ADS Informatique

Les automates temporisés

Source : M. Fleury, Emmanuel. *Les automates temporisés avec mises à jour.* Mathématiques [math]. École normale supérieure de Cachan

Travail demandé

Il vous est demandé d'étudier puis de présenter le texte joint à travers un exposé de synthèse d'une durée comprise entre 15 et 20 minutes.

Si l'étude de la totalité du dossier et la préparation d'un exposé cohérent dans la durée impartie ne vous paraît pas possible, vous pouvez décider de vous limiter à une partie du dossier.

Remarques générales

1. Les textes proposés, quelle que soit leur origine, peuvent présenter des défauts (coquilles typographiques, négligences ou sous-entendus de l'auteur, voire erreurs...) qui, sauf exception, n'ont pas été corrigés.

2. Les textes proposés peuvent contenir des exercices qu'il n'est pas demandé de résoudre. Néanmoins, vous pouvez vous aider des énoncés de ces exercices pour enrichir votre exposé.

Chapitre 1

Les automates temporisés

The Rabbit say to itself : "Oh dear ! Oh dear ! I shall be too late !"

— Lewis Carroll, *Alice in Wonderland*

Les automates temporisés ont été introduits par Alur et Dill en 1990 [AD90, AD94]. Ce modèle, largement étudié [AHV93, ACH94, Wil94] et étendu depuis [BDGP98, DZ98, CG00], permet de faire de la vérification de propriétés de sûreté sur des systèmes de transitions qui utilisent des variables temporelles à valeurs dans $\mathbb{N}$, $\mathbb{Q}_+$ ou même $\mathbb{R}_+$. Les automates temporisés ont de plus été implantés dans plusieurs logiciels de vérification (KRONOS [DOTY96, BDM+98], UPPAAL [BLL+96, LPY97, PL00], CMC [LL98, CL00a]) et ont permis de valider ou corriger un certain nombre de protocoles ou algorithmes qui utilisent des variables temporelles dans leur déroulement [DOY94, DOY95, JLS96, HSLL97].

Nous introduisons dans ce chapitre les automates temporisés 'classiques' tels qu'ils sont utilisés dans les outils de vérification. Contrairement au modèle original d'Alur et Dill [AD94], le modèle 'classique' utilise des gardes diagonales ce qui rend le modèle plus complexe. Nous commencerons par définir quelques notions de bases relatives aux automates temporisés (langages temporisés, horloges, gardes, automates de Büchi), puis nous définirons les automates temporisés 'classiques' proprement dit. Enfin, nous nous intéresserons aux propriétés de ce modèle et plus particulièrement au problème du vide.

1.1 Langage temporisé

Un langage temporisé associe à chaque lettre (action) d'un mot, une date. Les mots sont donc formés d'une suite de couples comprenant une lettre choisie dans un alphabet fini Σ et un temps à valeur dans un domaine de temps $\mathbb{T}$, qui sera égal à $\mathbb{N}$, $\mathbb{Q}_+$ ou $\mathbb{R}_+$.

Notation 1. *Soit Z un ensemble quelconque, alors Z^* (resp. Z^ω) est l'ensemble des séquences finies (resp. infinies) d'éléments de Z. On note $Z^\infty = Z^* \cup Z^\omega$.*

Une *séquence temporisée* sur $\mathbb{T}$ est une séquence croissante de $\mathbb{T}^\infty$. Un *mot temporisé* ω est une séquence de couples $(a_i, t_i)_{i>0} \in (\Sigma \times \mathbb{T})^\infty$ telle que $(t_i)_{i>0}$ soit une séquence temporisée. Finalement, un langage temporisé L de Σ^∞ est la donnée d'un sous-ensemble de $(\Sigma \times \mathbb{T})^\infty$.

Définition 1. *Soit L un langage temporisé de $(\Sigma \times \mathbb{T})^\infty$, on appelle $Untimed(L)$ le langage défini par :*

$$Untimed(L) = \{(a_i)_{i>0} \in \Sigma^\infty \mid \exists (t_i)_{i>0} \text{ tel que } (a_i, t_i)_{i>0} \in L\} \tag{1.1}$$

Exemple 1. *Voici quelques exemples de mots temporisés sur l'alphabet $\Sigma = \{a, b, c\}$ et sur le domaine de temps $\mathbb{T} = \mathbb{R}_+$.*

1. $(b, 4)(a, 4.75)(c, 6.3)(a, 12)$,
2. $(a, e)(c, \pi)(a, 2 \times \pi)$,
3. $(a, 2)(a, 4)(a, 6)(a, 8) \dots$,
4. $(a, 1)(b, 2)(c, 3)(a, 12)(b, 13)(c, 14)(a, 102)(b, 103)(c, 104) \dots$.

1.2 Horloges et gardes

On appelle *horloges* des variables à valeurs dans le domaine de temps $\mathbb{T}$, qui évoluent de manière synchrone dans le temps. On considère X un ensemble fini d'horloges.

L'ensemble des *gardes* (ou *contraintes*) $\mathcal{C}(X)$ sur l'ensemble des horloges X est défini par la grammaire suivante :

$$\begin{aligned} \varphi ::= & \; x \sim c \mid x - y \sim c \mid \varphi \wedge \varphi \mid true, \\ & \text{avec } x, y \in X, c \in \mathbb{Q}_+, \sim \in \{<, \leq, =, \neq, \geq, >\} \end{aligned} \tag{1.2}$$

On notera cet ensemble $\mathcal{C}$, lorsqu'il n'y aura pas d'ambiguïté.

Remarque 1. *Il est à noter que les automates temporisés introduit par Alur et Dill à l'origine [AD90, AD94] ne possédaient pas de garde diagonales ($x - y \sim c$). Cependant, il fut prouvé par la suite que le modèle qui possédait des gardes diagonales et celui qui n'en possédait pas étaient* langage-équivalent *[BDGP98]. Malgré ce résultat, il est important pour la suite de distinguer ici les gardes générales, que nous venons de voir, des gardes non-diagonales.*

On appelle *gardes non diagonales* (*diagonal free guards*), les gardes qui n'incluent pas de contraintes sur des différences d'horloges. Plus formellement, l'ensemble $\mathcal{C}_{df}(X)$ des gardes diagonales est défini par la grammaire :

$$\begin{aligned} \varphi_{df} ::= & \; x \sim c \mid \varphi_{df} \wedge \varphi_{df} \mid true \\ & \text{avec } x \in X, c \in \mathbb{Q}_+, \sim \in \{<, \leq, =, \neq, \geq, >\} \end{aligned} \tag{1.3}$$

Exemples 2. *Voici quelques exemples de gardes sur $X = \{x, y, z\}$.*

1. $x = 5$,
2. $x < y$,
3. $x - y < 4$,
4. $(x < 10) \wedge (y - z > 2) \wedge (x - y = 3)$.

Notations 2. *Soit X un ensemble fini d'horloges.*

- *Une fonction $v : X \to \mathbb{T}$ est appelée une* valuation sur les horloges,

– *Si* $v \in \mathbb{T}^X$ *est une valuation et* $t \in \mathbb{T}$*, alors* $v + t \in \mathbb{T}^X$ *est la valuation qui correspond à l'écoulement d'une durée* t*, et qui est définie par :*

$$(v + t)(x) = v(x) + t, \ \forall x \in X \tag{1.4}$$

– *Si* v *est une valuation telle que* $(v(x))_{x \in X}$ *satisfait la garde* φ*, on dit que* v vérifie φ *et on note* $v \models \varphi$*,*
– *Soient* $Y \subseteq X$ *et* $f : Y \to \mathbb{T}$*, alors la valuation* $v[x \leftarrow f(x)/x \in Y]$ *est définie par :*

$$v[x \leftarrow f(x)/x \in Y](y) = \begin{cases} v(y), & si\ y \notin Y \\ f(y), & si\ y \in Y \end{cases} \tag{1.5}$$

1.3 Automates de Büchi

Avant de définir des automates permettant de reconnaître des langages temporisés, nous rappelons brièvement la notion classique d'automate de Büchi [Büc62, MN66, Tho90] pour les langages non temporisés.

Un *automate de Büchi* est un système de transitions défini par $\mathcal{B} = (\Sigma, Q, T, I, R)$, avec Σ un alphabet fini, Q un ensemble fini d'états, $T \subseteq Q \times \Sigma \times Q$ un ensemble fini de transitions, $I \subseteq Q$ un ensemble d'états initiaux, $R \subseteq Q$ un ensemble d'états répétés.

Soit un mot infini $\sigma = \sigma_1\sigma_2 \dots$ avec $\sigma_i \in \Sigma$ pour tout $i > 0$. On appelle r une *exécution* (*run*) de $\mathcal{B}$ sur σ une suite :

$$r = q_0 \xrightarrow{\sigma_1} q_1 \xrightarrow{\sigma_2} q_2 \xrightarrow{\sigma_3} q_3 \dots, \text{ avec } q_0 \in I, \text{ et } \forall i > 0, \ (q_{i-1}, \sigma_i, q_i) \in T \tag{1.6}$$

On note $rep(r) \subseteq Q$, l'ensemble des états infiniment répétés durant l'exécution r. Une exécution r est *acceptante* si et seulement si $rep(r) \cap R \neq \emptyset$ (*condition de Büchi*). Le mot $\sigma = \sigma_1\sigma_2\sigma_3 \dots$ est *accepté* par $\mathcal{B}$ s'il existe une exécution acceptante de $\mathcal{B}$ sur σ. Le langage des mots acceptés par $\mathcal{B}$ est noté $L(\mathcal{B})$.

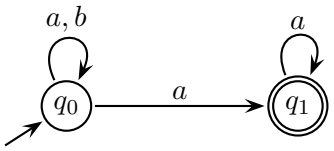

FIG. 1.1 – *Exemple d'automate de Büchi.*

Exemple 3. *Soit* $\mathcal{B} = (\Sigma, Q, T, I, R)$ *un automate de Büchi avec* $\Sigma = \{a, b\}$*,* $Q = \{q_0, q_1\}$*,* $T = \{(q_0, a, q_0), (q_0, b, q_0), (q_0, a, q_1), (q_1, a, q_1)\}$*,* $I = \{q_0\}$ *et* $R = \{q_1\}$ *(voir figure 1.1). Alors,* $\mathcal{B}$ *accepte le langage* $(a + b)^* a^\omega$*, constitué des mots infinis ayant un nombre fini de* b*.*

1.4 Automates temporisés

Un *automate temporisé* est un système de transitions défini par $\mathcal{A} = (\Sigma, Q, T, I, F, R, X)$ avec Σ un alphabet fini d'*actions*, Q un ensemble fini d'*états*, X un ensemble fini d'*horloges*, $I \subseteq Q$ l'ensemble des *états initiaux*, $F \subseteq Q$ l'ensemble des *états finaux*, $R \subseteq Q$ l'ensemble des

états répétés, et $T \subseteq Q \times [\mathcal{C}(X) \times \Sigma \times \mathcal{P}(X)] \times Q$ un ensemble fini de *transitions* (avec $\mathcal{P}(X)$ l'ensemble des parties de X.).

Ainsi, une transition est définie par un 5-uplet (q, g, a, R, q'), avec :
- $q, q' \in Q$ deux états,
- $g \in \mathcal{C}(X)$ une *garde*,
- $a \in \Sigma$ une *action*,
- $R \in \mathcal{P}(X)$ un sous-ensemble d'horloges *remises à zéro*.

Afin de caractériser les mots temporisés acceptés à partir du système de transitions que nous venons de définir, nous introduisons la notion de *chemin* et d'*exécution* sur un automate temporisé.

On appelle p (*path*) un *chemin* dans $\mathcal{A}$ une suite finie ou infinie de la forme :

$$p = q_0 \xrightarrow{g_1,\sigma_1,R_1} q_1 \xrightarrow{g_2,\sigma_2,R_2} q_2 \xrightarrow{g_3,\sigma_3,R_3} q_3 \dots, \quad \text{avec } q_0 \in I, \text{ et } \forall i > 0,\ (q_{i-1}, g_i, \sigma_i, R_i, q_i) \in T \tag{1.7}$$

On note $rep(p) \subseteq Q$ l'ensemble des états qui sont infiniment souvent répétés sur le chemin p. Le chemin fini (resp. infini) p est *acceptant* si l'état final est dans F (resp. si $rep(p) \cap R \neq \emptyset$, *condition de Büchi*).

Une *exécution* r (*run*) sur le chemin p est une suite de la forme :

$$r = \langle q_0, v_0 \rangle \xrightarrow[t_1]{g_1,\sigma_1,R_1} \langle q_1, v_1 \rangle \xrightarrow[t_2]{g_2,\sigma_2,R_2} \langle q_2, v_2 \rangle \dots, \text{ avec } \forall i \geq 0,\ v_i \in \mathbb{T}^X \tag{1.8}$$

avec $(t_i)_{i>0}$ une séquence temporisée et $(v_i)_{i\geq 0}$ des valuations d'horloges telles que :
- $v_0(x) = 0,\ \forall x \in X$,
- $\forall i > 0,\ v_{i-1} + (t_i - t_{i-1}) \models g_i$,
- $\forall i > 0$ et $\forall x \in X,\ v_i(x) = \begin{cases} 0, & si\ x \in R_i \\ v_{i-1}(x) + (t_i - t_{i-1}), & sinon \end{cases}$

On appelle les couples $\langle q_i, v_i \rangle$, des *états étendus*.

