8 Logique monadique de second ordre à k successeurs SkS

Ce paragraphe n'est qu'un rapide survol. On consultera [Tho97, GTW02] pour plus de détails.

8.1 Syntaxe et sémantique

Syntaxe On se donne un ensemble de symboles de variables de premier ordre et un ensemble de symboles de variables de second ordre (disjoints des autres alphabets).

Les *termes* de SkS constituent le plus petit ensemble tel que, si t est un terme, alors les expressions suivantes sont aussi des termes de SkS :
- ϵ
- les variables de premier ordre
- $t \cdot 1, \ldots, t \cdot k$ (aussi notés $S_1(t), \ldots, S_k(t)$)

Les formules de SkS constituent le plus petit ensemble tel que, si ϕ, ψ sont des formules et t_1, t_2 des termes, alors les expressions suivantes sont aussi des formules :
- **true**, **false**
- $t_1 \in X$ où X est une variable de second ordre
- $t_1 = t_2$, $t_1 \geq t_2$
- $\phi \vee \psi$, $\phi \wedge \psi$, $\neg \phi$
- $\forall x.\phi$, $\exists x.\phi$ où x est une variable de premier ordre
- $\exists^2 X.\phi$, $\forall^2 X.\phi$ où X est une variable de second ordre

Sémantique Le domaine d'interprétation des variables du premier ordre est $\{1, ..., k\}^*$ avec les opérations : concaténation à droite par $1, ..., k$, mot vide, ordre préfixe pour $S_1, \ldots, S_k$, ϵ, $\leq$ respectivement

Le domaine d'interprétation des variables de second ordre est l'ensemble des sous-ensembles de $\{1, ..., k\}^*$.

Les connecteurs et quantificateurs sont interpétés comme d'habitude.

Exercice 27
Exprimer en SkS les prédicats suivants :
1. Pair(X) qui est satisfait par l'ensemble des positions de longueur paire et cet ensemble seulement.
2. Chemin(X) qui est satsifait par les ensembles qui représentent des chemins infinis
3. Infini(X) satisfait par les ensembles infinis de positions
4. Origine(x, X) satisfait par les chemins X dont l'orgine est x
5. Succ(X, Y) satisfait par les chemins X et Y tels que Y est X privé de son origine
6. $\leq$ (sans utiliser $\leq$!)

Exercice 28
Codage d'un arbre quelconque à branchement fini : si un nœud possède k fils : 21, ... 2^k1. Plus précisément, $c(i \cdot w) = 2^i \cdot 1 \cdot c(w)$. On obtient ainsi une application dans l'arbre binaire, pour laquelle l'image du codage est expressible dans $S2S$ et le codage injectif. Si bien que $S_1, ..., S_n \models \phi$ ssi $c(S_1), ..., c(S_n) \models c(\phi)$.

De même que pour S1S, si $X_1, \ldots, X_n$ sont les variables libres (de second ordre) d'une formule ϕ de SkS, ϕ définit un langage (ensemble d'arbres) sur l'alphabet $\{0, 1\}^n$: un arbre (infini) de branchement k et étiqueté par $\{0, 1\}^n$ est une application de $\{1, ..., k\}^*$ dans $\{0, 1\}^n$. Un arbre T satisfait ϕ si les ensembles $S_1, ...S_n$ définis par : S_i est l'ensemble des positions de T dont l'étiquette a pour ième composante 1, satisfont ϕ. On note L_ϕ le langage ainsi défini.

Exemple 37 Soit ϕ la formule

$$\begin{array}{ll} 0 \in X \wedge \forall x. & (x \in X \Rightarrow x \notin Y \wedge x \cdot 1 \in X \wedge x \cdot 2 \notin X) \\ \wedge & (x \in Y \Rightarrow x \notin X \wedge x \cdot 1 \notin Y \wedge x \cdot 2 \notin Y) \end{array}$$

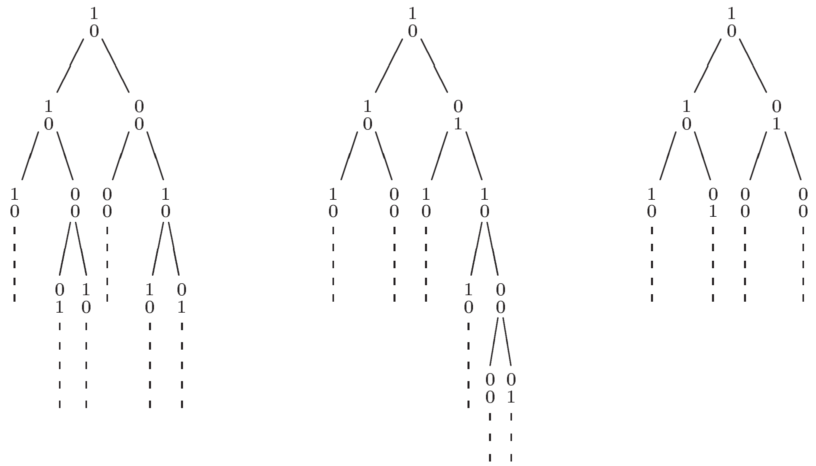

FIG. 18 – Quelques arbres binaires sur l'alphabet $\{0,1\}^2$

Les arbres de la figure 18 satisfont ϕ.

