Automates avancés

Examen du 19 mai 2009 - corrigé

1 Applications des cours

1.1 Sans étoiles

Pour chacun des langages ci-dessous sur l'alphabet $\Sigma = \{a, b\}$ l'exprimer en logique $FO(<, \Sigma)$ ou démontrer que c'est impossible.

1. [1] $L_1 = \{a^n b a^n \mid n \in \mathbb{N}\}$
   Solution.
   Ce langage n'est pas régulier comme on connaît des cours de la licence (facile à démontrer par pompage). Par conséquent il n'est pas exprimable en FO (ni même en MSO).
2. [2] $L_2 = (b^+ a)^*$ (on rappelle que $b^+ = bb^*$).
   Solution.
   L_3 contient les mots qui satisfont 3 contraintes :
   – ils commencent par un b ;
   – ils terminent par un a ;
   – ils ne contiennent pas deux a consécutifs.
   On traduit les trois contraintes en FO :

   $$b(0) \wedge \forall y(\mathbf{fin}(y+1) \Rightarrow a(y)) \wedge \forall z(a(z) \Rightarrow \neg a(z+1))$$

   (j'ai utilisé les opérations $+1$ et **fin** pour une meilleure lisibilité, mais comme on a vu en cours on peut s'en débarrasser).
3. [3] $L_3 = a(baba)^* a$.
   Solution.
   Soit $u = a; v = ba; w = a$. Il est clair que $uv^k w$ est dans le langage pour les k pairs, et hors langage pour les k impairs. Donc L_3 n'est pas apériodique. Par conséquent il est impossible de l'exprimer en FO.

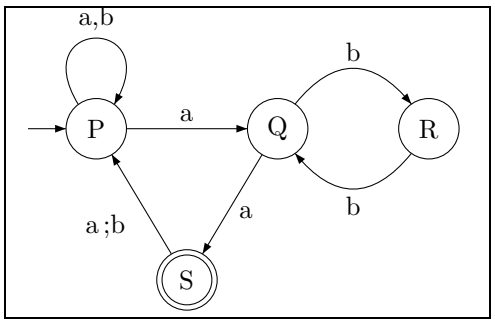
1.2 Un langage de mots infinis

On considère le langage L de tous les mots infinis sur $\{a, b\}$ qui contiennent une infinité de fois les motifs $ab^{2n}a$ (on ne demande pas que n soit le même).

1. [1] Trouvez une expression ω-régulière définissant L.
   Solution.

   $$\big((a+b)^* a (bb)^* a\big)^\omega$$

2. [1] Trouvez un automate de Büchi reconnaissant L.
   Solution.



3. [2] Trouvez une formule MSO définissant L.

Solution.
On traduit l'automate en expression MSO, comme décrit en cours

$$\exists P, Q, R, S\Big($$
$$\textit{on est toujours dans 1 état et 1 seul}$$
$$\forall x\big(P(x) \vee Q(x) \vee R(x) \vee S(x) \wedge$$
$$(P(x) \Rightarrow \neg Q(x) \wedge \neg R(x) \wedge S(x)) \wedge (Q(x) \Rightarrow \dots) \wedge \dots\big)$$
$$\textit{on commence dans } P$$
$$P(0) \wedge$$
$$\textit{on respecte la relation de transition}$$
$$\forall x\big((P(x) \wedge P(x+1)) \vee (P(x) \wedge a(x) \wedge Q(x+1)) \vee$$
$$(Q(x) \wedge a(x) \wedge S(x+1)) \vee (S(x) \wedge P(x+1)) \vee$$
$$(Q(x) \wedge b(x) \wedge R(x+1)) \vee (R(x) \wedge b(x) \wedge Q(x+1))\big) \wedge$$
$$\textit{et on accepte}$$
$$\forall x \exists y(y > x \wedge S(y))$$
$$\Big)$$

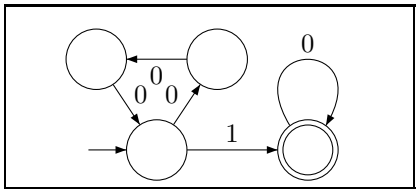
2 Méthodes génériques pour les résultats intéressants

2.1 On étend Presburger

1. [1] On enrichit l'arithmétique de Presburger en y ajoutant le prédicat P8 tel que $P8(x)$ est vrai si et seulement si x est une puissance de 8. Est-ce que la théorie obtenue est décidable ? Essayez d'étendre la preuve du cours (sans la répéter).