L'étiquette de r est le mot temporisé $(\sigma_1, t_1)(\sigma_2, t_2)\dots$, qui est *accepté* par l'automate temporisé $\mathcal{A}$. L'ensemble des mots qui sont l'étiquette d'une exécution sur un chemin acceptant de $\mathcal{A}$ forment le langage accepté (ou reconnu) par $\mathcal{A}$ et on le note $L(\mathcal{A}, \mathbb{T})$ ou plus simplement $L(\mathcal{A})$ lorsqu'il n'y a pas d'ambiguïté.

Remarque 2. *Dans ce modèle, il est à noter que l'on peut simuler tout automate temporisé utilisant des gardes générales par un automate temporisé utilisant uniquement des gardes non diagonales [BDGP98].*

Comme nous le verrons par la suite, cette simulation n'est plus possible dans le cadre du modèle des automates temporisés avec mises à jour (voir chapitre 2).

Exemple 4. *L'automate de la figure 1.2 page suivante est un exemple d'automate temporisé.*

Pour des raisons de lisibilité, lorsque la garde vaut true ou que l'ensemble des mises à jour vaut $\emptyset$, on n'écrit pas la composante en question.

Voici quelques exemples de mots acceptés par l'automate représenté sur la figure 1.2 :

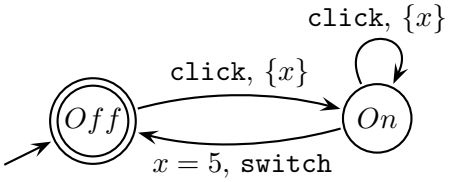

FIG. 1.2 – *Automate temporisé modélisant une minuterie.*

– *Le mot temporisé est :*

$$(\texttt{click}, 0)(\texttt{click}, 1)(\texttt{switch}, 6)\ldots(\texttt{click}, 7n)(\texttt{click}, 7n{+}1)(\texttt{switch}, 7n{+}6)\ldots$$

Une exécution acceptante est :

$$\langle Off, 0\rangle \xrightarrow[0]{\texttt{click},\{x\}} \langle On, 0\rangle \xrightarrow[1]{\texttt{click},\{x\}} \langle On, 0\rangle \xrightarrow[6]{x=5,\texttt{switch}} \langle Off, 5\rangle \ldots$$

– *Le mot temporisé est :*

$$(\texttt{click}, 0)(\texttt{click}, \sqrt{2})\ldots(\texttt{switch}, 5{+}\sqrt{2})(\texttt{click}, 7n)(\texttt{click}, 7n{+}\sqrt{2})(\texttt{switch}, 7n{+}5{+}\sqrt{2})\ldots$$

Une exécution acceptante est :

$$\langle Off, 0\rangle \xrightarrow[0]{\texttt{click},\{x\}} \langle On, 0\rangle \xrightarrow[\sqrt{2}]{\texttt{click},\{x\}} \langle On, 0\rangle \xrightarrow[5+\sqrt{2}]{x=5,\texttt{switch}} \langle Off, 5\rangle \ldots$$

Remarque 3. *Une variante du modèle des automates temporisés classiques consiste à introduire des actions silencieuses dans l'alphabet utilisé. Plus précisément, on considère des transitions dans $Q \times \mathcal{C}(X) \times (\Sigma \cup \{\varepsilon\}) \times \mathcal{P}(X) \times Q$. On appelle ce modèle les* automates temporisés avec ε-transitions *[AD90]. Ce modèle est strictement plus expressif que le modèle classique [BDGP98] et diffère sur la complexité de certains problèmes (voir section 1.5).*

Plus formellement, un automate temporisé avec ε-transitions est un automate temporisé classique tel que $\mathcal{A} = (\Sigma \cup \{\varepsilon\}, Q, T, I, F, R, X)$ avec ε une action silencieuse et dont le langage acceptant $L(\mathcal{A})$ est l'ensemble des mots temporisés obtenus en projetant les étiquettes des exécutions acceptantes sur $(\Sigma \times \mathbb{T})^\infty$. Autrement dit, en supprimant toutes les lettres du type (ε, t) avec $t \in \mathbb{T}$, des mots acceptés. On peut remarquer qu'avec cette définition une exécution infinie peut correspondre à un mot temporisé fini dans le langage accepté.

1.5 Propriétés des automates temporisés

Le modèle que nous venons de définir a certaines propriétés intéressantes que nous résumons brièvement ci-dessous. La problème du vide sera repris plus en détails dans la section suivante.

– **Problème du vide** : Pour tout automate temporisé $\mathcal{A}$, le *problème du vide* consiste à décider de la valeur de vérité de l'assertion $L(\mathcal{A}) = \emptyset$.
Ce problème a été montré *Pspace*-complet pour la classe des automates temporisés classiques [AD94],

– **Universalité** : Ce problème est le dual du problème du vide. Pour tout automate temporisé $\mathcal{A}$, le *problème de l'universalité* consiste à décider la valeur de vérité de l'assertion $L(\mathcal{A}) = (\Sigma \times \mathbb{T})^\infty$.
Ce problème a été montré *indécidable* pour la classe des automates temporisés classiques [AD94],

- **Inclusion de langage** : Pour tout couple d'automates temporisés $\mathcal{A}$ et $\mathcal{B}$, le *problème de l'inclusion des langages* est de décider la valeur de vérité de l'assertion $L(\mathcal{A}) \subseteq L(\mathcal{B})$. Ce problème a été montré *indécidable* pour la classe des automates temporisés classiques [AD94] (on se ramène au problème de l'universalité),
- **Équivalence de langage** : Pour tout couple d'automates temporisés $\mathcal{A}$ et $\mathcal{B}$ et leurs langages associés $L(\mathcal{A})$ et $L(\mathcal{B})$, le *problème de l'équivalence des langages* est de décider la valeur de vérité de l'assertion $L(\mathcal{A}) = L(\mathcal{B})$.
  Ce problème a été montré *indécidable* pour la classe des automates temporisés classiques [AD94] (on se ramène à un problème de double inclusion des langages),
- **Traces non temporisées** : Pour tout automate temporisé $\mathcal{A}$ et un mot σ tel que $\sigma = \sigma_1\sigma_2\sigma_3 \cdots \in \Sigma^*$. Le *problème des traces non temporisées* est de vérifier que l'on a $\sigma \in \mathit{Untimed}(\mathcal{A})$. Autrement dit, qu'il existe une exécution temporisée r sur $\mathcal{A}$ (voir équation page 34) telle que :

$$r = q_0 \xrightarrow[t_1]{g_1,\sigma_1,R_1} q_1 \xrightarrow[t_2]{g_2,\sigma_2,R_2} q_2 \xrightarrow[t_3]{g_3,\sigma_3,R_3} q_3 \dots \tag{1.9}$$

  Ce problème a été montré *NP*-complet pour la classe des automates temporisés classiques [AKV98] et *Pspace*-complet pour la classe des automates temporisés avec ε-transitions [AKV98],
- **Traces temporisées** : Pour tout automate temporisé $\mathcal{A}$, un mot $\sigma = \sigma_1\sigma_2 \cdots \in \Sigma^*$ et une séquence temporisée $\tau = t_1t_2 \dots$ (voir problème page 31). Le *problème des traces temporisées* est de vérifier que l'on a $(\sigma, t) \in L(\mathcal{A})$. Autrement dit, qu'il existe une exécution temporisée r sur $\mathcal{A}$ (voir équation page 34) telle que :

$$r = q_0 \xrightarrow[t_1]{g_1,\sigma_1,R_1} q_1 \xrightarrow[t_2]{g_2,\sigma_2,R_2} q_2 \xrightarrow[t_3]{g_3,\sigma_3,R_3} q_3 \dots \tag{1.10}$$

  Ce problème a été montré *NP*-complet pour la classe des automates temporisés classiques [AKV98] et *Pspace*-complet pour la classe des automates temporisés contenant des ε-transitions [AKV98],
- **Génération d'estampillage temporel** : Pour tout automate temporisé $\mathcal{A}$, et un chemin p sur $\mathcal{A}$ (voir équation page 34), tel que :

$$\begin{gathered} p = q_0 \xrightarrow{g_1,\sigma_1,R_1} q_1 \xrightarrow{g_2,\sigma_2,R_2} q_2 \xrightarrow{g_3,\sigma_3,R_3} q_3 \dots \xrightarrow{g_n,\sigma_n,R_n} q_n, \\ \text{avec } q_0 \in I, \text{ et } \forall i \leq n,\ (q_{i-1}, g_i, \sigma_i, R_i, q_i) \in T \end{gathered} \tag{1.11}$$

  Le *problème de la génération d'estampillage temporel* est de vérifier s'il existe une exécution temporisée r sur $\mathcal{A}$ (voir équation page 34) telle que :

$$r = q_0 \xrightarrow[t_1]{g_1,\sigma_1,R_1} q_1 \xrightarrow[t_2]{g_2,\sigma_2,R_2} q_2 \xrightarrow[t_3]{g_3,\sigma_3,R_3} q_3 \dots \xrightarrow{g_n,\sigma_n,R_n} q_n \tag{1.12}$$

  Ce problème a été montré de complexité $O(n.m^2)$, avec n la longueur du mot et et m le nombre des horloges, pour la classe des automates temporisés classiques et celle des automates temporisés avec ε-transitions [AKV98].

1.6 Le problème du vide

Le problème du vide (voir section 1.5 page précédente) est un problème majeur en vérification. Il permet, notamment de vérifier l'accessibilité d'un état sur un automate. Ce test est très

utile dans la pratique pour vérifier la présence ou l'absence de certains comportements dans un modèle.

Notre but est de démontrer la décidabilité du problème du vide sur la classe des automates temporisés. Pour ce faire, nous allons montrer que nous pouvons toujours construire, à partir d'un automate temporisé quelconque, un automate de Büchi qui reconnaît le langage *atemporisé* (*untimed*, voir définition 1 page 32) de l'automate temporisé original. Étant donné que le problème du vide est décidable sur les automates de Büchi, on en déduit le théorème attendu. On appellera cet automate de Büchi particulier l'*automate des régions*. La preuve de ce résultat a été proposée par Alur et Dill [AD90, AD94]. Nous présentons ici une généralisation de cette preuve aux automates temporisés avec gardes diagonales par le biais des *régions d'horloges*.

Le but de cette section est de décrire une méthode de construction d'un automate des régions à partir d'un automate temporisé, tout en prouvant qu'il existe *toujours* un tel automate, quelque soit l'automate temporisé initial.

Nous commençons par introduire les *régions d'horloges*, puis le *graphe des régions*, et nous finirons par la construction de l'*automate des régions* proprement dit. Grâce à la remarque suivante, nous pouvons nous restreindre aux automates temporisés dont les gardes n'utilisent que des constantes entières.

Remarque 4. *Nous avons défini les contraintes sur les horloges par des comparaisons avec des constantes dans* $\mathbb{Q}$*. Pour étudier le problème du vide, on peut se restreindre à des constantes dans* $\mathbb{Z}$*. En effet, soit* $\mathcal{A}$ *un automate temporisé et soit* d *le ppcm (plus petit commun multiple) des dénominateurs des constantes apparaissant dans* $\mathcal{A}$*. Enfin, soit* $\mathcal{A}_d$ *l'automate temporisé obtenu à partir de* $\mathcal{A}$ *en multipliant toutes les constantes de* $\mathcal{A}$ *par* d*. Par construction, les constantes de* $\mathcal{A}_d$ *sont toutes entières. De plus, le mot temporisé* $(a_i, t_i)_{i>0}$ *est accepté par* $\mathcal{A}$ *si et seulement si* $(a_i, d.t_i)_{i>0}$ *est accepté par* $\mathcal{A}_d$ *[AD94]. Ainsi,* $L(\mathcal{A})$ *est vide si et seulement si* $L(\mathcal{A}_d)$ *est vide.*

1.6.1 Les régions d'horloges

Vérifier le vide sur un automate temporisé consiste à tester si *aucune* exécution étendue n'est acceptée. Or, on peut voir facilement qu'il existe un nombre infini de ces exécutions. Pour contourner cette difficulté nous allons abstraire le problème. Nous ne considérerons plus les valuations une à une, mais par groupes de valuations équivalentes. On appellera ces groupes de valuations des *régions*.

Plus précisément, une *région* est un ensemble de valuations qui sont équivalentes du point de vue des gardes de l'automate temporisé que l'on considère. On dit alors que l'ensemble de régions est *compatible* avec l'ensemble des gardes. Dans notre cas, nous allons étudier une partition particulière de $\mathbb{T}^X$ qui forme un ensemble de régions que nous montrerons compatibles avec n'importe quel ensemble de contraintes.

On se donne un ensemble fini d'horloges X et $\lambda = ((max_x)_{x\in X}, (max_{x,y})_{(x,y)\in X^2})$ un ensemble de constantes entières qui représentent respectivement les plus grandes contraintes sur les horloges (max_x) et sur les différences d'horloges $(max_{x,y})$. Nous allons construire une partition $\mathcal{R}_\lambda$ de $\mathbb{T}^X$. On définit deux types de contraintes sur les valuations (les *contraintes simples*, les *contraintes diagonales*) dépendant de λ. Les régions seront ensuite définies par la conjonction de ces contraintes.

