université PARIS-SACLAY

Automates et Applications
M1–MPRI Orsay

TD n° 3

1 Logiques d'arbres

1. Définir les concepts suivants en MSO (ou FO, les quantifications du second ordre n'étant pas toujours nécessaires). On supposeras que l'on est sur un alphabet $\Sigma = \{f^2, g^2, a, b\}$.
   (a) Un prédicat $racine(x)$ valant vrai si x est la racine de l'arbre et faux sinon
   (b) Le langage des arbres ne contenant que des nœuds internes f et des feuilles a.
   (c) Le langage des arbres où tout chemin contient au moins deux f.
   (d) Le langage des arbres ayant exactement un a dans leurs feuilles.

   Il pourra être utile de définir des prédicats auxiliaires et de les utiliser comme des « macros »

Réponse:

(a) La racine est le nœud n'ayant pas de parent, *i.e.* qui n'est le fils de personne : $racine(x) = \forall z.\neg(\mathsf{child}_1(z,x) \vee \mathsf{child}_2(z,x))$

(b) $\forall x.\mathsf{lab}_a(x) \vee \mathsf{lab}_f(x)$

(c) On définit le prédicat auxiliaire $anc(x,Y)$ qui dit que Y est l'ensemble des ancêtres de x :

$$anc(x,y) \equiv \forall y.y \in Y \Leftrightarrow x = y \vee (\exists z.\mathsf{child}_1(y,z) \vee \mathsf{child}_2(y,z))$$

On peut ensuite définir le langage demandé. Les chemins sont les ancêtres des feuilles :

$$\forall x.\mathsf{lab}_a(x) \vee \mathsf{lab}_b(x) \Rightarrow (\exists Y.anc(x,Y) \wedge \exists z_1.\exists z_2.\mathsf{lab}_f(z_1) \wedge \mathsf{lab}_f(z_2) \wedge z_1 \in Y \wedge z_2 \in Y \wedge \neg(z_1 = z_2))$$

(d) On demande qu'un tel nœud existe et que tout autre étiqueté a soit le même nœud :

$$\exists x.\mathsf{lab}_a(x) \wedge \forall y.\mathsf{lab}_a(y) \Rightarrow x = y$$

2. Pour les cas (b), (c), (d) de l'exercice 1, donner un automate d'arbre reconnaissant le langage. L'automate devra être déterministe descendant lorque cela est possible.

Réponse:

(b) On peut produire un automate descendant déterministe à un seul état (un automate complet aurait un second état servant de puits).

$$\begin{array}{lcl} q_0, f & \to & (q_0, q_0) \\ q_0, a & \to & \{\} \end{array}$$

(c) On peut ici encore avoir un automate descendant déterministe. Les états mémorisent le nombre de f traversés pendant la descente :

$$\begin{array}{lcllcl} q_0, f & \to & (q_1, q_1) & q_2, f & \to & (q_2, q_2) \\ q_0, g & \to & (q_0, q_0) & q_2, g & \to & (q_2, q_2) \\ q_1, f & \to & (q_2, q_2) & q_2, a & \to & \{\} \\ q_1, g & \to & (q_1, q_1) & q_2, b & \to & \{\} \end{array}$$

Remarque : une propriété sur les chemins est typiquement quelque chose de reconnaissable par un automate descendant déterministe et il est dans ce cas, plus « naturel » qu'un automate ascendant.

(d) Au contraire, nous avons ici un langage clairement top-down **non déterministe** ou bottom-up. En effet, si on imagine un automate descendant, ce dernier doit faire un choix :
— s'il y a un a dans mon fils gauche, il ne faut pas qu'il y en ait dans le fils droit ou ;
— s'il n'y a pas de a dans le fils gauche, il faut en trouver un dans le fils droit.
On propose l'automate ascendant suivant $(\{q_0, q_1\}, \Sigma, \delta, \{q_1\})$

$$\begin{array}{lcl} a & \to & q_1 \\ b & \to & q_0 \\ f(q_1, q_0) & \to & q_1 \\ f(q_0, q_1) & \to & q_1 \\ g(q_1, q_0) & \to & q_1 \\ g(q_0, q_1) & \to & q_1 \end{array}$$

Cette automate fonctionne car il est incomplet. En particulier il n'y a pas de transition pour $f(q_1, q_1)$ ou $g(q_1, q_1)$ qui sont des arbres contenant deux a. On peut évidemment le compléter avec un état puits q_2 qui serait la destination de ces transitions.

3. On considère l'alphabet $\{a^2, b^2, c^2, d^2, \#\}$. Donner les prédicats suivants :
   — $P(x, y)$ il existe un chemin entre le nœud x et le nœud y passant par des nœuds de symbole a, b, c, d dans cet ordre (i.e. il y a 4 nœuds sur le chemin entre x et y)
   — $Q(x, Y)$ Y est l'ensemble de tous les nœuds séparés de x par un chemin comme celui de la question (a)
   — $R(x, y)$ il existe un unique chemin comme dans la question (a)

Réponse:

4. La formule peut simplement utiliser deux prédicats auxiliaires :

$$\begin{array}{lcl} child(x, y) & = & \mathsf{child}_1(x, y) \vee \mathsf{child}_2(x, y) \\ child_a(x, y) & = & child(x, y) \wedge \mathsf{lab}_a(y) \end{array}$$

   On peut ensuite poser :

$$P(x, y) = \exists z_1.\exists z_2.\exists z_3.\exists z_4.child_a(x, z_1) \wedge child_b(z_1, z_2) \wedge child_c(z_2, z_3) \wedge child_d(z_3, z_4) \wedge child(z_4, y)$$

5. Évidemment on a intéret à utiliser $P(x, y)$ comme une notation :

$$Q(x, Y) = \forall y.y \in Y \Leftrightarrow P(x, y)$$

   Attention, l'équivalence est importante. Si on ne la met pas, Y représente **un** ensemble de nœuds y, pas forcément celui de tous ces nœuds.

6. On rajoute une contrainte d'unicité :

$$R(x, y) = P(x, y) \wedge \exists z.P(x, z) \Rightarrow z = y$$