   Solution.

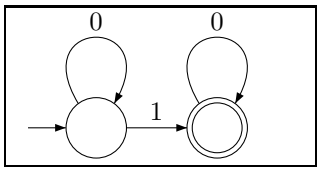
   Dans le cours on décide l'arithmétique de Presburger en traduisant chaque formule de cette logique en automate. Pour appliquer la même méthode à l'arithmétique de Presburger + P8 il suffit de construire l'automate pour la relation $P8(x)$ (avec le codage usuel binaire little-endian avec des 0 qui trainent). L'expression régulière pour ce langage est (000^*10^*). Voici l'automate :



2. [3] Même question pour le prédicat P10 tel que $P10(x)$ est vrai si et seulement si x est une puissance de 10.

   Solution.

   On peut refaire la preuve du cours en utilisant le codage **décimal**. Dans ce codage la relation $P10(x)$ correspond au langage régulier 0^*10^*, accepté par l'automate :



2.2 On compte comme dans le partiel [4]

Montrer qu'il existe un langage fini sur $\Sigma = \{a; b; c; d\}$ tel que
- tous les mots de G contiennent < 100 lettres ;
- ET toute expression régulière qui définit G contient > 1000000 symboles.

Solution.

Il y a $m > 4^{99}$ de mots qui contiennent < 100 lettres. Il y a $n = 2^m$ langages (sous-ensembles) de ces mots. On peut estimer m et n comme ceci :

$$m > 4^{99} = 2^{198} > (2^{10})^{19} > (10^3)^{19} = 10^{190}; \quad n > 2^{10^{190}}.$$

Chaque expression régulière de $\leq 10^6$ symboles peut être représentée comme une séquence de 10^6 symboles suivants : $a, b, c, d, \varepsilon, \emptyset, +, \cdot, *, (,), \sqcup$ (le dernier symbole sert à compléter l'expression à la fin jusqu à un million de lettres). Il existe $k = 12^{10^6}$ telle séquences (et moins que ça expressions régulières). On estime k comme ceci :

$$k = 12^{10^6} < 16^{10^6} = 2^{4 \cdot 10^6} < 2^{10^7}.$$

On constate que le nombre d'expressions k est beaucoup inférieur au nombre de langages n. Par conséquent, il existe un langage G (de mots de < 100 lettres) pour lequel il n'existe aucune expression de $\leq 10^6$ symboles.

2.3 On compile [10]

On considère le fragment $\mathcal{L}$ de LTL sans Until, qui contient des formules définies comme suit :

$$F ::== \mathbf{true} | \mathbf{false} | a | \circ F | \diamond F | \Box F | F \wedge F | F \vee F | \neg F$$

On cherche à démontrer le résultat suivant :

Théorème 1 *Chaque formule de la logique $\mathcal{L}$ (interprétée sur les mots finis) est équivalente à une expression sans étoile.*

La démonstration doit contenir un algorithme de traduction. On n'utilisera pas les résultats énoncées en cours sans preuve.

1. Certaines opérations de la logique $\mathcal{L}$ peuvent être exprimées en utilisant autres opérations. On peut donc les supprimer sans changer le pouvoir expressif de la logique. Expliquez quelles sont les opérations inutiles, pourquoi, et donnez une grammaire pour la logique simplifiée $\mathcal{L}'$.
2. Donnez le plan de démonstration du théorème par induction structurelle sur la grammaire de $\mathcal{L}$ ou $\mathcal{L}'$ (une liste de lemmes à prouver).
3. Faites le cas de base. Indication: N'oubliez pas que la formule de LTL a signifie que le mot commence par a

4. Faites le cas inductif pour les opérateurs booléens.
5. Faites le cas inductif pour les opérateurs temporels.
6. Terminez la preuve.
7. J'ai omis de définir la sémantique de cette logique - faites le, sans oublier que les mots sont finis.
8. Testez votre méthode de traduction sur $\diamond\Box b$

Solution.

