5 Propriété du calcul de l'opérateur $\triangleright$

Comme nous l'avons déjà vu dans la section 4.2, le calcul des ensembles caractéristiques des prédicats d'état $\phi_1 \vee \phi_2$, $\phi_1 \wedge \phi_2$, $\neg\phi_1$ et $\phi_1[x := 0]$, ne fait pas augmenter la valeur absolue des constantes dans les matrices utilisées dans la représentation de $\phi_1$ et $\phi_2$. Ceci est aussi vrai pour le calcul de l'opérateur $\triangleright$, comme le montre la preuve du lemme 4.10. Par conséquent, on a la proposition suivante.

**Proposition 4.12** *Pour tous prédicats d'états $\phi_1, \phi_2 \in \Phi(\mathcal{H})$ et $c \in \mathsf{N}$, tels que $\mathcal{R}(\mathcal{D}(\phi_i)) \subseteq \mathcal{M}_c$ pour $i = 1, 2$, on a $\mathcal{R}(\mathcal{D}(\phi_1) \blacktriangleright \mathcal{D}(\phi_2)) \subseteq \mathcal{M}_c$.*

Ce résultat est utilisé dans la section 4.4 pour développer un algorithme de model-checking symbolique pour les formules de TCTL.

## 4.4 Evaluation symbolique des formules de TCTL

Dans cette section nous présentons un algorithme de model-checking pour une formule $\varphi$ de TCTL et un graphe bien temporisé $G$. L'algorithme consiste à calculer pour $\varphi$ l'ensemble $\mathcal{S}(\varphi)$ d'états symboliques qui représente l'ensemble caractéristique de $\varphi$ relativement à l'ensemble $\Delta_G$ de séquences divergentes de $G$.

Soit $G = \langle S, H, E, s_I, \delta \rangle$ un graphe bien temporisé et soit $\varphi$ une formule de TCTL. $\mathcal{S}(\varphi)$ est construit par induction sur la structure de $\varphi$ de la façon suivante :

$$
\begin{array}{rcl}
\mathcal{S}(p) & = & \mathcal{D}(p) \\
\mathcal{S}(x \prec c) & = & \mathcal{D}(x \prec c) \\
\mathcal{S}(x - y \prec c) & = & \mathcal{D}(x - y \prec c) \\
\mathcal{S}(x.\varphi_1) & = & \mathcal{S}(\varphi_1)[x := 0] \\
\mathcal{S}(\neg\varphi_1) & = & \neg\mathcal{S}(\varphi_1) \\
\mathcal{S}(\varphi_1 \wedge \varphi_2) & = & \mathcal{S}(\varphi_1) \wedge \mathcal{S}(\varphi_2) \\
\mathcal{S}(\varphi_1 \,\exists\mathcal{U}\, \varphi_2) & = & \bigvee_{k \in \mathsf{N}} X_k \\
 & & \text{où } X_0 = \mathcal{S}(\varphi_2) \text{ et } X_{k+1} = X_k \vee \mathcal{S}(\varphi_1) \blacktriangleright X_k \\
\mathcal{S}(\varphi_1 \,\forall\mathcal{U}\, \varphi_2) & = & \bigvee_{k \in \mathsf{N}} X_k \\
 & & \text{où } X_0 = \mathcal{S}(\varphi_2) \text{ et } X_{k+1} = X_k \vee \neg(Y^k[z := 0]), \text{ et} \\
 & & Y^k = \bigvee_{j \in \mathsf{N}} Y^k_j, \\
 & & \text{avec } Y^k_0 = (\neg\mathcal{S}(\varphi_1) \wedge \neg X_k) \vee \mathcal{S}(z > 1) \text{ et} \\
 & & Y^k_{j+1} = Y^k_j \vee \neg X_k \blacktriangleright Y^k_j
\end{array}
$$

Le théorème suivant prouve la correction de l'algorithme.

**Théorème 4.1** *Pour tout graphe bien temporisé $G$ et pour toute formule $\varphi$ de* TCTL,

1. $\mathcal{S}(\varphi) = \bigcup_{s \in S}(s, \psi_s)$, *et*
2. $[\![\varphi]\!]_{\Delta_G} = [\![\mathcal{S}(\varphi)]\!]$.

**Preuve.** Soit $\varphi$ une formule de TCTL. Par la suite, on écrit $[\![\varphi]\!]$ au lieu de $[\![\varphi]\!]_{\Delta_G}$.

Si $\varphi$ est de la forme $p$, $x \prec c$ ou $x - y \prec c$, alors $\mathcal{S}(\varphi) = \mathcal{D}(\varphi)$ et donc la propriété 1 est vraie par construction et la propriété 2 suit de la proposition 4.1.

Supposons que les propriétés 1 et 2 soient vraies pour $\varphi_1$ et $\varphi_2$. On peut aisément vérifier que les deux propriétés sont vraies pour $x.\varphi_1$, $\neg\varphi_1$ et $\varphi_1 \wedge \varphi_2$.

Nous montrons à présent le résultat pour $\varphi_1 \,\exists\mathcal{U}\, \varphi_2$ et $\varphi_1 \,\forall\mathcal{U}\, \varphi_2$.

• Soit $\varphi$ la formule $\varphi_1 \,\exists\mathcal{U}\, \varphi_2$. D'après la proposition 3.4, on a :

$$[\![\varphi_1 \,\exists\mathcal{U}\, \varphi_2]\!] = \mu X.([\![\varphi_2]\!] \vee [\![\varphi_1]\!] \triangleright X)$$

Or, $F(X) = [\![\varphi_2]\!] \vee [\![\varphi_1]\!] \triangleright X$ est une fonctionnelle monotone (cf. proposition 3.1). Donc, le plus petit point fixe de $F$ peut être calculé comme : $\bigvee_{j \in \mathsf{N}} F_j$ où $F_0 = [\![\varphi_2]\!]$ et $F_{j+1} = F_j \vee [\![\varphi_1]\!] \triangleright F_j$.

Par conséquent, on a :

$$[\![\varphi_1 \exists\mathcal{U} \varphi_2]\!] = \bigvee_{j\in\mathsf{N}} F_j$$

Or, par hypothèse inductive, pour $i = 1, 2$, on a $[\![\varphi_i]\!] = [\![\mathcal{S}(\varphi_i)]\!]$. Donc, pour tout $j \in \mathsf{N}$, $F_j = [\![X_j]\!]$ où $X_0 = \mathcal{S}(\varphi_2)$ et $X_{j+1} = X_j \vee \mathcal{S}(\varphi_1) \blacktriangleright X_j$. Par conséquent, on a :

$$[\![\varphi_1 \exists\mathcal{U} \varphi_2]\!] = \bigvee_{j\in\mathsf{N}} [\![X_j]\!] = [\![\mathcal{S}(\varphi_1 \exists\mathcal{U} \varphi_2)]\!]$$

et donc la propriété 2 est vraie.

Soit $c \in \mathsf{N}$ tel que $\mathcal{R}(\mathcal{S}(\varphi_i)) \subseteq \mathcal{M}_c$ pour $i = 1, 2$. On peut vérifier par récurrence sur $j$ que pour tout $j \in \mathsf{N}$ :

- $X_j = \bigcup_{s\in S}(s, \psi_s^j)$
- $\mathcal{R}(X_j) \subseteq \mathcal{M}_c$

De $\mathcal{R}(X_j) \subseteq \mathcal{M}_c$ pour tout $j \in \mathsf{N}$ on déduit $\mathcal{R}(\bigvee_{j\in\mathsf{N}} X_j) \subseteq \mathcal{M}_c$, et par conséquent $\mathcal{R}(\bigvee_{j\in\mathsf{N}} X_j)$ est fini (†).

De plus, puisque $F$ est monotone on a $F_j \subseteq F_{j+1}$, et donc $[\![X_j]\!] \subseteq [\![X_{j+1}]\!]$.

Donc, il doit exister $k \in \mathsf{N}$ tel que $[\![X_k]\!] = [\![X_{k+1}]\!]$. Supposons que ceci ne soit pas vrai. On a donc, pour tout $j \in \mathsf{N}$, $[\![X_j]\!] \subset [\![X_{j+1}]\!]$, et par conséquent, $\hat{\mathcal{R}}^c(X_j) \subset \hat{\mathcal{R}}^c(X_{j+1})$. Ceci signifie qu'un nombre infini de matrices est nécessaire pour représenter l'ensemble $F$, ce qui contredit (†).

On a donc :

$$\begin{aligned}
\mathcal{S}(\varphi_1 \exists\mathcal{U} \varphi_2) &= \bigvee_{j\le k} X_j \\
&= \bigvee_{j\le k} \bigcup_{s\in S} (s, \psi_s^j) \\
&= \bigcup_{s\in S} (s, \bigvee_{j\le k} \psi_s^j)
\end{aligned}$$

Par conséquent, la propriété 1 est vraie.

• Soit $\varphi$ la formule $\varphi_1 \forall\mathcal{U} \varphi_2$. D'après la proposition 3.9, on a :

$$[\![\varphi_1 \forall\mathcal{U} \varphi_2]\!] = \mu X.([\![\varphi_2]\!] \vee \neg z.\mu Y.((\neg[\![\varphi_2]\!] \wedge \neg X) \vee [\![z > 1]\!] \vee \neg X \triangleright Y))$$

Soient $F$ et $L$ tels que :

$$\begin{aligned}
F(X) &= [\![\varphi_2]\!] \vee \neg(L(X)[z := 0]) \\
L(X) &= \mu Y.H(X, Y) \\
H(X, Y) &= (\neg[\![\varphi_2]\!] \wedge \neg X) \vee [\![z > 1]\!] \vee \neg X \triangleright Y
\end{aligned}$$

(1) On prouve à présent que $F$ est monotone. Soient $Q, Q' \subseteq \mathcal{Q}$ tels que $Q \subseteq Q'$. Il faut montrer $F(Q) \subseteq F(Q')$.

Soit $Q'' \subseteq \mathcal{Q}$, d'après la proposition 3.1 on a $\neg Q' \triangleright Q'' \subseteq \neg Q \triangleright Q''$. Donc, $H(Q', Q'') \subseteq H(Q, Q'')$ pour tout $Q'' \subseteq \mathcal{Q}$, et par conséquent, $L(Q') \subseteq L(Q)$.

On a ensuite, $L(Q')[z := 0] \subseteq L(Q)[z := 0]$ et donc $\neg L(Q')[z := 0] \subseteq \neg L(Q)[z := 0]$, et par conséquent $F(Q) \subseteq F(Q')$.

(2) On montre à présent que pour un $Q \subseteq \mathcal{Q}$ donné, la fonction $H(Q)$, telle que $H(Q)(Y) = H(Q, Y)$, est monotone. Soient $Q', Q'' \subseteq \mathcal{Q}$ tels que $Q' \subseteq Q''$. D'après la proposition 3.1 on a $\neg Q \triangleright Q' \subseteq \neg Q \triangleright Q''$, et donc $H(Q, Q') \subseteq H(Q, Q'')$. Par conséquent $H(Q)$ est monotone.

Donc, de (1) on déduit que le plus petit point fixe de $F$ peut être calculé comme : $\bigvee_{k \in \mathsf{N}} F_k$ où $F_0 = [\![\varphi_2]\!]$ et $F_{k+1} = F_k \vee \neg(L(F_k)[z := 0])$.

De plus, de (2) on déduit que le plus petit point fixe de $H(F_k)$ peut être calculé comme : $\bigvee_{j \in \mathsf{N}} H_j^k$ où $H_0^k = (\neg[\![\varphi_2]\!] \wedge \neg X) \vee [\![z > 1]\!]$ et $H_{j+1}^k = H_j^k \vee \neg F_k \triangleright H_j^k$.

Par conséquent, on a :

$$[\![\varphi_1 \forall\mathcal{U} \varphi_2]\!] = \bigvee_{j \in \mathsf{N}} F_j$$

Or, par hypothèse inductive, pour $i = 1, 2$, on a $[\![\varphi_i]\!] = [\![\mathcal{S}(\varphi_i)]\!]$.

Donc, pour tout $k \in \mathsf{N}$, $F_k = [\![X_k]\!]$ où :

$$\begin{array}{rcl} X_0 & = & \mathcal{S}(\varphi_2) \\ X_{k+1} & = & X_k \vee \neg(Y^k[z := 0]) \end{array}$$

avec $L(F_k) = [\![Y^k]\!] = [\![\bigvee_{j \in \mathsf{N}} Y_j^k]\!]$, et pour tout $j \in \mathsf{N}$, $H_j^k = [\![Y_j^k]\!]$ où :

$$\begin{array}{rcl} Y_0^k & = & (\neg\mathcal{S}(\varphi_1) \wedge \neg X_k) \vee \mathcal{S}(z > 1) \\ Y_{j+1}^k & = & Y_j^k \vee \neg X_k \blacktriangleright Y_j^k \end{array}$$

Par conséquent, on a :

$$[\![\varphi_1 \forall\mathcal{U} \varphi_2]\!] = [\![\mathcal{S}(\varphi_1 \forall\mathcal{U} \varphi_2)]\!]$$

et donc la propriété 2 est vraie.

Soit $c \in \mathsf{N}$ tel que $\mathcal{R}(\mathcal{S}(\varphi_i)) \subseteq \mathcal{M}_c$ pour $i = 1, 2$. Comme dans le cas précédent, on a pour tout $k, j \in \mathsf{N}$ :

- $X_k = \bigcup_{s \in S}(s, \psi_s^k)$
- $Y_j^k = \bigcup_{s \in S}(s, \psi_s^{kj})$
- $\mathcal{R}(X_k) \subseteq \mathcal{M}_c$
- $\mathcal{R}(Y_j^k) \subseteq \mathcal{M}_c$

Donc, $\mathcal{R}(\bigvee_{k \in \mathsf{N}} X_k) \subseteq \mathcal{M}_c$ et $\mathcal{R}(\bigvee_{k \in \mathsf{N}} Y_j^k) \subseteq \mathcal{M}_c$. et par conséquent $\mathcal{R}(\mathcal{S}(\varphi_1 \forall\mathcal{U} \varphi_2))$ est fini.

De plus, puisque $H(F_k)$ est monotone on a $H_j^k \subseteq H_{j+1}^k$, et donc $[\![Y_j^k]\!] \subseteq [\![X_{j+1}]\!]$, et par conséquent, il existe $j0 \in \mathsf{N}$ tel que $[\![Y_{j0}^k]\!] = [\![Y_{j0+1}^k]\!]$.

Donc, pour tout $k \in \mathsf{N}$,

$$\begin{aligned} \bigvee_{j\in\mathsf{N}} Y_j^k &= \bigvee_{j\leq j0} Y_j^k \\ &= \bigvee_{j\leq j0} \bigcup_{s\in S} (s, \psi_s^{kj}) \\ &= \bigcup_{s\in S} (s, \bigvee_{j\leq j0} \psi_s^{kj}) \end{aligned}$$

Par le même raisonnement on obtient qu'il existe $k0 \in \mathsf{N}$ tel que $[\![X_{k0}]\!] = [\![X_{k0+1}]\!]$.

On a donc :

$$\begin{aligned} \mathcal{S}(\varphi_1 \forall\mathcal{U} \varphi_2) &= \bigvee_{k\leq k0} X_k \\ &= \bigvee_{k\leq k0} \bigcup_{s\in S} (s, \psi_s^k) \\ &= \bigcup_{s\in S} (s, \bigvee_{k\leq k0} \psi_s^k) \end{aligned}$$

Par conséquent la propriété 1 est vraie. ■

## 4.5 Détection des graphes bien temporisés

Soient $G = \langle S, H, E, s_I, \delta\rangle$ un graphe temporisé et $T = \langle \mathcal{Q}, \rightarrow, q_I\rangle$ le modèle de $G$. Il est important de pouvoir détecter si $G$ est bien temporisé. D'après la proposition 3.10, $G$ est bien temporisé si et seulement si l'ensemble $\mathcal{Q}$ est inclus dans $[\![w.\exists\Diamond w = 1]\!]_{\Sigma_G}$.

A titre d'exemple, nous vérifions que le graphe illustré par la figure 3.3 (cf. exemple 3.2) n'est pas bien temporisé. Soit $\varphi$ la formule $w.\exists\Diamond w = 1$. On a :

$$\begin{aligned} \mathcal{S}(\varphi) &= \mathcal{S}(\exists\Diamond w = 1)[w := 0] \\ &= (\bigvee_{j\in\mathsf{N}} X_j)[w := 0] \end{aligned}$$

où :

$$\begin{aligned} X_0 &= Q_2 \\ X_{j+1} &= X_j \vee Q_1 \blacktriangleright X_j \end{aligned}$$

avec :

$$\begin{aligned} Q_1 &= \mathcal{S}(true) \\ &= (0, x \leq 5 \wedge y \leq 7) \cup (1, z \leq 8) \cup (2, z \leq 10) \\ Q_2 &= \mathcal{S}(w = 1) \\ &= (0, x \leq 5 \wedge y \leq 7 \wedge w = 1) \cup (1, z \leq 8 \wedge w = 1) \cup (2, z \leq 10 \wedge w = 1) \end{aligned}$$

Dans la première itération on obtient :

$$\begin{aligned} X_1 &= X_0 \vee Q_1 \blacktriangleright X_0 \\ &= (0,(w<1 \wedge x-w\leq 4 \wedge y-w\leq 6) \vee (x\leq 5 \wedge y\leq 7 \wedge w=1)) \\ &\quad \cup(1,(w<1 \wedge z-w\leq 7) \vee (z\leq 8 \wedge w=1)) \\ &\quad \cup(2,(w<1 \wedge z-w\leq 9) \vee (z\leq 10 \wedge w=1)) \end{aligned}$$

Dans la deuxième itération on obtient :

$$\begin{aligned} X_2 &= X_1 \vee Q_1 \blacktriangleright X_1 \\ &= X_1 \vee \\ &\quad (0,(x\leq 5 \wedge y\leq 7 \wedge (w-x<-4 \vee w-y<6))) \\ &\quad \cup(1,(z\leq 8 \wedge x-y<5 \wedge y-x<7 \wedge x-w\leq 4 \wedge y-w\leq 6 \wedge w-z<-7)) \\ &\quad \cup(2,(z\leq 10 \wedge x-y<5 \wedge y-x<7 \wedge x-w\leq 4 \wedge y-w\leq 6 \wedge w-z<-9)) \end{aligned}$$

Dans la troisième itération on obtient :

$$\begin{aligned} X_3 &= X_2 \vee Q_1 \blacktriangleright X_2 \\ &= X_2 \vee \\ &\quad (1,(x\leq 5 \wedge y\leq 7 \wedge z\leq 8 \wedge (w-z<-7 \wedge (w-x<-4 \vee w-y<-6)))) \\ &\quad \cup(2,(x\leq 5 \wedge y\leq 7 \wedge z\leq 10 \wedge (w-z<-9 \wedge (w-x<-4 \vee w-y<-6)))) \end{aligned}$$

On trouve finalement que $X_4 = X_3$.

On obtient donc :

$$\begin{aligned} \mathcal{S}(\varphi) &= X_3[w:=0] \\ &= (0,(x\leq 5 \wedge y\leq 7)) \\ &\quad \cup(1,(x\leq 5 \wedge y\leq 7 \wedge z\leq 8) \vee z\leq 7) \\ &\quad \cup(2,(x\leq 5 \wedge y\leq 7 \wedge z\leq 8) \vee z\leq 9) \end{aligned}$$

Par conséquent, le graphe $G$ n'est pas bien temporisé, car la formule $\varphi$, n'est pas satisfaite par les états définis par :

$$(1,(7<z \wedge z\leq 8 \wedge (5<x \vee 7<y))) \cup (2,(9<z \wedge z\leq 10 \wedge (5<x \vee 7<y)))$$

## 4.6 Exemples de vérification

Nous reprenons ici les exemples du contrôleur de température 2.2 de la souris 2.1 et nous utilisons l'algorithme de model-checking symbolique présenté dans la section 4.4 pour vérifier des propriétés de ces systèmes exprimées par des formules de TCTL.

### 4.6.1 Le contrôleur de température

La figure 4.1 illustre le graphe du contrôleur de température présenté dans l'exemple 2.2, où les valeurs des délais $t_1$, $t_2$ et $t_{max}$ sont 10, 20 et 35, respectivement, et les sommets sont étiquetés par des propositions. La proposition $r$ indique que le contrôleur a reçu l'ordre de réfrigérer. La proposition $b$ indique que le contrôleur a commencé le mouvement des barres.

La spécification du système établie que pour de raisons de sécurité :

si le contrôleur ne reçoit pas une nouvelle ordre de réfrigérer avant un temps $t_{max}$ depuis la dernière ordre reçue, alors il doit également réfrigérer.

Cette propriété s'exprime par la formule suivante :

$$\varphi \;=\; r \Rightarrow \neg z.\neg b\, \exists\mathcal{U}\, (r \wedge 35 < z)$$

c'est-à-dire, il n'est pas possible d'atteindre un état qui satisfait $r$ en un temps supérieur à 35s à partir d'un état qui satisfait $r$ sans passer par un état qui satisfait $b$. Autrement dit, si deux qui satisfont $r$ sont séparés de plus de 35s, alors il existe un état qui satisfait $b$ entre eux.

Soit $\varphi'$ la sous-formule :

$$\varphi' \;=\; \neg b\, \exists\mathcal{U}\, (r \wedge 35 < z)$$

Nous avons donc :

$$\mathcal{S}(\varphi') \;=\; \bigvee_{j\in\mathsf{N}} X_j$$

où :

$$\begin{aligned} X_0 &= Q_2 \\ X_{j+1} &= X_j \vee Q_1 \blacktriangleright X_j \end{aligned}$$

avec :

$$\begin{aligned} Q_1 &= \mathcal{S}(\neg b) \\ &= (0, \mathit{true}) \cup (1, x = 0) \cup (3, x \leq 35 \wedge y \leq 20) \\ &\quad \cup (4, y \leq 35) \cup (5, y = 0) \cup (6, \mathit{true}) \\ Q_2 &= \mathcal{S}(r \wedge 35 < z) \\ &= (1, x = 0 \wedge 35 < z) \cup (5, y = 0 \wedge 35 < z) \end{aligned}$$

Dans la première itération on obtient :

$$\begin{aligned} X_1 &= X_0 \vee Q_1 \blacktriangleright X_0 \\ &= (0, \mathit{true}) \cup (1, x = 0 \wedge 35 < z) \cup (3, y < 20 \wedge y - z < -15) \\ &\quad \cup (4, x < 35 \wedge x < z) \cup (5, y = 0 \wedge 35 < z) \end{aligned}$$

Dans la deuxième itération on obtient :

$$\begin{aligned} X_2 &= X_1 \vee Q_1 \blacktriangleright X_1 \\ &= (0, \mathit{true}) \cup (1, x = 0 \wedge 35 < z) \\ &\quad \cup (3, (y = 20 \wedge x - y < 15 \wedge x < z) \vee \\ &\qquad (y \leq 20 \wedge x < z \wedge z - y \leq 15) \vee (y < 20 \wedge y - z < -15)) \\ &\quad \cup (4, x < 35 \wedge x < z) \cup (5, y = 0 \wedge 35 < z) \end{aligned}$$

On trouve alors $X_3 = X_2$. On a donc :

$$\begin{aligned} &\mathcal{S}(z.\neg b\, \exists\mathcal{U}\, r \wedge 35 < z) = \\ &\quad = z.X_2 \end{aligned}$$

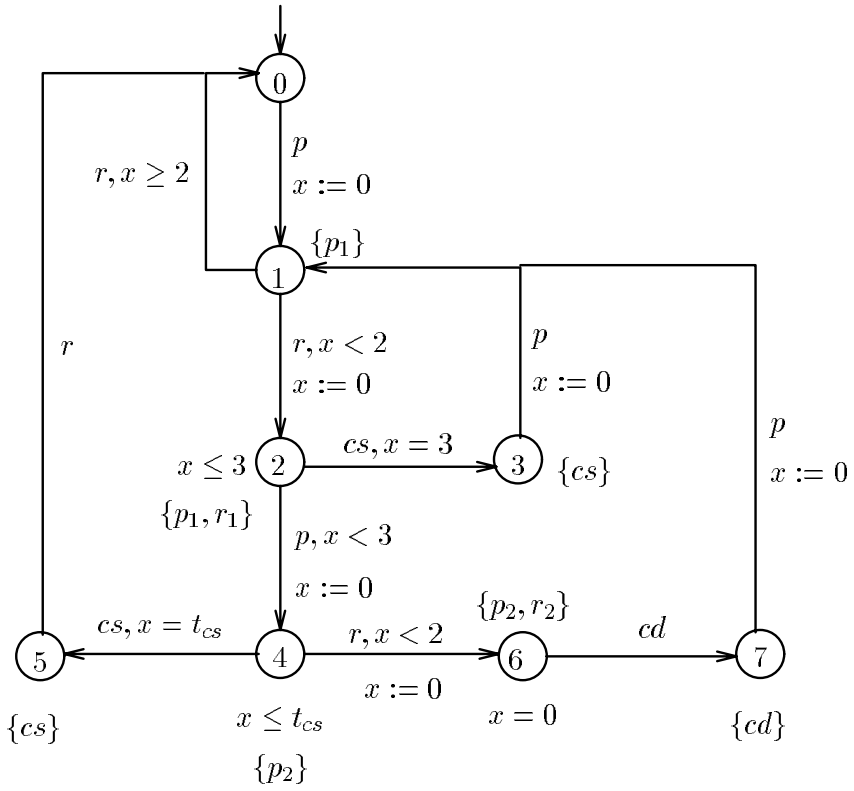



Figure 4.4: Graphe de la souris.

$$
\begin{aligned}
= \quad & (0, \mathit{true}) \cup (1, x = 0 \wedge 35 < 0) \\
& \cup (3, (y = 20 \wedge x - y < 15 \wedge x < 0) \vee \\
& \qquad (y \leq 20 \wedge x < 0 \wedge 0 - y \leq 15) \vee (y < 20 \wedge y - 0 < -15)) \\
& \cup (4, x < 35 \wedge x < 0) \cup (5, y = 0 \wedge 35 < 0) \\
= \quad & (0, \mathit{true})
\end{aligned}
$$

Donc, $\mathcal{S}(r) \subseteq \neg(0, \mathit{true})$. Par conséquent, le système satisfait la propriété.

### 4.6.2 La souris

Considérons le graphe temporisé illustré par la figure 4.4. Ce graphe est celui présenté dans l'exemple 2.1 où les valeurs de $t_{sc}$ et $t_{dc}$ sont 3 et 2, respectivement, et les sommet sont étiquetés par des propositions. Pour $i = 1, 2$, la proposition $p_i$ ($r_i$) indique que le bouton de la souris a été pressé (relâché) $i$ fois. Les propositions $sc$ et $dc$ indiquent la détection d'un click simple ou d'un click double respectivement.

Nous souhaitons vérifier la propriété de vivacité bornée suivante :

> si le bouton a été pressé et relâché une fois alors inévitablement avant un temps égal à $t_{sc} + t_{dc}$ le système détecte un click simple ou un click double.

Cette propriété est exprimée en TCTL par la formule :

$$\varphi \quad = \quad (p_1 \wedge r_1 \wedge x = 0) \Rightarrow \forall\Diamond_{<5}(sc \vee dc)$$

Soit $\varphi'$ la sous-formule :

$$\varphi' \;=\; \forall\Diamond_{<5}(sc \vee dc)$$

D'après la proposition 3.8, $\varphi'$ est équivalente à la formule :

$$\varphi'' \;=\; \neg z.(\neg(sc \vee dc)\, \exists\mathcal{U}\, 5 \leq z)$$

On a donc :

$$\mathcal{S}(\varphi'') \;=\; \neg(\bigvee_{j\in\mathsf{N}} X_j)[z := 0]$$

où :

$$\begin{aligned} X_0 &= Q_2 \\ X_{j+1} &= X_j \vee Q_1 \blacktriangleright X_j \end{aligned}$$

avec :

$$\begin{aligned} Q_1 &= \mathcal{S}(\neg(sc \vee dc)) \\ &= (0, true) \cup (1, true) \cup (2, x \leq 3) \cup (4, x \leq 2) \cup (6, x = 0) \\ Q_2 &= \mathcal{S}(5 \leq z) \\ &= (0, 5 \leq z) \cup (1, 5 \leq z) \cup (2, x \leq 3 \wedge 5 \leq z) \cup (3, 5 \leq z) \\ &\quad \cup (4, x \leq 2 \wedge 5 \leq z) \cup (6, x = 0 \wedge 5 \leq z) \cup (7, 5 \leq z) \end{aligned}$$

Le point fixe est calculé itérativement comme $X = \bigvee_{j\in\mathsf{N}} X_j$ où :

$$\begin{aligned} X_0 &= Q_2 \\ X_{j+1} &= X_j \vee Q_1 \blacktriangleright X_j \end{aligned}$$

Dans la première itération on obtient :

$$\begin{aligned} X_1 &= X_0 \vee Q_1 \blacktriangleright X_0 \\ &= (0, true) \cup (1, true) \cup (2, (z < 5 \wedge x - z \leq -2) \vee (x \leq 3 \wedge 5 \leq z)) \\ &\quad \cup (3, 5 \leq z) \cup (4, (z < 5 \wedge x - z \leq -3) \vee (x \leq 2 \wedge 5 \leq z)) \\ &\quad \cup (5, 5 \leq z) \cup (6, x = 0 \wedge 5 \leq z) \cup (7, 5 \leq z) \end{aligned}$$

Dans la deuxième itération on obtient :

$$\begin{aligned} X_2 &= X_1 \vee Q_1 \blacktriangleright X_1 \\ &= (0, true) \cup (1, true) \\ &\quad \cup (2, (z < 5 \wedge x - z \leq -2) \vee (x \leq 3 \wedge 5 \leq z) \vee (x < 3 \wedge x < z \wedge z - x < 2)) \\ &\quad \cup (3, 5 \leq z) \cup (4, (z < 5 \wedge x - z \leq -3) \vee (x \leq 2 \wedge 5 \leq z)) \\ &\quad \cup (5, 5 \leq z) \cup (6, x = 0 \wedge 5 \leq z) \cup (7, 5 \leq z) \end{aligned}$$

On trouve alors $X_3 = X_2$. On obtient donc :

$$\begin{aligned} \mathcal{S}(\varphi') &= \neg(X_2[z := 0]) \\ &= (2, x \leq 3) \cup (3, true) \cup (4, x \leq 2) \cup (6, x = 0) \cup (7, true) \end{aligned}$$

Nous constatons que :

$$\mathcal{S}(p_1 \wedge r_1 \wedge x = 0) = (2, x = 0)$$

est contenu dans $\mathcal{S}(\varphi'')$. Par conséquent, le graphe vérifie la propriété.

## 4.7 Conclusions du chapitre

Nous avons présenté dans ce chapitre un algorithme de model-checking symbolique qui calcule l'ensemble caractéristique d'une formule de TCTL comme une disjonction de contraintes temporelles sur les horloges.

L'algorithme développé consiste à représenter les prédicats d'états par des unions d'états symboliques et à calculer itérativement des points fixes. L'ensemble caractéristique d'une formule est représentable par un nombre fini de matrices de bornes, ce qui assure la terminaison des calculs des points fixes. La correction de l'algorithme est basée sur les résultats montrés dans le chapitre 3.

L'algorithme a été implémenté dans l'outil KRONOS [NSY92, NOSY92, OY92] de vérification de systèmes temps-réel. Les performances obtenues avec KRONOS sont illustrées dans le chapitre 8 à l'aide de quelques exemples significatifs.

# Chapitre 5

# L'algèbre de processus temporisés ATP

Dans ce chapitre nous présentons l'algèbre de processus temporisés ATP [NRSV90, NS90, NS91, NSY91, NSY92, Nic92]. Outre des opérateurs classiques de préfixage, choix non déterministe et composition parallèle trouvés dans les algèbres de processus communicants, ATP comporte des opérateurs qui permettent de décrire des contraintes temporelles.

ATP comporte un nombre restreint d'opérateurs temporels dont le choix est guidé par le souci de simplifier l'étude théorique. Pourtant, avec cet ensemble réduit d'opérateurs de base il est possible de décrire les opérateurs temporels qui apparaissent dans les langages concrets, comme les timeouts, les retards et les watchdogs.

Nous présentons tout d'abord la syntaxe de l'algèbre. La sémantique opérationnelle est décrite au moyen des systèmes de transitions étiquetées définis dans la section 2.1. Une étude approfondie d'ATP est réalisée dans [Nic92].

## 5.1 Syntaxe

Soit *Var* un ensemble de variables de processus.

**Définition 5.1** Les termes de l'algèbre ATP sont définis par la syntaxe suivante :

$$\begin{aligned} P \quad ::= \quad & \mathsf{idle} \mid X \mid aP \mid \mathsf{d}(a)P \mid P \ \mathsf{watchdog}[A](t)\ P \mid P + P \mid P \,||\, P \\ & \mid \mathsf{restrict}\ A\ \mathsf{in}\ P \mid \mathsf{rec}X \cdot P \end{aligned}$$

où $X \in Var$, $a \in Act$, $t \in \mathsf{R}^{>0}$ et $A \subseteq Act$. □

La signification intuitive des différents opérateurs est la suivante :

- L'opérateur **idle** représente le processus qui ne peut exécuter aucune action.
- $a\cdot$ est l'opérateur de *préfixage immédiat*. Le processus $aP$ doit exécuter l'action $a$ immédiatement, pour se comporter ensuite comme le processus $P$.
- L'opérateur $\mathsf{d}(\cdot)\cdot$ permet de *retarder* l'exécution d'une action. A la différence de $aP$, le processus $\mathsf{d}(a)P$ peut différer indéfiniment l'exécution de l'action $a$.

- Le *watchdog* est un opérateur temporel standard. Le terme $P_1$ $\mathsf{watchdog}[A](t)$ $P_2$ représente un processus qui lance l'exception $P_2$ si le processus $P_1$ n'a pas exécuté une action de $A$ dans un délai donné $t$.
- Les opérateurs $\cdot + \cdot$ et $\cdot \,||\, \cdot$ représentent le *choix non déterministe* et la *composition parallèle*.
- Le processus $\mathsf{restrict}\ A\ \mathsf{in}\ P$ se comporte comme $P$, mais ne peut pas exécuter les actions de $A$.
- Finalement, $\mathsf{rec}X\cdot$ est l'opérateur de récursion.

**Définition 5.2** Une variable $X$ est *libre* dans $P$ s'il existe une occurrence de $X$, dite occurrence libre, qui n'est dans la portée d'aucun $\mathsf{rec}X\cdot$, sinon $X$ est *liée*. L'ensemble des variables libres de $P$ est noté $vl(P)$. Un terme $P \in \mathrm{ATP}$ est *fermé* si $vl(P) = \emptyset$. □

Soit $P, P' \in \mathrm{ATP}$ et soit $X \in Var$. La substitution syntaxique de $P'$ à toutes les occurrences libres de la variable $X$ dans $P$ est notée $P[P'/X]$.

L'emploi de l'opérateur de récursion est parfois peu agréable en pratique. Pour décrire plus aisément les processus nous utilisons des systèmes d'équations mutuellement récursives. Par exemple, pour le terme $\mathsf{rec}X \cdot a\,\mathsf{d}(b)\,X$ nous écrivons :

$$\begin{aligned} P_1 &\stackrel{\text{def}}{=} a\,P_2 \\ P_2 &\stackrel{\text{def}}{=} \mathsf{d}(b)\,P_1 \end{aligned}$$

## 5.2 Sémantique opérationnelle

Nous décrivons à présent la sémantique opérationnelle du langage, qui associe à un terme un système de transitions étiquetées. Dans cette section nous ne considérons que des termes fermés.

La sémantique opérationnelle d'ATP est donnée au moyen d'une relation de transition entre les termes de l'algèbre définie par induction structurelle à la Plotkin [Plo81].

**Définition 5.3** Le modèle d'un terme $P \in \mathrm{ATP}$, noté $\mathcal{T}[\![P]\!]$, est le système de transitions étiquetées $\langle \mathrm{ATP}, \rightarrow, P\rangle$, où la relation de transition $\rightarrow$ est la plus petite relation satisfaisant aux règles ci-dessous. □

Le processus **idle** ne peut exécuter aucune action, il ne peut que laisser passer le temps. On a donc pour chaque $t \in \mathsf{R}^{>0}$,

$$\text{idle)} \quad \mathsf{idle} \xrightarrow{t} \mathsf{idle}$$

Le préfixage immédiat d'un terme $P$ par une action $a$ est noté $aP$. Le processus $aP$ ne peut qu'exécuter l'action $a$, et il doit le faire immédiatement, n'ayant pas la possibilité de laisser passer le temps. Ce comportement est décrit par la règle suivante :

$$\text{acti)} \quad aP \xrightarrow{a} P$$

Le préfixage immédiat permet de modéliser l'*urgence*. Au contraire, l'exécution d'une action $a$ peut être retardée indéfiniment au moyen de l'opérateur $\mathsf{d}(a)$. La sémantique du processus $\mathsf{d}(a)P$ est définie par les deux règles suivantes :

$$\text{actd)} \quad \mathsf{d}(a)P \xrightarrow{a} P \qquad\qquad \text{delay)} \quad \mathsf{d}(a)P \xrightarrow{t} \mathsf{d}(a)P$$

pour tout $t \in \mathsf{R}^{>0}$.

Le terme $P_1$ watchdog$[A](t)\ P_2$ définit un processus qui lance l'exception $P_2$ si $P_1$ n'a pas exécuté une action $a \in Act$ avant l'instant $t$. La sémantique du watchdog est définie par les règles suivantes. Pour les actions,

$$\text{wda1)}\quad \frac{P_1 \xrightarrow{a} P_1' \wedge a \in A}{P_1 \text{ watchdog}[A](t)\ P_2 \xrightarrow{a} P_1'}$$

$$\text{wda2)}\quad \frac{P_1 \xrightarrow{a} P_1' \wedge a \notin A}{P_1 \text{ watchdog}[A](t)\ P_2 \xrightarrow{a} P_1' \text{ watchdog}[A](t)\ P_2}$$

et pour le passage du temps,

$$\text{wdt1)}\quad \frac{P_1 \xrightarrow{t} P_1'}{P_1 \text{ watchdog}[A](t)\ P_2 \xrightarrow{t} P_2}$$

$$\text{wdt2)}\quad \frac{P_1 \xrightarrow{t'} P_1' \wedge t' < t}{P_1 \text{ watchdog}[A](t)\ P_2 \xrightarrow{t'} P_1' \text{ watchdog}[A](t - t')\ P_2}$$

La règle suivante assure l'additivité du temps :

$$\text{wdt3)}\quad \frac{P_1 \xrightarrow{t} P_1' \wedge P_2 \xrightarrow{t'} P_2'}{P_1 \text{ watchdog}[A](t)\ P_2 \xrightarrow{t+t'} P_2'}$$

Les règles de sémantique du choix non déterministe sont les suivantes :

$$\text{choice1)}\quad \frac{P_1 \xrightarrow{a} P_1'}{P_1 + P_2 \xrightarrow{a} P_1'} \qquad \text{choice2)}\quad \frac{P_2 \xrightarrow{a} P_2'}{P_1 + P_2 \xrightarrow{a} P_2'} \qquad \text{choice3)}\quad \frac{P_1 \xrightarrow{t} P_1' \wedge P_2 \xrightarrow{t} P_2'}{P_1 + P_2 \xrightarrow{t} P_1' + P_2'}$$

Le seul écoulement du temps ne sert pas à résoudre un choix non déterministe.

Le terme $P_1 \,||\, P_2$ définit la composition parallèle de $P_1$ et $P_2$. Cet opérateur se comporte comme celui d'ACP [BK84]. Les processus $P_1$ et $P_2$ peuvent exécuter de manière indépendante des actions locales ou communiquer selon la fonction de communication $\cdot \mid \cdot$ définie sur l'ensemble d'actions. Pour les actions exécutées de façon indépendante :

$$\text{par1)}\quad \frac{P_1 \xrightarrow{a} P_1'}{P_1 \,||\, P_2 \xrightarrow{a} P_1' \,||\, P_2} \qquad \text{par2)}\quad \frac{P_2 \xrightarrow{a} P_2'}{P_1 \,||\, P_2 \xrightarrow{a} P_1 \,||\, P_2'}$$

pour la synchronisation :

$$\text{par3)}\quad \frac{P_1 \xrightarrow{a_1} P_1' \wedge P_2 \xrightarrow{a_1} P_2' \wedge a_1 \mid a_2 \neq \bot}{P_1 \,||\, P_2 \xrightarrow{a_1 \mid a_2} P_1' \,||\, P_2'}$$

et pour le passage du temps :

$$\text{par4)}\quad \frac{P_1 \xrightarrow{t} P_1' \wedge P_2 \xrightarrow{t} P_2'}{P_1 \,||\, P_2 \xrightarrow{t} P_1' \,||\, P_2'}$$

**Remarque 5.1** Les règles choice3) et par4) de la sémantique du choix non déterministe et de l'opérateur de composition parallèle traduisent exactement l'hypothèse de temps global : toutes les composantes d'un système ont la même notion du temps. Le passage du temps doit donc s'effectuer de manière synchrone. □

L'opérateur **restrict** $A$ **in** $P$ permet de restreindre les actions qui peuvent être exécutées par le processus $P$. En effet, celui-ci ne peut exécuter aucune action de l'ensemble $A \subseteq Act$.

$$\text{restr)} \quad \frac{P \xrightarrow{\ell} P' \wedge \ell \notin A}{\textbf{restrict } A \textbf{ in } P \xrightarrow{\ell} \textbf{restrict } A \textbf{ in } P'}$$

Finalement, la règle pour la récursion est la suivante :

$$\text{rec)} \quad \frac{P[\mathsf{rec}X \cdot P/X] \xrightarrow{\ell} P'}{\mathsf{rec}X \cdot P \xrightarrow{\ell} P'}$$

**Proposition 5.1** *Pour tout terme fermé* $P \in$ ATP, *le système de transitions étiquetées* $\langle \text{ATP}, \rightarrow, P \rangle$ *est un système temporisé.*

**Preuve.** cf. [Nic92]. ■

La seconde étape dans la définition de la sémantique formelle d'une algèbre de processus consiste à définir une relation d'équivalence entre les termes à partir de leurs modèles. Cette relation d'équivalence spécifie les termes dont les comportements sont considérés indistinguables. Nous choisissons ici la relation d'équivalence forte [Mil80, Par81, Mou92].

**Définition 5.4** Pour $i = 1, 2$ soit $T_i = \langle \mathcal{Q}_i, \rightarrow_i, q_{I_i} \rangle$. Soit $\mathcal{R} \subseteq \mathcal{Q}_1 \times \mathcal{Q}_2$. $\mathcal{R}$ est une *bisimulation forte* si pour tout $(q_1, q_2) \in \mathcal{R}$ et $\ell \in Lab$ :

1. $\forall q_1' \in \mathcal{Q}_1.\ q_1 \xrightarrow{\ell}_1 q_1' \Rightarrow \exists q_2' \in \mathcal{Q}_2.\ q_2 \xrightarrow{\ell}_2 q_2' \wedge (q_1', q_2') \in \mathcal{R}$
2. $\forall q_2' \in \mathcal{Q}_2.\ q_2 \xrightarrow{\ell}_2 q_2' \Rightarrow \exists q_1' \in \mathcal{Q}_1.\ q_1 \xrightarrow{\ell}_1 q_1' \wedge (q_1', q_2') \in \mathcal{R}$

Si $T_1 = T_2$, on dit que $\mathcal{R}$ est une auto-bisimulation. On écrit $q_1 \sim q_2$ s'il existe une bisimulation forte $\mathcal{R} \subseteq \mathcal{Q}_1 \times \mathcal{Q}_2$ telle que $(q_1, q_2) \in \mathcal{R}$. $\sim$ est la plus grande bisimulation forte et est appelée *équivalence forte*. On écrit $T_1 \sim T_2$ si $q_{I_1} \sim q_{I_2}$. □

**Définition 5.5** Pour tout $P, P' \in$ ATP, $P \sim P'$ si $\mathcal{T}[\![P]\!] \sim \mathcal{T}[\![P']\!]$. □

Une axiomatisation consistante pour l'équivalence forte est défini dans [Nic92].

### 5.2.1 Les systèmes bien temporisés

Il est possible de décrire en ATP des processus qui ne sont pas bien temporisés. Un exemple simple est le processus $P$ défini par $\mathsf{rec}X \cdot aX$. $P$ exécute un nombre infini de fois l'action $a$ en temps 0.

Par contre, le processus $P'$ défini par $\mathsf{rec}X \cdot \mathsf{d}(a)X$, est bien temporisé. En effet, toute séquence finie de pas peut toujours être prolongée par une séquence infinie telle que le temps diverge.

De plus, l'opérateur de restriction peut aussi "bloquer" le temps. On peut vérifier que le $P_1$ défini par **restrict** $a$ **in** $P'$ est équivalent à **idle**. En effet, le modèle de $P_1$ n'a que des transitions temporelles. En revanche, le terme $P_2$ défini par **restrict** $a$ **in** $P$ n'a aucune transition et par conséquent la progression du temps est bloquée. On a donc que $P_1$ est bien temporisé tandis que $P_2$ ne l'est pas.

En d'autres termes, la sémantique donnée aux opérateurs permet de distinguer le système **idle** du *blocage*. Un processus bloqué ne peut, ni exécuter une action, ni laisser passer le temps, alors que **idle** ne peut que laisser passer le temps indéfiniment.

## 5.3 Opérateurs dérivés

Nous montrons ici comment les opérateurs temporels qui apparaissent dans les langages de programmation de systèmes temps réel peuvent être modélisés dans ATP.

Bien entendu, ces opérateurs n'enrichissent pas l'algèbre du point de vue de la classe de modèles, mais ils apportent plutôt une facilité de description qui est utile pour écrire plus aisément des systèmes temporisés. C'est pourquoi nous les considérons comme des opérateurs dérivés.

### 5.3.1 Retard

Il est nécessaire parfois de retarder l'exécution d'un processus pendant un temps déterminé, par exemple, pour indiquer qu'un certain temps est nécessaire pour finir ou commencer une tâche. Cet opérateur correspond au $WAIT\,t$ de Timed CSP [Hoa78, RR88] et OCCAM [INM84].

**Définition 5.6** Pour $P \in \mathrm{ATP}$ et $t \in \mathsf{R}^{>0}$,

$$\mathsf{wait}(t)\ P = \mathsf{idle}\ \mathsf{watchdog}[](t)\ P$$

Le retard est donc un cas particulier du watchdog où le processus contrôlé par le watchdog est **idle** et l'ensemble $A$ est $\emptyset$. □

*Exemple.* Considérons une ligne de communication simple qui se comporte comme un tampon de taille 1 sans possibilité de perte. Le comportement de la ligne est décrit par le terme suivant :

$$L \stackrel{\text{def}}{=} \mathsf{d}(in)\,\mathsf{wait}(t)\,out\,L$$

La réception d'un message est signalée par l'action $in$. La fréquence à laquelle les messages arrivent n'est pas connue. Donc, l'exécution de l'action $in$ est indéfiniment retardée. Lorsque la ligne reçoit un message, elle le transmettra correctement après un délai de $t$ unités de temps. Ceci est modélisé par l'opérateur $\mathsf{wait}(t)$ . La transmission est signalée par l'action $out$, exécutée immédiatement après l'expiration du délai. □

### 5.3.2 Timeout

Le timeout est un opérateur temporel largement utilisé pour la programmation de systèmes temporisés. Un timeout surveille si un processus $P_1$ exécute une action dans un délai donné $t$, et déclenche une exception $P_2$ si le délai n'est pas respecté.

