Complexité algorithmique de variantes de LTL pour la vérification

DEA Algorithmique 2003/2004

Notes de cours

Stéphane Demri[1]

23 Février 2004

La vérification de systèmes est une approche formelle dont le but est de vérifier si un programme satisfait certaines propriétés. La logique temporelle offre un langage de spécification priviligié pour la formulation de propriétés à vérifier.

Nous présentons les résultats de complexité algorithmique de la logique du temps à structure linéaire (LTL), de son extension avec des opérateurs temporels définis à l'aide d'automates finis, et de ses variantes avec contraintes atomiques exprimées avec des formules de l'arithmétique de Presburger.

Nous montrerons où se trouvent les sauts de complexité en étudiant des fragments de ces logiques et où se trouvent les limites de la décidabilité des extensions.

L'objet de ce cours est d'étudier différents aspects de la complexité algorithmique de la logique LTL omniprésente comme langage de spécification en considérant ses variantes et différentes approches pour mesurer la complexité.

[1]Email : demri@lsv.ens-cachan.fr.

Table des matières

1 Rappel : complexité algorithmique

1.1 Machines de Turing

On va rappeler ici brièvement quelques notions de base concernant les machines de Turing qui suffisent pour les développements à venir.

Une *machine de Turing* M (non-déterministe à un ruban) est une structure de la forme $\langle Q, \Sigma, \delta, s_0, s_A \rangle$ telle que :

- Q est un ensemble non vide et fini d'états ;
- Σ est un alphabet contenant le symbole blanc $\sharp$ et le symbole de début de ruban $\triangleright$;
- $s_0 \in Q$ est l´état initial ;
- $s_A \in Q$ est l'état acceptant ;
- $\delta \subseteq Q \times \Sigma \times Q \times \Sigma \times \{\leftarrow, \rightarrow, -\}$ (on utilise aussi $\{-1, +1, 0\}$ au lieu de $\{\leftarrow, \rightarrow, -\}$) est la relation de transition ;
- si $\langle s, \triangleright, s', a', m \rangle \in \delta$ alors $a' = \triangleright$ et $m \in \{\rightarrow, -\}$ (pas de déplacement de la tête à gauche de $\triangleright$).

Une configuration est un triplet $\langle s, w_1, w_2 \rangle \in Q \times \Sigma^+ \times \Sigma^*$ modélisant l'état de M à une étape du calcul :

- M est dans l'état de contrôle s,
- $w_1 \cdot w_2 \cdot \sharp^\omega$ est sur le ruban (infini à droite) et
- la tête est sur la dernière lettre de w_1.

Étant donné un mot $w \in (\Sigma \setminus \{\sharp, \rhd\})^*$, la configuration initiale $C(w)$ est $\langle s_0, \rhd, w\rangle$.

La relation de dérivation $\vdash_M$ est une relation binaire sur l'ensemble des configurations qui correspond à un pas de calcul avec M. Le *langage accepté* par la machine M est l'ensemble

$$\{w \in (\Sigma \setminus \{\sharp, \rhd\})^* : \exists w_1, w_2\ C(w) \vdash_M^* \langle s_A, w_1, w_2\rangle\}.$$

Un langage L sur l'alphabet Σ (ne contenant ni $\sharp$ ni $\rhd$) est *décidé* par la machine $M \overset{\text{def}}{\Leftrightarrow}$ l'alphabet de M est $\Sigma \cup \{\sharp, \rhd\}$, $\mathrm{L}(M) = \mathrm{L}$ et M n'a pas d'exécution infinie.

La complexité en temps d'une machine de Turing M (sans exécutions infinies) est la fonction $f : \mathbb{N} \to \mathbb{N}$ telle que $f(n)$ est le nombre maximal de pas de calcul pour accepter un mot d'entrée de longueur n. Ici, le nombre de pas de calcul pour accepter un mot est le minimum des longueurs des exécutions l'acceptant.

De façon analogue, la complexité en espace d'une machine de Turing M (sans exécutions infinies) est la fonction $g : \mathbb{N} \to \mathbb{N}$ telle que $g(n)$ est le nombre maximal de cases mémoire utilisées sur le ruban de travail pour accepter un mot d'entrée de longueur n. Pour la com-

plexité en espace, on utilise ici des machines ayant un ruban d'entrée muni d'une tête en mode de lecture seulement et un ruban de travail muni d'une tête en mode de lecture/écriture.

Par exemple, P est la classe des langages décidés par une machine de Turing déterministe dont la complexité en temps est polynomiale. De même, NPSPACE est la classe des langages acceptés par une machine de Turing non-déterministe dont la complexité en espace est polynomiale.

La relation de transition d'une machine déterministe est une fonction partielle de $Q \times \Sigma$ vers $Q \times \Sigma \times \{\leftarrow, \rightarrow, -\}$.

Un problème de décision est un langage de mots finis qui peut être aussi défini par sa fonction caractéristique (présentation adoptée dans la suite).

Quelques notions utilisées dans la suite mais non rappelées sont :

- transformation (en espace logarithmique, temps polynomial, ...) ;
- problème décidable/indécidable ;
- coC : complément de la classe de complexité C ;

– C-complétude, C-dureté.

1.2 Hiérarchie de classes

On note $P_1 \leq P_2$ s'il existe une transformation des problèmes de décision P_1 vers P_2 en utilisant un espace mémoire logarithmique. De plus $\equiv$ est la relation $\leq \cap \leq^{-1}$.

Voici quelques relations entre des classes de complexité usuelles :

- NLOGSPACE $\subseteq$ P $\subseteq$ NP $\subseteq$ PSPACE $\subseteq$ EXPTIME $\subseteq$ EXPSPACE ;
- P$\neq$ EXPTIME, PSPACE $\neq$ EXPSPACE ;
- PSPACE = NPSPACE.

1.3 Problèmes standards

Nous rappelons quelques problèmes qui caractérisent respectivement différentes classes de complexité usuelles. Des réductions de ces problèmes seront construites pour établir des bornes inférieures de complexité.

1.3.1 Accessibilité

Le problème d'accessibilité dans un graphe, noté GAP ("Graph Accessibility Problem") est le suivant :

entrée : un graphe orienté et deux sommets s et t ;

sortie : 1 s'il existe un chemin entre s et t, 0 sinon.

Proposition 1.1. [Jon75] GAP est NLOGSPACE-complet.

1.3.2 QBF

Le problème des formules booléennes quantifiées, appelé QBF ("Quantified Boolean Formula") est le suivant :

entrée : $\phi = \mathcal{Q}_1\ \mathrm{p}_1\ \ldots\ \mathcal{Q}_k\ \mathrm{p}_k\ \bigwedge_{i=1}^{n} \bigvee_{j=1}^{n_i} l_{i,j}$ où
- $\mathcal{Q}_1, \ldots, \mathcal{Q}_k \in \{\exists, \forall\}$,
- $\mathrm{p}_1, \ldots, \mathrm{p}_k$ sont des variables propositionnelles,
- les $l_{i,j}$ sont des littéraux construits sur $\mathrm{p}_1, \ldots, \mathrm{p}_k$.

sortie : 1 si ϕ est satisfaisable, 0 sinon.

Soit SAT, le problème de la satisfaisabilité du calcul propositionnel, defini comme la restriction de QBF aux instances telles que $\{\mathcal{Q}_1, \ldots, \mathcal{Q}_k\} = \{\exists\}$.

Proposition 1.2. SAT est NP-complet [Coo71].

Proposition 1.3. QBF est PSPACE-complet [SM73].

Le problème demeure PSPACE-difficile si on se restreint aux instances où il y a une stricte alternance de $\exists$ et $\forall$ et si la matrice de ϕ est en 3CNF.

1.3.3 Pavage

Un *jeu de dominos* $Dom = \langle C, D, Coul \rangle$ est un triplet composé

- d'un ensemble fini C de couleurs ;
- d'un ensemble fini D de dominos ;
- d'une fonction $Coul : D \times \{haut, bas, droite, gauche\} \to C$ qui associe une couleur à chaque côté de chaque domino.

On dit que Dom peut paver une surface $S \subseteq \mathbb{N} \times \mathbb{N}$ ssi il existe une façon de couvrir S avec les éléments de Dom en préservant les contraintes de motifs : seuls des côtés de même couleur peuvent être côte à côte (horizontalement, verticalement et sans possiblité de faire tourner les dominos). En fait, "domino" n'est pas le terme le plus heureux : "polymino" ou "quadrimino" serait plus adéquat.

Le problème suivant, noté DOM1, est NP-complet :

entrée : un jeu de dominos Dom, $n \in \mathbb{N}^+$ (codé en unaire) ;

sortie : 1, s'il est possible de paver $\{1, \ldots, n\}^2$ avec Dom. 0 sinon.

Si dans DOM1, l'entier n est codé en notation binaire, alors le problème est NEXPTIME-complet.

Le problème suivant, noté DOM2, est PSPACE-complet :

entrée : un jeu de dominos Dom avec deux couleurs c_1, c_2 et $n \in \mathbb{N}^+$ (en unaire) ;

sortie : 1, si Dom peut paver $\{1, \ldots, n\} \times \{1, \ldots, m\}$ pour un $m \geq 1$ tel que le bas du carreau en bas à gauche est c_1 et le haut du carreau en haut à gauche est c_2. 0 sinon.

Exercice 1.1. Montrer que pour résoudre DOM2, on peut se restreindre à $1 \leq m \leq 2^n$.

Le problème suivant est indécidable :

entrée : un jeu de dominos Dom ;

sortie : 1, si Dom peut paver $\mathbb{N} \times \mathbb{N}$. 0 sinon.

Une variante de ce problème, notée DOMREC, est définie ainsi :

entrée : un jeu de dominos Dom et une couleur c ;

sortie : 1, si Dom peut paver $\mathbb{N} \times \mathbb{N}$ et c apparaît infiniment souvent. 0 sinon.

DOMREC est "hautement indécidable" (Σ_1^1-complet). Voir par exemple [Har85] pour plus d'information sur les problèmes de pavage.

2 Complexité de de fragments de LTL

2.1 Rappel sur LTL

Les formules de LTL sont construites à partir de la grammaire abstraite suivante :

$$\phi ::= \mathrm{p}_i \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \mathrm{X}\phi \mid \mathrm{F}\phi \mid \phi \mathrm{U} \psi$$

où $\mathrm{PROP} = \{\mathrm{p}_1, \mathrm{p}_2, \ldots\}$ est un ensemble infini dénombrable de variables propositionnelles.

Les opérateurs $\vee$ et F sont définissables à partir des autres opérateurs. Par exemple $\mathrm{F}\phi$ est équivalent à $\top \mathrm{U} \phi$ (avec $\top = \mathrm{p} \vee \neg \mathrm{p}$).

On note $\mathrm{LTL}_n^k(\mathrm{H}_1, \mathrm{H}_2, \ldots)$ le fragment de LTL restreint

- aux opérateurs temporels $\mathrm{H}_1, \mathrm{H}_2, \ldots$;
- aux formules de hauteur temporelle au plus $k \geq 0$ et
- avec au plus $n \geq 1$ variables propositionnelles.

La temporelle hauteur d'une formule est le nombre maximal d'opérateurs temporels emboîtés. Par exemple $ht((\mathrm{XXp}) \vee (\mathrm{pU}\neg\mathrm{q})) = 2$.

Par exemple, $\mathrm{LTL}^2_\omega(\mathrm{F})$ dénote l'ensemble des formules de LTL de hauteur temporelle au plus 2 construites avec le seul opérateur temporel F.

$|\phi|$ dénote la taille de la formule ϕ vue comme une chaîne de caractères.

Un *structure pour LTL* est une séquence infinie $\sigma : \mathbb{N} \to \mathcal{P}(\mathrm{PROP})$, c'est-à-dire un mot infini de $\mathcal{P}(\mathrm{PROP})^\omega$.

Étant donnés une structure σ, $i \in \mathbb{N}$ et une formule ϕ, on définit par induction la relation de satisfaisabilité $\models$:

- $\sigma, i \models \mathrm{p} \overset{\text{def}}{\Leftrightarrow} \mathrm{p} \in \sigma_i$;
- $\sigma, i \models \neg\phi \overset{\text{def}}{\Leftrightarrow} \sigma, i \not\models \phi$ ($\vee$ et $\wedge$ ont leur sémantique usuelle) ;
- $\sigma, i \models \mathrm{X}\phi \overset{\text{def}}{\Leftrightarrow} \sigma, i+1 \models \phi$;
- $\sigma, i \models \mathrm{F}\phi \overset{\text{def}}{\Leftrightarrow}$ il existe $j \geq i$ tel que $\sigma, j \models \phi$;
- $\sigma, i \models \phi\mathrm{U}\psi \overset{\text{def}}{\Leftrightarrow}$ il existe $j \geq i$ tel que $\sigma, j \models \psi$ et pour $i \leq k < j$, $\sigma, k \models \phi$.

On note $\sigma \models \phi$ pour $\sigma, 0 \models \phi$.

Modèles(ϕ) $\stackrel{\text{def}}{=} \{\sigma \in \mathcal{P}(\text{PROP})^\omega : \sigma \models \phi\}$.

ϕ est *satisfaisable* (pour LTL) $\stackrel{\text{def}}{\Leftrightarrow}$ Modèles(ϕ) $\neq \emptyset$.

Le problème de la satisfaisabilité pour LTL, noté SAT(LTL), est le suivant :

entrée : une formule ϕ de LTL ;

sortie : 1 s'il existe une structure σ telle que $\sigma \models \phi$, 0 sinon.

Exercice 2.1. Montrer que SAT(LTL_1(X)) est NP-complet.

Intéressons nous maintenant au problème du model-checking. Une *structure de Kripke* $\mathcal{M} = \langle W, R, L\rangle$ est composée

- d'un ensemble non-vide d'états W,
- d'une relation binaire R et,
- d'une fonction d'interprétation $L : W \to \mathcal{P}(\text{PROP})$.

$\mathcal{M}$ est un graphe dont une interprétation du calcul propositionnel est associé à chaque état.

Un chemin de $\mathcal{M}$ est une séquence $s_0 s_1 \ldots$ (finie ou infinie) telle que $s_i R s_{i+1}$ pour $i \geq 0$. On note Chemins($\mathcal{M}, s_0$) l'ensemble des chemins infinis de $\mathcal{M}$ commençant par s_0.

Par abus, on note aussi Chemins($\mathcal{M}, s_0$) l'ensemble des chemins infinis commençant par s_0 sur l'alphabet $\mathcal{P}(\text{PROP})^\omega$. La première lettre de chacun de ces chemins est donc $L(s_0)$.

Le problème du model-checking pour LTL, noté $\text{MC}^\exists(\text{LTL})$, est le suivant :

entrée : une formule de LTL ϕ, une structure de Kripke finie et totale[2] $\mathcal{M}$ et $s_0 \in S$;

sortie : 1 s'il existe un chemin infini σ commençant par s_0 tel que $\sigma \models \phi$ (noté $\mathcal{M}, s_0 \models_\exists \phi$), 0 sinon.

Dans l'énoncé du problème ci-dessus, on suppose que le domaine de la fonction d'interprétation L dans $\mathcal{M}$ est restreint aux propositions atomiques apparaissant dans ϕ.

La taille d'une structure finie $\langle W, R, L\rangle$ est alors :

$$\text{card}(W) + \text{card}(R) + \Sigma_{w\in W}\text{card}(L(w)).$$

On a $\mathcal{M}, s_0 \models_\exists \phi$ ssi Chemins($\mathcal{M}, s_0$) $\cap$ Modèles(ϕ) $\neq \emptyset$.

Cette définition est duale de la définition aussi utilisée en vérification où une quantification universelle est

[2] $\forall x \in W, \exists y \in W,\ \langle x, y\rangle \in R$.

utilisée (problème dual noté $\mathrm{MC}^{\forall}$(LTL) avec la relation $\mathcal{M}, s_0 \models_{\forall} \phi$).

En utilisant des propriétés de base sur les classes de complexité définies avec des machines de Turing déterministes (voir par exemple [Pap94, Chapitre 7]), on peut montrer que $\mathrm{MC}^{\forall}$(LTL) est dans PSPACE [resp. PSPACE-difficile] ssi $\mathrm{MC}^{\exists}$(LTL) est dans PSPACE [resp. PSPACE-difficile].

2.2 Réduction du model-checking vers la satisfaisabilité

La première réduction que nous aborderons consiste à réduire $\mathrm{MC}^{\exists}$(LTL) à SAT(LTL). Une conséquence immédiate de ce résultat est que $\mathrm{MC}^{\exists}$(LTL) est dans PSPACE si SAT(LTL) est dans PSPACE.

Proposition 2.1. $\mathrm{MC}^{\exists}(\mathrm{LTL}) \leq \mathrm{SAT}(\mathrm{LTL})$ (réduction en espace logarithmique).

Preuve. L'idée de la preuve est de coder la structure de Kripke par une formule [SC85, page 740]. Soit $\mathcal{M} = \langle W, R, L \rangle$ une structure de Kripke finie et totale et ϕ une formule de LTL construites sur les variables propositionnelles $\mathrm{p}_1, \ldots, \mathrm{p}_k$. A chaque état s de W on associe une nouvelle variable propositionnelle p_s.

Pour chaque état s, on code la valuation $L(s)$ par la formule $varprop_s$:

$$varprop_s \stackrel{\text{def}}{=} \bigwedge\{\mathrm{p}_i : 1 \leq i \leq k, \mathrm{p}_i \in L(s)\} \wedge$$

$$\bigwedge\{\neg \mathrm{p}_i : 1 \leq i \leq k, \mathrm{p}_i \notin L(s)\}.$$

Pour chaque état s, on code l'ensemble des successeurs $R(s) = \{s' \in W : \langle s, s' \rangle \in R\}$ de s par la formule :

$$succ_s \stackrel{\text{def}}{=} \mathrm{X} \bigvee\{\mathrm{p}_{s'} : s' \in R(s)\}.$$

Chaque état s dans la structure $\mathcal{M}$ est codé par la formule $\phi_s \stackrel{\text{def}}{=} \mathrm{p}_s \Rightarrow (varprop_s \wedge succ_s)$.

Finalement, la structure $\mathcal{M}$ est codée par la formule

$$\phi_{\mathcal{M}} \stackrel{\text{def}}{=} \mathrm{G}(\bigwedge\{\phi_s : s \in W\} \wedge \mathrm{UNI})$$

où UNI est une formule propositionnelle (sans opérateurs temporels) qui énonce qu'une unique variable propositionnelle de $\{\mathrm{p}_s : s \in W\}$ est vérifiée dans l'état courant.

On peut montrer que $\mathcal{M}, s \models_{\exists} \phi$ (version existentielle) ssi $\phi_{\mathcal{M}} \wedge \phi \wedge \mathrm{p}_s$ est satisfaisable. C.Q.F.D.

Exercice 2.2. Le problème de la validité pour LTL, noté VAL(LTL), est défini comme suit :

entrée : une formule ϕ de LTL ;

sortie : 1 si pour toutes les structures σ, on a $\sigma \models \phi$, 0 sinon.

Montrer que $\mathrm{MC}^{\forall}(\mathrm{LTL}) \leq \mathrm{VAL}(\mathrm{LTL})$.

La preuve de la proposition 2.1 utilise une approche assez standard qui consiste à réduire une question de la forme $\mathcal{M} \models \phi$ à la satisfaisabilité/validité de $\psi_{\mathcal{M}} \wedge \phi$ où $\psi_{\mathcal{M}}$ code le modèle $\mathcal{M}$. Pour répondre à cette seconde question, des méthodes déductives peuvent alors être utilisées.

2.3 PSPACE-dureté de la satisfaisabilité

Le problème de dominos DOM2 peut-être facilement réduit à SAT(LTL) comme l'illustre la preuve de la proposition suivante.

Proposition 2.2. SAT(LTL) est PSPACE-difficile.

Preuve. Nous présentons une réduction de DOM2 vers $\mathrm{SAT}(\mathrm{LTL}^2(\mathrm{F}, \mathrm{X}))$, c'est-à-dire les formules construites seront au plus de hauteur temporelle 2.

Soit $Dom = \langle \{0, 1\}, \{d_1, \ldots, d_m\}, Coul \rangle$ un jeu de dominos et $n \geq 1$.

Les variables propositionnelles ci-dessous entrent en jeu dans la réduction :

- lv : vraie si l'état code une rangée de n dominos ;
- pour $1 \leq i \leq n$, pour $1 \leq j \leq m$, on utilise la variable propositionnelle $\boxed{\text{j}}_i$ (" sur une rangée la position i est occupée par le domino d_j") ;
- pour $1 \leq i \leq n$, on utilise les variables $haut_{i,0}$, $bas_{i,0}$, $gauche_{i,0}$, $droite_{i,0}$, $haut_{i,1}$, $bas_{i,1}$, $gauche_{i,1}$, $droite_{i,1}$. Par exemple, la variable propositionnelle $bas_{2,1}$ est vraie dans un état vérifiant lv ssi la couleur du deuxième domino est 1[3].

Sur chaque rangée où lv est vérifiée, chaque position de $\{1, \ldots, n\}$ est occupée par un unique domino :

$$\mathrm{G}(lv \Rightarrow \bigwedge_{i=1}^{n}(\bigvee_{j=1}^{m}(\boxed{\text{j}}_i \wedge \bigwedge_{j'=1, j' \neq j}^{m} \neg\boxed{\text{j}'}_i)))$$

Les variables propositionnelles pour les couleurs sont compatibles avec la définition des dominos :

$$\mathrm{G}(lv \Rightarrow \bigwedge_{i=1}^{n}\bigwedge_{j=1}^{m}\boxed{\text{j}}_i \implies$$

[3]Comme il n'y a que deux couleurs possibles, on pourrait aussi définir la réduction avec les seules variables $haut_{i,0}$, $bas_{i,0}$, $gauche_{i,0}$, $droite_{i,0}$.

$$\bigwedge_{cot\in\{haut,bas,droite,gauche\}} cot_{i,Coul(d_j,cot)} \wedge \neg cot_{i,1-Coul(d_j,cot)})$$

On rappelle que la fonction de couleur $Coul$ est de la forme $Coul : \{d_1, \ldots, d_m\} \times \{haut, bas, droite, gauche\} \rightarrow \{0, 1\}$.

Les contraintes de couleur sur les dominos adjacents sont vérifiées :

$$\mathrm{G}(lv \Rightarrow \bigwedge_{i=1}^{n-1}(\bigwedge_{c\in\{0,1\}} droite_{i,c} \Rightarrow gauche_{i+1,c}))\wedge$$

$$\bigwedge_{i=1}^{n-1} \mathrm{G}((lv \wedge \mathrm{X}\ lv) \Rightarrow \wedge \bigwedge_{c\in\{0,1\}} haut_{i,c} \Rightarrow \mathrm{X}\ bas_{i,c})).$$

À partir d'un état on ne prend plus en compte la valeur de dominos avec les conditions initiale et finale :

$$lv \wedge \mathrm{G}(\neg lv \Rightarrow \mathrm{X}\ \neg lv) \wedge bas_{1,0} \wedge \mathrm{F}(lv \wedge \mathrm{X}\ \neg lv \wedge haut_{1,1})$$

On peut montrer que la conjonction des formules ci-dessus est satisfaisable ssi il est possible de couvrir $\{1, \ldots, n\} \times \{1, \ldots, m\}$ pour un $m \geq 1$ tel que le bas du domino en bas à gauche est 0 et le haut du domino en haut à gauche est 1. C.Q.F.D.