Définition 2. *Soient un ensemble fini d'horloges X et un ensemble de constantes entières $\lambda = ((max_x)_{x\in X}, (max_{x,y})_{(x,y)\in X^2})$. Alors on appelle* contrainte simple *associée à λ une formule φ_x telle que :*

$$\begin{array}{rll} \varphi_x ::= & x = c & \\ & \mid \; c < x < c+1 & \\ & \mid \; max_x < x & \\ & & \text{Avec } c < max_x. \end{array} \tag{1.13}$$

Une contrainte diagonale *associée à λ est une formule $\varphi_{x,y}$ avec $(x,y) \in X^2$ telle que :*

$$\begin{array}{rll} \varphi_{x,y} ::= & max_{x,y} < x - y & \\ & \mid \; c < x - y < c+1 & \\ & \mid \; x - y = c & \\ & \mid \; x - y < -max_{y,x} & \\ & & \text{Avec } -max_{y,x} < c < max_{x,y}. \end{array} \tag{1.14}$$

Nous allons introduire quelques ensembles utiles par la suite.

Notations 3. *Soient X un ensemble fini d'horloges, $\lambda = ((max_x)_{x\in X}, (max_{x,y})_{(x,y)\in X^2})$ un ensemble de constantes entières, et $((\varphi_x)_{x\in X}, (\varphi_{x,y})_{(x,y)\in X^2})$ un ensemble de contraintes issues de λ. On pose :*

- *$X_0 = \{x \in X \mid \varphi_x = (v(x) = c)\}$ l'ensemble des horloges dont la partie fractionnaire est* nulle *et dont la valeur est inférieure à max_x,*
- *$X_{frac} = \{x \in X \mid \varphi_x = (c < v(x) < c+1)\}$, l'ensemble des horloges dont la partie fractionnaire est* non nulle *et dont la valeur est inférieure à max_x.*
- *$X_\infty = \{x \in X \mid \varphi_x = (v(x) > max_x)\}$, l'ensemble des horloges dont la valeur est* supérieure à *max_x.*

Enfin, on définit une région d'horloges comme suit.

Définition 3. *Soient un ensemble fini d'horloges X et un ensemble de constantes entières $\lambda = ((max_x)_{x\in X}, (max_{x,y})_{(x,y)\in X^2})$. Alors, une* région *est la donnée de :*

- *$(\varphi_x)_{x\in X}$ un ensemble de contraintes simples associées à λ,*
- *$(\varphi_{x,y})_{(x,y)\in X^2}$ un ensemble de contraintes diagonales associées à λ,*
- *$\succ$ un préordre total sur l'ensemble des horloges $x \in X_{frac}$.*

La région $((\varphi_x)_{x\in X}, (\varphi_{x,y})_{(x,y)\in X^2}, \succ)$ représente l'ensemble de valuations $\alpha \subset \mathbb{T}^X$, tel que :

$$\alpha = \left\{ v \in \mathbb{T}^X \;\middle|\; \begin{array}{l} \forall x \in X,\ v(x) \models \varphi_x, \\ \forall (x,y) \in X^2,\ \text{avec } x \in X_\infty \text{ ou } y \in X_\infty,\ v(x) - v(y) \models \varphi_{x,y}, \\ \forall x, y \in X_{frac},\ x \succ y \Leftrightarrow frac(v(x)) \geq frac(v(y)) \end{array} \right\} \tag{1.15}$$

Avec, $frac(t) = t - \lfloor t \rfloor$ la partie fractionnaire de t.

Enfin, $\mathcal{R}_\lambda$ est la partition finie formée par toutes les régions engendrées par λ. Par commodité, nous noterons $\alpha = ((\varphi_x)_{x\in X}, (\varphi_{x,y})_{(x,y)\in X^2}, \succ)$.

Exemple 5. *Soit $X = \{x, y\}$ un ensemble d'horloges et $C = \{x < 2, y - x > 1, y = 1\}$ un ensemble de contraintes sur les horloges. On a $max_x = 2$ et $max_y = 1$, les deux constantes maximales respectivement sur x et sur y ainsi que $max_{x,y} = max_{y,x} = 1$.*

La partition $\mathcal{R}_\lambda$ associée est représentée sur la figure 1.3. On peut y distinguer trois types de régions :

- *Les régions ponctuelles, qui sont au nombre de neuf,*
- *Les régions linéaires, qui sont au nombre de vingt-trois,*
- *Les régions surfaciques, qui sont au nombre de quatorze.*

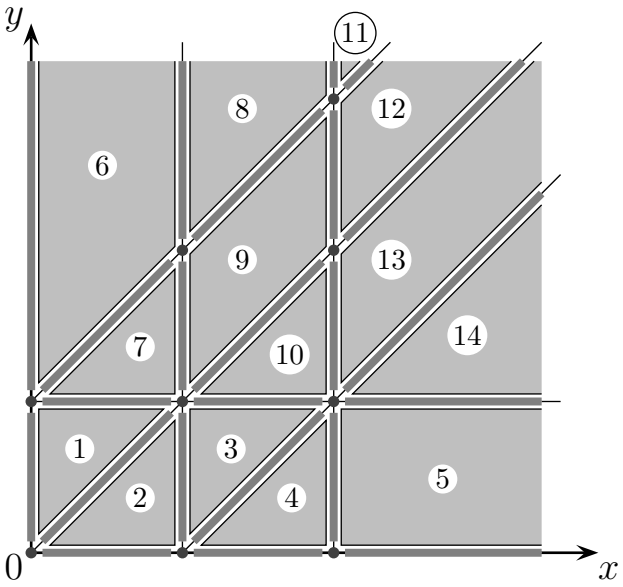

FIG. 1.3 – *Un exemple de régions avec deux horloges x et y.*

Voici quelques exemples de régions :

Région ponctuelle *:* $\alpha_{(2,1)} = \{(x = 2) \wedge (y = 1),\ \emptyset\}$,
Région linéaire bornée *:* $\alpha_{(1,1),(2,2)} = \{(1 < x < 2) \wedge (y > 1), \{(x \succ y), (y \succ x)\}\}$,
Région linéaire non bornée *:* $\alpha_{(2,1),(\infty,\infty)} = \{(x > 2) \wedge (y > 1) \wedge (x - y = 1), \{y \succ x\}\}$,
Région surfacique bornée *:* $\alpha_4 = \{(1 < x < 2) \wedge (0 < y < 1), \{x \succ y\}\}$,
Région surfacique non bornée *:* $\alpha_8 = \{(1 < x < 2) \wedge (y > 1) \wedge (x - y > 1), \{x \succ y\}\}$.

Nous allons à présent introduire la notion de *compatibilité* entre un ensemble de régions et des contraintes sur les horloges. Le but est de s'assurer que les régions ne regroupent que des valuations qui ont des comportements similaires face aux contraintes d'horloges que l'on se donne.

Définition 4. *Soit X un ensemble fini d'horloges, $C \subseteq \mathcal{C}(X)$ un ensemble de contraintes et $\mathcal{R}$ un ensemble fini de régions sur $\mathbb{T}^X$. Alors, $\mathcal{R}$ est* compatible *avec C si on a :*

$$\forall \psi \in C,\ \forall \alpha \in \mathcal{R},\ (\alpha \models \psi)\ ou\ (\alpha \models \neg\psi) \tag{1.16}$$

Nous allons nous intéresser maintenant au cas particulier de la partition $\mathcal{R}_\lambda$ de $\mathbb{T}^X$ que nous venons de définir. La proposition suivante est immédiate de par la définition de $\mathcal{C}(X)$ et de $\mathcal{R}_\lambda$.

Proposition 1. *Soient X, un ensemble fini d'horloges, $C \subseteq \mathcal{C}(X)$ un ensemble de contraintes sur les horloges et $\lambda = ((max_x)_{x\in X}, (max_{x,y})_{(x,y)\in X^2})$ un ensemble de constantes entières telles que :*

- *$\forall x \in X$, $max_x = \max\{c \mid (x \sim c) \in C\}$,*
- *$\forall (x,y) \in X^2$, on pose $C_{x,y}$ comme étant l'ensemble des contraintes $(x - y \sim c)$ et des contraintes $(y - x \sim c)$. Plus formellement, $C_{x,y} = \{(x - y \sim c) \in C$ et $(y - x \sim c) \in C\}$, et on définit :*

$$max_{x,y} = \begin{cases} \max\{c \mid (x - y \sim c) \in C_{x,y} \text{ et } (y - x \sim c) \in C_{x,y}\}, & \text{si } C_{x,y} \neq \emptyset \\ 0, & \text{si } C_{x,y} = \emptyset \end{cases}$$

Alors, $\mathcal{R}_\lambda$ est compatible avec C.

1.6.2 Le graphe des régions

Soit $\mathcal{R}$ un ensemble fini de régions formant une partition de $\mathbb{T}^X$. Le *graphe des régions* $\mathcal{G}$ associé à $\mathcal{R}$ établit les relations de précédence qu'ont les régions entre elles. Il est défini par $\mathcal{G} = (\mathcal{R}, \to)$ où la fonction de transition $\to \subseteq \mathcal{R} \times \mathcal{R}$ est définie à partir des fonctions de transitions suivantes :

$\xrightarrow{time}$: représente l'écoulement du temps :

$$\forall \alpha, \alpha' \in \mathcal{R},\ \alpha \xrightarrow{time} \alpha' \quad \overset{def}{\iff} \quad \exists v \in \alpha,\ \exists t > 0,\ v + t \in \alpha' \tag{1.17}$$

$\xrightarrow{reset}$: représente les remises à zéro d'horloges de X :

$$\forall \alpha, \alpha' \in \mathcal{R},\ \alpha \xrightarrow{reset} \alpha' \quad \overset{def}{\iff} \quad \exists v \in \alpha,\ \exists Y \subseteq X,\ \exists v' \in \alpha', \text{ tel que } v' = v[x \leftarrow 0 / x \in Y]. \tag{1.18}$$

La fonction de transitions $\to$ du graphe des régions $\mathcal{G} = (\mathcal{R}, \to)$ est définie par l'union des transitions *time* et *reset* :

$$\to = \xrightarrow{time} \cup \xrightarrow{reset} \tag{1.19}$$

Exemple 6. *Si l'on prend l'exemple de la figure 1.3 page précédente, le futur de la région surfacique* 3*, c'est à dire les régions accessibles via des transitions de type* time*, est constitué des régions suivantes :*

1. *région* $](1,1),(2,1)[$,
2. *région* 10,
3. *région* $](2,1),(2,2)[$,
4. *région* 13.

De même, les remises à zéro de la région surfacique 3*, c'est à dire les régions accessibles via une transition* reset*, sont les suivantes :*

- *Pour $Y = \{x\}$, on a la région* $](1,0),(2,0)[$,
- *Pour $Y = \{y\}$, on a la région* $](0,0),(0,1)[$,
- *Pour $Y = \{x,y\}$, on a la région* $\{(0,0)\}$

Cette définition du graphe des régions est cependant très générale. Notre but est à présent de contraindre le graphe des régions $\mathcal{G}$ afin de s'assurer que l'automate des régions $\mathcal{A}_R$ que nous allons construire capture l'ensemble de tous les comportements de l'automate original $\mathcal{A}$. À la fois pour les comportements temporisés et pour les "*remise à zéro*". Pour cela nous introduisons les conditions suivantes :

$$\forall \alpha, \alpha' \in \mathcal{R},\ \alpha \xrightarrow{time} \alpha' \Longrightarrow \forall v \in \alpha, \exists t \geq 0, v + t \in \alpha' \tag{$\dagger_1$}$$

$$\forall \alpha, \alpha' \in \mathcal{R},\ \alpha \xrightarrow{reset} \alpha' \Longrightarrow \forall v \in \alpha,\ \exists Y \subseteq X,\ \exists v' \in \alpha', \text{ tel que } v' = v[x \leftarrow 0 / x \in Y]. \tag{$\dagger_2$}$$

Il est immédiat que l'ensemble de tous les graphes des régions possibles ne vérifient pas forcément ($\dagger_1$) et ($\dagger_2$). ($\dagger_1$) est une condition assez naturelle, elle exprime le fait que deux valuations d'une région sont équivalentes et indistinguables vis-à-vis de l'écoulement du temps. La condition ($\dagger_2$) exprime, quant à elle, le fait que deux valuations d'une région sont équivalentes et indistinguables vis-à-vis d'une transition *reset*.

Ces deux conditions servent d'hypothèses de bases dans la démonstration du théorème 3 qui montre la décidabilité du problème du vide pour la classe des automates temporisés. Dans le cas des graphes des régions de type $\mathcal{G}_\lambda = (\mathcal{R}_\lambda, \rightarrow)$ où $\mathcal{R}_\lambda$ est la partition définie comme précédemment (voir définition 3 page 38), on peut montrer que ($\dagger_1$) et ($\dagger_2$) sont vraies moyennant une hypothèse sur les contraintes C.

Théorème 2. *Soient X un ensemble fini d'horloges et $\lambda = ((max_x)_{x\in X}, (max_{x,y})_{(x,y)\in X^2})$, un ensemble de constantes entières tel que pour tout couple $(x, y) \in X^2$, on ait $max_{x,y} \leq max_x$. Alors, le graphe des régions $\mathcal{G}_\lambda = (\mathcal{R}_\lambda, \rightarrow)$ vérifie les conditions ($\dagger_1$) et ($\dagger_2$).*

Nous montrons ($\dagger_1$) puis ($\dagger_2$) pour $\mathcal{G}_\lambda = (\mathcal{R}_\lambda, \rightarrow)$ avec $\lambda = ((max_x)_{x\in X}, (max_{x,y})_{(x,y)\in X^2})$ tel que pour $(x, y) \in X^2$, on ait $max_{x,y} \leq max_x$.

