7.2 Les résultats de décision

Les résultats obtenus pour S1S se généralisent à SkS (au prix d'une complexité supplémentaire) :

Theorème 40 *Le vide des automates d'arbres avec condition de parité (resp. de Rabin) est dans co-NP. (En fait co-NP-complet).*

Preuve: On ramène le problème du non-vide des automates d'arbres avec condition de parité au problème du non vide des automates alternants de mots, avec condition de parité, sur un alphabet à une lettre : l'automate de mots a pour états $Q \times A$, le rang est défini par $r'(q, f) = r(q)$, les états initiaux sont (q^0, f) et la relation de transition est définie par :

$$\delta'(q, f) = \bigvee_{(q,f,(q_1,\ldots,q_k))\in\delta} \bigvee_{g\in A} \bigwedge_{i=1}^{k} (q_i, g)$$

Il existe un arbre t sur lequel l'automate d'arbres a un calcul réussi si et seulement si l'automate de mots a un calcul réussi : ce calcul donne en même temps l'arbre t qui est deviné à chaque étape.

On utilise alors le théorème 35. □

Theorème 41 (Rabin, 1969) *: Un ensemble d'arbres infinis est définissable en SkS si et seulement si il est reconnu par un automate de Rabin.*

Nous ne donnons pas la preuve du théorème ici (voir par exemple l'excellent article de synthèse [Tho97]). Il faut d'abord montrer les propriétés de cloture des automates à parité. Le point le plus délicat est la clôture par complémentaire qui s'obtient, comme dans le chapitre précédent, grâce à un résultat de détermination des jeux, dans le cas général des jeux à parité.

Comme le vide est décidable pour les automates de Rabin, on obtient en corollaire :

Corollaire 42 *SkS est décidable.*

8 Logiques du temps arborescent (CTL*et CTL)

Syntaxe

formules d'états ϕ_s, ψ_s
- **true**, **false**
- P pour $P \in \mathcal{P}$
- $\phi_s \wedge \psi_s$, $\phi_s \vee \psi_s$, $\neg\phi_s$
- $A\phi_p$, $E\phi_p$

formules de chemins ϕ_p, ψ_p
- ϕ_s
- $\phi_p \wedge \psi_p$, $\phi_p \vee \psi_p$, $\neg\phi_p$
- $\mathbf{X}\phi_p$
- $\phi_p \,\mathcal{U} \psi_p$

On utilise aussi comme en LTL les abréviations $\Diamond\phi \stackrel{\text{def}}{=} \mathbf{true}\,\mathcal{U}\phi$ et $\Box\phi \stackrel{\text{def}}{=} \neg(\mathbf{true}\,\mathcal{U}\neg\phi)$.

Sémantique Soit $\mathcal{K}$ une structure de Kripke. $T_{\mathcal{K}}$ est le dépliage de $\mathcal{K}$: c'est un arbre infini étiqueté par les positions de $\mathcal{K}$. Un *chemin* d'origine s de $T_{\mathcal{K}}$ est une suite de positions de $\mathcal{K}$: c'est une éxecution de $\mathcal{K}$ à partir de s.

$\mathcal{K} \models \phi_s$ ssi $T, \epsilon \models \phi_s$.

formules d'états ϕ_s, ψ_s
- $T, t \models$ **true**, et $T, t \not\models$ **false**
- $T, t \models P$ ssi $P \in T(t)$
- $T, t \models \phi_s \wedge \psi_s$ ssi $T, t \models \phi_s$ et $T, t \models \psi_s$, etc...
- $T, t \models A\phi_p$, ssi pour tout chemin π de T d'origine t, $\pi, 0 \models_t \phi_p$
- $T, t \models E\phi_p$ ssi il existe un chemin π de T d'origine t tel que $\pi, 0 \models_t \phi_p$

formules de chemins ϕ_p, ψ_p
- $\pi, n \models_{t_0} \phi_s$ ssi $T, t \models \phi_s$ où $t = t_0 \cdot t_1$ et t_1 est le préfixe de π de longueur n.
- $\pi, t \models_{t_0} \phi_p \wedge \psi_p$ ssi $\pi, t \models_{t_0} \phi_p$ et $\pi, t \models_{t_0} \psi_p$, etc ...
- $\pi, t \models_{t_0} \mathbf{X}\phi_p$ ssi $\pi, t+1 \models_{t_0} \phi_p$
- $\pi, t \models_{t_0} \phi_p \,\mathcal{U} \psi_p$ ssi il existe $t' \geq t$ tel que
  - $\pi, t' \models_{t_0} \psi_p$
  - pour tout $t'', t \leq t' < t''$, $\pi, t'' \models_{t_0} \phi_p$

Exercice 29 Soit $\mathcal{K}$ la structure de Kripke donnée dans la figure 18. Représenter visuellement son dépliage et dire (en le justifiant) si $\mathcal{K}$ vérifie la formule $A\Diamond A\Box P$.