La preuve de la proposition 2.2 est celle de [Har85].

2.4 Borner la hauteur temporelle - Model-checking

Soit ϕ une instance du problème QBF. ϕ est de la forme

$$\mathcal{Q}_1 \mathrm{p}_1 \ldots \mathcal{Q}_n \mathrm{p}_n \overbrace{\bigwedge_{i=1}^{m} \bigvee_{j=1}^{k_i} l_{i,j}}^{\phi_0}$$

où $\{\mathcal{Q}_1, \ldots, \mathcal{Q}_n\} \subseteq \{\forall, \exists\}$, ϕ_0 est une formule propositionnelle sous forme normale conjonctive.

Chaque $l_{i,j}$ est une variable propositionnelle $\mathrm{p}_{r(i,j)}$ ou sa négation $\neg \mathrm{p}_{r(i,j)}$ de l'ensemble $X = \{\mathrm{p}_1, \ldots, \mathrm{p}_n\}$, c'est-à-dire chaque formule $l_{i,j}$ est un littéral.

On peut facilement montrer que ϕ est satisfaisable ssi il existe un ensemble non-vide $\mathcal{V}$ d'interprétations propositionnelles de la forme $v : X \to \{\text{Vrai}, \text{Faux}\}$ tel que :

(correction) pour $v \in \mathcal{V}$, $v \models \phi_0$ (au sens propositionnel) ;

(fermeture) pour $v \in \mathcal{V}$, pour $1 \leq r \leq n$, tels que $\mathcal{Q}_r = \forall$, il existe $v' \in \mathcal{V}$ tel que

1. $v'(\mathrm{p}_r) \neq v(\mathrm{p}_r)$ et,
2. pour tous les $1 \leq r' < r$, $v'(\mathrm{p}_{r'}) = v(\mathrm{p}_{r'})$.

En effet, $\exists$ p ψ est équivalent à $\psi[\mathrm{p} \leftarrow \bot] \vee \psi[\mathrm{p} \leftarrow \top]$ et $\forall$ p ψ est équivalent à $\psi[\mathrm{p} \leftarrow \bot] \wedge \psi[\mathrm{p} \leftarrow \top]$.

Exercice 2.3. Calculer un ensemble $\mathcal{V}$ d'interprétations qui témoigne de la satisfaction de $\forall\, \mathrm{p}_1\ \forall\, \mathrm{p}_2\ \exists\, \mathrm{p}_3\ \neg \mathrm{p}_3 \Leftrightarrow (\mathrm{p}_1 \wedge \mathrm{p}_2)$.

À ϕ on associe une structure de Kripke $\mathcal{M}_\phi$ définie dans la figure 1 en utilisant les variables propositionnelles dans

$$\{A_0, A_1, \ldots, B_0, B_1, \ldots \mathrm{p}_1^T, \ldots, L_1^1, \ldots\}.$$

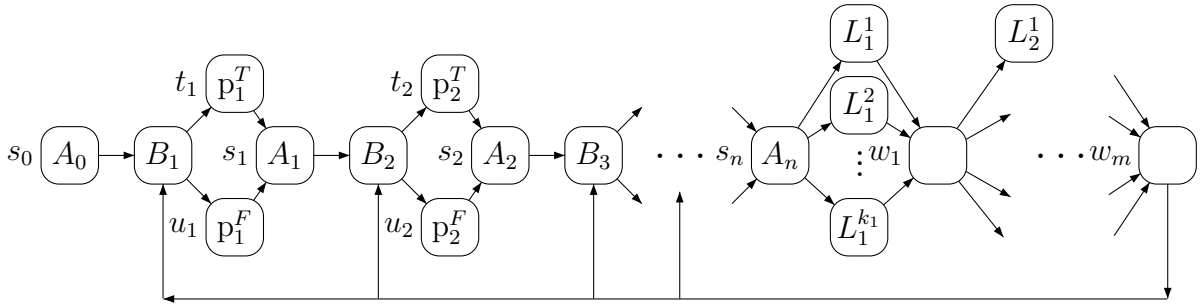


FIG. 1 – Modèle $\mathcal{M}_\phi$ associé à $\phi \equiv \mathcal{Q}_1 \mathrm{p}_1 \ldots \mathcal{Q}_n \mathrm{p}_n \bigwedge_{i=1}^{m} \bigvee_{j=1}^{k_i} l_{i,j}$

Chaque variable propositionnelle de $\{A_0, A_1, \ldots, B_0, B_1, \ldots \mathrm{p}_1^T, \ldots,$ n'est vérifiée que dans un seul état de $\mathcal{M}_\phi$.

Supposons que σ soit un chemin infini commençant en s_0. Entre s_0 et s_n, σ code une interprétation propositionnelle pour toutes les variables de X, atteint w_m et ensuite retourne vers un état étiqueté par un B_r pour un

$1 \leq r \leq n$ où de nouvelles valeurs pour l'interprétation de $\mathrm{p}_r, \ldots, \mathrm{p}_n$ sont fixées.

À chaque position de σ entre s_n et w_m on introduit une notion d'interprétation courante qui associe Vrai ou Faux à toutes les variables propositionnelles p_r dépendant du dernier noeud u_r ou t_r visité.

On associe l'ensemble $\mathcal{V}(\sigma)$ des interprétations propositionnelles qui sont courantes aux positions quand σ visite s_n. Il y a un nombre infini de telles positions.

Soit $1 \leq r \leq n$ tel que $\mathcal{Q}_r = \forall$ et supposons que quand σ visite le noeud s_{r-1} alors il visite les noeuds t_r et u_r avant de visiter éventuellement à nouveau s_{r-1}. Cette propriété s'exprime dans LTL(U) :

$$\psi_r \stackrel{\text{def}}{=} \mathrm{G}(A_{r-1} \Rightarrow (\neg B_{r-1} \mathrm{U} \mathrm{p}_r^T) \wedge (\neg B_{r-1} \mathrm{U} \mathrm{p}_r^F))).$$

Soit $\psi_{\text{fermeture}} \stackrel{\text{def}}{=} \bigwedge\{\psi_r \mid \mathcal{Q}_r = \forall\}$: si σ satisfait $\psi_{\text{fermeture}}$, alors $\mathcal{V}(\sigma)$ vérifie la propriété (fermeture).

Maintenant, quand σ visite un état étiqueté par L_i^j, nous disons qu'il est en accord avec l'interprétation courante v si $v \models l_{i,j}$.

Cela peut s'écrire dans LTL(U) en utilisant le fait que la valeur de p_r pour l'interprétation courante ne peut pas changer sans avoir d'abord visité l'état étiqueté par B_r. Pour $1 \leq i \leq m$ (nombre de clauses), pour $1 \leq j \leq k_i$, on définit $\psi_{i,j}$ de la façon suivante :

$$\psi_{i,j} \stackrel{\text{def}}{=} \begin{cases} \mathrm{G}[\mathrm{p}_r^F \Rightarrow \mathrm{G}\neg L_i^j \vee \neg L_i^j \mathrm{U} B_r] & \text{si } l_{i,j} = \mathrm{p}_r, \\ \mathrm{G}[\mathrm{p}_r^T \Rightarrow \mathrm{G}\neg L_i^j \vee \neg L_i^j \mathrm{U} B_r] & \text{si } l_{i,j} = \neg \mathrm{p}_r. \end{cases}$$

Soit $\psi_{\text{correction}} \stackrel{\text{def}}{=} \bigwedge_{i=1}^{m} \bigwedge_{j=1}^{k_i} \psi_{i,j}$: si σ satisfait $\psi_{\text{correction}}$, alors $\mathcal{V}(\sigma)$ vérifie la propriété (correction).

Lemme 2.3. $\mathcal{M}_\phi, s_0 \models_\exists \psi_{\text{fermeture}} \wedge \psi_{\text{correction}}$ ssi ϕ est satisfaisable.

Preuve. Si $\sigma \models_\exists \psi_{\text{fermeture}} \wedge \psi_{\text{correction}}$, alors l'ensemble $\mathcal{V}(\sigma)$ est non-vide, fermé et correct et donc ϕ est satisfaisable.

Supposons à présent que ϕ soit satisfaisable. Il existe un ensemble d'interprétations $\mathcal{V}$ vérifiant les conditions (fermeture) et (correction).

À partir de $\mathcal{V}$, on peut construire un chemin infini σ commençant en s_0 tel que $\mathcal{V}(\sigma) = \mathcal{V}$ et $\sigma \models \psi_{\text{fermeture}} \wedge \psi_{\text{correction}}$: à partir d'une énumération lexicographique de $\mathcal{V}$, σ peut être facilement construit tel que $\sigma \models \psi_{\text{fermeture}}$.

Cette énumération correspond à l'énumération par ordre croissant des entiers codés en binaire issus des valuations de $\mathrm{p}_1, \ldots, \mathrm{p}_n$.

Ensuite, pour s'assurer que $\sigma \models \psi_{\text{correction}}$, chaque noeud visité entre s_n et le plus proche w_m, σ visite seulement les états étiquetés par L_i^j validés par la valuation courante v, ce qui est possible car $v \models \phi_0$. C.Q.F.D.

On peut remarquer que $\psi_{\text{fermeture}} \wedge \psi_{\text{correction}}$ appartient à $\mathrm{LTL}^2(\mathrm{U})$ (en utilisant la définition de G à l'aide de U).

Comme $\mathcal{M}_\phi$ et $\psi_{\text{fermeture}} \wedge \psi_{\text{correction}}$ peuvent être calculés en utilisant un espace logarithmique en $|\phi|$, nous pouvons déduire que :

Proposition 2.4. $\mathrm{MC}^{\exists}(\mathrm{LTL}^2(\mathrm{U}))$ est PSPACE-difficile.

Corollaire 2.5. $\mathrm{MC}^{\forall}(\mathrm{LTL})$ est PSPACE-difficile.

Une construction analogue permet de montrer que $\mathrm{LTL}(\mathrm{F}, \mathrm{X})$ est aussi PSPACE-difficile.

2.5 Borner le nombre de propositions atomiques

2.5.1 Deux variables et l'opérateur "until"

Nous allons voir dans cette section comment restreindre le nombre de variables propositionnelles tout en conservant la PSPACE-dureté.

Soit $\mathcal{M} = \langle W, R, L \rangle$ une structure de Kripke finie et totale, $w \in W$ et ϕ une formule de LTL ayant $n \geq 1$ variables propositionnelles.

Nous allons construire une structure de Kripke $\mathcal{M}_n$ et une formule ϕ_n telles que

1. $\mathcal{M}_n$ et ϕ_n se calculent en espace logarithmique en $|\mathcal{M}| + |\phi|$;
2. $\mathcal{M} \models_\exists \phi$ ssi $\mathcal{M}_n \models_\exists \phi_n$;
3. $\phi_n \in \mathrm{LTL}_2(\mathrm{U})$.

Un corollaire de la construction est le suivant :

Proposition 2.6. $\mathrm{MC}^\exists(\mathrm{LTL}_2(\mathrm{U}))$ est PSPACE-difficile.

Construisons à présent $\mathcal{M}_n = \langle W', R', L' \rangle$ et la formule ϕ_n. On suppose que $\mathrm{PROP}(\phi) = \{\mathrm{p}_1, \ldots, \mathrm{p}_n\}$ et le codomaine de L' est $\mathcal{P}(\{\mathrm{q}, \mathrm{r}\})$.

Chaque état s de W est codé dans $\mathcal{M}_n$ par $2 \times n + 2$ états successifs. La valeur $2 \times n + 2$ s'obtient de la façon suivante.

La valeur de vérité de chaque p_i dans s est codé par 2 états, ce qui fait un total de $2 \times n$ états. Les deux états supplémentaires servent de marqueurs pour passer dans $\mathcal{M}_n$ des états codant s aux états codant s' si sRs'.

La figure 2 montre un exemple d'une telle construction.

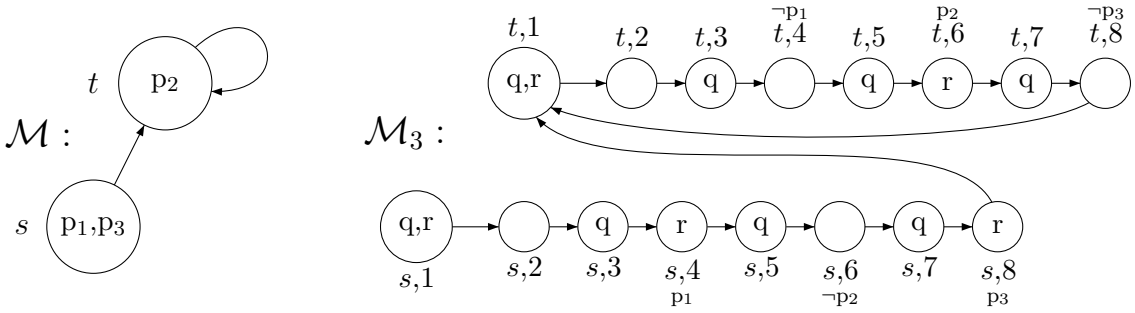


FIG. 2 – Un exemple de $\mathcal{M}$ et $\mathcal{M}_n$

La définition formelle de $\mathcal{M}_n$ est la suivante :

$$W' \stackrel{\text{def}}{=} \{\langle s,i\rangle : s \in W, 1 \leq i \leq 2n+2\}$$

$$\langle s,i\rangle R' \langle s',i'\rangle \stackrel{\text{def}}{\Leftrightarrow} \begin{cases} s = s' \text{ et } i' = i+1, \text{ ou bien} \\ sRs' \text{ et } i = 2n+2 \text{ et } i' = 1, \end{cases}$$

$$L'(\langle s,1\rangle) \stackrel{\text{def}}{=} \{\mathrm{q},\mathrm{r}\}, \qquad L'(\langle s,2j+1\rangle) \stackrel{\text{def}}{=} \{\mathrm{q}\},$$

$$L'(\langle s,2\rangle) \stackrel{\text{def}}{=} \{\}, \qquad L'(\langle s,2j+2\rangle) \stackrel{\text{def}}{=} \begin{cases} \{\mathrm{r}\} \text{ si } \mathrm{p}_j \in L(s), \\ \{\} \text{ sinon.} \end{cases}$$

avec $j = 1, \ldots, n$. Les états $\langle s,1\rangle$ et $\langle s,2\rangle$ sont des marqueurs de début de codage d'un état de $\mathcal{M}$. Ensuite, l'alternance d'états vérifiant q et $\neg$q permet de coder les valeurs de vérité des variables propositionnelles. Dans la i$^{\text{ieme}}$ alternations q $\cdot$ $\neg$q, r est vérifiée en $\neg$q ssi p_i est vérifiée en s.

Afin d'accéder aux états codant les variables propositionnelles, on introduit une famille de formules $(Alt_n^k)_{0 \leq k \leq n}$ où chaque formule Alt_n^k exprime qu'il reste q $\cdot$ $\neg$q alternations avant d'attendre un état qui satisfasse q $\wedge$ r :

- $Alt_n^0 \stackrel{\text{def}}{=} \mathrm{q} \wedge \mathrm{r}$;
- $Alt_n^{k+1} \stackrel{\text{def}}{=} \neg\mathrm{r} \wedge \mathrm{q} \wedge (\mathrm{q}\mathrm{U}(\neg\mathrm{q} \wedge (\neg\mathrm{q}\mathrm{U}Alt_n^k)))$.
  Le prochain état vérifiant q satisfait Alt_n^k.

Les états vérifiant q $\wedge$ r sont à distinguer car pour $\langle s,i\rangle \in W'$, $\mathcal{M}_n, \langle s,i\rangle \models Alt_n^0$ ssi $i = 1$.

Une fois que la structure $\mathcal{M}$ a été transformée il faut aussi répercuter cette transformation au niveau des formules. On note t_n la fonction de traduction et $\phi_n \stackrel{\text{def}}{=} t_n(\phi)$.

$$
\begin{array}{rcl}
t_n(\mathrm{p}_i) & \stackrel{\text{def}}{=} & Alt_n^0 \mathrm{U}\Big(\neg Alt_n^0 \ \wedge \ \neg Alt_n^0 \mathrm{U}\big(Alt_n^{n+1-i} \wedge \mathrm{qUr}\big)\Big); \\
t_n(\psi \wedge \psi') & \stackrel{\text{def}}{=} & t_n(\psi) \wedge t_n(\psi'); \\
t_n(\neg\psi) & \stackrel{\text{def}}{=} & \neg t_n(\psi); \\
t_n(\mathrm{X}\phi) & \stackrel{\text{def}}{=} & Alt_n^0 \mathrm{U}\Big(\neg Alt_n^0 \wedge (\neg Alt_n^0 \mathrm{U}(Alt_n^0 \wedge t_n(\phi))\Big); \\
t_n(\mathrm{F}\psi) & \stackrel{\text{def}}{=} & \mathrm{F}(Alt_n^0 \wedge t_n(\psi)); \\
t_n(\psi \mathrm{U} \psi') & \stackrel{\text{def}}{=} & (Alt_n^0 \Rightarrow t_n(\psi))\mathrm{U}(Alt_n^0 \wedge t_n(\psi')).
\end{array}
$$

La définition de $t_n(\mathrm{p}_i)$ est motivée par la séquence suivante dans $\mathcal{M}_n$:

$$
\{\mathrm{q},\mathrm{r}\}\cdot\emptyset\cdot\{\mathrm{q}, Alt_n^n\}\cdot\left\{\begin{array}{l}\emptyset \\ \{\mathrm{r}\}\end{array}\right. \cdot\{\mathrm{q}, Alt_n^{n-1}\}\cdot\ldots\{\mathrm{q}, Alt_n^{n+1-i}\}\cdot\left\{\begin{array}{l}\emptyset \\ \{\mathrm{r}\}\end{array}\right. \cdot\{\mathrm{q}, Alt_n^1\}\cdot\left\{\begin{array}{l}\emptyset \\ \{\mathrm{r}\}\end{array}\right.
$$

On peut montrer que $\mathcal{M}, s \models_\exists \phi$ ssi $\mathcal{M}_n, \langle s, 1\rangle \models_\exists t_n(\phi)$. $\mathcal{M}_n$ peut être construit en espace logarithmique en $|\mathcal{M}| + |\phi|$ tandis que ϕ_n peut être construite en espace logarithmique en $|\phi|$.

La PSPACE-complétude de SAT(LTL) et $\mathrm{MC}^\exists$(LTL) sont des résultats attribués à [SC85].

Des opérateurs temporels faisant référence au passé sont aussi traités dans cet article. Les constructions dans

cette section proviennent de [DS02].

Un état de l'art sur la complexité du model-checking pour les logiques temporelles peut être trouvé dans [Sch03].

2.5.2 Une variable avec l'opérateur "until" dans P

On a vu que $\mathrm{MC}^{\exists}(\mathrm{LTL}_2(\mathrm{U}))$ est PSPACE-difficile. Par contre, on peut montrer que $\mathrm{MC}^{\exists}(\mathrm{LTL}_1(\mathrm{U}))$ est dans P.

Définition 2.1. Soient $\sigma, \sigma' : \mathbb{N} \to \{\emptyset, \{\mathrm{p}\}\}$. $\sigma \approx \sigma' \overset{\text{def}}{\Leftrightarrow} \sigma$ et σ' sont les mêmes séquences modulo les répétitions de $\emptyset$ et $\{\mathrm{p}\}$. ∇

Par exemple,

$$\emptyset^3 \cdot \{\mathrm{p}\}^5 \cdot \emptyset^4 \cdot \{\mathrm{p}\}^\omega \approx \emptyset^5 \cdot \{\mathrm{p}\} \cdot \emptyset^{15} \cdot \{\mathrm{p}\}^\omega.$$

Lemme 2.7. Pour $\sigma, \sigma' : \mathbb{N} \to \{\emptyset, \{\mathrm{p}\}\}$ telles que $\sigma \approx \sigma'$, $\sigma \models \psi$ ssi $\sigma' \models \psi$ pour $\psi \in \mathrm{LTL}_1(\mathrm{U})$.

Le lemme ci-dessus est la conséquence d'un lemme plus général de L. Lamport [Lam83].

Tout modèle $\sigma : \mathbb{N} \to \{\emptyset, \{\mathrm{p}\}\}$ a une des formes suivantes, modulo $\approx$:
– $\sigma_1^n = (\{\mathrm{p}\} \cdot \emptyset)^n \cdot \{\mathrm{p}\}^\omega$;

– $\sigma_2^n = \emptyset \cdot (\{\mathrm{p}\} \cdot \emptyset)^n \cdot \{\mathrm{p}\}^\omega$;
– $\sigma_3^n = (\{\mathrm{p}\} \cdot \emptyset)^\omega$;
– $\sigma_4^n = (\emptyset \cdot \{\mathrm{p}\})^n \cdot \emptyset^\omega$;
– $\sigma_5^n = \{\mathrm{p}\} \cdot (\emptyset \cdot \{\mathrm{p}\})^n \cdot \emptyset^\omega$;
– $\sigma_6^n = (\emptyset \cdot \{\mathrm{p}\})^\omega$.

Ainsi tester l'existence d'un modèle pour $\phi \in \mathrm{LTL}_1(\mathrm{U})$, revient à chercher un modèle de la forme σ_j^n. Le lemme suivant permet de simplifier davantage.

Lemme 2.8. Pour $j \in \{1, \ldots, 6\}$, $\phi \in \mathrm{LTL}_1(\mathrm{U})$, et n supérieur ou égale à la hauteur temporelle de ϕ, $\sigma_j^{n+1} \models \phi$ ssi $\sigma_j^n \models \phi$.

Preuve. La preuve est par induction structurelle sur ϕ et utilise le fait que le premier suffixe d'un σ_j^n est un $\sigma_{j'}^{n'}$. Par exemple, le premier suffixe de σ_1^{n+1} est σ_2^n et le premier suffixe de σ_2^{n+1} est σ_1^{n+1}.

Le cas de base $ht(\phi) = 0$ est évident.

Hypothèse d'induction : Pour ψ telle que $|\psi| \leq N$, pour $j \in \{1, \ldots, 6\}$, pour $n \geq ht(\psi)$, $\sigma_j^n \models \psi$ ssi $\sigma_j^{n+1} \models \psi$.

Soit ψ telle que $|\psi| = N + 1$. Les cas où l'opérateur principal de ψ est Booléen est omis.

Supposons que $\psi = \psi_1 \mathrm{U} \psi_2$. Seul le cas $j = 1$ est présenté.

Supposons que $\sigma_1^{n+1} \models \psi$. Il existe donc $j' \geq 0$ tel que $\sigma_1^{n+1}, j' \models \psi_2$ et pour $0 \leq k < j'$, $\sigma_1^{n+1}, k \models \psi_1$.

Si $j' \geq 2$, alors $\sigma_1^{n+1}, 2 \models \psi_1 \mathrm{U} \psi_2$ et donc $\sigma_1^n \models \psi$.