Lemme 2.1. *Soient X un ensemble fini d'horloges et $\lambda = ((max_x)_{x\in X}, (max_{x,y})_{(x,y)\in X^2})$, un ensemble de constantes entières. Alors pour toute région $\alpha = ((\varphi_x)_{x\in X}, (\varphi_{x,y})_{(x,y)\in X^2}, \succ) \in \mathcal{R}_\lambda$, il existe un ensemble fini de régions $\alpha_i = ((\varphi^i_x)_{x\in X}, (\varphi^i_{x,y})_{(x,y)\in X^2}, \succ_i)$ avec $i \geq 0$ telles que $\alpha_0 = \alpha$ et que l'on ait :*

1. $\forall \alpha' \in \mathcal{R}_\lambda,\ \alpha \xrightarrow{time} \alpha' \Rightarrow \exists i \geq 0,\ \alpha_i = \alpha'$,
2. $\forall i > 0,\ \forall v \in \alpha,\ \exists t > 0,\ v + t \in \alpha_i$.

Avant de démontrer formellement ce lemme, nous allons donner quelques idées intuitives de sa preuve. Tout d'abord, on ordonne les $(\alpha_i)_{i\geq 0}$ de façon chronologique suivant i. On a :

$$\forall i \geq 0,\ \alpha_i \xrightarrow{time} \alpha_{i+1} \tag{1.20}$$

On peut séparer les régions $(\alpha_i)_{i\geq 0}$ en trois catégories :

– **Les régions telles que $\mathbf{X_0}(\alpha_\mathbf{i}) \neq \emptyset$:**
Pour ce type de région, il existe au moins une horloge $x \in X$ telle que $\varphi_x = (v(x) = c)$. Dans ce cas, le temps est fixé par la valeur des horloges de $X_0(\alpha_i)$ et ne peut s'écouler sans que l'on sorte de la région. Par conséquent, si on laisse s'écouler un intervalle de temps, aussi petit soit-il, on aboutit dans une région qui substitue tous les $\varphi_x = (v(x) = c)$ par des $\varphi_x = (c < v(x) < c + 1)$. Le préordre, lui, est enrichi par toutes les horloges qui vérifiaient $\varphi_x = (v(x) = c)$ en tant qu'éléments minimaux.

- **Les régions telles que $\mathbf{X_0}(\alpha_\mathbf{i}) = \emptyset$ et $\mathbf{X_{frac}}(\alpha_\mathbf{i}) \neq \emptyset$:**
  Dans ce type de région, le temps peut s'écouler car aucune horloge $x \in X$ n'est telle que $\varphi_x = (v(x) = c)$ et il existe une horloge $x \in X$ telle que $\varphi_x = (c < v(x) < c+1)$. On détermine la région suivante en jouant sur le fait que ce sont les horloges qui correspondent aux éléments maximaux du préordre qui atteindront les premières une valeur entière. La nouvelle région s'obtient en substituant pour toutes les horloges maximales du préordre les $\varphi_x = (c < v(x) < c+1)$ par des $\varphi_x = (v(x) = c+1)$ et en retirant ces horloges du préordre.
- **Les régions telles que $\mathbf{X_0}(\alpha_\mathbf{i}) = \emptyset$ et $\mathbf{X_{frac}}(\alpha_\mathbf{i}) = \emptyset$:**
  Cette condition est la condition d'arrêt de l'algorithme qui construit les α_i. Dans ce cas, toutes les horloges $x \in X$ vérifient $\varphi_x = (v(x) > max_x)$. Comme, par définition, toutes les constantes c sont plus petites que max_x, la valeur de vérité de toutes contraintes $x \sim c$ ne changera plus. Par conséquent, la région qui capture ces comportements n'est pas bornée et complète ainsi le futur de α.

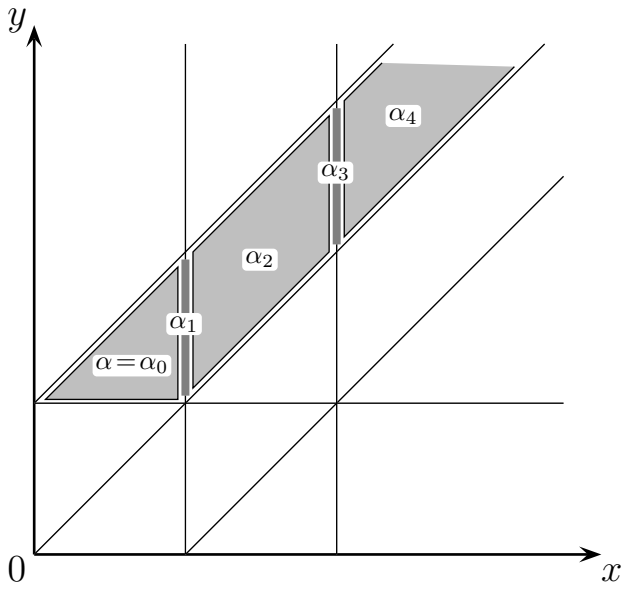

FIG. 1.4 – *Un exemple de régions $(\alpha_i)_{i\geq 0}$ reliées entre elles par la relation $\xrightarrow{time}$.*

Exemple 7. *Afin d'illustrer les différents types de régions, nous détaillons ici chacun des types des régions qui se trouvent sur la figure 1.4 :*

- $X_0(\alpha_0) \neq \emptyset$ *et* $X_{frac}(\alpha_0) \neq \emptyset$,
- $X_0(\alpha_1) = \emptyset$ *et* $X_{frac}(\alpha_1) \neq \emptyset$,
- $X_0(\alpha_2) \neq \emptyset$ *et* $X_{frac}(\alpha_2) \neq \emptyset$,
- $X_0(\alpha_3) = \emptyset$ *et* $X_{frac}(\alpha_3) \neq \emptyset$,
- $X_0(\alpha_4) = \emptyset$ *et* $X_{frac}(\alpha_4) = \emptyset$.

Démonstration. La preuve de ce lemme consiste à caractériser de façon inductive l'ensemble des $(\alpha_i)_{i\geq 0}$ à partir de $\alpha = ((\varphi_x)_{x\in X}, (\varphi_{x,y})_{(x,y)\in X^2}, \succ)$ puis à prouver les propriétés voulues sur cette caractérisation.

Plus précisément, on pose $\alpha_0 = \alpha$ et on construit α_{i+1} à partir de α_i de la façon suivante :

Soit $\alpha_i = ((\varphi^i_x)_{x\in X}, (\varphi^i_{x,y})_{(x,y)\in X^2}, \succ_i)$, la région $\alpha_{i+1} = ((\varphi^{i+1}_x)_{x\in X}, (\varphi^{i+1}_{x,y})_{(x,y)\in Y^2}, \succ_{i+1})$, est définie par :

– Si $X_0(\alpha_i) \neq \emptyset$, alors :

• $\varphi^{i+1}_x = \begin{cases} \varphi^i_x, & \text{si } x \notin X_0(\alpha_i) \\ (c < v_{i+1}(x) < c+1), & \text{si } \varphi^i_x = (v_i(x) = c) \text{ et } c < max_x \\ (v_{i+1}(x) > max_x), & \text{si } \varphi^i_x = (v_i(x) = max_x) \end{cases}$

• On ajoute les horloges contenues dans $X_0(\alpha_i)$ en tant qu'éléments minimaux du préordre $\succ_i$. Formellement, on a :

$$\succ_{i+1} = \succ_i \cup \{y \succ_i x \mid x \in X_0(\alpha_i) \text{ et } y \in X_{frac}(\alpha_{i+1})\} \tag{1.21}$$

– Si $X_0(\alpha_i) = \emptyset$ et $X_{frac}(\alpha_i) \neq \emptyset$, alors, on pose $X_{max}(\alpha_i)$ l'ensemble des éléments maximaux pour α_i de $X_{frac}(\alpha_i)$. Plus formellement, on a :

$$X_{max}(\alpha_i) = \{x \in X_{frac}(\alpha_i) \mid \forall y \in X_{frac}(\alpha_i),\ y \not\succ_i x\} \tag{1.22}$$

Si on note $\lfloor t \rfloor$, la partie entière inférieure de $t \in \mathbb{R}$, on a alors :

• $\varphi^{i+1}_x = \begin{cases} \varphi^i_x, & \text{si } x \notin X_{max}(\alpha_i) \\ (v_{i+1}(x) = \lfloor v_i(x) \rfloor + 1), & \text{si } x \in X_{max}(\alpha_i) \end{cases}$

• On retire du préordre toutes les horloges contenues dans $X_{max}(\alpha_i)$. Formellement, on a :

$$\succ_{i+1} = \succ_i \setminus \{x \succ_i y \mid x \in X_{max}(\alpha_i) \text{ et } y \in X\} \tag{1.23}$$

– Si $X_0(\alpha_i) = \emptyset$ et $X_{frac}(\alpha_i) = \emptyset$, alors $\alpha_i = \alpha_{i+1}$.

Maintenant que nous avons défini α_{i+1} en fonction de α_i, et comme l'ensemble $\mathcal{R}_\lambda$ est fini, on peut construire une suite stationnaire $(\alpha_i)_{0\leq i\leq n}$. Il est maintenant immédiat de vérifier que :

1. Par construction, on a :
   – $\alpha_0 = \alpha$,
   – $\forall i \geq 0,\ \alpha_i \xrightarrow{time} \alpha_{i+1}$,
   – $\forall \alpha' \in \mathcal{R}_\lambda,\ \alpha_i \xrightarrow{time} \alpha' \Rightarrow \alpha' = \alpha_i$ ou $\alpha' \xrightarrow{time} \alpha_{i+1}$.
   On peut, donc, en déduire que : $\forall \alpha' \in \mathcal{R}_\lambda,\ \alpha \to \alpha' \Rightarrow \exists i \geq 0,\ \alpha_i = \alpha'$
2. De plus, toujours par construction, on a :
   – $\alpha_0 = \alpha$,
   – $\forall v \in \alpha_i,\ \exists t > 0,\ v + t \in \alpha_{i+1}$.
   On peut, donc, en déduire par induction que : $\forall i \geq 0,\ \forall v \in \alpha,\ \exists t \geq 0,\ v + t \in \alpha_i$

Ce qui permet de conclure que la propriété $(\dagger_1)$ est vraie. □

Nous allons nous intéresser maintenant à la propriété $(\dagger_2)$. C'est dans cette seconde partie de la preuve que nous aurons besoin de l'hypothèse qui assure que pour tout couple $(x, y) \in X^2$, on a $max_{x,y} \leq max_x$.

Lemme 2.2. *Soient X un ensemble fini d'horloges et $\lambda = ((max_x)_{x\in X}, (max_{x,y})_{(x,y)\in Y^2})$, tels que pour tout couple $(x, y) \in X^2$, on ait $max_{x,y} \leq max_x$.*

Alors pour toute région $\alpha = ((\varphi_x)_{x\in X}, (\varphi_{x,y})_{(x,y)\in X^2}, \succ)$ de $\mathcal{R}_\lambda$, il existe un ensemble fini de régions $\alpha_Y = ((\varphi^Y_x)_{x\in X}, (\varphi^Y_{x,y})_{(x,y)\in X^2}, \succ_Y)$ telles que :

1. $\forall \alpha' \in \mathcal{R}_\lambda,\ \alpha \xrightarrow{reset} \alpha' \Rightarrow \exists Y \subseteq X,\ \alpha' = \alpha_Y$,

2. $\forall Y \subseteq X$, $\forall v \in \alpha$, $\exists v_Y \in \alpha_Y$, $v_Y = v[x \leftarrow 0 / x \in Y]$.

Démonstration. La preuve de ce lemme consiste à caractériser les ensembles α_Y pour un $Y \subseteq X$ donné. On prouve ensuite les deux propriétés.

Soit $Y \subseteq X$. On définit $\alpha_Y = ((\varphi_x^Y)_{x\in X}, (\varphi_{x,y}^Y)_{(x,y)\in X^2}, \succ_Y)$ par :

- $\succ' = \succ \setminus \{(x,y) \in X^2 \mid (x \in Y) \vee (y \in Y)\}$,
- $\varphi'_x = \begin{cases} \varphi_x, & \text{si } x \notin Y \\ (v'(x) = 0), & \text{sinon} \end{cases}$
- $\varphi'_{x,y}$ est tel que :
  - Si $x, y \notin Y$, alors $\varphi'_{x,y} = \varphi_{x,y}$,
  - Si $x, y \in Y$, alors $\varphi'_{x,y} = (v'(x) - v'(y) = 0)$,
  - Si $x \in Y$ et $y \notin Y$, alors :
    - Soit $\varphi_y = (v(y) = c)$ et on a : $\varphi'_{x,y} = (v'(x) - v'(y) = -c)$,
    - Soit $\varphi_y = (c < v(y) < c+1)$ et on a $\varphi'_{x,y} = (-(c+1) < v'(x) - v'(y) < -c)$,
    - Soit $\varphi_y = (max_y < v(y))$ et comme par hypothèse on a $max_{y,x} \leq max_y$, on en déduit que $-v(y) < -max_y \leq -max_{y,x}$ et donc que $0 - v(y) < -max_{y,x}$. Dans ce cas, on peut conclure que $\varphi'_{x,y} = (v'(x) - v'(y) < -max_{y,x})$.
  - Si $x \notin Y$ et $y \in Y$, alors, de façon symétrique, on a :
    - Soit $\varphi_x = (v(x) = c)$ et on a : $\varphi'_{x,y} = (v'(x) - v'(y) = c)$,
    - Soit $\varphi_x = (c < v(x) < c+1)$ et on a $\varphi'_{x,y} = (c < v'(x) - v'(y) < c+1)$,
    - Soit $\varphi_x = (max_x < v(x))$ et comme par hypothèse on a $max_{x,y} \leq max_x$, on en déduit que $max_{x,y} \leq max_x < v(x)$ et donc que $max_{x,y} < v(x) - 0$. Dans ce cas, on peut conclure que $\varphi'_{x,y} = (max_{x,y} < v'(x) - v'(y))$.