**Définition 5.7** Pour $P_1, P_2 \in \text{ATP}$ et $t \in \mathsf{R}^{>0}$,

$$P_1 \ \mathsf{timeout}(t)\ P_2 = P_1 \ \mathsf{watchdog}[Act](t)\ P_2$$

Le timeout est donc un cas particulier du watchdog où l'ensemble $A$ est $Act$. □

*Exemple.* Pour illustrer l'utilisation d'un timeout, nous considérons un distributeur automatique de café et de thé dont le comportement est décrit par :

$$\begin{aligned}
D_1 &\stackrel{\text{def}}{=} \mathsf{d}(monnaie)D_2 \\
D_2 &\stackrel{\text{def}}{=} (\mathsf{d}(cafe)\,D_3 + \mathsf{d}(the)\,D_4)\ \mathsf{timeout}(d)\ r_monnaie\,D_1 \\
D_3 &\stackrel{\text{def}}{=} \mathsf{wait}(c)\ p_cafe\,D_1 \\
D_4 &\stackrel{\text{def}}{=} \mathsf{wait}(t)\ p_the\,D_1
\end{aligned}$$

Initialement, le distributeur est prêt à accepter une pièce de monnaie. La fréquence d'utilisation de la machine n'est pas connue. Lorsqu'une pièce est introduite, il offre du café ou du thé. Si l'utilisateur ne fait pas son choix avant $d$ unités de temps, le distributeur rend la pièce de monnaie et revient à son état initial. Ce comportement est modélisé par un timeout. Une fois le choix fait, le distributeur produit la boisson commandée. La préparation d'une boisson prend un certain temps : $c$ unités de temps pour un café et $t$ pour un thé. Ceci est modélisé au moyen de l'opérateur $\mathsf{wait}()$ . □

### 5.3.3 Délai d'exécution

Cet opérateur correspond à la construction $\mathsf{do}\ P_1\ \mathsf{upto}\ d;\ P_2$ du langage ESTEREL [BC85, BdS91]. Le processus $P_1$ est interrompu au bout de $d$ unités de temps pour exécuter $P_2$.

**Définition 5.8** Pour $P_1, P_2 \in \text{ATP}$ et $t \in \mathsf{R}^{>0}$,

$$P_1 \ \mathsf{upto}(t)\ P_2 = P_1 \ \mathsf{watchdog}[](t)\ P_2$$

Le délai d'exécution est un cas particulier du watchdog où $A$ est l'ensemble vide. □

*Exemple.* L'utilisation du délai d'exécution est montrée par l'exemple suivant. Un capteur de température scrute l'environnement de façon périodique toutes les $p$ unités de temps. Ceci est modélisé par l'action *lire_temp*. Il écrit la valeur courante de la température au moyen de l'action *ecrire_temp* lorsqu'il en reçoit une demande. Le comportement du capteur est le suivant :

$$\begin{aligned}
C_1 &\stackrel{\text{def}}{=} lire_temp[C_2\ \mathsf{upto}(p)\,C_1] \\
C_2 &\stackrel{\text{def}}{=} \mathsf{d}(ecrire_temp)\,C_2
\end{aligned}$$

Le capteur peut exécuter l'action *put_temp* un nombre arbitraire de fois dans l'intervalle de temps $[0, p)$. Précisément à l'instant $p$, le délai expire et le capteur revient à son état initial pour faire une nouvelle lecture de la température. □

De plus, cet exemple montre comment les tâches périodiques peuvent être modélisées en ATP.

## 5.4 Exemples

Nous présentons ici deux exemples d'utilisation des opérateurs temporels.

**Exemple 5.1** Le premier exemple consiste à spécifier le système qui reconnaît les clicks simple et les clicks double d'une souris à un seul bouton que nous avons présenté dans l'exemple 2.1.

Ce système est modélisé en ATP de la façon suivante :

$$
\begin{aligned}
M_1 &\stackrel{\text{def}}{=} \mathsf{d}(p)\, M_2 \\
M_2 &\stackrel{\text{def}}{=} \mathsf{d}(r)\, M_3\ \mathsf{timeout}(t_{cs})\, \mathsf{d}(r)\, M_1 \\
M_3 &\stackrel{\text{def}}{=} \mathsf{d}(p)\, M_4\ \mathsf{timeout}(t_{cd})\, cs\, M_1 \\
M_4 &\stackrel{\text{def}}{=} \mathsf{d}(r)\, cd\, M_1\ \mathsf{timeout}(t_{cs})\, cs\, \mathsf{d}(r)\, M_1
\end{aligned}
$$

Initialement, le système attend que le bouton soit pressé pour la première fois (action $p$). Etant donné que la fréquence d'utilisation de la souris n'est pas connue, le temps d'attente est indéfini.

Lorsque le bouton est pressé, le processus lance un timeout pour surveiller si le bouton est relâché (action $r$) avant $t_{cs}$ unités de temps. Si cela ne se produit pas, le système attend indéfiniment l'action $r$ pour revenir en suite à l'état initial. Aucun click n'est reconnu dans ce cas.

Dans le cas contraire, et afin de pouvoir distinguer entre les clicks simples et doubles, le système lance un nouveau timeout qui surveille si le bouton est pressé encore une fois avant $t_{cd}$ unités de temps. Si ce n'est pas le cas alors le système reconnaît un click simple (action $cs$).

Lorsque le bouton est pressé pour une seconde fois (action $p$), un troisième timeout est créé pour contrôler si le bouton est relâché (action $r$) avant $t_{cs}$ unités de temps. Dans ce cas, le système reconnaît un click double (action $cd$). Dans le cas contraire, le système reconnaît le click simple antérieur et attend l'action $r$. □

**Exemple 5.2** Nous présentons ici un processus ATP qui décrit le comportement du contrôleur de température défini dans l'exemple 2.2.

Ce comportement est modélisé par le processus ATP suivant :

$$
\begin{aligned}
C_1 &\stackrel{\text{def}}{=} \mathsf{d}(r)\, C_2 \\
C_2 &\stackrel{\text{def}}{=} b_1\, C_3 \\
C_3 &\stackrel{\text{def}}{=} C_4\ \mathsf{watchdog}[r](t_{max})\, C_2 \\
C_4 &\stackrel{\text{def}}{=} \mathsf{d}(r)\, C_6\ \mathsf{timeout}(t_1)\, (b_2\, C_5 + r\, C_6) \\
C_5 &\stackrel{\text{def}}{=} \mathsf{d}(r)\, C_6\ \mathsf{timeout}(t_2)\, e\, C_1 \\
C_6 &\stackrel{\text{def}}{=} h\, \mathsf{idle}
\end{aligned}
$$

Initialement, le contrôleur attend le signal $r$ du capteur. Le début de l'opération de mouvement de la barre 1 est signalé par l'action $b_1$.

Le watchdog surveille si un nouveau signal $r$ arrive avant un temps $t_{max}$. Si ce n'est pas le cas, par exemple, parce que le capteur est tombé en panne, le contrôleur déclenche une nouvelle opération de réfrigération.

Le premier timeout contrôle le temps de mouvement de la barre 1. Ce timeout est lancé au même instant que le watchdog. Si un signal $r$ arrive avant que le mouvement de la barre soit terminé, le contrôleur signale l'erreur au moyen de l'action $h$ et s'arrête.

L'expiration du timeout correspond à la terminaison de l'opération avec la barre 1. Le mouvement de la barre 2 commence immédiatement. Un second timeout contrôle la durée du mouvement de la barre 2. Comme précédemment, l'arrivée d'un signal $r$ avant que l'opération soit achevée provoque une erreur. La terminaison de l'opération est signalée au moyen de l'action $e$. □

# Chapitre 6

# Représentation de termes d'ATP par des graphes temporisés finis

Dans ce chapitre nous présentons une méthode guidée par la syntaxe de construction de graphes temporisés finis pour les termes d'ATP [NSY91, NSY92, NSY93]. Afin de pouvoir construire le graphe temporisé d'un terme à partir des graphes de ses sous-termes il est nécessaire d'étendre le modèle présenté dans la section 2.2 avec des informations utiles pour la composition des sous-graphes déjà construits. Pour cela nous étendons les graphes temporisés de la façon suivante :

- Puisque le graphe est construit par récurrence sur la structure du terme, il est nécessaire de savoir traiter les sous-termes ouverts et de pouvoir détecter les occurrences libres des variables. Pour ceci, certains sommets sont étiquetés par des variables : l'étiquette $X$ sur un sommet indique que le sommet correspond à une occurrence libre de la variable $X$. Cette information est utilisée lors de la construction du graphe pour l'opérateur de récursion.

- Les contraintes temporelles déterminées par les opérateurs temporels d'ATP sont modélisées au moyen d'horloges et de conditions portant sur celles-ci. Afin d'exprimer correctement ces contraintes il est nécessaire de savoir quand chaque horloge doit être mise à 0. Pour ceci, chaque sommet est étiqueté avec un ensemble d'horloges telles que tout arc arrivant vers le sommet initialise un sous-ensemble d'entre elles. Cet ensemble sert aussi à déterminer quelles sont les horloges "significatives" pour un sommet, c'est-à-dire, les horloges qui interviennent dans la condition d'activité et dans les conditions sur les arcs issus du sommet.

- Finalement, des arcs spéciaux étiquetés par l'action $\varepsilon$ sont introduits. L'utilisation des $\varepsilon$-transitions est usuelle dans les méthodes de constructions compositionnelles [ASU86, Gar89]. Dans notre cas, l'introduction d'arcs étiquetés par $\varepsilon$ permet de construire le graphe temporisé d'un processus de la forme $P_1$ watchdog$[A](d)$ $P_2$ d'une façon simple à partir des graphes temporisés de $P_1$ et $P_2$.

L'organisation de ce chapitre est la suivante. Dans la section 6.1 nous présentons un exemple introductif afin d'expliquer les idées essentielles de la méthode de construction. Dans la section 6.2 nous définissons formellement les graphes temporisés étendus. Ceux-ci sont des graphes temporisés avec les extensions décrites ci-dessus. La méthode de construction n'est pas définie pour tous les termes. En effet, certains termes d'ATP ne sont pas représentables par des graphes finis. Ces termes sont identifiés dans la section 6.3. La section 6.4 est consacrée à la présentation de la méthode de construction. Cette méthode est consistante au sens où le modèle du terme et

le modèle du graphe correspondant satisfont le même ensemble de formules de TCTL. La preuve de la consistance est montrée dans l'annexe A.

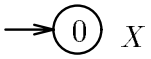
## 6.1 Exemple introductif

Avant de présenter formellement les règles de construction, nous illustrons le principe général de la méthode avec un exemple simple. Soit $P$ le terme $\mathsf{rec}X \cdot P'$ où $P'$ est le terme

$$\mathsf{d}(a)\,\mathsf{idle}\;\mathsf{watchdog}[](d)\,b\,X.$$

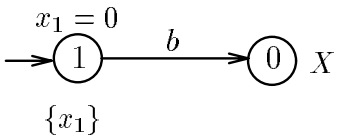
La construction progressive du graphe de $P$ en composant les graphes correspondants aux sous-termes de $P$ est la suivante :

1. Pour $X$, le graphe construit est :



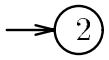

   L'étiquette $X$ indique que le sommet représente une occurrence libre de la variable $X$.

2. Pour $b\,X$, le graphe construit est :



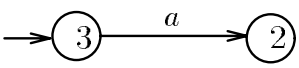

   Ce graphe est obtenu de la façon suivante. Un nouveau sommet et une nouvelle horloge sont créés. Le sommet 1 est le sommet initial du graphe. L'arc de 1 vers 0 étiqueté par $b$ modélise l'exécution de l'action $b$. La condition sur l'arc est *true*. De même, l'ensemble des horloges à mettre à 0 lorsque l'arc est franchi est vide puisque aucune horloge n'est associée au sommet 0. L'horloge $x_1$ permet de définir la contrainte temporelle qui conditionne l'exécution de l'action $b$. Il suffit d'imposer au sommet comme condition d'activité la contrainte $x_1 = 0$. En effet, celle-ci empêche le temps de progresser dans le sommet et oblige à franchir l'arc $b$ immédiatement. La condition sur l'arc peut être également $x_1 = 0$, mais ceci est redondant avec la condition d'activité. L'étiquette $\{x_1\}$ indique que $x_1$ est une horloge attachée au sommet 1.

3. Pour idle, le graphe construit est :



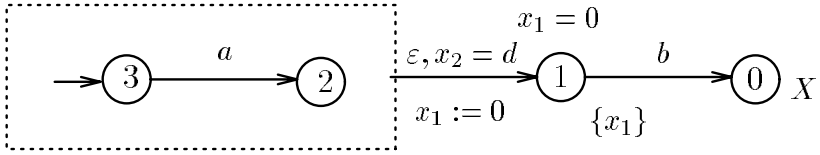

   Le graphe de idle a un seul sommet dont la condition d'activité est *true*.

4. Pour $\mathsf{d}(a)\,\mathsf{idle}$, le graphe construit est :



Un nouveau sommet est créé dont la condition d'activité est *true*. Ce sommet est le sommet initial du graphe. L'arc de 3 vers 2 étiqueté par $a$ représente l'exécution de l'action $a$. La condition sur l'arc est *true*. L'ensemble des horloges à mettre à 0 est vide puisque aucune horloge n'est associée au sommet 2. Comme l'exécution de $a$ peut être indéfiniment retardée, la condition d'activité du sommet est *true*.

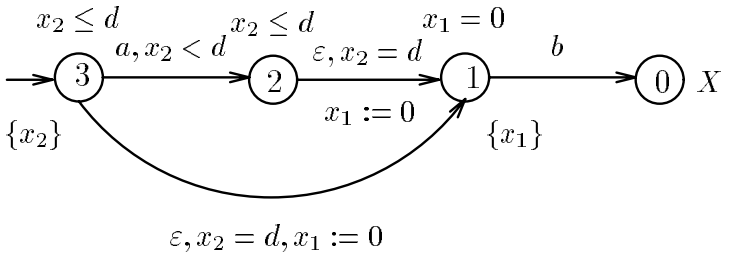
5. Pour $\mathsf{d}(a)\,\mathsf{idle}\,\mathsf{watchdog}[](d)\,b\,X$, l'effet de l'opérateur watchdog est représenté de manière schématique par la figure suivante :




Le graphe correspondant au sous-terme $\mathsf{d}(a)\,\mathsf{idle}$ qui est sous la portée du watchdog est entouré par une boîte. Ceci représente de manière graphique que le comportement défini par ce sous-graphe est conditionné par le watchdog.

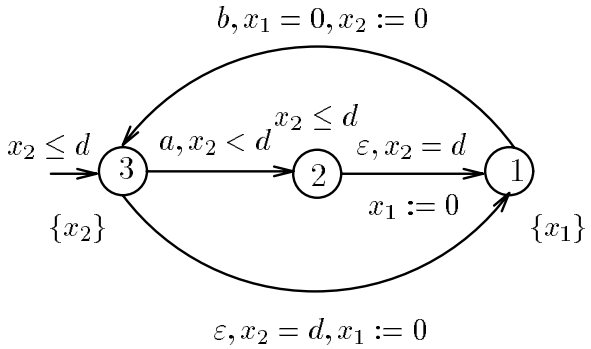
Une nouvelle horloge $x_2$ est introduite afin de pouvoir exprimer les contraintes temporelles imposées par le watchdog. Intuitivement, l'horloge $x_2$ "mémorise" le temps qui s'écoule dans la boîte. A tout instant, la différence entre $d$ et la valeur $x_2$ est le temps qui peut encore s'écouler avant l'expiration du watchdog. Lorsque $x_2$ atteint la valeur $d$ la boîte *doit être quittée* et *le contrôle doit être transféré* au sommet initial du graphe du processus $b\,X$. Ceci est représenté graphiquement par un arc sortant de la boîte dont l'étiquette est $\varepsilon$, la condition est $x_2 = d$ et les horloges à mettre à 0 sont celles associées au sommet initial du graphe de $b\,X$.

Ce comportement schématique se traduit par le graphe temporisé suivant :

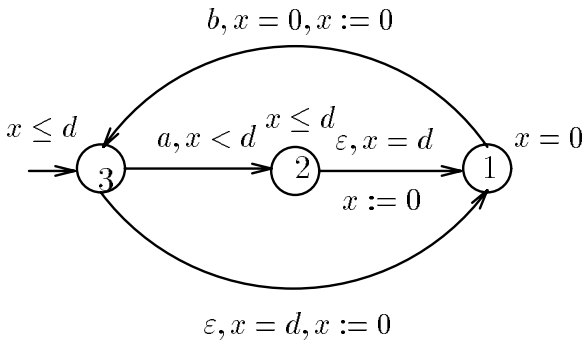



La condition $x_2 < d$ est ajoutée sur l'arc étiqueté par $a$ puisque l'exécution de l'action $a$ n'est possible que lorsque le délai $d$ n'a pas encore expiré. Autrement dit, un arc dans la boîte n'est franchissable que sous la condition $x_2 < d$. A chaque sommet de la boîte est associé la condition d'activité $x_2 \leq d$ et de chacun de ces sommets part un arc étiqueté par $\varepsilon$ avec la condition $x_2 = d$ vers le sommet 1. L'horloge $x_2$ est attachée à tous les sommets de la boîte.

6. Finalement, le graphe de $\mathsf{rec}X \cdot P'$ est obtenu en identifiant le sommet 0 étiqueté avec la variable $X$ et le sommet 3, qui est le sommet initial du graphe de $P'$.




Le graphe construit pour $P$ a deux horloges. Pourtant, le même comportement peut être décrit avec une seule horloge comme illustré par la figure suivante :




Cela est dû au fait que notre méthode de construction introduit les horloges de manière systématique : une pour l'action immédiate $b$ et une pour l'opérateur watchdog. Ce genre de problème est caractéristique de toute méthode de construction compositionnelle. Nous laissons pour l'instant de côté le problème de la réduction du nombre d'horloges, qui sera traité dans le chapitre 7.

## 6.2 Graphes temporisés étendus

Nous définissons dans cette section les graphes temporisés étendus. Soit $Act_\varepsilon = Act \cup \{\varepsilon\}$. Un élément de $Act_\varepsilon$ est noté $\alpha$.

**Définition 6.1** Un *graphe temporisé étendu* est un tuple $\langle S, H, E, s_I, \delta, F\rangle$ où $\langle S, H, E, s_I, \delta\rangle$ est un graphe temporisé dont les étiquettes sur les arcs appartiennent à $Act_\varepsilon$, et $F \subseteq S \times (Var \cup H)$ est un ensemble d'*extensions*. Pour tout $s \in S$, on définit :

$$F(s) = \{X \in Var \mid (s, X) \in F\} \cup \{x \in H \mid (s, x) \in F\}$$

La fonction $\delta$ est restreint aux sommets $s \in S$ tels que $F(s) \cap Var \neq \emptyset$. □

Une extension $(s, X)$ indique que le sommet $s$ représente la variable libre $X$. Elle est utilisée pour la construction de $\mathsf{rec}X\cdot$ comme nous l'avons montré dans l'exemple introductif. Notons que $F(s) \cap Var = \emptyset$ indique que le sommet $s$ ne représente pas une variable libre. Une extension $(s, x)$ indique que l'horloge $x \in H$ est attachée au sommet $s$. Ceci veut dire que l'horloge $x$ intervient dans la condition d'activité du sommet et sur les arcs issus. Cette extension est aussi utilisée pour déterminer les horloges qui doivent être initialisées par les arcs.

**Définition 6.2** Soit $G = \langle S, H, E, s_I, \delta, F\rangle$ un graphe temporisé étendu. Le modèle de $G$, noté $\mathcal{T}[\![G]\!]$, est le système de transitions étiquetées $\langle \mathcal{Q}, \rightarrow, q_I\rangle$ où

1. $\mathcal{Q} = \{(s,v) \mid \delta_s(v)\} \cup \{(s,v) \mid F(s) \cap Var \neq \emptyset\}$,
2. $q_I = (s_I, v_I)$,
3. la relation de transition $\rightarrow$ est la plus petite relation définie par les règles suivantes :

$$\frac{\langle s, \alpha, \psi, H', s'\rangle \in E \ \wedge \ \psi(v) \ \wedge \ F(s) \cap Var = \emptyset}{(s,v) \xrightarrow{\alpha} (s', v[H' := 0])} \tag{6.1}$$

$$\frac{\delta_s(v+t) \ \wedge \ F(s) \cap Var = \emptyset}{(s,v) \xrightarrow{t} (s, v+t)} \tag{6.2}$$

□

**Remarque 6.1** 1. Un sommet $s \in S$ tel que $F(s) \cap Var \neq \emptyset$ n'a aucune transition car il représente un terme qui contient des occurrences libres de variables.

2. Aucune différence n'est faite entre les arcs étiquetés par des actions dans $Act$ et les arcs dont l'étiquette est $\varepsilon$.

□

Nous généralisons ici la notion de sous-graphe accessible pour les graphes temporisés étendus.

**Définition 6.3** Soit $G = \langle S, H, E, s_I, \delta, F\rangle$ un graphe temporisé étendu. On définit :

$$Reach(G) = \langle \bar{S}, H, \bar{E}, s_I, \bar{\delta}, \bar{F}\rangle$$

où $\bar{S}$ est le plus petit ensemble tel que :

1. $s_I \in \bar{S}$ et
2. si $s \in \bar{S}$ et $\langle s, \ell, \psi, H', s'\rangle \in E$ alors $s' \in \bar{S}$.

et

$$\begin{array}{rcl} \bar{E} & = & E \cap (\bar{S} \times Act_\varepsilon \times \Psi(H) \times \mathbf{2}^H \times \bar{S}) \\ \bar{\delta} & = & \delta \cap (\bar{S} \times \Psi) \\ \bar{F} & = & F \cap (\bar{S} \times (Var \cup H)) \end{array}$$

$Reach(G)$ est donc le sous-graphe des sommets de $G$ accessibles à partir du sommet initial $s_I$. □

**Remarque 6.2** Pour tout graphe temporisé étendu $G$, $\mathcal{T}[\![G]\!] \sim \mathcal{T}[\![Reach(G)]\!]$. □

## 6.3 La récursion interdite

Le but de la méthode de construction que nous présentons dans cette section est de produire, à partir d'un terme d'ATP, le graphe temporisé correspondant. Pour ceci, il faut que le comportement décrit par le terme soit représentable par un graphe fini.

Il est bien connu que dans les algèbres de processus les termes qui comportent des instanciations récursives de variables de processus à travers l'opérateur de composition parallèle, sont susceptibles de produire des comportements non réguliers. C'est pourquoi les méthodes et les outils développés pour la génération des modèles finis pour les termes des algèbres de processus [Gar89] interdisent ce type de récursion ainsi que la récursion non gardée, même si dans certains cas ces termes peuvent être représentés par des modèles finis.

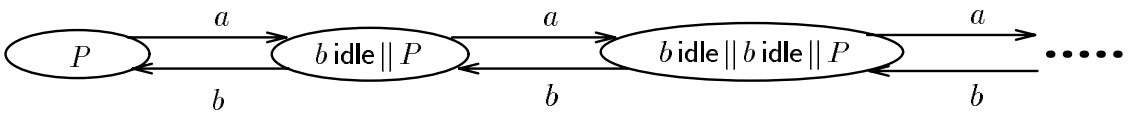
Dans le cas d'ATP, interdire ces termes n'est toutefois pas suffisant car l'opérateur watchdog introduit d'autres problèmes. Les restrictions qui sont imposées sur les termes afin que l'application de la méthode de construction soit toujours possible, sont énoncées et justifiées ci-dessous.

Une récursion est interdite dans un terme $P$ si :

- Elle comporte une instanciation récursive à travers l'opérateur de composition parallèle $||$ ou à travers l'opérateur de restriction **restrict in** . Par exemple, soit $P$ le terme :

  $$\mathsf{rec}X \cdot a(b\,\mathsf{idle}\,||\,X)$$

  $P$ contient une récursion interdite. Dans ce cas, le système de transition du terme est le suivant :

  Il ne peut pas être représenté par un graphe fini.

- Elle comporte une instanciation récursive à travers l'opérateur watchdog qui n'est gardée par aucune action dont l'exécution annule l'effet du watchdog. Autrement dit, si dans un terme de la forme

  $$P\ \mathsf{watchdog}[A](d)\ P'$$

  il y a une occurrence libre d'une variable dans le terme $P$ qui n'est pas gardée par une action $a \in A$. Par exemple, soit $P$ le terme :

  $$\mathsf{rec}X \cdot \mathsf{d}(a)X\ \mathsf{watchdog}[](d)\ \mathsf{idle}$$

  $P$ contient une récursion interdite. D'après les règles sémantiques, ce dernier terme a une transition pour $t < d$ vers le terme :

  $$\mathsf{d}(a)P\ \mathsf{watchdog}[](d-t)\ \mathsf{idle}$$

  De plus, ce terme a une transition par l'action $a$ vers le terme :

  $$\mathsf{rec}X \cdot \mathsf{d}(a)X\ \mathsf{watchdog}[](d)\ \mathsf{idle}\ \mathsf{watchdog}[](d-t)\ \mathsf{idle}$$

qui contient une nouvelle instance de l'opérateur watchdog avec une valeur différente de délai. En effet, en appliquant les règles sémantiques il se produit une création infinie d'instances de l'opérateur watchdog *avec des valeurs différentes de délais qui sont toutes actives au même temps*. Intuitivement, ceci correspond à créer un nombre infini d'horloges, une pour chacune de ces instances.

- Elle comporte une instanciation récursive à travers l'opérateur de choix non déterministe qui n'est pas gardée par une action. Par exemple, soit $P$ le terme :

$$\mathsf{rec}X \cdot \mathsf{wait}(1)\ X + \mathsf{wait}(3)\ X$$

$P$ contient une récursion interdite. D'après la sémantique d'ATP, $P$ a une transition temporelle de durée 1 vers le terme :

$$\mathsf{wait}(1)\ P + \mathsf{wait}(3)\ P + \mathsf{wait}(2)\ P$$

qui contient une nouvelle instance d'un opérateur watchdog. Ce problème est similaire au cas précédent au sens où ils comportent une création dynamique d'opérateurs watchdog avec des valeurs différentes des paramètres. Un autre exemple de ce type de récursion interdite est le terme $\mathsf{rec}X \cdot X + P$.

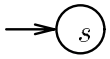
Pour chaque terme fermé d'ATP qui ne contient pas de récursion interdite, la méthode présentée ci-dessous construit un graphe temporisé fini qui décrit le même comportement. Certains termes qui comportent des récursions interdites peuvent néanmoins être représentés par des graphes temporisés finis. En général, ces comportements peuvent être décrits plus simplement sans utiliser de récursion interdite.

## 6.4 Construction guidée par la syntaxe

Nous présentons ici la méthode de construction d'un graphe temporisé étendu pour un terme d'ATP. Cette construction est définie seulement pour les termes qui ne comportent pas de récursion interdite. Nous exigeons de plus que les délais des opérateurs watchdog soient des entiers naturels positifs. La construction est guidée par la syntaxe et le résultat obtenu pour un terme $P$ est noté $\mathcal{E}[\![P]\!]$.

### 6.4.1 Le processus idle

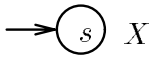
Le graphe temporisé étendu construit pour le processus **idle** a un seul sommet dont aucun arc n'est issu et dont la condition d'activité est *true*.




**Définition 6.4** Pour le processus **idle**, $\mathcal{E}[\![\mathsf{idle}]\!] = \langle \{s\}, \emptyset, \emptyset, s, \{(s, true)\}, \emptyset \rangle$. □

### 6.4.2 Les variables de processus

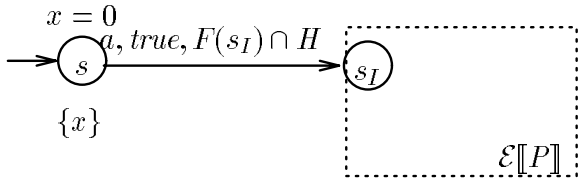
Le graphe temporisé étendu construit pour la variable $X \in Var$ a un seul sommet étiqueté avec $X$, ce qui indique que le sommet représente une occurrence de la variable libre $X$. Aucune condition d'activité n'est associée au sommet.



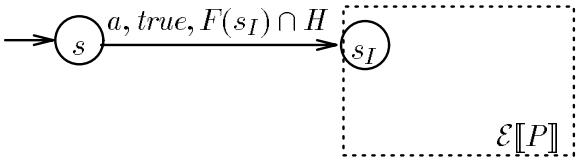

**Définition 6.5** Pour une variable $X \in Var$, $\mathcal{E}[\![X]\!] = \langle \{s\}, \emptyset, \emptyset, s, \emptyset, \{(s, X)\}\rangle$. □

### 6.4.3 Le préfixage

Supposons que le graphe $\mathcal{E}[\![P]\!] = \langle S, H, E, s_I, \delta, F\rangle$ de $P$ soit déjà construit. Pour le processus $aP$, un nouveau sommet $s$ et une nouvelle horloge $x$ sont créés, ainsi qu'un arc de $s$ vers le sommet initial $s_I$ du graphe de $P$ dont l'étiquette est $a$, la condition est *true* et l'ensemble d'horloges à mettre à 0 est $F(s_I) \cap H$, i.e., les horloges attachées au sommet initial du graphe de $P$. La condition d'activité associée au sommet $s$ est $x = 0$, ce qui exprime le fait que $a$ est une action immédiate. Le sommet $s$ est le sommet initial du graphe de $aP$ et l'ensemble $F(s)$ d'horloges associées à $s$ est $\{x\}$.




La construction pour $\mathsf{d}(a)P$ est similaire. La seule différence est qu'il n'est pas nécessaire de créer une nouvelle horloge. En effet, puisque l'exécution de l'action $a$ est indéfiniment retardable, il suffit d'associer au sommet initial $s$ la condition d'activité *true*.




**Définition 6.6** Pour $P \in$ ATP, soit $\mathcal{E}[\![P]\!] = \langle S, H, E, s_I, \delta, F\rangle$. Soit $a \in Act$.

Pour le terme $aP$,

$$\mathcal{E}[\![aP]\!] = \langle S \cup \{s\}, H \cup \{x\}, E \cup \{e\}, s, \delta \cup \{(s, x = 0)\}, F \cup \{(s, x)\}\rangle$$

où $s \notin S$, $x \notin H$ et $e = \langle s, a, true, H', s_I\rangle$ avec $H' = \{x \in H \mid x \in F(s_I)\}$.

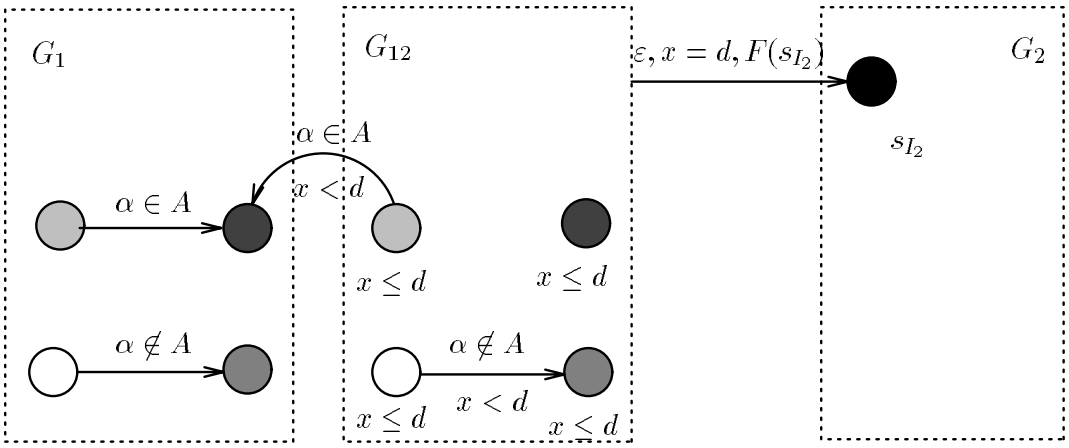
Pour le terme $\mathsf{d}(a)P$,

$$\mathcal{E}[\![\mathsf{d}(a)P]\!] = \langle S \cup \{s\}, H, E \cup \{e\}, s, \delta \cup \{(s, true)\}, F\}\rangle$$

où $s \notin S$, et $e = \langle s, a, true, H', s_I\rangle$ avec $H' = \{x \in H \mid x \in F(s_I)\}$. □

### 6.4.4 L'opérateur watchdog

Soit $P$ le terme $P_1$ $\mathsf{watchdog}[A](d)$ $P_2$. Supposons que les graphes $G_1 = \mathcal{E}[\![P_1]\!]$ et $G_2 = \mathcal{E}[\![P_2]\!]$ soient déjà construits. La construction de $\mathcal{E}[\![P]\!]$ est illustrée de manière schématique par la figure suivante :




Le graphe de $P$ est constitué des deux graphes $G_1$ et $G_2$ des termes $P_1$ et $P_2$ plus le graphe $G_{12}$ qui est construit de la façon suivante. Les sommets de $G_{12}$ sont des copies des sommets de $G_1$. Une nouvelle horloge est introduite afin d'exprimer la contrainte temporelle imposée par l'opérateur watchdog. A tout instant, la différence entre $d$ et la valeur $x$ est le temps qui peut encore s'écouler avant l'expiration du watchdog.

Les sommets de $G_{12}$ sont notés $(s_1, s_{I_2})$ où $s_1$ est un sommet de $G_1$ et $s_{I_2}$ est le sommet initial de $G_2$. Chaque sommet $(s_1, s_{I_2})$ de $G_{12}$ représente le sommet $s_1$ de $G_1$ sous la portée de l'opérateur watchdog.

Les arcs issus de $(s_1, s_{I_2})$ sont déterminés de la façon suivante :

1. Tout sommet $(s_1, s_{I_2})$ a un arc vers $s_{I_2}$, le sommet initial de $G_2$, dont l'étiquette est $\varepsilon$, la condition est $x = d$, et les horloges à mettre à 0 sont celles associées au sommet $s_{I_2}$ (cf. 6.3 dans la définition 6.7 ci-dessous). Ces arcs modélisent l'expiration du délai imposé par l'opérateur watchdog. Ceci se produit lorsque la valeur de $x$ est égale à $d$. Ces arcs sont représentés dans la figure comme un seul arc issus de la boîte qui entoure le graphe $G_{12}$.

2. Si $s_1$ a un arc vers $s_1'$ dont l'étiquette $\alpha$ n'appartient pas à l'ensemble d'actions qui annulent l'effet du watchdog, le sommet $(s_1, s_{I_2})$ a un arc vers $(s_1', s_{I_2})$. Ceci signifie que franchir l'arc n'amène pas hors de la portée du watchdog. Les cas $\alpha \in Act$ et $\alpha = \varepsilon$ sont traités de la même manière puisque d'après la sémantique d'ATP, l'expiration d'un watchdog plus interne n'a pas d'influence sur les plus externes (cf. 6.4 dans la définition 6.7 ci-dessous).

3. Le dernier cas est constitué par les arcs de $s_1$ vers un sommet $s_1'$, étiquetés par des actions appartenant à l'ensemble $A$ des actions qui annulent l'effet du watchdog. Dans ce cas, le sommet $(s_1, s_{I_2})$ a un arc vers $s_1'$ qui quitte le graphe $G_{12}$. Ceci modélise le fait que l'exécution d'une de ces actions amènent hors de la portée du watchdog (cf. 6.5 dans la définition 6.7 ci-dessous).

La contrainte $x < d$ est ajoutée sur tous les arcs de $G_{12}$ qui sont en correspondance avec les arcs de $G_1$. Ces arcs de $G_{12}$ ne sont donc franchissables que lorsque le délai imposé par le watchdog n'a pas encore expiré. Lorsque $x$ atteint la valeur $d$, plus aucun arc de $G_{12}$ ne peut être franchi, sauf les arcs étiquetés par $\varepsilon$ qui amènent vers le sommet initial de $G_2$. Pour tout sommet $(s_1, s_{I_2})$ de $G_{12}$ la condition d'activité est la conjonction de celle de $s_1$ et de $x \leq d$ et l'ensemble d'horloges attachées est l'ensemble d'horloges attachées à $s_1$ plus $x$.

**Définition 6.7** Soit $d \in \mathsf{N}$ et $d > 0$. Pour $P_1, P_2 \in$ ATP, soit $\mathcal{E}[\![P_i]\!] = \langle S_i, H_i, E_i, s_{I_i}, \delta_i, F_i \rangle$, $i \in \{1,2\}$, tels que $S_1 \cap S_2 = H_1 \cap H_2 = \emptyset$.

$$\mathcal{E}[\![P_1 \ \mathsf{watchdog}[A](d)\ P_2]\!] = \mathit{Reach}(\langle S, H, E, s_I, \delta, F \rangle)$$

où

$$\begin{aligned}
S &= S_1 \cup S_2 \cup (S_1 \times \{s_{I_2}\}) \\
H &= H_1 \cup H_2 \cup \{x\} \qquad x \notin H_1 \cup H_2 \\
E &= E_1 \cup E_2 \\
&\quad \cup \{\langle (s_1, s_{I_2}), \varepsilon, x = d, H', s_{I_2} \rangle \mid H' = \{x_2 \in H_2 \mid x_2 \in F_2(s_{I_2})\}\} && (6.3) \\
&\quad \cup \{\langle (s_1, s_{I_2}), \alpha, \psi \wedge x < d, H', (s_1', s_{I_2}) \rangle \mid \langle s_1, a, \psi, H', s_1' \rangle \in E_1 \wedge \alpha \notin A\} && (6.4) \\
&\quad \cup \{\langle (s_1, s_{I_2}), a, \psi \wedge x < d, H', s_1' \rangle \mid \langle s_1, a, \psi, H', s_1' \rangle \in E_1 \wedge a \in A\} && (6.5) \\
F &= F_1 \cup F_2 \cup \{((s_1, s_{I_2}), f) \mid f \in F_1(s_1) \cup \{x\}\}
\end{aligned}$$

et $s_I = (s_{I_1}, s_{I_2})$ et pour chaque $s \in S$,

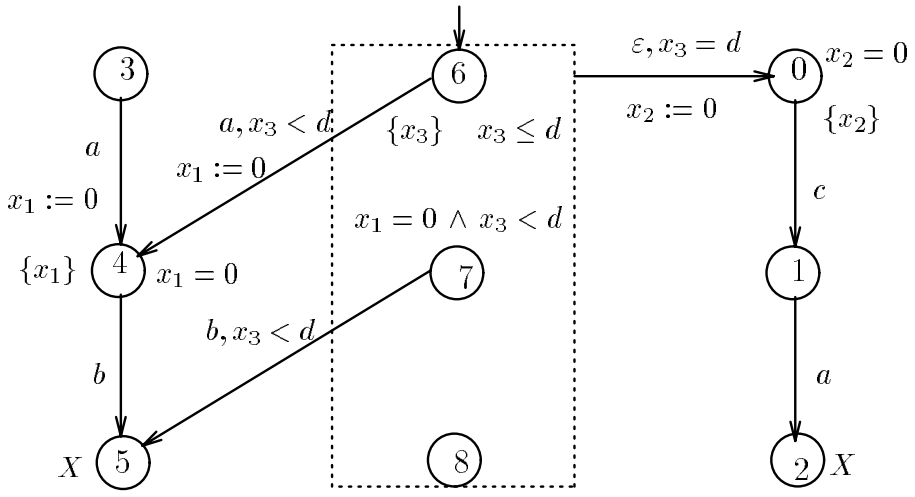
$$\delta_s = \begin{cases} \delta_i[s] & \text{si } s \in S_i, \text{ pour } i = 1,2, \\ \delta_1[s_1] \wedge x \leq d & \text{si } s = (s_1, s_{I_2}) \text{ et } F(s_1) \cap \mathit{Var} = \emptyset. \end{cases}$$

□

*Exemple.* La construction pour l'opérateur watchdog est illustrée avec l'exemple suivant. Soit $P$ le terme

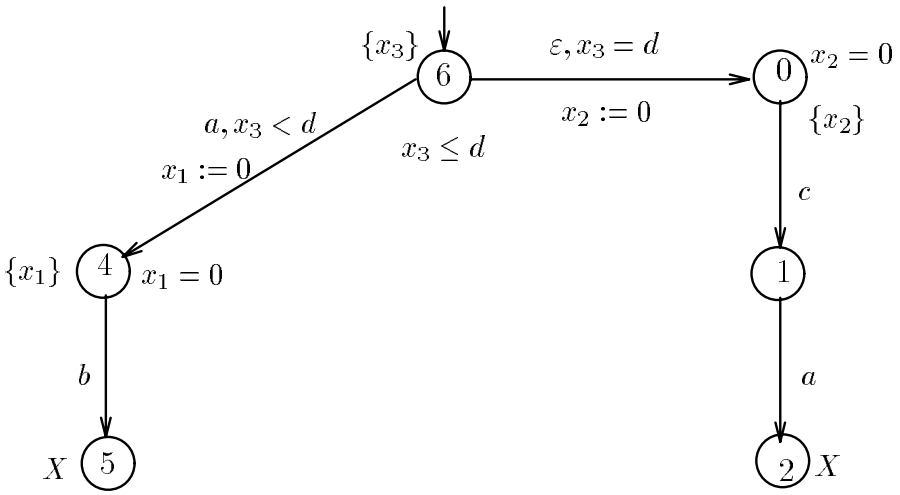
$$\mathsf{d}(a)\, b\, X\ \mathsf{watchdog}[a,b](d)\, c\, \mathsf{d}(a)\, X$$

La figure suivante montre le graphe temporisé étendu construit avant de calculer le sous-graphe accessible à partir du sommet initial :




Les sommets 6, 7 et 8 correspondent aux sommets 3, 4 et 5 du graphe de $\mathsf{d}(a)\, b\, X$. L'arc de 6 vers 4 et l'arc de 7 vers 5, correspondent aux arcs de 3 vers 4 et de 4 vers 5. Tous les arcs sortent de la boîte car toutes les actions annulent l'effet du watchdog. La condition $x_3 < d$ est ajoutée à tous les arcs. L'arc étiqueté par $\varepsilon$ vers 0 désigne de façon générique les arcs issus de 6, 7 et 8 et vers 0.

La figure suivante illustre le graphe final obtenu :




Les sommets 7 et 8 ne sont pas accessibles depuis le sommet initial 6, alors ils n'appartiennent pas au graphe final. □

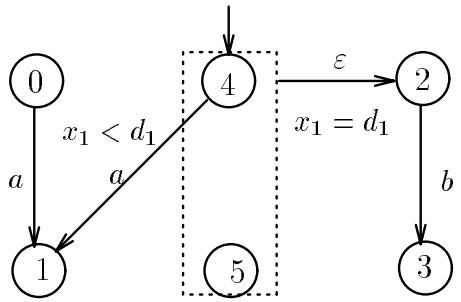
*Exemple.* L'exemple suivant illustre le fonctionnement de la méthode pour un terme qui contient deux opérateurs watchdog imbriqués. Soit $P$ le terme

$$(\mathsf{d}(a)\,\mathsf{idle}\ \mathsf{watchdog}[a](d_1)\ \mathsf{d}(b)\,\mathsf{idle})\ \mathsf{watchdog}[b](d_2)\ \mathsf{idle}$$

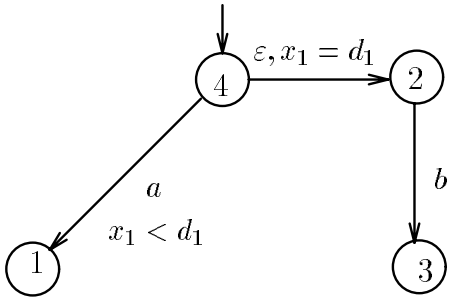
Le graphe construit pour le sous-terme

$$\mathsf{d}(a)\,\mathsf{idle}\ \mathsf{watchdog}[a](d_1)\ \mathsf{d}(b)\,\mathsf{idle}$$

à partir des graphes de $\mathsf{d}(a)\,\mathsf{idle}$ et $\mathsf{d}(b)\,\mathsf{idle}$ est le suivant :

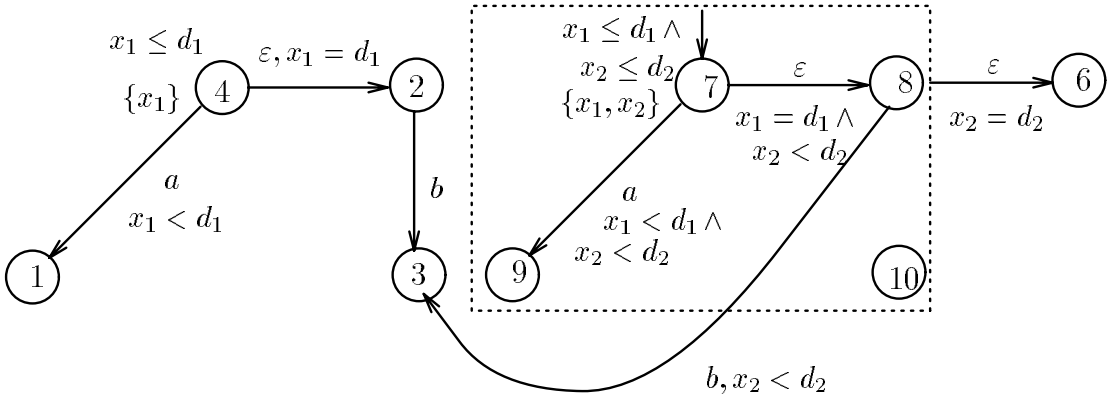



graphe transitoire

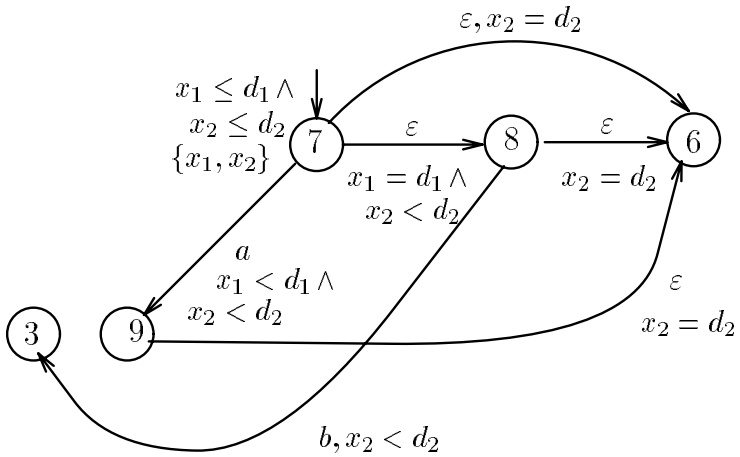



graphe final

La construction du graphe de $P$ est illustrée ci-dessous. Pour l'opérateur watchdog le plus externe la construction commence par faire une copie de ce graphe. Les sommets 7, 8, 9 et 10 correspondent aux sommets 4, 2, 1 et 3 respectivement. L'arc étiqueté par l'action $a$ ne sort pas de la boîte car cette action n'annule pas l'effet du watchdog le plus externe. Il en va de même pour l'arc étiqueté par $\varepsilon$ qui représente l'expiration du watchdog de durée $d_1$. Par contre, l'arc dont l'étiquette est $b$ quitte la boîte puisque l'action $b$ supprime le watchdog de durée $d_2$. L'arc dont l'étiquette est $\varepsilon$ et la condition est $x_2 = d_2$ représente l'ensemble d'arcs qui modélisent l'expiration du watchdog de durée $d_2$.