Si $j' = 1$, alors $\sigma_1^{n+1}, 0 \models \psi_1$ et $\sigma_1^{n+1}, 1 \models \psi_2$ (équivalent à $\sigma_2^n, 0 \models \psi_2$). Comme $|\psi_1|$, $|\psi_2| \leq N$, par hypothèse d'induction $\sigma_1^n, 0 \models \psi_1$ et $\sigma_2^{n-1}, 0 \models \psi_2$ car $n - 1 \geq ht(\psi_1), ht(\psi_2)$. En fait $ht(\psi) \geq 1 + max(ht(\psi_1), ht(\psi_2))$. Ainsi $\sigma_1^n \models \psi$.

Si $j' = 0$, alors $\sigma_1^{n+1}, 0 \models \psi_2$. Comme $|\psi_2| \leq N$, par hypothèse d'induction $\sigma_1^n, 0 \models \psi_2$ car $n - 1 \geq ht(\psi_2)$. Ainsi $\sigma_1^n \models \psi$.

Supposons à présent que $\sigma_1^n \models \psi$. Il existe donc $j' \geq 0$ tel que $\sigma_1^n, j' \models \psi_2$ et pour $0 \leq k < j'$, $\sigma_1^n, k \models \psi_1$. La preuve est analogue au cas précédent en analysant les différents cas pour j'.

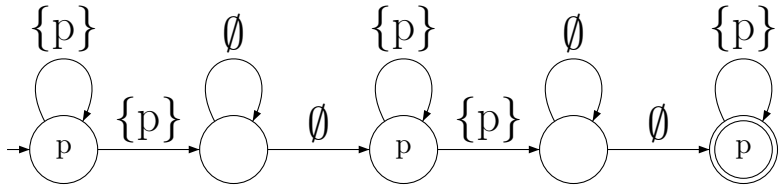
C.Q.F.D.

Lemme 2.9. Pour $i \in \{1, \ldots, 6\}$ et pour $n \in \mathbb{N}$, il existe de automates de Büchi $\mathcal{A}_i^{\geq n}$ et $\mathcal{A}_i^{=n}$ tels que

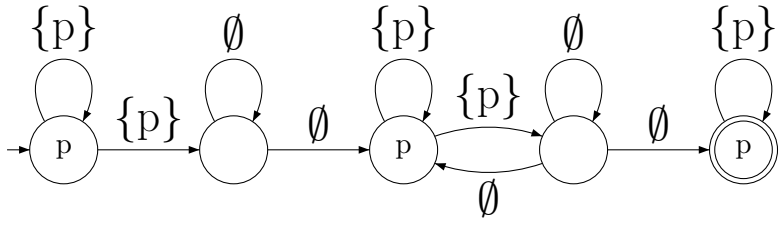
- $\mathcal{A}_i^{\geq n}$ et $\mathcal{A}_i^{=n}$ ont $\mathcal{O}(n)$ états et peuvent être construits uniformément avec un espace mémoire en $\mathcal{O}(log\ n)$;
- $\mathrm{L}(\mathcal{A}_i^{\geq n})$ est précisément l'ensemble des mots infinis sur $\{\{\mathrm{p}\}, \emptyset\}$ tels que $\sigma \in \mathrm{L}(\mathcal{A}_i^{\geq n})$ ssi il existe $m \geq n$ tel que $\sigma \approx \sigma_i^m$;
- $\mathrm{L}(\mathcal{A}_i^{=n})$ est précisément l'ensemble des mots infinis sur $\{\{\mathrm{p}\}, \emptyset\}$ tels que $\sigma \in \mathrm{L}(\mathcal{A}_i^{\geq n})$ ssi $\sigma \approx \sigma_i^n$.

Quelques exemples d'automates :

$\sigma_1^{=2}$:



$\sigma_1^{\geq 2}$:



En utilisant le lemme suivant :

Lemme 2.10. Etant donné un automate de Büchi $\mathcal{A}$, $\mathcal{M}$ une structure de Kripke, et s_0 un état de $\mathcal{M}$ vérifier s'il y a une exécution de $\mathcal{M}$ commençant en s_0 qui est un mot de $\mathrm{L}(\mathcal{A})$ peut se calculer en temps polynomial en $|\mathcal{M}| + |\mathcal{A}|$ avec une machine déterministe.

On obtient alors :

Proposition 2.11. $\mathrm{MC}^{\exists}(\mathrm{LTL}_1(\mathrm{U}))$ est dans P.

Preuve. Soient $\mathcal{M} = \langle W, R, L\rangle$ une structure de Kripke finie et totale, $s_0 \in W$ et $\phi \in \mathrm{LTL}_1(\mathrm{U})$.

$\mathcal{M}, s_0 \models_{\exists} \phi$ ssi il existe $j \in \{1, \ldots, 6\}$ et $n \in \mathbb{N}$ tels que $\sigma_j^n \models \phi$ et σ_j^n est un chemin de $\mathcal{M}$ commençant en s_0 modulo $\approx$.

D'après le lemme 2.8, $\mathcal{M}, s_0 \models_{\exists}$ ssi il existe $j \in \{1, \ldots, 6\}$ et $n \leq ht(\phi)$ tels que $\sigma_j^n \models \phi$ et σ_j^n est un chemin de $\mathcal{M}$ commençant en s_0 modulo $\approx$.

Pour $n \leq ht(\phi)$, vérifier si σ_j^n est un chemin de $\mathcal{M}$ commençant en s_0 modulo $\approx$ se calcule en temps polynomial en $|\mathcal{M}|+|\phi|$ d'après le lemme 2.10 et le lemme 2.9.

De plus, pour $n \leq ht(\phi)$, vérifier si $\sigma_j^n \models \phi$ se calcule en temps polynomial en utilisant le lemme 2.9 et la com-

plexité du problème de model-checking pour CTL.

Ainsi, vérifier si $\mathcal{M}, s_0 \models_\exists \phi$ se calcule en temps polynomial en $|\mathcal{M}| + |\phi|$. C.Q.F.D.

Par opposition, $\mathrm{MC}^\exists(\mathrm{LTL}_1(\mathrm{F}, \mathrm{X}))$ et $\mathrm{SAT}(\mathrm{LTL}_1(\mathrm{F}, \mathrm{X}))$ sont PSPACE-difficiles.

2.6 Borner la hauteur temporelle à 1

Nous nous intéressons au fragment de LTL avec des formules de hauteur temporelle au plus 1. Nous allons montrer que $\mathrm{SAT}(\mathrm{LTL}^1(\mathrm{U}, \mathrm{X}))$ et $\mathrm{MC}^\exists(\mathrm{LTL}^1(\mathrm{U}, \mathrm{X}))$ sont NP-complets.

Définition 2.2. (séquence extraite) Soient $\sigma : \mathbb{N} \to \mathcal{P}(\mathrm{PROP})$ et $t = (n_i)_{i \in \mathbb{N}}$ une séquence d'entiers positifs telle qu'il existe $k \geq 0$ vérifiant $n_0 < \ldots < n_k$ et pour $j > k$, $n_j \geq n_k$. La séquence $\sigma' : \mathbb{N} \to \mathcal{P}(\mathrm{PROP})$ extraite de σ et t est définie ainsi : pour $j \in \mathbb{N}$, $\sigma'(j) = \sigma(n_j)$. ∇

Définition 2.3. (témoins) Soient $\phi \in \mathrm{LTL}^1(\mathrm{U}, \mathrm{X})$ et $\sigma : \mathbb{N} \to \mathcal{P}(\mathrm{PROP})$. Nous allons extraire de σ un ensemble de positions "témoins" de σ qui témoignent de la satisfaction ou de la non satisfaction de $\sigma \models \phi$.

0 est toujours un témoin et pour toute formule temporelle sous-formule de ϕ, nous allons définir des témoins :

- $\mathrm{X}\psi$ a pour témoin 1 ;
- pour $\psi_1 \mathrm{U} \psi_2$ on distingue trois cas :
  1. si $\sigma \models \psi_1 \mathrm{U} \psi_2$, le témoin i est le plus petit indice tel que $\sigma, i \models \psi_2$;
  2. si $\sigma \not\models \mathrm{F}\psi_2$ alors aucun témoin n'est associé à cette formule ;
  3. si $\sigma \not\models \psi_1 \mathrm{U} \psi_2$ et $\sigma \models \mathrm{F}\psi_2$ alors le témoin i est le plus petit indice tel que $\sigma, i \not\models \psi_1$.

∇

Exemple 2.12.

$$\phi = ((\mathrm{p} \vee \neg \mathrm{r})\mathrm{U}\mathrm{q}) \wedge \neg(\mathrm{p}\mathrm{U}\mathrm{q}) \wedge (\neg \mathrm{X}\mathrm{p} \vee \neg(\mathrm{p}\mathrm{U}\mathrm{r})) \wedge \neg \mathrm{r}$$

$$\sigma = \{\mathrm{p}\}^4 \cdot \emptyset \cdot \{\mathrm{p}\}\{\mathrm{q}\}^2\{\mathrm{p}\} \cdots$$

- $\sigma \models (\mathrm{p} \vee \neg \mathrm{r})\mathrm{U}\mathrm{q}$ et $(\mathrm{p} \vee \neg \mathrm{r})\mathrm{U}\mathrm{q}$ a pour témoin 6 ;
- $\sigma \not\models \mathrm{p}\mathrm{U}\mathrm{q}$ et $\mathrm{p}\mathrm{U}\mathrm{q}$ a pour témoin 4 ;
- $\sigma \models \mathrm{X}\mathrm{p}$ et $\mathrm{X}\mathrm{p}$ a pour témoin 1 ;
- $\sigma \not\models \mathrm{p}\mathrm{U}\mathrm{r}$ et $\mathrm{p}\mathrm{U}\mathrm{r}$ n'a pas de témoin.

∇

Lemme 2.13. Soient $\phi \in \mathrm{LTL}^1(\mathrm{U}, \mathrm{X})$, $\sigma : \mathbb{N} \to \mathcal{P}(\mathrm{PROP})$, et $t = (n_i)_{i \in \mathbb{N}}$ une séquence d'entiers positifs vérifiant les conditions de la définition 2.2 et contenant les témoins

de ϕ dans σ dans la partie $\{n_0, \ldots, n_k\}$. La séquence extraite de σ et t vérifie : pour $\psi \in \mathrm{sub}(\phi)$, $\sigma \models \psi$ ssi $\sigma' \models \psi$.

Preuve. Par induction sur la structure des sous-formules. Le cas de base ($\psi = \mathrm{p}$) et les cas de l'étape d'induction relatifs à $\neg$, $\vee$, $\wedge$, et X sont omis.

Seul le cas $\psi = \psi_1 \mathrm{U} \psi_2$ est traité.

Supposons que $\sigma, 0 \models \psi_1 \mathrm{U} \psi_2$ et que i soit le témoin de $\psi_1 \mathrm{U} \psi_2$ avec $i = n_l$.

Nous avons

$$\sigma, 0 \models \psi_1 \cdots \sigma, n_l - 1 \models \psi_1, \quad \sigma, n_l \models \psi_2.$$

Par conséquent, comme les formules ψ_1 et ψ_2 sont sans opérateurs temporels,

$$\sigma', 0 \models \psi_1 \cdots \sigma', l - 1 \models \psi_1, \quad \sigma', l \models \psi_2.$$

Par conséquent, $\sigma' \models \psi_1 \mathrm{U} \psi_2$.

Supposons à présent que $\sigma \not\models \psi_1 \mathrm{U} \psi_2$.

Si $\sigma \not\models \mathrm{F}\psi_2$, alors pour $l \in \mathbb{N}$, $\sigma, l \not\models \psi_2$. En particulier, pour $l \in \mathbb{N}$, $\sigma, n_l \not\models \psi_2$ et donc pour $l \in \mathbb{N}$, $\sigma', l \not\models \psi_2$. Ainsi $\sigma' \not\models \psi_1 \mathrm{U} \psi_2$.

Si $\sigma \models \mathrm{F}\psi_2$, alors soit i le témoin de $\psi_1 \mathrm{U} \psi_2$ avec $i = n_l$.

Nous avons

$$\sigma, 0 \models \neg\psi_2 \wedge \psi_1 \cdots \sigma, n_l - 1 \models \neg\psi_2 \wedge \psi_1, \quad \sigma, n_l \models \neg\psi_2 \wedge \neg\psi_1.$$

Par conséquent, comme les formules ψ_1 et ψ_2 sont sans opérateurs temporels,

$$\sigma', 0 \models \neg\psi_2 \wedge \psi_1 \cdots \sigma', l - 1 \models \neg\psi_2 \wedge \psi_1, \quad \sigma', l \models \neg\psi_2 \wedge \neg\psi_1.$$

Par conséquent, $\sigma' \not\models \psi_1 \mathrm{U} \psi_2$. C.Q.F.D.

Proposition 2.14. Soit $\phi \in \mathrm{LTL}^1(\mathrm{U}, \mathrm{X})$. ϕ est satisfaisable ssi il existe $\sigma : \mathbb{N} \to \mathcal{P}(\mathrm{PROP})$ tel que $\sigma \models \phi$ et pour $i, j \geq |\phi|$, $\sigma(i) = \sigma(j)$.

Preuve. Soit σ tel que $\sigma, 0 \models \phi$ avec témoins $n_0, \ldots, n_k$ pour un $k \leq |\phi|$. D'après le lemme 2.13,

$$\sigma(n_0) \cdots \sigma(n_{k-1}) \sigma(n_k)^\omega \models \phi.$$

C.Q.F.D.

Corollaire 2.15. SAT($\mathrm{LTL}^1(\mathrm{U},\mathrm{X})$) est dans NP et donc NP-complet.

Qu'en est-il de la complexité de $\mathrm{MC}^{\exists}(\mathrm{LTL}^1(\mathrm{U},\mathrm{X}))$?

Exercice 2.4. Montrer que $\mathrm{MC}^{\exists}(\mathrm{LTL}^1(\mathrm{F}))$ est NP-difficile en réduisant SAT.

Proposition 2.16. $\mathrm{MC}^{\exists}(\mathrm{LTL}^1(\mathrm{U},\mathrm{X}))$ est dans NP.

Preuve. Soient $\mathcal{M} = \langle W, R, L\rangle$ une structure de Kripke finie et totale, $s_0 \in W$ et $\phi \in \mathrm{LTL}^1(\mathrm{U},\mathrm{X})$.

$\mathcal{M}, s_0 \models_{\exists} \phi$ ssi il existe un chemin infini $s_0 s_1 \ldots$ dans $\mathcal{M}$ tel que $\sigma \models \phi$ avec $\sigma(i) = L(s_i)$ pour $i \in \mathbb{N}$.

Soit $\{n_0, \ldots, n_k\}$ l'ensemble des témoins de ϕ dans $\mathcal{M}$.

Soit n_{k+1} le plus petit indice strictement supérieur à n_k tel que $s_{n_{k+1}} = s_{n_{k+1}+j}$ pour un $j \leq |W|$.

On a bien $n_{k+1} + j - n_k \leq |W| + 1$.

Ainsi $\mathcal{M}, s_0 \models_{\exists} \phi$ ssi ll existe donc un chemin $\sigma = s_0 s_1 \cdots s_l (s_{l+1} \ldots s_{l+m})^{\omega}$ dans $\mathcal{M}$ tel que
– $\sigma \models \phi$;

– $l \leq |\phi| \times |W|$;
entre deux témoins s_{n_i} et $s_{n_{i+1}}$ on peut trouver un chemin de longueur au plus $|W|$ et le nombre de témoins est borné par $|\phi|$;
– $m \leq |W|$.

Pour obtenir la borne supérieure, il suffit de remarquer que l'on devine une structure témoin σ de taille polynomiale et que l'on doit vérifier si $\sigma \models \phi$ et qu'elle est un chemin de $\mathcal{M}$ commençant en s_0. C.Q.F.D.

3 Rappel : borne supérieure PSPACE pour LTL

3.1 Automates de Büchi

Un *automate fini* $\mathcal{A}$ est une structure $\mathcal{A} = \langle \Sigma, S, S_0, \rho, F \rangle$ telle que

– Σ est un ensemble fini de symboles, appelé *alphabet*;
– S est un ensemble fini d'*états*;
– $S_0 \subseteq S$ est un ensemble non-vide d'*états initiaux*;
– $\rho : S \times \Sigma \to \mathcal{P}(S)$ est une *fonction de transition*;
– $F \subseteq S$ est un ensemble d'*états acceptants* ou *finaux*.

$\mathcal{A}$ est dit *déterministe* ssi $\text{card}(S_0) = 1$ et pour $s \in S$, pour $a \in \Sigma$, $\text{card}(\rho(s, a)) \leq 1$.

Un *calcul* c est une séquence infinie $s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \ldots$ telle pour $i \geq 0$, $s_{i+1} \in \rho(s_i, a_i)$ et $s_0 \in S_0$.

Étant donné un calcul c, $inf(c)$ dénote l'ensemble des états qui apparaissent infiniment souvent dans c.

Un calcul c est *réussi à la Büchi* $\overset{\text{def}}{\Leftrightarrow}$ $inf(c) \cap F \neq \emptyset$.

Le mot infini $\sigma \in \Sigma^\omega$ est reconnu par le calcul réussi $c = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \ldots$ si $\sigma = a_0 a_1 a_2 \ldots$.

Le mot infini σ est dit *accepté par* $\mathcal{A}$. On note L($\mathcal{A}$) le langage des mots infinis acceptés par l'automate de Büchi $\mathcal{A}$.

L'ensemble Modèles(ϕ) pour $\phi \in$ LTL est un langage de mots infinis sur l'alphabet $\mathcal{P}$(PROP). Nous allons montrer qu'il existe un automate de Büchi $\mathcal{A}_\phi$ tel que L($\mathcal{A}_\phi$) = Modèles(ϕ).

Un automate généralisé est une structure

$$\mathcal{A} = \langle \Sigma, S, S_0, \rho, \{F_1, \ldots, F_k\} \rangle$$

telle que $F_1, \ldots, F_k \subseteq S$. On garde la même notion de calcul.

Un calcul c est *réussi* $\stackrel{\text{def}}{\Leftrightarrow}$ pour $1 \leq i \leq k$, $inf(c) \cap F_i \neq \emptyset$.

Il existe d'autres modes d'acceptation mais nous n'en parlerons pas ici.

Lemme 3.1. Soit $\mathcal{A} = \langle \Sigma, S, S_0, \rho, \{F_1, \ldots, F_k\} \rangle$ un automate généralisé. Il existe un automate de Büchi $\mathcal{A}^b = \langle \Sigma, S^b, S_0^b, \rho^b, F^b \rangle$ tel que $\mathrm{L}(\mathcal{A}^b) = \mathrm{L}(\mathcal{A})$.

En fait on peut construire $\mathcal{A}^b$ en utilisant un espace mémoire logarithmique en $|\phi|$. Les automates généralisés sont en fait des automates de Büchi spécialisés pour la traduction des formules de LTL.

Une expression ω-rationnelle est une expression rationnelle contenant l'opérateur unaire $\cdot^\omega$ d'itération infinie (en plus des opérateurs éventuels d'union, de concaténation et d'itération finie). La classe des langages reconnus par des automates de Büchi admet diverses caractérisations (avec la logique monadique du second-ordre avec un successeur, avec LTL augmentée d'opérateurs de point fixe, ...) : par exemple un langage est ω-régulier si et seulement si il est accepté par un automate de Büchi.

Proposition 3.2. La famille des langages ω-réguliers est fermée par intersection, union et complémentation.

Il est bien connu qu'un ensemble de mots finis est reconnu par un automate fini ssi il est reconnu par un automate fini déterministe. Cette situation est différente pour les langages de mots infinis reconnus par des automates de Büchi.

Proposition 3.3. Il existe un langage ω-régulier qui n'est reconnu par aucun automate de Büchi déterministe.

Le problème du vide pour les automates de Büchi est le suivant :

entrée : un automate de Büchi $\mathcal{A}$;

sortie : 1, si $\mathrm{L}(\mathcal{A}) \neq \emptyset$, 0 sinon.

Proposition 3.4. [VW94] Le problème du vide pour les automates de Büchi est NLOGSPACE-complet.

Le problème du vide sur les automates reconnaissant des mots finis est aussi NLOGSPACE-complet. Par contre,

Proposition 3.5. Le problème de l'universalité pour les automates de Büchi est PSPACE-complet.

3.2 Traduction de formules LTL en automates

Dans la suite, nous rappelons l'approche qui réduit le problème du model-checking (et de la satisfaisabilité) vers le problème du vide pour les automates de Büchi. Il s'agit de transformer une formule ϕ en un automate $\mathcal{A}_\phi$ tel que Modèles$(\phi) = \mathrm{L}(\mathcal{A}_\phi)$.

Cette approche permet à la fois d'obtenir les bornes de complexité optimales et de proposer des algorithmes qui peuvent se révéler efficaces en pratique.

Soient $\mathcal{M} = \langle W, R, L\rangle$ une structure de Kripke finie et totale, $s_0 \in W$. Un automate de Büchi $\mathcal{A}_{\mathcal{M},s_0}$ tel que $\mathrm{L}(\mathcal{A}_{\mathcal{M},s_0}) = \text{Chemins}(\mathcal{M}, s_0)$ est défini de la façon suivante

$$\mathcal{A}_{\mathcal{M},s_0} = \langle \mathcal{P}(\mathrm{PROP}), W, \{s_0\}, \rho, W\rangle$$

où $\rho(s,a) \stackrel{\text{def}}{=} \{s' : \langle s, s'\rangle \in R,\ a = L(s)\}$ pour $s \in W$ et $a \subseteq \mathrm{PROP}$.

Le reste de cette section est dédiée à montrer le résultat suivant.

Proposition 3.6. [VW94] Pour toute formule de ϕ, il existe un automate de Büchi $\mathcal{A}_\phi$ tel que $\mathrm{L}(\mathcal{A}_\phi) = \text{Modèles}(\phi)$ et $|\mathcal{A}_\phi|$ est en $2^{\mathcal{O}(|\phi|)}$.

La proposition 3.6 se comprend lorsqu'on réduit l'ensemble des variables propositionnelles PROP aux variables apparaissant dans ϕ. De même dans la définition de $\mathcal{A}_{\mathcal{M},s_0}$ ci-dessus, PROP est fini.