Ainsi, par construction, les $(\alpha_Y)_{Y\subseteq X}$ ainsi définis vérifient bien :

1. $\forall \alpha' \in \mathcal{R}_\lambda$, $\alpha \xrightarrow{reset} \alpha' \Rightarrow \exists Y \subseteq X$, $\alpha' = \alpha_Y$,
2. $\forall Y \subseteq X$, $\forall v \in \alpha$, $\exists v_Y \in \alpha_Y$, $v_Y = v[x \leftarrow 0 / x \in Y]$.

□

À partir des lemmes 2.1 et 2.2 et des constructions présentées dans les démonstrations, on définit la relation :

$$\rightarrow = \xrightarrow{time} \cup \xrightarrow{reset} \tag{1.24}$$

Pour finir, $\mathcal{G}_\lambda = (\mathcal{R}_\lambda, \rightarrow)$ ainsi défini vérifie bien le théorème 2 grâce aux lemmes 2.1 et 2.2 que nous avons établis.

1.6.3 L'automate des régions

L'automate des régions est un automate de Büchi classique $\mathcal{A}_R$ qui accepte le langage $Untimed(L(\mathcal{A}))$. Vérifier le vide sur $\mathcal{A}_R$ est clairement équivalent à vérifier le vide sur $\mathcal{A}$. Sommairement, $\mathcal{A}_R$ est obtenu en faisant le produit cartésien de $\mathcal{A}$ et de $\mathcal{G}_\lambda$, en effaçant les contraintes et les remises à zéro.

Définition 5. *Soient $\mathcal{A} = (\Sigma, Q, T, I, F, R, X)$ un automate temporisé, $C \subseteq \mathcal{C}$ l'ensemble des contraintes sur les horloges de $\mathcal{A}$ et $\lambda = ((max_x)_{x\in X}, (max_{x,y})_{(x,y)\in X^2})$ un ensemble des constantes entières tel que $\mathcal{R}_\lambda$ soit compatible avec C. Enfin, soit $\mathcal{G}_\lambda = (\mathcal{R}_\lambda, \rightarrow)$ le graphe des régions associé à λ.*

*Alors l'*automate des régions *$\mathcal{A}_R = (\Sigma', Q', T', I', R')$ est l'automate de Büchi tel que :*

- $\Sigma' = \Sigma$,

- $Q' = Q \times \mathcal{R}_\lambda$,
- $I' = I \times [v_0]$ *avec* $v_0(x) = 0$ *pour tout* $x \in X$,
- $R' = R \times \mathcal{R}_\lambda$,
- $T' \subseteq \langle Q \times \mathcal{R}_\lambda \rangle \times \Sigma \times \langle Q \times \mathcal{R}_\lambda \rangle$ *avec* $(\langle q, \alpha \rangle, \sigma, \langle q', \alpha' \rangle) \in T'$ *si et seulement si on a l'un des deux cas suivants :*
  1. $\exists (q, g, \sigma, r, q') \in T$ *telle que :*
     - $\alpha \subseteq g$,
     - $\exists \xrightarrow{reset} \in \rightarrow$, $\alpha \xrightarrow{reset} \alpha'$ *avec* $\alpha' = \alpha[r \leftarrow 0]$.
  2. $\exists \xrightarrow{time} \in \rightarrow$, $\alpha \xrightarrow{time} \alpha'$ *pour tout* $q = q'$ *et avec* $\sigma = \varepsilon$.

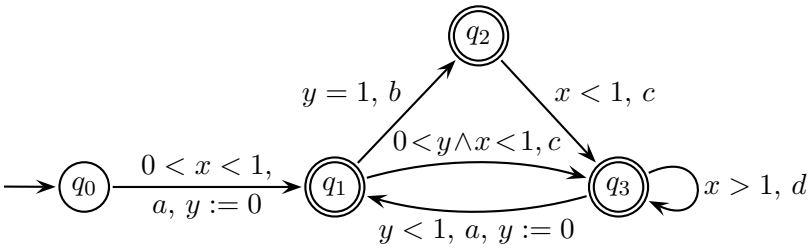

FIG. 1.5 – *Automate temporisé* $\mathcal{A}_0$.

Exemple 8. *Pour illustrer l'automate des régions, nous reprenons l'exemple de [AD94]*[1]*. Soit* $\mathcal{A}_0$ *l'automate temporisé représenté figure 1.5. L'automate des régions qui correspond à* $\mathcal{A}_0$ *est représenté sur la figure 1.6.*

L'état initial est $\langle q_0, (x = y = 0) \rangle$*. Les états répétés sont indiqués par une double ligne autour de l'état.*

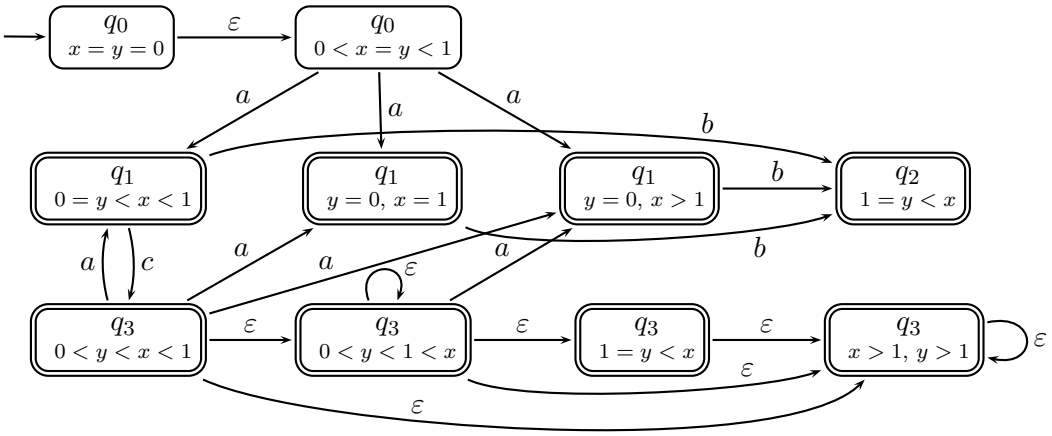

FIG. 1.6 – *Automate des régions associé à* $\mathcal{A}_0$.

L'automate des régions $\mathcal{A}_R$ une fois défini, en utilisant $\dagger_1$ et $\dagger_2$, on peut montrer le résultat suivant dû à Alur et Dill :

[1] Avec quelques corrections.

Théorème 3 ([AD94]). *Soient un automate temporisé $\mathcal{A} = (\Sigma, Q, T, I, F, R, X)$ et son automate des régions $\mathcal{A}_R = (\Sigma', Q', T', I', R')$. Alors, $\mathcal{A}_R$ accepte le langage $Untimed(L(A))$.*

Démonstration. Nous allons montrer que pour toute exécution de $\mathcal{A}$:

$$\langle q_0, v_0\rangle \xrightarrow[t_1]{g_1,\sigma_1,R_1} \langle q_1, v_1\rangle \xrightarrow[t_2]{g_2,\sigma_2,R_2} \langle q_2, v_2\rangle \dots$$

Il existe une exécution de $\mathcal{A}_R$:

$$\langle q_0, \alpha_0\rangle \xrightarrow{\varepsilon}{}^{*} \xrightarrow{\sigma_1} \langle q_1, \alpha_1\rangle \xrightarrow{\varepsilon}{}^{*} \xrightarrow{\sigma_2} \langle q_2, \alpha_2\rangle \dots$$

Telle que les régions α_i contiennent les valuations v_i. Puis nous montrerons la réciproque.

1. Soit une exécution de $\mathcal{A}$:

   $$\langle q_0, v_0\rangle \xrightarrow[t_1]{g_1,\sigma_1,R_1} \langle q_1, v_1\rangle \xrightarrow[t_2]{g_2,\sigma_2,R_2} \langle q_2, v_2\rangle \dots$$

   On suppose que l'on a déjà construit les i premières étapes de l'exécution de $\mathcal{A}_R$:

   $$\langle q_0, \alpha_0\rangle \xrightarrow{\varepsilon}{}^{*} \xrightarrow{\sigma_1} \langle q_1, \alpha_1\rangle \xrightarrow{\varepsilon}{}^{*} \xrightarrow{\sigma_2} \langle q_2, \alpha_2\rangle \dots \xrightarrow{\varepsilon}{}^{*} \xrightarrow{\sigma_{i-1}} \langle q_{i-1}, \alpha_{i-1}\rangle$$

   Tel que pour tout $0 \leq j < i$, on ait $v_j \in \alpha_j$.

   Soit la transition $(q_{i-1}, g_i, \sigma_i, R_i, q_i)$ de $\mathcal{A}$ et un temps $t \in \mathbb{T}$ tel que $(v_{i-1} + t) \models g_i$ et $v_i = (v_{i-1} + t)[0 \to x \mid x \in R_i]$. Alors, d'après la propriété $(\dagger_1)$ (p. 41), on peut, à partir de l'état $\langle q_{i-1}, \alpha_{i-1}\rangle$ de $\mathcal{A}_R$, atteindre l'état $\langle q_{i-1}, \alpha'_{i-1}\rangle$ avec $(v_{i-1} + t) \in \alpha'_{i-1}$ en prenant une succession de transitions temporelles ($\xrightarrow{\varepsilon}{}^{*}$). De plus, on sait par hypothèse que $(v_{i-1} + t) \models g_i$, donc $\alpha'_{i-1} \models g_i$ car le graphe des régions de $\mathcal{A}_R$ est compatible avec les contraintes de $\mathcal{A}$ et on suppose donc que $\alpha \models g$ ou $\alpha \models \neg g$. Donc, par définition de $\mathcal{A}_R$, on a :

   $$\langle q_{i-1}, \alpha_{i-1}\rangle \xrightarrow{\varepsilon}{}^{*} \langle q_{i-1}, \alpha'_{i-1}\rangle \xrightarrow{\sigma_i} \langle q_i, \alpha_i\rangle = \langle q_{i-1}, \alpha_{i-1}\rangle \xrightarrow{\varepsilon}{}^{*} \xrightarrow{\sigma_i} \langle q_i, \alpha_i\rangle$$

   Ce qui achève la première partie de la preuve.
2. Considérons à présent une exécution de $\mathcal{A}_R$:

   $$\langle q_0, \alpha_0\rangle \xrightarrow{\varepsilon}{}^{*} \xrightarrow{\sigma_1} \langle q_1, \alpha_1\rangle \xrightarrow{\varepsilon}{}^{*} \xrightarrow{\sigma_2} \langle q_2, \alpha_2\rangle \dots$$

   On suppose que l'on a déjà construit les i premières étapes de l'exécution de $\mathcal{A}$:

   $$\langle q_0, v_0\rangle \xrightarrow[t_1]{g_1,\sigma_1,R_1} \langle q_1, v_1\rangle \xrightarrow[t_2]{g_2,\sigma_2,R_2} \langle q_2, v_2\rangle \dots \xrightarrow[t_{i-1}]{g_{i-1},\sigma_{i-1},R_1} \langle q_{i-1}, v_{i-1}\rangle$$

   Tel que pour tout $0 \leq j < i$, on ait $v_j \in \alpha_j$.

   On suppose que l'exécution de $\mathcal{A}_R$ contient : $\langle q_{i-1}, \alpha_{i-1}\rangle \xrightarrow{\varepsilon}{}^{*} \langle q_{i-1}, \alpha'_{i-1}\rangle \xrightarrow{\sigma_i} \langle q_i, \alpha_i\rangle$. Nous avons donc la relation $\alpha_{i-1} \xrightarrow{time} \alpha'_{i-1}$. Or, par construction de $\mathcal{A}_R$, on en déduit qu'il existe dans l'automate temporisé $\mathcal{A}$ une transition $(q_{i-1}, g_i.\sigma_i, R_i, q_i)$ telle que $\alpha'_{i-1} \models g_i$ et $\alpha'_{i-1} \xrightarrow{reset} \alpha_i$.

   Par $(\dagger_1)$, on peut en conclure aussi qu'il existe un $t \in \mathbb{T}$ tel que $(v_{i-1} + t) \in \alpha'_{i-1}$ qui vérifie $(v_{i-1} + t) \models g_i$.
   Enfin, grâce à $(\dagger_2)$, on en déduit qu'il existe $v_i = (v_{i-1} + t)[0 \leftarrow x \mid x \in R_i]$ avec $v_i \in \alpha_i$. Ce qui conclu la deuxième et dernière partie de cette preuve.

□

Comme le problème du vide sur un automate fini (ou de Büchi) est décidable [HU79] et comme, de plus, le vide de $L(A)$ est équivalent à celui de $Untimed(L(A))$, on en déduit le corollaire suivant :

Corollaire 3.1. *Le problème du vide est décidable pour les automates temporisés classiques.*

Alur et Dill ont aussi établi la complexité du problème du vide sur un automate temporisé :

Théorème 4 ([AD94]). *Le problème de décider du vide sur un automate temporisé est* Pspace-*complet.*

Idée de la preuve. Voici une intuition de la preuve de la *Pspace*-complétude du problème du vide pour les automates temporisés.