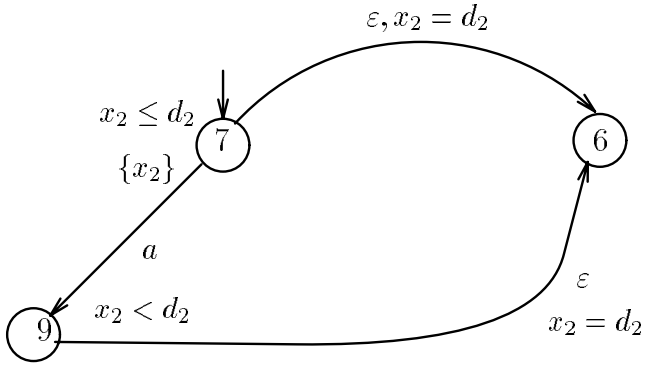



Finalement, le graphe temporisé obtenu pour $P$ est le suivant :




La méthode de construction ne tient pas compte des valeurs des paramètres des opérateurs watchdog. Ceci produit parfois des arcs qui ne seront jamais franchissables pour des valeurs particulières des paramètres.

Par exemple, si $d_1$ et $d_2$ sont tels que $d_2 < d_1$, l'arc de 7 vers 8 ne peut pas être franchi. Cet arc peut alors être éliminé et avec lui le sommet 8 et tous les arcs issus de 8. Le graphe obtenu est le suivant :

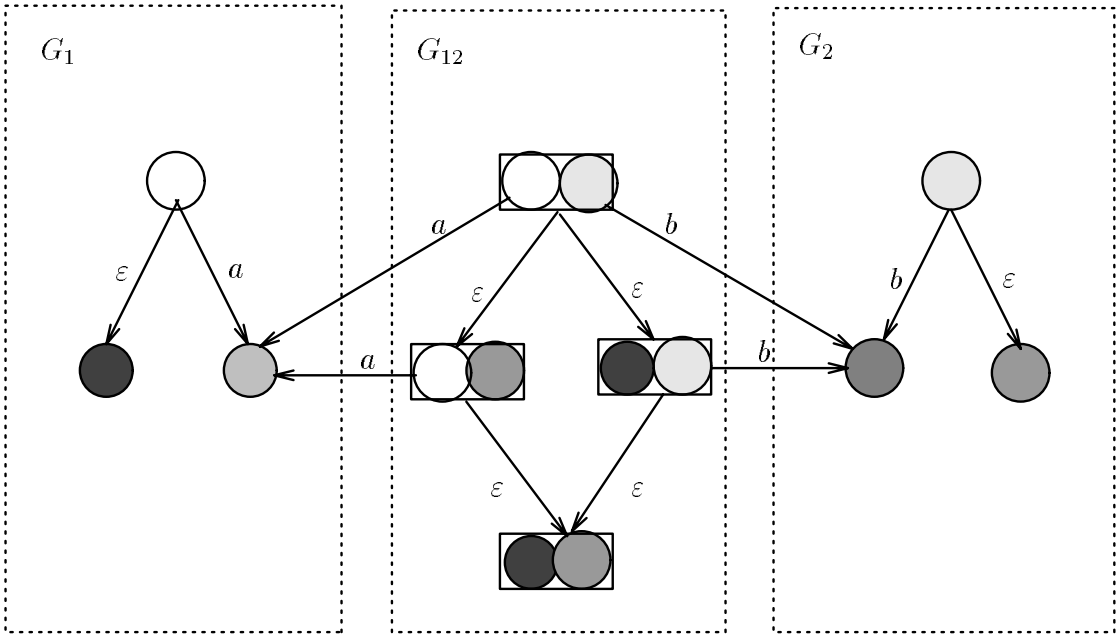



La condition $x_1 = d_1$ ne sera jamais vraie puisque $d_2 < d_1$ et puisqu'à tout instant la valeur de $x_1$ *est égale* à la valeur de $x_2$ car $x_1$ *et* $x_2$ *sont toutes deux initialisées par l'arc d'entrée au sommet.* □

Cependant, en général, il n'est pas possible de déterminer si une condition sera satisfaisable seulement à partir des valeurs des constantes. Il est nécessaire de connaître aussi les relations existantes entre toutes les horloges qui interviennent dans la condition. Celles-ci ne peuvent pas être en général calculées pendant la construction. Toutefois, quelques optimisations peuvent être faites à la fin, lorsque le graphe complet a été construit.

### 6.4.5 Le choix non déterministe

Soit $P$ le terme $P_1 + P_2$ et supposons que les graphes $G_1$ et $G_2$ pour les termes $P_1$ et $P_2$ sont déjà construits. La construction pour le choix non déterministe est illustrée de façon schématique par la figure suivante :




La méthode consiste à construire un graphe $G_{12}$ à partir des graphes $G_1$ et $G_2$ dont les sommets sont des paires $(s_1, s_2)$ où $s_i$ appartient au graphe $G_i$. Les arcs issus d'un sommet $(s_1, s_2)$ de $G_{12}$ sont construits comme suit :

1. Si $s_1$ a un arc vers $s_1'$ dont l'étiquette est une action $a \in Act$, alors $(s_1, s_2)$ a un arc étiqueté par $a$ vers $s_1'$. Ceci correspond à la règle sémantique d'ATP qui dit que l'exécution d'une action résout un choix non déterministe. Evidemment, il en va de même pour $s_2$ (cf. 6.7 et 6.8 dans la définition 6.8 ci-dessous).

2. Si $s_1$ a un arc vers $s_1'$ dont l'étiquette est $\varepsilon$, alors $(s_1, s_2)$ a un arc étiqueté par $\varepsilon$ vers $(s_1', s_2)$. Ceci est dû au fait qu'un arc dont l'étiquette est $\varepsilon$ représente l'expiration d'un opérateur watchdog; il correspond donc à une transition temporelle, et d'après la sémantique d'ATP, celles-ci ne résolvent pas un choix non déterministe (cf. 6.6 dans la définition 6.8 ci-dessous).

   En fait, d'après les règles sémantiques, l'expiration simultanée de plusieurs opérateurs watchdog est modélisée comme une *unique* transition temporelle. Ce comportement pourrait être représenté par la synchronisation des arcs $\varepsilon$ correspondants.

   C'est-à-dire, si $s_1$ et $s_2$ ont des arcs $\varepsilon$ vers $s_1'$ et $s_2'$, alors un arc $\varepsilon$ dont la condition est la conjonction des conditions de ces arcs, pourrait être ajouté de $(s_1, s_2)$ vers $(s_1', s_2')$.

Ceci devrait être fait pour *tous les sous-ensembles* d'arcs étiquetés par $\varepsilon$ afin de pouvoir couvrir *toutes les expirations simultanées possibles* des opérateurs watchdog.

Cette synchronisation *massive* des arcs étiquetés par $\varepsilon$ est donc très coûteuse, car elle produit une explosion combinatoire d'arcs $\varepsilon$, mais elle n'est toutefois pas nécessaire, car les règles de construction garantissent que les expirations simultanées sont correctement modélisées par l'interleaving des arcs $\varepsilon$ correspondants.

La condition d'activité attachée à un sommet $(s_1, s_2)$ est la conjonction des conditions d'activités de $s_1$ et $s_2$. De même, l'ensemble des horloges attachées à $(s_1, s_2)$ est l'union des ensembles correspondants à $s_1$ et $s_2$.

**Définition 6.8** Pour $P_1, P_2 \in$ ATP, soit $\mathcal{E}[\![P_i]\!] = \langle S_i, H_i, E_i, s_{I_i}, \delta_i, F_i \rangle$, $i \in \{1,2\}$, tels que $S_1 \cap S_2 = H_1 \cap H_2 = \emptyset$.

$$\mathcal{E}[\![P_1 + P_2]\!] = \mathit{Reach}(\langle S, H, E, s_I, \delta, F \rangle)$$

où

$$\begin{aligned}
S &= S_1 \cup S_2 \cup (S_1 \times S_2) \\
H &= H_1 \cup H_2 \\
E &= E_1 \cup E_2 \\
&\quad \cup \{\langle (s_1, s_2), a, \psi, H', s \rangle \mid \langle s_i, a, \psi, H', s \rangle \in E_i, i = 1,2\} && (6.6) \\
&\quad \cup \{\langle (s_1, s_2), \varepsilon, \psi, H', (s_1', s_2) \rangle \mid \langle s_1, \varepsilon, \psi, H', s_1' \rangle \in E_1\} && (6.7) \\
&\quad \cup \{\langle (s_1, s_2), \varepsilon, \psi, H', (s_1, s_2') \rangle \mid \langle s_2, \varepsilon, \psi, H', s_2' \rangle \in E_2\} && (6.8) \\
F &= F_1 \cup F_2 \cup \{((s_1, s_2), f) \mid f \in F_1(s_1) \cup F_2(s_2), s_i \in S_i, i = 1,2\}
\end{aligned}$$

et $s_I = (s_{I_1}, s_{I_2})$ et pour chaque $s \in S$,

$$\delta_s = \begin{cases} \delta_i[s] & \text{si } s \in S_i, \text{ pour } i = 1,2, \\ \delta_1[s_1] \wedge \delta_2[s_2] & \text{si } s = (s_1, s_2) \in S_1 \times S_2 \text{ et } F(s_i) \cap \mathit{Var} = \emptyset, \text{ pour } i = 1,2. \end{cases}$$

□

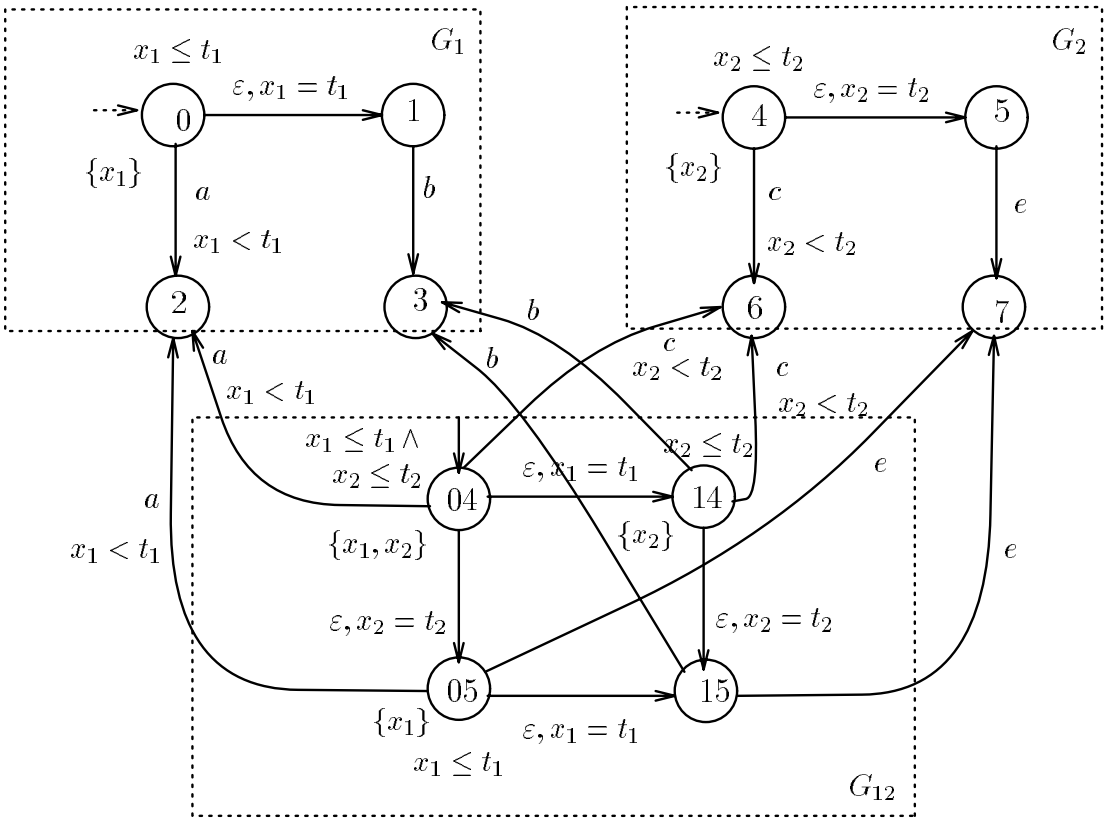
*Exemple.* La construction pour l'opérateur de choix non déterministe est illustrée par l'exemple suivant. Soit $P$ le terme $P_1 + P_2$ où $P_1$ est le terme :

$$\mathsf{d}(a)\,\mathsf{idle}\ \mathsf{watchdog}[a](t_1)\ \mathsf{d}(b)\,\mathsf{idle}$$

et $P_2$ est le terme :

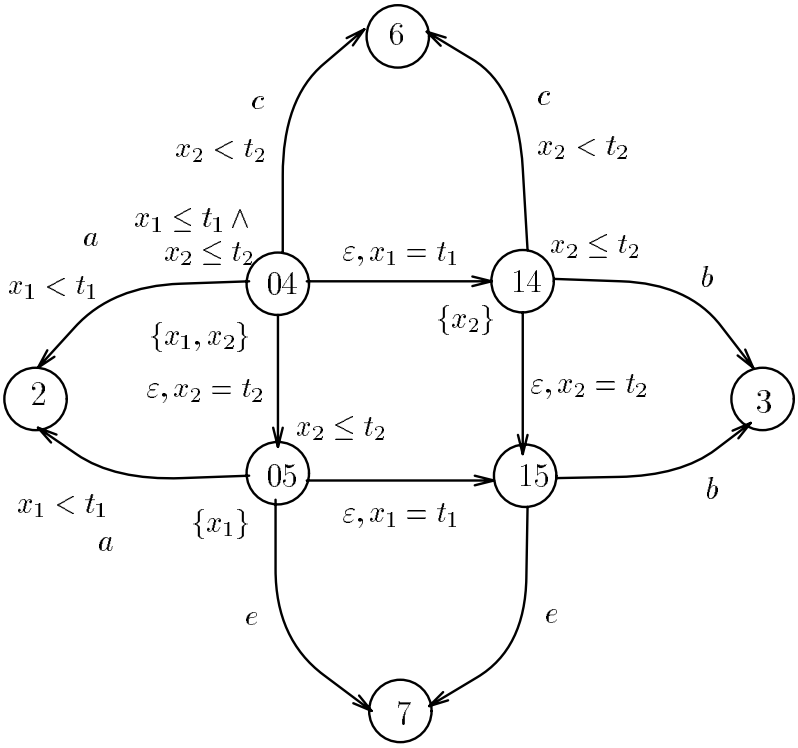
$$\mathsf{d}(c)\,\mathsf{idle}\ \mathsf{watchdog}[c](t_2)\ \mathsf{d}(e)\,\mathsf{idle}$$

Le graphe construit pour $P$ est le suivant :




Seuls les sommets accessibles de $G_{12}$ ont été dessinés. Le sommet initial est 04, composé des sommets initiaux de $G_1$ et $G_2$. Les horloges attachées à 04 sont $x_1$ et $x_2$, qui sont respectivement les horloges attachées à 0 et à 4. La condition d'activité est $x_1 \leq t_1 \wedge x_2 \wedge \leq t_2$. Puisque 0 a un arc vers 1 dont l'étiquette est $\varepsilon$, le sommet 04 a une copie de cet arc vers 14. Cet arc correspond à l'expiration du watchdog de durée $t_1$. De même, 04 a un arc vers 05 qui est une copie de l'arc étiqueté par $\varepsilon$ de 4 vers 5, correspondant à l'expiration du délai $t_2$. La construction est similaire pour les autres sommets de $G_{12}$. Tous les arcs étiquetés par une action issus d'un sommet de $G_{12}$ arrivent vers un sommet soit de $G_1$ soit de $G_2$. Seuls les arcs dont l'étiquette est $\varepsilon$ arrivent vers un sommet de $G_{12}$.

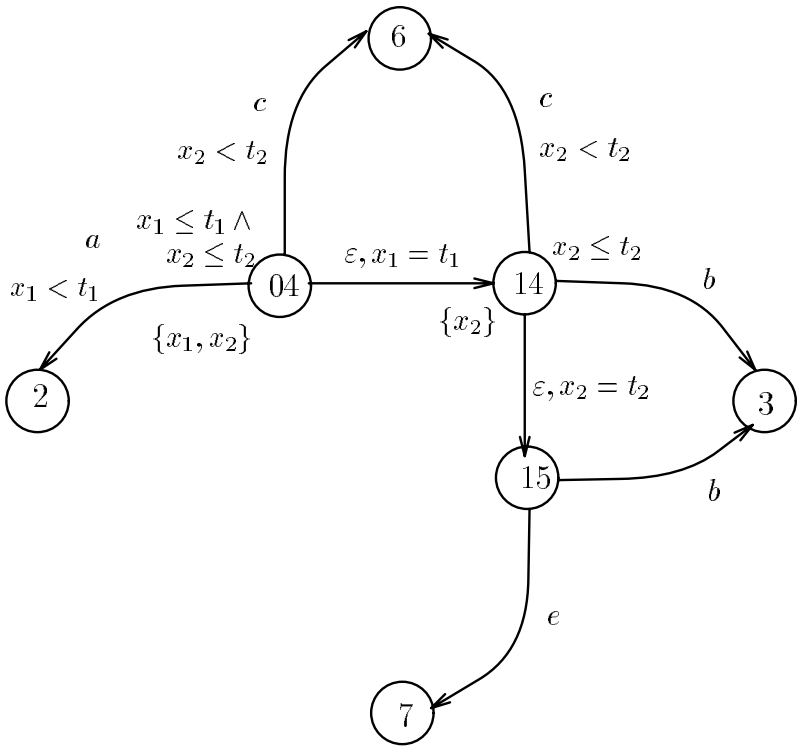
Le graphe accessible à partir du sommet initial est le suivant :




□

La construction pour le choix non déterministe ne tient pas compte des relations entre les différents délais introduits par les opérateurs watchdog. Cette méthode produit parfois des arcs qui ne seront jamais franchissables.

Dans l'exemple précédent, si les délais $t_1$ et $t_2$ des deux opérateurs watchdog sont tels que $t_1 < t_2$, l'arc de 04 vers 05 dont la condition est $x_2 = t_2$ ne peut pas être franchi car la condition n'est jamais satisfaite. Il est donc possible d'éliminer cet arc. Il en va de même pour le sommet 05 et tous les arcs qui en sont issus. Le graphe résultant est le suivant :




Comme il a été déjà remarqué, il ne suffit pas de connaître les valeurs des délais pour simplifier un arc. Une telle optimisation ne peut être faite que lorsque la relation *entre les différentes horloges* est connue et celle-ci ne peut pas être en général calculée statiquement. Dans cette exemple, une telle simplification est possible car les horloges $x_1$ et $x_2$ sont toutes deux initialisées par l'unique arc d'entrée.

### 6.4.6 L'opérateur de composition parallèle

Soit $P$ le terme $P_1 \,||\, P_2$. Supposons que les graphes $G_1 = \mathcal{E}[\![P_1]\!]$ et $G_2 = \mathcal{E}[\![P_2]\!]$ soient déjà construits. La construction pour $P$ consiste à faire le produit cartésien entre les graphes $G_1$ de $P_1$ et $G_2$ de $P_2$. Pour chaque sommet $(s_1, s_2)$ de ce graphe, les arcs issus sont calculés de la façon suivante :

1. Si $s_1$ a un arc vers $s_1'$ dont l'étiquette $\alpha$ est une action $a \in Act$ ou $\varepsilon$, alors $(s_1, s_2)$ a un arc étiqueté par $\alpha$ vers $(s_1', s_2)$. Evidemment, il en va de même pour $s_2$ (cf. 6.9 et 6.10 dans la définition 6.8 ci-dessous). Comme pour l'opérateur de choix non déterministe, aucune synchronisation des arcs étiquetés par $\varepsilon$ n'est calculée.

2. Si $s_1$ a un arc vers $s_1'$ dont l'étiquette est l'action $a_1$ et $s_2$ a un arc vers $s_2'$ dont l'étiquette est l'action $a_2$, et les actions $a_1$ et $a_2$ peuvent communiquer, i.e., $a_1 \mid a_2 \neq \perp$, alors $(s_1, s_2)$ a un arc étiqueté par $a_1 \mid a_2$ vers $(s_1', s_2')$ (cf. 6.11 dans la définition 6.8 ci-dessous).

La condition d'activité attachée à un sommet $(s_1, s_2)$ est la conjonction des conditions d'activité de $s_1$ et $s_2$. De même, l'ensemble des horloges attachées à $(s_1, s_2)$ est l'union des ensembles correspondants à $s_1$ et $s_2$.

**Définition 6.9** Pour $P_1, P_2 \in$ ATP, soit $\mathcal{E}[\![P_i]\!] = \langle S_i, H_i, E_i, s_{I_i}, \delta_i, F_i\rangle$, $i \in \{1,2\}$, tels que $S_1 \cap S_2 = H_1 \cap H_2 = \emptyset$.

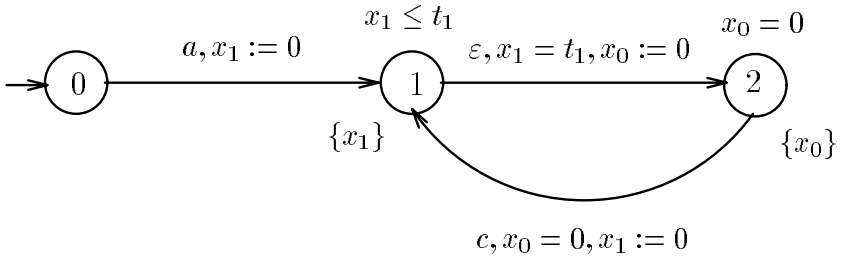
$$\mathcal{E}[\![P_1 \,||\, P_2]\!] = \langle S_1 \times S_2, H_1 \cup H_2, E, s_I, \delta, F\rangle$$

où

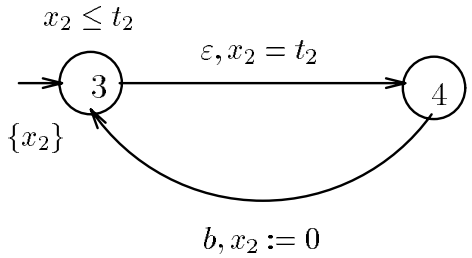
$$\begin{aligned}
E &= \{\langle (s_1,s_2), \alpha, \psi, H', (s_1', s_2)\rangle \mid \langle s_1, \alpha, \psi, H', s_1'\rangle \in E_1\} && (6.9)\\
&\cup \{\langle (s_1,s_2), \alpha, \psi, H', (s_1, s_2')\rangle \mid \langle s_2, \alpha, \psi, H', s_2'\rangle \in E_2\} && (6.10)\\
&\cup \{\langle (s_1,s_2), a_1 \mid a_2, \psi_1 \wedge \psi_2, H_1' \cup H_2', (s_1', s_2')\rangle \mid \\
&\qquad\qquad \langle s_i, a_i, \psi_i, H_i', s_i'\rangle \in E_i \wedge a_1 \mid a_2 \neq \bot\} && (6.11)\\
F &= \{((s_1,s_2), f) \mid f \in F(s_1) \cup F(s_2), s_i \in S_i, i = 1,2\}
\end{aligned}$$

et $s_I = (s_{I_1}, s_{I_2})$ et pour chaque $(s_1, s_2) \in S_1 \times S_2$, tel que $F(s_i) \cap \mathit{Var} = \emptyset$, $\delta[(s_1,s_2)] = \delta_1[s_1] \wedge \delta_2[s_2]$. □

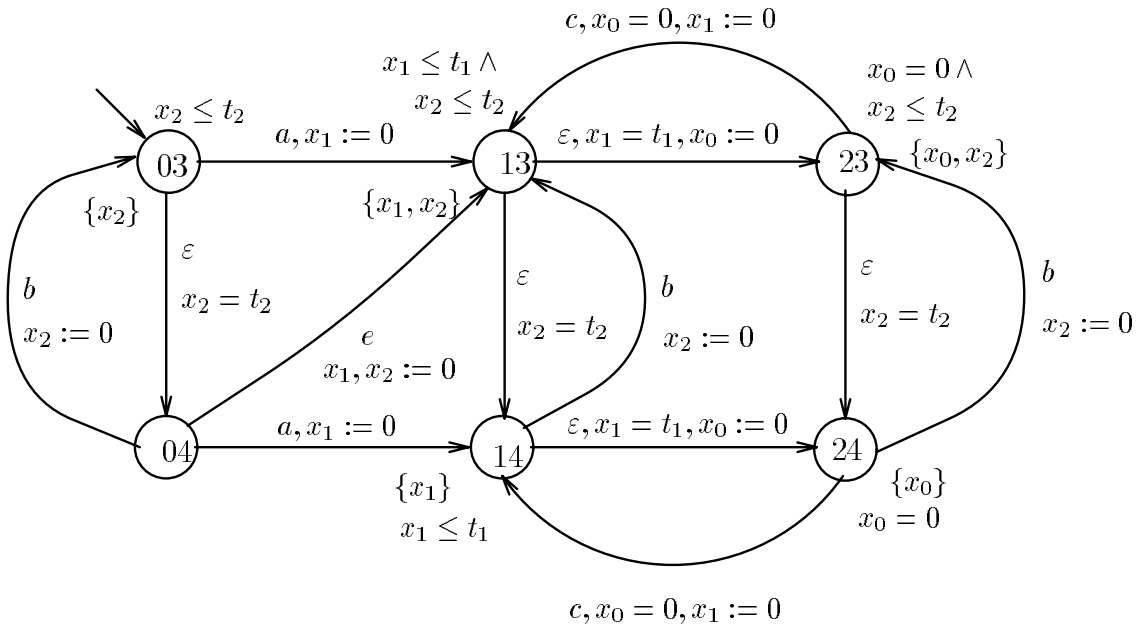
*Exemple.* La construction pour l'opérateur de composition parallèle est illustrée par l'exemple suivant. Soit $P$ le terme $P_1 \,||\, P_2$ où le graphe de $P_1$ est :




et le graphe de $P_2$ est :




En supposant que $a \mid b = e$ et $c \mid b = \bot$, le graphe construit pour $P$ est le suivant :




Dans ce cas, aucune simplification ne peut être faite. Par exemple, chacun des quatre arcs vers le sommet 13 initialise un ensemble différent d'horloges. Cela signifie qu'au sommet 13 il est possible que $x_1$ soit égale, inférieure ou supérieure à $x_2$. Par conséquent, aucune analyse basée sur les valeurs des $t_1$ et $t_2$ n'est possible car les instants où les horloges sont initialisées ne coïncident pas nécessairement. □

### 6.4.7 L'opérateur de restriction

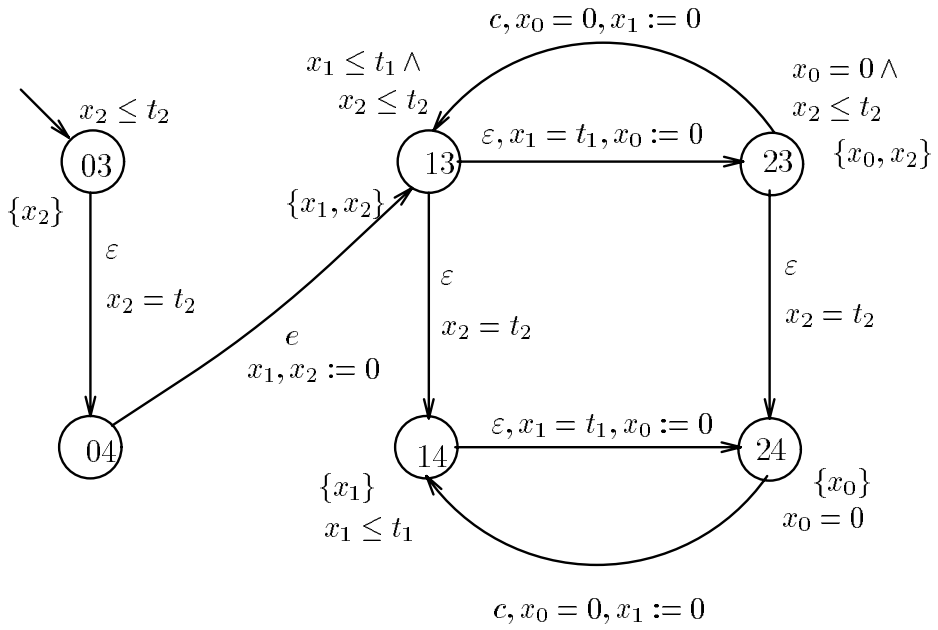
Supposons que le graphe temporisé étendu pour $P$ soit déjà construit. Le graphe de **restrict** $A$ **in** $P$ est obtenu à partir du graphe de $P$ en éliminant tous les arcs dont l'étiquette appartient à l'ensemble $A$.

**Définition 6.10** Soit $A \subseteq Act$. Pour $P \in \mathrm{ATP}$, soit $\mathcal{E}[\![P]\!] = \langle S, H, E, s_I, \delta, F \rangle$.

$$\mathcal{E}[\![\textbf{restrict } A \textbf{ in } P]\!] = \mathit{Reach}(\langle S, H, E', s_I, \delta, F \rangle)$$

où $E' = E - \{\langle s, a, \psi, H', s' \rangle \in E \mid a \in A\}$. □

*Exemple.* La construction pour l'opérateur de restriction est illustrée par l'exemple suivant. Soit $P$ le terme **restrict** $a, b$ **in** $P'$ où $P'$ est le terme dont le graphe est celui construit dans l'exemple précédent. Le graphe construit pour $P$ est le suivant :




Ce graphe est obtenu à partir du graphe de $P'$ en supprimant tous les arcs dont l'étiquette est $a$ ou $b$. □

### 6.4.8 L'opérateur de récursion

Supposons que le graphe pour le terme $P$ soit déjà construit. Le graphe pour le terme $\mathsf{rec}X \cdot P$ est obtenu à partir du graphe de $P$ en identifiant le sommet initial avec tous les sommets qui représentent des occurrences libres de la variable $X$, c'est-à-dire, avec les sommets étiquetés par $X$. Les restrictions syntaxiques imposées aux termes garantissent qu'un sommet dont l'étiquette est $X$ n'a aucune autre étiquette. Autrement dit, si $X \in F(s)$ alors $F(s) = \{X\}$.

Soit $S^-$ l'ensemble des sommets étiquetés par $X$. L'ensemble des sommets du graphe de $\mathsf{rec}X \cdot P$ est formé par tous les sommets du graphe de $P$ sauf les sommets de $S^-$. L'ensemble des arcs vers un sommet de $S^-$ est noté $E^-$. Tout arc de $E^-$ est remplacé par un arc vers le sommet initial $s_I$. Ce nouvel arc a la même étiquette et la même condition que l'ancien. Les horloges à mettre à 0 sont celles attachées au sommet initial. L'ensemble des arcs ainsi créés est noté $E^+$.

**Définition 6.11** Soit $X \in \mathit{Var}$. Pour $P \in \mathrm{ATP}$, soit $\mathcal{E}[\![P]\!] = \langle S, H, E, s_I, \delta, F\rangle$ On définit :

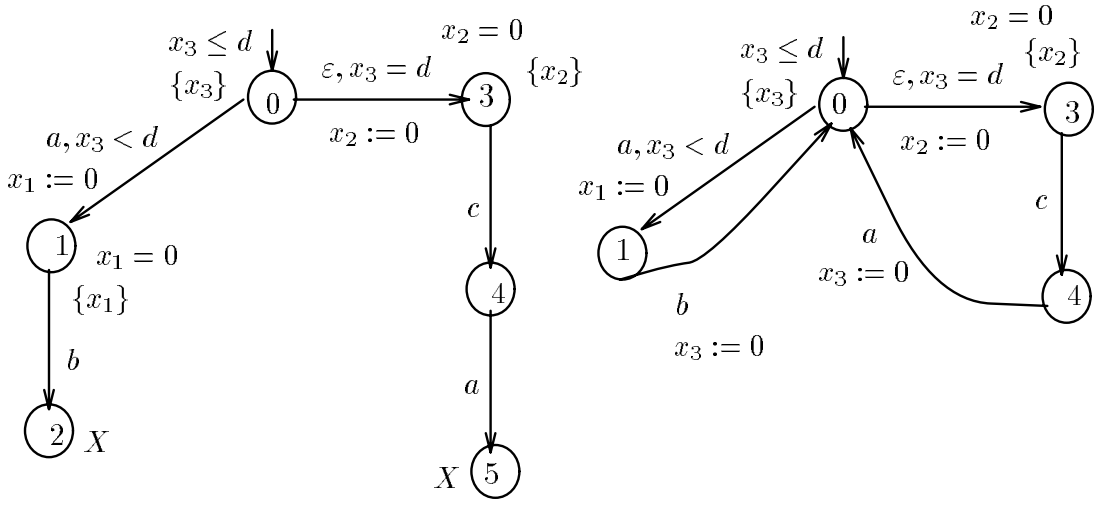
$$\mathcal{E}[\![\mathsf{rec}X \cdot P]\!] = \langle S - S^-, H, E - E^- \cup E^+, s_I, \delta, F - (S \times \{X\})\rangle$$

où

$$S^- = \{s \in S \mid F(s) = \{X\}\} \tag{6.12}$$
$$E^- = \{\langle s, a, \psi, H', s'\rangle \in E \mid s' \in S^-\} \tag{6.13}$$
$$E^+ = \{\langle s, a, \psi, F(s_I), s_I\rangle \mid \langle s, a, \psi, H', s'\rangle \in E^-\} \tag{6.14}$$

□

*Exemple.* La figure suivante illustre la construction pour l'opérateur de récursion.




Pour cet exemple, on a :

$$\begin{aligned} S^- &= \{2, 5\} \\ E^- &= \{\langle 1, b, \mathit{true}, \emptyset, 2\rangle, \langle 4, a, \mathit{true}, \emptyset, 5\rangle\} \\ E^+ &= \{\langle 1, b, \mathit{true}, \{x_3\}, 0\rangle, \langle 4, a, \mathit{true}, \{x_3\}, 0\rangle\} \end{aligned}$$

□

Pour tout terme fermé $P \in \mathrm{ATP}$ qui ne contient pas de récursions interdites, le graphe temporisé de $P$ est obtenu à partir du graphe temporisé étendu construit selon les règles présentées ci-dessus de la façon suivante.

**Définition 6.12** Pour tout terme fermé $P \in \mathrm{ATP}$ tel que $\mathcal{E}[\![P]\!] = \langle S, H, E, s_I, \delta, F\rangle$ est défini, le graphe temporisé associé à $P$, noté $\mathcal{G}[\![P]\!]$, est $\langle S, H, E, s_I, \delta\rangle$. □

## 6.5 Exemples

Nous illustrons la méthode de construction avec le terme d'ATP qui définit le comportement de la souris avec un seul bouton. Dans l'exemple 5.1 nous avons présenté une spécification en termes d'équations mutuellement récursives au lieu d'utiliser l'opérateur de récursion.

En utilisant cet opérateur, le terme $M$ qui définit le comportement de la souris est $\mathsf{rec}X \cdot M_1$ où : $M_1$ est le terme :

$$\mathsf{d}(p)\, M_2$$

$M_2$ est le terme :

$$\mathsf{d}(r)\, M_3\ \mathsf{timeout}(t_{sc})\ \mathsf{d}(r)\, X$$

$M_3$ est le terme :

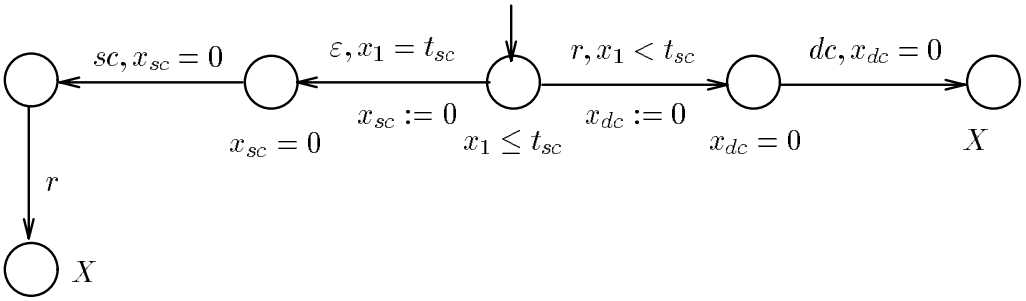
$$\mathsf{d}(p)\, M_4\ \mathsf{timeout}(t_{dc})\ sc\, X$$

et $M_4$ est le terme :

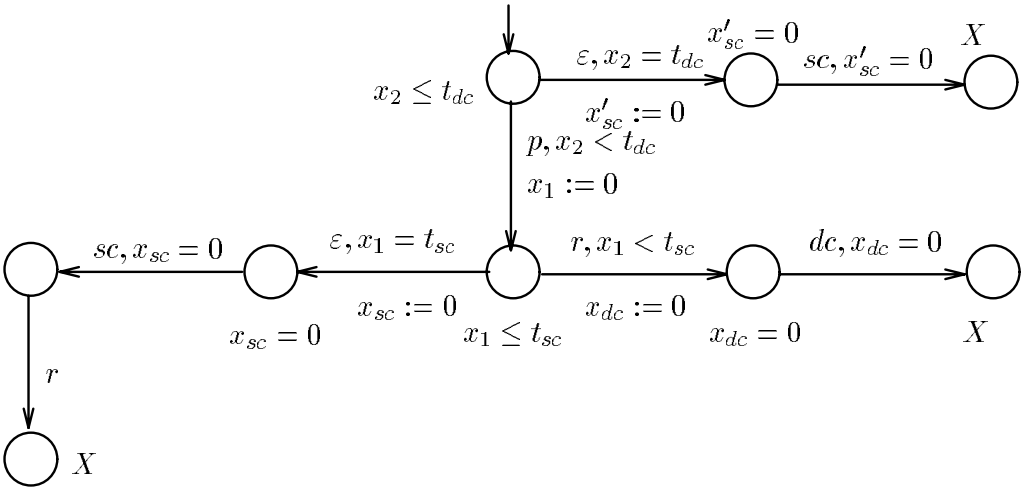
$$\mathsf{d}(r)\, dc\, X\ \mathsf{timeout}(t_{sc})\ sc\, \mathsf{d}(r)\, X$$

Nous présentons la construction pas à pas du graphe de $M$ à partir des graphes de $M_1$, $M_2$, $M_3$ et $M_4$.

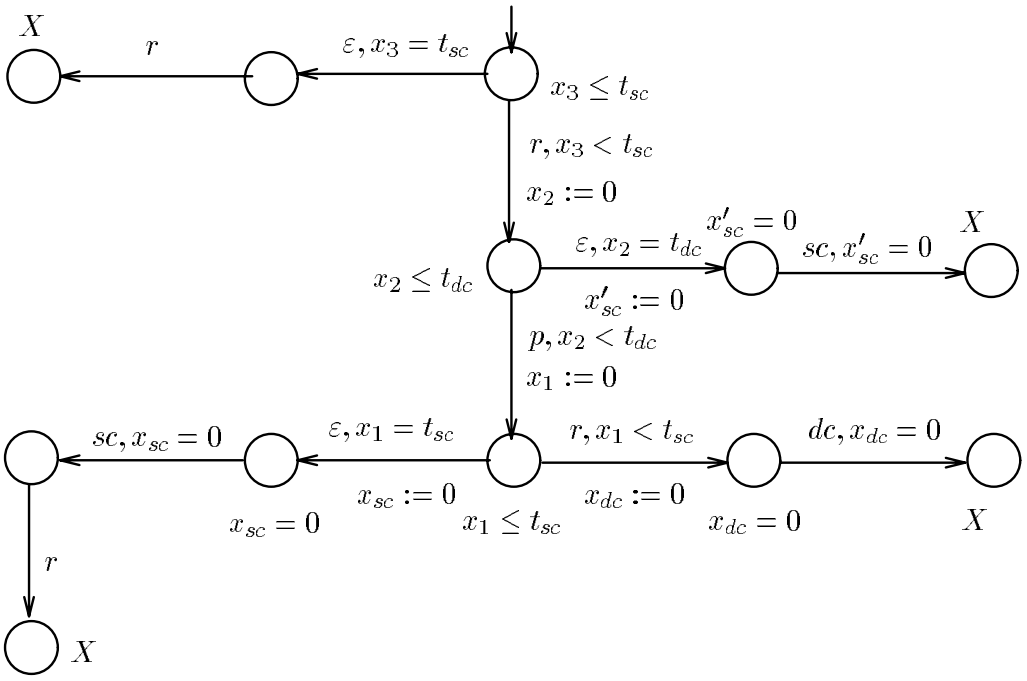
- Pour le sous-terme $M_4$ le graphe construit est :



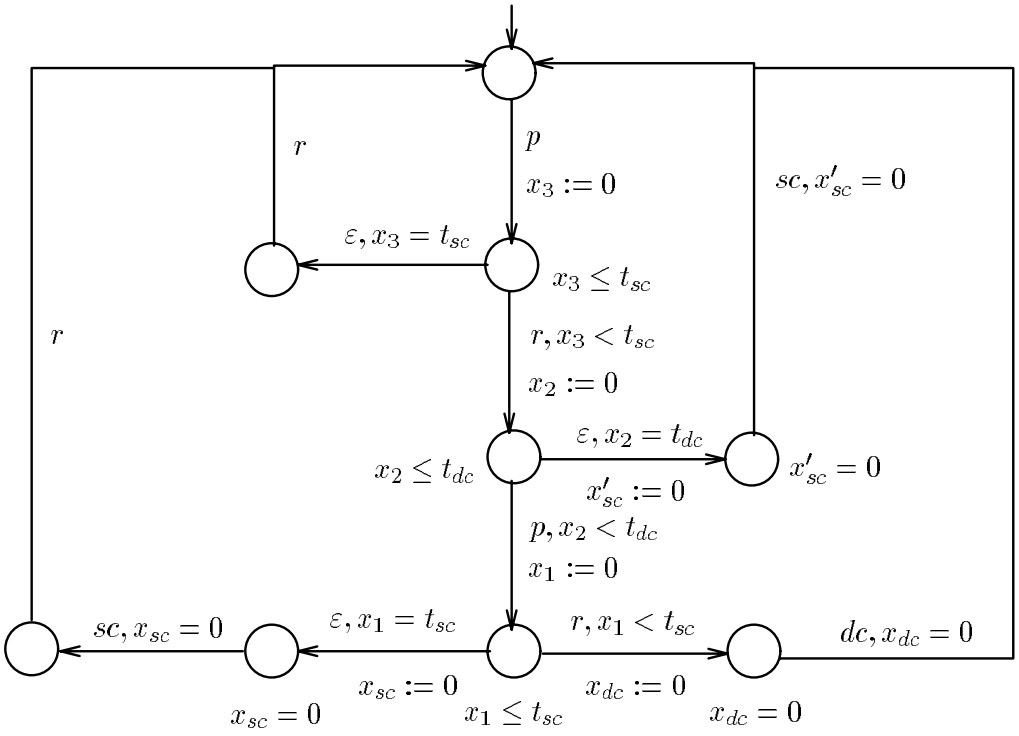

- Pour le sous-terme $M_3$ le graphe construit est :




- Pour le sous-terme $M_2$ le graphe construit est :




- Finalement, pour $M$ le graphe construit est :




## 6.6 Conclusions du chapitre

Nous avons présenté un méthode compositionnelle de construction d'un graphe temporisé fini pour les termes d'ATPqui ne contiennent pas des récursions interdites.

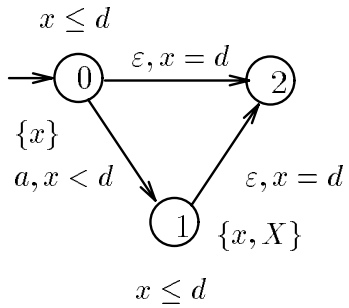
### La récursion interdite

La restriction imposée aux termes est justifiée car elle garantit que les duplications non gardées de processus ne peuvent pas se produire, ce qui permet d'obtenir toujours un graphe temporisé fini.

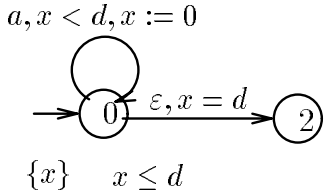
Comme nous l'avons déjà remarqué, certains termes contenant des récursions interdites peuvent néanmoins être représentés par des graphes temporisés finis. Un exemple est le terme $P$ défini comme $\mathsf{rec}X \cdot P'$ où $P'$ est le terme :

$$\mathsf{d}(a)\, X \;\mathsf{watchdog}[](d)\;\mathsf{idle}$$

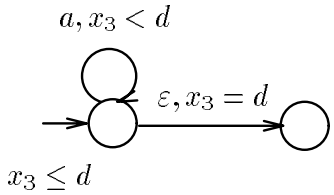
L'application de la méthode de construction pour $P'$ donne le graphe suivant :




L'application de la construction pour l'opérateur de récursion donne le graphe suivant :




L'horloge $x$ est initialisée par l'arc étiqueté par l'action $a$ ce qui ne correspond pas au comportement attendu. Pourtant, le comportement de $P$ peut être modélisé par le graphe temporisé fini suivant :




Ce graphe est similaire au graphe précédent sauf que l'arc étiqueté par l'action $a$ n'initialise pas l'horloge $x$. Ce graphe ne peut pas être construit par composition des graphes des sous-termes de $P$. De plus, $P$ est équivalent au terme :

$$(\mathsf{rec}X \cdot \mathsf{d}(a)\, X)\;\mathsf{watchdog}[](d)\;\mathsf{idle}$$

dont le graphe construit en appliquant la méthode est justement le graphe montré ci-dessus. En général, les comportements décrits avec l'utilisation des récursions interdites qui sont représentables par des graphes finis, peuvent être décrits plus simplement sans elle.

### Caractéristique de la méthode

Une caractéristique essentielle de la méthode de construction est que les paramètres des opérateurs watchdog sont traités symboliquement, c'est-à-dire, dans tous les cas, le graphe construit ne dépend pas des valeurs de ces paramètres, ce qui produit parfois des arcs qui ne seront jamais franchissables pour des valeurs particulières des délais.

Toutefois, plusieurs raisons justifient cette approche :

- Pour détecter les relations entre les délais introduits par les opérateurs watchdog il est nécessaire de connaître la structure des sous-termes. Ceci est contraire à l'approche compositionnelle.
- Il n'est pas possible de déterminer si la condition sur un arc est satisfaisable seulement à partir des valeurs des constantes. Pour ce faire, il faut savoir quelle est la relation qui existe entre les horloges qui interviennent dans la condition, mais celle-ci ne peut en général calculée *pendant la construction*. Toutefois, quelques optimisations peuvent êtres faites à la fin, lorsque le graphe a été complètement construit.

Enfin, ce traitement *symbolique* des contraintes temporelles présente quelques avantages intéressants :

- La taille du graphe généré est complètement indépendante des valeurs des paramètres.
- Il est possible d'étudier le comportement d'un système décrit par un terme d'ATP, avec différentes valeurs des délais, sans être obligé de construire le graphe à chaque fois. Ceci est important pour une mise au point affinée du système.