1. On peut éliminer les opérations $\mathbf{true}, \mathbf{false}, \Box, \wedge$ comme suit :

$$\mathbf{true} \equiv a \vee \neg a; \quad \mathbf{false} \equiv \neg\mathbf{true}; \quad \Box f \equiv \neg \diamond \neg f; \quad f \wedge g \equiv \neg(\neg f \vee \neg g)$$

   Donc la logique $\mathcal{L}'$ sans ces 4 opérations a le même pouvoir expressif que $\mathcal{L}$. La définition d'une formule de $\mathcal{L}'$ suit :

$$F ::== a| \circ F| \diamond F|\neg F|F \vee F$$

2. Pour être moins verbeux on dira dans la suite "formule" (ou juste f, g) pour une formule de $\mathcal{L}'$, $[f]$ pour la sémantique d'une formule f (le langage défini par cette formule). On dira "expression" pour une expression sans étoile.

   Pour démontrer le théorème par induction structurelle (pour la logique $\mathcal{L}'$) il suffit de démontrer les lemmes suivantes :

   A Le langage $[a]$ peut être défini par une expression.

   B Si le langage $[f]$ peut être défini par une expression e, alors le langage $[\circ f]$ peut aussi être défini par une expression.

   C Même chose pour $\diamond f$.

   D Même chose pour $\neg f$.

   E Si deux langages $[f_1]$ et $[f_2]$ peuvent être défini par les expressions e_1 et e_2, alors le langage $[f_1 \vee f_2]$ peut aussi être défini par une expression.
3. Cas de base A. L'expression pour $[a]$ est $a \cdot U$.
4. Cas inductif booléen D. L'expression pour $[\neg f]$ est $\overline{e}$.
   Cas inductif booléen E. L'expression pour $[f_1 \vee f_2]$ est $e_1 + e_2$.
5. Cas inductif temporel B. L'expression pour $[\circ f]$ est $\Sigma \cdot e$, où Σ est juste la somme de toute les lettres de l'alphabet.
   Cas inductif temporel C. L'expression pour $[\diamond f]$ est $U \cdot e$.
6. Par induction structurelle on déduit que pour chaque formule f de $\mathcal{L}'$, sa sémantique $[f]$ est définissable par une expression sans étoile.
7. Rien de plus simple - ça ressemble beaucoup aux points 3-6. :

$$\begin{aligned} [a] &= a \cdot \Sigma^* \\ [\circ f] &= \Sigma \cdot [f] \\ [\diamond f] &= \Sigma^* \cdot [f] \\ [\neg f] &= \overline{[f]} \\ [f_1 \vee f_2] &= [f_1] \cup [f_2] \end{aligned}$$

8. Tout d'abord on traduit en $\mathcal{L}'$ en éliminant le $\Box$:

$$\diamond\Box b \equiv \diamond\neg \diamond \neg b$$

Ensuite, en partant de l'intérieur on traduit chacune des sous-formules en expréssion :

$$\begin{aligned} b &\equiv b \cdot U \\ \neg b &\equiv \overline{b \cdot U} \\ \diamond \neg b &\equiv U \cdot \overline{b \cdot U} \\ \neg \diamond \neg b &\equiv \overline{U \cdot \overline{b \cdot U}} \\ \diamond \neg \diamond \neg b &\equiv U \cdot \overline{U \cdot \overline{b \cdot U}} \end{aligned}$$

On obtient la réponse $U \cdot \overline{U \cdot \overline{b \cdot U}}$.

3 Exercices subsidiaires difficiles

Si vous avez tout fait, et il vous reste beaucoup de temps, vous pouvez réfléchir aux questions suivantes pour lesquelles je ne connais pas de solution simple :

1. Question du partiel : comment démontrer que le langage qui contient les puissances de 3 en base 2 n'est pas régulier ?
2. Comment démontrer que l' arithmétique de Presburger étendue à la fois par P8 et P10 (de l'exo 2.1) est indécidable ?
3. Comment étendre la traduction de 3.2 aux formules avec Until ?

Solution.
Je ne connais pas de solution simple