CTL*est au moins aussi expressive que LTL qu'elle contient (ce sont les formules de chemins) mais il y a nombre de propriétés qui sont expressibles en CTL*et pas en LTL. Par exemple les propriétés de vivacité comme

$$A\Box(P \Rightarrow E\Diamond Q)$$

"Quand P est vrai, il est possible d'obtenir Q ensuite".

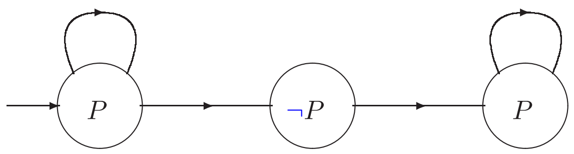

FIG. 18 – Une structure de Kripke

8.1 CTL

CTL est le sous-ensemble de CTL*dans lequel les formules de chemins $\mathbf{X}\phi_p$ et $\phi_p\,\mathcal{U}\psi_p$ sont immédiatement précédées d'un quantificateur sur les chemins A ou E.

Exercice 30 Montrer que $\Box\Diamond P$ n'est pas expessible en CTL.

8.2 Traduction en SkS

On considère seulement ici les structures de Kripke à branchement borné par un nombre fixe k. Nous verrons (lemme 49) que ce n'est pas une restriction. Par abus de notation, si K est une structure de Kripke, on note encore $t_\mathcal{K}$ le dépliage de K dans lequel les étiquettes $s \in S$ sont remplacées par $f(s) \in 2^\mathcal{P}$. On ordonne de plus arbitrairement les variables propositionnelles : $\mathcal{P} = \{P_1, \dots, P_n\}$.

Theorème 43 *Pour toute structure de Kripke de branchement au plus k et toute formule ϕ de* CTL**, il existe une formule $\widetilde{\phi}(t, X_1, \dots, X_n)$ de SkS (effectivement calculable en temps linéaire) telle que $\mathcal{K} \models \phi$ si et seulement si $\epsilon, t_\mathcal{K} \models \widetilde{\phi}(t, X_1, \dots, X_n)$*

Preuve: On peut toujours coder un nombre arbitraire (fini) de successeurs à l'aide de deux successeurs, l'image étant définissable en S2S. On se limite ainsi à deux successeurs.

On code les formules d'états comme des formules de S2S ayant une variable libre de premier ordre t et les formules de chemins comme des formules ayant une variable libre de second ordre C. ($X_1, \dots, X_n$ qui sont libres dans toutes les formules sont sous-entendus).

On utilise aussi les prédicats Origine,Suc et Chaîne, qui sont définissables en S2S, comme nous l'avons vu en exercice.

formules d'états ϕ_s, ψ_s

- $\widetilde{\mathbf{true}}(t) \stackrel{\text{def}}{=} \mathbf{true}$, $\widetilde{\mathbf{false}}(t) \stackrel{\text{def}}{=} \mathbf{false}$
- $\widetilde{P_i}(t) \stackrel{\text{def}}{=} t \in X_i$
- $\widetilde{\phi_s \wedge \psi_s}(t) \stackrel{\text{def}}{=} \widetilde{\phi_s}(t) \wedge \widetilde{\psi_s}(t)$, etc...
- $\widetilde{A\phi_p}(t) \stackrel{\text{def}}{=} \forall C.\text{Chaîne}(C, t) \Rightarrow \widetilde{\phi_p}(C)$, etc...

formules de chemins ϕ_p, ψ_p

- $\widetilde{\phi_s(C)} \stackrel{\text{def}}{=} \exists t.\text{Origine}(t, C) \wedge \widetilde{\phi_s}(t)$
- $\widetilde{\phi_p \wedge \psi_p}(C) \stackrel{\text{def}}{=} \widetilde{\phi_p}(C) \wedge \widetilde{\psi_p}(C)$, etc ...
- $\widetilde{\mathbf{X}\phi_p}(C) \stackrel{\text{def}}{=} \exists C'.\text{Suc}(C, C') \wedge \widetilde{\phi_p}(C')$
- $\widetilde{\phi_p\, \mathcal{U} \psi_p}(C) \stackrel{\text{def}}{=} \exists t.t \in C \wedge (\exists C'.\text{Origine}(t, C') \wedge \widetilde{\psi_p}(C') \wedge \forall t' \in C, \exists C''.t' \geq t \vee (\text{Origine}(t', C'') \wedge \widetilde{\phi_p}(C'')))$

La vérification de l'équivalence est alors routinière. $\square$

9 Automates d'arbres alternants

La notion clef cette fois ci est celle des automates d'arbres alternants (ou plus exactement des sous-classes de celle-ci). Nous introduisons et étudions les automates d'arbres alternants faibles (WAA) et leur algorithmique. Nous présentons ensuite tout ou partie des algorithmes optimaux de model-checking qui en découle :

- - CTL (polynomial)
- - mu-calcul sans alternation (polynomial)
- - CTL* (PSPACE-complet)
- - mu-calcul (NP inter co-NP)

Pourquoi la complexité pourrait-elle être réduite par rapport au problème de vérification en LTL, alors que les automates sont plus complexes ? En bref, c'est parce qu'on teste l'appartenance et non l'inclusion.