### Consistance de la méthode

Le problème initial que nous nous avons posé était de répondre à la question si un système temps-réel décrit au moyen d'un terme $P$ d'ATP satisfait une propriété exprimée par une formule $\varphi$ de la logique TCTL. Afin de résoudre ce problème, nous avons proposé dans de chapitre un algorithme de traduction des termes d'ATP vers des graphes temporisés, pour pouvoir appliquer ensuite l'algorithme de model-checking développé dans le chapitre 4.

Pour assurer la correction de cette approche, il faut prouver que la méthode de traduction est *consistante*, c'est-à-dire, que le modèle de $P$ et le modèle du graphe $G$ construit satisfont le même ensemble de formules. La preuve de la consistance de la méthode de construction est montrée dans l'annexe A.

# Chapitre 7

# Méthode de compilation

Une implémentation directe des règles de traduction données dans le chapitre 6 est possible; toutefois, ceci n'est pas convenable d'un point de vue pratique, car cette approche peut conduire à une dégradation des performances.

En effet, il n'est pas souhaitable d'opérer directement au niveau des graphes temporisés, car ceci entraîne la manipulation des structures coûteuses en mémoire. De plus, les règles produisent des sommets qui ne sont pas accessibles depuis le sommet initial, ce qui est corrigé par l'application de la fonction *Reach*. Cette procédure est assez coûteuse en temps d'exécution, car elle entraîne plusieurs parcours des graphes des sous-termes.

Nous proposons d'effectuer une traduction par étapes successives [NSY92, Yov92]. Cette approche a été aussi appliquée dans la compilation de langages comme LUSTRE [Ray91, Roc92] et LOTOS [Gar89].

La construction d'un graphe temporisé pour un terme d'ATP se fait en deux étapes. La première étape consiste à générer une forme intermédiaire compacte et simple à manipuler, à partir de laquelle un graphe temporisé peut être construit de façon performante dans la seconde étape.

Nous utilisons une forme intermédiaire basée sur les réseaux de Petri [Pet81, Rei85], qui ont la propriété de permettre une modélisation concise du parallélisme, ce qui évite l'explosion combinatoire de la taille de la représentation. Nous avons introduit des extensions au modèle des réseaux de Petri afin de prendre en compte les caractéristiques spéciales de certains opérateurs d'ATP, en particulier l'opérateur watchdog.

Nous présentons tout d'abord informellement la forme intermédiaire, appelée *réseau*, et nous expliquons le principe général de la méthode de compilation à l'aide de quelques exemples. Nous définissons ensuite formellement les réseaux et nous décrivons la méthode de génération d'un réseau pour un terme d'ATP. Finalement, nous montrons comment un graphe temporisé est construit à partir d'un réseau.

## 7.1 Présentation informelle de la méthode de compilation

La forme intermédiaire utilisée, appelée *réseau*, est une extension des réseaux de Petri. Un réseau comporte un ensemble de *places* et de *transitions*. Chaque transition possède un ensemble de *places d'entrée* et un ensemble de *places de sortie*. A la différence des réseaux de Petri, l'ensemble de places d'entrée d'une transition est divisé en deux sous-ensembles :

- un sous-ensemble de places *obligatoires*, et

- un sous-ensemble de places *optionnelles*.

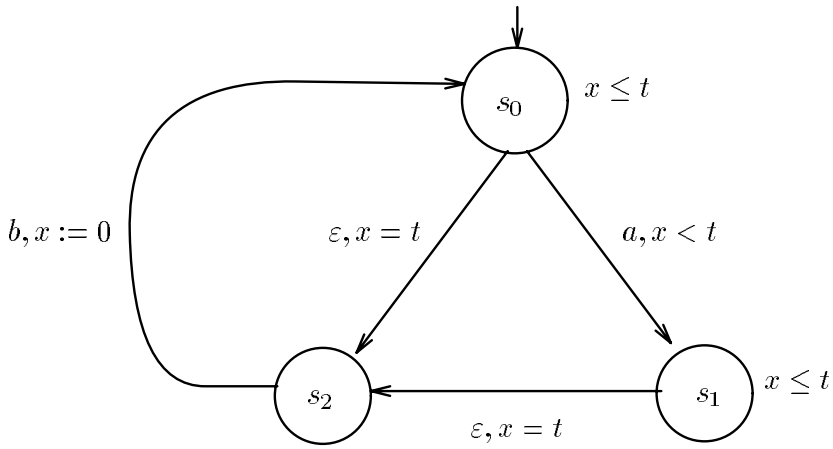
Une transition ne peut être franchie que si toutes les places d'entrée obligatoires sont marquées. Après franchissement, les places d'entrée obligatoires ainsi que les optionnelles qui sont marquées, perdent leurs marques, tandis que chaque place de sortie en reçoit une.

Nous présentons les idées générales de la méthode de compilation à l'aide de quelques exemples.

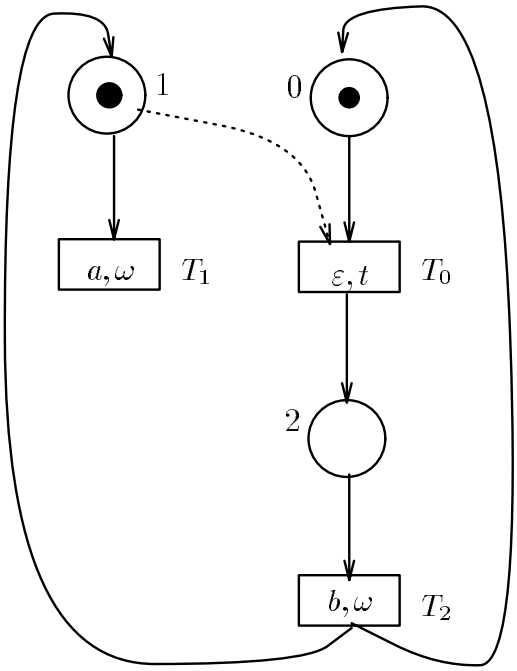
**Exemple 7.1** Considérons le terme $P$ suivant :

$$\mathsf{rec}X \cdot \mathsf{d}(a)\,\mathsf{idle}\ \mathsf{watchdog}[](t)\ \mathsf{d}(b)\,X$$

dont le graphe temporisé obtenu d'après la méthode présentée dans le chapitre 6 est :




Le réseau construit pour $P$ est le suivant :



Les transitions du réseau sont étiquetées par une action et par une valeur entière ou $\omega$ :

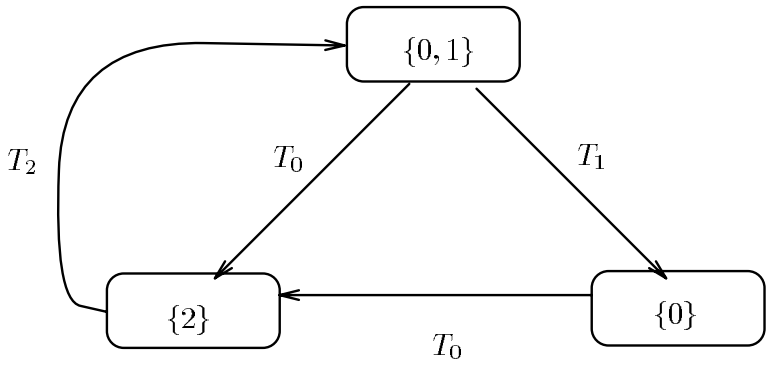
- La transition $T_1$ est étiquetée par $a$ et $\omega$, car elle représente l'exécution indéfiniment retardable de l'action $a$. Il en va de même pour la transition $T_2$ dont l'étiquette est $(b,\omega)$.
- La transition $T_0$ est étiquetée par $\varepsilon$ et $t$ ce qui représente l'expiration d'un watchdog dont le paramètre délai est $t$.

Les places d'entrée optionnelles sont indiquées au moyen des arcs en pointillé. Le comportement du réseau est le suivant. Initialement, les places 0 et 1 sont marquées. Ceci représente le fait que l'action $a$ et l'opérateur watchdog sont actifs. A partir de ce marquage deux situations peuvent se produire :

1. La transition $T_0$ est franchie, ce qui représente l'expiration du watchdog. L'exécution de cette transition *vide* toutes les places d'entrées, y compris les optionnelles. Dans le marquage résultant seule la place 2 est marquée.
2. La transition $T_1$ est franchie, ce qui représente l'exécution de l'action $a$. L'ensemble de places de sortie est vide, par conséquent aucune place neuve n'est marquée. Dans le marquage résultant seule la place 0 est marquée, ce qui exprime le fait que l'exécution de l'action $a$ n'annule pas l'effet du watchdog.

Considérons à présent le marquage où seule la place 0 est marquée. A partir de ce marquage, la transition $T_0$ est franchissable, même s'il n'y a pas de marque dans la place 1, car celle-ci est une place optionnelle.

Le système de transitions correspondant à ce réseau est le suivant :




Nous constatons qu'il y a une correspondance entre le graphe temporisé de $P$ et le système de transitions du réseau.

- Le sommet $s_0$ correspond au marquage $\{0,1\}$, où le watchdog et l'action $a$ sont actifs. Deux arcs sont issus de $s_0$ :
  - l'un est étiqueté par $\varepsilon$ et la condition $x = t$, correspondant à l'arc produit par la transition $T_0$.
  - l'autre est étiqueté par $a$ et la condition $x < t$, correspondant à l'arc produit par la transition $T_1$.
- Le sommet $s_1$ correspond au marquage $\{0\}$, où le watchdog est actif. L'arc de $s_1$ vers $s_2$ étiqueté par $\varepsilon$ correspond à l'arc du marquage $\{0\}$ vers le marquage $\{2\}$ produit par la transition $T_0$.

- Le sommet $s_2$ correspond au marquage $\{2\}$, où l'action $b$ est active. L'arc de $s_2$ vers $s_0$ étiqueté par $b$ correspond à l'arc de $\{2\}$ vers $\{0,1\}$ produit par la transition $T_2$.

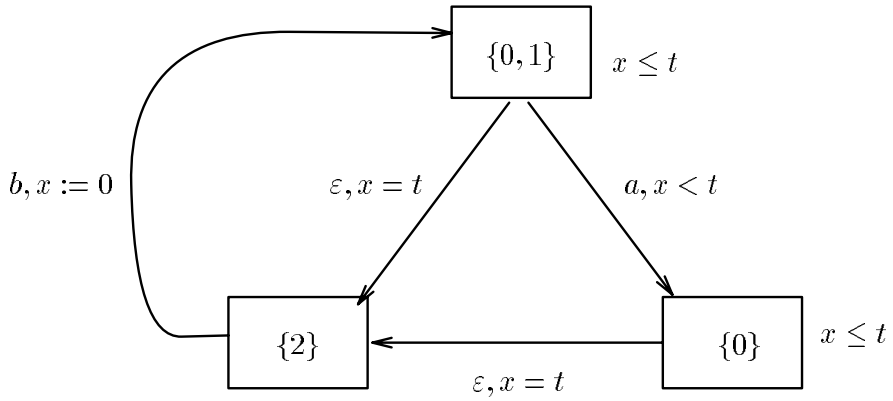
Les deux graphes sont donc isomorphes. Il reste alors à savoir comment étiqueter les marquages et les arcs du modèle du réseau pour obtenir ainsi le graphe temporisé. Pour ce faire, nous associons tout d'abord l'horloge $x$ à la transition $T_0$ étiquetée par $\varepsilon$. Nous pouvons à présent étiqueter le système de transitions du réseau pour obtenir le graphe temporisé de $P$ :

- La condition $x \leq t$ est associée aux marquages $\{0,1\}$ et $\{0\}$ car la transition $T_0$ est franchissable dans les deux cas. Aucune condition n'est imposée par la transition $T_1$ dont l'étiquette est $\omega$, car elle représente l'exécution d'une action indéfiniment retardable.
- La condition *true* est associée au marquage $\{2\}$, car la seule transition franchissable est étiquetée par $\omega$.

Les arcs sont étiquetés de la façon suivante :

- Aux arcs produits par la transition $T_0$ nous associons la condition $x = t$, car ils représentent l'expiration du watchdog.
- A l'arc produit par $T_1$ nous associons la condition $x < t$, ce qui est déterminé par le fait que la place d'entrée de la transition $T_1$, est une place d'entrée optionnelle de la transition $T_0$. Cet arc correspond à l'exécution de l'action $a$ sous la portée du watchdog.
- A l'arc produit par la transition $T_2$ nous associons la condition *true*, car il s'agit d'une action retardable. Puisque la transition $T_0$ est franchissable dans le marquage résultant, l'arc doit initialiser l'horloge $x$.

Le graphe temporisé ainsi obtenu est le suivant :

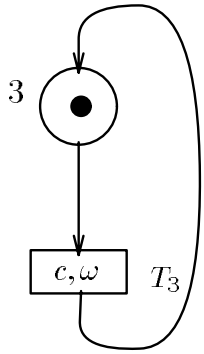



□

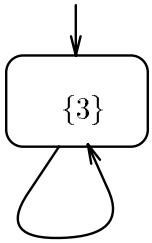
**Exemple 7.2** Considérons le terme $P'$ suivant :

$$\mathsf{rec}Y \cdot \mathsf{d}(c)\,Y$$

Le réseau produit pour ce terme est le suivant :




Le graphe temporisé construit à partir de ce réseau est le suivant :

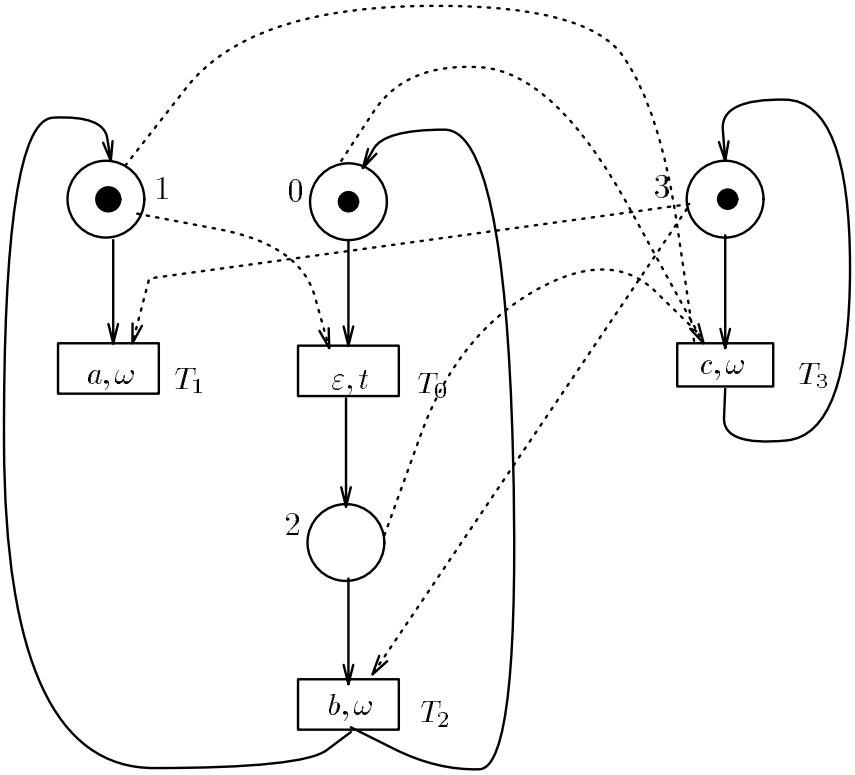



$c$

Considérons à présent le terme $P + P'$, où $P$ est le terme traité dans l'exemple 7.1, $P + P'$ est :

$$\mathsf{rec}X \cdot \mathsf{d}(a)\,\mathsf{idle}\ \mathsf{watchdog}[](t)\,\mathsf{d}(b)\,X + \mathsf{rec}Y \cdot \mathsf{d}(c)\,Y$$

Le réseau pour $P + P'$, obtenu a partir des réseaux de $P$ et de $P'$, est le suivant :

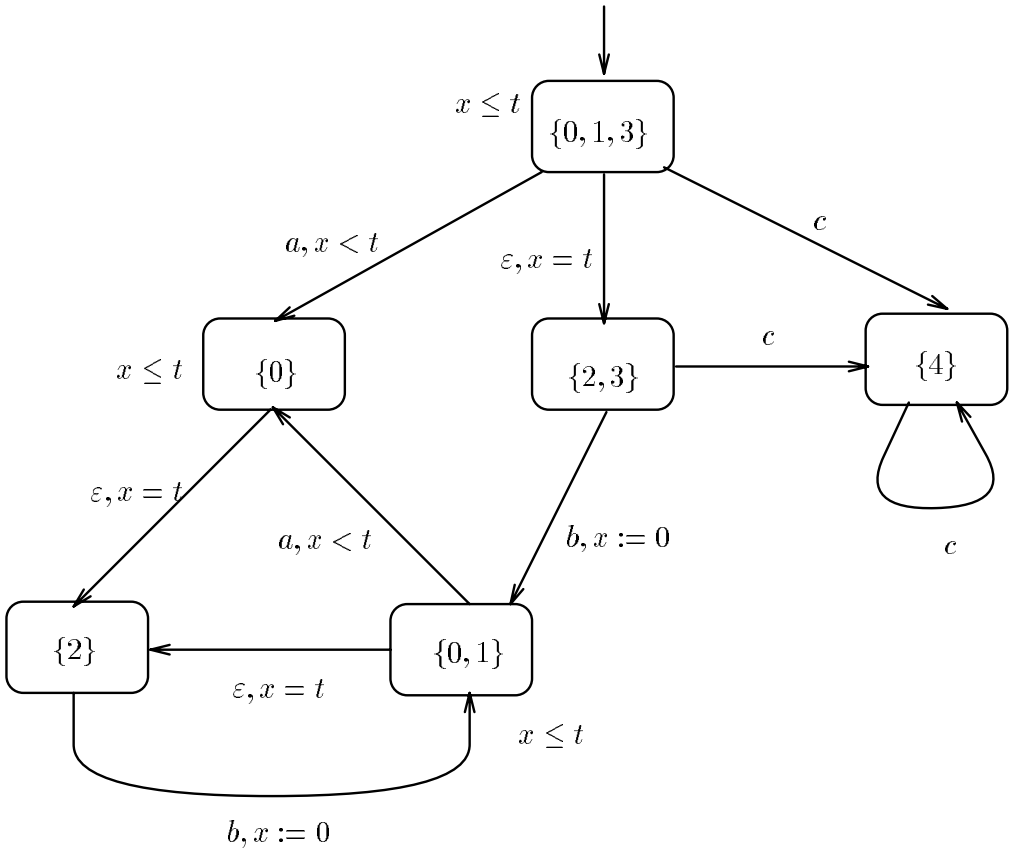


Ce réseau est l'union des réseaux de $P$ et $P'$, plus des liens en pointillé établis entre les places du réseau de $P$ (respectivement $P'$) et les transitions du réseau de $P'$ (respectivement $P$) étiquetées par des actions.

D'après la sémantique de l'opérateur de choix non déterministe, $P$ et $P'$ sont exécutés en parallèle jusqu'à l'instant où l'un des deux exécute une action qui interrompt l'autre et qui lui permet de continuer tout seul.

L'union des deux réseaux exprime le fait que $P$ et $P'$ sont exécutés en parallèle. Interrompre l'exécution d'un processus consiste à enlever toutes les marques du réseau correspondant. Pour ce faire, les places du réseau de $P$ (respectivement $P'$) sont considérées comme des places d'entrée optionnelles des transitions du réseau de $P'$ (respectivement $P$) dont les étiquettes sont des actions. Par exemple, l'ensemble des places d'entrée de la transition $T_3$ est $\{0,1,2\}$. Au contraire, aucune place optionnelle n'est ajoutée à $T_0$ car il s'agit d'une transition étiquetée par $\varepsilon$ qui correspond à l'expiration du watchdog.

Le graphe temporisé construit à partir du réseau est le suivant :




Afin de définir les conditions d'activité et les conditions sur les arcs nous associons à la transition étiquetée par $\varepsilon$ l'horloge $x$.

Ensuite, à tout marquage qui contient la place 0, qui est la place d'entrée obligatoire de la transition $T_0$, nous associons la condition d'activité $x \leq t$.

Pour définir la condition sur un arc de $M$ à $M'$ il faut déterminer si la transition $T_0$ est franchissable à partir de $M$, et si son ensemble de places d'entrée optionnelles contient des places d'entrée obligatoires de la transition qui produit l'arc.

Considérons par exemple l'arc de $\{0,1,3\}$ vers $\{0\}$, qui est produit par la transition $T_1$ dont l'étiquette est $(a,\omega)$ et dont l'ensemble des places d'entrée obligatoires est $\{1\}$. La transition

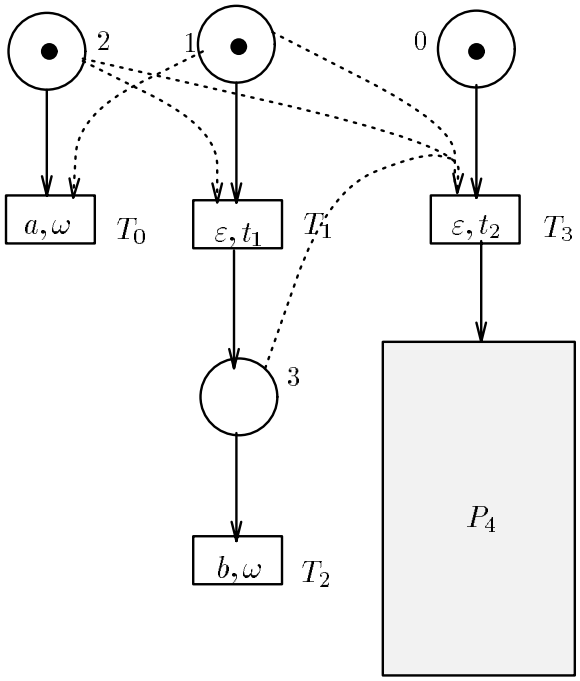
$T_0$ peut être franchie à partir de $\{0,1,3\}$, et son ensemble de places optionnelles est $\{1\}$ qui est égal à l'ensemble des places d'entrée obligatoires de $T_1$. Dans ce cas, nous associons à l'arc la condition $x < t$, ce qui exprime le fait que l'action $a$ est exécutée sous la portée du watchdog.

Considérons à présent l'arc de $\{0,1,3\}$ vers $\{2,3\}$. Dans ce cas, la condition est $x = t$ car il s'agit d'un arc produit par la transition $T_0$ dont l'étiquette est $(\varepsilon, t)$.

Si l'arc est produit par une transition étiquetée par une action et $\omega$ dont les places d'entrée obligatoires ne sont pas des places d'entrée optionnelles de la transition $T_0$, alors la condition sur l'arc est *true*. Ceci exprime le fait que l'action n'est pas sous la portée du watchdog.

Nous pouvons constater que ce graphe est identique à celui construit par la méthode présentée dans le chapitre 6. □

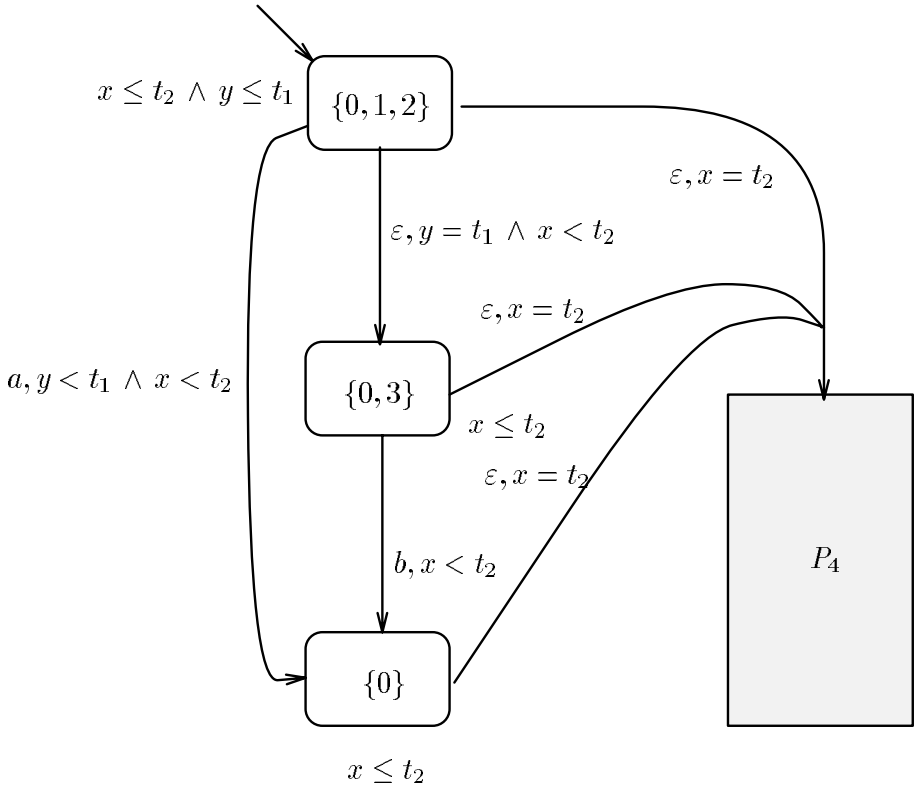
**Exemple 7.3** Soit $P_1$ le terme $\mathsf{d}(a)\,\mathsf{idle}$, $P_2$ le terme $P_1$ $\mathsf{watchdog}[a](t_1)$ $\mathsf{d}(b)\,\mathsf{idle}$, et $P_3$ le terme $P_2$ $\mathsf{watchdog}[](t_2)$ $P_4$ où $P_4$ est un terme quelconque. Le réseau obtenu pour $P_3$ est le suivant :




La transition $T_3$ étiquetée par $\varepsilon$ et $t_2$ représente l'expiration du watchdog dont le paramètre délai est $t_2$. Toutes les places du réseau de $P_2$, i.e., les places 1, 2 et 3, sont des places optionnelles de cette transition, ce qui représente le fait que l'expiration du watchdog interrompt l'exécution du processus $P_2$. Par conséquent, le franchissement de cette transition enlève toutes les marques dans le réseau de $P_2$.

La transition $T_1$ étiquetée par $\varepsilon$ et $t_1$ représente l'expiration du watchdog de durée $t_1$. Dans ce cas, toutes les places du réseau de $P_1$, i.e, la place 2, sont enlevées par l'exécution de cette transition. De plus, la place 1, qui est la place d'entrée obligatoire de $T_1$, est une place optionnelle de la transition $T_0$, ce qui exprime le fait que l'exécution de l'action $a$ annule l'effet du watchdog.

Le graphe temporisé construit à partir de ce réseau est le suivant :




Afin de définir les conditions d'activité et les conditions sur les arcs nous associons à chacune des transitions étiquetées par $\varepsilon$ une horloge : $x$ à $T_3$ et $y$ à $T_1$.

Les conditions d'activité sont définies comme suit :

- La condition du marquage $\{0,1,2\}$ est $x \leq t_2 \wedge y \leq t_1$, car les transitions $T_1$ et $T_3$ sont franchissables à partir de ce marquage.
- La condition des marquages $\{0,3\}$ et $\{0\}$ est $x \leq t_2$, car est $T_3$ franchissable, mais pas $T_1$.

Pour déterminer la condition sur un arc il faut considérer toutes les transitions étiquetées par $\varepsilon$ qui :

- sont franchissables à partir du marquage, et
- dont l'ensemble des places d'entrée optionnelles contient des places d'entrée de la transition qui produit l'arc.

Par exemple :

- La condition associée à l'arc de $\{0,1,2\}$ vers $\{0\}$ est $x < t_2 \wedge y < t_1$.
- La condition associée à l'arc de $\{0,1,2\}$ vers $\{0,3\}$ est $y = t_1 \wedge x < t_2$.

Comme dans l'exemple précédent, nous pouvons constater que ce graphe est identique à celui construit en appliquant la méthode présentée dans le chapitre 6. □

Nous avons illustré à l'aide de quelques exemples comment construire un réseau pour un terme d'ATP pour ensuite obtenir son graphe temporisé à partir du modèle du réseau. Dans la section suivante, nous définissons formellement les réseaux et nous décrivons la méthode de compilation.

## 7.2 Construction d'un réseau pour un terme

Nous présentons ici la méthode de construction d'un réseau pour un terme d'ATP.

### 7.2.1 Définition de réseau

Nous définissons tout d'abord la notion de réseau.

**Définition 7.1** Un *réseau* est un triplet $N = \langle \mathcal{O}, \mathcal{T}, M_I \rangle$ où :

- $\mathcal{O}$ est l'ensemble des *places*,
- $\mathcal{T} \subseteq \mathcal{O} \times \mathcal{O} \times (Act_\varepsilon \times \mathtt{N} \cup \{\omega\}) \times \mathcal{O}$ est l'ensemble des *transitions*,
- $M_I \subseteq \mathcal{O}$ est un sous-ensemble de places.

□

Soit $T = \langle \mathcal{O}_1, \mathcal{O}_2, \alpha, d, \mathcal{O}_3 \rangle$ une transition.

- $\mathcal{O}_1$ est l'ensemble des *places d'entrée obligatoires* de $T$, noté $\bullet T$.
- $\mathcal{O}_2$ est l'ensemble des *places d'entrée optionnelles* de $T$, noté $\circ T$.
- $\mathcal{O}_3$ est l'ensemble des *places de sortie* de $T$, noté $T\bullet$.
- L'attribut d'une transition $T$ est le couple $(\alpha, d)$, $\alpha$ est notée $etiq(T)$ et $d$ est noté $borne(T)$.

Un *marquage* est un ensemble de places. Le marquage $M_I$ est appelé *marquage initial*. La sémantique d'un réseau est un système de transitions entre les marquages. Pour une transition $T \in \mathcal{T}$ et deux marquages $M$ et $M'$, $M \xrightarrow{T} M'$ indique qu'il est possible de passer du marquage $M$ au marquage $M'$ en franchissant la transition $T$ :

- Pour que la transition $T$ puisse être franchie à partir de $M$ il faut que toutes les places d'entrée *obligatoires* de $T$ soient marquées dans $M$.
- Le marquage $M'$ est obtenu à partir de $M$ dans lequel les places d'entrée obligatoires et optionnelles de $T$ perdent leur marque, tandis que les places de sortie en reçoivent une.

La relation de transition est formellement définie par la règle suivante :

$$\frac{\bullet T \subseteq M \;,\; M' = M - (\bullet T \cup \circ T) \cup T\bullet}{M \xrightarrow{T} M'}$$

L'ensemble de tous les marquages *accessibles* à partir du marquage initial $M_I$ est noté $\mathbf{M}$.

Nous ne considérons que des réseaux *saufs*, i.e., tels qu'à tout instant chaque place contient *au plus* une marque. Le système de transitions de $N$ est $\langle \mathbf{M}, \rightarrow, M_I \rangle$.

### 7.2.2 Construction d'un réseau pour un terme

Pour un terme $P$ d'ATP qui ne contient pas de récursion interdite, le réseau de $P$, noté $\mathcal{N}[\![P]\!]$, est construit par induction sur la structure de $P$ selon les règles suivantes.

### Le processus idle

Le réseau du processus **idle**, est simplement un réseau vide.

$$\mathcal{N}[\![\mathsf{idle}]\!] \quad = \quad \langle \emptyset, \emptyset, \emptyset \rangle$$

### Les variables de processus

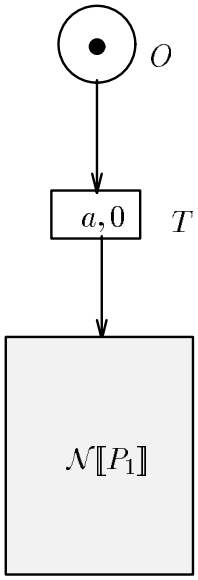
Pour une variable $X \in Var$, le réseau construit a une seule place $O$ dont aucune transition n'est issue. Cette place est étiquetée par $X$ afin de pouvoir l'identifier au moment de la construction du réseau pour le terme $\mathsf{rec}X\cdot$.

$$\mathcal{N}[\![X]\!] \quad = \quad \langle \{O\}, \emptyset, \{O\} \rangle$$

Nous associons à $O$ l'attribut $var(O) = X$ qui indique que $O$ représente une occurrence libre de la variable $X$.

### L'opérateur de préfixage

Soit $a \in Act$ et $P$ le terme $aP_1$. Dans ce cas, une nouvelle place $O$ est créée ainsi qu'une transition $T$ dont $O$ est la place d'entrée obligatoire et les places initiales du réseau de $P_1$ sont les places de sortie. L'attribut de $T$ est le couple $(a, 0)$, ce qui exprime le fait qu'il s'agit d'un préfixage immédiat par l'action $a$.




Si le réseau de $P_1$ est $\mathcal{N}[\![P_1]\!] = \langle \mathcal{O}_1, \mathcal{T}_1, M_I^1 \rangle$, alors le réseau de $aP_1$ est :

$$\mathcal{N}[\![aP_1]\!] \quad = \quad \langle \mathcal{O}_1 \cup \{O\}, \mathcal{T}_1 \cup \{T\}, \{O\} \rangle$$

où $O \notin \mathcal{O}_1$ et $T$ est la transition $\langle \{O\}, \emptyset, a, 0, M_I^1 \rangle$.

Le réseau construit pour le terme $\mathsf{d}(a)P_1$, est similaire à celui de $aP_1$. Dans ce cas, l'attribut de $T$ est le couple $(a, \omega)$, ce qui représente le fait qu'il s'agit d'un préfixage retardable par l'action $a$.

$$\mathcal{N}[\![\mathsf{d}(a)P_1]\!] \quad = \quad \langle \mathcal{O}_1 \cup \{O\}, \mathcal{T}_1 \cup \{T\}, \{\{O\}\} \rangle$$

où $O \notin \mathcal{O}_1$ et $T$ est la transition $\langle \{O\}, \emptyset, a, \omega, M_I^1 \rangle$.

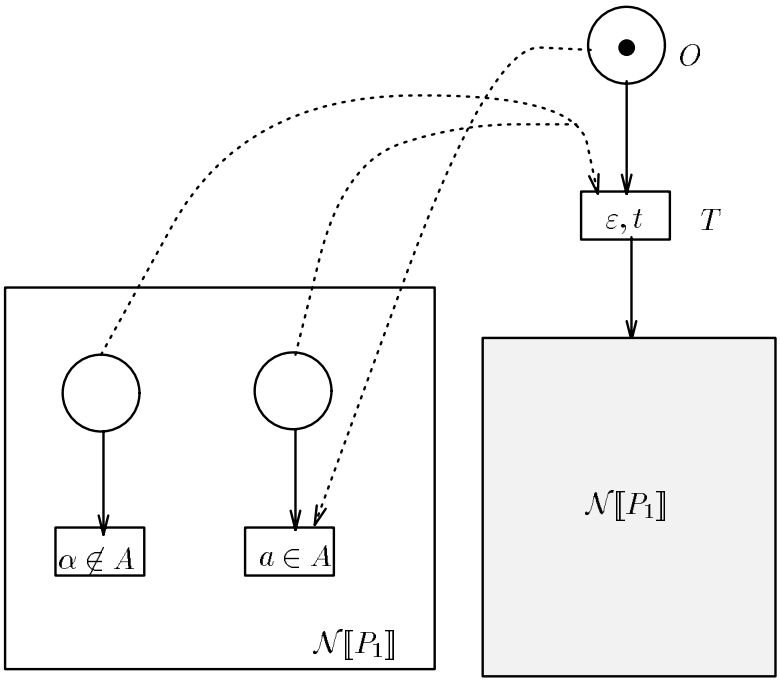
### L'opérateur watchdog

Soit $P$ le terme $P_1$ watchdog$[A](t)$ $P_2$. La construction du réseau de $P$ commence par la création d'une nouvelle place $O$ et d'une nouvelle transition $T$ ayant $O$ comme place d'entrée obligatoire et les places initiales du réseau de $P_2$ comme places de sortie.

La place $O$ représente le watchdog. Si elle est marquée, alors le watchdog est actif, i.e., le comportement de $P_1$ est contrôlé par le watchdog.

Pour exprimer le fait que $P_1$ peut être interrompu par l'expiration du watchdog pour exécuter $P_2$, toutes les places du réseau de $P_1$ sont liées à la transition $T$ comme des places d'entrée optionnelles.

Ensuite, afin d'exprimer le fait que $P_1$ peut annuler l'effet du watchdog au moyen d'une action de $A$, la place $O$ est liée comme place d'entrée optionnelle à toutes les transitions du réseau de $P_1$ étiquetées par une action de $A$.




Si pour les termes $P_1$ et $P_2$, les réseaux construits sont $\mathcal{N}[\![P_i]\!] = \langle \mathcal{O}_i, \mathcal{T}_i, M_I^i \rangle$, pour $i = 1, 2$, alors le réseau pour $P$ est :

$$\mathcal{N}[\![P_1 \text{ watchdog}[A](d)\ P_2]\!] \quad = \quad \langle \mathcal{O}_1 \cup \mathcal{O}_2 \cup \{O\}, \mathcal{T}_1' \cup \mathcal{T}_2 \cup \{T\}, M_I^1 \cup \{O\} \rangle$$

où $O \notin \mathcal{O}_1 \cup \mathcal{O}_2$,

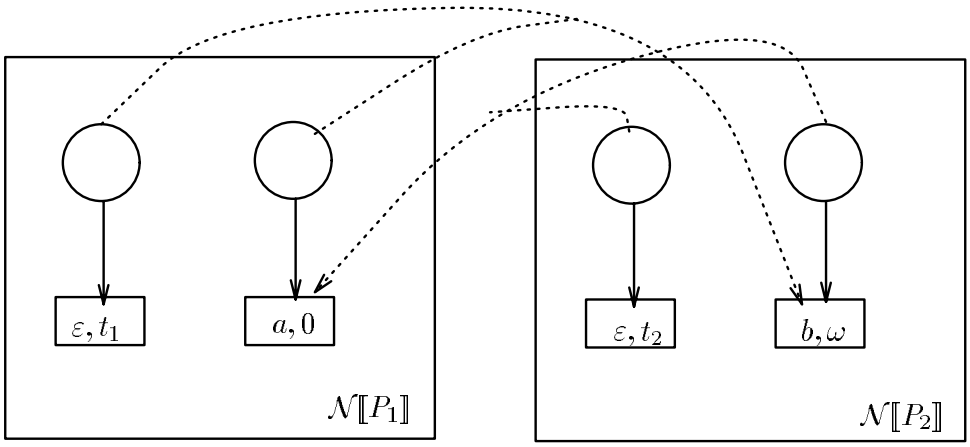
$$\begin{aligned} \mathcal{T}_1' \quad = \quad & \{T_1 \in \mathcal{T}_1 \mid etiq(T_1) \notin A\} \\ & \cup \{\langle \bullet T_1, \circ T_1 \cup \{O\}, etiq(T_1), borne(T_1), T_1 \bullet \rangle \mid T_1 \in \mathcal{T}_1 \wedge etiq(T_1) \in A\} \end{aligned}$$

et $T$ est la transition $\langle \{O\}, \mathcal{O}_1, \varepsilon, d, M_I^2 \rangle$.

### L'opérateur de choix non déterministe

Soit $P$ le terme $P_1 + P_2$. Le réseau de $P$ est l'union des réseaux construits pour $P_1$ et $P_2$. Afin d'exprimer le fait que l'exécution d'une action de la part de $P_1$ ($P_2$) interrompt $P_2$ ($P_1$), toutes

les places du réseau de $P_2$ ($P_1$) sont des places optionnelles des transitions du réseau de $P_1$ ($P_2$) qui sont étiquetées par des actions.




Si pour les termes $P_1$ et $P_2$, les réseaux construits sont $\mathcal{N}[\![P_i]\!] = \langle \mathcal{O}_i, \mathcal{T}_i, M_I^i \rangle$, pour $i = 1, 2$, alors le réseau pour $P$ est :

$$\mathcal{N}[\![P_1 + P_2]\!] \quad = \quad \langle \mathcal{O}_1 \cup \mathcal{O}_2, \mathcal{T}_1' \cup \mathcal{T}_2', M_I^1 \cup M_I^2 \rangle$$

où

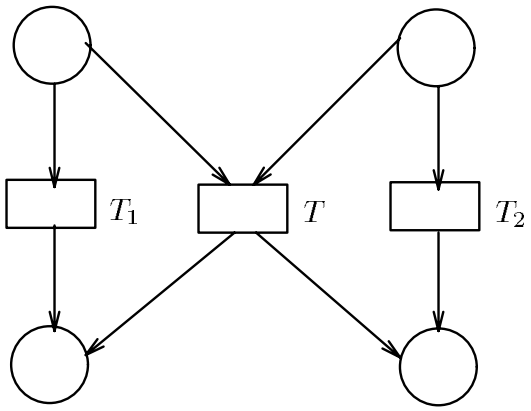
$$\begin{array}{rcl} \mathcal{T}_1' & = & \{T \in \mathcal{T}_1 \mid etiq(T) = \varepsilon\} \\ & & \cup \{\langle \bullet T, \circ T \cup \mathcal{O}_2, etiq(T), borne(T), T\bullet \rangle \mid T \in \mathcal{T}_1 \wedge etiq(T) \in Act\} \\ \mathcal{T}_2' & = & \{T \in \mathcal{T}_2 \mid etiq(T) = \varepsilon\} \\ & & \cup \{\langle \bullet T, \circ T \cup \mathcal{O}_1, etiq(T), borne(T), T\bullet \rangle \mid T \in \mathcal{T}_2 \wedge etiq(T) \in Act\} \end{array}$$

### L'opérateur de composition parallèle

Soit $P$ le terme $P_1 \,||\, P_2$. La construction du réseau pour $P$ commence par unir les réseaux de $P_1$ et $P_2$, afin d'exprimer le fait que $P_1$ et $P_2$ doivent être exécutés en parallèle.

Ensuite, nous procédons au *couplage* des transitions afin de modéliser la communication entre $P_1$ et $P_2$. On dit que $T_1$ et $T_2$ sont *synchronisables* si $etiq(T_1) \mid etiq(T_2) \in Act$, i.e., si $T_1$ et $T_2$ représentent les exécutions de deux actions $a_1$ et $a_2$ qui peuvent communiquer. Coupler deux transitions synchronisables $T_1$ et $T_2$ consiste à créer une nouvelle transition $T$ telle que :

$$\begin{array}{rcl} \bullet T & = & \bullet T_1 \cup \bullet T_2 \\ T\bullet & = & T_1 \bullet \cup T_2 \bullet \end{array}$$




L'ensemble des places optionnelles de $T$ est aussi l'union des places optionnelles de $T_1$ et $T_2$ :

$$\circ T \quad = \quad \circ T_1 \cup \circ T_2$$

L'attribut de $T$ est le couple $(a, d)$ tel que :

$$\begin{aligned} a &= \mathit{etiq}(T_1) \mid \mathit{etiq}(T_2) \\ d &= \min(\mathit{borne}(T_1), \mathit{borne}(T_2)) \end{aligned}$$

c'est-à-dire, coupler une transition $T_1$ avec $\mathit{borne}(T_1) = 0$ et une transition $T_2$ quelconque donne une transition $T$ avec $\mathit{borne}(T) = 0$, ce qui exprime le fait que si $\mathit{etiq}(T_1)$ est une action immédiate alors la communication entre $\mathit{etiq}(T_1)$ et $\mathit{etiq}(T_2)$ doit se produire immédiatement.

Si pour les termes $P_1$ et $P_2$, les réseaux construits sont $\mathcal{N}[\![P_i]\!] = \langle \mathcal{O}_i, \mathcal{T}_i, M_I^i \rangle$, pour $i = 1, 2$, alors le réseau pour $P$ est :

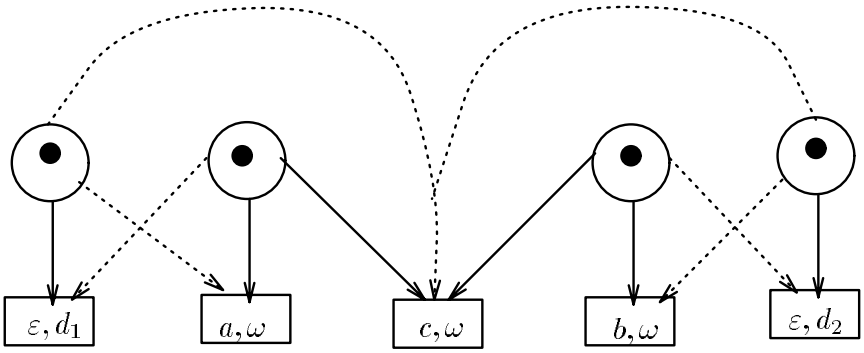
$$\mathcal{N}[\![P_1 \,||\, P_2]\!] \quad = \quad \langle \mathcal{O}_1 \cup \mathcal{O}_2, \mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{T}, M_I^1 \cup M_I^2 \rangle$$

où $\mathcal{T}$ est l'ensemble des transitions produites par les couples $(T_1, T_2)$ de transitions synchronisables.

**Exemple 7.4** Soit $P$ le terme :

$$\mathsf{d}(a)\,\mathsf{idle}\ \mathsf{watchdog}[a](d_1)\ \mathsf{idle} \,||\, \mathsf{d}(b)\,\mathsf{idle}\ \mathsf{watchdog}[b](d_2)\ \mathsf{idle}$$

et supposons que $a \mid b = c$. Le réseau construit pour $P$ est le suivant :




□

### L'opérateur de restriction

Soit $P$ le terme **restrict** $A$ **in** $P_1$. Le réseau de $P$ est obtenu à partir du réseau de $P_1$ en éliminant toutes les transitions $T$ dont l'étiquette $etiq(T)$ appartient à $A$.

Si le réseau de $P_1$ est $\mathcal{N}[\![P_1]\!] = \langle \mathcal{O}_1, \mathcal{T}_1, M_I^1 \rangle$, alors le réseau de **restrict** $A$ **in** $P_1$ est :

$$\mathcal{N}[\![\textsf{restrict } A \textsf{ in } P_1]\!] \quad = \quad \langle \mathcal{O}_1, \mathcal{T}_1 - \mathcal{T}^-, M_I^1 \rangle$$

où $\mathcal{T}^- = \{T \in \mathcal{T}_1 \mid etiq(T) \in A\}$.

### L'opérateur de récursion

Soit $P$ le terme **rec**$X \cdot P_1$. Le réseau de $P$ est obtenu à partir du réseau de $P_1$ en identifiant toutes les places étiquetées par la variable $X$ avec les places initiales de $P_1$. Autrement dit, toute transition allant vers une place étiquetée par la variable $X$ est remplacée par une transition allant vers les places initiales de $P_1$.