Définition 3.1. Soit $\phi \in$ LTL. La *fermeture* de ϕ, notée $clo(\phi)$, est le plus petit ensemble contenant les sous-formules de ϕ et leur négation modulo le fait que nous identifions $\neg\neg\psi$ avec ψ pour toute formule ψ. ∇

Définition 3.2. Soient $\phi \in$ LTL et σ un modèle de LTL. La *séquence issue de* σ et ϕ est un mot infini de $\mathcal{P}(clo(\phi))^\omega$, noté

$$satseq(\sigma, \phi) = satseq(\sigma, \phi, 0)satseq(\sigma, \phi, 1)\ldots,$$

tel que pour $i \geq 0$, $satseq(\sigma, \phi, i) \stackrel{\text{def}}{=} \{\psi \in clo(\phi) : \sigma, i \models \psi\}$. ∇

Soient $\phi \in$ LTL et σ un modèle de LTL tels que $\sigma \models \phi$. Le mot infini $satseq(\sigma, \phi)$ vérifie les propriétés suivantes (pour $i \in \mathbb{N}$, p, $\psi, \psi' \in clo(\phi)$) :

1. p $\in satseq(\sigma, \phi, i)$ ssi p $\in \sigma_i$;
2. $\phi \in satseq(\sigma, \phi, 0)$;
3. $\psi \wedge \psi' \in satseq(\sigma, \phi, i)$ ssi $\psi \in satseq(\sigma, \phi, i)$ et $\psi' \in satseq(\sigma, \phi, i)$;
4. $\neg\psi \in satseq(\sigma, \phi, i)$ ssi $\psi \notin satseq(\sigma, \phi, i)$;

5. $\mathrm{X}\psi \in satseq(\sigma, \phi, i)$ ssi $\psi \in satseq(\sigma, \phi, i+1)$;
6. $\psi \mathrm{U} \psi' \in satseq(\sigma, \phi, i)$ ssi $\psi' \in satseq(\sigma, \phi, i)$ ou bien $\psi \in satseq(\sigma, \phi, i)$ et $\psi \mathrm{U} \psi' \in satseq(\sigma, \phi, i+1)$;
7. si $\psi \mathrm{U} \psi' \in satseq(\sigma, \phi, i)$ alors il existe $j \geq i$ tel que $\psi' \in satseq(\sigma, \phi, j)$.

Les propriétés 1.-4. ne font appel qu'à l'ensemble $satseq(\sigma, \phi, i)$ tandis que les propriétés 5. et 6. ne font appel qu'aux ensembles $satseq(\sigma, \phi, i)$ et $satseq(\sigma, \phi, i+1)$. On parlera ici de propriétés *locales*. Quant à la propriété 7., il n'y a pas localité car l'entier j peut être arbitrairement grand.

Définition 3.3. Une *séquence de Hintikka* pour σ et ϕ est un mot infini $\alpha_0 \alpha_1 \ldots \in \mathcal{P}(clo(\phi))^\omega$ vérifiant les conditions suivantes (pour $i \in \mathbb{N}$, $\mathrm{p}, \psi, \psi' \in clo(\phi)$) :

1. $\mathrm{p} \in \alpha_i$ ssi $\mathrm{p} \in \sigma_i$;
2. $\phi \in \alpha_0$;
3. $\psi \wedge \psi' \in \alpha_i$ ssi $\psi \in \alpha_i$ et $\psi' \in \alpha_i$;
4. $\neg\psi \in \alpha_i$ ssi $\psi \notin \alpha_i$;
5. $\mathrm{X}\psi \in \alpha_i$ ssi $\psi \in \alpha_{i+1}$;
6. $\psi \mathrm{U} \psi' \in \alpha_i$ ssi $\psi' \in \alpha_i$ ou bien $\psi \in \alpha_i$ et $\psi \mathrm{U} \psi' \in \alpha_{i+1}$;
7. si $\psi \mathrm{U} \psi' \in \alpha_i$ alors il existe $j \geq i$ tel que $\psi' \in \alpha_j$.

∇

La proposition 3.7 ci-dessous énonce l'adéquation entre les séquences $satseq(\sigma, \phi)$ définies sémantiquement et les séquences de Hintikka définies de façon syntaxique. Il s'agit d'un résultat intermédiaire mais central qui permet de saisir les constructions à venir d'automates.

Proposition 3.7. $\alpha_0\alpha_1 \ldots \in \mathcal{P}(clo(\phi))^\omega$ est une séquence d'Hintikka pour σ et ϕ ssi $\alpha_0\alpha_1 \ldots = satseq(\sigma, \phi)$.

Soit $\phi \in$ LTL. Soit $\mathcal{A}_\phi^H$ l'automate généralisé suivant :

$$\langle \mathcal{P}(\text{PROP}) \times S, S, S_0, \rho, \{F_1, \ldots, F_k\} \rangle$$

tel que

- S est l'ensemble des sous-ensembles de $clo(\phi)$ tel que chaque élément de S est maximalement consistent au niveau propositionnel ;
- $S_0 \stackrel{\text{def}}{=} \{\alpha \in S : \phi \in \alpha\}$;
- $\rho(\alpha, \langle a, \beta \rangle) \stackrel{\text{def}}{=} \{\alpha' \in S : \alpha = \beta,\ a = \text{PROP} \cap \alpha,\ \alpha \rightsquigarrow \alpha'\}$ où la relation $\rightsquigarrow$ est définie comme suit : $\alpha \rightsquigarrow \alpha'$ ssi
  1. $\text{X}\psi \in \alpha$ ssi $\psi \in \alpha'$;
  2. $\psi \text{U} \psi' \in \alpha$ ssi $\psi' \in \alpha$ ou bien $\psi \in \alpha$ et $\psi \text{U} \psi' \in \alpha'$;

– Si l'opérateur temporel U n'apparaît pas dans ϕ, alors $\{F_1\} \stackrel{\text{def}}{=} \{S\}$. Sinon supposons que $\{\psi_1 \mathrm{U} \psi'_1, \ldots, \psi_k \mathrm{U} \psi'_k\}$ est l'ensemble des formules de $clo(\phi)$ dont l'opérateur principal est U. Nous avons alors $F_i = \{\alpha \in S : \psi_i \mathrm{U} \psi'_i \notin \alpha \text{ ou bien } \psi'_i \in \alpha\}$.

Proposition 3.8. $\mathcal{A}_\phi^H$ accepte $\langle a_0, \alpha_0 \rangle \langle a_1, \alpha_1 \rangle \ldots$ ssi $\alpha_0 \alpha_1 \ldots$ est une séquence d'Hintikka pour $a_0 a_1 \ldots$ et ϕ.

À partir de $\mathcal{A}_\phi^H$, nous construisons un automate généralisé $\mathcal{A}_\phi$ en simplifiant l'alphabet :

$$\mathcal{A}_\phi = \langle \mathcal{P}(\mathrm{PROP}), S, S_0, \rho', \{F_1, \ldots, F_k\} \rangle$$

où $\rho'(\alpha, a) \stackrel{\text{def}}{=} \{\alpha' \in S : a = \mathrm{PROP} \cap \alpha,\ \alpha \rightsquigarrow \alpha'\}$.

Proposition 3.9. $\mathrm{L}(\mathcal{A}_\phi) = \text{Modèles}(\phi)$.

Il est temps de récolter les fruits de nos efforts.

ϕ est satisfaisable ssi $\text{Modèles}(\phi) \neq \emptyset$. Comme nous l'avons vu, le problème du vide pour les automates de Büchi est NLOGSPACE-complet (proposition 3.4) et un automate généralisé peut être transformé en un automate de Büchi en espace logarithmique reconnaissant le même langage de mots infinis (lemme 3.1).

La taille de $\mathcal{A}_\phi$ est en $2^{\mathcal{O}(|\phi|)}$. En testant le vide de $\mathrm{L}(\mathcal{A}_\phi)$ en construisant $\mathcal{A}_\phi$ à la volée (cf. la preuve de la

proposition 3.4), on peut conclure que SAT(LTL) $\in$ NPSPACE.

Construire $\mathcal{A}_\phi$ à la volée signifie que la structure entière n'est jamais construite.

Le Théorème de Savitch établit que NPSPACE= PSPACE ce qui implique

Proposition 3.10. SAT(LTL) $\in$ PSPACE.

En fait, pour tout polynôme $p(n) \geq log(n)$, nous avons $\text{NSPACE}(p(n)) \subseteq \text{SPACE}((p(n))^2)$ (voir par exemple [Pap94, page 150]).

On a vu que $\mathcal{M}, s_0 \models_\forall \phi$ ssi $\text{L}(\mathcal{A}_{\mathcal{M},s_0}) \subseteq \text{L}(\mathcal{A}_\phi)$.

Par conséquent, $\mathcal{M}, s_0 \models_\forall \phi$ ssi $\text{L}(\mathcal{A}_{\mathcal{M},s_0}) \cap \text{L}(\mathcal{A}_{\neg\phi}) = \emptyset$.

Étant donné l'automate $\mathcal{A}_{\neg\phi} = \langle \mathcal{P}(\text{PROP}), S, S_0, \rho, F \rangle$, l'automate de Büchi

$$\mathcal{M} \mid\mid \mathcal{A}_{\neg\phi} = \langle \{0\}, W \times S, \{s_0\} \times S_0, \rho', W \times F \rangle$$

avec $\rho'(\langle u, s \rangle, 0) = \{\langle v, s' \rangle : \langle u, v \rangle \in R, s' \in \rho(s, L(u))\}$ reconnait précisément le langage de mots infinis $\text{L}(\mathcal{A}_{\mathcal{M},s_0}) \cap \text{L}(\mathcal{A}_{\neg\phi})$.

La taille de $\mathcal{M} \mid\mid \mathcal{A}_{\neg\phi}$ est en $2^{\mathcal{O}(|\phi|)} \times |\mathcal{M}|$ et donc en le construisant à la volée, $\mathcal{M}, s_0 \models_\forall \phi$ peut être vérifié avec un espace mémoire en $\mathcal{O}((|\phi| + log|\mathcal{M}|))$ par un algorithme non-déterministe.

Proposition 3.11. $\mathrm{MC}^\forall(\mathrm{LTL}) \in$ PSPACE.

Corollaire 3.12. $\mathrm{MC}^\exists(\mathrm{LTL}) \in$ PSPACE.

Le corollaire 3.12 est aussi un corollaire de la proposition 3.10 et de la proposition 2.1. Le gain de l'approche ci-dessus est de n'utiliser qu'un espace $\mathcal{O}((|\phi|+log|\mathcal{M}|)^2)$ avec un algorithme déterministe.

L'approche décrite dans cette partie du cours est issue de [VW94, Var96].

4 LTL étendu aux opérateurs réguliers

4.1 Définition

D'après le Théorème de Kamp [Kam68], LTL a le même pouvoir d'expression que la logique du premier ordre à un successeur. Cependant, il existe des propriétés assez simples que nous aimerions exprimer qui ne peuvent l'être avec LTL.

Proposition 4.1. [Wol83] Il n'existe pas de formule de LTL ϕ construite sur la seule variable propositionnelle p telle que Modèles(ϕ) est l'ensemble des structures de LTL pour lesquelles p est vérifiée dans au moins tous les états pairs.

Preuve. Soit ϕ une formule construite sur la seule variable propositionnelle p et on note $|\phi|_{\mathrm{X}}$ le nombre d'occurrences du symbole X apparaissant dans la formule ϕ. Soit $\mathbb{N}(\phi)$ l'ensemble d'entiers suivant :

$$\mathbb{N}(\phi) \stackrel{\text{def}}{=} \{i \in \mathbb{N} : \{\mathrm{p}\}^i \cdot \emptyset \cdot \{\mathrm{p}\}^\omega \models \phi\}.$$

Nous allons montrer que pour $n \geq |\phi|_{\mathrm{X}} + 1$, $n \in \mathbb{N}(\phi)$ ssi $n + 1 \in \mathbb{N}(\phi)$ (preuve par induction structurelle).

Cas de base : Si $\phi = \mathrm{p}$ alors $\mathbb{N}(\phi) = \mathbb{N} \setminus \{0\}$ et $|\phi|_{\mathrm{X}} = 0$. On a bien pour $n \geq 1$, $n \in \mathbb{N}(\phi)$ ssi $n + 1 \in \mathbb{N}(\phi)$.

Hypothèse d'induction : pour ϕ telle que $|\phi| \leq N$, pour $n \geq |\phi|_{\mathrm{X}} + 1$, $n \in \mathbb{N}(\phi)$ ssi $n + 1 \in \mathbb{N}(\phi)$.

Soit ϕ avec $|\phi| \leq N + 1$.

Cas 1 : $\phi = \phi_1 \wedge \phi_2$ (les cas avec $\neg$ et $\vee$ sont analogues).

$|\phi|_{\mathrm{X}} = |\phi_1|_{\mathrm{X}} + |\phi_2|_{\mathrm{X}}$.

Nous avons l'équivalence entre les propositions suivantes ($n \geq |\phi|_{\mathrm{X}} + 1$) :
– $n \in \mathbb{N}(\phi)$;
– $n \in \mathbb{N}(\phi_1)$ et $n \in \mathbb{N}(\phi_2)$ (sémantique de $\wedge$) ;
– $n + 1 \in \mathbb{N}(\phi_1)$ et $n + 1 \in \mathbb{N}(\phi_2)$
(par hypothèse d'induction car $|\phi_1|, |\phi_2| \leq N$ et $n \geq |\phi_1|_{\mathrm{X}} + 1, |\phi_2|_{\mathrm{X}} + 1$) ;
– $n + 1 \in \mathbb{N}(\phi)$ (sémantique de $\wedge$).

Cas 2 : $\phi = \mathrm{X}\psi$.

$|\phi|_{\mathrm{X}} = 1 + |\psi|_{\mathrm{X}}$.

Nous avons l'équivalence entre les propositions suivantes ($n \geq |\phi|_{\mathrm{X}} + 1 \geq 2$) :
– $n \in \mathbb{N}(\phi)$;
– $n - 1 \in \mathbb{N}(\psi)$ (sémantique de X) ;
– $n \in \mathbb{N}(\psi)$
(par hypothèse d'induction car $|\psi| \leq N$ et $n-1 \geq |\psi|_{\mathrm{X}}$) ;
– $n + 1 \in \mathbb{N}(\phi)$ (sémantique de X).

Cas 3 : $\phi = \phi_1 \mathrm{U} \phi_2$.

$|\phi|_{\mathrm{X}} = |\phi_1|_{\mathrm{X}} + |\phi_2|_{\mathrm{X}}$.

Soit $n \geq |\phi|_{\mathrm{X}} + 1$ et supposons $\sigma^n = \{\mathrm{p}\}^n\emptyset\{\mathrm{p}\}^\omega \models \phi$. Il existe $j \geq 0$ tel que $\sigma^n, j \models \phi_2$ et pour $0 \leq k < j$, $\sigma^n, k \models \phi_1$.

Cas 3.1 : $j = 0$

$|\phi_2| \leq N$ et $n \geq |\phi_2|_{\mathrm{X}} + 1$.

Par hypothèse d'induction, $n+1 \in \mathbb{N}(\phi_2)$ et donc $n + 1 \in \mathbb{N}(\phi)$.

Cas 3.2 : $j \geq 1$

$|\phi_1| \leq N$ et $n \geq |\phi_1|_{\mathrm{X}} + 1$.

Par hypothèse d'induction, $n+1 \in \mathbb{N}(\phi_1)$ et donc $n + 1 \in \mathbb{N}(\phi)$.

Supposons à présent que $\sigma^{n+1} = \{\mathrm{p}\}^{n+1}\emptyset\{\mathrm{p}\}^\omega \models \phi$. Il existe $j \geq 0$ tel que $\sigma^{n+1}, j \models \phi_2$ et pour $0 \leq k < j$, $\sigma^{n+1}, k \models \phi_1$.

3.1(bis) : $j = 0$

$|\phi_2| \leq N$ et $n \geq |\phi_2|_{\mathrm{X}} + 1$.

Par hypothèse d'induction, $n \in \mathbb{N}(\phi_2)$ et donc $n \in \mathbb{N}(\phi)$.

3.2(bis) : $j \geq 1$

$\sigma^{n+1}, 1 \models \phi$ et donc $n \in \mathbb{N}(\phi)$.

Raisonnons maintenant par l'absurde. Supposons qu'il existe une formule de LTL ϕ construite sur p telle que Modèles(ϕ) est l'ensemble des structures de LTL telles que la variable propositionnelle p soit vérifiée dans au moins tous les états pairs.

On a $|\phi|_{\mathrm{X}} + 1 \in \mathbb{N}(\phi)$ ssi $|\phi|_{\mathrm{X}} + 2 \in \mathbb{N}(\phi)$, ce qui conduit à une contradiction car une unique structure parmi $\{\mathrm{p}\}^{|\phi|_{\mathrm{X}}+1} \cdot \emptyset \cdot \{\mathrm{p}\}^{\omega}$ et $\{\mathrm{p}\}^{|\phi|_{\mathrm{X}}+2} \cdot \emptyset \cdot \{\mathrm{p}\}^{\omega}$ appartient à Modèles(ϕ). C.Q.F.D.

Nous allons définir une extension de LTL qui consiste à ajouter des opérateurs temporels à l'aide d'automates finis, en fait à l'aide de langages réguliers de mots finis définis par des grammaires linéaires à droite.

Soit $\mathcal{A} = \langle \Sigma, S, S_0, \rho, F \rangle$ un automate fini dont les lettres de Σ sont linéairement ordonnées, par exemple

avec $a_1 < \ldots < a_k$. L'extension ETL_f de LTL (f pour "mots finis") consiste à considérer les formules de la forme $\mathcal{A}(\phi_1, \ldots, \phi_k)$ pour tous les automates avec la sémantique suivante :

- $\sigma, i \models \mathcal{A}(\phi_1, \ldots, \phi_k) \stackrel{\text{def}}{\Leftrightarrow}$
  - soit $S_0 \cap F \neq \emptyset$ ($\epsilon \in \mathrm{L}(\mathcal{A})$)
  - soit il existe un mot *fini* $b_1 b_2 \ldots b_n \in \mathrm{L}(\mathcal{A})$ tel que pour $0 \leq i' < n$, si $b_{i'+1} = a_j$ alors $\sigma, i + i' \models \phi_j$.

Si $S_0 \cap F \neq \emptyset$, alors $\mathcal{A}(\phi_1, \ldots, \phi_k)$ est équivalent à $\top$.

Exercice 4.1. On note ETL_r la variante de ETL_f qui consiste à ne considérer que les mots infinis. Typiquement, $\sigma, i \models \mathcal{A}(\phi_1, \ldots, \phi_k) \stackrel{\text{def}}{\Leftrightarrow}$ il existe un mot infini $b_1 b_2 \ldots$ accepté par $\mathcal{A}$ tel que pour $i' \geq 0$, si $b_{i'+1} = a_j$ alors $\sigma, i + i' \models \phi_j$. Montrer que $\mathrm{SAT}(\mathrm{ETL}_f) \leq \mathrm{SAT}(\mathrm{ETL}_r)$.

Proposition 4.2. [VW94] ETL_f et ETL_r ont le même pouvoir d'expression que les automates de Büchi, c'est-à-dire, un langage L de mots infinis est reconnu par un automate de Büchi si et seulement si il existe une formule de ETL_f telle que $\mathrm{Modèles}(\phi) = \mathrm{L}$.

Autrement dit, la classe de langages définis par des formules de ETL_f est égale à la classe des langages

- reconnus par des automates de Büchi (proposition 4.2) ;

– définis par des formules de la logique monadique du second-ordre à un successeur (S1S) ;
– ω-réguliers ;
– définis par des formules de l'extension de LTL avec quantification sur les variables propositionnelles ;
– définis avec variante LTL où l'opérateur until est indicé par des langages réguliers de mots finis [HT99] ;
– définis par des formules de l'extension de LTL avec opérateur de point fixe [Var88].

Ainsi ETL_f est une extension puissante de LTL mais l'équivalence d'expressivité des formalismes cités ci-dessus ne présume pas de leur concision.

En effet,
– le problème du vide pour les automates de Büchi est NLOGSPACE-complet ;
– $\mathrm{MC}^{\exists}(\mathrm{ETL}_f)$ et $\mathrm{SAT}(\mathrm{ETL}_f)$ sont PSPACE-complets (voir la preuve plus loin) ;
– la satisfaisabilité pour LTL avec opérateurs de point fixe est PSPACE-complet [Var88] ;
– le problème de la satisfaisabilité pour S1S et pour LTL avec quantification sur les variables propositionnelles sont non-élémentaires (complexité en temps n'est pas une tour d'exponentielles de hauteur fixe).

S1S apparaît comme le langage le plus concis pour décrire les langages de mots infinis ω-réguliers.

Exercice 4.2. Définir une formule ϕ de ETL_f construite sur p telle que Modèles(ϕ) est l'ensemble des structures telles que p est vérifiée dans au moins tous les états pairs.

Exercice 4.3. Définir les opérateurs temporels U et X dans ETL_f à l'aide d'automates.

Dans la suite, par défaut, on suppose que les opérateurs temporels de ETL_f sont seulement ceux définis à partir d'automates finis.

4.2 Extension hors-contexte

Il est facile d'étendre la définition de ETL_f en remplaçant les formules de la forme $\mathcal{A}(\phi_1, \ldots, \phi_n)$ par des formules de la forme $\mathrm{L}(\phi_1, \ldots, \phi_n)$ où L est un langage de mots finis sur un alphabet $\Sigma = \{a_1, \ldots, a_n\}$, langage spécifié dans un formalisme choisi.

Pour une classe de langages $\mathcal{C}$, on note $\mathrm{LTL} + \mathcal{C}$ l'extension de LTL aux formules de la forme $\mathrm{L}(\phi_1, \ldots, \phi_n)$ pour $\mathrm{L} \in \mathcal{C}$.

Évidemment, ETL est précisément LTL + REG où REG est la classe des langages réguliers de mots finis représentés par des automates finis.

Il est naturel de vouloir étudier LTL + HC où HC est la classe des langages hors-contexte représentés par des grammaires hors-contexte.

4.2.1 Indécidabilité de LTL + HC

Comme de nombreux problèmes sont indécidables pour la classe des grammaires hors-contexte, il n'est pas très étonnant d'avoir le résultat suivant.

Proposition 4.3. SAT(LTL + HC) est indécidable.

Preuve. Nous pouvons coder le problème de l´égalité de langage entre grammaires hors-contexte, qui est indécidable, dans SAT(LTL + HC).

Soient G_1 et G_2 deux grammaires HC sur l'alphabet (terminal) $\Sigma = \{a_1, \ldots, a_n\}$.

On note G_1^+ et G_2^+ les grammaires hors-contexte sur l'alphabet (terminal) $\Sigma^+ = \{a_1, \ldots, a_n, a_{n+1}\}$ telles que

$$\mathrm{L}(G_1^+) = \mathrm{L}(G_1) \cdot \{a_{n+1}\} \text{ et } \mathrm{L}(G_2^+) = \mathrm{L}(G_2) \cdot \{a_{n+1}\}.$$

La lettre a_{n+1} est un marqueur de fin de mot. On a $\mathrm{L}(G_1) = \mathrm{L}(G_2)$ ssi $\mathrm{L}(G_1^+) = \mathrm{L}(G_2^+)$.

Nous allons construire une formule ϕ_{G_1,G_2} de LTL + HC telle que ϕ_{G_1,G_2} est satisfaisable dans LTL + HC ssi $\mathrm{L}(G_1) \neq \mathrm{L}(G_2)$.

La formule ϕ_{G_1,G_2} est construite sur les variables propositionnelles $\mathrm{p}_1, \ldots, \mathrm{p}_{n+1}$ et nous considérons les structures pour lesquelles exactement un élément de $\mathrm{p}_1, \ldots, \mathrm{p}_{n+1}$ est vérifiée dans chaque état et p_{n+1} est vérifiée dans un unique état.