9.1 Automates alternants d'arbres ; différentes conditions d'acceptance

Les arbres considérés ici sont infinis et leur branchement est variable (éventuellement infini), l'ordre des fils n'étant pas significatif. Pour être plus précis, on considère un ensemble D (par exemple l'ensemble des entiers naturels). Un arbre T (sous-entendu à branchements dans D) étiqueté par A est une application d'un ensemble $Pos(T) \subseteq D^*$ clos par préfixe (appelé *ensemble des positions de* T) dans A. Si $t \in Pos(t)$, on note $s_T(t)$ l'ensemble des successeurs de t dans T : $s_T(t) \stackrel{\text{def}}{=} \{d \mid d \in D, t \cdot d \in Pos(T)\}$.

Un automate d'arbres alternant $\mathcal{A}$ est un tuple (A, Q, δ, q_0, F) où Q est ensemble fini d'états, $q_0 \in Q$ est l'état initial, A est l'alphabet d'entrée, supposé fini. $\delta : Q \times A \to \mathcal{B}^+(\{\Box, \Diamond\} \times Q)$.

La taille d'un automate alternant est (comme pour les mots) égale au nombre d'états + taille de l'alphabet + taille du système de transition. Cette dernière est définie comme la somme, pour chaque transition qui ne va pas à **false**de la taille de la formule résultat.

Un *calcul* de l'automate sur un arbre T est un arbre T_r dont la racine est étiquetée par q_0 et tous les autres nœuds sont étiquetés par un élément de $Q \times Pos(T)$. De plus, si $T_r(u) = (q, t)$, $T(t) = a$, alors $\delta(q, a)$ est défini et, il existe un ensemble (éventuellement vide) $S = \{(\Box, q_1), \ldots, (\Box, q_k), (\Diamond, q'_1), \ldots, (\Diamond, q'_m)\}$ tel que S satisfait $\delta(q, a)$ et

- pour tout $d \in s_T(t)$, pour tout $i = 1, ...k$, il existe un $j \in s_{T_r}(u)$ tel que $T_r(u \cdot j) = (q_i, t \cdot d)$
- pour tout $i = 1, ..., m$, il exite un $d_i \in s_T(t)$, il existe un $j \in s_{T_r}(u)$ tel que $T_r(u \cdot j) = (q'_i, t \cdot d_i)$

Un calcul est *réussi* s'il satisfait les conditions définies par F (ce peut être des conditions de Büchi ou de Rabin par exemple).

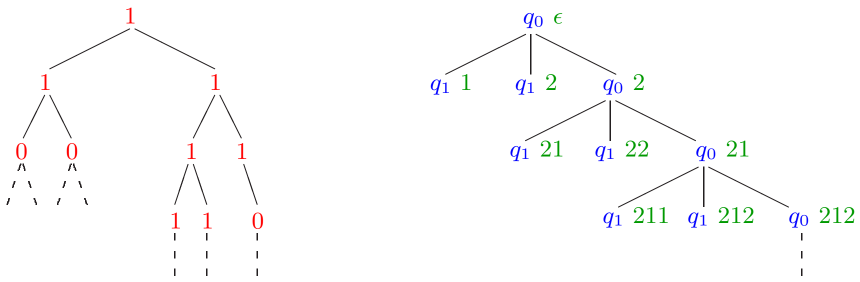

FIG. 19 – Un exemple de calcul d'un automate alternant d'arbres

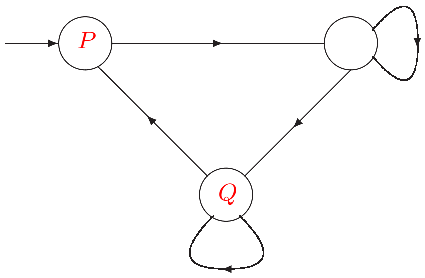

FIG. 20 – Une structure de Kripke

Exemple 44 Soit δ la relation de transition définie par la table :

δ	0	1
q_1	**false**	**true**
q_0	**true**	$(\Box, q_1) \wedge (\Diamond, q_0)$

L'état initial étant q_0, un arbre et un calcul de l'automate sont représentés sur la figure 19.

Exercice 31 Soit l'automate d'arbres alternants dont les transitions sont données par la table :

	$\begin{smallmatrix}0\\0\end{smallmatrix}$	$\begin{smallmatrix}0\\1\end{smallmatrix}$	$\begin{smallmatrix}1\\0\end{smallmatrix}$	$\begin{smallmatrix}1\\1\end{smallmatrix}$
q_0	**true**	**true**	$((\Box, q_2) \vee (\Diamond, q_1)) \wedge (\Box, q_0)$	**true**
q_1	$(\Box, q_2)$	$(\Box, q_2)$	$(\Box, q_2) \vee (\Diamond, q_1)$	**true**
q_2	**false**	**true**	**false**	**true**

L'état initial est q_0. Donner un calcul de l'automate sur le dépliage de la structure de Kripke représentée sur la figure 20.

La *condition d'acceptance de Büchi* est donnée par un sous-ensemble Q_f de Q. Un calcul ρ est alors réussi si tout chemin infini de ρ contient une infinité d'éléments de Q_f.