- **Pspace-easy** : Soit une entrée de taille n qui représente un automate temporisé $\mathcal{A}$. Comme l'automate des régions est de taille exponentielle en nombre d'horloges, on ne peut écrire la représentation de $\mathcal{A}_R$ sur un ruban sans perdre l'aspect PSPACE. On construit donc, à la volée, les états de $\mathcal{A}_R$ dont on a besoin. Mais la représentation des régions sur le ruban est de taille polynômiale en n sur le ruban. On en déduit que le problème est au moins en espace polynômial. Comme la vérification du vide sur un automate de Büchi est en NLOGSPACE [Var96], on en déduit que le problème est en PSPACE.
- **Pspace-hard** : Comme dans [AD94], on réduit le problème du vide sur les automates temporisés avec mises à jour au problème de l'arrêt sur une machine de Turing dont la tête de lecture ne peut dépasser le marqueur de fin du mot d'entrée. Ce problème étant Pspace-complet, on en déduit la Pspace-complétude du problème du vide.

□

Conclusion

Nous avons présenté dans ce chapitre les automates temporisés 'classiques' qui peuvent être vu comme une extension des automates temporisés originaux d'Alur et Dill [AD90, AD94] auxquels on a ajouté des gardes diagonales. Le modèle des automates temporisés 'classiques' est le modèle utilisé dans les logiciels de vérification KRONOS [DOTY96, BDM+98], CMC [LL98, CL00a] et UPPAAL [BLL+96, LPY97, PL00]. Nous avons définis le cadre de ce modèle, puis nous avons introduit la notion d'automate temporisé. Nous avons ensuite fait un rapide tour d'horizon de la complexité de différents problèmes dans ce modèle. Enfin, nous avons exhibé une preuve détaillée de la décidabilité du problème du vide.

Dans le chapitre suivant nous allons proposer une extension de ce modèle qui permet une plus grande variété de mises à jour. Comme nous le verrons dans le reste de ce document, les propriétés de cette extension sont des plus intéressantes.

Chapitre 2

Les automates temporisés avec mises à jour

There is a theory which states that if ever anyone discovers exactly what the Universe is for and why it is here, it will instantly disappear and be replaced by something even more bizarre and inexplicable.

There is another which states that this has already happened.

— Douglas Adams, *The Hitchhiker's Guide To The Galaxy*

Dans le modèle classique des automates temporisés on autorise la remise à zéro d'un sous-ensemble d'horloges. Nous proposons dans ce chapitre un modèle qui élargit cette possibilité en autorisant des *mises à jour* des valeurs des horloges. On peut lors d'une transition donner, par exemple, des valeurs constantes aux horloges, ou leur attribuer la valeur d'une autre horloge, ou encore leur assigner des valeurs de manière non-déterministe. Nous appellerons ce modèle les *automates temporisés avec mise à jour* (*updatable timed automata*) [BDFP00a, BDFP00b].

2.1 Les mises à jour

Nous étendons le cadre des automates temporisés classiques décrit dans le chapitre 1 en permettant des transformations plus variées sur la valeur des horloges.

Les mises à jour déterministes rassemblent toutes les mises à jours qui affectent à une horloge donnée une constante $x := c$ ou la somme d'une horloge et d'une constante $x := y + c$. Ces affectations sont dites *déterministes* dans le sens où elles déterminent exactement la valeur de l'horloge sans aucune alternative. À l'inverse, les mises à jour non-déterministes associent à une horloge un intervalle d'affectations possibles dont les bornes sont constituées soit d'une constante (c), soit de la somme d'une horloge et d'une constante ($y + c$). Lors d'une mise à jour de ce type, l'horloge considérée peut prendre, de façon *non-déterministe*, n'importe quelle valeur de cet ensemble d'intervalles.

Nous commençons par définir la syntaxe de ces différentes classes de mises à jour, puis, nous définissons plus formellement leur sémantique.

2.1.1 Syntaxe des mises à jour déterministes

Une *mise à jour déterministe* consiste à affecter à une horloge donnée la valeur d'une constante ou la somme d'une horloge et d'une constante. Plus précisément, les mises à jour déterministes sont engendrées par la grammaire suivante :

$$\varphi ::= x := c \,|\, x := y + c, \text{ avec } c \in \mathbb{Q}, \text{ et } x, y \in X \tag{2.1}$$

2.1.2 Syntaxe des mises à jour non-déterministes

Une *mise à jour non-déterministe* est un ensemble de contraintes définissant un intervalle sur lequel l'horloge mise à jour doit prendre sa valeur. Plus précisément, les mises à jour non-déterministes sont engendrées par la grammaire suivante :

$$\varphi ::= x :\sim c \,|\, x :\sim y + c \,|\, \varphi \wedge \varphi \text{ avec } c \in \mathbb{Q},\ x, y \in X \text{ et } \ \sim \in \{<, \leq, \geq, >\} \tag{2.2}$$

2.1.3 Syntaxe des mises à jour générales

Une *mise à jour générale* φ, est la conjonction de mises à jour déterministes et/ou non-déterministes. Plus précisément, les mises à jour générales sont engendrées par la grammaire suivante :

$$\begin{array}{l} \varphi ::= x :\sim c \,|\, x :\sim y + c \,|\, \varphi \wedge \varphi \\ \qquad\qquad \text{avec } c \in \mathbb{Q},\ x, y \in X \text{ et } \ \sim \in \{<, \leq, =, \geq, >\} \end{array} \tag{2.3}$$

2.1.4 Sémantique des mises à jour

Nous allons à présent introduire la sémantique des mises à jour. D'une manière générale, une *mise à jour* peut être définie comme une fonction $up : \mathbb{T}^X \mapsto \mathcal{P}(\mathbb{T}^X)$ qui assigne à une valuation donnée un ensemble de valuations.

Nous appellerons up_x une *mise à jour locale* à x, une conjonction de mises à jours simples sur x dont la syntaxe est donnée par :

$$\begin{array}{l} up_x ::= x :\sim c \,|\, x :\sim y + c \,|\, up_x \wedge up_x, \\ \qquad\qquad \text{avec } c \in \mathbb{Q},\ y \in X \text{ et } \ \sim \{<, \leq, =, \geq, >\} \end{array} \tag{2.4}$$

On considère une mise à jour locale comme une fonction $[\![up_x]\!] : \mathbb{T}^X \mapsto \mathcal{P}(\mathbb{T})$ dont la sémantique est la suivante :

$$\begin{array}{lcl} [\![up_x^1 \wedge up_x^2]\!](v) & = & [\![up_x^1]\!] \cap [\![up_x^2]\!] \\ [\![x :\sim c]\!](v) & = & \{w \in \mathbb{T} \mid w \sim c\} \\ [\![x :\sim y + c]\!](v) & = & \{w \in \mathbb{T} \mid w \sim v(y) + c\} \end{array} \tag{2.5}$$

Remarque 5. *Si v est une valuation, on peut avoir $[\![up_x]\!](v) = \emptyset$.*

Par abus de langage, nous écrirons par la suite up_x à la place de $[\![up_x]\!]$.

Soient $Y \subseteq X$ un sous-ensemble d'horloges et $(up_x)_{x \in Y}$ un ensemble de mises à jour locales sur Y. Alors, la *mise à jour up* est la fonction $up : \mathbb{T}^X \mapsto \mathcal{P}(\mathbb{T}^X)$ associée à $(up_x)_{x \in Y}$ dont la sémantique est définie pour toute valuation $v \in \mathbb{T}$, par :

$$up(v) = \left\{ v' \in \mathbb{T}^X \mid \begin{array}{l} v'(x) \in up_x(v), \text{ si } x \in Y \\ v'(x) = v(x), \text{ si } x \notin Y \end{array} \right\} \tag{2.6}$$

La mise à jour est dite *valide* si $up(v) \neq \emptyset$. Nous noterons aussi $\mathcal{U}(X)$ l'ensemble de toutes les mises à jour possibles sur X. Enfin, par extension, si on définit $\mathcal{R}$ un ensemble de régions qui forment une partition de $\mathbb{T}^X$ et α une région, on pose :

$$up(\alpha) = \{\alpha' \in \mathcal{R} \mid \exists v \in \alpha,\ up(v) \cap \alpha' \neq \emptyset\} \tag{2.7}$$

Exemple 9. *Soient* $X = \{x, y, z\}$, $Y = \{x, y\}$, $up_x = (x := 3)$ *et* $up_y = (y :> 2) \wedge (y :< 10)$, *alors on aura :*

$$up(v) = \left\{ v' \in \mathbb{T}^X \;\middle|\; \begin{array}{l} v'(x) = 3 \\ 2 < v'(y) < 10 \\ v'(z) = v(z) \end{array} \right\} \tag{2.8}$$

2.2 Les automates temporisés avec mise à jour

De la même façon que les automates temporisés classiques, un *automate temporisé avec mise à jour* est un 7-uplet $\mathcal{A} = (\Sigma, Q, T, I, F, R, X)$ avec Σ un alphabet fini d'*actions*, Q un ensemble fini d'*états*, X un ensemble fini d'*horloges*, $I \subseteq Q$ l'ensemble des *états initiaux*, $F \subseteq Q$ l'ensemble des *états finaux*, $R \subseteq Q$ l'ensemble des *états répétés*, et un ensemble fini de *transitions* $T \subseteq Q \times [\mathcal{C}(X) \times \Sigma \times \mathcal{U}(X)] \times Q$. Ainsi, une transition est définie par un quintuplet (q, g, a, u, q'), tel que :

- $q, q' \in Q$, deux états,
- $g \in \mathcal{C}(X)$, une *garde*,
- $a \in \Sigma$, une *action*,
- $u \in \mathcal{U}(X)$, une *mise à jour*.

Les notions de *chemins* et d'*exécution* s'étendent sans difficulté aux automates temporisés avec mises à jour.

Un *chemin* (*path*) p dans $\mathcal{A} = (\Sigma, Q, T, I, F, R, X)$ est une suite finie ou infinie de la forme :

$$\begin{array}{c} p = q_0 \xrightarrow{g_1,\sigma_1,up_1} q_1 \xrightarrow{g_2,\sigma_2,up_2} q_2 \xrightarrow{g_3,\sigma_3,up_3} q_3 \dots, \\ \text{avec } q_0 \in I,\ \text{et } \forall i > 0,\ (q_{i-1}, g_i, \sigma_i, up_i, q_i) \in T \end{array} \tag{2.9}$$

Nous noterons $rep(p) \subset Q$ l'ensemble des états qui sont infiniment souvent répétés sur le chemin p. Le chemin fini (resp. infini) p est *acceptant* si l'état final est dans F (resp. si $rep(p) \cap R \neq \emptyset$, *condition de Büchi*).

Une *exécution* (*run*) r sur le chemin p est une suite de la forme :

$$r = \langle q_0, v_0 \rangle \xrightarrow[t_1]{g_1,\sigma_1,up_1} \langle q_1, v_1 \rangle \xrightarrow[t_2]{g_2,\sigma_2,up_2} \langle q_2, v_2 \rangle \dots,\ \text{avec } \forall i \geq 0,\ v_i \in \mathbb{T}^X \tag{2.10}$$

avec $(t_i)_{i>0}$ une séquence temporisée et $(v_i)_{i\geq 0}$ des valuations d'horloges telles que :

- $v_0(x) = 0$, $\forall x \in X$,
- $\forall i > 0$, $v_{i-1} + (t_i - t_{i-1}) \models g_i$,
- $\forall i > 0$, $v_i \in up_i(v_{i-1} + (t_i - t_{i-1}))$.

Les couples $\langle q_i, v_i \rangle$ sont encore appelés des *états étendus*.

L'étiquette de r est le mot temporisé $(\sigma_1, t_1)(\sigma_2, t_2)\dots$, qui est dit *accepté* par l'automate temporisé $\mathcal{A}$. L'ensemble des mots qui sont l'étiquette d'une exécution acceptante de $\mathcal{A}$ forment le langage accepté (ou reconnu) par $\mathcal{A}$ et on le note $L(\mathcal{A}, \mathbb{T})$ ou plus simplement $L(\mathcal{A})$.

Exemple 10. *L'automate de la figure 2.1 est un exemple d'automate temporisé avec mises à jour comprenant, entre autres, des mises à jour de type* $x := y$, $x := c$ *et* $x :> c$.

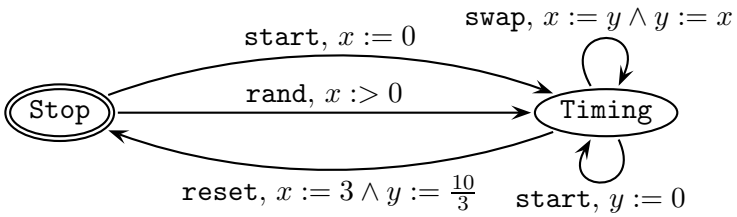

FIG. 2.1 – *Exemple d'automate temporisé avec mise à jour.*

Nous noterons $Aut(C(X), U(X))$ l'ensemble de tous les automates temporisés avec mises à jour construits avec des contraintes sur les horloges contenues dans $C(X)$ et des mises à jour dans $U(X)$. Nous écrirons simplement $Aut(C, U)$ lorsqu'il n'y aura pas d'ambiguïté.