Soit UNI la formule suivante exprimant cette propriété :

$$\mathrm{UNI} \stackrel{\text{def}}{=} \mathrm{G}(\bigvee_{1\leq i\leq n+1} \mathrm{p}_i) \wedge \mathrm{G}(\bigwedge_{1\leq i\leq n+1} (\mathrm{p}_i \Rightarrow \bigwedge_{1\leq j\neq i\leq n+1} \neg \mathrm{p}_j) \wedge$$
$$((\mathrm{p}_{n+1} \wedge \mathrm{XG}\neg \mathrm{p}_{n+1}) \vee \neg \mathrm{p}_{n+1} \mathrm{U} (\mathrm{p}_{n+1} \wedge \mathrm{XG}\neg \mathrm{p}_{n+1}))$$

On peut montrer l'équivalence des propositions suivantes :

- $\mathrm{L}(G_1^+) \neq \mathrm{L}(G_2^+)$;
- $\mathrm{UNI} \wedge \neg(\mathrm{L}(G_1^+)(\mathrm{p}_1, \ldots, \mathrm{p}_{n+1}) \Leftrightarrow \mathrm{L}(G_2^+)(\mathrm{p}_1, \ldots, \mathrm{p}_{n+1}))$ est satisfaisable.

En effet, si $\mathrm{L}(G_1^+) \neq \mathrm{L}(G_2^+)$, disons que $a_{i_1} a_{i_2} \cdots a_{i_l} a_{n+1} \in \mathrm{L}(G_1^+)$ et $a_{i_1} a_{i_2} \cdots a_{i_l} a_{n+1} \notin \mathrm{L}(G_2^+)$.

Sans perte de généralité on suppose $l \geq 1$. Soit σ le modèle

$$\{\mathrm{p}_{i_1}\} \cdot \{\mathrm{p}_{i_2}\} \cdots \{\mathrm{p}_{i_l}\} \cdot \{\mathrm{p}_{n+1}\} \cdot \{\mathrm{p}_1\}^{\omega}.$$

On a bien

- $\sigma \models \mathrm{UNI}$,
- $\sigma \models \mathrm{L}(G_1^+)(\mathrm{p}_1, \ldots, \mathrm{p}_{n+1})$ et,
- $\sigma \not\models \mathrm{L}(G_2^+)(\mathrm{p}_1, \ldots, \mathrm{p}_{n+1})$ car le seul mot fini finissant par $\{\mathrm{p}_{n+1}\}$ dans σ est $\{\mathrm{p}_{i_1}\} \cdot \{\mathrm{p}_{i_2}\} \cdots \{\mathrm{p}_{i_l}\} \cdot \{\mathrm{p}_{n+1}\}$ et $a_{i_1} a_{i_2} \cdots a_{i_l} a_{n+1} \notin \mathrm{L}(G_2^+)$.

Supposons à présent que $\sigma, 0 \models \mathrm{UNI} \wedge \neg(\mathrm{L}(G_1^+)(\mathrm{p}_1, \ldots, \mathrm{p}_{n+1}) \Leftrightarrow \mathrm{L}(G_2^+)(\mathrm{p}_1, \ldots, \mathrm{p}_{n+1}))$ pour un modèle σ.

Disons par exemple que $\sigma \models \mathrm{L}(G_1^+)(\mathrm{p}_1, \ldots, \mathrm{p}_{n+1})$ et $\sigma \not\models \mathrm{L}(G_2^+)(\mathrm{p}_1, \ldots, \mathrm{p}_{n+1})$. Un simple raisonnement par l'absurde permet d'établir que $\mathrm{L}(G_1^+) \neq \mathrm{L}(G_2^+)$. C.Q.F.D.

La proposition 4.3 est intéressante mais finalement elle repose sur le fait que LTL + HC peut assez facilement exprimer l'équivalence entre grammaires hors-contexte et pas tellement sur les relations entre les opérateurs temporels usuels X et G et les formules de la forme $\mathrm{L}(\phi_1, \ldots, \phi_n)$.

Par exemple, existe-t-il un langage hors-contexte L tel que SAT(LTL + {L}) soit aussi indécidable ?

Évidemment, la preuve de la proposition 4.3 ne serait pas réutilisable de même que toute preuve qui réduise un problème indécidable pour une classe infinie de grammaires hors-contexte.

4.2.2 Indécidabilité de LTL + $\{a_1^n \cdot a_2 \cdot a_1^{n-1} \cdot a_3 : n \geq 1\}$

En fait, on peut répondre par l'affirmative à la question précédente. Soit L_0 le langage $\{a_1^n \cdot a_2 \cdot a_1^{n-1} \cdot a_3 : n \geq 1\}$.

Exercice 4.4. Définir une grammaire linéaire pour L_0 (grammaire hors-contexte dont les parties droites des règles contiennent au plus un symbole non terminal).

On note L_1 le langage $\{a_1^n \cdot a_2 \cdot a_1^n \cdot a_3 : n \geq 0\}$.

Exercice 4.5. Vérifier la validité dans LTL + $\{\mathrm{L}_0, \mathrm{L}_1\}$ de la formule suivante :

$$\mathrm{L}_1(\mathrm{p}, \mathrm{q}, \mathrm{r}) \Leftrightarrow (\mathrm{q} \wedge \mathrm{Xr}) \vee L_0(\mathrm{p}, \mathrm{q}, \mathrm{p} \wedge \mathrm{Xr}).$$

Conclure que SAT(LTL + $\{\mathrm{L}_0, \mathrm{L}_1\}$) est indécidable ssi SAT(LTL + $\{\mathrm{L}_0\}$) est indécidable.

On peut aussi montrer que l'on peut définir F avec L_1 : Fϕ est équivalent à $\mathrm{L}_1(\top, \phi, \top)$. Ainsi c'est bien l'indécidabilité de SAT(LTL(X) + L_0) qui va être établie (et de $\mathrm{MC}^{\exists}$(LTL(X) + L_0)).

Proposition 4.4. SAT(LTL + $\{\mathrm{L}_0\}$) est indécidable.

Preuve. Nous allons réduire DOMREC à SAT(LTL + $\{\mathrm{L}_0, \mathrm{L}_1\}$). Soit $Dom = \langle C, D, Coul \rangle$ un jeu de dominos avec $C = \{1, \ldots, n\}$, $D = \{d_1, \ldots, d_m\}$, et $c = 1$ (pour avoir une instance de DOMREC). $Coul$ est de la forme

$$Coul : D \times \{haut, bas, gauche, droite\} \rightarrow \{1, \ldots, n\}.$$

Les variables propositionnelles suivantes sont utilisées :

- in est une variable propositionnelle vraie lorsque l'état code une position de $\mathbb{N}^2$.
  En effet tous les états du modèle ne vont pas correspondre à une position dans $\mathbb{N}^2$;
- pour $1 \leq j \leq m$, on utilise la variable propositionnelle $\boxed{\mathrm{j}}$: "l'état est occupé par le domino d_j" ;
- pour $1 \leq i \leq n$, on utilise les variables $haut_i$, bas_i, $gauche_i$, $droite_i$. Par exemple, la variable propositionnelle $haut_1$ est vraie dans un état correspondant à une position de $\mathbb{N}^2$ ssi la couleur en haut du domino est 1.

Chaque état codant une position de $\mathbb{N}^2$ est occupé par un unique domino :

$$\mathrm{G}(in \Rightarrow \bigvee_{j=1}^{m} (\boxed{\mathrm{j}} \wedge \bigwedge_{j'=1, j' \neq j}^{m} \neg \boxed{\mathrm{j'}}))$$

Les variables propositionnelles pour les couleurs sont compatibles avec la définition des dominos :

$$\mathrm{G}(in \Rightarrow \bigwedge_{j=1}^{m} \boxed{\mathrm{j}} \implies$$

$$\bigwedge_{cot \in \{haut,bas,droite,gauche\}} cot_{Coul(d_j,cot)} \wedge \bigwedge_{1 \leq j' \neq Coul(d_j,cot) \leq n} \neg cot_{j'})$$

On note PAVAGE la conjonction des formules ci-dessus.

Nous allons à présent définir les états du modèle qui correspondent à une position de $\mathbb{N}^2$.

On note SERPENT la conjonction des formules suivantes :

– $in \wedge \mathrm{X}\neg in \wedge \mathrm{XX}in \wedge \mathrm{XXX}in \wedge \mathrm{XXXX}\neg in$;

– $\mathrm{G}(\neg in \Rightarrow \mathrm{XL}_1(in, \neg in, in \wedge \mathrm{X}\neg in))$.
$(\mathrm{L}_1 = \{a_1^n \cdot a_2 \cdot a_1^n \cdot a_3 : n \geq 0\})$.

La seule structure (construite avec seulement in) satisfaisant SERPENT est :

$$\{in\} \cdot \emptyset \cdot \{in\}^2 \cdot \emptyset \cdot \{in\}^3 \cdot \emptyset \cdot \{in\}^4 \ldots$$

Cette séquence fait référence au parcours de la figure 3.

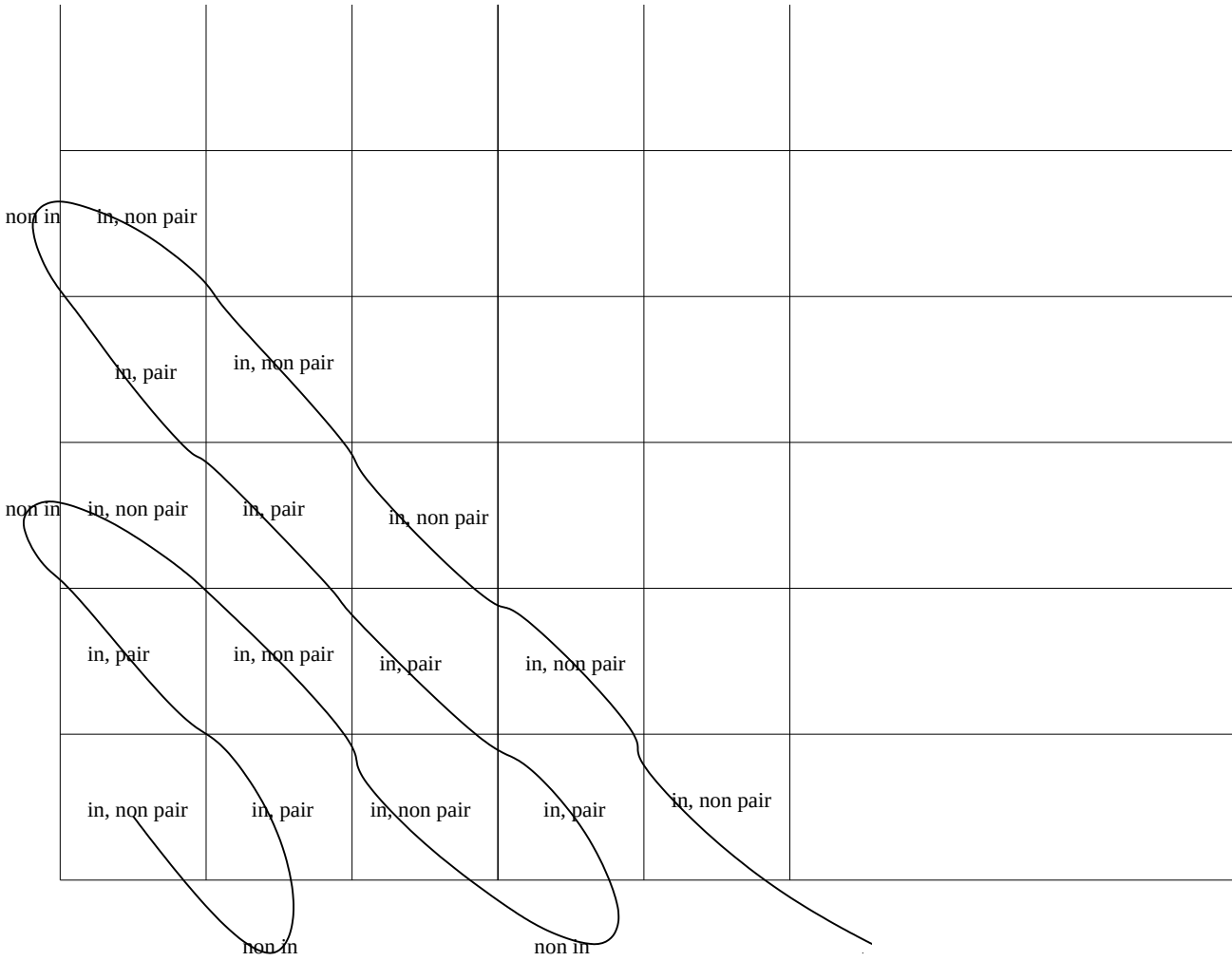


FIG. 3 – Parcours de $\mathbb{N}^2$

Pour chaque état in, on désire se souvenir s'il apparaît dans une séquence de in avec un nombre pair ou impair d'éléments. En effet, ce critère est pertinent pour accéder aux voisins de droite et d'en haut.

On introduit la variable propositionnelle $pair$. On note PARITE la conjonction des formules suivantes :

- $\mathrm{G}(\neg in \Rightarrow \neg pair)$ (valeur par défaut) ;
- $\neg pair$;
- $\mathrm{G}(in \wedge \mathrm{X}in \wedge pair \Rightarrow \mathrm{X}pair)$ ("on reste sur pair") ;
- $\mathrm{G}(in \wedge \mathrm{X}in \wedge \neg pair \Rightarrow \mathrm{X}\neg pair)$ ("on reste sur impair") ;
- $\mathrm{G}(in \wedge \mathrm{X}\neg in \wedge pair \Rightarrow \mathrm{XX}\neg pair)$ ("on passe de pair à impair") ;
- $\mathrm{G}(in \wedge \mathrm{X}\neg in \wedge \neg pair \Rightarrow \mathrm{XX}pair)$ ("on passe d'impair à pair").

Les quatre dernières formules peuvent être remplacées (de façon équivalente) par :

$$\mathrm{G}((in \wedge \mathrm{X}in) \Rightarrow (pair \Leftrightarrow \mathrm{X}pair) \wedge (in \wedge \mathrm{X}\neg in) \Rightarrow (pair \Leftrightarrow \mathrm{XX}\neg pair))$$

La seule structure (construite avec seulement in et $pair$) satisfaisant SERPENT $\wedge$ PARITE est :

$$\{in\} \cdot \emptyset \cdot \{in, pair\}^2 \cdot \emptyset \cdot \{in\}^3 \cdot \emptyset \cdot \{in, pair\}^4 \ldots$$

Cette structure code le parcours de $\mathbb{N}^2$ décrit dans la figure 3.

La variable propositionnelle *pair* est vérifiée dans les états appartenant à une "partie montante" du parcours.

Ce parcours permet d'accéder aux états adjacents de la façon suivante :

- dans un état $\{in, pair\}$, on accède au voisin de droite avec L_1 ;
- dans un état $\{in, pair\}$, on accède au voisin en haut avec L_0 ;
- dans un état $\{in, \neg pair\}$, on accède au voisin de droite avec L_0 ;
- dans un état $\{in, \neg pair\}$, on accède au voisin en haut avec L_1.

On note CONTRAINTES la conjonction des formules suivantes qui exprime les contraintes de couleur sur les dominos adjacents :

- $\mathrm{G}(in \wedge pair \Rightarrow (\bigwedge_{1 \leq i \leq n} droite_i \Rightarrow \mathrm{L}_1(in, \neg in, gauche_i)))$;
- $\mathrm{G}(in \wedge pair \Rightarrow (\bigwedge_{1 \leq i \leq n} haut_i \Rightarrow \mathrm{L}_0(in, \neg in, bas_i)))$;
- $\mathrm{G}(in \wedge \neg pair \Rightarrow (\bigwedge_{1 \leq i \leq n} droite_i \Rightarrow \mathrm{L}_0(in, \neg in, gauche_i)))$;
- $\mathrm{G}(in \wedge \neg pair \Rightarrow (\bigwedge_{1 \leq i \leq n} haut_i \Rightarrow \mathrm{L}_1(in, \neg in, bas_i)))$.

On note REC la formule qui exprime que la couleur 1 apparaît infiniment souvent :

$$\mathrm{GF}(in \wedge \bigvee_{cot \in \{gauche, droite, haut, bas\}} cot_1).$$

On peut à présent facilement vérifier que Dom pave $\mathbb{N}^2$ en répétant infiniment souvent la couleur 1 ssi

PAVAGE∧SERPENT∧PARITE∧CONTRAINTES∧REC

est satisfaisable dans LTL + $\{\mathrm{L}_0\}$. C.Q.F.D.

SAT(LTL + $\{\mathrm{L}_0\}$) est donc Σ_1^1-difficile et donc n'est pas récursivement énumérable.

La preuve de la proposition 4.4 est inspirée de la preuve d'indécidabilité de la logique propositionnelle dynamique (PDL) augmentée du langage hors-contexte $\{a_1^n \cdot a_2 \cdot a_1^n : n \geq 0\}$ (voir par exemple [HKT00, chapitre 9] pour plus de détails).

Pour la vérification formelle, le résultat suivant est plus significatif :

Corollaire 4.5. $\mathrm{MC}^{\exists}(\mathrm{LTL(X)} + \{\mathrm{L}_0\})$ est indécidable.

Exercice 4.6. Faire la preuve du corollaire 4.5 en reduisant par exemple SAT(LTL(X) + $\{\mathrm{L}_0\}$) à $\mathrm{MC}^{\exists}$(LTL(X) + $\{\mathrm{L}_0\}$) (en temps exponentiel).

Exercice 4.7. Montrer que le problème du model-checking pour le calcul propositionnel augmenté du seul opérateur temporel L_1 est indécidable.

4.3 ETL dans PSPACE

Par contre, en utilisant [VW94] nous allons montrer :

Proposition 4.6. $\mathrm{SAT}(\mathrm{ETL}_f), \mathrm{MC}^{\exists}(\mathrm{ETL}_f) \in$ PSPACE.

La PSPACE-dureté est immédiate car ETL_f étend LTL.

Dans le même ordre de résultat, toute extension de LTL avec un nombre fini d'opérateurs temporels définissables dans MSO est dans PSPACE [GK03].

D' ailleurs Moshe Vardi et Pierre Wolper ont reçu le prix Gödel pour leur article [VW94] (voir `http://www.eatcs.org/Activities/Awards/goedel2000.html`).

4.3.1 Automates sous-mot de parties

Afin d'établir la borne de complexité PSPACE, nous allons introduire la classe des automates sous-mot ainsi que la sous-classe des automates sous-mot de parties (appelés respectivement "subword automaton" et "set-subword automaton" dans [VW94]).

Ces automates peuvent être vus comme des automates de Büchi spécialisés, ils seront particulièrement adaptés au langage ETL_f. Ces automates ont été définis dans [VW94].

Définition 4.1. Un *automate sous-mot* $\mathcal{A}$ est une structure $\langle \Sigma, S, \rho, \xi, F \rangle$ telle que
- Σ est l'alphabet (fini) ;
- S est un ensemble fini d'états ;
- $\rho : S \times \Sigma \to \mathcal{P}(S)$ est la fonction de transition ;
- $\xi : \Sigma \to S$ est la fonction d'étiquetage ;
- $F \subseteq S$ est un ensemble non-vide d'états acceptants.

∇

Différences avec les automates de Büchi :
- pas d'ensemble distingué d'états initiaux ;
- ajout de la fonction d'étiquetage ;
- le mode d'acceptation (cf. ci-dessous).

Un mot infini $\sigma \in \Sigma^\omega$ est accepté par $\mathcal{A}$ si

(étiquetage) pour $i \in \mathbb{N}$, $\xi(\sigma_{i+1}) \in \rho(\xi(\sigma_i), \sigma_i)$;

(sous-mot) pour $i \in \mathbb{N}$, il existe $j \geq i$ et une fonction $f : [i, j] \to S$ tels que
- $f(i) = \xi(\sigma_i)$,
- $f(j) \in F$,
- $f(k+1) \in \rho(f(k), \sigma_k)$ pour $i \leq k < j$.

La condition (étiquetage) implique que l'acceptation de σ par $\mathcal{A}$ s'effectue avec un unique calcul réussi

$$\xi(\sigma_0) \xrightarrow{\sigma_0} \xi(\sigma_1) \xrightarrow{\sigma_1} \xi(\sigma_2) \dots$$

De plus, la condition (sous-mot) force l'existence d'un mot reconnu par l'automate fini $\langle \Sigma, S, \{\xi(\sigma_i)\}, \rho, F \rangle$ préfixe de $\sigma_i \sigma_{i+1} \sigma_{i+2} \dots$.

Cette condition est en rapport direct avec la définition des opérateurs temporels liés aux automates. En effet, $\sigma, i \models \mathcal{A}(\phi_1, \dots, \phi_n)$ avec $\mathcal{A} = \langle \Sigma, S, S_0, \rho, F \rangle$ et $\Sigma = \{a_1, \dots, a_n\}$ ssi il existe $a_{\alpha_i} \cdot \dots \cdot a_{\alpha_{j-1}} \in \mathrm{L}(\mathcal{A})$ et $f' : [i, j] \to S$ tels que

- $f(i) \in S_0$;
- $f(j) \in F$;
- $f(k+1) \in \rho(f(k), a_{\alpha_k})$ et $\sigma, k \models \phi_{\alpha_k}$ pour $i \leq k < j$.

Nous commençons par établir un lemme technique.

Lemme 4.7. Soit $\mathcal{A} = \langle \Sigma, S, \rho, \xi, F \rangle$ un automate sous-mot. $\mathcal{A}$ accepte un mot $\sigma : \mathbb{N} \to \Sigma$ ssi

- pour $i \in \mathbb{N}$, $\xi(\sigma_{i+1}) \in \rho(\xi(\sigma_i), \sigma_i)$;
- pour $i \in \mathbb{N}$, il existe $j > i$ et une fonction $f : [i, j] \to S$ tels que
  - $f(i) = \xi(\sigma_i)$,
  - $f(j) \in F$,

– $f(k+1) \in \rho(f(k), \sigma_k)$ pour $i \leq k < j$,

Par rapport à la définition d'acceptation initiale, on impose que l'indice j soit strictement supérieur à l'indice i.

Preuve. Supposons que $\sigma : \mathbb{N} \to \Sigma$ soit accepté par $\mathcal{A}$ et soit $i \in \mathbb{N}$. Il existe $j \geq i$ et $f_i : [i, j] \to S$ tels que
– $f_i(i) = \xi(\sigma_i)$,
– $f_i(j) \in F$,
– $f_i(k+1) \in \rho(f_i(k), \sigma_k)$ pour $i \leq k < j$.

Dans l'hypothèse où $i = j$ (seul cas intéressant pour la preuve), nous savons qu'il existe $k \geq i+1$ et $f_{i+1} : [i+1, k] \to S$ tels que
– $f_{i+1}(i+1) = \xi(\sigma_{i+1})$,
– $f_{i+1}(k) \in F$,
– $f_{i+1}(k'+1) \in \rho(f_{i+1}(k'), \sigma_{k'})$ pour $i+1 \leq k' < k$.
Comme la condition d'étiquetage est vérifiée (en particulier $\xi(\sigma_{i+1}) \in \rho(\xi(\sigma_i), \sigma_i)$), la fonction $g_i : [i, k] \to S$ telle que
– $g_i(i) = f_i(i)$ et,
– $g_i(k') = f_{i+1}(k')$ pour $i+1 \leq k' \leq k$,
a les bonnes propriétés. C.Q.F.D.

Lemme 4.8. Chaque automate sous-mot $\mathcal{A}$ avec m états est équivalent à un automate de Büchi $\mathcal{A}'$ avec $\mathcal{O}(m^2)$ états, c'est-à-dire $\mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{A}')$.