Si le réseau de $P_1$ est $\mathcal{N}[\![P_1]\!] = \langle \mathcal{O}_1, \mathcal{T}_1, M_I^1 \rangle$, alors le réseau de **rec**$X \cdot P_1$ est :

$$\mathcal{N}[\![\textsf{rec}X \cdot P_1]\!] \quad = \quad \langle \mathcal{O}_1 - \mathcal{O}^-, \mathcal{T}, M_I^1 \rangle$$

où $\mathcal{O}^- = \{O \in \mathcal{O}_1 \mid var(O) = X\}$ et

$$\mathcal{T} \quad = \quad \{\langle {}^\bullet T, T^\circ, etiq(T), borne(T), T^\bullet - \mathcal{O}^- \cup M_I^1 \rangle \mid T \in \mathcal{T}_1\}$$

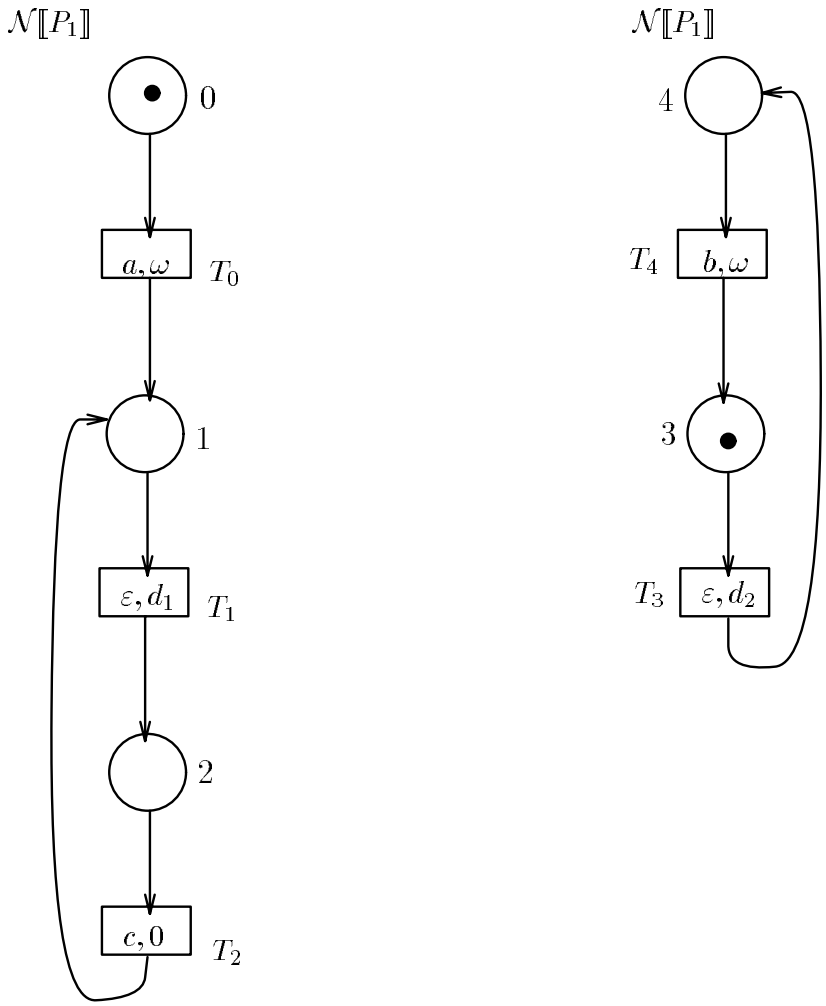
**Exemple 7.5** Soit $P_1$ le terme :

$$\textsf{d}(a)\,\textsf{rec}X \cdot \textsf{idle watchdog}[](d_1)\; c\, X$$

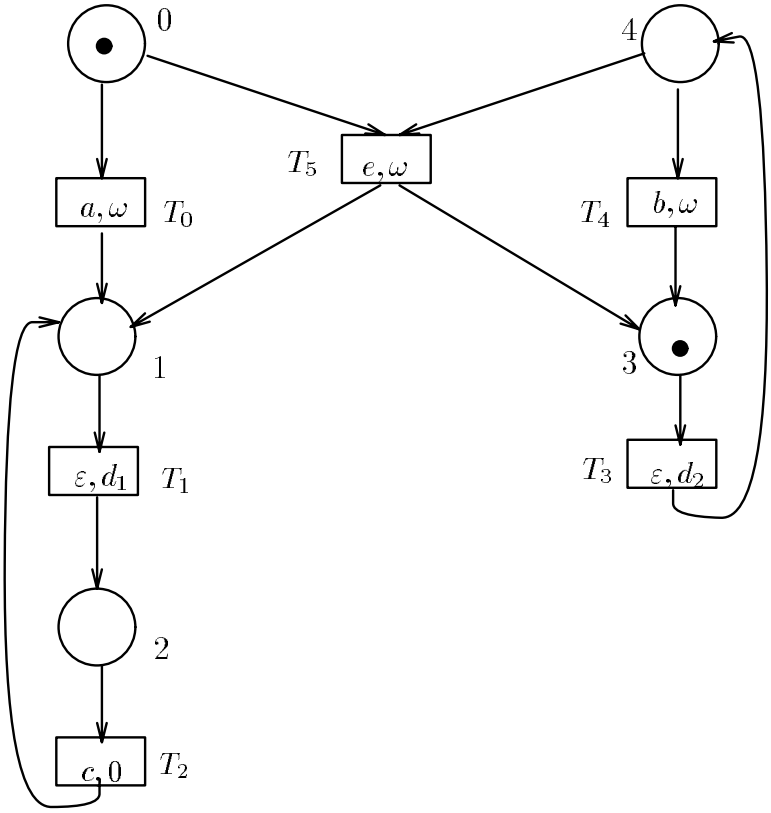
et $P_2$ le terme :

$$\textsf{rec}Y \cdot \textsf{idle watchdog}[](d_2)\; \textsf{d}(b)\, Y$$

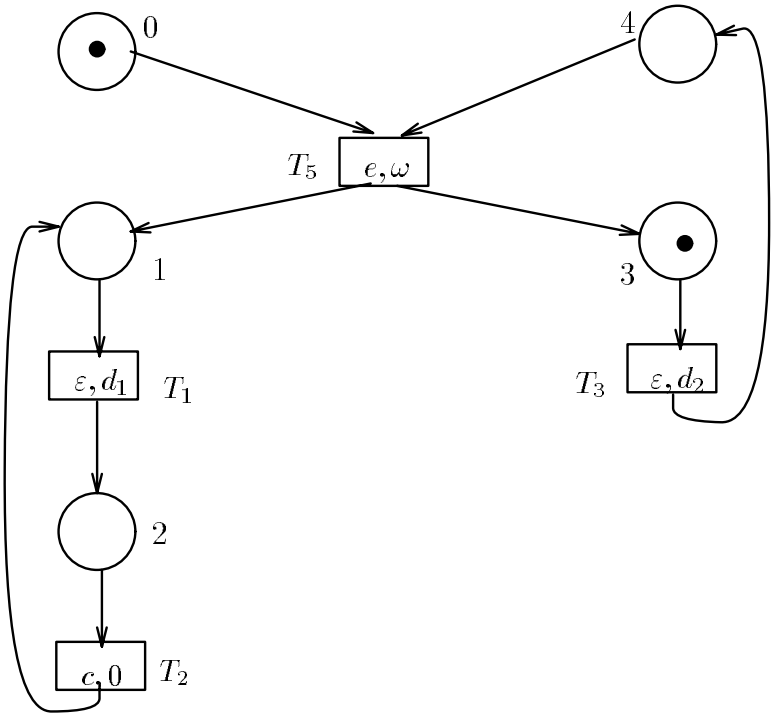
Les réseaux construits pour $P_1$ et $P_2$ sont les suivants :




Soit $P$ le terme $P_1 \,||\, P_2$. En supposant que $a \mid b = e$ et $c \mid b = \bot$, le réseau construit pour $P$ est le suivant :




Soit $P'$ le terme **restrict** $a, b$ **in** $P$. Le réseau construit pour $P'$ est le suivant :




□

## 7.3 Construction d'un graphe temporisé

La seconde étape consiste à construire le graphe temporisé d'un terme $P$ à partir du réseau $\mathcal{N}[\![P]\!]$ généré dans la première étape.

Soit $N = \langle \mathcal{O}, \mathcal{T}, M_I \rangle$ le réseau de $P$. Le graphe temporisé de $P$ est construit comme suit :

1. A chaque transition $T \in \mathcal{T}$ telle que $borne(T) \neq \omega$ nous associons une horloge notée $hor(T)$.
2. Ensuite, le modèle de $N$ est construit.
3. La condition d'activité d'un marquage est la conjonction des conditions de la forme $hor(T) \leq borne(T)$ pour toutes les transitions $T$ telles que $borne(T) \neq \omega$ qui sont franchissables à partir du marquage. Si le marquage est vide, alors la condition d'activité est *true*.
4. Soit $e$ l'arc de $M$ vers $M'$ produit par la transition $T$ :
   (a) L'étiquette de $e$ est $etiq(T)$.
   (b) La condition de $e$ est la conjonction de plusieurs conditions. Il faut distinguer deux cas :
      i. Si $borne(T) \neq \omega$ alors le premier facteur est $hor(T) = borne(T)$, ce qui représente l'exécution d'une action immédiate ou l'expiration d'un watchdog.
      ii. Si $borne(T) = \omega$ alors le premier facteur est *true*, ce qui correspond à l'exécution d'une action retardable.

   Pour déterminer les autres facteurs il faut considérer toutes les transitions $T'$ qui satisfont :
      i. $\bullet T' \subseteq M$
      ii. $etiq(T') = \varepsilon$
      iii. $\circ T' \cap \bullet T \neq \emptyset$

   D'après les règles de construction du réseau, si $T'$ satisfait ces trois conditions alors elle correspond à un watchdog qui (1) contrôle l'événement (exécution d'une action ou expiration d'un watchdog) modélisé par la transition $T$, et qui (2) n'a pas encore expiré.

Soit $\langle \mathbf{M}, \rightarrow, M_I \rangle$ le modèle de $N$. Le graphe $G = \langle S, H, E, s_I, \delta \rangle$ construit à partir de $N$ est tel que :

1. L'ensemble des sommets $S$ est l'ensemble des marquages $\mathbf{M}$ accessibles depuis le marquage initial $M_I$.
2. Le sommet initial $s_I$ est le marquage initial $M_I$.
3. $H$ est l'ensemble des horloges associées aux transitions $T \in \mathcal{T}$ telles que $borne(T) \neq \omega$.
4. Pour tout $M \subseteq \mathbf{M}$, la condition d'activité est :

$$\delta_M \quad = \quad \bigwedge_{\bullet T \subseteq M, borne(T) \neq \omega} hor(T) \leq borne(T)$$

5. Soient $M, M' \subseteq \mathbf{M}$ et $T \in \mathcal{T}$ tels que $M \xrightarrow{T} M'$ et $\mathcal{T}'$ l'ensemble de transitions défini par :

$$\mathcal{T}' \quad = \quad \{T' \in \mathcal{T} \mid {}^\bullet T' \subseteq M \wedge etiq(T') = \varepsilon \wedge {}^\bullet T \cap {}^\circ T' \neq \emptyset\}$$

A l'arc $M \xrightarrow{T} M'$ correspond un arc $e = \langle M, etiq(T), \psi, H', M' \rangle$ tel que :

(a) la condition $\psi$ est la conjonction $\psi_1 \wedge \psi_2$ où $\psi_1$ est :

$$\psi_1 = \begin{cases} hor(T) = borne(T) & \text{si } borne(T) \neq \omega, \\ true & \text{sinon.} \end{cases}$$

et $\psi_2$ est :

$$\psi_2 \quad = \quad \bigwedge_{T' \in \mathcal{T}'} hor(T') < borne(T')$$

(b) l'ensemble $H'$ des horloges à initialiser est :

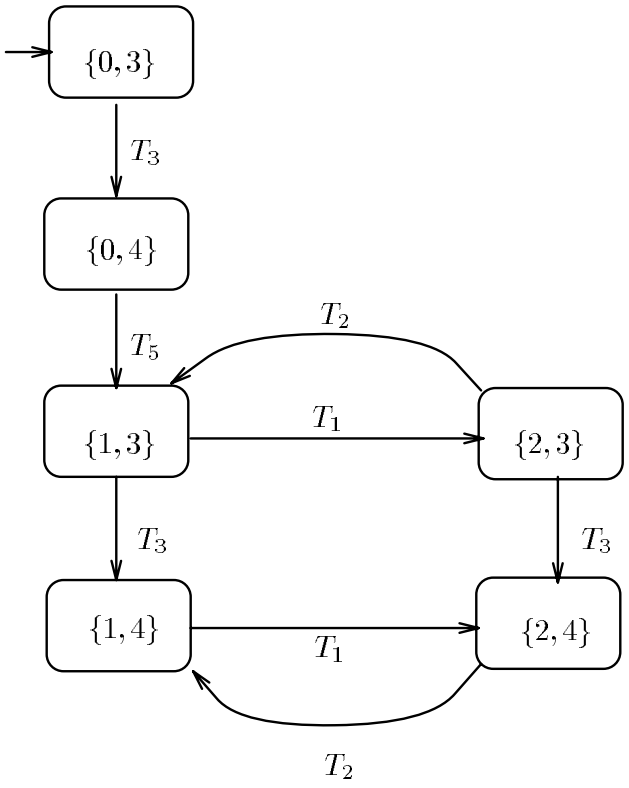
$$H' \quad = \quad \{hor(T') \mid {}^\bullet T' \subseteq M' \wedge {}^\bullet T' \not\subseteq M\}$$

c'est-à-dire, l'ensembles des horloges associées aux transitions qui deviennent franchissables après l'exécution de $T$.

**Exemple 7.6** Soit $P$ le terme :

$$\mathsf{restrict}\ a, b\ \mathsf{in}\ (\mathsf{d}(a)\,\mathsf{rec}X \cdot \mathsf{idle}\ \mathsf{watchdog}[](d_1)\,c\,X \,||\, \mathsf{rec}Y \cdot \mathsf{idle}\ \mathsf{watchdog}[](d_2)\,\mathsf{d}(b)\,Y)$$

et $a \mid b = e$ et $c \mid b = \perp$. Le réseau construit pour $P$ est illustré dans l'exemple 7.5. Le modèle du réseau de $P$ est le suivant :



A partir de ce graphe, le graphe temporisé de $P$ est construit en appliquant la méthode décrite ci-dessus.

Tout d'abord, nous associons une horloge à chaque transition $T$ qui satisfait $borne(T) \neq \omega$ :

$$\begin{aligned} hor(T_1) &= x \\ hor(T_2) &= y \\ hor(T_3) &= z \end{aligned}$$

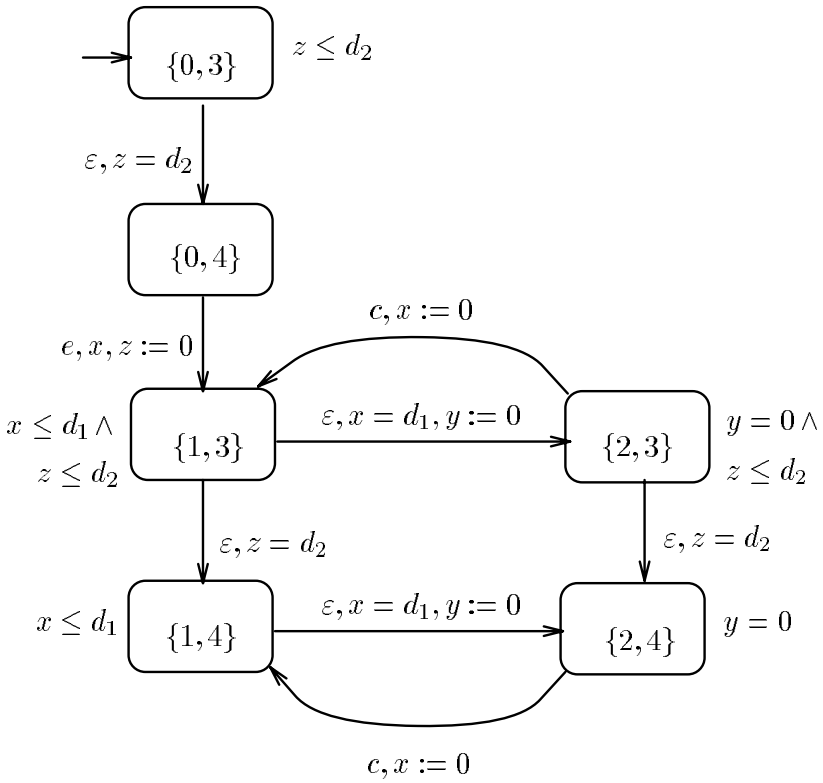
Pour chaque marquage, la condition d'activité est obtenue de la façon suivante :

1. La condition d'activité de $\{0,3\}$ est $z \leq d_2$, car la seule transition franchissable à partir de ce marquage est $T_3$ dont la borne est $d_2$ et dont l'horloge associée est $z$.
2. Pour le marquage $\{0,4\}$, la condition est *true*.
3. La condition associée au marquage $\{1,3\}$ est $x \leq d_1 \wedge z \leq d_2$, car les transitions $T_1$ et $T_3$ sont franchissables à partir de ce marquage.
4. La condition d'activité de $\{2,3\}$ est $y = 0 \wedge z \leq d_2$.
5. Pour $\{1,4\}$, la condition associée est $x \leq d_1$.
6. La condition associée à $\{2,4\}$ est $y = 0$.

Pour toute transition, l'ensemble des places optionnelles est vide. Donc, pour la transition $T_5$ la condition sur l'arc correspondant est *true* et pour toutes les autres la condition est $hor(T) = borne(T)$.

Considérons, par exemple, l'arc de $\{0,4\}$ vers $\{1,3\}$ produit par la transition $T_5$. Dans ce cas, l'ensemble des horloges à initialiser par l'arc correspondant du graphe temporisé est $\{x,z\}$, car $x$ et $z$ sont les horloges associées aux transitions $T_1$ et $T_3$ qui deviennent franchissables.

Le graphe temporisé obtenu est le suivant :



Nous pouvons constater que le graphe ainsi obtenu est identique au graphe construit par la méthode présentée dans le chapitre 6. □

**Exemple 7.7** Considérons le terme $P$ qui modélise le comportement de la souris à un seul bouton décrit dans l'exemple 5.1. Le terme $P$ est $\mathsf{rec}X \cdot M_1$ où :

$$M_1 \stackrel{\text{def}}{=} \mathsf{d}(p)\, M_2$$
$$M_2 \stackrel{\text{def}}{=} \mathsf{d}(r)\, M_3\ \mathsf{watchdog}[r](t_{cs})\ \mathsf{d}(r)\, X$$
$$M_3 \stackrel{\text{def}}{=} \mathsf{d}(p)\, M_4\ \mathsf{watchdog}[p](t_{cd})\ cs\, X$$
$$M_4 \stackrel{\text{def}}{=} \mathsf{d}(r)\, cd\, X\ \mathsf{watchdog}[r](t_{cs})\ cs\, \mathsf{d}(r)\, X$$

Le réseau illustré par la figure 7.1 est le réseau construit pour $P$ où nous avons fait quelques optimisations.

Considérons, par exemple, la transition $T_2$ qui représente l'expiration du watchdog correspondant au terme $M_2$. D'après les règles de construction données, l'ensemble des places d'entrée optionnelles de $T_2$ devrait être l'ensemble de toutes les places du réseau de $M_2$. Toutefois, franchir la transition $T_1$ dont l'étiquette est $r$ enlève la marque de la place 2 qui est la place d'entrée obligatoire de $T_2$. Il suffit donc de lier la place 1 à $T_2$ comme place d'entrée optionnelle. En effet, la place 2 n'est marque dans aucun des marquages accessibles après le franchissement de la transition $T_1$; elle ne reçoit une marque qu'après le franchissement de $T_0$.

De plus, la place 2 devrait être une place d'entrée optionnelle de la transition $T_6$ dont l'étiquette est $r$. En suivant le même raisonnement que ci-dessus, nous constatons que si la place 6, i.e, la place d'entrée obligatoire de $T_6$, est marquée, alors la place 2, ne l'est pas. Donc, il n'est pas nécessaire de lier la place 2 à la transition $T_6$ par un trait en pointillé.

Nous avons suivi la même procédure d'optimisation pour toutes les autres transitions étiquetées par $\varepsilon$. □

**Exemple 7.8** Considérons le terme $P$ qui modélise le comportement du contrôleur de température décrit dans l'exemple 5.2. Le terme $P$ est $\mathsf{d}(r)\mathsf{rec}X \cdot C_2$ où :

$$C_2 \stackrel{\text{def}}{=} b_1\, C_3$$
$$C_3 \stackrel{\text{def}}{=} C_4\ \mathsf{watchdog}[r](t_{max})\ X$$
$$C_4 \stackrel{\text{def}}{=} \mathsf{d}(r)\, C_6\ \mathsf{watchdog}[r](t_1)\ (b_2\, C_5 + r\, C_6)$$
$$C_5 \stackrel{\text{def}}{=} \mathsf{d}(r)\, C_6\ \mathsf{watchdog}[r](t_2)\ e\ \mathsf{d}(r)\, X$$
$$C_6 \stackrel{\text{def}}{=} h\ \mathsf{idle}$$

Le réseau construit pour $P$ est illustré par la figure 7.2. Les places 7, 8, 9, 10, 13 et 14 sont aussi des places d'entrée optionnelles de la transition étiquetée par $(\varepsilon, t_{max})$, mais nous n'avons pas dessiné les liens en pointillé afin d'éviter de compliquer la figure. □

## 7.4 Implémentation

Nous avons développé un compilateur qui implémente la méthode de traduction décrite dans les sections précédentes. L'architecture du compilateur est illustrée par la figure 7.3.

Le compilateur comporte plusieurs phases dont les plus importantes sont la construction du réseau et la génération du graphe temporisé.

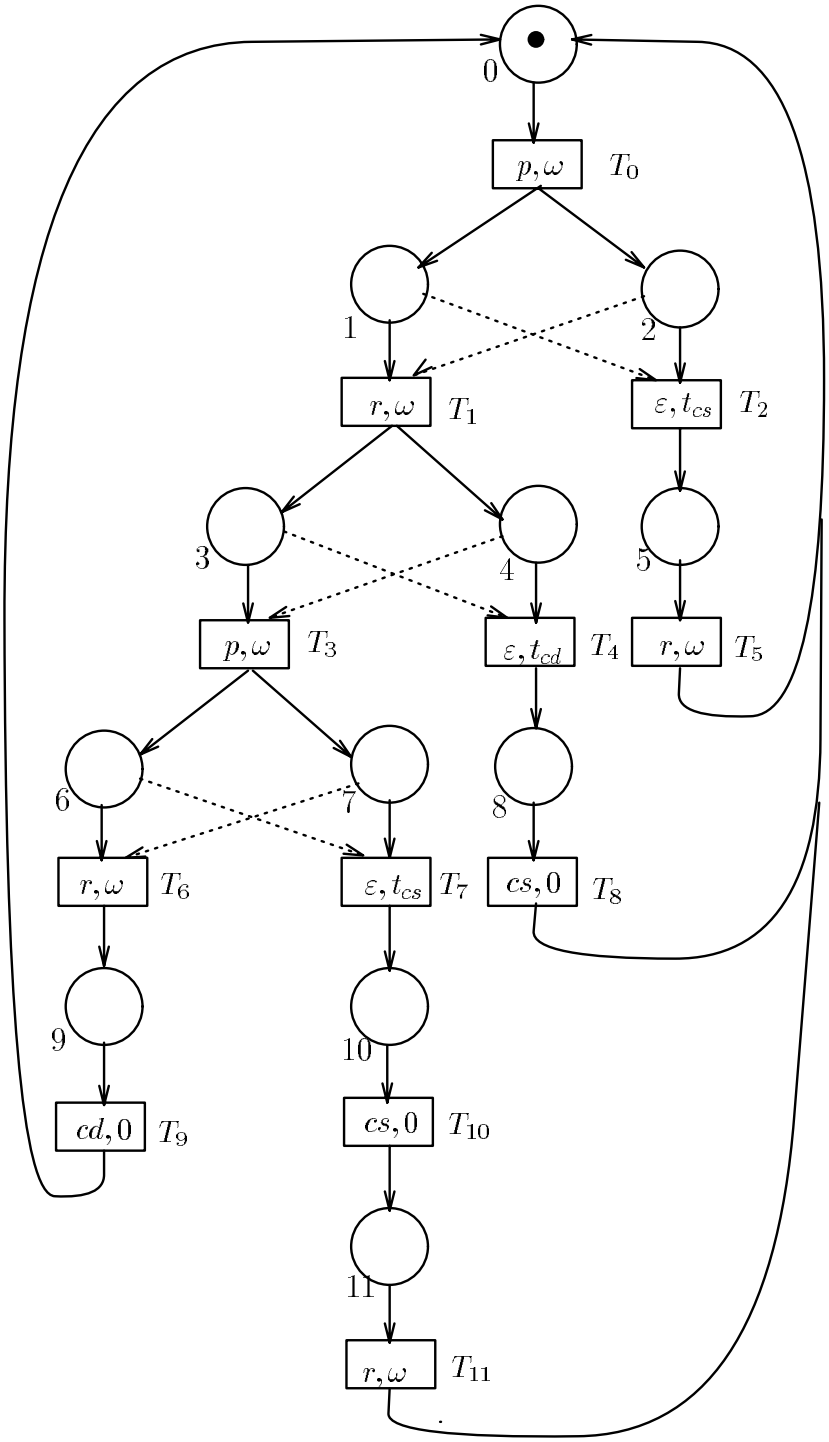



Figure 7.1: Réseau de la souris.

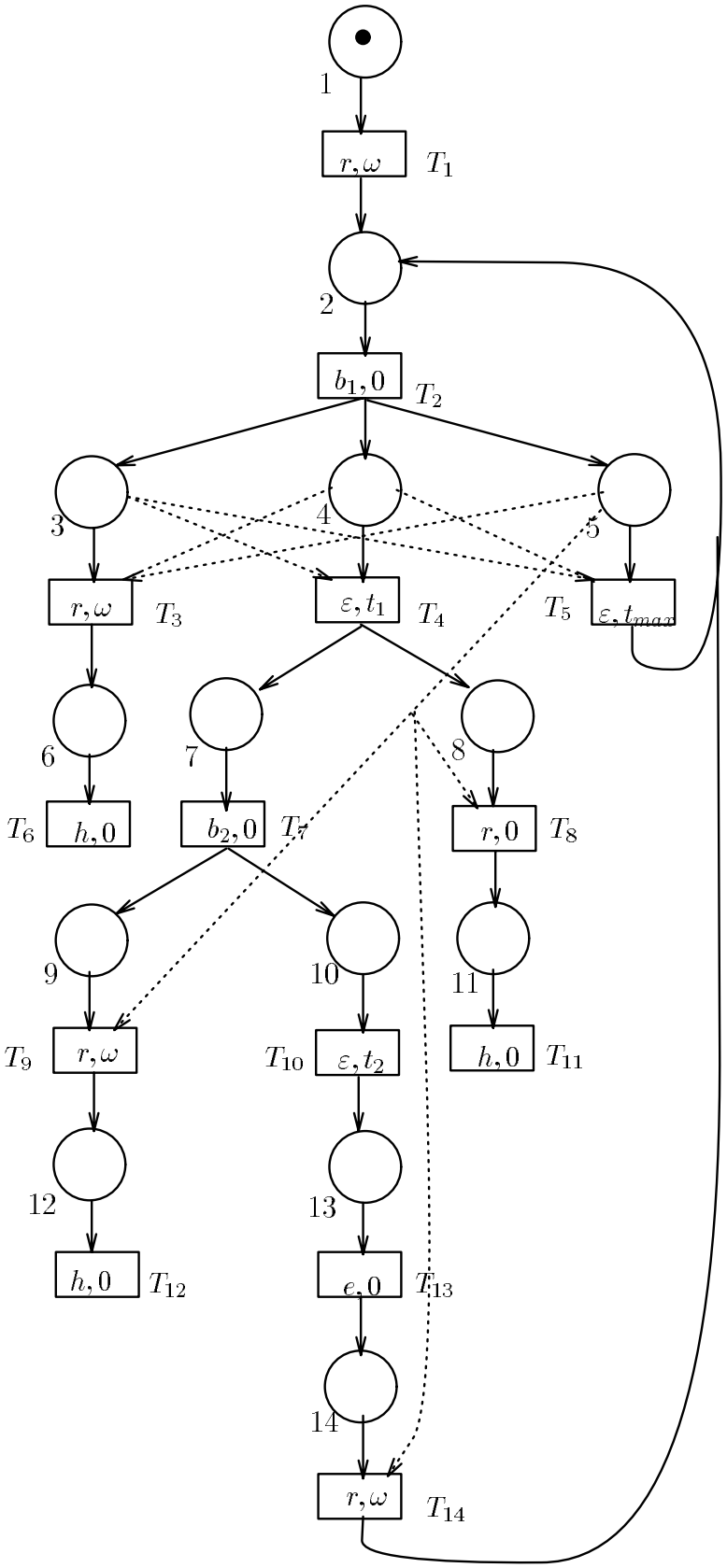



Figure 7.2: Réseau du contrôleur de température.

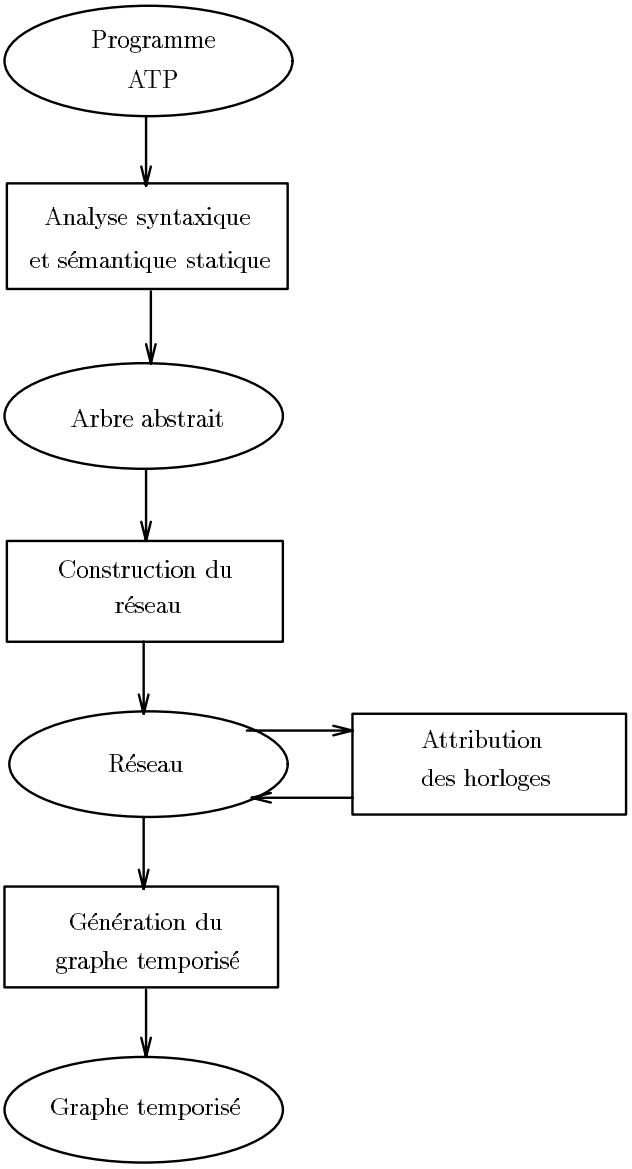



Figure 7.3: Architecture du compilateur.

1. La première phase comporte l'analyse syntaxique et sémantique statique du programme ATP. Pour la mise en œuvre de l'analyse syntaxique nous avons utilisé les outils SYNTAX [BD88] et CÆSAR.ADT [Gar89]. L'analyse sémantique statique consiste notamment à détecter les termes qui contiennent des récursions interdites.

2. La deuxième phase comporte la construction du réseau en appliquant la méthode décrite dans la section 7.2.2. L'étape d'attribution des horloges consiste à associer des horloges aux transitions.

3. La phase finale est la génération du graphe temporisé à partir du réseau en appliquant les règles présentées dans la section 7.3.

Nous présentons ici l'algorithme et les structures de données utilisées pour la mise en œuvre de la construction du graphe temporisé.

### 7.4.1 Construction du graphe

Soit $N = \langle \mathcal{O}, \mathcal{T}, M_I \rangle$ le réseau construit pour le terme $P$. L'algorithme de génération du graphe temporisé est le suivant :

**begin**
  $S^+ := \emptyset$
  $S^- := \{M_I\}$
  $E := \emptyset$
  **while** $S^- \neq \emptyset$
    **let** $M \in S^-$
    $S^+ := S^+ \cup \{M\}$
    $S^- := S^- - \{M\}$
    *construire_condition*$(M)$
    **forall** $T, M' : \langle M, T, M' \rangle$
      $e :=$ *construire_arc*$(M, T, M')$
      $E := E \cup \{e\}$
      **if** $M' \notin S^+ \cup S^-$
      **then** $S^- := S^- \cup \{M'\}$
    **end**
  **end**
**end**

L'algorithme mémorise les marquages dans un ensemble divisé en deux parties. Le sous-ensemble noté $S^+$ contient les marquages visités dont les successeurs ont été déterminés. Le sous-ensemble $S^-$ contient le reste des marquages visités.

Chaque marquage est représenté par une chaîne de bits telle qu'à chaque place correspond un bit, qui vaut 1 si la place appartient au marquage et 0 sinon.

Les marquages sont mémorisés dans une table extensible formée d'une liste chaînée de pages de mémoire, chaque page pouvant contenir un nombre fixé de marquages. La table est gérée de la façon suivante :

- Une variable $N$ donne le nombre de marquages dans la table. $N$ est incrémentée chaque fois qu'un nouvel marquage est inséré. Chaque marquage est repéré par un indice variant de 0 à $N-1$.
- La table est partagée en deux zones par un indice $K$. Les marquages dont l'indice est entre 0 et $K-1$ appartient à $S^+$. Les autres sont ceux de $S^-$.
- A tout instant la valeur de $K$ est l'indice du prochain marquage à traiter. La construction est finie lorsque $K$ est égal à $N$.
- Lorsque l'algorithme construit un marquage, on cherche dans la table au moyen de son indice et on l'insère s'il n'est pas présent.
- Pour la recherche d'un marquage on utilise une structure de données auxiliaire qui permet un accès rapide par hachage (figure 7.4). Les indices des marquages qui ont la même image par la fonction de hachage sont mémorisés hors de la table sous forme d'une liste chaînée de paquets. La recherche d'un marquage dans une liste se fait séquentiellement.

Cette structure de données est similaire à celle utilisée par l'outil CÆSAR [Gar89] de compilation de programmes LOTOS.

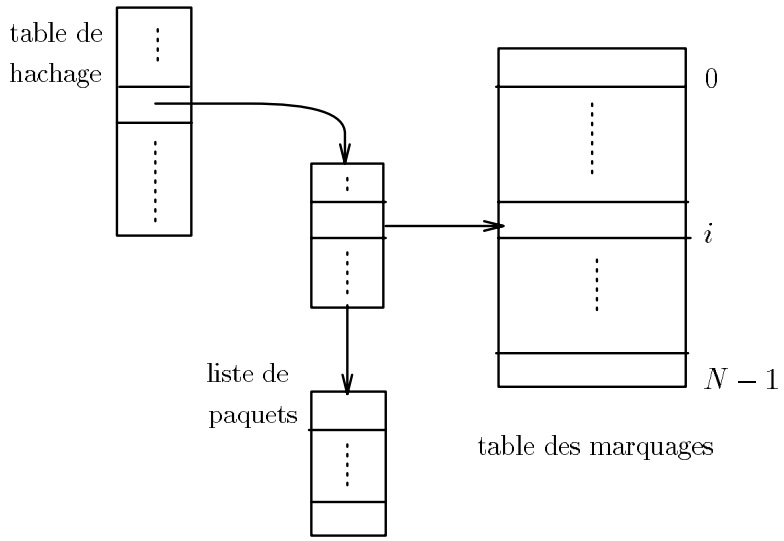



Figure 7.4: Table des marquages.

Finalement, nous constatons que les arcs du graphe ne sont jamais consultés. Par conséquent, nous pouvons ranger les arcs en mémoire secondaire dans un fichier à accès séquentiel, où les arcs sont ajoutés au fur et à mesure de leur création.

### 7.4.2 Réduction du nombre d'horloges

Comme nous l'avons déjà remarqué, la méthode de construction donnée dans le chapitre 6 introduit les horloges de manière systématique, ce qui produit en général des graphes avec un nombre

trop grand d'horloges. Nous présentons ici un algorithme qui permet de réduire le nombre d'horloges utilisées.

Dans la méthode présentée dans la section 7.3, les horloges sont introduites après la construction du réseau et sont associées aux certaines transitions. L'algorithme consiste tout d'abord à détecter s'il est possible d'associer à deux transitions $T_1$ et $T_2$ la même horloge :

- Nous constatons tout d'abord que si $T_1$ et $T_2$ sont telles que $borne(T_1) = borne(T_2) = 0$ alors il n'est pas nécessaire d'avoir une horloge pour $T_1$ et une autre pour $T_2$. En effet, une seule horloge suffit pour toutes les transitions dont la borne est 0.

- Considérons à présent le réseau construit pour le terme $P_1$ watchdog$[A](t)\ P_2$. Soit $T$ la transition qui représente l'expiration du watchdog. Dans ce cas, il n'est pas possible d'associer la même horloge à $T$ et à une transition $T_1$ du réseau de $P_1$ telle que $\bullet T_1 \in \circ T$. Soient $N_1$ et $N_2$ les réseaux des termes $P_1$ et $P_2$ respectivement.

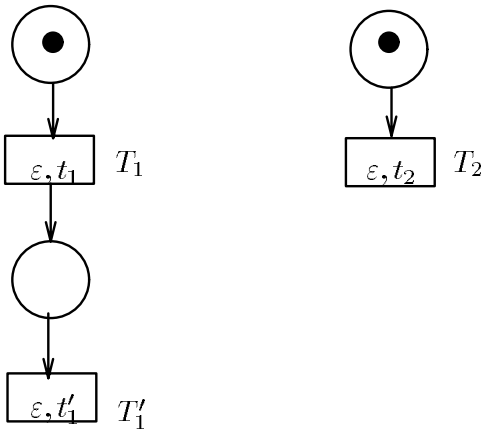
  Nous avons donc que l'attribution d'horloges doit satisfaire pour toute transition $T_1$ de $N_1$, si $\bullet T_1 \subseteq \circ T$ alors $hor(T_1) \neq hor(T)$.

  Par exemple, dans le cas du réseau du contrôleur de température illustré par la figure 7.2, nous avons $hor(T_4) \neq hor(T_5)$ et $hor(T_{10}) \neq hor(T_5)$.

- Considérons le terme $P_1 + P_2$. Soit $N_i$ le réseau de $P_i$, pour $i = 1, 2$.

  Si $T_1 \in \mathcal{T}_1$ et $T_2 \in \mathcal{T}_2$ sont franchissables dans le marquage initial alors l'horloge de $T_1$ doit être différente à celle de $T_2$, car il s'agit de deux opérateurs watchdog qui sont actifs au même temps.

  Considérons à présent une transition $T_1'$ telle que $\bullet T_1' \subseteq T_1\bullet$. Dans ce cas, il faut aussi que l'horloge associée à $T_1'$ soit différente à celle associée à $T_2$. Il en va de même pour les transitions $T_2'$ telles que $\bullet T_2' \subseteq T_2\bullet$.




  Pour $i = 1, 2$, soit $\mathcal{T}_i^{\varepsilon}$ l'ensemble des transitions étiquetées par $\varepsilon$. Nous définissons une relation $\succ_i$ entre les transitions de $\mathcal{T}_i^{\varepsilon}$ telle que pour tout $T_i, T_i' \in \mathcal{T}_i^{\varepsilon}$, $T_i \succ_i T_i'$ si $\bullet T_i' \subseteq T_i\bullet$.

  La fonction *hor* doit satisfaire pour tout $T_1, T_1' \in \mathcal{T}_1^{\varepsilon}$ et pour tout $T_2, T_2' \in \mathcal{T}_2^{\varepsilon}$, si $\bullet T_i \subseteq M_I^i$ et $T_i \succ_i^* T_i'$, alors $hor(T_1') \neq hor(T_2')$.

- Pour le terme $P_1 \,||\, P_2$, l'ensemble des horloges associées aux transitions des réseaux de $P_1$ de $P_2$ doivent être disjoint, i.e., pour toute transition $T_1$ de $N_1$ et $T_2$ de $N_2$, $hor(T_1) \neq hor(T_2)$.

On construit ensuite un graphe non orienté $\langle V, A\rangle$ où $V$ est l'ensemble des sommets et $A$ est l'ensemble des arcs, tel que :

- $V$ est l'ensemble $\mathcal{T}^{\varepsilon}$ des transitions étiquetées par $\varepsilon$,
- et il y a un arc $(T_1, T_2) \in A$ si la fonction *hor* doit satisfaire $hor(T_1) \neq hor(T_2)$.

On construit ensuite une partition $\Pi = \{\Pi_1, \ldots, \Pi_n\}$ de $V$ telle que pour tout $T_1, T_2 \in V$, si $\{T_1, T_2\} \subseteq \Pi_i$ alors $(T_1, T_2) \notin A$. Finalement, on crée une horloge $x_i$ pour chaque $\Pi_i$ et on définit $hor(T) = x_i$ pour tout $T \in \Pi_i$.

Une horloge additionnelle $x_0$ est utilisée pour les transitions $T$ telles que $borne(T) = 0$.

Donc, trouver le nombre minimal d'horloges consiste à trouver le plus petit $n$ tel que $\Pi = \{\Pi_1, \ldots, \Pi_n\}$ est une partition qui satisfait la propriété ci-dessus, ce qui est équivalent au problème de coloriage d'un graphe non orienté avec un nombre minimal de couleurs, lequel est NP-complet [AHU74, PS82].

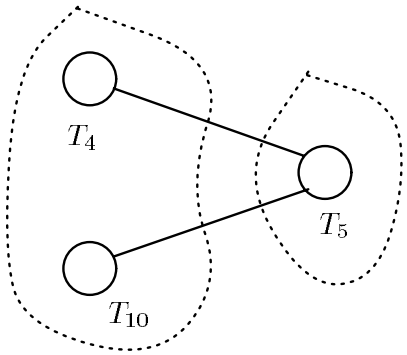
L'algorithme utilisé pour construire une partition $\Pi$ est le suivant :

**begin**
  **forall** $v \in V$
    $marque[v] := false$
    $interdit[v] := \emptyset$
  **end**
  **forall** $v \in V$
    **if** $\neg marque[v]$ **then** $coloriage(v)$
  **end**
  **let** $\Pi_c = \{v \mid couleur[v] = c\}$
  $\Pi := \bigcup\{\Pi_c\}$
**end**

**proc** coloriage(v)
**begin**
  $marque[v] := true$
  **forall** $v' \in V : (v, v') \in A$
    **if** $\neg marque[v]$ **then** $coloriage(v)$
  **end**
  **let** $c \notin interdit[v]$
  $couleur[v] := c$
  **forall** $v' \in V : (v, v') \in A$
    $interdit[v'] := c$
  **end**
**end**

Bien entendu, la partition construite par l'algorithme n'est pas en général optimal. Cependant, les résultats obtenus en pratique sont satisfaisants.

*Exemple.* Considérons le réseau du contrôleur de température illustré par la figure 7.2. Nous avons que la fonction *hor* doit satisfaire $hor(T_4) \neq hor(T_5)$ et $hor(T_{10}) \neq hor(T_5)$. Par conséquent, le graphe construit est le suivant :




La partition II est $\{\{T_4, T_{10}\}, T_5\}$. Nous avons donc $hor(T_4) = hor(T_{10}) = x_1$, $hor(T_5) = x_2$. De plus, pour $T \in \{T_6, T_7, T_8, T_{11}, T_{12}, T_{13}\}$, nous avons $hor(T) = x_0$. □

## 7.5 Conclusions du chapitre

Nous avons présenté dans ce chapitre une méthode de construction d'un graphe temporisé, basée sur la génération d'une structure intermédiaire, à partir de laquelle le graphe peut être construit efficacement.

La structure intermédiaire utilisée est inspirée des réseaux de Petri, mais elle diffère de ceux-ci au sens où toute transition a un ensemble de places d'entrée obligatoires, qui déterminent si la transition est franchissable à partir d'un marquage, ainsi qu'un ensemble de places d'entrée optionnelles, dont le marquage n'est pas nécessaire pour la franchir, mais qui sont également vidées lorsque la transition est franchie.

La distinction entre places d'entrée obligatoires et places d'entrée optionnelles permet d'enlever la marque d'une place mais sans empêcher le franchissement de la transition si la place n'est pas marquée. L'avantage principal de ce mécanisme est que le réseau obtenu est compacte.

La méthode présentée dans ce chapitre a été implémentée et le programme développé a environ 10000 lignes de code C. Les performances sont illustrées dans le chapitre 8.

# Chapitre 8

# Applications

L'objectif de ce chapitre est illustrer l'utilisation de la méthode proposée pour spécifier et vérifier des applications temps-réels.

## 8.1 Un système téléphonique

L'exemple traité dans cette section est un système téléphonique inspiré du système présenté dans [Das85, GK91]. Ce système est une simplification d'un système téléphonique réel, néanmoins il comporte des propriétés temps-réel intéressantes.

Le système est composé de plusieurs postes dont les communications sont gérées par un petit central téléphonique. L'objectif de cette section est donner une modélisation en ATP du central téléphonique et vérifier qu'elle satisfait les propriétés désirées.

### 8.1.1 Spécification en ATP

Le central téléphonique est composée de trois sous-systèmes :

- l'unité de *contrôle* (UC),
- l'unité de *gestion du numéro* (UGN), et
- l'unité de *gestion de la sonnerie* (UGS).

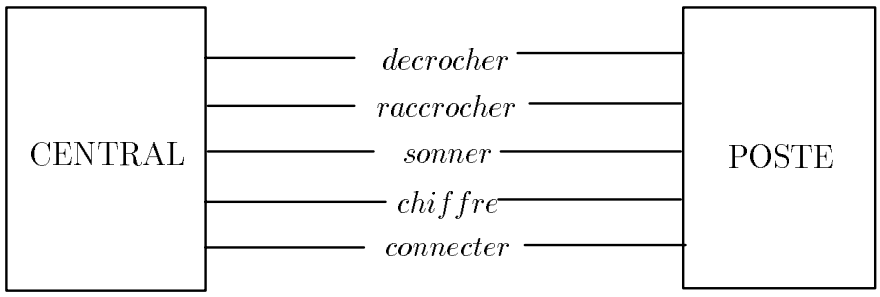
La figure 8.1 illustre la composition du central téléphonique.

#### L'unité de contrôle

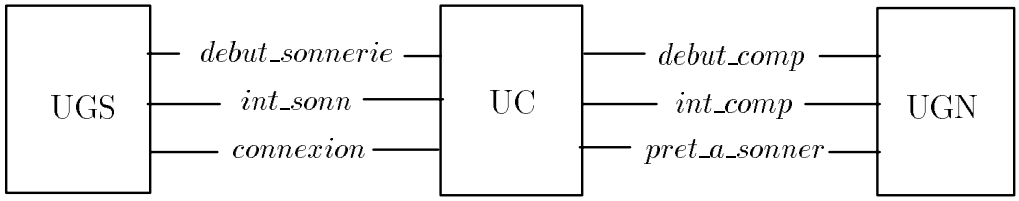
La tâche de l'unité de contrôle consiste à accepter les demandes de communication provenant des postes téléphoniques. Une demande se produit lorsqu'un poste est décroché. Nous supposons que le central ne traite qu'un appel à la fois et qu'une connexion ne peut être interrompue par un autre appel.

Initialement, l'unité de contrôle attend une demande de communication. Lorsqu'un poste est décroché, le central est pret à gérer la composition du numéro appelé. Cette tâche est réalisée par l'unité de gestion du numéro.

L'unité de contrôle et l'unité de gestion du numéro communiquent au moyen des actions suivantes :




Liens entre le central téléphonique et un poste




Composition du central téléphonique

Figure 8.1: Système téléphonique.