Par la suite, nous aurons besoin de distinguer plusieurs sous-classes d'automates temporisés avec mises à jour. À cet effet, on note

- $\mathcal{C}$: L'ensemble de toutes les contraintes (voir la formule 1.2 page 32),
- $\mathcal{C}_{df}$: L'ensemble des contraintes non diagonales (voir la formule 1.3 page 32),
- $\mathcal{U}$: L'ensemble de toutes les mises à jour (voir la formule 2.3 page 50),
- $\mathcal{U}_{det}$: L'ensemble des mises à jour déterministes (voir équation 2.1 page 50).
- $\mathcal{U}^{+}_{det}$: L'ensemble des mises à jour déterministes avec des constantes sur $\mathbb{Q}_+$ (voir équation 2.1 page 50).
- $\mathcal{U}_{ndet}$: L'ensemble des mises à jour non-déterministes (voir équation 2.2 page 50).
- $\mathcal{U}^{+}_{ndet}$: L'ensemble des mises à jour non-déterministes avec des constantes sur $\mathbb{Q}_+$ (voir équation 2.2 page 50).
- $\mathcal{U}_{local}$: L'ensemble de toutes les mises à jours locales (up_x, avec $x \in X$),
- $\mathcal{U}_0$: L'ensemble des mises à jour du type $x := 0$ (remises à zéro),
- $\mathcal{U}_{cst}$: L'ensemble des mises à jour du type $x := c$.

On peut remarquer que $Aut(\mathcal{C},\mathcal{U}_0)$ correspond à l'ensemble des automates temporisés classiques.

2.3 Conclusion

Le modèle que nous proposons ici, étend les automates temporisés classiques en permettant des mises à jours en lieu et place des remises à zéro habituelles. Ce nouveau modèle soulève deux questions importantes :

- Apporte-t-il un pouvoir expressif plus important que le modèle original ?
- Le problème du vide est-il toujours décidable ? Et si oui, avec quelle complexité ?

Nous pourrions a priori nous intéresser à l'expressivité et à la décidabilité des modèles $Aut(C, U)$ constitués par les classes que nous avons définies ci-dessus. Cependant, il s'avère que le découpage est trop grossier pour permettre d'appréhender correctement ce modèle. Afin de mieux cerner les mécanismes qui régissent la décidabilité ou l'expressivité de ce modèle, nous allons nous intéresser à des classes de mises à jours plus précises présentées sur le tableau 2.1 page ci-contre. Chaque case de ce tableau présente un couple *mises à jour* et

contraintes d'horloges qu'il nous a semblé pertinent de considérer. Il est toutefois à noter que les remises à zéro des horloges sont possibles partout (c'est le cas classique). Les chapitres qui vont suivre utiliseront ce tableau comme un guide dans l'exploration des sous-classes du modèle que nous avons introduit ici.

	Mises à jour	Contraintes non diagonales	Contraintes générales
Déterministes	$x := 0$	PSPACE/TA	PSPACE/TA
	$x := c,\ c \in \mathbb{Q}_+$	?	?
	$x := y + c,\ c \in \mathbb{Q}_+$	?	?
	$x := y - 1$	?	?
	$x := y + c,\ c \in \mathbb{Q}$	?	?
Non-déterministes	$x :< c,\ c \in \mathbb{Q}_+$	?	?
	$x :> c,\ c \in \mathbb{Q}_+$	?	?
	$x :< y$	?	?
	$x :> y$	?	?
	$x :< y + c,\ c \in \mathbb{Q}_+$	?	?
	$x :> y + c,\ c \in \mathbb{Q}_+$	?	?
	$y + c <: x :< y + d,$ $c, d \in \mathbb{Q}_+$	?	?
	$y + c <: x :< z + d,$ $c, d \in \mathbb{Q}_+,\ y \neq z$	?	?

TAB. 2.1 – *Décidabilité et expressivité des différentes sous-classes du modèle.*

On peut remarquer dès à présent que nous avons fait le choix de séparer les différentes sous-classes suivant deux critères orthogonaux. Le premier critère est de séparer les *contraintes non-diagonales* des *contraintes générales*. Comme nous le verrons par la suite, la présence de contraintes du type $x - y \sim c$ induit d'importants changements dans la décidabilité du modèle. Cela peut paraître surprenant au premier abord lorsqu'on sait que pour la classe particulière des automates temporisés classiques, les contraintes $x - y \sim c$ étaient simulables par des contraintes de type $x \sim c$ [BDGP98]. Le deuxième critère de séparation concerne les mises à jour elles-mêmes. On distingue les contraintes déterministes des contraintes non-déterministes.

Nous pouvons compléter la ligne qui comporte la mise à jour de type $(x := 0)$ car elle correspond aux automates temporisés classiques dont nous avons rappelé au chapitre 1 la PSPACE-complétude. Nous noterons TA, l'ensemble des langages qui peuvent être générés par un automate temporisé classique. PSPACE/TA signifie que le problème du vide est PSPACE et que l'expressivité de ce modèle est égale à celle des automates temporisés classiques.

Dans les chapitres suivants, nous commencerons par étudier les aspects d'indécidabilité (chapitre 3) et de décidabilité (chapitre 4) du problème du vide sur le modèle des automates temporisés avec mises à jour. Nous présenterons, ensuite, nos résultats sur l'expressivité de ce modèle comparé aux automates temporisés classiques (chapitre 5).

Chapitre 3

Indécidabilité

A child of five could understand this...
Fetch me a child of five.

— Groucho Marx

Le présent chapitre s'attache à exposer les différents résultats d'indécidabilité que nous avons établis. En premier lieu, le modèle dans son ensemble est indécidable. On affinera cependant ce résultat en exhibant les sous-classes qui en sont à l'origine. Toutes les preuves d'indécidabilité reposent sur des réductions d'une machine de Minsky [Min67] en un automate temporisé avec des mises à jour. Pour conclure ce chapitre, nous compléterons le tableau 2.1 page 53 que nous avons proposé au chapitre 2 et nous discuterons des résultats.

3.1 La machine à deux compteurs

Nous allons rapidement rappeler ce qu'est une machine de Minsky et les propriétés qui sont utiles pour nos démonstrations (indécidabilité du problème du vide).

Définition 6. *Une* machine à deux compteurs *ou* machine de Minsky $\mathcal{M}$ *est un ensemble fini d'instructions* $(p_k)_{0\leq k\leq f}$ *sur deux compteurs entiers* x *et* y *dont la valeur initiale est* 0. *On distingue une instruction initiale* (p_0) *et une instruction finale* (p_f) *:*

- *Initialisation :* p_0 : `INIT; goto` p_1
- *Arret :* p_f : `HALT`

Et des instructions d'incrémentation et de décrémentation sur les compteurs :

- *Incrémentation* $(+1_i)$ *:* `p`: `i:=i+1 ; goto q`
- *Décrementation* (-1_i) *:*

```
p: if i>0
     then i:=i-1 ; goto q
     else goto q'
```

Avec `p`, `q` *et* `q'` *des instructions de* $\mathcal{M}$ *et* $i \in \{x, y\}$.

Exemple 11. *Voici un exemple de machine de Minsky.*

p_0 : `INIT; goto` p_1
p_1 : $x := x + 1$; `goto` p_3
p_2 : $y := y + 1$; `goto` p_2
p_3 : `if` $y > 0$
`then` $y := y - 1$; `goto` p_1
`else goto` p_2
p_4 : `HALT`

`INIT` correspond à la remise à zéro des deux compteurs x et y. Les instructions p_1 et p_2 réalisent chacune une incrémentation, et p_3 un décrémentation. Comme, on le voit facilement, cette machine de Minsky ne termine pas (i.e. elle n'atteint jamais l'état d'arrêt p_4).

Malgré son apparente simplicité, la machine de Minsky possède le même pouvoir d'expression qu'une machine de Turing. Minsky a d'ailleurs prouvé le théorème suivant :

Théorème 5 ([Min67]). *Le problème de l'arrêt est indécidable sur une machine de Minsky.*

L'intérêt de ce modèle est qu'il est extrêmement facile à simuler. Pour cela, il suffit d'exprimer l'opération d'incrémentation et de décrémentation avec test à zéro pour obtenir l'indécidabilité du problème de l'arrêt pour le modèle considéré. C'est donc ce modèle que nous allons simuler avec certaines sous-classes des automates temporisés avec mises à jour.

3.2 Indécidabilité des automates temporisés avec mises à jour

La première évidence est que le modèle des automates temporisés avec mises à jour dans son ensemble est indécidable. Il est en effet aisé de simuler une machine de Minsky avec des mises à jour $x := x + 1$ et $x := x - 1$ et des tests sur les horloges ($x = 0$ et $x > 0$).

Théorème 6. *Soit X un ensemble fini d'horloges contenant au moins 3 éléments. Alors, le problème du vide sur $Aut(\mathcal{C},\mathcal{U})$ est indécidable.*

Démonstration. La preuve revient à simuler une machine de Minsky avec des automates temporisés avec mises à jour. On considère un automate temporisé avec $X = \{x, y, z\}$. Les compteurs de la machine de Minsky sont représentés par les horloges x et y. On simule l'incrémentation en utilisant une mise à jour $x := x + 1$ (figure 3.1) et la décrémentation avec une mise à jour $x := x - 1$ (figure 3.2 page suivante). L'horloge z sert à figer le temps et à ne faire prendre aux compteurs que des valeurs entières. □

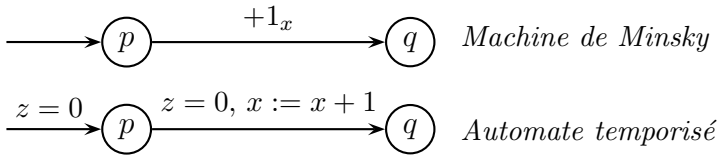

FIG. 3.1 – *Simulation d'une opération "$+1_x$".*

L'indécidabilité de l'ensemble du modèle étant établie, nous allons nous intéresser à certaines sous-classes strictes de $Aut(\mathcal{C},\mathcal{U})$ afin de mieux cerner l'origine de cette indécidabilité et de trouver quelques îlots de décidabilité.

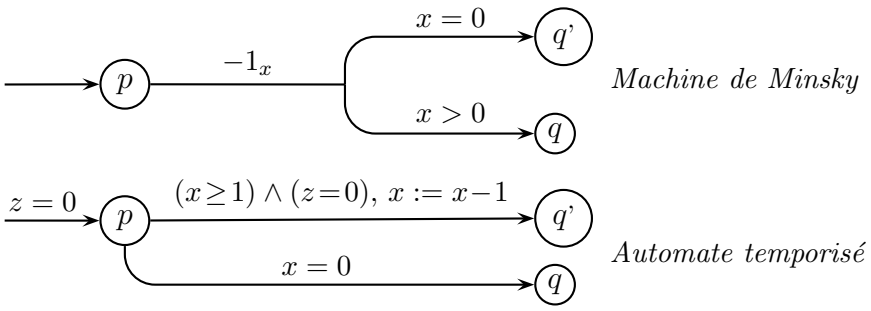

FIG. 3.2 – *Simulation d'une opération "-1_x".*

3.3 Les automates avec mises à jour $x := x - 1$

Nous allons commencer par nous intéresser aux mises à jour de type $x := x - 1$ avec des gardes non diagonales. Pour ce fragment précis, simuler l'opération de décrémentation est direct. L'incrémentation par contre est plus subtile.

Théorème 7. *Soit X un ensemble fini d'horloges tel que $\{x, y, z\} \subseteq X$ et un ensemble de mises à jour $U = \{x{:=}x-1,\ y{:=}y-1,\ z{:=}0\}$. Alors le problème du vide sur les sous-classes $Aut(\mathcal{C}, U)$ et $Aut(\mathcal{C}_{df}, U)$ est indécidable.*

Démonstration. La preuve consiste à simuler une machine de Minsky avec un automate temporisé avec mises à jour qui contiennent des gardes classiques et des mises à jour du type $x := x - 1$. On simule l'opération d'incrémentation ($+1_x$) en laissant écouler une unité de temps et en retranchant 1 à l'autre compteur (figure 3.3). L'opération de décrémentation (-1_x) est aisément simulable par les mises à jour $x := x - 1$ (figure 3.4 page suivante). Enfin, les tests à zéro sont obtenus par de simples gardes $x = 0$. □

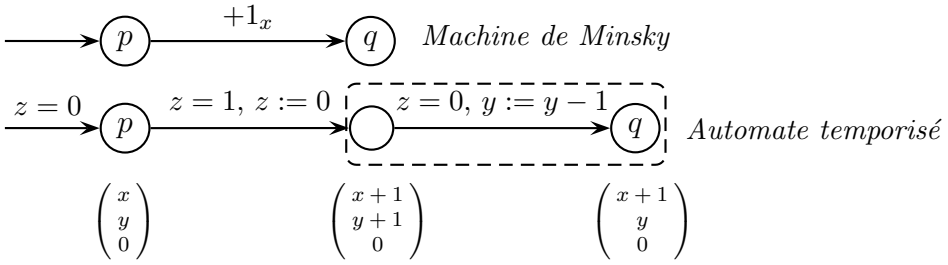

FIG. 3.3 – *Simulation d'une opération "$+1_x$".*

Remarque 6. *Il est intéressant de noter que dans notre réduction, seule une transition (encadrée en pointillé sur les figures 3.3 et 3.4) fait appel à des mises à jour autres que des remises à zéro. On peut donc en conclure que simuler cette transition (figure 3.5) (et non la machine de Minsky complète) suffit à prouver l'indécidabilité. C'est ce que nous allons faire par la suite.*