Preuve. Soit $\mathcal{A} = \langle \Sigma, S, \rho, \xi, F \rangle$ un automate sous-mot. Nous définissons deux nouvelles fonctions de transition $\rho_1, \rho_2 : S \times \Sigma \to \mathcal{P}(S)$:

- $\rho_1(s, a) = \rho(s, a)$ si $s = \xi(a)$, sinon $\rho_1(s, a) = \emptyset$ (pour l'étiquetage).
- $\rho_2(s, a) = \rho(\xi(a), a)$ si $s \in F$, sinon $\rho_2(s, a) = \rho(s, a)$ (pour le sous-mot).

Avec ces fonctions de transition nous définissons les automates de Büchi $\mathcal{A}_1 = \langle \Sigma, S, S, \rho_1, S \rangle$ et $\mathcal{A}_2 = \langle \Sigma, S, S, \rho_2, F \rangle$. Nous allons montrer que $\mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{A}_1) \cap \mathrm{L}(\mathcal{A}_2)$.

L'automate de Büchi $\mathcal{A}'$ tel que $\mathrm{L}(\mathcal{A}') = \mathrm{L}(\mathcal{A}_1) \cap \mathrm{L}(\mathcal{A}_2)$ a en effet $\mathcal{O}(|S|^2)$ états et peut être construit avec un espace mémoire logarithmique en $|\mathcal{A}|$.

Soit $\sigma : \mathbb{N} \to \Sigma$ un mot accepté par $\mathcal{A}_1$ et $\mathcal{A}_2$.

Pour $j \in \{1, 2\}$, $s_{j,0} s_{j,1} s_{j,2} \cdots$ est un calcul réussi de $\mathcal{A}_j$ sur σ.

Vérifions d'abord que σ satisfait la condition d'étiquetage.

Pour $i \in \mathbb{N}$, $\rho_1(s_{1,i}, \sigma_i)$ est non-vide et donc $s_{1,i} = \xi(\sigma_i)$. Par conséquent pour $i \in \mathbb{N}$, $\xi(\sigma_{i+1}) = s_{1,i+1} \in \rho_1(s_{1,i}, \sigma_i) = \rho(\xi(\sigma_i), \sigma_i)$.

Vérifions à présent la condition du sous-mot. Soit $i \in \mathbb{N}$. Comme $s_{2,0}s_{2,1}s_{2,2}\cdots$ est réussi, il existe $j \geq i$ tel que $s_{2,j} \in F$. De même, il existe $k > j$ tel que $s_{2,k} \in F$ et pour $j < l < k$, $s_{2,l} \notin F$.

Nous allons montrer que $[i, k]$ vérifie la propriété du sous-mot en définissant $f : [i, k] \to S$ de la façon suivante :

$$f(l) = \begin{cases} \xi(\sigma_l) \text{ si } l \in [i, j], \\ s_{2,l} \text{ si } l \in [j+1, k]. \end{cases}$$

On a bien $f(i) = \xi(\sigma_i)$ et $f(k) \in F$. Il reste à vérifier que pour $i \leq l < k$, $f(l+1) \in \rho(f(l), \sigma_l)$.

– pour $i \leq l < j$,

$$f(l+1) = \xi(\sigma_{l+1}) \in \rho(\xi(\sigma_l), \sigma_l) = \rho(f(l), \sigma_l).$$

La condition d'étiquetage est en effet vérifiée.

– pour $l = j$,

$$f(j+1) = s_{2,j+1} \in \rho_2(s_{2,j}, \sigma_j) = \rho(\xi(\sigma_j), \sigma_j) = \rho(f(j), \sigma_j).$$

– pour $j + 1 \leq l < k$,

$$f(l+1) = s_{2,l+1} \in \rho_2(s_{2,l}, \sigma_l) = \rho(s_{2,l}, \sigma_l) = \rho(f(l), \sigma_l).$$

Dans la construction ci-dessus, si on définissait f sur $[i, j]$ seulement, on n'aurait pas la garantie que $\xi(\sigma_j) \in F$.

Montrons maintenant que $\mathrm{L}(\mathcal{A}) \subseteq \mathrm{L}(\mathcal{A}_1) \cap \mathrm{L}(\mathcal{A}_2)$. Soit $\sigma : \mathbb{N} \to \Sigma \in \mathrm{L}(\mathcal{A})$ accepté avec le calcul réussi $s_0 s_1 s_2 \cdots$. Il est évident que $\sigma \in \mathrm{L}(\mathcal{A}_1)$ car σ est accepté par $\mathcal{A}_1$ avec le même calcul réussi. Montrons qu'il existe un calcul $c_2 = s_{2,0} s_{2,1} \cdots$ qui accepte σ avec $\mathcal{A}_2$.

Nous construisons une suite (infinie) strictement croissante $j_1, j_2, \ldots$ d'entiers telle que $s_{2,j_i} \in F$ pour chaque j_i.

On commence par $j_1 = 0$, $s_{2,0} = s$ pour un état arbitraire de F.

Supposons que le calcul c_2 est défini sur l'intervalle $[0, j_n]$. Par hypothèse d'induction, $s_{2,j_n} \in F$.

D'après le lemme 4.7, il existe $j_{n+1} > j_n$ et une fonction $f : [j_n, j_{n+1}] \to S$ telle que
- $f(j_n) = \xi(\sigma_{j_n})$,
- $f(j_{n+1}) \in F$, et
- pour $j_n \leq l < j_{n+1}$, $f(l+1) = \rho(f(l), \sigma_l)$.

Sans perte de généralité, on peut supposer que pour $j_n < l < j_{n+1}$, $f(l) \notin F$.

Pour $j_n < l \leq j_{n+1}$, nous posons $s_{2,l} = f(l)$.

Nous devons montrer que pour $j_n \leq l < j_{n+1}$, $s_{2,l+1} \in \rho_2(s_{2,l}, \sigma_l)$.

- Si $l = j_n$, alors $s_{2,l} \in F$. Par conséquent, $s_{2,l+1} = f(l+1) \in \rho(\xi(\sigma_l), \sigma_l) = \rho_2(s_{2,l}, \sigma_l)$.
- Pour $j_n < l < j_{n+1}$, $s_{2,l} \notin F$ et donc $s_{2,l+1} = f(l+1) = \rho(f(l), \sigma_l) = \rho_2(s_{2,l}, \sigma_l)$.

De cette façon nous pouvons compléter la définition du calcul réussi de c_2. C.Q.F.D.

La classe des automates sous-mot de parties (introduite ci-dessous) est une classe spécialisée d'automates sous-mot où

1. l'ensemble des états et l'alphabet forment un même ensemble, en l'occurrence un ensemble d'ensembles de formules ;
2. la condition d'acceptation est liée aux formules et pas seulement aux ensembles de formules.

Définition 4.2. Un *automate sous-mot de parties* $\mathcal{A}$ est une structure $\langle \Psi, \rho \rangle$ telle que

– Ψ est un ensemble fini de symboles (des formules logiques dans la suite). $\mathcal{P}(\Psi)$ sera à la fois l'alphabet Σ et l'ensemble S des états de l'automate sous-mot sous-jacent. Les éléments de $\mathcal{P}(\Psi)$ sont notés alternativement par s ou a selon qu'ils sont considérés comme des états de S ou des lettres de Σ.
– $\rho : S \times \Sigma \to \mathcal{P}(S)$ est telle que

1. $\rho(s, a) \neq \emptyset$ ssi $s \subseteq a$. Les formules d'un état s sont les formules que l'automate essaie de vérifier. Une lettre a est un ensemble de formules supposées vraies.
2. $\emptyset \in \rho(\emptyset, a)$.
3. si $s \subseteq s'$, $s_1 \subseteq s'_1$, et $s_1 \in \rho(s', a)$ alors $s'_1 \in \rho(s, a)$. Une transition de l'automate est une contrainte suffisante sur les formules de l'état suivant pour vérifier les formules de l'état courant.
4. si $s'_1 \in \rho(s_1, a)$ et $s'_2 \in \rho(s_2, a)$ alors $s'_1 \cup s'_2 \in \rho(s_1 \cup s_2, a)$. Il n'y a pas d'interaction entre les formules que l'automate essaie de vérifier.

∇

Les conditions 2. et 3. sont relatives à la monotonie de ρ.

Un mot $\sigma : \mathbb{N} \to \Sigma$ ($\Sigma = S = \mathcal{P}(S)$) est accepté par $\mathcal{A}$ si pour $i \in \mathbb{N}$ et pour $\phi \in \sigma_i$, il existe un intervalle $[i, j]$ et une fonction $f : [i, j] \to S$ tels que
- $f(i) = \{\phi\}$;
- $f(j) = \emptyset$;
- pour $i \leq k < j$, $f(k+1) \in \rho(f(k), \sigma_k)$.

On peut remarquer que la condition d'acceptation est adaptée à la sémantique de $\mathcal{A}(\psi_1, \ldots, \psi_n)$. Le traitement est effectué formule par formule. Le lemme 4.9 ci-dessous permettra de passer facilement des automates sous-mot de parties aux automates sous-mot.

Lemme 4.9. Soient $\mathcal{A} = \langle \Psi, \rho \rangle$ un automate sous-mot de parties, $\sigma : \mathbb{N} \to \mathcal{P}(\Psi)$ accepté par $\mathcal{A}$, et $i \in \mathbb{N}$. Les propositions suivantes sont équivalentes :

1. il existe $j \geq i$ et $f : [i, j] \to \mathcal{P}(\Psi)$ tels que
   - $f(i) = \sigma(i)$;
   - $f(j) = \emptyset$;
   - $f(k+1) \in \rho(f(k), \sigma_k)$ pour $i \leq k < j$.
2. pour $\phi \in \sigma_i$, il existe $j_\phi \geq i$ et $f_\phi : [i, j_\phi] \to \mathcal{P}(\Psi)$ tels que
   - $f_\phi(i) = \{\phi\}$;
   - $f_\phi(j) = \emptyset$;
   - $f_\phi(k+1) \in \rho(f_\phi(k), \sigma_k)$ pour $i \leq k < j_\phi$.

La Condition 2. reprend les conditions d'acceptation des automates sous-mot de parties. La Condition 1. est celle pour les automates sous-mot.

Preuve. 1. implique 2.

Soit $\phi \in \sigma_i$. Prenons $j_\phi = j$ et définissons f_ϕ par :
– $f_\phi(i) = \{\phi\}$;
– $f_\phi(k) = f(k)$ pour $i < k \leq j_\phi$.

f_ϕ et f ne diffère que pour la valeur i.

Nous devons vérifier que pour $i \leq k < j_\phi$, $f_\phi(k+1) \in \rho(f_\phi(k), \sigma_k)$.

Pour $i < k < j_\phi$, cela est vérifié par définition de f_ϕ, j_ϕ à partir de f, j.

Comme $f_\phi(i) \subseteq f(i)$ et $f(i+1) \in \rho(f(i), \sigma_i)$, par monotonie (Condition 3.),

$$f(i+1) = f_\phi(i+1) \in \rho(f_\phi(i), \sigma_i).$$

2. implique 1.

On distinguera le cas $\sigma_i = 0$ qui demande un traite-

ment particulier.

Si $\sigma_i = \emptyset$ alors $j = i$ sinon $j = max_{\phi \in \sigma_i}\{j_\phi\}$.

Pour $\phi \in \sigma_i$, on étend f_ϕ à $[i, j]$ avec $f_\phi(k) = \emptyset$ pour $j_\phi < k \leq j$.

Si $\sigma_i = \emptyset$ alors $f(i) = \emptyset$, sinon pour $k \in [i, j]$,

$$f(k) = \bigcup_{\phi \in \sigma_i} f_\phi(k).$$

Comme $j \geq j_\phi$, pour $\phi \in \sigma_i$, $f(j) = \emptyset$.

De même $f(i) = \sigma_i$.

On a pour $i \leq k < j$, $f_\phi(k+1) \in \rho(f_\phi(k), \sigma_k)$. La Condition 2. garantit que $\emptyset \in \rho(\emptyset, \sigma_k)$.

Par la condition 4., on obtient

$$\bigcup_{\phi \in \sigma_i} f_\phi(k+1) \in \rho(\bigcup_{\phi \in \sigma_i} f_\phi(k), \sigma_k).$$

C'est-à-dire, $f(k+1) \in \rho(f(k), \sigma_k)$ pour $i \leq k < j$. C.Q.F.D.

La propriété fondamentale des automates sous-mot de parties est la suivante :

Proposition 4.10. L'automate sous-mot de parties $\mathcal{A} = \langle \Psi, \rho \rangle$ est équivalent à l'automate sous-mot

$$\mathcal{A}' = \langle \mathcal{P}(\Psi), \mathcal{P}(\Psi), \rho, \xi, \{\emptyset\} \rangle$$

où ξ est l'identité, c'est-à-dire $\mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{A}')$.

Preuve. D'après le lemme 4.9, si $\sigma \in \mathrm{L}(\mathcal{A}')$ alors $\sigma \in \mathrm{L}(\mathcal{A})$.

De même d'après le lemme 4.9, si $\sigma \in \mathrm{L}(\mathcal{A})$ alors la condition (sous-mot) de $\mathcal{A}'$ est satisfaite. Il reste à montrer que ξ vérifie la condition (étiquetage).

Nous devons prouver que pour $i \in \mathbb{N}$, $\sigma_{i+1} \in \rho(\sigma_i, \sigma_i)$ (ξ est l'identité).

Soit $i \in \mathbb{N}$. Comme $\sigma \in \mathrm{L}(\mathcal{A})$, il existe $j \geq i$ et $f : [i, j] \to \mathcal{P}(\Psi)$ tels que
- $f(i) = \sigma(i)$;
- $f(j) = \emptyset$;
- $f(k+1) \in \rho(f(k), \sigma_k)$ pour $i \leq k < j$.

Si $j = i$ alors $\sigma_i = \emptyset$ et par monotonie $\sigma_{i+1} \in \rho(\sigma_i, \sigma_i)$.

Sinon $f(i+1) \in \rho(\sigma_i, \sigma_i)$. De même si $f(i+1) = \emptyset$ alors $f(i+1) \subseteq \sigma_{i+1}$ et donc par monotonie $\sigma_{i+1} \in \rho(\sigma_i, \sigma_i)$.

Cela comprend le cas $j = i + 1$.

Sinon ($j \geq i + 2$), $\rho(f(i+1), \sigma_{i+1}) \neq \emptyset$ car $f(i+2) \in \rho(f(i+1), \sigma_{i+1})$.

Par satisfaction de la Condition 1., $f(i+1) \subseteq \sigma_{i+1}$.

Comme par ailleurs $f(i+1) \in \rho(\sigma_i, \sigma_i)$. Par monotonie, $\sigma_{i+1} \in \rho(\sigma_i, \sigma_i)$. C.Q.F.D.

4.3.2 Traduction de formules de ETL en automates

Nous allons construire à présent un automate de Büchi qui reconnaisse les modèles de $\phi \in \mathrm{ETL}_f$ en utilisant les automates sous-mot de parties.

Pour un automate $\mathcal{A} = \langle \Sigma, S, S_0, \rho, F \rangle$ utilisé dans les formules de ETL_f, $\mathcal{A}_s$ dénote l'automate $\langle \Sigma, S, \{s\}, \rho, F \rangle$ pour $s \in S$.

Définition 4.3. La fermeture/clôture de la formule ϕ de ETL_f, notée $cl(\phi)$, est le plus petit ensemble (pour l'inclusion ensembliste) vérifiant :
- $\phi \in cl(\phi)$;
- si $\psi_1 \wedge \psi_2 \in cl(\phi)$ alors $\psi_1, \psi_2 \in cl(\phi)$;

– si $\psi \in cl(\phi)$ alors $\neg\psi \in cl(\phi)$ ($\neg\neg\varphi$ est identifiée avec φ) ;
– si $\mathcal{A}(\psi_1, \ldots, \psi_n) \in cl(\phi)$ alors $\psi_1, \ldots, \psi_n \in cl(\phi)$;
– si $\mathcal{A}(\psi_1, \ldots, \psi_n) \in cl(\phi)$ alors $\mathcal{A}_s(\psi_1, \ldots, \psi_n) \in cl(\phi)$ pour $s \in S$.

∇

Cette notion de clôture étend naturellement celle pour LTL.

Lemme 4.11. Le cardinal de $cl(\phi)$ est polynomial en $|\phi|$.

Soit $\mathcal{A}(\psi_1, \ldots, \psi_n) \in cl(\phi)$ avec $\mathcal{A} = \langle \Sigma, S, S_0, \rho, F \rangle$, $\Sigma = \{a_1, \ldots, a_n\}$ et $a_1 < \cdots < a_n$.

On définit la fonction $suiv_{\mathcal{A}} : \mathcal{P}(cl(\phi)) \to \mathcal{P}(S)$ de la façon suivante :

$$suiv_{\mathcal{A}}(X) = \{s \in \rho(s_0, a_l) : s_0 \in S_0, a_l \in \Sigma, \psi_l \in X\}.$$

Lire la lettre a_l sur $\mathcal{A}$ nécessite la satisfaction de la formule ψ_l. $suiv_{\mathcal{A}}(X)$ dépend en fait de $\mathcal{A}(\psi_1, \ldots, \psi_n)$ et non seulement de $\mathcal{A}$.

Une séquence de Hintikka (Jaakko Hintikka logicien scandinave) pour ϕ est une séquence $\Pi : \mathbb{N} \to \mathcal{P}(cl(\phi))$ vérifiant les conditions ci-dessous ($i \in \mathbb{N}$) :

1. $\phi \in \Pi_0$;
2. $\psi \in \Pi_i$ ssi $\neg\psi \notin \Pi_i$;
3. $\psi \wedge \psi' \in \Pi_i$ ssi $\psi \in \Pi_i$ et $\psi' \in \Pi_i$;
4. si $\mathcal{A}(\psi_1, \ldots, \psi_n) \in \Pi_i$ alors soit $S_0 \cap F \neq \emptyset$ soit il existe $j > i$ et une fonction $f : [i, j] \to \mathcal{P}(cl(\phi))$ tels que
   - $f(k) \subseteq \Pi_k$ pour $i \leq k \leq j$;
   - $\mathcal{A}_{s_0}(\psi_1, \ldots, \psi_n) \in f(i)$ pour un $s_0 \in S_0$;
   - $f(j) = \emptyset$;
   - pour $i \leq k < j$, si $\mathcal{A}_s(\psi_1, \ldots, \psi_n) \in f(k)$ alors soit $s \in F$ soit il existe $t \in suiv_{\mathcal{A}_s}(\Pi_k)$ tel que $\mathcal{A}_t(\psi_1, \ldots, \psi_n) \in f(k+1)$;
5. si $\mathcal{A}(\psi_1, \ldots, \psi_n) \in cl(\phi) \setminus \Pi_i$ alors $S_0 \cap F = \emptyset$ et $\mathcal{A}_s(\psi_1, \ldots, \psi_n) \in cl(\phi) \setminus \Pi_{i+1}$ pour $s \in suiv_{\mathcal{A}}(\Pi_i)$.

La proposition ci-dessous énonce que les séquences de Hintikka sont une bonne abstraction des modèles de ϕ.

Proposition 4.12. ϕ a un modèle ssi ϕ a une séquence de Hintikka.

Preuve. Soient ϕ une formule de ETL_f et $\sigma : \mathbb{N} \to \mathcal{P}(\mathrm{PROP})$ un modèle pour ϕ, c'est-à-dire $\sigma, 0 \models \phi$.

Soit $\Pi : \mathbb{N} \to \mathcal{P}(cl(\phi))$ la séquence définie par

$$\Pi(i) = \{\psi \in cl(\phi) : \sigma, i \models \psi\}.$$

En utilisant la sémantique de ETL_f, on peut facilement montrer que Π est une séquence de Hintikka.

Soit ϕ une formule de ETL_f et $\Pi : \mathbb{N} \to \mathcal{P}(cl(\phi))$ une séquence de Hintikka pour ϕ. Soit $\sigma : \mathbb{N} \to \mathcal{P}(\mathrm{PROP})$ la structure définie par $\sigma(i) = \Pi(i) \cap \mathrm{PROP}$. Nous allons établir que $\sigma, 0 \models \phi$ en montrant par induction sur la taille de ψ que pour $\psi \in cl(\phi)$, $i \in \mathbb{N}$,

$$\psi \in \Pi(i) \text{ ssi } \sigma, i \models \psi.$$

Le cas de base avec les variables propositionnelles est évident comme les étapes d'induction avec les opérateurs Booléens $\neg$ et $\wedge$.

Supposons que $\mathcal{A}(\psi_1, \ldots, \psi_n) \in \Pi_i$. Si $S_0 \cap F \neq \emptyset$ alors $\sigma, i \models \mathcal{A}(\psi_1, \ldots, \psi_n)$. Sinon d'après la condition 4. de Hintikka et d'après l'hypothèse d'induction, il existe $a_{i_1} \cdots a_{i_l} \in \mathrm{L}(\mathcal{A})$ tel que pour $0 \leq j < l$, $\sigma, i+j \models \psi_{i_{j+1}}$, c'est-à-dire $\sigma, i \models \mathcal{A}(\psi_1, \ldots, \psi_n)$.

Supposons à présent que $\mathcal{A}(\psi_1, \ldots, \psi_n) \notin \Pi_i$. En particulier $S_0 \cap F = \emptyset$. Raisonnons par l'absurde en supposons de plus que $\sigma, i \models \mathcal{A}(\psi_1, \ldots, \psi_n)$. Il existe donc un calcul réussi $s_0 \xrightarrow{a_{i_1}} s_1 \xrightarrow{a_{i_2}} \cdots \xrightarrow{a_{i_l}} s_l$ $(l \geq 1)$ tel que pour $0 \leq j < l$, $\sigma, i+j \models \psi_{i_{j+1}}$. D'après la condition 5. de Hintikka et l'hypothèse d'induction $\mathcal{A}_{s_j}(\psi_1, \ldots, \psi_n) \notin \Pi_{i+j}$

pour $0 \leq j \leq l$. En particulier, $\mathcal{A}_{s_l}(\psi_1, \ldots, \psi_n) \notin \Pi_{i+l}$ mais comme $s_l \in F$, la condition 5. de Hintikka n'est pas vérifiée en Π_{i+l}, ce qui conduit à une contradiction. C.Q.F.D.

Nous allons construire un automate de Büchi sur l'alphabet $\mathcal{P}(cl(\phi))$ qui accepte précisément les séquences de Hintikka de ϕ.

En fait, un premier automate de Büchi $\mathcal{A}_L$ vérifie les contraintes locales (conditions 1.-3.,5.) tandis qu'un second automate sous-mot de parties $\mathcal{A}_E$ vérifie la contrainte globale (condition 4.).