- *debut_comp* indique le début de la composition du numéro,
- *int_comp* indique que la composition du numéro a été interrompue,
- *pret_a_sonner* indique que la composition du numéro a terminé et que le central est pret à provoquer la sonnerie du poste appelé.

L'unité de contrôle et l'unité de gestion de la sonnerie communiquent au moyen des actions suivants :

- *debut_sonnerie* indique le début de la sonnerie,
- *int_sonnerie* indique que la sonnerie a terminé, mais la connexion est annulée,
- *connexion* indique que la connexion est établie.

La durée d'une communication est limitée. Si l'utilisateur n'a pas raccroché avant $t_m$ secondes, la communication est interrompue.

Le comportement de l'unité de contrôle est le suivant :

$$
\begin{aligned}
P0 &= \mathsf{d}(\mathit{decrocher})\, P1 \\
P1 &= \mathit{debut\_comp}!\, P2 \\
P2 &= \mathsf{d}(\mathit{int\_comp}?)\, P0 + \mathsf{d}(\mathit{pret\_a\_sonner}?)\, P3 \\
P3 &= \mathit{debut\_sonnerie}!\, P4 \\
P4 &= \mathsf{d}(\mathit{int\_sonnerie}?)\, P0 + \mathsf{d}(\mathit{connexion}?)\, P5 \\
P5 &= \mathsf{d}(\mathit{raccrocher})\, P0\ \mathsf{timeout}(t_m)\ \mathit{int\_comm}\, P0
\end{aligned}
$$

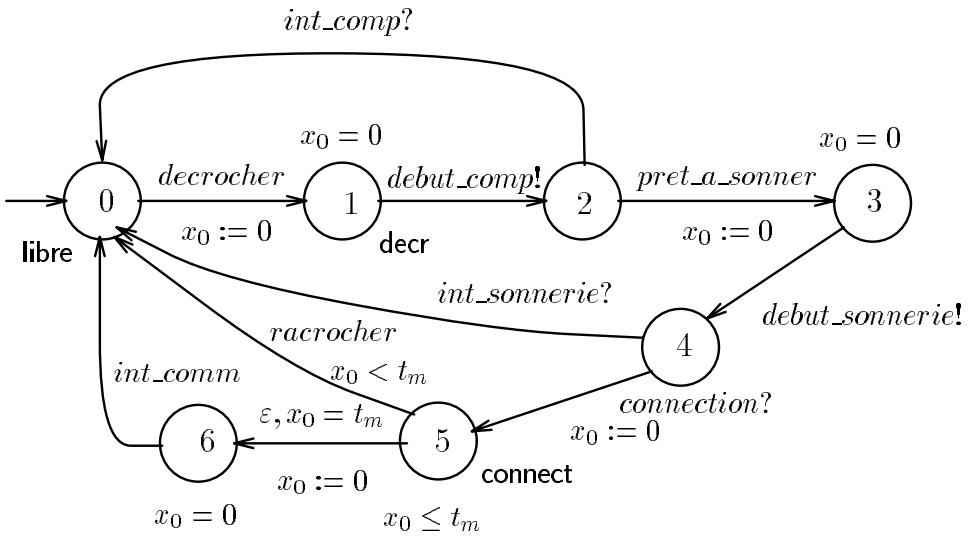



Figure 8.2: L'unité de contrôle.

La figure 8.2 illustre le graphe temporisé de l'unité de contrôle. Nous avons étiqueté les sommets avec des propositions :

- La proposition libre associée au sommet 0 indique que le central est libre à ce sommet.
- La proposition decr indique qu'un poste vient d'être décroché.
- La proposition connect indique qu'une communication a été établie.

### L'unité de gestion du numéro

La tâche de l'unité de gestion du numéro consiste à gérer la composition du numéro appelé. Nous supposons que les numéros ont quatre chiffres et que la composition d'un numéro doit respecter les contraintes suivantes :

1. L'utilisateur doit composer le premier chiffre au plus tard $t_p$ secondes après la réception de la tonalité.
2. Le délai entre deux chiffres successifs ne doit pas dépasser $t_s$ secondes.
3. La composition du numéro doit prendre au plus $t_c$ secondes.

La composition d'un numéro est interrompue si ces contraintes temporelles ne sont pas respectées ou si l'utilisateur raccroche le poste.

Le comportement de l'unité de gestion du numéro est modélisé par le programme suivant :

$$
\begin{aligned}
D0 &= \mathsf{d}(\mathit{debut\_comp}?)\, D1 \\
D1 &= D2\ \mathsf{watchdog}[\mathit{int\_comp}?, \mathit{pret\_a\_sonner}?, \mathit{raccrocher}](t_c)\ D7 \\
D2 &= (\mathsf{d}(\mathit{chiffre1})\, D3 + \mathsf{d}(\mathit{raccrocher})\, D7)\ \mathsf{timeout}(t_p)\ D7
\end{aligned}
$$

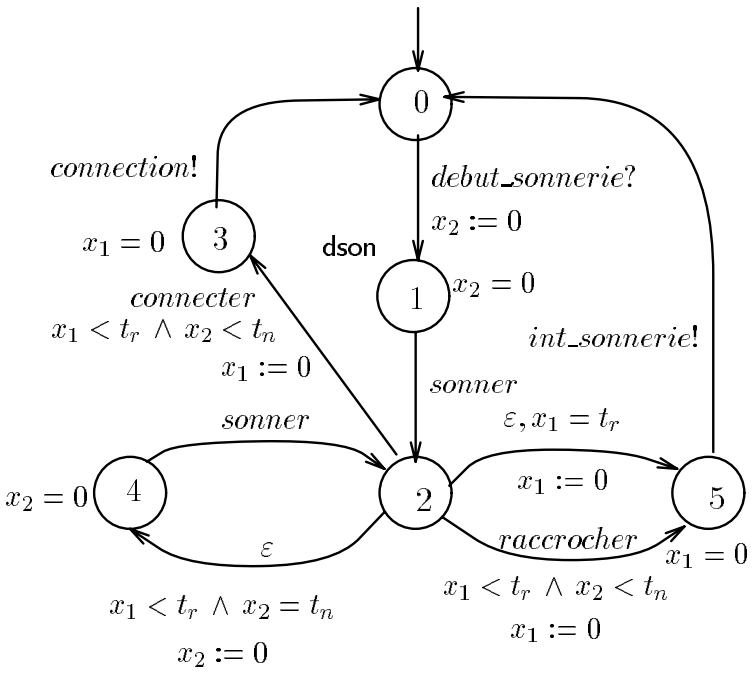



Figure 8.3: L'unité de gestion de la sonnerie.

$$
\begin{aligned}
D3 &= (\mathsf{d}(chiffre2)\, D4 + \mathsf{d}(raccrocher)\, D7)\ \mathsf{timeout}(t_s)\ D7 \\
D4 &= (\mathsf{d}(chiffre3)\, D5 + \mathsf{d}(raccrocher)\, D7)\ \mathsf{timeout}(t_s)\ D7 \\
D5 &= (\mathsf{d}(chiffre4)\, D6 + \mathsf{d}(raccrocher)\, D7)\ \mathsf{timeout}(t_s)\ D7 \\
D6 &= pret\_a\_sonner!\, D0 \\
D7 &= int\_comp!\, D0
\end{aligned}
$$

Le graphe temporisé de l'unité de gestion du numéro est illustré par la figure 8.4. Nous avons étiqueté les sommets avec les propositions chiff1, chiff2, chiff3 et chiff4 qui indiquent respectivement que le premier, deuxième, troisième et quatrième chiffre a été composé.

### L'unité de gestion de la sonnerie

La tâche de l'unité de gestion de la sonnerie consiste à provoquer une sonnerie du poste appelé toutes les $t_n$ secondes. Si le poste appelé n'a pas répondu au bout de $t_r$ secondes, l'appel est annulé.

Le comportement de l'unité de gestion de la sonnerie est le suivant :

$$
\begin{aligned}
R0 &= \mathsf{d}(debut\_sonnerie?)\, sonner\, R1 \\
R1 &= R3\ \mathsf{watchdog}[connecter, raccrocher](t_r)\ R4 \\
R2 &= (\mathsf{d}(connecter)\, R3 + \mathsf{d}(raccrocher)\, R4)\ \mathsf{timeout}(t_n)\ sonner\, R2 \\
R3 &= connexion!\, R0 \\
R4 &= int\_sonnerie!\, R0
\end{aligned}
$$

Le graphe temporisé de l'unité de gestion de la sonnerie est illustré par la figure 8.3. Nous avons étiqueté le sommet 1 avec la proposition dson qui indique le début de la sonnerie.

### Le central téléphonique

Le système complet est obtenu par la composition parallèle des trois sous-systèmes définis ci-dessus :

$$CT = \textsf{restrict}\ A\ \textsf{in}\ P0 \,||\, D0 \,||\, R0$$

où

$$\begin{aligned} A \;=\; & \{debut\_comp\#, int\_comp\#, pret\_a\_sonner\#, debut\_sonnerie\#, \\ & int\_sonnerie\#, connexion\# \mid \# \in \{?,!\}\} \end{aligned}$$

La fonction de communication est telle que $a! \mid a? = a$.

## 8.1.2 Vérification

Nous désirons vérifier les propriétés suivantes :

1. Le système est bien temporisé : $\forall\Box\exists\Diamond_{=1} true$.
2. L'utilisateur doit raccrocher au plus tard $t_m$ secondes après que la communication a été établie. Si ce n'est pas le cas, le central doit interrompre la communication. Autrement dit, le central téléphonique est libre au plus tard $t_m$ secondes après la connexion :

   $$\textsf{connect} \Rightarrow \forall\Diamond_{\leq t_m} \textsf{libre}$$

3. Le central est de nouveau libre au plus tard $t_c + t_r + t_m$ secondes après la demande de communication :

   $$\textsf{libre} \Rightarrow \forall\Box(\textsf{decr} \Rightarrow \forall\Diamond_{\leq t_c+t_r+t_m} \textsf{libre})$$

4. Inévitablement le premier chiffre est composé ou la composition du numéro est interrompue au plus tard $t_p$ secondes après la réception de la tonalité :

   $$\textsf{decr} \Rightarrow \forall\Diamond_{\leq t_p}(\textsf{chiff1} \vee \textsf{libre})$$

5. Inévitablement le numéro complet est composé ou la composition est interrompue au plus tard $t_c$ secondes après la réception de la tonalité :

   $$\textsf{decr} \Rightarrow \forall\Diamond_{\leq t_c}(\textsf{chiff4} \vee \textsf{libre})$$

6. Si le poste appelé n'a pas répondu au plus tard $t_r$ secondes après le début de la sonnerie, alors la communication est annulée :

   $$\textsf{dson} \Rightarrow \forall\Diamond_{\leq t_r}(\textsf{connect} \vee \textsf{libre})$$

### 8.1.3 Résultats

Nous avons expérimenté les outils de compilation et de vérification avec le programme ATP qui décrit le comportement du central téléphonique. Les résultats suivants ont été obtenus sur une machine SUN4 Sparc Station avec 16MB de mémoire.

#### Construction du graphe temporisé

Le graphe temporisé construit par le compilateur a 36 sommets, 52 arcs et 6 horloges. Le temps de compilation a été de 0.15s.

Notons que les figures 8.2, 8.3 et 8.4, montrent que 5 horloges suffisent pour modéliser les contraintes temporelles du système. Cependant, le compilateur en génère 6, l'horloge additionnelle est introduite afin d'exprimer les contraintes temporelles imposées par les actions immédiates.

#### Vérification des propriétés

Nous avons vérifié les propriétés pour différentes valeurs des paramètres $t_m$, $t_c$, $t_s$, $t_p$, $t_n$ et $t_r$. Les résultats obtenus avec l'outil KRONOS [NSY92, NOSY92, OY92] sont illustrés par les tableaux suivants.

| Paramètres | | | | | | |
|---|---|---|---|---|---|---|
| $t_m = 180$ | $t_c = 40$ | $t_p = 20$ | $t_s = 10$ | $t_r = 60$ | $t_n = 3$ | |
| Formule | (1) | (2) | (3) | (4) | (5) | (6) |
| Nombre d'itérations | 7 | 14 | 53 | 11 | 8 | 48 |
| Temps (en secondes) | 2.02 | 4.14 | 73.11 | 2.60 | 2.02 | 51.40 |

| Paramètres | | | | | | |
|---|---|---|---|---|---|---|
| $t_m = 240$ | $t_c = 120$ | $t_p = 60$ | $t_s = 25$ | $t_r = 90$ | $t_n = 9$ | |
| Formule | (1) | (2) | (3) | (4) | (5) | (6) |
| Nombre d'itérations | 7 | 14 | 33 | 11 | 8 | 28 |
| Temps (en secondes) | 2.01 | 4.15 | 25.21 | 2.56 | 2.06 | 14.07 |

| Paramètres | | | | | | |
|---|---|---|---|---|---|---|
| $t_m = 360$ | $t_c = 120$ | $t_p = 60$ | $t_s = 25$ | $t_r = 120$ | $t_n = 9$ | |
| Formule | (1) | (2) | (3) | (4) | (5) | (6) |
| Nombre d'itérations | 7 | 14 | 40 | 11 | 8 | 36 |
| Temps (en secondes) | 2.01 | 4.16 | 41.00 | 2.54 | 2.05 | 25.11 |

Dans tous les cas, le résultat produit par l'évaluateur est le prédicat *true*, ce qui indique que tous les états du modèle du graphe temporisé satisfont la formule considérée.

Notons que le nombre d'itérations et le temps d'exécution pour l'évaluation des certaines formules n'est pas sensible au changement des valeurs des paramètres. Ceci est dû au fait que la validité de ces formules ne dépend pas des valeurs relatives des délais, ce qui est exploité par la méthode d'évaluation symbolique présentée dans le chapitre 4.

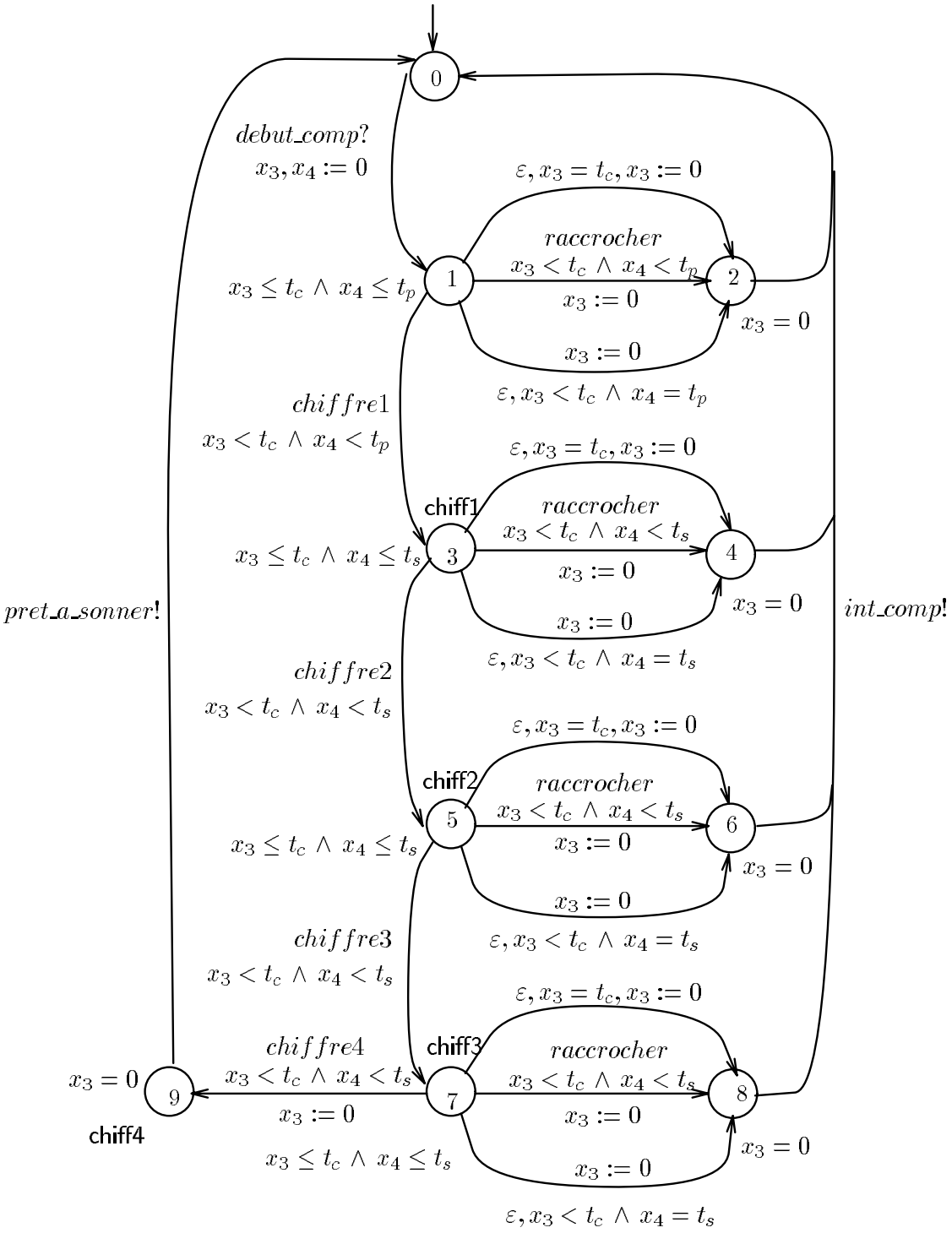



Figure 8.4: L'unité de gestion du numéro.

## 8.2 Le protocole CSMA/CD

Considérons un réseau de stations qui communiquent par diffusion (broadcasting) des messages à travers une unique ligne de communication appelée *canal d'accès multiple*. Le problème principal dans un tel réseau est l'assignation de l'usage du canal lorsque plusieurs stations veulent envoyer des messages simultanément.

L'un des protocoles connus pour résoudre ce problème est le *Carrier Sense, Multiple Access with Collision Detection* ou CSMA/CD [Tan89, IEE85].

Une description en CCS qui ne tient pas compte des contraintes temporelles du protocole, est donnée dans [Par86]. Celles-ci ont été considérées dans [Han91], mais dans un modèle de temps discret.

Notre objectif est de modéliser les contraintes temporelles du protocole dans un modèle de temps dense.

### 8.2.1 Description informelle du protocole

Le fonctionnement du protocole est le suivant. Lorsqu'une station a un message à envoyer, elle écoute le canal. Si celui-ci est *libre*, i.e., aucune autre n'a l'usage du canal, alors la station commence à envoyer son message. Dans le cas contraire, si le canal est *occupé*, la station attend un certain temps au hasard pour répéter l'opération.

Une *collision* a lieu lorsque plusieurs stations transmettent simultanément. Une collision est détectée par toutes les stations concernées, lesquelles cessent immédiatement de transmettre et attendent un certain temps au hasard pour recommencer. En cas de collision, tous les messages se perdent.

#### Le canal

Le délai de propagation sur le canal joue un rôle important dans le fonctionnement du protocole. Supposons qu'une station reçoive un message juste après qu'une autre ait commencé à transmettre. Si la première écoute le canal avant que le message envoyé par l'autre soit arrivé jusqu'elle, alors elle trouvera le canal libre et commencera à transmettre. Cette situation provoque une collision.

Soit $\sigma$ le temps de propagation d'un signal entre les deux stations plus éloignées. Supposons qu'au temps $t_0$ une station commence à transmettre. Alors, dans l'intervalle de temps $[t_0, t_0 + \sigma)$ il est possible qu'une autre station, en écoutant le canal libre, transmette et provoque une collision. Après l'instant $t_0 + \sigma$ toutes les stations vont trouver le canal occupé pendant le temps de transmission du message courant. Par conséquent, $\sigma$ est le temps maximum pendant lequel le canal peut être trouvé libre par une station après qu'une autre ait commencé à transmettre.

#### Détection d'une collision

D'après la discussion ci-dessus, on pourrait penser qu'une station qui n'ait pas détecté une collision dans un temps égal à $\sigma$, peut être sûre qu'aucune autre station n'a interféré. Mais, cette conclusion est erronée car le bruit de la collision peut prendre un temps $\sigma$ pour se propager. Dans le pire cas, une collision ne sera pas détectée avant un temps de $2\sigma$.

Lorsqu'une collision est détectée, chacune des stations $S_i$ concernées attend un temps $t_i$ dans l'intervalle $[0, 2\sigma]$ pour retransmettre. L'intervalle peut être élargi selon le nombre de collisions. En général, après $n$ collisions, chaque station attend un temps $t_i$ entre 0 et $2^n\sigma$. De plus, si

le nombre des collisions devient trop élevé, e.g., 16 dans la norme standard 802.3 [IEE85], une erreur peut être rapportée.

#### Temps de transmission

Nous supposons que le temps de transmission d'un message, noté $\lambda$, est fixe. Par exemple, pour une ligne de communication de type Ethernet de 10Mbps avec $2\sigma$ égal à 51.2$\mu$s, nous avons un temps de transmission $\lambda$ d'environ 782$\mu$s pour un message dont la taille est de 1024 octets.

### 8.2.2 Description formelle du protocole

Nous faisons l'hypothèse que le canal de communication est bidirectionnel et sans erreur, i.e., sans perte ni duplication, et se comporte comme un tampon de taille 1. Nous ne modélisons que la transmission des messages.

#### Les émetteurs

Le système est composé de plusieurs stations, appelées aussi émetteurs, et de la ligne de communication. Les actions *ready* et *busy* indiquent respectivement que le canal est libre ou occupé. L'action *begin* représente le commencement d'une transmission. La fin est signalée par l'action *end*. Une collision est modélisée par l'action *cd*. Des indices sont utilisés pour distinguer les différentes stations.

La spécification de l'émetteur $i$ est la suivante :

$$
\begin{aligned}
S_i^0 &\stackrel{\text{def}}{=} \mathsf{d}(send_i?)S_i^1 + \mathsf{d}(cd_i?)S_i^0 \\
S_i^1 &\stackrel{\text{def}}{=} ready_i?S_i^2 + busy_i?S_i^4 + cd_i?S_i^4 \\
S_i^2 &\stackrel{\text{def}}{=} begin_i!S_i^3 \\
S_i^3 &\stackrel{\text{def}}{=} \mathsf{d}(cd_i?)S_i^4\ \mathsf{timeout}(\lambda_i)\ end_i!S_i^0 \\
S_i^4 &\stackrel{\text{def}}{=} S_i^5\ \mathsf{watchdog}[\tau_i](2\sigma)\ \tau_i S_i^1 \\
S_i^5 &\stackrel{\text{def}}{=} \mathsf{d}(\tau_i)S_i^1 + \mathsf{d}(cd_i?)S_i^5
\end{aligned}
$$

où l'action $\tau_i$ représente la décision de l'émetteur $i$ de retransmettre.

Le graphe temporisé de l'émetteur est illustré par la figure 8.5.

#### Le canal de communication

Le comportement du canal de communication est le suivant. Initialement, le canal est libre et peut accepter un message de tout émetteur. Lorsque l'un des émetteurs commence à transmettre, le canal est détecté libre pour tous les autres pendant un temps inférieur au délai de propagation $\sigma$. Après $\sigma$, le canal est occupé pendant toute la transmission du message. Une collision se produit lorsqu'une autre station transmet avant $\sigma$. Le canal signale la collision en un temps inférieur ou égal à $\sigma$.

Le système d'équations qui définit le comportement du canal est le suivant :

$$
\begin{aligned}
M^0 &\stackrel{\text{def}}{=} \mathsf{d}(begin?)M^1 + \mathsf{d}(ready!)M^0 \\
M^1 &\stackrel{\text{def}}{=} M^3\ \mathsf{watchdog}[begin](\sigma)\ M^2
\end{aligned}
$$

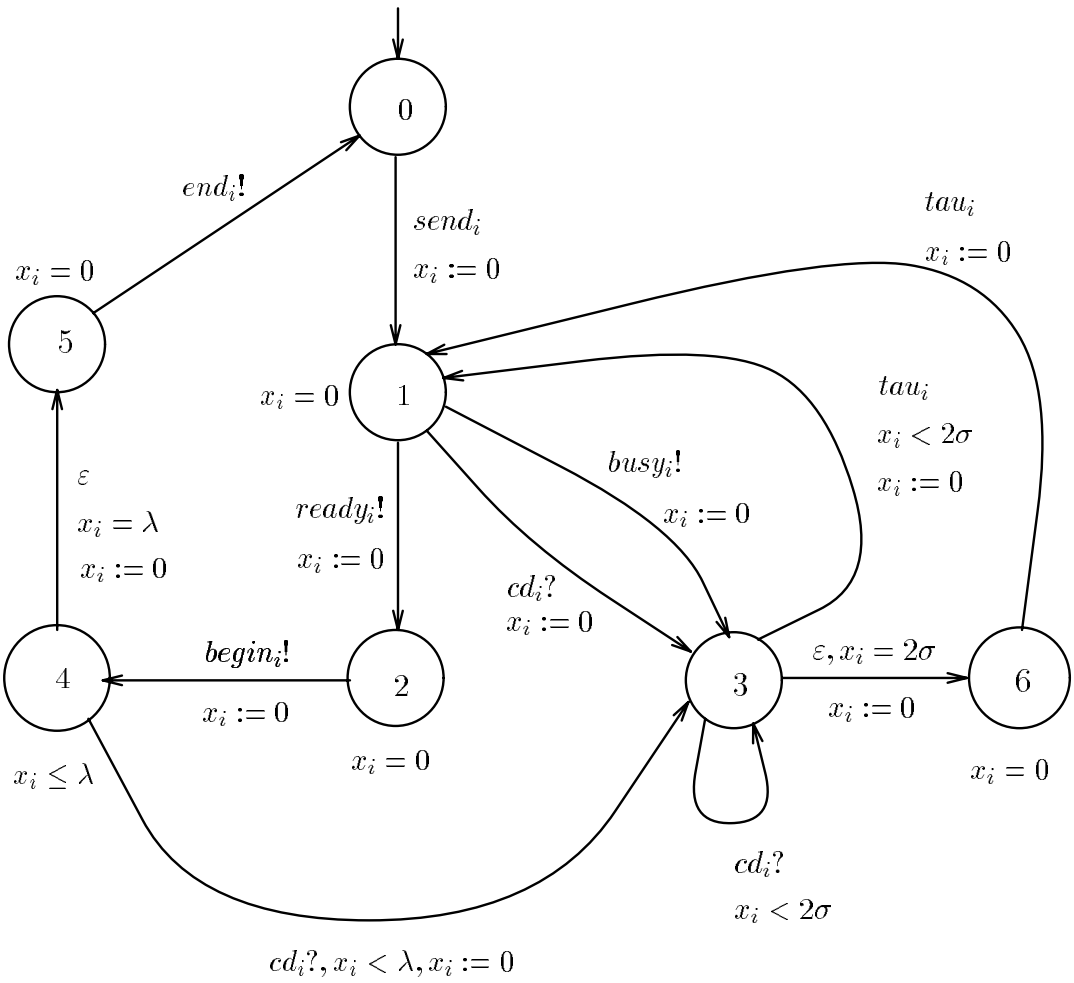



Figure 8.5: Graphe temporisé de l'émetteur.

$$
\begin{aligned}
M^2 &\stackrel{\text{def}}{=} \mathsf{d}(busy!)M^2 + \mathsf{d}(end?)M^0 \\
M^3 &\stackrel{\text{def}}{=} \mathsf{d}(ready!)M^3 + \mathsf{d}(begin!)M^4 \\
M^4 &\stackrel{\text{def}}{=} \mathsf{d}(cd!)M^0 \ \mathsf{timeout}(\sigma)\ cd!M^0
\end{aligned}
$$

Le graphe temporisé du canal de communication est illustré par la figure 8.6.

### Le système complet

Le système complet est obtenu par la composition parallèle des processus définis ci-dessus, de la façon suivante :

$$CSMA \stackrel{\text{def}}{=} \mathsf{restrict}\ A\ \mathsf{in}\ S_1^0 \| \ldots \| S_n^0 \| M^0$$

où

$$
\begin{aligned}
A \ = \ & \{begin_i!, end_i!, cd_i!, ready_i?, busy_i?, cd_i? \mid i = 1, \ldots, n\} \\
& \cup \{begin?, end?, ready!, busy!, nb!, ne!, nb?, ne?\}
\end{aligned}
$$

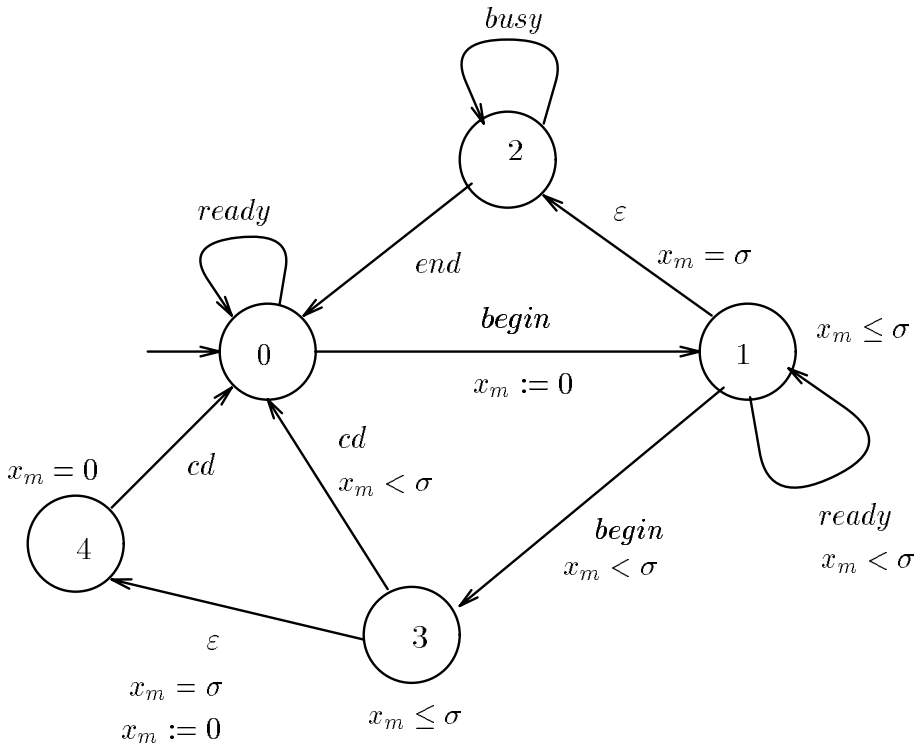



Figure 8.6: Graphe temporisé du canal de communication.

La fonction de communication est définie de la façon suivante :

- Pour $a \in \{begin, end, ready, busy\}$,

$$a \quad = \quad a_i? \mid a!$$

- Pour $cd$,

$$cd \quad = \quad cd_1? \mid \ldots \mid cd_n? \mid cd!$$

Une collision est donc détectée simultanément par tous les émetteurs. D'après la description du canal, ceci a lieu à un certain instant $t$ dans l'intervalle $[0, \sigma]$.

### 8.2.3 Résultats

Nous avons expérimenté les outil de compilation et de vérification avec le programme ATP décrit ci-dessus pour deux émetteurs et valeurs des paramètres $\sigma = 26\mu$s et $\lambda = 782\mu$s.

Les résultats suivants ont été obtenus sur une machine SUN4 Sparc Station avec 16MB de mémoire.

Le graphe temporisé construit par le compilateur 133 sommets, 250 arcs et 4 horloges. Le temps de compilation a été de 5s.

Les propriétés vérifiées ont été les suivantes :

1. Afin de vérifier que le graphe temporisé construit été bien temporisé, nous avons évalué la formule 4 :

   $$\mathsf{init} \Rightarrow \forall\Box\exists\Diamond_{=1} true$$

   où init est une proposition qui vaut vraie au sommet initial du graphe.

   Les calculs des points fixes correspondants aux opérateurs $\exists\Diamond$ et $\forall\Box$ ont demandés 9 et 7 itérations, respectivement, et un temps total de calcul de 18 secondes.

2. Nous avons vérifier que lorsque les deux émetteurs ont l'usage du canal et commencent à transmettre leurs messages, une collision est inévitablement détectée au plus tard $\sigma\mu$s après. Cette propriété s'exprime en TCTL par la formule :

   $$\mathsf{init} \Rightarrow \forall\Box((\mathsf{trans}_1 \wedge \mathsf{trans}_2) \Rightarrow \forall\Diamond_{\leq\sigma}\mathsf{cd})$$

   La proposition $\mathsf{trans}_i$ vaut vraie dans les états où l'émetteurs $i$ est en train de transmettre le message. La proposition cd vaut vraie dans les états où une collision a été détectée.

   Les calculs des points fixes correspondants aux opérateurs $\forall\Diamond_{\leq\sigma}$ et $\forall\Box$ ont demandés 9 et 1 itérations, respectivement, et un temps total de calcul de 16.5 secondes.

3. Finalement, nous avons que le système satisfait la formule suivante :

   $$\mathsf{init} \Rightarrow \forall\Box(\mathsf{begin}_i \Rightarrow \exists\Diamond(\neg\mathsf{cd}\,\exists\mathcal{U}_{=\sigma}\,(\neg\mathsf{cd}\,\forall\mathcal{U}_{=\lambda-\sigma}\,\mathsf{end}_i)))$$

   qui signifie que lorsqu'un émetteurs commence à transmettre, il existe une transmission réussie, c'est-à-dire, une transmission où aucune collision ne se produit et le message est transmis en un temps égal à $\lambda$.

   Les calculs des points fixes correspondants aux opérateurs $\forall\Diamond_{=\lambda-\sigma}$, $\exists\mathcal{U}_{\leq\sigma}$ , $\exists\Diamond$ et $\forall\Box$, ont demandés 9, 9, 10 et 6 itérations, respectivement, et un temps total de calcul de 61 secondes.

Dans tous les cas, le résultat produit par l'évaluateur est le prédicat *true*, ce qui indique que tous les états du modèle du graphe temporisé satisfont la formule considérée.

# Chapitre 9

# Conclusions

## Bilan

Ce travail a été motivé par l'intérêt de développer des méthodes d'analyse de systèmes temps-réel. Pour ce faire, nous nous sommes inspirés de l'approche mise en pratique dans le cas de langages synchrones [BdS91, HCRP91, Mar90, GGBM91]. Celle-ci consiste à compiler les programmes vers des automates étendus à partir desquels il est possible de vérifier des propriétés.

### La méthode de compilation

Nous avons choisi comme langage de spécification l'algèbre ATP [NRSV90, NS90, NS91, NSY91, NSY92, Nic92], une extension des algèbres de processus communicants. ATP comporte des opérateurs pour exprimer les contraintes temporelles.

Nous avons utilisé ATP pour décrire des applications temps-réel significatives, comme le central téléphonique [Das85, GK91] et le protocole CSMA/CD [Tan89, IEE85] permettant d'arbitrer l'accès au canal d'un réseau de stations.

Cette expérience montre que les constructeurs du langage permettent une description naturelle des mécanismes significatifs des applications temps-réel, comme les retards, les timeouts et les watchdogs.

L'un des résultats de ce travail a été de montrer que les comportements décrits par une classe importante de programmes ATP peuvent être représentés par des automates étendus finis. La taille de ces automates et la complexité de l'algorithme de construction sont indépendants des valeurs des paramètres délais.

Dans notre cas, le temps écoulé entre deux transitions successives d'un de ces automates peut être arbitrairement petit. Ceci est une différence importante avec l'approche synchrone dans laquelle les transitions des automates générés par les compilateurs correspondent aux ticks d'une horloge.

Autrement dit, le mode d'exécution sous-jacent est "pas-à-pas" dans le cas synchrone, tandis que la sémantique des graphes temporisés induit naturellement un mode d'exécution "événementiel" dû à l'absence d'une granularité déterminée pour la progression du temps.

Nous avons montré que la compilation de programmes ATP peut être faite efficacement. En effet, nous avons développé une technique de compilation basée sur la génération d'une structure intermédiaire à partir de laquelle le graphe temporisé d'un programme ATP est construit de manière efficace.

Ce codage intermédiaire est indépendant du langage source, ce qui fait envisager son utilisation comme langage intermédiaire pour la compilation de langages de programmation temps-réel autres qu'ATP.

### La vérification de propriétés

Le second résultat de ce travail concerne la vérification de propriétés temps-réel sur les graphes temporisés. Nous avons développé une méthode de model-checking symbolique pour la logique TCTL, fondée sur la caractérisation des formules par point fixe. Dans ce cas, nous nous sommes inspirés des résultats connus pour la logique CTL, dont les opérateurs temporels s'expriment en terme de points fixes de fonctionnelles monotones à l'aide d'un opérateur "état suivant".

Nous avons défini un opérateur "état suivant" pour un domaine temporel continu, qui est fortement lié au mode d'exécution "événementiel" des systèmes temps-réel asynchrones.

Les résultats théoriques plus une méthode de représentation symbolique des ensembles d'états, ont permis la mise en pratique efficace de l'algorithme de model-checking dans l'outil KRONOS [NSY92, NOSY92, OY92].

Nous avons utilisé KRONOS pour vérifier des propriétés des applications temps-réel décrites dans le langage ATP avec des bonnes performances. Cette expérience montre que les techniques développées peuvent être appliquées à la vérification de systèmes temps-réel importants.

D'autres méthodes de vérification ont été proposées dans [ACD+92b, ACD+92a, AIKY92, CDCT92]. Toutefois, ces méthodes ont des différences importantes avec celle présentée dans ce travail.

Les méthodes développées dans [AIKY92, CDCT92] sont des extensions de l'outil COSPAN [HK89]. Dans [AIKY92] le système et la propriété à vérifier sont donnés par des $\omega$-automates étendus avec des contraintes temporelles. Le problème de la vérification est posé comme un problème d'inclusion de langages. L'algorithme consiste tout d'abord à vérifier si le système obtenu en ignorant les délais satisfait la propriété. En cas d'échec, on vérifie si la séquence qui viole la propriété est consistante avec les contraintes temporelles. Si ce n'est pas le cas, on enlève la séquence et on itère la procédure. L'approche suivie dans [CDCT92] consiste à augmenter la description du système avec un "moniteur" généré automatiquement. Ce moniteur exclue les séquences inconsistantes avec les contraintes temporelles.

L'algorithme développé dans [ACD90] repose sur la construction d'un modèle fini, appelé *graphe de régions*, réduit modulo une relation d'équivalence entre les états. Cette relation garantit que deux états équivalents ne peuvent pas être distingués par les formules de TCTL. La méthode proposée dans [ACD+92b] consiste à appliquer un algorithme de minimisation [BFH90] afin de réduire la taille du graphe de régions. Le modèle minimal ainsi obtenu ne peut pas être utilisé directement pour vérifier des formules de TCTL. En effet, ce modèle ne contient pas l'information nécessaire pour déterminer si une séquence de transitions est possible dans un intervalle de temps donné. Afin de résoudre ce problème, le graphe minimal est raffiné en "découpant" les régions, i.e., les classes d'équivalences, autant que nécessaire pour évaluer la formule. Une implémentation de cet algorithme est présentée dans [ACD+92a].

## Perspectives

Un prolongement envisageable du travail consiste à étendre les modèles et les techniques symboliques proposées, afin de pouvoir spécifier et vérifier des systèmes hybrides. Les *systèmes*

*hybrides* comportent des composantes discrètes et continues. En général, les composantes continues représentent un environnement physique qui évolue continûment, tandis que les composantes discrètes correspondent à des contrôleurs digitaux qui interagissent avec l'environnement.

Deux directions semblent intéressantes. D'une part, il est nécessaire d'identifier des sous-classes décidables de systèmes hybrides et les méthodes de décision associées. Cette approche a été suivie dans [KPSY92] où une solution algorithmique est proposée au problème d'accessibilité pour des systèmes hybrides particuliers appelés *graphes d'intégration*. De plus, dans [NOSY92] l'outil KRONOS a été utilisé pour vérifier des propriétés de systèmes hybrides pour lesquels les lois d'évolution sont exprimées par des contraintes linéaires. D'autre part, il est possible d'adopter des solutions approchées par application de techniques d'interprétation abstraite [CC77, CC92]. Cette solution a été suivie par exemple dans [Hal93].

# Bibliographie


[ACD90] R. Alur, C. Courcoubetis, et D. Dill. Model-checking for real-time systems. Dans *Proc. 5ème LICS*, pages 414–425. IEEE Computer Society Press, 1990.

[ACD+92a] A. Alur, C. Courcoubetis, D. Dill, N. Halbwachs, et H. Wong-Toi. An implementation of three algorithms for timing verification based on automata emptiness. Dans *Proc. 13ème IEEE Real-Time Systems Symposium*. IEEE Computer Society Press, 1992.

[ACD+92b] A. Alur, C. Courcoubetis, D. Dill, N. Halbwachs, et H. Wong-Toi. Minimization of timed transition systems. Dans W.R. Cleaveland, éditeur, *Proc. CONCUR 92: Theories of Concurrency*, pages 340–354. LNCS 630, Springer-Verlag, 1992.

[AD90] R. Alur et D. Dill. Automata for modeling real-time systems. Dans *Proc. 17ème ICALP*, pages 322–335. LNCS 443, Springer-Verlag, 1990.

[ADA83] ADA. *The programming language ADA reference manual.* LNCS 155, Springer-Verlag, 1983.

[AHU74] A.V. Aho, J. E. Hopcroft, et J. D. Ullman. *The design and analysis of computer algorithms.* Addison-Wesley, 1974.

[AIKY92] R. Alur, A. Itai, R. Kurshan, et M. Yannakakis. Timing verification by successive approximation. Dans *Proc. 4ème CAV*. Springer-Verlag, 1992.

[Alu91] R. Alur. *Techniques for automatic verification of real-time systems.* PhD thesis, Department of Computer Science, Stanford University, Août 1991.

[ASU86] A.V. Aho, R. Sethi, et J.D. Ullman. *Compilers: Principles, Techniques and Tools.* Addison-Wesley, 1986.

[BB91] A. Benveniste et G. Berry. The synchronous approach to reactive and real-time systems. *Proceedings of the IEEE*, 79:1270–1282, Septembre 1991.

[BC85] G. Berry et L. Cosserat. The ESTEREL synchronous programming language and its mathematical semantics. Dans *Proc. CMU Seminar on Concurrency.* LNCS 197, Springer-Verlag, 1985.

[BCD+90] J.B. Burch, E.M. Clarke, D.Dill, L.J. Hwang, et K.L. McMillan. Symbolic model checking: $10^{20}$ states and beyond. Dans *Proc. 5ème LICS*, pages 428–439. IEEE Computer Society Press, 1990.

[BCG87] G. Berry, P. Couronné, et G. Gonthier. Programmation synchrone des systèmes réactifs : le langage ESTEREL. *Technique et Science Informatiques*, 4:305–316, 1987.

[BD88] P. Boullier et Ph. Deschamp. *Le système* SYNTAX *: Manuel d'utilisation et de mise en oeuvre sous* UNIX., Juillet 1988.

[BdS91] F. Boussinot et R. de Simone. The ESTEREL language. *Proceedings of the IEEE*, 79:1293–1304, Septembre 1991.

[Ber89] G. Berry. Real-time programming: Special purpose or general purpose languages. Dans *IFIP World Computer Congress*, 1989.

[BFH90] A. Bouajjani, J.Cl. Fernandez, et N. Halbwachs. Minimal model generation. Dans E.M. Clarke et R.P. Kurshan, éditeurs, *Proc. 2ème CAV*, pages 197–203. LNCS 531, Springer-Verlag, 1990.

[BG88] G. Berry et G. Gonthier. The synchronous programming language ESTEREL, design, semantics, implementation. Rapport de Recherche 842, INRIA, France, Décembre 1988.

[BHR84] S.D. Brookes, C.A.R. Hoare, et A.W. Roscoe. A Theory of Communicating Sequential Processes. *Journal of the ACM*, 31(3):560–599, 1984.

[BK84] J.A. Bergstra et J.W. Klop. Algebra of Communicating Processes. Rapport de Recherche CS-R8420, Centre for Mathematics and Computer Science, Pays-Bas, 1984.

[BK86] J.A. Bergstra et J.W. Klop. Process algebra : specification and verification in bisimulation semantics. Dans M. Hazewinkel, J.K. Lenstra, et L.G.L.T. Meertens, éditeurs, *Proc. CWI Symp. Mathematics and Computer Science II (CWI Monograph 4)*, pages 61–94. North-Holland, 1986.

[Bry86] R.E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–692, 1986.

[CBM89] O. Coudert, C. Berthet, et J.C. Madre. Verification of synchronous sequential machines based on symbolic execution. Dans J. Sifakis, éditeur, *Proc. 1er CAV*, France, 1989. LNCS 407, Springer-Verlag.

[CC77] P. Cousot et R. Cousot. Abstract interpretation: an unified lattice model for static analysis of programs by construction or approximation of fixpoints. Dans *Proc. 18ème POPL*, Janvier 1977.

[CC92] P. Cousot et R. Cousot. Abstract interpretation and application to logic programs. Rapport technique, École Polytechnique, Juin 1992.

[CDCT92] C. Courcoubetis, D. Dill, M. Chatzaki, et P. Tsounakis. Verification with real-time COSPAN. Dans *Proc. 4ème CAV*. Springer-Verlag, 1992.

[CES86] E.M. Clarke, E.A. Emerson, et A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[CHPP87] P. Caspi, N. Halbwachs, D. Pilaud, et J. Plaice. LUSTRE: a declarative language for programming synchronous systems. Dans *Proc. 18ème POPL*, Janvier 1987.

[CPPS90] Ph. Couronné, J.P. Paris, J. Plaice, et J.B. Saint. The LUSTRE-ESTEREL portable format (version oc3). Rapport technique, École Nationale Supérieure de Mines de Paris, 1990.

[Das85] B. Dasarthy. Timing constraints of real-time systems: constructs for expressing them, methods for validating them. *IEEE Transactions on Software Engineering*, 11(1), Janvier 1985.

[Dil89] D. Dill. Timing assumptions and verification of finite-state concurrent systems. Dans J. Sifakis, éditeur, *Proc. 1er CAV*, France, 1989. LNCS 407, Springer-Verlag.

[EC82] E. A. Emerson et E. Clarke. Using branching-time temporal logic to synthesize synchronization skeletons. *Science of Computer Programming*, 2:241–266, 1982.

[Eme90] E.A. Emerson. Temporal and modal logic. Dans J. van Leeuwen, éditeur, *Handbook of Theoretical Computer Science*, pages 995–1072. Elsevier Science Publishers (North-Holland), 1990.

[Gar89] H. Garavel. Compilation et vérification de programmes LOTOS. Thèse, Université Joseph Fourier, Grenoble, France, 1989.

[GBBG85] P. Le Guernic, A. Benveniste, P. Bournal, et T. Gautier. Signal: a data flow oriented language for signal processing. Rapport de Recherche 246, IRISA, Rennes, France, 1985.

[GGBM91] P. Le Guernic, T. Gautier, M. Le Borgne, et C. Le Maire. Programming real-time applications with Signal. *Proceedings of the IEEE*, 79:1321–1336, Septembre 1991.

[GK91] C. Ghezzi et R.A. Kemmerer. ASTRAL: an assertion language for specifying real-time systems. Dans *Proc. 3ème European Software Engineering Conference*, Italie, Octobre 1991. LNCS 550, Springer-Verlag.