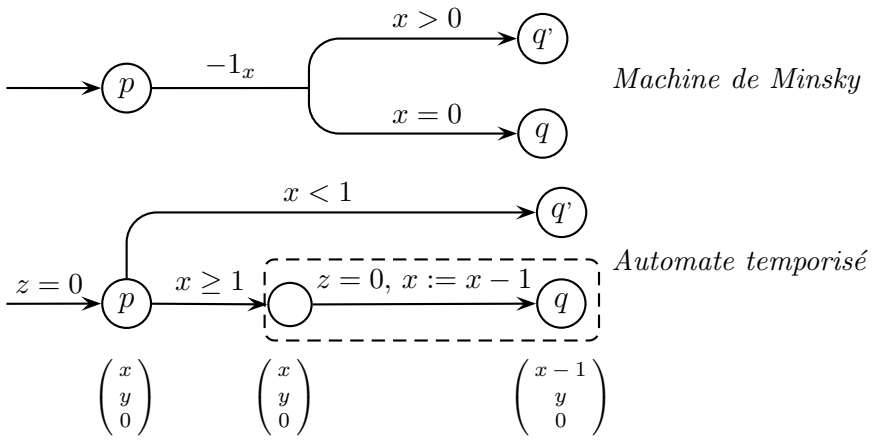

FIG. 3.4 – *Simulation d'une opération "-1_x".*

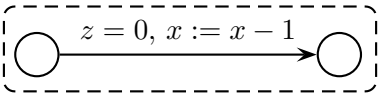

FIG. 3.5 – *Transition utilisant les mises à jour dans la démonstration.*

3.4 Les automates temporisés avec mises à jour $x := x + 1$

Nous abordons ici les mises à jour $x := x + 1$ avec des gardes diagonales. À nouveau pour ce fragment, l'une des deux opérations est aisée. Il s'agit ici de l'incrémentation. La décrémentation telle que nous l'avons codée fait intervenir une horloge supplémentaire qui sert de référence aux deux autres.

Théorème 8. *Soit X un ensemble fini d'horloges tel que $\{w, x, y, z\} \subseteq X$ et un ensemble de mises à jour $U = \{w{:=}0,\ x{:=}0,\ y{:=}0,\ x{:=}x+1,\ y{:=}y+1,\ w{:=}w+1\}$. Alors le problème du vide sur la sous-classe $Aut(\mathcal{C}, U)$ est indécidable.*

Démonstration. On se sert d'une méthode non déterministe pour simuler une décrémentation des compteurs (figure 3.6). Pour ce faire, on utilise une horloge intermédiaire w qui est incrémentée de façon aléatoire en un temps nul (les contraintes sur z arrêtent l'écoulement du temps). Puis on teste le fait que $x - w = 1$ en sortie d'état. Enfin, on incrémente le compteur x de façon aléatoire et, comme précédemment, on teste $x = w$. □

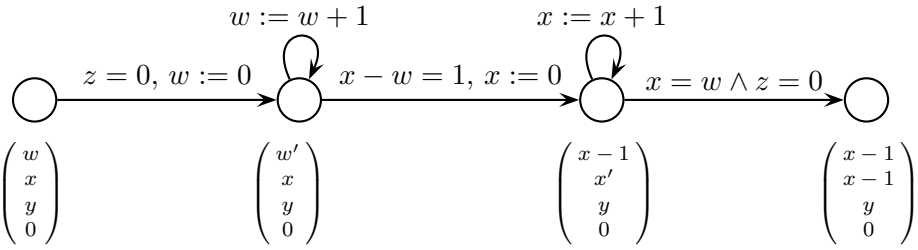

FIG. 3.6 – *Simulation d'une mise à jour "$x := x - 1$" avec "$x := x + 1$".*

Remarque 7. *Il est important de remarquer que l'on utilise des contraintes $x-y=1$. Comme on le verra par la suite, l'absence de ce type de contraintes rend la sous-classe correspondante décidable.*

3.5 Les automates temporisés avec mises à jour $x :> 0$

Nous passons ici aux mises à jours non déterministes avec les mises à jour du type $x :> c$. Ici, nous ne considérons que les mises à jour de type $x :> 0$, mais la preuve peut facilement s'étendre à $x :> c$ par un corollaire assez simple. La technique que nous introduisons ici joue sur la condition d'acceptation du mot temporisé et sur le fait que l'on peut faire un choix non déterministe. Ce méthode de preuve sera reprise pour presque toutes les preuves d'indécidabilité des fragments qui incluent des mises à jour non déterministes.

Théorème 9. *Soit X un ensemble fini d'horloges tel que $\{w,x,y,z\} \subseteq X$ et un ensemble de mises à jour tel que $U = \{x:>0,\ y:>0,\ w:>0\}$. Alors le problème du vide sur la sous-classe $Aut(\mathcal{C},U)$ est indécidable.*

Démonstration. On procède exactement comme pour les $x := x+1$ (incrémentation aléatoire), excepté que l'on remplace les boucles successives par une unique transition avec une mise à jour non déterministe de type $x :> 0$ (figure 3.7). □

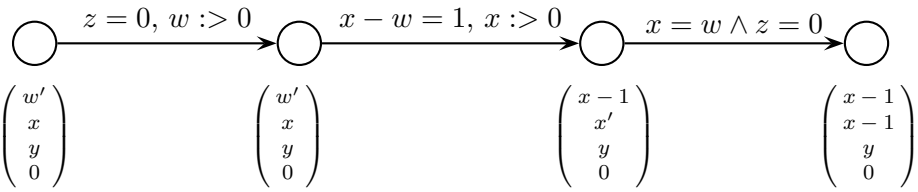

FIG. 3.7 – *Simulation d'une mise à jour "$x := x-1$" avec "$x :> 0$".*

3.6 Les automates temporisés avec mises à jour $x :> y$

Nous nous intéressons ici aux mises à jour de type $x :> y$ avec gardes diagonales. La principale astuce, ici, consiste à utiliser des mises à jour de type $w :> w$ alors que l'on a précédemment remis le contenu de l'horloge w à zéro. L'horloge peut prendre virtuellement n'importe quelle valeur, mais une seule nous intéresse : $x-1$. Dès lors il suffit d'ajouter une garde $w = x-1$ sur la seule transition sortante de l'état suivant la mise à jour $w :> w$.

Théorème 10. *Soit X un ensemble fini d'horloges tel que $\{w,x,y,z\} \subseteq X$ et un ensemble de mises à jour tel que $U = \{w:=0,\ x:=0\, y:=0,\ x:>x,\ y:>y,\ w:>w\}$. Alors le problème du vide sur la sous-classe $Aut(\mathcal{C},U)$ est indécidable.*

Démonstration. Analogue aux deux cas précédents en remplaçant les incrémentations aléatoires par une remise à zéro de l'horloge w, suivie d'une assignation non déterministe (similaire à celle de $x :> 0$). □

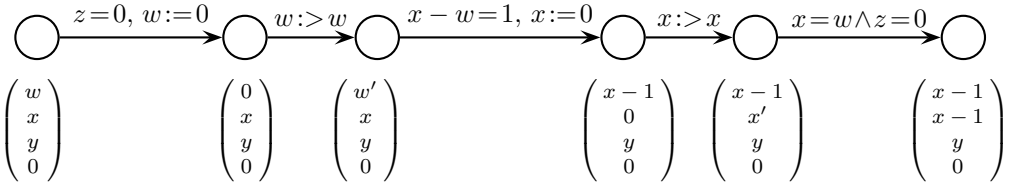

FIG. 3.8 – *Simulation d'une mise à jour "$x := x - 1$" avec "$x :> y$".*

3.7 Les automates temporisés avec mises à jour $x :< y$

Nous traitons ici les mises à jour de type $x :< y$ avec des gardes diagonales. Comme nous voulons trouver de façon non déterministe la valeur $x - 1$, il suffit de faire prendre à w n'importe quelle valeur inférieure à x puis de vérifier qu'elle est bien égale à $x - 1$.

Théorème 11. *Soit X un ensemble fini d'horloges tel que $\{w, x, y, z\} \subseteq X$ et un ensemble de mises à jour tel que $U = \{x :< x,\ y :< y,\ w :< x,\ w :< y\}$. Alors le problème du vide sur la sous-classe $Aut(\mathcal{C}, U)$ est indécidable.*

Démonstration. On utilise la mise à jour $w :< x$ pour deviner la valeur de $x - 1$. Puis, on teste si la condition est bien réalisée ($x - w = 1$) et on fait la même chose avec $x :< x$ en le comparant à w en sortie. □

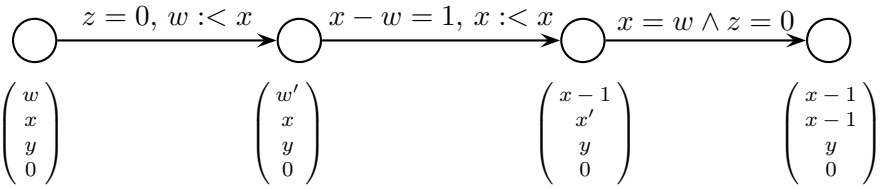

FIG. 3.9 – *Simulation d'une mise à jour "$x := x - 1$" avec "$x :< y$".*

3.8 Les automates temporisés avec mises à jour $y+c<:x:<z+d$

On va maintenant montrer que l'ajout d'une mise à jour du type $y+c <: x :< z+d$ permet de simuler les comparaisons d'horloges et rend indécidable des sous classes de $Aut(\mathcal{C}_{df}, \mathcal{U})$ de la même manière que $Aut(\mathcal{C}, \mathcal{U})$.

Théorème 12. *Soit X un ensemble fini d'horloges tel que $\{x, y, z\} \subseteq X$ et U un ensemble de mises à jour. Si $Aut(\mathcal{C}, U)$ est indécidable.*

Alors la classe $Aut(\mathcal{C}_{df}, U \cup \{y + c <: x :< z + d \mid x, y, z \in X,\ c, d \in \mathbb{Q}_+\})$ l'est aussi.

Démonstration. Afin de prouver ce théorème, nous allons montrer que nous pouvons simuler la contrainte diagonale $x - y < 0$ par une mise à jour $x <: z :< y$ (figure 3.10). Étendre ce résultat pour des contraintes diagonales $x - y < d$ et des mises à jour $x + c <: z :< y + d$ est ensuite immédiat.

La transition qui contient la mise à jour $x <: z :< y$ ne peut être prise que si l'ensemble des valuations possibles n'est pas vide. Or, si $x - y \geq 0$ cet ensemble est vide. Par conséquent,

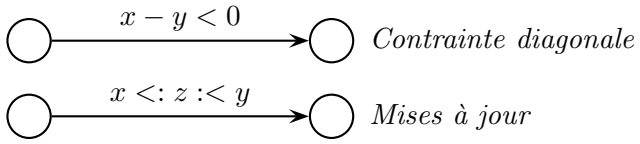

FIG. 3.10 – *Simulation d'une contrainte "$x - y \sim 0$" par "$y <: x :< z$".*

cette transition ne peut être tirée que si $x - y < 0$. On procède de façon similaire pour $>$, $=$, et $\neq$. On peut donc simuler n'importe quelle contrainte sur une différence d'horloge par une mise à jour de type $y \sim: x :\sim z$.

Le résultat pour la contrainte $x - y < d$ avec des mises à jour de type $x + c <: z :< y + d$ s'obtient de la même façon. □

On peut alors en déduire le corollaire suivant qui établit que considérer des gardes diagonales ou des mises à jour de type $x + c <: z :< y + d$ est équivalent.

Corollaire 12.1. *La décidabilité du problème du vide pour des sous-classes de $Aut(\mathcal{C}_{df}, \mathcal{U})$ qui contiennent des mises à jour du type $y + c <: x :< z + d$ est la même que pour les sous-classes de $Aut(\mathcal{C}, U')$ avec $U' = U \setminus \{y + c <: x :< z + d \mid x, y, z \in X,\ c, d \in \mathbb{Q}_+\}$.*

3.9 Conclusion

Les résultats que nous venons d'établir nous permettent de compléter plus avant le tableau 3.1 que nous avions introduit au chapitre 2. Comme on peut le voir, l'indécidabilité du modèle qui autorise les contraintes diagonales a été largement établie.

	Mises à jours	Contraintes non diagonales	Contraintes générales
Déterministes	$x := 0$	Pspace	Pspace
	$x := c,\ c \in \mathbb{Q}_+$	?	?
	$x := y + c,\ c \in \mathbb{Q}_+$	?	Indécidable
	$x := y - 1$	Indécidable	
	$x := y + c,\ c \in \mathbb{Q}$		
Non déterministes	$x :< c,\ c \in \mathbb{Q}_+$	?	?
	$x :> c,\ c \in \mathbb{Q}_+$	?	Indécidable
	$x :< y$	?	
	$x :> y$	?	
	$x :< y + c,\ c \in \mathbb{Q}_+$	?	
	$x :> y + c,\ c \in \mathbb{Q}_+$	?	
	$y + c <: x :< y + d,$ $c, d \in \mathbb{Q}_+$	?	
	$y + c <: x :< z + d,$ $c, d \in \mathbb{Q}_+,\ y \neq z$	Indécidable	

TAB. 3.1 – *Décidabilité des différentes sous-classes d'automates temporisés.*