L'automate de Büchi

$$\mathcal{A}_L = \langle \mathcal{P}(cl(\phi)), \mathcal{P}(cl(\phi)), S_\phi, \rho_L, \mathcal{P}(cl(\phi)) \rangle$$

est le suivant :

- $S_\phi = \{s \in \mathcal{P}(cl(\phi)) : \phi \in s\}$;
- $s' \in \rho_L(s, a)$ ssi $a = s$ et
  - $\psi \in s$ ssi $\neg\psi \notin s$;
  - $\psi_1 \wedge \psi_2 \in s$ ssi $\psi_1, \psi_2 \in s$;
  - si $\neg\mathcal{A}(\psi_1, \ldots, \psi_n) \in s$ alors $S_0 \cap F = \emptyset$ et pour $s'' \in suiv_{\mathcal{A}}(a)$, $\neg\mathcal{A}_{s''}(\psi_1, \ldots, \psi_n) \in s'$.

L'automate sous-mot de parties $\mathcal{A}_E$ est la structure

$\langle cl(\phi), \rho_E \rangle$ telle que $s' \in \rho_E(s, a)$ ssi les conditions suivantes sont vérifiées :

1. $s \subseteq a$,
2. si $\mathcal{A}(\psi_1, \ldots, \psi_n) \in s$ alors soit $S_0 \cap F \neq \emptyset$ soit il existe $s'' \in suiv_{\mathcal{A}}(a)$ tel que $\mathcal{A}_{s''}(\psi_1, \ldots, \psi_n) \in s'$.

Proposition 4.13. Soit ϕ une formule de ETL_f et $\Pi : \mathbb{N} \rightarrow \mathcal{P}(cl(\phi))$. Π est une séquence de Hintikka pour ϕ ssi $\Pi \in \mathrm{L}(\mathcal{A}_L) \cap \mathrm{L}(\mathcal{A}_E)$.

$\mathcal{A}_E$ accepte exactement les séquences qui vérifient la condition 4. de Hintikka.

En utilisant le lemme 4.8 et la proposition 4.10, on peut construire un automate de Büchi $\mathcal{A}_\phi^{\mathrm{HINT}}$ tel que $\mathrm{L}(\mathcal{A}_\phi^{\mathrm{HINT}}) = \mathrm{L}(\mathcal{A}_L) \cap \mathrm{L}(\mathcal{A}_E)$ et qui contienne $2^{\mathcal{O}(|\phi|)}$ états.

Or, $\mathrm{L}(\mathcal{A}_\phi^{\mathrm{HINT}}) \neq \emptyset$ ssi ϕ est satisfaisable.

Cependant ce que nous avons besoin est la projection des éléments de $\mathrm{L}(\mathcal{A}_L) \cap \mathrm{L}(\mathcal{A}_E)$ sur $\mathcal{P}(\mathrm{PROP})$, ce qui peut être facilement obtenu.

Proposition 4.14. Étant donnée une formule ϕ de ETL_f construite sur les variables propositionnelles de PROP,

nous pouvons construire un automate de Büchi $\mathcal{A}_\phi$ sur l'alphabet $\mathcal{P}(\text{PROP})$ dont la taille est en $2^{\mathcal{O}(|\phi|)}$ et qui accepte précisément les modèles de ϕ.

Vérifier si ϕ est satisfaisable revient à tester le vide du langage $\text{L}(\mathcal{A}_\phi)/\text{L}(\mathcal{A}_\phi^{\text{HINT}})$ ce qui peut être fait en espace polynomial en $|\phi|$ (algorithme à la volée). En effet, vérifier ρ_E et ρ_L peut se calculer avec un espace mémoire polynomial en $|\phi|$.

Proposition 4.15. $\text{MC}^{\exists}(\text{ETL}_f)$ et $\text{SAT}(\text{ETL}_f)$ sont PSPACE-complets.

5 LTL sur des domaines concrets

Nous allons étudier des variantes de LTL où les variables propositionnelles sont remplacées par des termes constraints interprétés sur un domain concret, par exemple $\langle \mathbb{R}, <, = \rangle$. Il s'agit de raffiner la nature des formules atomiques et de prendre en compte des variables interprétées dans un domaine privilégié (entiers, réels, chaînes de caractères, etc.).

5.1 Domaines concrets

Un domaine concret $\mathcal{D} = \langle D, R_1, \ldots, R_n \rangle$ est une structure (relationnelle) où D est un ensemble non-vide, $n \geq 1$, et chaque R_i est une relation incluse dans D^{a_i}. a_i est l'arité de R_i.

On se restreint ici au cas sans constante et avec un nombre fini de relations. On peut cependant coder un nombre fini de constantes en imposant que R_i soit un singleton. Les cas intéressants sont lorsque $R_i \neq \emptyset$ et $R_i \neq D^{a_i}$ ($\mathcal{D}$ non trivial).

$\mathcal{D}$ est en fait un domaine d'interprétation privilégié selon la nature des objets : compteurs ($\mathbb{Z}$, $\mathbb{N}$), horloges ($\mathbb{R}$), chaînes (Σ^*), etc..

A chaque relation R_i on associe un symbole de prédicat R_i d'arité a_i. A partir de ces symboles, des termes contraints interprétés sur $\mathcal{D}$ vont être construits.

Exemples de domaines concrets :
- $\langle D, =, < \rangle$ pour $D \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$.
- $\langle \mathbb{N}, =, succ \rangle$ avec $succ = \{\langle n, n+1 \rangle : n \in \mathbb{N}\}$.
- $\langle \mathbb{R}^n, =, <_{lex} \rangle$ ($<_{lex}$ ordre lexicographique).

– $\langle \Sigma^*, =, \subseteq \rangle$ ($\subseteq$ relation sous-mot ou préfixe).

5.2 Définition de la logique CLTL(D)

CLTL($\mathcal{D}$) : LTL avec les variables propositionnelles remplacées par des termes constraints issus du domaine concret $\mathcal{D}$.

Exemple 5.1. LTL peut être vu comme un fragment de la logique CLTL($\langle \{0,1\}, =, P_{=1} \rangle$) avec $P_{=1} = \{1\}$:

$$\{\mathrm{p}_2, \mathrm{p}_3\} \cdot \{\mathrm{p}_3\} \cdot \{\mathrm{p}_{\mathbf{1}}, \mathrm{p}_{\mathbf{3}}\} \ldots \models \mathrm{F}(\mathrm{p}_1 \wedge \mathrm{p}_3)$$

$$\begin{array}{llllll} x_1 & 0 & 0 & \mathbf{1} & \ldots & \\ x_2 & 1 & 0 & 0 & \ldots & \models \mathrm{F}(P_{=1}(x_1) \wedge P_{=1}(x_3)) \\ x_3 & 1 & 1 & \mathbf{1} & \ldots & \end{array}$$

La variable propositionnelle p_i est équivalente au terme contraint $P_{=1}(x_i)$ et l'équivalence $\mathrm{p}_i \Leftrightarrow \mathrm{XXp}_j$ correspond à l'égalité $x_i = \mathrm{X}^2 x_j$. ∇

Soit $\mathcal{D} = \langle D, R_1, \ldots, R_n \rangle$ un domaine concret et VAR un ensemble dénombrable de variables.

Les formules atomiques de CLTL($\mathcal{D}$) sont les expressions de forme

$$\mathsf{R}(\mathrm{X}^{n_1} x_1, \ldots, \mathrm{X}^{n_k} x_k),$$

où

- $k \geq 1$ est la dimension de la relation R associée au symbole de prédicat R ;
- $n_1, \ldots, n_k \geq 0$;
- $x_1, \ldots, x_k$ sont des variables de VAR.

$\mathrm{X}^n x$ sera interprété comme la valeur de x dans le n^{ieme} successeur de l'état courant.

Les formules de CLTL($\mathcal{D}$) sont engendrées comme les formules de LTL avec la différence que les formules atomiques de CLTL($\mathcal{D}$) sont des termes contraints. On pourrait aussi définir d'autres versions contraintes de logiques avec des extensions de LTL (em remplaçant par exemple LTL par ETL_f).

Une structure de CLTL($\mathcal{D}$) est une séquence infinie d'assignements de la forme VAR $\rightarrow D$.

Dans de très nombreux cas, on peut se restreindre aux assignements de la forme $V \rightarrow D$ pour un sous-ensemble fini V de VAR. C'est généralement le cas quand on cherche la satisfaisabilité d'une formule donnée (ayant un nombre fini V de variables).

Soit $\sigma : \mathbb{N} \to (V \to D)$ et $i \in \mathbb{N}$.

$$\sigma, i \models \mathsf{R}(\mathrm{X}^{n_1} x_1, \ldots, \mathrm{X}^{n_k} x_k) \stackrel{\text{def}}{\Leftrightarrow}$$

$$(\sigma(i+n_1)(x_1), \ldots, \sigma(i+n_k)(x_k)) \in R.$$

V est un sous-ensemble fini de VAR et $\{x_1, \ldots, x_k\} \subseteq V$.

Les opérateurs Booléens ($\wedge$, $\vee$, $\neg$) et temporels (X, U, F) ont leur sémantique usuelle.

Par exemple, la formule $\mathrm{G}(x = \mathrm{X}x)$ caractérise les modèles où la valeur de la variable x est constante dans tout le modèle.

Les valeurs des variables d'états différents peuvent être comparées.

Un modèle de CLTL($\langle \mathbb{Q}, =, < \rangle$) :

$$\begin{array}{cccccc} x_1 & 0 & \frac{3}{8} & \frac{1}{9} & 3 & \ldots \\ x_2 & \frac{1}{2} & \mathbf{0} & \frac{3}{4} & 2 & \ldots \\ x_3 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \mathbf{1} & \ldots \\ t & 1 & 2 & 3 & 4 & \ldots \end{array} \models \mathrm{F}(x_2 < \mathrm{X}^2 x_3)$$

Exemple 5.2. t variable interprétée dans $\mathbb{R}$ codant le temps :

– $\mathrm{G}((t < \mathrm{X}t) \vee (t = \mathrm{X}t))$ (propriété de monotonie) ;
– $\forall\ x\ \mathrm{F}(t > x)$ (propriété de progrès mais $\forall$ n'est un opérateur du langage)
– $\downarrow_{x=t} \phi(x)$.

∇

Le problème de la satisfaisabilité est défini comme le problème de l'existence d'une structure vérifiant $\sigma, 0 \models \phi$ (étant donné ϕ). Dans la suite, nous allons voir que ce problème est analogue aux problèmes de model-checking (définis dans la suite), comme pour LTL.

La formule $\mathrm{G}(\mathrm{X}x < x)$ est satisfaisable dans $\mathrm{CLTL}(\langle \mathbb{R}, =, < \rangle)$ mais pas dans $\mathrm{CLTL}(\langle \mathbb{N}, =, < \rangle)$.

La satisfaisabilité n'est pas toujours un problème décidable :
– la satisfaisabilité pour $\mathrm{CLTL}(\langle \mathbb{N}, =, \mathit{succ} \rangle)$ est indécidable (voir la section 5.4) ;
– la satisfaisabilité pour $\mathrm{CLTL}(\langle \mathbb{R}, =, < \rangle)$ est un problème PSPACE-complet (voir la section 5.5) ;
– il n'est pas connu si la satisfaisabilité pour $\mathrm{CLTL}(\langle \Sigma^*, =, \subseteq \rangle)$ est décidable.

Exercice 5.1. Soit $\mathcal{D} = \langle D, =, \ldots \rangle$ un domaine concret avec égalité et CLTL$^2(\mathcal{D})$ la restriction de CLTL$(\mathcal{D})$ avec les seules variables x et y. Définir une réduction (en espace logarithmique) du problème de la satisfaisabilité pour CLTL$(\mathcal{D})$ au problème de la satisfaisabilité pour CLTL$^2(\mathcal{D})$.

5.3 Automates à contraintes et model-checking

5.3.1 Model-checking 1

Soit $\mathcal{D} = \langle D, R_1, \ldots, R_n \rangle$ un domaine concret. Une $\mathcal{D}$-structure de Kripke est une structure $\mathcal{M} = \langle W, R, L \rangle$ telle que W est un ensemble non-vide d'états, R est une relation (binaire) de transition, et L est de la forme

$$W \to (V \to D),$$

pour un sous-ensemble fini V de VAR.

Problème du model-checking sur les $\mathcal{D}$-structures de Kripke MC1(CLTL($\mathcal{D}$))

entrée : une $\mathcal{D}$-structure de Kripke $\mathcal{M}$ finie et totale, $s \in W$, et ϕ une formule de CLTL$(\mathcal{D})$;

sortie : 1 s'il existe une chemin σ dans $\mathcal{M}$ commençant en s tel que $\sigma \models \phi$. 0 sinon.

Dans le codage d'une $\mathcal{D}$-structure de Kripke, on suppose qu'un assignement $V \to D$ est codé comme une séquence finie de valeurs de D. On suppose que D est récursivement énumérable ou que l'on s'intéresse à un fragment de D qui soit récursivement énumérable. En effet, les éléments de D doivent pouvoir être codés dans L.

Ce problème est une variante du problème du model-checking existentiel de LTL.

Proposition 5.3. Soit $\mathcal{D} = \langle D, R_1, \ldots, R_n \rangle$ un domaine concret tel que

1. il existe une relation S telle que $S \neq \emptyset$ et $S \neq D^a$ ($\mathcal{D}$ est non-trivial) ;
2. chaque relation de $\mathcal{D}$ peut se décider en espace polynomial (avec le codage approprié des éléments de D).

MC1(CLTL($\mathcal{D}$)) est PSPACE-complet.

L'idée de la preuve est de montrer que MC1(CLTL($\mathcal{D}$)) est une variante de MC$^{\exists}$(LTL).

Preuve. Pour montrer la PSPACE-dureté, nous allons réduire MC$^{\exists}$(LTL) à MC1(CLTL($\mathcal{D}$)).

Soit $\mathcal{M} = \langle W, R, L \rangle$ une structure de Kripke finie et totale, $s \in W$, et $\phi \in$ LTL.

Par hypothèse, il existe une relation S de $\mathcal{D}$ qui est non triviale. Disons que la dimension de S est $a \geq 1$.

Pour chaque variable propositionnelle p de ϕ, nous associons un ensemble $\{x_1^{\mathrm{p}}, \ldots, x_a^{\mathrm{p}}\}$ de variables de VAR.

Nous allons construire une $\mathcal{D}$-structure de Kripke $\mathcal{M}' = \langle W', R', L' \rangle$, $s' \in W'$ et $\phi' \in \mathrm{CLTL}(\mathcal{D})$ telle que $\mathcal{M}, s \models_\exists \phi$ ssi $\mathcal{M}', s' \models_\exists \phi'$.

ϕ' est obtenue à partir de ϕ en remplaçant chaque occurrence de p par $\mathsf{S}(x_1^{\mathrm{p}}, \ldots, x_a^{\mathrm{p}})$.

$\mathcal{M}'$ est défini ainsi :
- $W' = W$, $R' = R$, et $s' = s$;
- L' est défini de sorte que $\mathrm{p} \in L(t)$ ssi $(L'(t)(x_1^{\mathrm{p}}), \ldots, L'(t)(x_a^{\mathrm{p}})) \in S$. Le calcul de $L'(t)(x_1^{\mathrm{p}}), \ldots, L'(t)(x_a^{\mathrm{p}})$ se fait en temps constant par rapport à $|\mathcal{M}| + |\phi|$ (paramètre dépendant de $\mathcal{D}$).

Montrons à présent que MC1(CLTL($\mathcal{D}$)) est dans PSPACE.

On note CONS(V, k) l'ensemble de formules atomiques de CLTL$(\mathcal{D})$ utilisant les variables de l'ensemble V (généralement fini) préfixées d'au plus $k-1$ symboles X.

Soient $\mathcal{M}$ une $\mathcal{D}$-structure de Kripke finie et totale, $s \in W$, et ϕ une formule de CLTL$(\mathcal{D})$. On note V l'ensemble des variables apparaissant dans $\mathcal{M}$ et ϕ, et k le plus grand entier i apparaissant dans $\mathrm{X}^i x$ augmenté de 1.

Le cardinal de CONS(V, k) est polynomial en $|\phi| + |\mathcal{M}|$.

Soit $\mathcal{M}' = \langle W', R', L' \rangle$ le modèle de Kripke défini sur l'ensemble des variables propositionnelles

$$\{\mathrm{p}_c : c \in \mathrm{CONS}(V, k)\}$$

défini de la façon suivante :

- W' est l'ensemble des tuples $\langle t_1, \ldots, t_k \rangle$ de W^k tel que pour $1 \leq i \leq k-1$, $\langle t_i, t_{i+1} \rangle \in R$;
- $\langle \langle t_1, \ldots, t_k \rangle, \langle s_1, \ldots, s_k \rangle \rangle \in R'$ ssi pour $1 \leq i \leq k-1$, $s_i = t_{i+1}$ (décalage d'une transition) ;
- $L'(\langle t_1, \ldots, t_k \rangle) = \{\mathrm{p}_c : c \in \mathrm{CONS}(V, k),\ \mathcal{M}, t_1 \ldots t_k \models c\}$.

Soit ϕ' la formule de LTL obtenue à partir de ϕ en remplaçant la formule atomique c par p_c.

Nous avons $\mathcal{M}, s \models \phi$ ssi $\mathcal{M}', \langle s_1, \ldots, s_k \rangle \models \phi'$ pour un état $\langle s_1, \ldots, s_k \rangle$ de W' avec $s_1 = s$.

On peut alors montrer que cette dernière propriété peut être vérifiée avec un espace mémoire polynomial en $|\mathcal{M}| + |\phi|$ et la réduction ne nécessite aussi qu'un espace mémoire polynomial. C.Q.F.D.

Corollaire 5.4. MC1(CLTL($\mathcal{D}$)) pour $\mathcal{D}$ dans

$$\{\langle \mathbb{Q}, =, < \rangle, \langle \mathbb{N}, =, < \rangle, \langle \Sigma^*, =, \subseteq \rangle, \langle \mathbb{N}, =, succ \rangle\}$$

est PSPACE-complet.

Dans le résultat ci-dessus, on utilise un codage "raisonnable" des éléments de $\mathbb{N}$, $\mathbb{Q}$, et Σ^*.

5.3.2 Model-checking 2

Nous allons dans cette section introduire un second problème de model-checking qui utilise directement des automates avec contraintes.

Définition 5.1. Un $\mathcal{D}$-automate $\mathcal{A}$ est un automate de Büchi sur l'alphabet (infini) des CLTL($\mathcal{D}$) formules. Les transitions de $\mathcal{A}$ sont donc de la forme $q \xrightarrow{\phi} q'$. ∇

La figure 4 présente un $\langle \mathbb{R}, =, < \rangle$-automate construit sur les variables $\{x, y, z\}$.

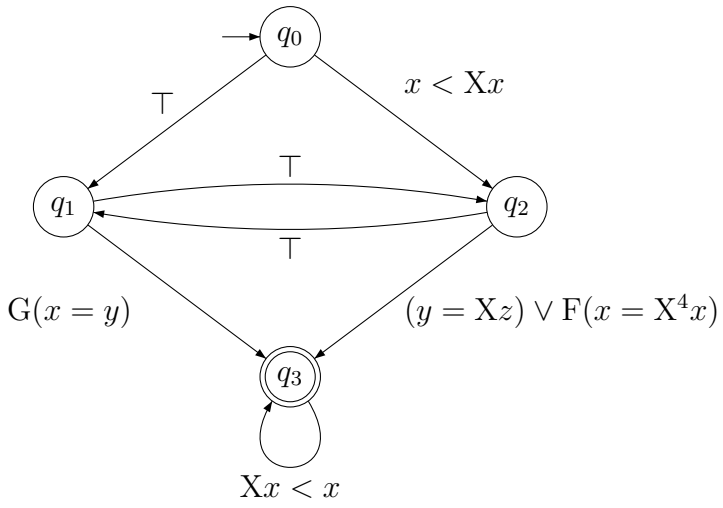


FIG. 4 – Un exemple de $\langle \mathbb{R}, =, < \rangle$-automate

Un $\mathcal{D}$-automate reconnaît un langage $\mathrm{L}(\mathcal{A})$ sur l'alphabet infini des CLTL($\mathcal{D}$) formules comme les automates temporisés reconnaissent des mots sur un alphabet composé de triplets action/garde/reset.

On note $l(\mathcal{A})$ l'ensemble des modèles $\sigma : \mathbb{N} \to (V \to D)$ ($\mathcal{A}$ est construit sur V) tels qu'il existe $\phi_0\phi_1 \ldots$ dans

L($\mathcal{A}$) avec pour $i \in \mathbb{N}$, $\sigma, i \models \phi_i$.

σ peut être vu comme une réalisation de $\phi_0\phi_1 \ldots$.

Pour $\mathcal{A}$ de la figure 4 avec $\mathcal{D} = \langle \mathbb{R}, =, < \rangle$ $l(\mathcal{A})$ est non-vide alors que si le domaine sous-jacent est $\mathcal{D} = \langle \mathbb{N}, =, < \rangle$, $l(\mathcal{A})$ est vide.

Problème du model-checking sur les $\mathcal{D}$-automates MC2(CLTL($\mathcal{D}$))

entrée : un $\mathcal{D}$-automate et une CLTL($\mathcal{D}$) formule construits sur les variables de V.

sortie : 1 s'il existe $\sigma \in l(\mathcal{A})$ tel que $\sigma \models \phi$. 0 sinon.

Proposition 5.5. Soit $\mathcal{D}$ un domaine concret non trivial. MC2(CLTL($\mathcal{D}$)) et SAT(CLTL($\mathcal{D}$)) sont inter-réductibles en espace logarithmique.

Preuve. SAT(CLTL($\mathcal{D}$)) peut être réduit à MC2(CLTL($\mathcal{D}$)) en observant que ϕ est CLTL($\mathcal{D}$) satisfaisable ssi $\mathcal{A}_\top \models_\exists \phi$ où $\mathcal{A}_\top$ est l'automate avec un seul état, $q_0 \xrightarrow{\top} q_0$, et $Q_0 = F = \{q_0\}$.

Pour réduire MC2(CLTL($\mathcal{D}$)) à SAT(CLTL($\mathcal{D}$)) on peut adapter la preuve de la réduction de MC$^\exists$(LTL) à SAT(LTL) (laissée en exercice). C.Q.F.D.

5.4 Indécidabilité de LTL avec contraintes de Presburger élémentaires

5.4.1 Machine de Minsky à 2 compteurs

Une machine à deux compteurs non-déterministe est composée de deux compteurs C_0 et C_1 et d'une séquence de n instructions. La l^{ieme} instruction a une des deux formes suivantes :

- $add\langle l, C_i, l', l''\rangle$ signifiant "ajouter 1 au compteur C_i et aller soit à l'instruction l' soit à l'instruction l'' ;
- $sub\langle l, C_i, l', l''\rangle$ signifiant "si $C_i > 0$, alors soustraire 1 à C_i et aller à l'instruction l', sinon aller à l'instruction l''.

Une configuration est un triplet $\langle ic, n_0, n_1\rangle \in \{1, \ldots, n\} \times \mathbb{N} \times \mathbb{N}$ où ic est le compteur d'instruction, et n_i est la valeur du compteur C_i.

Une exécution est une séquence infinie de configurations commençant à la configuration initiale $\langle 1, 0, 0\rangle$ et telle que deux configurations successives soient admissibles avec la définition de la machine.

Une exécution est récurrente ssi elle admet une exécution avec un nombre infini de configurations ayant le compteur

d'instruction égale à 1 (condition de Büchi).

Proposition 5.6. Déterminer si une machine à deux compteurs non-déterministe admet une exécution récurrente est un problème indécidable.