8.2 Automates d'arbres

Un automate d'arbres est un tuple (Q, q_0, A, δ, F) où Q est un ensemble fini d' états, $q_0 \in Q$ est l'état initial, A est un ensemble fini de symboles d'entrée, F est une condition d'acceptance (nous serons plus précis ultérieurement) et δ est un ensemble de règles de la forme :

$$q(f(x_1, \ldots, x_k)) \to f(q_1(x_1), \ldots, q_k(x_k))$$

Autrement dit :

$$\delta \subseteq Q \times F \times Q^k$$

Un *calcul* ρ de l'automate sur un arbre infini T (de branchement constant k) étiqueté par A est un arbre étiqueté par Q (i.e. une application de $\{1, \ldots, k\}^*$ dans Q tel que $\rho(\epsilon) = q_0$ et :

$$\left.\begin{array}{r} \rho(t) = q \\ \forall i = 1..k,\ \rho(t \cdot i) = q_i \\ T(t) = f \end{array}\right\} \Rightarrow \exists q(f(x_1, \ldots, x_k)) \to f(q_1(x_1), \ldots, q_k(x_k)) \in \delta$$

Exemple 38 Soit l'automate défini par : $Q = \{q, q'\}, q_0 = q$, $A = \{a, b\}$, et δ est donné par :

$$\begin{array}{rcl} q(a(x,y)) & \to & a(q(x), q(y)) \\ q'(a(x,y)) & \to & a(q(x), q(y)) \\ q(b(x,y)) & \to & b(q'(x), q'(y)) \\ q'(b(x,y)) & \to & b(q'(x), q'(y)) \end{array}$$

Un calcul de l'automate est représenté sur la figure 19.

La *condition d'acceptance de Büchi* est donnée par $F \subseteq Q$; un calcul de l'automate est réussi si toutes ses branches contiennent une infinité d'éléments de F. Un arbre T est accepté par l'automate $\mathcal{A}$ s'il existe au moins un calcul réussi sur T.

Exemple 39 Si l'automate de l'exemple 38 est complété par la condition d'acceptance de Büchi $F = \{q\}$, l'automate accepte les arbres dont toutes les branches contiennent une infinité de a.

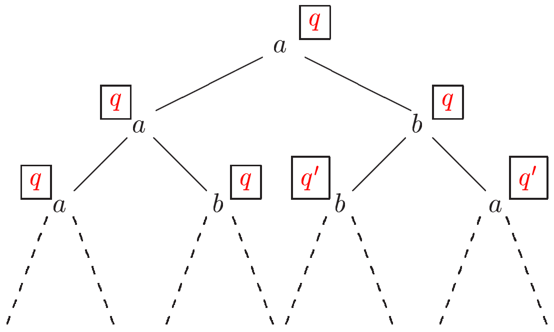

FIG. 19 – Un calcul d'automate d'arbres

Exercice 29
Donner un automate de Büchi d'arbres qui accepte les arbres binaires sur l'alphabet $\{a, b\}$ tels qu'au moins une branche contient une infinité de a.

Au contraire des automates de mots, la condition de Büchi n'est pas stable par complémentaire :

Exercice 30
Montrer que le complémentaire du langage de l'exercice 29 n'est reconnu par aucun automate de Büchi d'arbres.

La *condition d'acceptance de Rabin* est donnée par un ensemble fini de paires de sous ensembles de Q : $F = \{(L_1, U_1), \ldots, (L_n, U_n)\}$. Un calcul ρ sur T est *réussi* si, pour tout chemin π de ρ, il existe un indice i tel que π ne contienne qu'un nombre fini d'éléments de L_i et un nombre infini d'éléments de U_i.

Exercice 31
Montrer que le langage de l'exercice précédent, c'est-à-dire l'ensemble des arbres dont tous les chemins ne contiennent qu'un nombre fini de a est accepté par l'automate de l'exemple 38 avec avec une condition d'acceptance de Rabin que l'on précisera.

La *condition de parité* est donnée par une application r de l'ensemble des états dans les entiers. Un arbre t est accepté s'il existe un calcul ρ sur t tel que, pour tout chemin infini π de ρ, le minimum des $r(q)$ pour les états q apparaissant infiniment souvent dans π, est pair.

Exercice 32
Construire un automate à parité acceptant le complémentaire du langage de l'exercice 29.

Les conditions d'acceptance de Rabin et de parité sont équivalentes du point de vue de l'expressivité (voir par exemple **??**).

Exercice 33
Montrer que la condition de *chaine de Rabin* pour accepter un arbre est équivalente, du point de vue expressivité, à la condition de parité. La condition de chaine de Rabin est donnée par une suite de sous-ensembles des états $E_1 \subset F_1 \subset E_2 \cdots \subset F_n$. Un calcul ρ est réussi si, pour tout chemin π de ρ, il existe un indice i tel que π contienne un nombre fini d'éléments de E_i et un nombre infini d'éléments de F_i.

8.3 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 est dans NP $\cap$ co-NP.*

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 :

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

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 36. □

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

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 sur un damier arborescent. Ensuite, faut montrer, comme pour S1S, comment construire un automate à parité acceptant les modèles de chacune des formules de base. Enfin, étant donné un automate à parité, il faut construire une formule de SkS dont l'ensemble des modèles est exactement l'ensemble des arbres acceptés par l'automate.

Exercice 34
Construire des automates à parité qui acceptent respectivement les modèles de $X \subseteq Y$, $X = S_1(Y)$.

Exercice 35
Soit $\mathcal{A}$ un automate à parité sur l'alphabet $\{0, 1\}$, montrer que l'ensemble des arbres (binaires) acceptés est l'ensemble des modèles d'une formule de S2S.

Comme le vide est décidable pour les automates à parité, on obtient en corollaire :

Corollaire 42 *SkS est décidable.*