[GW89] R.J. Van Glabbeek et W.P. Weijland. Branching-time and abstraction in bisimulation semantics (extended abstract). Rapport de recherche, Centre for Mathematics and Computer Science, Pays-Bas, Mai 1989.

[Hal93] N. Halbwachs. Delay analysis in synchronous programs. Dans C. Courcoubetis, éditeur, *Proc. 5ème CAV*, Grèce, Juin 1993. LNCS 697.

[Han91] H.A. Hansson. *Time and Probability in Formal Design of Distributed Systems*. PhD thesis, Uppsala University, Sweden, Septembre 1991.

[Har87] D. Harel. Statecharts : a visual approach to complex systems. *Science of Computer Programming*, 8(3):231–275, 1987.

[HCRP91] N. Halbwachs, P. Caspi, P. Raymond, et D. Pilaud. The synchronous dataflow programming language LUSTRE. *Proceedings of the IEEE*, 79:1305–1320, Septembre 1991.

[HK89] Z. Har'El et R. Kurshan. Automatic verification of coordinating systems. Dans J. Sifakis, éditeur, *Proc. 1er CAV*. LNCS 407, Springer-Verlag, 1989.

[HLR92] N. Halbwachs, F. Lagnier, et Ch. Ratel. Programming and verifying real-time systems by means of the synchronous data-flow language LUSTRE. *IEEE Transactions on Software Engineering*, 18(9):785–793, Septembre 1992.

[HNSY92] T.A. Henzinger, X. Nicollin, J. Sifakis, et S. Yovine. Symbolic model-checking for real-time systems. Dans *Proc. 7ème LICS*, pages 394–406. IEEE Computer Society Press, 1992.

[Hoa78] C.A.R Hoare. Communicating Sequential Processes. *Communications of the ACM*, 21(8), 1978.

[HRR91] N. Halbwachs, P. Raymond, et Ch. Ratel. Generating efficient code from data-flow programs. Dans *Proc. Symp. Programming Language Implementation and Logic Programming*, Passau, Août 1991.

[IEE85] IEEE. ANSI/IEEE 802.3, ISO/DIS 8802/3. IEEE Computer Society Press, 1985.

[INM84] INMOS Ltd. OCCAM *programming manual*. Prentice-Hall, 1984.

[ISO85] ISO/TC97/SC21. *Estelle: A formal description technique based on an extended transition model*. ISO, 1985.

[JLHM91] M. Jaffe, N. Leveson, M. Heimdahl, et B. Melhart. Software requirements analysis for real-time process-control systems. *IEEE Transactions on Software Engineering*, 17(3):241–258, 1991.

[KPSY92] Y. Kesten, A. Pnueli, J. Sifakis, et S. Yovine. Integration graphs: a class of decidable hybrid systems. Dans *Workshop on Theory of Hybrid Systems*, Lyngby, Denmark, Octobre 1992.

[Lam83] L. Lamport. What good is temporal logic? Dans *Proc. 9ème IFIP World Computer Congress*, pages 657–668. Elsevier Science Publishers (North-Holland), 1983.

[Mar90] F. Maraninchi. ARGOS : un langage graphique pour la conception, la description et la validation des systèmes réactifs. Thèse, Université Joseph Fourier, Grenoble, France, 1990.

[MB83] M. Menasche et B. Berthomieu. Time petri nets for analyzing and verifying time dependent communication protocols. Dans H. Rudin et C.H. West, éditeurs, *Protocol Specification, Testing and Verification, III*. IFIP, North-Holland, 1983.

[Mil80] R. Milner. A calculus of communicationg systems. Dans *LNCS 92*. Springer-Verlag, 1980.

[Mil89] R. Milner. *Communication and concurrency*. Prentice-Hall, 1989.

[Mou92] L. Mounier. Méthodes de vérification de spécifications comportementales : étude et mise en œuvre. Thèse, Université Joseph Fourier, Grenoble, France, 1992.

[MP81] Z. Manna et A. Pnueli. The temporal framework for concurrent programs. Dans R.S. Boyer et J.S. Moore, éditeurs, *The correctness problem in Computer Science*, pages 215–274. Academic Press, 1981.

[MP89] Z. Manna et A. Pnueli. The anchored version of the temporal framework. Dans J.W. de Bakker, W.-P. de Roever, et G. Rozenberg, éditeurs, *LNCS 354*. Springer-Verlag, 1989.

[Nic92] X. Nicollin. ATP : une algèbre pour la spécification et l'analyse des systèmes temps reel. Thèse, Institut National Polytechnique de Grenoble, France, Mai 1992.

[NMV90] R. De Nicola, U. Montanari, et F.W. Vaandrager. Back and forth bisimulations. Rapport de recherche, Centre for Mathematics and Computer Science, Pays-Bas, Mai 1990.

[NOSY92] X. Nicollin, A. Olivero, J. Sifakis, et S. Yovine. An approach to the description and analysis of hybrid systems. Dans *Workshop on Theory of Hybrid Systems*, Lyngby, Denmark, Octobre 1992.

[NRSV90] X. Nicollin, J.-L. Richier, J. Sifakis, et J. Voiron. ATP: an Algebra for Timed Processes. Dans *Proc. IFIP TC 2 Working Conference on Programming Concepts and Methods*, Israël, Avril 1990.

[NS90] X. Nicollin et J. Sifakis. The algebra of timed processes ATP: theory and application. Rapport de Recherche RT-C26, LGI-IMAG, France, Décembre 1990.

[NS91] X. Nicollin et J. Sifakis. An overview and synthesis on timed process algebras. Dans K.G. Larsen et A. Skou, éditeurs, *Proc. 3ème CAV*, pages 376–398, Danemark, Juillet 1991. LNCS 575, Springer-Verlag.

[NSY91] X. Nicollin, J. Sifakis, et S. Yovine. From ATP to timed graphs and hybrid systems. Dans J.W. de Bakker, K. Huizing, W.-P. de Roever, et G. Rozenberg, éditeurs, *Proc. REX Workshop "Real-Time: Theory in Practice"*, pages 549–572. LNCS 600, Springer-Verlag, 1991.

[NSY92] X. Nicollin, J. Sifakis, et S. Yovine. Compiling real-time specifications into extended automata. *IEEE TSE Special Issue on Real-Time Systems*, 18(9):794–804, Septembre 1992.

[NSY93] X. Nicollin, J. Sifakis, et S. Yovine. From ATP to timed graphs and hybrid systems. *Acta Informatica*, 30:181–202, 1993.

[OY92] A. Olivero et S. Yovine. *Kronos: a Tool for Verifying Real-time Systems. User's Guide and Reference Manual.* VERIMAG, Grenoble, France, 1992.

[Par81] D. Park. Concurrency and automata on infinite sequences. Dans *LNCS 104*. Springer-Verlag, 1981.

[Par86] J. Parrow. Verifying a CSMA/CD-protocol with CCS. Rapport de recherche, Laboratory of Foundations of Computer Science, University of Edinburgh, Décembre 1986.

[Pet81] J.L. Peterson. *Petri Net Theory and the Modeling of Systems.* Prentice-Hall, Englewood Cliffs, 1981.

[Plo81] G. Plotkin. A structural approach to operational semantics. Rapport de Recherche DAIMI FN-19, Århus University, Danemark, 1981.

[Pnu77] A. Pnueli. The temporal logic of programs. Dans *Proc. 18ème FOCS*, pages 46–57. IEEE Computer Society Press, 1977.

[PS82] Papadimitriou et Steiglitz. *Combinatorial optimization. Algorithms and complexity.* Prentice-Hall, 1982.

[Rat92] Ch. Ratel. LESAR : un outil pour la vérification d'invariants de programmes LUSTRE. Thèse, Université Joseph Fourier, Grenoble, France, 1992.

[Ray91] P. Raymond. Compilation efficace d'un langage déclaratif synchrone : le générateur de code LUSTRE-V3. Thèse, Institut National Polytechnique de Grenoble, France, 1991.

[Rei85] W. Reisig. *Petri Nets: an Introduction.* EACTS Monographs on Theoretical Computer Science. Springer-Verlag, 1985.

[Roc92] F. Rocheteau. Extension du langage LUSTRE et application à la conception de circuits : Le langage LUSTRE-V4 et le système POLLUX. Thèse, Institut National Polytechnique de Grenoble, France, 1992.

[RR88] G.M. Reed et A.W. Roscoe. A timed model for Communicating Sequential Processes. *Theoretical Computer Science*, 58:249–261, 1988.

[Sha92] A. Shaw. Communicating real-time state machines. *IEEE Transactions on Software Engineering*, 18(9):805–816, 1992.

[Sif82a] J. Sifakis. Global and local invariants in transition systems. *Information and Computation*, 53(1), Avril 1982.

[Sif82b] J. Sifakis. A unified approach for studying the properties of transition systems. *Theoretical Computer Science*, 18, 1982.

[Tan89] A.S. Tanenbaum. *Computer Networks.* Prentice-Hall, Englewood Cliffs, second édition, 1989.

[Wan90] Wang Yi. Real-time behaviour of asynchronous agents. Dans J.C.M. Baeten et J.W. Klop, éditeurs, *Proc. CONCUR 90: Theories of Concurrency.* LNCS 458, Springer-Verlag, 1990.

[Yov92] S. Yovine. Compiling timed algebras into timed automata. Dans *XVIII Conferencia Latinoamericana de Informática, PANEL'92*, Espagne, Août 1992.

# Annexe A

# Consistance de la méthode de traduction

Un problème important concernant la consistance de la méthode est du à l'introduction des transitions par $\varepsilon$. En effet, les formules de TCTL sont interprétées sur un modèle où les transitions étiquetées par les actions, y compris les transitions par $\varepsilon$, sont considérées comme des transitions instantanées. Autrement dit, aucun traitement spécial n'est consacré aux transitions par $\varepsilon$, qui ne sont donc pas distinguées des transitions par des actions. Par conséquent, c'est au niveau de la méthode de construction qu'il faut assurer que les transitions par $\varepsilon$ ne changent pas le comportement attendu du système, c'est-à-dire, que leur introduction n'a aucun effet sur la validité des formules.

Une manière de prouver la consistance de la méthode de traduction consiste à montrer que le modèle de $P$ et le modèle de $G$ sont *équivalents* modulo une relation d'équivalence *consistante* pour TCTL, au sens où deux états équivalents satisfont les mêmes formules. Nous constatons immédiatement que $G$ n'est pas en général fortement équivalent à $P$, car l'étiquette $\varepsilon$ introduite par la méthode de construction n'apparaît comme étiquette d'aucune transition dans le modèle de $P$, pourtant elle apparaît dans le modèle de $G$.

Il est nécessaire, par conséquent, de définir une autre relation d'équivalence. Pour ceci nous présentons tout d'abord une nouvelle sémantique d'ATP où l'action $\varepsilon$ est introduite de manière explicite pour modéliser les expirations des watchdogs. Nous prouvons que si $\mathcal{T}[\![P]\!]_\varepsilon$ est le système de transitions de $P$ avec cette sémantique, alors il est fortement équivalent à $\mathcal{T}[\![G]\!]$.

Nous définissons ensuite une relation d'équivalence $\sim_b$, similaire à la bisimulation de branchement [GW89, NMV90, Mou92], qui tient compte des transitions par $\varepsilon$. Nous prouvons que, sur les modèles avec des transitions par $\varepsilon$, la relation $\sim_b$ correspond exactement à la bisimulation forte $\sim$ sur les modèles sans actions $\varepsilon$. Autrement dit, $\sim_b$ ne permet distinguer ni plus ni moins des termes d'ATP que la relation d'équivalence forte $\sim$. Finalement, nous montrons que $\sim_b$ est consistante pour de TCTL, ce qui prouve que la correction de l'approche proposée.

## A.1 Une sémantique d'ATP avec des transitions par $\varepsilon$

Nous définissons tout d'abord l'algèbre de processus $\text{ATP}_\varepsilon$, une extension d'ATP où la constante 0 est permise comme paramètre de l'opérateur watchdog.

**Définition A.1** Les termes $\text{ATP}_\varepsilon$ de l'algèbre sont définis par la syntaxe suivante :

$$P ::= \mathsf{idle} \mid X \mid aP \mid \mathsf{d}(a)P \mid P \ \mathsf{watchdog}[A](t)\ P \mid P + P \mid P \,||\, P \mid \mathsf{restrict}\ A\ \mathsf{in}\ P \mid \mathsf{rec}X \cdot P$$

où $X \in \mathit{Var}$, $a \in \mathit{Act}$, $t \in \mathsf{R}^+$ et $A \subseteq \mathit{Act}$. □

Les notions de variable *gardée* et de terme *régulier* sont définies de la façon suivante.

**Définition A.2** Une variable $X \in \mathit{Var}$ est *gardée* dans un terme $P$ si toute occurrence libre de $X$ dans $P$ apparaît, soit dans l'argument d'un opérateur de préfixage, soit dans le second argument d'un opérateur watchdog. □

**Définition A.3** L'ensemble des termes *réguliers*, noté $\text{ATP}^r_\varepsilon$, est défini de la façon suivante :

- $\mathsf{idle} \in \text{ATP}^r_\varepsilon$,
- pour toute variable $X \in \mathit{Var}$, $X \in \text{ATP}^r_\varepsilon$,
- si $P \in \text{ATP}^r_\varepsilon$, alors $aP \in \text{ATP}^r_\varepsilon$ et $\mathsf{d}(a)P \in \text{ATP}^r_\varepsilon$, pour tout $a \in \mathit{Act}$,
- si $P, Q \in \text{ATP}^r_\varepsilon$, et pour toute variable $X \in \mathit{vl}(P) \cup \mathit{vl}(Q)$, $X$ est gardée par une action dans $P$ et dans $Q$, alors $P + Q \in \text{ATP}^r_\varepsilon$,
- si $P, Q \in \text{ATP}^r_\varepsilon$ et $\mathit{vl}(P) = \mathit{vl}(Q) = \emptyset$, alors $P \,||\, Q \in \text{ATP}^r_\varepsilon$,
- si $P \in \text{ATP}^r_\varepsilon$ et $\mathit{vl}(P) = \emptyset$, alors $\mathsf{restrict}\ A\ \mathsf{in}\ P \in \text{ATP}^r_\varepsilon$,
- si $P \in \text{ATP}^r_\varepsilon$, et pour toute variable $Y \in \mathit{vl}(P)$, $Y$ est gardée dans $P$, alors $\mathsf{rec}X \cdot P \in \text{ATP}^r_\varepsilon$.

On note $\text{ATP}^r$ le sous-ensemble des termes d'$\text{ATP}^r_\varepsilon$ qui appartiennent à ATP. □

*Exemple.* Les termes $aX + Y$, $a(X + Y)$, $\mathsf{rec}X \cdot aX + Y$, $aX \,||\, b\mathsf{idle}$ et $\mathsf{rec}X \cdot a(\mathsf{restrict}\ a\ \mathsf{in}\ bX)$ ne sont pas dans $\text{ATP}^r_\varepsilon$. □

**Définition A.4** L'ensemble $\text{ATP}^{fr}_\varepsilon$ est le sous-ensemble des termes *fermés* et *réguliers* de $\text{ATP}_\varepsilon$. Autrement dit, $\text{ATP}^{fr}_\varepsilon$ est le sous-ensemble des termes fermés de $\text{ATP}^r_\varepsilon$. On note $\text{ATP}^{fr}$ le sous-ensemble des termes de $\text{ATP}^{fr}_\varepsilon$ qui appartiennent à ATP. □

Nous décrivons à présent la sémantique opérationnelle, qui associe à un terme un système temporisé, i.e., un système de transitions qui satisfait les propriétés de déterminisme et d'additivité temporelles.

A la différence de la sémantique d'ATP présentée dans le chapitre 5, des transitions étiquetées par $\varepsilon$ sont introduites par les règles sémantiques d'$\text{ATP}_\varepsilon$, afin de représenter explicitement l'expiration d'un opérateur watchdog. Le système de transitions d'un terme d'$\text{ATP}_\varepsilon$ est défini sur l'ensemble d'étiquettes $\mathit{Lab} = \mathit{Act}_\varepsilon \cup \mathsf{R}^{>0}$. Nous supposons que $\varepsilon$ ne peut communiquer avec aucune action, i.e., $\varepsilon \mid \alpha = \bot$ pour tout $\alpha \in \mathit{Act}_\varepsilon$.

**Définition A.5** Le modèle d'un terme $P \in \text{ATP}_\varepsilon$, noté $\mathcal{T}[\![P]\!]_\varepsilon$, est le système temporisé $\langle \text{ATP}_\varepsilon, \rightarrow, P \rangle$, où la relation de transition $\rightarrow$ est la plus petite relation satisfaisant aux règles ci-dessous. □

Les règles pour les termes **idle**, $X \in Var$, $aP$ et $\mathsf{d}(a)P$ pour $a \in Act$, **restrict** $A$ **in** $P$ et **rec**$X \cdot P$ sont les mêmes d'ATP(cf. chapitre 5). Pour les autres opérateurs nous avons les règles suivantes :

• Dans le cas de l'opérateur watchdog les règles sont conservées lorsque le délai $t$ est supérieur à 0. Pour $\alpha \in Act_\varepsilon$,

$$\text{wda1-}\varepsilon) \quad \frac{P_1 \xrightarrow{\alpha} P_1' \wedge \alpha \in A \wedge t > 0}{P_1 \ \mathsf{watchdog}[A](t)\ P_2 \xrightarrow{\alpha} P_1'}$$

et

$$\text{wda2-}\varepsilon) \quad \frac{P_1 \xrightarrow{\alpha} P_1' \wedge \alpha \notin A \wedge t > 0}{P_1 \ \mathsf{watchdog}[A](t)\ P_2 \xrightarrow{\alpha} P_1' \ \mathsf{watchdog}[A](t)\ P_2}$$

et pour le passage du temps,

$$\text{wdt-}\varepsilon) \quad \frac{P_1 \xrightarrow{t'} P_1' \wedge 0 < t' \leq t}{P_1 \ \mathsf{watchdog}[A](t)\ P_2 \xrightarrow{t'} P_1' \ \mathsf{watchdog}[A](t - t')\ P_2}$$

où les règles wdt1) et wdt2) d'ATP sont réunies dans une seule règle. Notons que si $t' = t$, nous avons :

$$P_1 \ \mathsf{watchdog}[A](t)\ P_2 \xrightarrow{t'} P_1' \ \mathsf{watchdog}[A](0)\ P_2$$

Pour $t = 0$, nous avons :

$$\text{wd0-}\varepsilon) \quad P_1 \ \mathsf{watchdog}[A](0)\ P_2 \xrightarrow{\varepsilon} P_2$$

Dans cette sémantique l'expiration d'un watchdog est modélisée par une suite de deux transitions : une transition temporelle étiquetée par un temps égal au paramètre du watchdog, suivie d'une transition étiquetée par $\varepsilon$.

• Pour l'opérateur de choix non déterministe, les règles sont les mêmes d'ATP en ce qui concerne le passage du temps et les actions $a$ de $Act$. Pour $\varepsilon$ nous ajoutons les règles suivantes :

$$\text{choice1-}\varepsilon) \quad \frac{P_1 \xrightarrow{\varepsilon} P_1'}{P_1 + P_2 \xrightarrow{\varepsilon} P_1' + P_2} \qquad \text{choice2-}\varepsilon) \quad \frac{P_2 \xrightarrow{\varepsilon} P_2'}{P_1 + P_2 \xrightarrow{\varepsilon} P_1 + P_2'}$$

c'est-à-dire, $\varepsilon$ ne résout pas un choix non déterministe.

• Pour l'opérateur de composition parallèle, les règles sont les mêmes d'ATP pour le passage du temps et les actions $a$ de $Act$. Pour $\varepsilon$ nous ajoutons les règles suivantes :

$$\text{par1-}\varepsilon) \quad \frac{P_1 \xrightarrow{\varepsilon} P_1'}{P_1 \parallel P_2 \xrightarrow{\varepsilon} P_1' \parallel P_2} \qquad \text{par2-}\varepsilon) \quad \frac{P_2 \xrightarrow{\varepsilon} P_2'}{P_1 \parallel P_2 \xrightarrow{\varepsilon} P_1 \parallel P_2'}$$

Il n'y a pas des règles de synchronisation des transitions $\varepsilon$ car $\varepsilon$ ne peut communiquer avec aucune action.

## A.2 La bisimulation de branchement

En général, pour un terme $P$ d'ATP, la différence entre entre les modèles $\mathcal{T}[\![P]\!]$ et $\mathcal{T}[\![P]\!]_\varepsilon$, est qu'une transition temporelle $P \xrightarrow{t} P'$ dans $\mathcal{T}[\![P]\!]$ correspond à une suite de transitions

$$P \xrightarrow{\varepsilon}{}^{*} Q_1 \xrightarrow{t_1} R_1 \ldots \xrightarrow{\varepsilon}{}^{*} Q_n \xrightarrow{t_n} P'$$

dans $\mathcal{T}[\![P]\!]_\varepsilon$, où $t_1 + \ldots + t_n = t$. Autrement dit, dans $\mathcal{T}[\![P]\!]_\varepsilon$ l'écoulement du temps est "coupé" par des transitions par $\varepsilon$ qui représentent les expirations des opérateurs watchdog. Egalement, une transition $P \xrightarrow{a} P'$ dans $\mathcal{T}[\![P]\!]$ correspond à une suite de transitions

$$P \xrightarrow{\varepsilon}{}^{*} Q \xrightarrow{a} P'$$

dans $\mathcal{T}[\![P]\!]_\varepsilon$. Ceci motive la définition d'une relation d'équivalence, notée $\sim_b$, qui est inspirée de la bisimulation de branchement définie dans [GW89, NMV90, Mou92] où $\varepsilon$ joue le rôle de l'action $\tau$.

**Définition A.6** Pour $i = 1, 2$ soient $T_i = \langle \mathcal{Q}_i, \rightarrow_i, q_{I_i} \rangle$. $\mathcal{R} \subseteq \mathcal{Q}_1 \times \mathcal{Q}_2$ est une *bisimulation de branchement* si pour tout $(q_1, q_2) \in \mathcal{R}$ et $\ell \in Lab$ :

1. pour tout $q_1' \in \mathcal{Q}_1$ tel que $q_1 \xrightarrow{\ell} q_1'$,
   (a) si $\ell \in Act$, alors
      i. $\ell = \varepsilon$ et $(q_1', q_2) \in \mathcal{R}$, ou
      ii. ils existent $q_2', q_2'' \in \mathcal{Q}_2$ tels que $q_2 \xrightarrow{\varepsilon}{}^{*} q_2'' \xrightarrow{\ell} q_2'$, $(q_1, q_2'') \in \mathcal{R}$ et $(q_1', q_2') \in \mathcal{R}$.
   (b) si $\ell = t > 0$, alors il existe $n > 0$ et pour $i = 1, \ldots, n$, ils existent $r_2^i, p_2^i \in \mathcal{Q}_2$ et $t_i > 0$, tels que $t_1 + \ldots + t_n = t$ et
   $$q_2 \xrightarrow{\varepsilon}{}^{*} r_2^1 \xrightarrow{t_1} p_2^1 \ldots \xrightarrow{\varepsilon}{}^{*} r_2^n \xrightarrow{t_n} p_2^n$$
   et pour les uniques[1] $q_1^i$ pour $i = 1, \ldots, n$ satisfaisant $r_1^n = q_1'$ et
   $$q_1 \xrightarrow{t_1} q_1^1 \ldots q_1^{n-1} \xrightarrow{t_n} q_1^n$$
   on a $(q_1, r_2^1) \in \mathcal{R}$, $(q_1^i, p_2^i) \in \mathcal{R}$, $(q_1^i, r_2^{i+1}) \in \mathcal{R}$ pour $i = 1, \ldots, n-1$ et $(q_1^n, p_2^n) \in \mathcal{R}$.
2. et symétriquement pour $q_2$.

Si $T_1 = T_2$, on dit que $\mathcal{R}$ est une auto-bisimulation. On note $q_1 \sim_b q_2$ s'il existe une bisimulation de branchement $\mathcal{R} \subseteq \mathcal{Q}_1 \times \mathcal{Q}_2$ telle que $(q_1, q_2) \in \mathcal{R}$, i.e., $\sim_b$ est la plus grande bisimulation de branchement. On note $T_1 \sim_b T_2$ si $q_{I_1} \sim_b q_{I_2}$. □

Il peut paraître intéressant de renforcer la définition ci-dessus en exigeant que tous les états intermédiaires dans $q \xrightarrow{\varepsilon}{}^{*} q'$ soient aussi équivalents à $q$. Le lemme suivant montre que la relation d'équivalence ainsi obtenue coïncide avec $\sim_b$.

**Lemme A.1** *Soient $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ un système temporisé et $q_0, \ldots, q_n \in \mathcal{Q}$, $n > 0$, tels que $q_0 \xrightarrow{\varepsilon} q_1 \ldots q_{n-1} \xrightarrow{\varepsilon} q_n$ avec $q_0 \sim_b q_n$. Alors, pour tout $0 \leq i \leq n$, on a $q_0 \sim_b q_i$.*

**Preuve.** Similaire à [NMV90]. ■

**Remarque A.1** Notons que si deux systèmes temporisés sont équivalents modulo la bisimulation forte, alors ils sont également équivalents pour la bisimulation de branchement, c'est-à-dire, si $T_1 \sim T_2$, alors $T_1 \sim_b T_2$. □

[1] à cause du déterminisme temporel de $\rightarrow$.

De plus, la bisimulation de branchement sur des modèles sans transitions $\varepsilon$ coïncide avec la bisimulation forte. En effet, nous avons le lemme suivant.

**Lemme A.2** *Soit $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ un système temporisé défini sur $Lab = Act \cup \mathsf{R}^{>0}$. Si $\mathcal{R} \subseteq \mathcal{Q} \times \mathcal{Q}$ est une bisimulation de branchement alors $\mathcal{R}$ est une bisimulation forte.*

**Preuve.** Soient $q_1, q_2 \in \mathcal{Q}$ tels que $(q_1, q_2) \in \mathcal{R}$. Supposons que $q_1 \stackrel{\ell}{\rightarrow} q_1'$. Si $\ell = a$ alors il existe $q_2'$ tel que $q_2 \stackrel{\ell}{\rightarrow} q_2'$ et $(q_1', q_2') \in \mathcal{R}$. Si $\ell = t$ alors ils existent $t_1, \ldots, t_n$ tels que $q_2 \stackrel{t_1}{\rightarrow} \ldots \stackrel{t_n}{\rightarrow} q_2'$, $t_1 + \ldots + t_n = t$ et $(q_1', q_2') \in \mathcal{R}$. D'après la propriété d'additivité temporelle on a $q_2 \stackrel{t}{\rightarrow} q_2'$. ■

Comme nous l'avons déjà dit dans l'introduction du chapitre, notre intention est de montrer tout d'abord que pour tout terme $P$ d'ATP, les modèles $\mathcal{T}[\![P]\!]$ et $\mathcal{T}[\![P]\!]_\varepsilon$ sont équivalents modulo la relation $\sim_b$. Nous ne considérons que l'ensemble des termes fermés et réguliers. Nous voulons prouver la proposition suivante.

**Proposition A.1** *Pour tout $P \in \mathrm{ATP}^{fr}$, $\mathcal{T}[\![P]\!] \sim_b \mathcal{T}[\![P]\!]_\varepsilon$.*

Supposons pour un instant que nous avons déjà prouvé ce résultat. Il faut montrer ensuite que pour tout terme $P$ d'ATP, le modèle du graphe temporisé $G = \mathcal{G}[\![P]\!]$ construit est équivalent, modulo la bisimulation forte, au modèle de $P$ avec des transitions $\varepsilon$. Nous voulons donc montrer la proposition suivante.

**Proposition A.2** *Pour tout $P \in \mathrm{ATP}^{fr}$ et $G = \mathcal{G}[\![P]\!]$, $\mathcal{T}[\![P]\!]_\varepsilon \sim \mathcal{T}[\![G]\!]$.*

A partir des propositions A.1 et A.2, du lemme A.2 et de la remarque A.1, nous avons les deux corollaires suivants.

**Corollaire A.1** *Pour tout $P \in \mathrm{ATP}^{fr}$ et $G = \mathcal{G}[\![P]\!]$, $\mathcal{T}[\![P]\!] \sim_b \mathcal{T}[\![G]\!]$.*

**Corollaire A.2** *Pour tout $P, Q \in \mathrm{ATP}^{fr}$, $\mathcal{T}[\![P]\!] \sim \mathcal{T}[\![Q]\!]$ si et seulement si $\mathcal{T}[\![P]\!]_\varepsilon \sim_b \mathcal{T}[\![Q]\!]_\varepsilon$.*

La section suivante est consacrée à la preuve des propositions A.1 et A.2.

## A.3 Preuve de la consistance de la méthode de traduction

Afin de prouver les propositions A.1 et A.2, nous montrons tout d'abord quelques lemmes qui sont utiles.

**Lemme A.3** *Soient $P, Q, R \in \mathrm{ATP}_\varepsilon$, $\ell \in Lab$ et $X \in Var$. Si $P \stackrel{\ell}{\rightarrow} Q$ alors $P[R/X] \stackrel{\ell}{\rightarrow} Q[R/X]$.*

**Preuve.** Par induction dans la longueur de la preuve de $P \stackrel{\ell}{\rightarrow} Q$. ■

**Lemme A.4** *Soient $P, Q, R \in \mathrm{ATP}_\varepsilon$, $X \in Var$ gardée dans $P$, et $\ell \in Lab$. Si $P[Q/X] \stackrel{\ell}{\rightarrow} R$, alors il existe $P' \in \mathrm{ATP}_\varepsilon$ tel que $R$ est le terme $P'[Q/X]$ et $P \stackrel{\ell}{\rightarrow} P'$.*

**Preuve.** Par induction dans la longueur de la preuve de $P[Q/X] \xrightarrow{\ell} R$. ■

**Lemme A.5** *Soient $P, P' \in \mathrm{ATP}_\varepsilon$ et $\ell \in Lab$ tels que $P \xrightarrow{\ell} P'$. Pour toute variable $X \in Var$, si $X$ est gardée par une action $a \in Act$ et $a \neq \ell$, alors $X$ est gardée par $a$ dans $P'$.*

**Preuve.** Par induction sur la longueur de la preuve de $P \xrightarrow{\ell} P'$. ■

L'ensemble $\mathrm{ATP}^r_\varepsilon$ des termes réguliers est fermé pour la relation de transition entre les termes, c'est-à-dire, $\mathrm{ATP}^r_\varepsilon$ satisfait la propriété suivante.

**Lemme A.6** *Pour tout terme $P \in \mathrm{ATP}^r_\varepsilon$, s'ils existent $\ell \in Lab$ et $P' \in \mathrm{ATP}_\varepsilon$ tels que $P \xrightarrow{\ell} P'$, alors $P' \in \mathrm{ATP}^r_\varepsilon$.*

**Preuve.** Par induction sur la longueur de la preuve de $P \xrightarrow{\ell} P'$, en appliquant les lemmes A.3, A.4 et A.5. ■

Une relation d'équivalence entre termes est qu'elle doit être préservée par substitution, c'est-à-dire, si $P$ et $Q$ sont équivalents, alors $P[R/X]$ et $Q[R/X]$ sont équivalents pour tous $R$ et $X$.

Considérons, par exemple, les termes $aX$ et $aY$. Ces termes sont fortement équivalents, en effet, $\{\langle aX, aY\rangle, \langle X, Y\rangle\}$ est une bisimulation forte. Par contre, les termes $(aX)[P/X]$ et $(aY)[P/X]$, pour tout terme $P$, ne sont pas équivalents.

Cette différence de comportement de la substitution doit se traduire dans la sémantique. Un solution consiste à introduire des transitions pour les variables de la façon suivante. Pour tout $X \in Var$ :

$$X \xrightarrow{X} \mathsf{idle}$$

Cette règle permet donc de distinguer les termes $aX$ et $aY$.

Il n'est pas nécessaire d'introduire ces règles si l'on ne considère que des termes fermés. Toutefois, pour prouver les résultats énoncés, il faut traiter les sous-termes ouverts afin de pouvoir raisonner par induction sur la structure des termes. Dans ce cas, on a besoin de distinguer les occurrences libres des différentes variables pour que la relation d'équivalence soit préservée par substitution.

On note $\mathcal{T}[\![P]\!]^*$ le modèle d'un terme défini en tenant compte des transitions par des variables. Pour prouver la proposition A.1 nous montrons que la propriété énoncée est valide pour les termes d'$\mathrm{ATP}^r_\varepsilon$. La proposition A.1 suit immédiatement à partir du lemme suivant.

**Lemme A.7** *Pour tout $P \in \mathrm{ATP}^r$, $\mathcal{T}[\![P]\!]^* \sim_b \mathcal{T}[\![P]\!]^*_\varepsilon$.*

Par contre, pour prouver la proposition A.2, nous considérons les modèles sans transitions par des variables, car il s'agit de comparer le modèle d'un terme au modèle d'un graphe. Dans ce cas, nous montrons que la propriété énoncée est valide pour les termes d'$\mathrm{ATP}^r_\varepsilon$ et pour le graphe temporisé *étendu* correspondant. La proposition A.2 suit immédiatement à partir du lemme suivant.

**Lemme A.8** *Pour tout $P \in \mathrm{ATP}^r$ et $G = \mathcal{E}[\![P]\!]$, $\mathcal{T}[\![P]\!]_\varepsilon \sim \mathcal{T}[\![G]\!]$.*

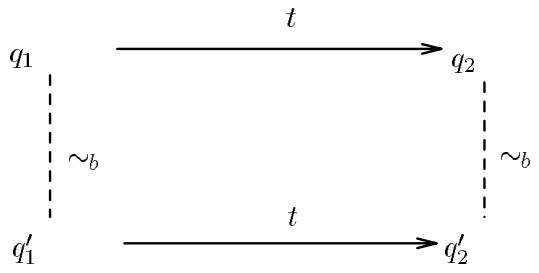



Figure A.1: Propriété de la relation $\sim_b$.

Afin de prouver ces deux lemmes, nous avons besoin de quelques lemmes auxiliaires. D'après les lemmes A.3, A.4 et A.6, nous avons les corollaires suivants.

**Corollaire A.3** *Soient* $P, Q, R \in \mathrm{ATP}^r_\varepsilon$ *et* $X \in \mathrm{Var}$. *Si* $P \xrightarrow{\ell} Q$ *et* $\ell \neq X$, *alors* $P[R/X] \xrightarrow{\ell} Q[R/X]$.

**Corollaire A.4** *Soient* $P, Q, R \in \mathrm{ATP}^r_\varepsilon$ *et* $X \in \mathrm{Var}$. *Si* $P[Q/X] \xrightarrow{\ell} R$ *et* $\ell \neq X$, *alors il existe* $P' \in \mathrm{ATP}^r_\varepsilon$ *tel que* $R$ *est le terme* $P'[Q/X]$ *et* $P \xrightarrow{\ell} P'$.

Les modèles des termes d'$\mathrm{ATP}^r_\varepsilon$ satisfont que les état qui ont des transitions temporelles n'ont pas des transitions par $\varepsilon$. En effet, nous avons le lemme suivant.

**Lemme A.9** *Pour tout* $P \in \mathrm{ATP}^r_\varepsilon$, *s'il existe* $P'$ *tel que* $P \xrightarrow{t} P'$ *alors* $P \overset{\varepsilon}{\not\rightarrow}$.

**Preuve.** Par induction sur la longueur de la preuve de $P \xrightarrow{t} P'$. ■

**Lemme A.10** *Soit* $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ *un système temporisé tel que pour tout* $q \in \mathcal{Q}$, *si* $q \xrightarrow{t} q'$ *alors* $q \overset{\varepsilon}{\not\rightarrow}$. *Si* $q_1, q'_1 \in \mathcal{Q}$ *sont tels que* $q_1 \sim_b q'_1$, *alors pour tout* $q_2, q'_2 \in \mathcal{Q}$ *et* $t > 0$, *tels que* $q_1 \xrightarrow{t} q_2$ *et* $q'_1 \xrightarrow{t} q'_2$, *on a* $q_2 \sim_b q'_2$. *(cf. figure A.1).*

**Preuve.** D'après la définition de $\sim_b$ et les propriétés de déterminisme et d'additivité des transitions temporelles. ■

Les transitions par $\varepsilon$ satisfont les propriétés suivantes.

**Lemme A.11** *Soient* $P_1, P'_1 \in \mathrm{ATP}^r_\varepsilon$ *tels que* $P_1 \xrightarrow{\varepsilon}{}^* P'_1$. *Alors, pour tout* $P_2 \in \mathrm{ATP}^r_\varepsilon$, $A \in Act$ *et* $t > 0$ *:*

1. $P_1$ watchdog$[A](t)\ P_2 \xrightarrow{\varepsilon}{}^* P'_1$ watchdog$[A](t)\ P_2$
2. $P_1 + P_2 \xrightarrow{\varepsilon}{}^* P'_1 + P_2$
3. $P_1 \,\|\, P_2 \xrightarrow{\varepsilon}{}^* P'_1 \,\|\, P_2$

**Preuve.** Par induction sur la longueur $n$ de la séquence de transitions par $\varepsilon$. ■

Nous avons comme corollaire que les transitions par $\varepsilon$ sont confluentes.

**Corollaire A.5** *Soient* $P_1, P_2, P_1', P_2' \in \mathrm{ATP}_\varepsilon^r$ *tels que* $P_1 \xrightarrow{\varepsilon}{}^* P_1'$ *et* $P_2 \xrightarrow{\varepsilon}{}^* P_2'$. *Alors,*

1. $P_1 + P_2 \xrightarrow{\varepsilon}{}^* P_1' + P_2'$
2. $P_1 \,||\, P_2 \xrightarrow{\varepsilon}{}^* P_1' \,||\, P_2'$

Les lemmes suivants montrent que si $P_1 \sim_b R_1$ et $P_2 \sim_b R_2$, alors $P_1 + P_2 \sim_b R_1 + R_2$ et $P_1 \,||\, P_2 \sim_b R_1 \,||\, R_2$. En d'autres termes, l'équivalence de branchement est préservée par les opérateurs de choix non déterministe et de composition parallèle.

**Lemme A.12** *Soient* $\ddagger \in \{+, ||\}$, $P_1, P_2, R_1, R_2 \in \mathrm{ATP}_\varepsilon^r$ *tels que* $P_1 \sim_b R_1$ *et* $P_2 \sim_b R_2$. *Pour tous* $P_1', P_2' \in \mathrm{ATP}_\varepsilon^r$ *et* $\alpha \in Act_\varepsilon$, *si* $P_1 \ddagger P_2 \xrightarrow{\alpha} P_1' \ddagger P_2'$, *alors ils existent* $R_i', R_i'' \in \mathrm{ATP}_\varepsilon^r$, *pour* $i = 1, 2$, *tels que* $R_1 \ddagger R_2 \xrightarrow{\varepsilon}{}^* R_1'' \ddagger R_2'' \xrightarrow{\alpha} R_1' \ddagger R_2'$ *et* $P_i \sim_b R_i''$ *et* $P_i' \sim_b R_i'$.

**Preuve.** Immédiate d'après la définition de $\sim_b$ et le corollaire A.5. ■

**Lemme A.13** *Soient* $\ddagger \in \{+, ||\}$, $P_1, P_2, R_1, R_2 \in \mathrm{ATP}_\varepsilon^r$ *tels que* $P_1 \sim_b R_1$ *et* $P_2 \sim_b R_2$, $P_1', P_2' \in \mathrm{ATP}_\varepsilon^r$ *et* $t > 0$ *tels que* $P_1 \xrightarrow{t} P_1'$ *et* $P_2 \xrightarrow{t} P_2'$.

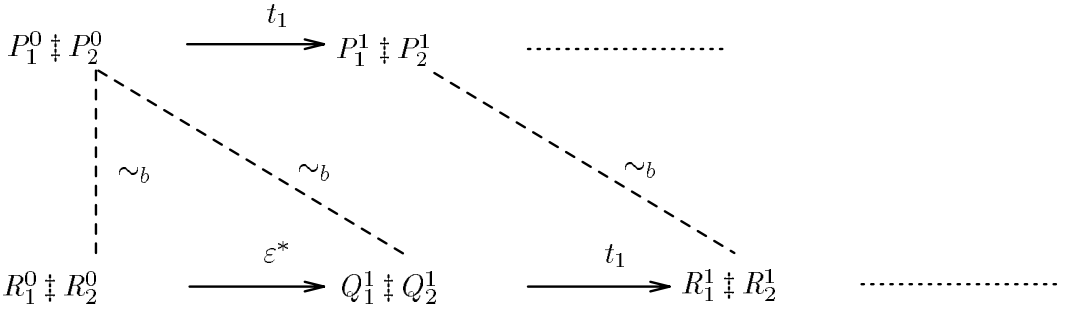
*Alors, pour* $i = 1, 2$, $k = 0, \ldots, r$ *et* $r > 0$, *ils existent* $R_i^k, Q_i^k \in \mathrm{ATP}_\varepsilon^r$, *avec* $R_i^0 = R_i$ *et* $t_1, \ldots, t_r > 0$ *avec* $t_1 + \ldots + t_r = t$ *tels que pour les uniques* $P_i^k$ *avec* $P_i^0 = P_i$ *et* $P_i^r = P_i'$ *satisfaisant*

$$P_1^0 \ddagger P_2^0 \xrightarrow{t_1} P_1^1 \ddagger P_2^1 \ldots P_1^{r-1} \ddagger P_2^{r-1} \xrightarrow{t_r} P_1^r \ddagger P_2^r$$

*on a :*

$$R_1^0 \ddagger R_2^0 \xrightarrow{\varepsilon}{}^* Q_1^1 \ddagger Q_2^1 \xrightarrow{t_1} R_1^1 \ddagger R_2^1 \ldots \xrightarrow{\varepsilon}{}^* Q_1^r \ddagger Q_2^r \xrightarrow{t_r} R_1^r \ddagger R_2^r$$

*et* $P_i^k \sim_b Q_i^{k+1}$ *pour* $k = 0, \ldots, r-1$ *et* $P_i^k \sim_b R_i^k$ *pour* $k = 0, \ldots, r$.




**Preuve.** D'après la définition de bisimulation de branchement (cf. définition A.6), on a :

1. Soient $P_1, P_1', R_1 \in \mathrm{ATP}^r_\varepsilon$ et $t > 0$ tels que $P_1 \xrightarrow{t} P_1'$ et $P_1 \sim_b R_1$. Alors, ils existent $\bar{R}_1^j, \tilde{R}_1^j \in \mathrm{ATP}^r_\varepsilon$ pour $j = 1, \ldots, n$ et $t_1^1, \ldots, t_1^n$ tels que $t_1^1 + \ldots + t_1^n = t$ et

   $$R_1^0 \xrightarrow{\varepsilon}{}^* \bar{R}_1^1 \xrightarrow{t_1^1} \tilde{R}_1^1 \ldots \xrightarrow{\varepsilon}{}^* \bar{R}_1^n \xrightarrow{t_1^n} \tilde{R}_1^n$$

   où $R_1^0 = R_1$ et pour les uniques $P_1^j$, $j = 0, \ldots, n$ avec $\bar{P}_1^0 = P_1$ et $\bar{P}_1^n = P_1'$, satisfaisant

   $$\bar{P}_1^0 \xrightarrow{t_1^1} \bar{P}_1^1 \ldots \bar{P}_1^{n-1} \xrightarrow{t_1^n} \bar{P}_1^n$$

   on a $\bar{P}_1^j \sim_b \bar{R}_1^{j+1}$ pour $j = 0, \ldots, n-1$ et $\bar{P}_1^j \sim_b \tilde{R}_1^j$ pour $j = 0, \ldots, n$.

2. Soient $P_2, P_2', R_2 \in \mathrm{ATP}^r_\varepsilon$ et $t > 0$ tels que $P_2 \xrightarrow{t} P_2'$ et $P_2 \sim_b R_2$. Alors, ils existent $\bar{R}_2^l, \tilde{R}_2^l \in \mathrm{ATP}^r_\varepsilon$ pour $l = 1, \ldots, m$ et $t_2^1, \ldots, t_2^m$ tels que $t_2^1 + \ldots + t_2^m = t$ et

   $$R_2^0 \xrightarrow{\varepsilon}{}^* \bar{R}_2^1 \xrightarrow{t_2^1} \tilde{R}_2^1 \ldots \xrightarrow{\varepsilon}{}^* \bar{R}_2^m \xrightarrow{t_2^m} \tilde{R}_2^m$$

   où $R_2^0 = R_2$ et pour les uniques $P_2^l$, $l = 0, \ldots, m$ avec $P_2^0 = P_2$ et $P_2^m = P_2'$, satisfaisant

   $$\bar{P}_2^0 \xrightarrow{t_2^1} \bar{P}_2^1 \ldots \bar{P}_2^{m-1} \xrightarrow{t_2^m} \bar{P}_2^m$$

   on a $\bar{P}_2^l \sim_b \bar{R}_2^{l+1}$ pour $l = 0, \ldots, m-1$ et $\bar{P}_2^l \sim_b \tilde{R}_2^l$ pour $l = 0, \ldots, m$.

On construit la séquence

$$R_1^0 \ddagger R_2^0 \xrightarrow{\varepsilon}{}^* Q_1^1 \ddagger Q_2^1 \xrightarrow{t_1} R_1^1 \ddagger R_2^1 \ldots \xrightarrow{\varepsilon}{}^* Q_1^r \ddagger Q_2^r \xrightarrow{t_r} R_1^r \ddagger R_2^r$$

de la façon suivante.
Pour $i = 1, 2$, soient $Q_i^1 = \bar{R}_i^1$. On a donc $R_1^0 \xrightarrow{\varepsilon}{}^* Q_1^1$ et $R_2^0 \xrightarrow{\varepsilon}{}^* Q_2^1$ et d'après le lemme A.11, $R_1^0 \ddagger R_2^0 \xrightarrow{\varepsilon}{}^* Q_1^1 \ddagger Q_2^1$.
Soit $t_1 = \min(t_1^1, t_2^1)$. On a trois cas :

1. Si $t_1 = t_1^1 < t_2^1$ on définit $R_1^1 = \tilde{R}_1^1$ et $R_2^1$ est tel que $R_2^1 \xrightarrow{t'} \tilde{R}_2^1$ où $t' = t_2^1 - t_1$.
   De plus, on définit $P_1^1 = \bar{P}_1^1$ et $P_2^1$ est tel que $P_2^1 \xrightarrow{t'} \bar{P}_2^1$.
   Donc, $P_1^1 \sim_b R_1^1$, et d'après le lemme A.10, $P_2^1 \sim_b R_2^1$.

2. Si $t_1 = t_2^1 < t_1^1$ on définit $R_2^1 = \tilde{R}_2^1$ et $R_1^1$ est tel que $R_1^1 \xrightarrow{t'} \tilde{R}_1^1$ où $t' = t_1^1 - t_1$.
   De plus, on définit $P_2^1 = \bar{P}_2^1$ et $P_1^1$ est tel que $P_1^1 \xrightarrow{t'} \bar{P}_1^1$.
   Donc, $P_2^1 \sim_b R_2^1$, et d'après le lemme A.10, $P_1^1 \sim_b R_1^1$.

3. Si $t_1 = t_1^1 = t_2^1$ on définit $R_1^1 = \tilde{R}_1^1$ et $R_2^1 = \tilde{R}_2^1$.
   De plus, on définit $P_1^1 = \bar{P}_1^1$ et on a $P_2^1 = \bar{P}_2^1$.
   Donc, $P_1^1 \sim_b R_1^1$, et $P_2^1 \sim_b R_2^1$.