Le résultat de Minsky initial est l'indécidabilité du problème de l'arrêt des machines (déterministes) à deux compteurs. Le problème ci-dessus est plus difficile.

5.4.2 La preuve d'indécidabilité

L'indécidabilité de SAT(CLTL($\mathcal{D}$)) pour certains domaines $\mathcal{D}$ n'est pas très difficile à obtenir. Nous en donnons un exemple dans la proposition 5.7 qui peut être adaptée à d'autres domaines comme $\langle \mathbb{R}, =, succ \rangle$, $\langle \mathbb{N}, =, \times_2 \rangle$ etc. La preuve est inspirée d'une preuve similaire faite dans [AH93].

Proposition 5.7. SAT(CLTL($\langle \mathbb{N}, =, succ \rangle$)) est indécidable.

Preuve. Soit M une machine à deux compteurs non-déterministe.

Nous allons construire une CLTL($\langle \mathbb{N}, =, succ \rangle$) formule ϕ_M (en espace logarithmique en la taille de M) telle que ϕ_M est CLTL($\langle \mathbb{N}, =, succ \rangle$) satisfaisable ssi M a une exécution récurrente.

Pour coder les configurations d'une machine M, nous utilisons les variables ic, n_1, et n_2.

Comme $\langle \mathbb{N}, =, succ \rangle$ ne contient pas de constantes, nous allons introduire des variables supplémentaires qui ont une valeur constante pour tout le modèle :

- $\mathsf{d}_1, \ldots, \mathsf{d}_\mathsf{n}$ codent les n valeurs différentes du compteur d'instruction ;
- $\mathsf{d}_{\mathsf{n}+1}$ et $\mathsf{d}_{\mathsf{n}+2}$ sont les valeurs initiales des compteurs C_0 et C_1.

Maintenant, nous définissons une formule ϕ_{inst} qui force l'interprétation de $\mathsf{d}_1, \ldots, \mathsf{d}_\mathsf{n}$ comme des valeurs constantes et distinctes :

$$\phi_{inst} \stackrel{\text{def}}{=} \overbrace{\bigwedge_{1 \leq i < j \leq n} (\neg(\mathsf{d}_i = \mathsf{d}_j))}^{d_1,\ldots,d_n \text{ sont distincts}} \wedge \mathrm{G}(\bigwedge_{1 \leq i \leq n} \mathsf{d}_i = \mathrm{X}\mathsf{d}_i).$$

De façon analogue, nous définissons une formule ϕ_{init-c} qui force l'interprétation de $\mathsf{d}_{\mathsf{n}+1}$ et $\mathsf{d}_{\mathsf{n}+2}$ comme constantes (non nécessairement distinctes) :

$$\phi_{init-c} \stackrel{\text{def}}{=} \mathrm{G}(\mathsf{d}_{n+1} = \mathrm{X}\mathsf{d}_{n+1} \wedge \mathsf{d}_{n+2} = \mathrm{X}\mathsf{d}_{n+2}).$$

Nous définissons une formule qui énonce quelle est la valeur initiale du compteur d'instruction et quelles sont

les valeurs initiales des compteurs :

$$\phi_{init} \stackrel{\text{def}}{=} (\mathsf{ic} = \mathsf{d}_1) \wedge (\mathsf{n}_0 = \mathsf{d}_{n+1}) \wedge (\mathsf{n}_1 = \mathsf{d}_{n+2}).$$

Il reste à spécifier comment les valeurs des configurations courantes évoluent. Nous devons spécifier les relations entre deux configurations successives.

Pour chaque instruction l, nous introduisons une formule φ_l qui spécifie ces effets sur les différents compteurs.

Par exemple, ajouter 1 au compteur C_i est codé par la formule atomique $succ(\mathsf{n}_i, \mathrm{X}\mathsf{n}_i)$.

Si la l^{ieme} instruction est de la forme $add\langle l, C_i, l', l''\rangle$, la formule φ_l est la suivante :

$$\overbrace{((\mathsf{d}_{l'} = \mathrm{X}\mathsf{ic}) \vee (\mathsf{d}_{l''} = \mathrm{X}\mathsf{ic}))}^{\text{aller en } l' \text{ ou } l''} \wedge \overbrace{succ(\mathsf{n}_i, \mathrm{X}\mathsf{n}_i)}^{\text{ajouter 1 a } C_i} \wedge \overbrace{(\mathrm{X}\mathsf{n}_{1-i} = \mathsf{n}_{1-i})}^{C_{1-i} \text{ ne change pas}}.$$

De même, si la l^{ieme} instruction est de la forme $sub\langle l, C_i, l', l''\rangle$, la formule φ_l est la suivante :

$$\overbrace{(\neg(\mathsf{n}_i = \mathsf{d}_{n+1+i})}^{C_i \neq 0} \Rightarrow \overbrace{(\mathsf{d}_{l'} = \mathrm{X}\mathsf{ic})}^{\text{aller en } l'} \wedge \overbrace{succ(\mathrm{X}\mathsf{n}_i, \mathsf{n}_i)}^{\text{soustraire 1 a } C_i} \wedge \overbrace{(\mathrm{X}\mathsf{n}_{1-i} = \mathsf{n}_{1-i})}^{C_{1-i} \text{ ne change pas}})$$

$$\wedge(\overbrace{(\mathsf{n}_i = \mathsf{d}_{n+1+i})}^{C_i = 0} \Rightarrow \overbrace{(\mathsf{d}_{l''} = \mathrm{X}\mathsf{ic})}^{\text{aller en } l''} \wedge \overbrace{(\mathrm{X}\mathsf{n}_0 = \mathsf{n}_0)}^{C_0 \text{ ne change pas}} \wedge \overbrace{(\mathrm{X}\mathsf{n}_1 = \mathsf{n}_1)}^{C_1 \text{ ne change pas}}).$$

Il reste à exprimer que l'exécution est récurrente avec la formule ϕ_{rec} :

$$\mathrm{G}(\top \mathrm{U}(\mathsf{ic} = \mathsf{d}_1)).$$

Soit ϕ_M la formule ci-dessous :

$$\phi_{inst} \wedge \phi_{init-c} \wedge \wedge \phi_{rec} \wedge \mathrm{G}(\bigwedge_{1 \leq l \leq n} ((\mathsf{ic} = \mathsf{d}_l) \Rightarrow \varphi_l)).$$

On peut vérifier que M a une exécution récurrente ssi ϕ_M est CLTL($\langle \mathbb{N}, =, succ \rangle$) satisfaisable. C.Q.F.D.

Exercice 5.2. Montrer que CLTL($\langle \mathbb{R}, =, \times_2 \rangle$) est indécidable avec $\times_2 = \{\langle c, 2 \times c \rangle : c \in \mathbb{R}\}$.

5.5 Décidabilité avec propriétés de complétion

5.5.1 Sémantique à la LTL

Dans la suite on identifie le terme $\mathrm{X}^n x$ ($n \in \mathbb{N}$ et $x \in$ VAR) sous-terme des formules atomiques de CLTL($\mathcal{D}$) avec la paire $\langle n, x \rangle$.

Pour un ensemble fini U de $\mathbb{N} \times$VAR, on note CONS(U) l'ensemble des formules atomiques de CLTL($\mathcal{D}$) construites sur les termes de U.

CONS(V, k) dénote l'ensemble

$$\text{CONS}(\{0, \ldots, k-1\} \times V)$$

pour un sous-ensemble fini V de VAR.

Exemple 5.8. $\mathcal{D} = \langle \mathbb{N}, =, < \rangle$ et CONS$(\{0,1\} \times \{x\}) =$ CONS$(\{x\},2)$ est égale à :

$$\{x = \mathrm{X}x, \mathrm{X}x = x, x < \mathrm{X}x, \mathrm{X}x < x\} \cup$$

$$\{\mathrm{X}x = \mathrm{X}x, x = x, x < x, \mathrm{X}x < \mathrm{X}x\}.$$

∇

VAR(ϕ) : variables de VAR apparaissant dans ϕ.

X(ϕ) : plus petit k tel que toutes les formules atomiques de ϕ sont dans CONS(V, k) avec $V =$ VAR(ϕ).

Définition 5.2. Une U-valuation est une fonction $v : U \to D$. v satisfait $c = \mathsf{R}(\mathrm{X}^{n_1}x_1, \ldots, \mathrm{X}^{n_a}x_a)$ en supposant que les termes de c sont dans U, si

$$(v(\langle n_1, x_1 \rangle), \ldots, v(\langle n_a, x_a \rangle)) \in R$$

(noté $v \models c$). ∇

Définition 5.3. Un ensemble $X \subseteq \mathrm{CONS}(U)$ est maximalement consistant ssi il existe une U-valuation $v : U \to D$ telle que $X = \{c \in \mathrm{CONS}(U) : v \models c\}$. ∇

Les modèles $\sigma : \mathbb{N} \to (V \to D)$ peuvent être abstraits comme des séquences $\mathbb{N} \to \mathcal{P}(\mathrm{CONS}(V, k))$. $\overline{\sigma} : \mathbb{N} \to \mathcal{P}(\mathrm{CONS}(V, k))$ est définie à partir de σ de la façon suivante : pour $i \in \mathbb{N}$,

$$\overline{\sigma}(i) = \{c \in \mathrm{CONS}(V, k) : \sigma, i \models c\}.$$

L'opérateur $\overline{\cdot}$ dépend de V et de $k \geq 1$.

Une séquence $\sigma : \mathbb{N} \to \mathcal{P}(\mathrm{CONS}(V, k))$ est $\mathcal{D}$-réalisable ssi il existe $\sigma' : \mathbb{N} \to (V \to D)$ telle que $\overline{\sigma'} = \sigma$.

Par exemple, $\{x = x, \mathrm{X}x < x, \mathrm{X}x = \mathrm{X}x\}^\omega$ n'est pas $\langle \mathbb{N}, =, < \rangle$-réalisable bien que $\{x = x, \mathrm{X}x < x, \mathrm{X}x = \mathrm{X}x\}$ soit maximalement consistant.

Pour $\sigma : \mathbb{N} \to \mathcal{P}(\mathrm{CONS}(V, k))$ et $\phi \in \mathrm{CLTL}(\mathcal{D})$ construite sur les formules atomiques de $\mathrm{CONS}(V, k)$, on dénote par $\sigma, i \models_{\mathrm{LTL}} \phi$ la satisfaction de ϕ dans l'état i quand les formules atomiques de $\mathrm{CONS}(V, k)$ sont lues comme des variables propositionnelles : $\sigma, i \models_{\mathrm{LTL}} c \overset{\text{def}}{\Leftrightarrow} c \in \sigma(i)$ pour $c \in \mathrm{CONS}(V, k)$.

Lemme 5.9. Pour $\sigma : \mathbb{N} \to (V \to D)$, $\phi \in \mathrm{CLTL}(\mathcal{D})$ avec $V = \mathrm{VAR}(\phi)$ et $k = \mathrm{X}(\phi)$, pour $i \in \mathbb{N}$, $\sigma, i \models \phi$ ssi $\overline{\sigma}, i \models_{\mathrm{LTL}} \phi$.

La preuve du lemme ci-dessus est immédiate. Une conséquence intéressante est le résultat suivant.

Lemme 5.10. Soit $\phi \in \mathrm{CLTL}(\mathcal{D})$ avec $k = \mathrm{X}(\phi)$ et $V = \mathrm{VAR}(\phi)$. ϕ est satisfaisable ssi il existe $\sigma : \mathbb{N} \to \mathcal{P}(\mathrm{CONS}(V, k))$ tel que

1. $\sigma, 0 \models_{\mathrm{LTL}} \phi$;
2. σ est $\mathcal{D}$-réalisable.

Exercice 5.3. Montrer qu'il n'existe pas de formule ϕ de $\mathrm{CLTL}(\langle \mathbb{R}, = \rangle)$ construite sur la seule variable x telle que la classe des modèles pour ϕ est précisément la classe des modèles $\sigma : \mathbb{N} \to (\{x\} \to \mathbb{R})$ avec pour $i < j$, $\sigma(i)(x) \neq \sigma(j)(x)$.

D'après les résultats sur LTL que nous connaissons, nous pouvons construire (éventuellement à la volée) un automate de Büchi qui reconnaisse les mots de $\mathcal{P}(\mathrm{CONS}(V, k))^\omega$ vérifiant ϕ avec la relation de satisfaction $\models_{\mathrm{LTL}}$.

S'il est possible de construire effectivement un automate de Büchi qui reconnaisse les mots de $\mathcal{P}(\mathrm{CONS}(V, k))^\omega$

$\mathcal{D}$-réalisables, alors la décidabilité de CLTL($\mathcal{D}$) est assurée.

Remarquez que dans le cas général, on ne peut garantir la ω-régularité de

$$\{\overline{\sigma} : \sigma, 0 \models \phi \ \& \ \sigma : \mathbb{N} \to (V \to D)\}.$$

5.5.2 Séquence localement consistante

Définition 5.4. Une séquence $\sigma : \mathbb{N} \to \mathcal{P}(\mathrm{CONS}(V, k))$ est localement consistante ssi

1. pour $i \in \mathbb{N}$, $\sigma(i)$ est maximalement consistant,
2. pour $i \in \mathbb{N}$, pour $\mathsf{R}(\mathrm{X}^{n_1}x_1, \ldots, \mathrm{X}^{n_a}x_a) \in \mathrm{CONS}(V, k)$ telle que $n_1, \ldots, n_a \geq 1$,

$$\mathsf{R}(\mathrm{X}^{n_1}x_1, \ldots, \mathrm{X}^{n_a}x_a) \in \sigma(i) \text{ ssi}$$
$$\mathsf{R}(\mathrm{X}^{n_1-1}x_1, \ldots, \mathrm{X}^{n_a-1}x_a) \in \sigma(i+1).$$

∇

Lemme 5.11. Pour $\sigma : \mathbb{N} \to (V \to D)$ et $k \geq 1$, $\overline{\sigma} : \mathbb{N} \to \mathcal{P}(\mathrm{CONS}(V, k))$ est localement consistante.

Par conséquent, une condition nécessaire d'être $\mathcal{D}$-réalisable pour une séquence $\sigma : \mathbb{N} \to \mathcal{P}(\mathrm{CONS}(V, k))$ est d'être localement consistante.

5.5.3 Domaines vérifiant la propriété de complétion

Un domaine $\mathcal{D} = \langle D, R_1, \ldots, R_n \rangle$ vérifie la propriété de complétion ssi pour tout ensemble fini $U \subseteq \mathbb{N} \times V$, pour $X \subseteq \mathrm{CONS}(U)$ maximalement consistant, si

- $U' \subseteq U$;
- $v : U' \to D$ avec $X \cap \mathrm{CONS}(U') = \{c \in \mathrm{CONS}(U') : v \models c\}$;
  v satisfait exactement les contraintes de X qui utilisent des termes de U' ;

alors il existe une extension $v' : U \to D$ de v telle que

$$X = \{c \in \mathrm{CONS}(U) : v' \models c\}.$$

Le point crucial dans la définition ci-dessus est que v' est une extension de v.

Lemme 5.12. $\langle \mathbb{Q}, =, < \rangle$, $\langle \mathbb{R}, =, < \rangle$, et $\langle \mathbb{R}^n, =, <_{lex} \rangle$ vérifient la propriété de complétion. $\langle \mathbb{N}, =, < \rangle$ et $\langle \mathbb{Z}, =, < \rangle$ ne vérifient pas la propriété de complétion.

La propriété de complétion garantit que la $\mathcal{D}$-réalisation est une propriété ω-régulière.

Lemme 5.13. Soit $\mathcal{D}$ un domaine concret qui vérifie la propriété de complétion. Si $\sigma : \mathbb{N} \to \mathcal{P}(\mathrm{CONS}(V, k))$ est localement consistant alors σ est $\mathcal{D}$-réalisable.

Cette propriété a été utilisée dans [BC02].

Preuve. Soit $\sigma : \mathbb{N} \to \mathcal{P}(\mathrm{CONS}(V,k))$ une séquence localement consistante. Nous allons définir $\sigma' : \mathbb{N} \to (V \to D)$ telle que $\overline{\sigma'} = \sigma$, c'est-à-dire pour $i \in \mathbb{N}$, pour $c \in \mathrm{CONS}(V,k)$, $\sigma', i \models c$ ssi $c \in \sigma(i)$.

Comme σ est localement consistante, pour $i \in \mathbb{N}$, il existe $v_i : \{0,\ldots,k-1\} \times V \to D$ telle que pour $c \in \mathrm{CONS}(V,k)$, $c \in \sigma(i)$ ssi $v_i \models c$.

Dans la suite on suppose $k \geq 2$ (le cas $k = 1$ étant immédiat).

En particulier il existe $v_0 : \{0,\ldots,k-1\} \times V \to D$ telle que pour $c \in \mathrm{CONS}(V,k)$, $c \in \sigma(0)$ ssi $v_0 \models c$.

Nous posons $\sigma'(\alpha, x) = v_0(\alpha, x)$ pour $\langle \alpha, x \rangle \in \{0,\ldots,k-1\} \times V$.

Supposons que σ' soit définie sur $\{0,\ldots,\beta\} \times V$ pour un $\beta \geq k-1$ et pour $c \in \mathrm{CONS}(V,k)$, $\sigma', \beta-(k-1) \models c$ ssi $c \in \sigma(\beta-(k-1))$. Nous allons étendre la définition de σ' à $\{\beta+1\} \times V$ vérifiant pour $c \in \mathrm{CONS}(V,k)$, $\sigma', \beta-(k-1)+1 \models c$ ssi $c \in \sigma(\beta-(k-1)+1)$.

Nous savons que
– $\sigma(\beta - k + 2)$ est maximalement consistant ;
– En posant $U' = \{0, \ldots, k-2\} \times V$ et $v : U' \to D$ telle que $v(\alpha, x) = \sigma'(\beta - k + 2 + \alpha, x)$, nous avons

$$\sigma(\beta - k + 2) \cap \mathrm{CONS}(U') = \{c \in \mathrm{CONS}(U') : v \models c\}.$$

En effet, comme σ est localement consistant, pour $c \in \mathrm{CONS}(U')$, $c \in \sigma(\beta - k + 2)$ ssi $\sigma', \beta - k + 2 \models c$ (par HI) ssi $v \models c$ (par définition de v).

Comme $\mathcal{D}$ vérifie la propriété de complétion, il existe $v' : \{0, \ldots, k-1\} \times V \to D$ extension de v telle que $\sigma(\beta - k + 2) = \{c \in \mathrm{CONS}(V, k) : v' \models c\}$.

Nous posons $\sigma'(\beta + 1, x) = v'(k - 1, x)$ pour $x \in V$.

En définissant ainsi σ' sur $\mathbb{N} \times V$, nous garantissons que $\overline{\sigma'} = \sigma$. C.Q.F.D.

5.5.4 PSPACE-complétude

Le problème de la $\mathcal{D}$-consistance maximale est le suivant :

entrée : un ensemble fini X inclus dans $\mathrm{CONS}(U)$, avec $U \subseteq \mathbb{N} \times \mathrm{VAR}$ fini ;

sortie : 1 si X maximalement consistant (par rapport à CONS(U)). 0 sinon.

Lemme 5.14. Le problème de la $\langle \mathbb{R}, =, < \rangle$-consistance maximale est NLOGSPACE-complet.

Exercice 5.4. Montrer que si $\mathcal{D}$ a un problème de consistance maximale dans PSPACE et si $\mathcal{D}$ vérifie la propriété de complétion, alors SAT(CLTL($\mathcal{D}$)) est dans PSPACE.

Exercice 5.5. Montrer que si $\mathcal{D}$ est non-trivial alors SAT(CLTL($\mathcal{D}$)) est PSPACE-difficile.

Corollaire 5.15. [BC02, DD03] Pour $\mathcal{D} \in \{\langle \mathbb{Q}, =, < \rangle, \langle \mathbb{R}, =, < \rangle, \langle \mathbb{R}^n, =, <_{lex} \rangle\}$, SAT(CLTL($\mathcal{D}$)) et MC2(CLTL($\mathcal{D}$)) sont PSPACE-complets.

Références

[AH93] R. Alur and T. Henzinger. Real-time logics : complexity and expressiveness. *Information and Computation*, 104(1) :35–77, 1993.

[BC02] Ph. Balbiani and J.F. Condotta. Computational complexity of propositional linear temporal logics based on qualitative spatial or temporal reasoning. In A. Armando, editor, *Frontiers of Combining Systems (FroCoS'02)*, volume 2309 of *Lecture Notes in Artificial Intelligence*, pages 162–173. Springer, Berlin, 2002.

[Coo71] S. Cook. The complexity of theorem-proving procedures. In *Third Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[DD03] S. Demri and D. D'Souza. An automata-theoretic approach to constraint LTL. Technical Report LSV-03-11, LSV, August 2003. 40 pages.

[DS02] S. Demri and Ph. Schnoebelen. The complexity of propositional linear temporal logics in simple cases. *Information and Computation*, 174(1) :84–103, 2002.

[GK03] P. Gastin and D. Kuske. Satisfiability and model checking for MSO-definable temporal logics are in PSPACE. In *CONCUR'03, Marseille, France*, volume 2761 of *Lecture Notes in Computer Science*, pages 222–236. Springer, 2003.

[Har85] D. Harel. Recurring dominoes : making the highly undecidable highly understandable. *Annals of Discrete Mathematics*, 24 :51–72, 1985.

[HKT00] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, Cambridge, MA, 2000.

[HT99] J. Henriksen and P. Thiagarajan. Dynamic linear time temporal logic. *Annals of Pure and Applied Logic*, 96(1–3) :187–207, 1999.

[Jon75] N. Jones. Space-bounded reducibility among combinatorial problems. *Journal of Computer and System Sciences*, 11(1) :68–85, 1975.

[Kam68] J. Kamp. *Tense Logic and the theory of linear order*. PhD thesis, UCLA, USA, 1968.

[Lam83] L. Lamport. What good is temporal logic ? In *Information Processing'83. Proc. IFIP 9th World Computer Congress, Sep. 1983, Paris, France*, pages 657–668. North-Holland, 1983.

[Pap94] Ch. Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, MA, 1994.

[SC85] P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the Association for Computing Machinery*, 32(3) :733–749, 1985.

[Sch03] Ph. Schnoebelen. The complexity of temporal logic model checking. In *Advances in Modal Logic, vol. 4, selected papers from 4th Conf. Advances in Modal Logic (AiML'2002), Sep.-Oct. 2002, Toulouse, France*, pages 437–459. King's College Publication, 2003.

[SM73] L. Stockmeyer and A. Meyer. Word problems requiring exponential-time. In *5th ACM Symposium on the Theory of Computing*, pages 1–9, 1973.

[Var88] M. Vardi. A temporal fixpoint calculus. In *15th Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, San Diego*, pages 250–259. ACM, 1988.

[Var96] M. Vardi. An automata-theoretic approach to linear temporal logic. In F. Moller and G. Birtwistle, editors, *Logics of Concurrency : Structure versus Automata*, volume 1043 of *Lecture Notes in Computer Science*, pages 238–266. Springer, Berlin, 1996.

[VW94] M. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115 :1–37, 1994.

[Wol83] P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56 :72–99, 1983.