Dans tous les cas, $R_1^1$ et $R_2^1$ sont tels que $Q_1^1 \ddagger Q_2^1 \xrightarrow{t_1} R_1^1 \ddagger R_2^1$.

Par conséquent, $R_1^0 \ddagger R_2^0 \xrightarrow{\varepsilon}{}^* Q_1^1 \ddagger Q_2^1 \xrightarrow{t_1} R_1^1 \ddagger R_2^1$.

De plus, $P_i^0 \sim_b R_i^0$, $P_i^0 \sim_b Q_i^1$ et $P_i^1 \sim_b R_i^1$.

Supposons à présent qu'on a construit la séquence jusqu'à $s < n + m$, on a :

$$R_1^0 \ddagger R_2^0 \xrightarrow{\varepsilon}{}^* Q_1^1 \ddagger Q_2^1 \xrightarrow{t_1} R_1^1 \ddagger R_2^1 \ldots \xrightarrow{\varepsilon}{}^* Q_1^s \ddagger Q_2^s \xrightarrow{t_s} R_1^s \ddagger R_2^s$$

où

1. $R_1^s = \tilde{R}_1^j$ pour $j < n$ et $R_2^s \xrightarrow{t'} \tilde{R}_2^l$ pour $l < m$ avec $t' = t_2^l - t_s$, ou
2. $R_2^s = \tilde{R}_2^l$ pour $l < m$ et $R_1^s \xrightarrow{t'} \tilde{R}_1^j$ pour $j < n$ avec $t' = t_1^j - t_s$, ou
3. $R_1^s = \tilde{R}_1^j$ et $R_2^s = \tilde{R}_2^l$.

et $P_i^k \sim_b Q_i^{k+1}$ pour $k = 0, \ldots, s-1$ et $P_i^k \sim_b R_i^k$ pour $k = 0, \ldots, s$.

Pour déterminer $P_i^{s+1}$, $R_i^{s+1}$, $Q_i^{s+1}$ et $t_{s+1}$ il faut distinguer trois cas. Supposons que $R_1^s = \tilde{R}_1^j$ pour $j < n$ et $R_2^s \xrightarrow{t'} \tilde{R}_2^l$ pour $l < m$ avec $t' = t_2^l - t_s$. Les autres deux cas sont identiques.

On définit $Q_1^{s+1} = \bar{R}_1^{j+1}$ et $Q_2^{s+1} = R_2^s$.

Donc, $R_i^s \xrightarrow{\varepsilon}{}^* Q_i^{s+1}$, et d'après le lemme A.11, $R_1^s \ddagger R_2^s \xrightarrow{\varepsilon}{}^* Q_1^{s+1} \ddagger Q_2^{s+1}$.

Soit $t_{s+1} = \min(t', t_1^{j+1})$. On a trois cas :

1. Si $t_{s+1} = t_1^{j+1} < t'$ on définit $R_1^{s+1} = \tilde{R}_1^{j+1}$ et $R_2^{s+1}$ est tel que $R_2^{s+1} \xrightarrow{t''} \tilde{R}_2^l$ où $t'' = t_2^l - t_{s+1}$.
   De plus, on définit $P_1^{s+1} = \bar{P}_1^{j+1}$ et $P_2^{s+1}$ est tel que $P_2^{s+1} \xrightarrow{t''} \bar{P}_2^l$.
   Donc, $P_1^{s+1} \sim_b R_1^{s+1}$, et d'après le lemme A.10, $P_2^{s+1} \sim_b R_2^{s+1}$.
2. Si $t_{s+1} = t' < t_1^{j+1}$ on définit $R_2^{s+1} = \tilde{R}_1^l$ et $R_1^{s+1}$ est tel que $R_1^{s+1} \xrightarrow{t''} \tilde{R}_1^{j+1}$ où $t'' = t_1^{j+1} - t_{s+1}$.
   De plus, on définit $P_2^{s+1} = \bar{P}_2^l$ et $P_1^{s+1}$ est tel que $P_1^{s+1} \xrightarrow{t''} \bar{P}_1^{j+1}$.
   Donc, $P_2^{s+1} \sim_b R_2^{s+1}$, et d'après le lemme A.10, $P_1^{s+1} \sim_b R_1^{s+1}$.
3. Si $t_{s+1} = t_1^{j+1} = t'$ on définit $R_1^{s+1} = \tilde{R}_1^{j+1}$ et $R_2^{s+1} = \tilde{R}_1^l$.
   De plus, on définit $P_1^{s+1} = \bar{P}_1^{j+1}$ et $P_2^{s+1} = \bar{P}_2^l$.
   Donc, $P_1^{s+1} \sim_b R_1^{s+1}$, et $P_2^{s+1} \sim_b R_2^{s+1}$.

Donc, $Q_1^{s+1} \ddagger Q_2^{s+1} \xrightarrow{t_{s+1}} R_1^{s+1} \ddagger R_2^{s+1}$.

Par conséquent, $R_1^s \ddagger R_2^s \xrightarrow{\varepsilon}{}^* Q_1^{s+1} \ddagger Q_2^{s+1} \xrightarrow{\hat{t}_{s+1}} R_1^{s+1} \ddagger R_2^{s+1}$.

On peut donc répéter cette méthode de construction, on a qu'à chaque pas, $l$ et $j$ augmentent au moins 1, et par conséquent, $l$ et $j$ atteignent les valeurs $m$ et $n$, respectivement. Pour $r = m+n$, on obtient :

$$R_1^0 \ddagger R_2^0 \xrightarrow{\varepsilon}{}^* Q_1^1 \ddagger Q_2^1 \xrightarrow{t_1} R_1^1 \ddagger R_2^1 \ldots \xrightarrow{\varepsilon}{}^* Q_1^r \ddagger Q_2^r \xrightarrow{t_r} R_1^r \ddagger R_2^r$$

où $t = t_1 + \ldots + t_r$ et $P_i^k \sim_b Q_i^{k+1}$ pour $k = 0, \ldots, r-1$ et $P_i^k \sim_b R_i^k$ pour $k = 0, \ldots, r$. ■

Nous pouvons montrer à présent le lemme A.7 qui dit que pour tout terme $P \in \mathrm{ATP}^r$, les modèles $\mathcal{T}[\![P]\!]$ et $\mathcal{T}[\![P]\!]_\varepsilon$ sont équivalents modulo la relation $\sim_b$.

**Preuve. (Lemme A.7)**

Soient $T = \mathcal{T}[\![P]\!]$, $\hat{T} = \mathcal{T}[\![P]\!]_\varepsilon$. On prouve par induction sur la structure de $P$ qu'il existe une bisimulation de branchement $\mathcal{R}$ pour $T$ et $\hat{T}$.

La preuve est simple pour les termes **idle**, $X \in Var$, $aP_1$, $\mathsf{d}(a)P_1$ et **restrict** $A$ **in** $P_1$. Pour $P_1 + P_2$ et $P_1 \,||\, P_2$ la preuve suit immédiatement des lemmes A.12 et A.13.

On montre en détail pour l'opérateur watchdog et pour la récursion. Soient $T_i = \mathcal{T}[\![P_i]\!]$ et $\hat{T}_i = \mathcal{T}[\![P_i]\!]_\varepsilon$ pour $i = 1, 2$.

• Soit $P$ le terme $P_1$ **watchdog**$[A](t)\ P_2$. Comme $P \in \mathrm{ATP}^r$, alors $t > 0$.

Par h.i. il existe un bisimulation de branchement $\mathcal{R}_i$ pour $T_i$ et $\hat{T}_i$, $i = 1, 2$. On définit :

$$\begin{aligned}\mathcal{R} \;=\; & \mathcal{R}_1 \cup \mathcal{R}_2 \\ & \cup\{\langle P_1' \text{ watchdog}[A](t')\ P_2, R_1 \text{ watchdog}[A](t')\ P_2\rangle \mid \langle P_1', R_1\rangle \in \mathcal{R}_1 \wedge 0 < t' \leq t\} \\ & \cup\{\langle P_2, R_1 \text{ watchdog}[A](0)\ P_2\rangle \mid R_1 \in \hat{T}_1\}\end{aligned}$$

Nous montrons que $\mathcal{R}$ est une bisimulation de branchement pour $T$ et $\hat{T}$.

Puisque $\langle P_1, P_1\rangle \in \mathcal{R}_1$ on a $\langle P, P\rangle \in \mathcal{R}$.

**1.** Posons $\langle P_1' \text{ watchdog}[A](t')\ P_2, R_1 \text{ watchdog}[A](t')\ P_2\rangle \in \mathcal{R}$.

**i.** Supposons qu'ils existent $P'$ et $\ell \in Lab$ tels que $P_1' \text{ watchdog}[A](t')\ P_2 \xrightarrow{\ell} P'$.

wda1) $\ell = a \in A$ et alors $P_1' \xrightarrow{a} P'$.

Par h.i. ils existent $R$ et $Q_1$ tels que $R_1 \xrightarrow{\varepsilon}{}^* Q_1 \xrightarrow{a} R$, $\langle P_1', Q_1\rangle \in \mathcal{R}_1$ et $\langle P', R\rangle \in \mathcal{R}_1$.

Par lemma A.11 on a :

$$R_1 \text{ watchdog}[A](t')\ P_2 \xrightarrow{\varepsilon}{}^* Q_1 \text{ watchdog}[A](t')\ P_2$$

et d'après la règle wda1-$\varepsilon$) on a :

$$Q_1 \text{ watchdog}[A](t')\ P_2 \xrightarrow{a} R$$

et $\langle P_1' \text{ watchdog}[A](t')\ P_2, Q_1 \text{ watchdog}[A](t')\ P_2\rangle \in \mathcal{R}$ et $\langle P', R\rangle \in \mathcal{R}$.

wda2) $\ell = a \notin A$. Similaire au cas précédent.

wdt1) $\ell = t'$ et $P' = P_2$.

D'après la règle wdt1), il existe $P_1''$ tel que $P_1' \xrightarrow{t'} P_1''$.

Par h.i. ils existent $n > 0$ et $R_1^i, Q_1^i$ et $t_i > 0$ tels que $t_1 + \ldots + t_n = t'$, $R_1^0 = R_1$, et pour les uniques $P_1^i$ avec $P_1^0 = P_1'$ et $P_1^n = P_1''$ tels que

$$P_1^0 \xrightarrow{t_1} \ldots \xrightarrow{t_n} P_1^n$$

on a

$$R_1^0 \stackrel{\varepsilon}{\rightarrow}{}^* Q_1^1 \stackrel{t_1}{\rightarrow} R_1^1 \ldots \stackrel{\varepsilon}{\rightarrow}{}^* Q_1^n \stackrel{t_n}{\rightarrow} R_1^n$$

et $\langle P_1^i, R_1^i \rangle \in \mathcal{R}_1$ pour $i = 0, \ldots, n$ et $\langle P_1^i, Q_1^{i+1} \rangle \in \mathcal{R}_1$ pour $i = 0, \ldots, n-1$.
D'après la règle wdt1) on a :

$$P_1^0 \ \mathsf{watchdog}[A](t')\ P_2 \stackrel{t_1}{\rightarrow} P_1^1 \ \mathsf{watchdog}[A](t')\ P_2 \ldots \stackrel{t_n}{\rightarrow} P2$$

D'après le lemma A.11 et la règle wdt-$\varepsilon$) on a :

$$R_1^0 \ \mathsf{watchdog}[A](t')\ P_2 \stackrel{\varepsilon}{\rightarrow}{}^* Q_1^1 \ \mathsf{watchdog}[A](t')\ P_2 \stackrel{t_1}{\rightarrow} R_1^1 \ \mathsf{watchdog}[A](t' - t_1)\ P_2 \ldots$$

et

$$R_1^n \ \mathsf{watchdog}[A](0)\ P_2 \stackrel{\varepsilon}{\rightarrow}{}^* P_2$$

Soit $\bar{t}_i = \sum_{j \leq i} t_j$ et $\bar{t}_0 = 0$. Par construction, pour $i = 0, \ldots, n-1$, on a :

$$\langle P_1^i \ \mathsf{watchdog}[A](t' - \bar{t}_i)\ P_2, R_1^i \ \mathsf{watchdog}[A](t' - \bar{t}_i)\ P_2 \rangle \in \mathcal{R}$$

et

$$\langle P_1^i \ \mathsf{watchdog}[A](t' - \bar{t}_i)\ P_2, Q_1^{i+1} \ \mathsf{watchdog}[A](t' - \bar{t}_i)\ P_2 \rangle \in \mathcal{R}$$

et $\langle P_2, R_1^n \ \mathsf{watchdog}[A](0)\ P_2 \rangle \in \mathcal{R}$.

wdt2) $\ell = t'' < t$. Similaire au cas précédent.

wdt3) $\ell = t' + t''$ et $P_1' \stackrel{t}{\rightarrow} P_1''$ et $P_2 \stackrel{t}{\rightarrow} P_2'$. Il faut appliquer le même raisonnement que dans le cas précédent pour $P_2$ et la règle wd0-$\varepsilon$) pour prouver $R_1^n \ \mathsf{watchdog}[A](0)\ P_2 \stackrel{\varepsilon}{\rightarrow} P_2$.

**ii.** Supposons qu'ils existent $R$ et $\ell \in Lab$ tels que $R_1 \ \mathsf{watchdog}[A](t')\ P_2 \stackrel{\ell}{\rightarrow} R$. Dans ce sens, la preuve suit de la même manière sauf qu'elle est plus simple car il n'y a pas des transitions par $\varepsilon$ dans $T$.

**2.** Posons $\langle P_2, R_1 \ \mathsf{watchdog}[A](0)\ P_2 \rangle \in \mathcal{R}$. Dans ce cas, la propriété est vraie car la seule transition possible pour le terme $R_1 \ \mathsf{watchdog}[A](0)\ P_2$ est par $\varepsilon$ vers $P_2$ et $\langle P_2, P_2 \rangle \in \mathcal{R}$.

• Soit $P$ le terme $\mathsf{rec}X \cdot P_1$. Par h.i., il existe une bisimulation de branchement pour $T_1$ et $\hat{T}_1$. On définit :

$$\begin{aligned} \mathcal{R} \ = \ & \{\langle Q, Q \rangle \mid Q \in \mathrm{ATP}_\varepsilon^r\} \\ & \cup \{\langle P, \hat{P}[P/X] \rangle \mid \hat{P} \neq X \wedge \langle X, \hat{P} \rangle \in \mathcal{R}_1\} \\ & \cup \{\langle \bar{P}[P/X], \hat{P}[P/X] \rangle \mid \bar{P} \neq X \wedge \hat{P} \neq X \wedge \langle \bar{P}, \hat{P} \rangle \in \mathcal{R}_1\} \end{aligned}$$

Nous montrons que $\mathcal{R}$ est une bisimulation de branchement pour $T$ et $\hat{T}$. Nous constatons tout d'abord que $\langle P, P \rangle \in \mathcal{R}$.

**1.** Supposons $\langle P, \hat{P}[P/X]\rangle \in \mathcal{R}$. Alors, $\hat{P} \neq X$ et $\langle X, \hat{P}\rangle \in \mathcal{R}_1$. Nous avons donc, $X \xrightarrow{X}$ **idle** et ils existent $\hat{P}_0, \ldots \hat{P}_n$, tels que $\hat{P}_0$ est le terme $\hat{P}$ et :

$$\hat{P}_0 \xrightarrow{\varepsilon} \hat{P}_1 \ldots \xrightarrow{\varepsilon} \hat{P}_n \xrightarrow{\varepsilon} X$$

D'après le corollaire A.3, on a :

$$Q_0 \xrightarrow{\varepsilon} Q_1 \ldots \xrightarrow{\varepsilon} Q_n \xrightarrow{\varepsilon} P$$

avec $Q_i = \hat{P}_i[P/X]$ pour $i = 0, \ldots, n$. Par conséquent, $\hat{P}[P/X] \xrightarrow{\varepsilon}{}^{*} P$ et $\langle P, P\rangle \in \mathcal{R}$.

**2.** Supposons $\langle \bar{P}[P/X], \hat{P}[P/X]\rangle$. Alors, $\bar{P} \neq X$, $\hat{P} \neq X$, et $\langle \bar{P}, \hat{P}\rangle \in \mathcal{R}_1$.

**i.** Supposons $\bar{P}[P/X] \xrightarrow{\ell} P'$. Or, $\ell \neq X$ et par conséquent, d'après le corollaire A.4, il existe $R$ tel que $\bar{P} \xrightarrow{\ell} R$ et $P'$ est le terme $R[P/X]$.

Il faut distinguer deux cas :

a) Si $\ell = a$, alors ils existent $Q$ et $Q'$ tels que $\hat{P} \xrightarrow{\varepsilon}{}^{*} Q' \xrightarrow{a} Q$, $\langle R, Q'\rangle \in \mathcal{R}_1$ et $\langle R, Q\rangle \in \mathcal{R}_1$. D'après le corollaire A.3,

$$\hat{P}[P/X] \xrightarrow{\varepsilon}{}^{*} Q'[P/X] \xrightarrow{a} Q[P/X]$$

On a deux cas :

- Si $R = X$, alors $R[P/X] = P$ et donc $P' = P$.
  Or, $Q' \xrightarrow{a} Q$ et $\langle R, Q'\rangle \in \mathcal{R}_1$. Donc, par construction $\langle P, Q'[P/X]\rangle$.
  Pour $Q$ on a deux cas. Si $Q = X$, alors $Q[P/X] = P$ et donc $\langle P, P\rangle \in \mathcal{R}$. Si $Q \neq X$, alors par construction, $\langle P, Q[P/X]\rangle$.
- Si $R \neq X$, alors par construction, $\langle R[P/X], Q'[P/X]\rangle \in \mathcal{R}$ et $\langle R[P/X], Q[P/X]\rangle \in \mathcal{R}$.

b) Si $\ell = t$, alors ils existent $\hat{P}_i$, $\hat{Q}_i$ et $t_i$, avec $i = 1, \ldots, n$ tels que $t_1 + \ldots + t_n = t$, et pour $\hat{P}_0 = \hat{P}$, on a :

$$\hat{P}_0 \xrightarrow{\varepsilon}{}^{*} \hat{Q}_1 \xrightarrow{t_1} \hat{P}_1 \ldots \xrightarrow{t_n} \hat{P}_n$$

et pour les uniques $\bar{P}_i$, pour $i = 0, \ldots, n$ tels que $\bar{P}_0 = \bar{P}$, $\bar{P}_n = R$ et

$$\bar{P}_0 \xrightarrow{t_1} \bar{P}_1 \ldots \xrightarrow{t_n} \bar{P}_n$$

on a $\langle \bar{P}_i, \hat{P}_i\rangle \in \mathcal{R}_1$ pour $i = 0, \ldots, n$ et $\langle \bar{P}_i, \hat{Q}_{i+1}\rangle \in \mathcal{R}_1$ pour $i = 0, \ldots, n-1$.

D'après le corollaire A.3, on a :

$$\tilde{P}_0 \xrightarrow{\varepsilon}{}^{*} \tilde{Q}_1 \xrightarrow{t_1} \tilde{P}_1 \ldots \xrightarrow{t_n} \tilde{P}_n$$

où $\tilde{P}_i = \hat{P}_i[P/X]$ et $\tilde{Q}_i = \hat{Q}_i[P/X]$, pour $i = 0, \ldots, n$, et

$$R_0 \xrightarrow{t_1} R_1 \ldots \xrightarrow{t_n} R_n$$

où $R_i = \bar{P}_i[P/X]$, pour $i = 0, \ldots, n$.

Par construction, $\langle R_i, \tilde{P}_i\rangle \in \mathcal{R}$ et $\langle R_i, \tilde{Q}_{i+1}\rangle \in \mathcal{R}$, pour $o = 1, \ldots, n-1$.

Il faut montrer que $\langle R_n, \tilde{P}_n\rangle \in \mathcal{R}$. Il y a deux cas :

- Si $\bar{P}_n = X$, alors $R_n = P$ et donc $P' = P$.
  Pour $\hat{P}_n$ on a deux cas. Si $\hat{P}_n = X$, alors $\tilde{P}_n = P$, et donc $\langle P, P\rangle \in \mathcal{R}$. Si $\hat{P}_n \neq X$, alors par construction, $\langle P, \tilde{P}_n\rangle \in \mathcal{R}$.
- Si $\bar{P}_n \neq X$, alors $\hat{P}_n \neq X$, et donc, par construction, $\langle R_n, \tilde{P}_n\rangle \in \mathcal{R}$.

c) Si $\ell = Y \neq X$, alors $\bar{P} = Y$ et $R = P' = \mathsf{idle}$. Donc, $\hat{P} \xrightarrow{\varepsilon}{}^{*} Y \xrightarrow{Y} \mathsf{idle}$, et par définition $\langle \mathsf{idle}\rangle \mathsf{idle} \in \mathcal{R}$.

**ii.** Supposons $\hat{P}[P/X] \xrightarrow{\ell} P'$. Or, $\ell \neq X$ et par conséquent, d'après le corollaire A.4, il existe $R$ tel que $\hat{P} \xrightarrow{\ell} R$ et $P'$ est le terme $R[P/X]$. La preuve suit comme dans le cas précédent. ■

Nous montrons à présent le lemme A.8 qui dit que le modèle du graphe temporisé étendu $G = \mathcal{E}[\![P]\!]$ est équivalent, modulo la bisimulation forte, au modèle de $P$ avec des transitions $\varepsilon$.

**Preuve. (Lemme A.8)**

Soient $P \in \mathrm{ATP}^{r}_{\varepsilon}$ et $G = \mathcal{E}[\![P]\!]$. On note $T$ le système de transition définit par les termes accessibles à partir de $P$, i.e., $T = \mathit{Reach}(\mathcal{T}[\![P]\!]_{\varepsilon})$, et $\hat{T}$ le système de transition $\mathcal{T}[\![G]\!]$.

On montre par induction sur la structure de $P$ :

**A.** il existe une bisimulation forte $\mathcal{R}$ pour $T$ et $\hat{T}$, et

**B.** si $\langle P', (s, v)\rangle \in \mathcal{R}$ alors

- $P' \notin \mathit{Var}$ et $F(s) \cap \mathit{Var} = \emptyset$, ou
- il existe une variable $X \in \mathit{Var}$ telle que $P = X$ et $F(s) = \{X\}$.

La preuve est simple pour les termes $\mathsf{idle}$, $X \in \mathit{Var}$, $aP_1$ et $\mathsf{d}(a)P_1$. On montre en détail pour l'opérateur watchdog et pour la récursion. Pour les opérateurs $P_1 + P_2$, $P_1 \,||\, P_2$ et $\mathsf{restrict}\ A\ \mathsf{in}\ P_1$ la preuve suit de la même manière.

Soient $G = \langle S, H, E, s_I, \delta, F\rangle$, $G_i = \mathcal{E}[\![P_i]\!] = \langle S_i, H_i, E_i, s_{I_i}, \delta_i, F_i\rangle$, $T_i = \mathit{Reach}(\mathcal{T}[\![P_i]\!]_{\varepsilon})$ et $\hat{T}_i = \mathcal{T}[\![G_i]\!])$.

• On montre tout d'abord pour l'opérateur watchdog.

Soit $P$ le terme $P_1\ \mathsf{watchdog}[A](d)\ P_2$ pour $d \in \mathsf{N}, d > 0$ et $A \subseteq \mathit{Act}$. Par h.i. pour $i = 1, 2$ il existe une bisimulation forte $\mathcal{R}_i$ pour $T_i$ et $\hat{T}_i$.

Soit $x$ la nouvelle horloge, $x \notin H_1 \cup H_2$. On définit

$$\begin{aligned}\mathcal{R} \;=\;& \mathcal{R}_1 \cup \mathcal{R}_2 \cup \\ & \{\langle P_1'\ \mathsf{watchdog}[A](t)\ P_2, (s_1, s_{I_2}, v_1)\rangle \mid \langle P_1', (s_1, v_1)\rangle \in \mathcal{R}_1 \wedge v_1(x) = d - t \wedge t \leq d\}\end{aligned}$$

**A.** On prouve que $\mathcal{R}$ est une bisimulation forte pour $T$ et $\hat{T}$.

Tout d'abord on a $\langle P_1, (s_{I_1}, v)\rangle \in \mathcal{R}_1$ où $v(x') = 0$ pour tout $x' \in H$. Donc, $\langle P, (s_1, s_{I_2}, v)\rangle \in \mathcal{R}$.

Il faut distinguer deux cas :

**1.** $\langle P_1'\ \mathsf{watchdog}[A](t)\ P_2, (s_1, s_{I_2}, v_1)\rangle \in \mathcal{R}$ avec $t > 0$.

**i.** Supposons qu'ils existent $\bar{P}$ et $\ell \in \mathit{Lab}_{\varepsilon}$ tels que $P_1'\ \mathsf{watchdog}[A](t)\ P_2 \xrightarrow{\ell} \bar{P}$.

a) $\ell = a \in A \subseteq Act$ et $P'_1 \xrightarrow{a} \bar{P}$.

Or, $\langle P'_1, (s_1, v_1)\rangle \in \mathcal{R}_1$. Donc, par h.i., il existe $(s'_1, v'_1)$ tel que $(s_1, v_1) \xrightarrow{a} (s'_1, v'_1)$ et $\langle \bar{P}, (s'_1, v'_1)\rangle \in \mathcal{R}_1$.

Par conséquent, il existe $\langle s_1, a, \psi_1, H', s'_1\rangle \in E_1$ tel que $v_1$ satisfait $\psi_1$ et $v'_1 = v_1[H' := 0]$.

Par construction, $\langle (s_1, s_{I_2}), a, \psi_1 \wedge x < d, H', s'_1\rangle \in E$.

Puisque $x \notin H_1$ et $v_1(x) = d - t < d$ on a $v'_1(x) = d - t$ et $(s_1, s_{I_2}, v_1) \xrightarrow{a} (s'_1, v'_1)$.

De plus, $\langle \bar{P}, (s'_1, v'_1)\rangle \in \mathcal{R}$.

b) $\ell = \alpha \in Act_\varepsilon$ mais $\alpha \notin A$ et il existe $P''_1$ tel que $P'_1 \xrightarrow{\alpha} P''_1$ et $\bar{P} = P''_1$ watchdog$[A](t)\ P_2$.

Or, $\langle P'_1, (s_1, v_1)\rangle \in \mathcal{R}_1$. Donc, par h.i., il existe $(s'_1, v'_1)$ tel que $(s_1, v_1) \xrightarrow{\alpha} (s'_1, v'_1)$ et $\langle P''_1, (s'_1, v'_1)\rangle \in \mathcal{R}_1$.

Par conséquent, il existe $\langle s_1, \alpha, \psi_1, H', s'_1\rangle \in E_1$ tel que $v_1$ satisfait $\psi_1$ et $v'_1 = v_1[H' := 0]$.

Par construction, $\langle (s_1, s_{I_2}), \alpha, \psi_1 \wedge x < d, H', (s'_1, s_{I_2})\rangle \in E$.

Puisque $x \notin H_1$ et $v_1(x) = d - t < d$ on a $v'_1(x) = d - t$ et $(s_1, s_{I_2}, v_1) \xrightarrow{\alpha} (s'_1, s_{I_2}, v'_1)$.

De plus, $\langle \bar{P}, (s'_1, s_{I_2}, v'_1)\rangle \in \mathcal{R}$.

c) $\ell = t' \leq t$ et il existe $P''_1$ tel que $P'_1 \xrightarrow{t'} P''_1$ et $\bar{P} = P''_1$ watchdog$[A](t - t')\ P_2$.

Or, $\langle P'_1, (s_1, v_1)\rangle \in \mathcal{R}_1$. Donc, par h.i., il existe $(s'_1, v'_1)$ tel que $(s_1, v_1) \xrightarrow{t'} (s'_1, v'_1)$ et $\langle P''_1, (s'_1, v'_1)\rangle \in \mathcal{R}_1$.

On a donc $s'_1 = s_1$, $v'_1 = v_1 + t'$ et $v'_1$ satisfait $\delta(s_1)$.

Puisque $t' \leq t$ on a $v'_1(x) = v_1(x) + t' \leq v_1(x) + t = d$ et donc $v'_1$ satisfait la condition $x \leq d$.

Donc, $v'_1$ satisfait $\delta(s_1, s_{I_2})$.

Par conséquent, $(s_1, s_{I_2}, v_1) \xrightarrow{t'} (s_1, s_{I_2}, v'_1)$ et $\langle \bar{P}, (s_1, s_{I_2}, v'_1)\rangle \in \mathcal{R}$.

**ii.** Supposons qu'ils existent $(\bar{s}, \bar{v})$ et $\ell \in Lab_\varepsilon$ tels que $(s_1, s_{I_2}, v_1) \xrightarrow{\ell} (\bar{s}, \bar{v})$.

a) $\ell = a \in A$ et $(s_1, s_{I_2}, v_1) \xrightarrow{a} (\bar{s}, \bar{v})$.

Il existe $\langle (s_1, s_{I_2}), a, \psi_1 \wedge x < d, H', \bar{s}\rangle \in E$ tel que $v_1$ satisfait $\psi \wedge x < d$ et $\bar{v} = v_1[H' := 0]$.

Par construction, $\langle s_1, a, \psi_1, H', s'_1\rangle \in E_1$. Donc, $(s_1, v_1) \xrightarrow{a} (\bar{s}, \bar{v})$.

Or, $\langle P'_1, (s_1, v_1)\rangle \in \mathcal{R}_1$. Donc, par h.i., il existe $\bar{P}$ tel que $P'_1 \xrightarrow{a} \bar{P}$ et $\langle \bar{P}, (\bar{s}, \bar{v})\rangle \in \mathcal{R}_1$.

D'après la règle wda1), $P'_1$ watchdog$[A](t)\ P_2 \xrightarrow{\ell} \bar{P}$. De plus, $\langle \bar{P}, (\bar{s}, \bar{v})\rangle \in \mathcal{R}$.

b) $\ell = \alpha \in Act_\varepsilon$ mais $\alpha \notin A$ et il existe $(\bar{s}, \bar{v})$ tel que $(s_1, s_{I_2}, v_1) \xrightarrow{\alpha} (\bar{s}, \bar{v})$.

Il existe $\langle (s_1, s_{I_2}), \alpha, \psi, H', \bar{s}\rangle \in E$ tel que $v_1$ satisfait $\psi$ et $\bar{v} = v_1[H' := 0]$.

Or, $v_1(x) = d - t < d$. Donc, il existe $\langle s_1, \alpha, \psi_1, H', s'_1\rangle \in E_1$ tel que $\bar{s} = (s'_1, s_{I_2})$ et $\psi = \psi_1 \wedge x < d$. Donc, $s_1, v_1 \xrightarrow{\alpha} (s'_1, \bar{v})$.

Or, $\langle P'_1, (s_1, v_1)\rangle \in \mathcal{R}_1$. Donc, par h.i., il existe $\bar{P}$ tel que $P'_1 \xrightarrow{\alpha} \bar{P}$ et $\langle \bar{P}, (s'_1, \bar{v})\rangle \in \mathcal{R}_1$.

D'après la règle wda2), $P'_1$ watchdog$[A](t)\ P_2 \xrightarrow{\ell} \bar{P}$. De plus, $\langle \bar{P}, (\bar{s}, \bar{v})\rangle \in \mathcal{R}$.

c) $\ell = t' \leq t$ et il existe $(\bar{s}, \bar{v})$ tel que $(s_1, s_{I_2}, v_1) \xrightarrow{t'} (\bar{s}, \bar{v})$.

On a donc $\bar{s} = (s_1, s_{I_2})$, $\bar{v} = v_1 + t'$ et $\bar{v}$ satisfait $\delta(s_1, s_{I_2})$.

Par conséquent, $\bar{v}$ satisfait $\delta(s_1)$ et $(s_1, v_1) \xrightarrow{t'} (s_1, \bar{v})$.

Or, $\langle P_1', (s_1, v_1)\rangle \in \mathcal{R}_1$. Donc, par h.i., il existe $P_1''$ tel que $(s_1, v_1) \xrightarrow{t'} (s_1, \bar{v})$ et $\langle P_1'', (s_1, \bar{v})\rangle \in \mathcal{R}_1$.

D'après la règle wdt1), on a $P_1'$ $\mathsf{watchdog}[A](t)\ P_2 \xrightarrow{t'} P_1''$ $\mathsf{watchdog}[A](t - t')\ P_2$.

De plus, $\bar{v}(x) = v_1(x) + t' = d - (t - t')$ et donc $\langle P_1''$ $\mathsf{watchdog}[A](t - t')\ P_2, (\bar{s}, \bar{v})\rangle \in \mathcal{R}$.

**2.** Supposons $\langle P_1'$ $\mathsf{watchdog}[A](0)\ P_2, (s_1, s_{I_2}, v_1)\rangle \in \mathcal{R}$.

On a $v_1(x) = d$ et par conséquent $P_1'$ $\mathsf{watchdog}[A](0)\ P_2 \xrightarrow{\varepsilon} P_2$.

Il existe $v_2$ tel que $(s_1, s_{I_2}, v_1) \xrightarrow{\varepsilon} (s_{I_2}, v_2)$.

Par construction, $\langle (s_1, s_{I_2}), \varepsilon, x = d, H', s_{I_2}\rangle \in E$ où $H' = \{x_2 \in H_2 \mid x_2 \in F_2(s_{I_2})\}$.

On a donc $v_2 = v_1[H' := 0]$.

Par h.i. $\langle P_2, (s_{I_2}, v_2)\rangle \in \mathcal{R}_2$. Par conséquent, $\langle P_2, (s_{I_2}, v_2)\rangle \in \mathcal{R}$.

**B.** Supposons $\langle P', (s, v)\rangle \in \mathcal{R}$. Si $s \in S_1 \cup S_2$, alors $\langle P', (s, v)\rangle \in \mathcal{R}_1 \cup \mathcal{R}_2$, et donc la propriété suit par h.i. Si $s = (s_1, s_{I_2})$, alors par construction, $P' = P_1'$ $\mathsf{watchdog}[A](t)\ P_2$. Or, $F(s) = F(s_1) \cup \{x\}$. Puisque $P \in \mathrm{ATP}^r_\varepsilon$, on a $P_1' \notin Var$, car toute occurrence d'une variable libre dans $P$ est gardée par une action de $A$. Donc, par h.i., $F(s_1) \cap Var = \emptyset$. Par conséquent, $F(s) \cap Var = \emptyset$.

• On montre à présent pour la récursion.

Soit $P$ le terme $\mathsf{rec}X \cdot P_1$.

Par h.i. il existe une bisimulation forte pour $T_1$ et $\hat{T}_1$. On définit :

$$\begin{aligned}\mathcal{R} \quad = \quad & \{\langle P, (s_{I_1}, v_1)\rangle \mid \forall x_1 \in H_1.\ v_1(x_1) = 0\} \\ & \cup \{\langle R, (s, v)\rangle \mid \langle \bar{P}, (s, v)\rangle \in \mathcal{R}_1 \wedge \bar{P} \neq X \wedge R = \bar{P}[P/X]\}\end{aligned}$$

**A.** Soit $\langle R, (s, v)\rangle \in \mathcal{R}$.

**1.** On montre que si $R \xrightarrow{\ell} \tilde{P}$ alors il existe $(\tilde{s}, \tilde{v})$ tel que $(s, v) \xrightarrow{\ell} (\tilde{s}, \tilde{v})$ et $\langle \tilde{P}, (\tilde{s}, \tilde{v})\rangle \in \mathcal{R}$.

a) $\langle R, (s, v)\rangle = \langle P, (s_{I_1}, v_1)\rangle$ et pour tout $x_1 \in H_1$, $v_1(x_1) = 0$.

Supposons que $P \xrightarrow{\ell} \tilde{P}$. D'après la règle rec) on a $P_1[P/X] \xrightarrow{\ell} \tilde{P}$. Or, $X$ est gardée dans $P_1$. Donc, d'après le lemme A.4, il existe $P'$ tel que $\tilde{P} = P'[P/X]$ et $P_1 \xrightarrow{\ell} P'$.

Par h.i., $(s_{I_1}, v_1) \xrightarrow{\ell} (s', v')$ et $\langle P', (s', v')\rangle \in \mathcal{R}_1$.

Il faut distinguer deux cas. Si $P' \neq X$ alors par définition $\langle \tilde{P}, (s', v')\rangle \in \mathcal{R}$, et de plus, par h.i., $F(s') \cap Var = \emptyset$. Si $P' = X$, on a $\tilde{P} = P$, et par h.i., $F(s') = \{X\}$. Donc, d'après les règles de construction du graphe temporisé, $(s_{I_1}, v_1) \xrightarrow{\ell} (s_{I_1}, v_1)$ où $v_1 = v[H_1 := 0]$, et par définition, $\langle P, (s_{I_1}, v_1)\rangle \in \mathcal{R}$.

b) $R = \bar{P}[P/X]$ et $\langle \bar{P}, (s, v)\rangle \in \mathcal{R}_1$.

Supposons que $R \xrightarrow{\ell} \tilde{P}$. Or, $\bar{P} \neq X$. Donc, d'après le lemme A.4, il existe $P'$ tel que $\tilde{P} = P'[P/X]$ et $\bar{P} \xrightarrow{\ell} P'$.

Or, par h.i. $(s,v) \xrightarrow{\ell} (s',v')$ et $\langle P', (s',v')\rangle \in \mathcal{R}_1$.

Il faut distinguer deux cas. Si $P' \neq X$ alors par définition $\langle \tilde{P}, (s',v')\rangle \in \mathcal{R}$, et de plus, par h.i., $F(s') \cap Var = \emptyset$. Si $P' = X$, on a $\tilde{P} = P$, et par h.i., $F(s') = \{X\}$. Donc, d'après les règles de construction du graphe temporisé, $(s,v) \xrightarrow{\ell} (s_{I_1}, v_1)$ où $v_1 = v[H_1 := 0]$ et par définition, $\langle P, (s_{I_1}, v_1)\rangle \in \mathcal{R}$.

**2.** On montre que si $(s,v) \xrightarrow{\ell} (\tilde{s},\tilde{v})$ alors il existe $\tilde{P}$ tel que $R \xrightarrow{\ell} \tilde{P}$ et $\langle \tilde{P}, (\tilde{s},\tilde{v})\rangle \in \mathcal{R}$.

a) $\langle R, (s,v)\rangle = \langle P, (s_{I_1}, v_1)\rangle$ et pour tout $x_1 \in H_1$, $v_1(x_1) = 0$.

Supposons que $(s_{I_1}, v_1) \xrightarrow{\ell} (\tilde{s},\tilde{v})$. Par h.i., $P_1 \xrightarrow{\ell} \bar{P}$ et $\langle \bar{P}, (\tilde{s},\tilde{v})\rangle \in \mathcal{R}_1$. D'après le lemme A.3, $P_1[P/X] \xrightarrow{\ell} \bar{P}[P/X]$. Donc, d'après la règle rec), $P \xrightarrow{\ell} \bar{P}[P/X]$.

Il faut distinguer deux cas. Si $\bar{P} \neq X$ alors par définition $\langle \bar{P}[P/X], (\tilde{s},\tilde{v})\rangle \in \mathcal{R}$, de plus, $F(\tilde{s}) \cap Var = \emptyset$. Si $\bar{P} = X$, on a $\bar{P}[P/X] = P$, et par h.i., $F(\tilde{s}) = \{X\}$. Donc, d'après les règles de construction du graphe temporisé, $(s_{I_1}, v_1) \xrightarrow{\ell} (s_{I_1}, v_1)$ où $v_1 = v[H_1 := 0]$ et par définition, $\langle P, (s_{I_1}, v_1)\rangle \in \mathcal{R}$.

b) $R = \bar{P}[P/X]$ et $\langle \bar{P}, (s,v)\rangle \in \mathcal{R}_1$.

Supposons que $(s,v) \xrightarrow{\ell} (\tilde{s},\tilde{v})$. Par h.i., $\bar{P} \xrightarrow{\ell} Q$ et $\langle Q, (\tilde{s},\tilde{v})\rangle \in \mathcal{R}_1$. Par lemme A.3 $R[P/X] \xrightarrow{\ell} Q[P/X]$.

Il faut distinguer deux cas. Si $Q \neq X$ alors par définition $\langle Q[P/X], (\tilde{s},\tilde{v})\rangle \in \mathcal{R}$. Si $\tilde{P} = X$, on a $Q[P/X] = P$, et par h.i., $F(\tilde{s}) = \{X\}$. Donc, d'après les règles de construction du graphe temporisé, $(s,v) \xrightarrow{\ell} (s_{I_1}, v_1)$ où $v_1 = v[H_1 := 0]$ et par définition, $\langle P, (s_{I_1}, v_1)\rangle \in \mathcal{R}$.

**B.** Par construction. ■

## A.4 Consistance des équivalences pour TCTL

Nous montrons dans cette section que la relation d'équivalence $\sim_b$ est consistante pour un fragment significatif de la logique TCTL, au sens où deux états équivalents satisfont le même ensemble de formules.

Le fragment de TCTL que nous considérons, noté $\text{TCTL}_B$, est l'ensemble des formules obtenues à partir des propositions, des opérateurs logiques et des opérateurs temporels bornés.

**Définition A.7** Les formules de $\text{TCTL}_B$ son définies par la syntaxe suivante :

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\, \exists\mathcal{U}_{\#c}\, \varphi \mid \varphi\, \forall\mathcal{U}_{\#c}\, \varphi$$

où $p \in Pro$, $c \in \mathbb{N}$ et $\# \in \{<, \leq, >, \geq, =\}$. □

Nous supposons que deux états équivalents satisfont les mêmes propositions. On peut aisément vérifier que si $q \sim q'$ alors $q$ satisfait $\varphi$ si et seulement si $q'$ satisfait $\varphi$. Nous avons la proposition suivante.

**Proposition A.3** *Soit $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ un système temporisé. Pour tout $q, q' \in \mathcal{Q}$ tels que $q \sim q'$, pour tout ensemble de séquences $\Sigma \subseteq \Sigma_T$, et pour toute formule $\varphi \in \mathrm{TCTL}_B$, $q \models_\Sigma \varphi$ si et seulement si $q' \models_\Sigma \varphi$.*

Pour montrer un résultat similaire pour la relation d'équivalence $\sim_b$ nous exigeons que le système temporisé soit tel que si un état a des transitions temporelles alors il n'a pas des transitions par $\varepsilon$. Ceci n'est pas restrictif car, comme nous l'avons déjà montré, le modèle du graphe temporisé construit pour un terme d'ATP satisfait cette propriété. Nous avons la proposition suivante.

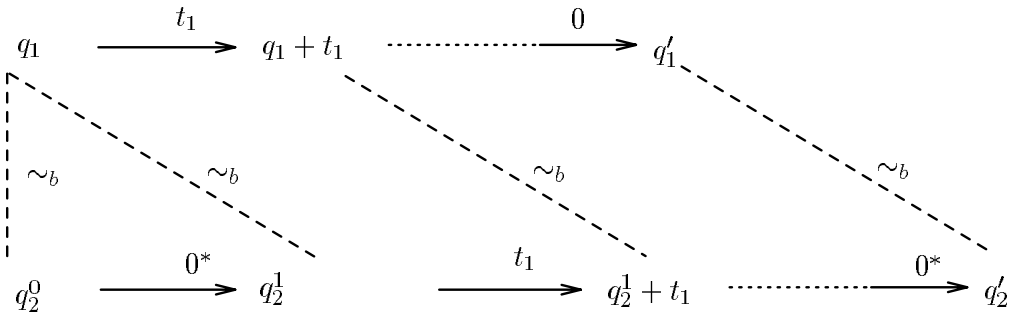
**Proposition A.4** *Soit $T = \langle \mathcal{Q}, \rightarrow, q_I \rangle$ un système temporisé tel que si $q \xrightarrow{t} q'$ alors $q \not\xrightarrow{\varepsilon}$. Pour tout $q, q' \in \mathcal{Q}$ tels que $q \sim_b q'$, pour tout ensemble de séquences $\Sigma \subseteq \Sigma_T$, et pour toute formule $\varphi \in \mathrm{TCTL}_B$, $q \models_\Sigma \varphi$ si et seulement si $q' \models_\Sigma \varphi$.*

Afin de prouver la proposition A.4, nous montrons tout d'abord le lemme suivant.

**Lemme A.14** *Soient $q_1, q_1' \in \mathcal{Q}$ et $t \in \mathsf{R}^+$ tels que $q_1 \xrightarrow{t} q_1 + t \xrightarrow{0} q_1'$. Pour tout $q_2 \in \mathcal{Q}$ tel que $q_1 \sim_b q_2$, ils existent $q_2^i \in \mathcal{Q}$ et $t_i \in \mathsf{R}^+$, avec $t_0 = 0$ et $t_1 + \ldots + t_n = t$, tels que $q_2^0 = q_2$ et*

$$q_2^0 \xrightarrow{0}{}^* q_2^1 \xrightarrow{t_1} q_2^1 + t_1 \ldots \xrightarrow{t_n} q_2^n + t_n \xrightarrow{0}{}^* q_2'$$

*et $q_1 + t_i \sim_b q_2^i + t_i$ pour $i = 0, \ldots, n$ et $q_1 + t_i \sim_b q_2^{i+1}$ pour $i = 0, \ldots, n-1$, et $q_1' \sim_b q_2'$.*




**Preuve.** D'après la définition de la relation $\sim_b$. ■

Le lemme A.14 dit que si l'état $q_1$ a un pas de durée $t$ vers l'état $q_1'$, alors tout état $q_2$ équivalent, modulo la relation d'équivalence de branchement, atteint un état $q_2'$ équivalent à $q_1'$ en un temps $t$, mais en général par une séquence de pas, telle que tous les états visités dans la séquence sont en correspondance, i.e., liés par la relation $\sim_b$, aux états visités dans le pas.

**Preuve. (Proposition A.4)** La preuve suit par induction sur la structure des formules, en appliquant les lemmes A.14 et A.10. ■

## A.5 Conclusions

Les résultats prouvés dans ce chapitre montrent que la méthode de traduction développée dans le chapitre 6 est consistante, au sens où le modèle du graphe temporisé construit et le modèle du terme d'ATP satisfont les mêmes formules d'un fragment significatif de TCTL. Ce fragment, noté $\text{TCTL}_B$, est l'ensemble des formules obtenues à partir des propositions, des opérateurs logiques et des opérateurs temporels bornés.

Pour montrer ce résultat nous avons proposé une nouvelle sémantique pour ATP. Cette sémantique introduit des transitions $\varepsilon$ pour modéliser les expirations des opérateurs watchdog. Nous avons montré que le graphe construit par la méthode de traduction est fortement équivalent au modèle du terme, si nous considérons cette sémantique.

De plus, nous avons proposé une nouvelle relation d'équivalence, inspirée de la relation de branchement [GW89, NMV90, Mou92], qui tient compte des transitions $\varepsilon$. Nous avons montré ensuite que cette relation est compatible avec la relation de bisimulation pour les termes d'ATP, au sens où deux termes sont fortement bisimilaires si et seulement si sont équivalents modulo la relation de branchement.

Ensuite, nous avons montré que deux états équivalents modulo la relation de branchement satisfont les mêmes formules de $\text{TCTL}_B$.

Finalement, ces résultats assurent la correction de l'approche proposée pour la vérification de systèmes temporisés : la traduction des spécifications temps-réels décrites en ATP, vers des graphes temporisés.