Examen de démonstration automatique 2001

Correction.

Documents autorisés (en particulier le poly).

I. Automates alternants généralisés

On considère une signature du premier ordre consistant en un ensemble fini arbitraire de symboles de fonction f, g, ..., et un ensemble fini de symboles de prédicats *unaires*.

On appelle *clause automatique* toute clause de la forme :

$$\neg P_1(t_1) \vee \ldots \vee \neg P_m(t_m) \vee P_{m+1}(t_{m+1}) \vee \ldots \vee P_n(t_n)$$

où $0 \leq m \leq n$, les t_i ($1 \leq i \leq n$) sont des termes tels que :

1. si tous les t_i sont des variables, alors ils sont tous la même variable x;
2. sinon, tous les t_i non variables sont identiques, de la forme $f(x_1, \ldots, x_p)$, où $\{x_1, \ldots, x_p\}$ contient toutes les variables libres de la clause.

Dans le premier cas, on dit que la clause est un ϵ-*bloc*.

L'intuition est la suivante. Chaque prédicat P correspond à un état d'un automate d'arbres classique. Une transition $f(P_1, \ldots, P_k) \to P$ (exprimant que si t_1 est reconnu en P_1, ..., t_k est reconnu en P_k, alors $f(t_1, \ldots, t_k)$ est reconnu en P) est codée par la clause:

$$\neg P_1(x_1) \vee \ldots \vee \neg P_n(x_n) \vee P(f(x_1, \ldots, x_n))$$

On peut coder les ϵ-transitions $P \stackrel{\epsilon}{\longrightarrow} Q$ ("si t est reconnu en P, alors t est reconnu en Q") par:

$$\neg P(x) \vee Q(x)$$

Dans les automates usuels, un terme t est reconnu en P si et seulement s'il *existe* une transition aboutissant à P telle qu'un certain nombre de conditions soient vérifiées : la sémantique de la reconnaissance est une disjonction. Ici, on peut aussi coder les conjonctions $P_1 \wedge \ldots \wedge P_n \to P$ ("si t est reconnu en P_1 et ... et en P_n, alors il est reconnu en P"):

$$\neg P_1(x) \vee \ldots \vee \neg P_n(x) \vee P(x)$$

On peut aussi coder les négations ("P reconnaît exactement les termes non reconnus en Q") :

$$\neg P(x) \vee \neg Q(x)$$
$$P(x) \vee Q(x)$$

ainsi que les tests d'égalité entre frères ("si les t_i sont reconnus en P_i et $t_1 = t_2$ alors $f(t_1, \ldots, t_n)$ est reconnu en P"):

$$\neg P_1(x_1) \vee \neg P_2(x_1) \vee \neg P_3(x_3) \vee \ldots \vee \neg P_n(x_n) \vee P(f(x_1, x_1, x_3, \ldots, x_n))$$

1. Montrer qu'il existe un ordre strict $>$ sur les atomes tel que (i) $>$ est stable, autrement dit pour toute substitution σ, si $A > B$ alors $A\sigma \gg B\sigma$; et (ii) pour tous P, Q, f, x_1, …, x_n, on a $P(f(x_1, \ldots, x_n)) > Q(x_i)$, $1 \leq i \leq n$. Dans la suite, on considérera la résolution ordonnée au-dessus de $>$.

   (Il fallait évidemment lire $>$ au lieu de $\gg$ dans l'énoncé.)

   On peut par exemple prendre un ordre récursif sur les chemins, fondé sur une priorité qui considère tous les symboles de prédicats comme équivalents. Mais il y avait plein d'autres possibilités, l'une des plus simples étant de prendre l'ordre de sous-terme strict $\succ$ et de déclarer que $P(s) > Q(t)$ si et seulement si $s \succ t$.

2. Montrer que tout facteur d'une clause automatique est une clause automatique, dès que $>$ vérifie la condition (ii).

   Soit C une clause automatique comme décrite dans l'énoncé. Supposons que C soit (sans perte de généralité) de la forme $C_0 \vee P(t_1) \vee P(t_2)$, avec $\sigma = mgu(t_1, t_2)$. (Le cas $C_0 \vee \neg P(t_1) \vee \neg P(t_2)$ est symétrique.)

   Si C est une clause de la forme 1, alors $t_1 = t_2 = x$, et donc le facteur $C_0 \vee P(x)$ est encore de la forme 1.

   Si C est de la forme 2, on a plusieurs cas. Si t_1 et t_2 sont des variables x et x', alors $\sigma = \{x := x'\}$, et C_0 contient au moins un atome de la forme $f(x_1, \ldots, x_p)$ contenant toutes les variables libres de C: il est clair que $C\sigma$ est de la forme 2. Si t_1 ou t_2 n'est pas une variable, supposons sans perte de généralité t_1 non variable; donc $t_1 = f(x_1, \ldots, x_p)$. Comme $f(x_1, \ldots, x_p)$ contient toutes les variables libres de C, t_2 ne peut pas être une variable, sinon ce serait l'une de x_1, …, x_p, et t_1 et t_2 ne seraient donc pas unifiables. Donc $t_1 = t_2 = f(x_1, \ldots, x_p)$. Donc $C = C_0 \vee P(t_1)$ est encore de la forme 2.

3. Montrer que tout résolvant binaire ordonné de deux clauses automatiques est une clause automatique ou une disjonction d'ϵ-blocs.

   Soient C et C' deux clauses automatiques, et supposons que $C = C_0 \vee P(t)$, $C' = C'_0 \vee \neg P(t')$, où le résolvant est $(C_0 \vee C'_0)\sigma$ avec $\sigma = mgu(t, t')$.

   Si C et C' sont de la forme 1, alors t est l'unique variable x de C et t' est l'unique variable x' de C', donc σ est $[x := x']$ (ou $[x' := x]$), et il est clair que le résolvant $C_0[x := x'] \vee C'_0$ est encore de la forme 1.

   Si C est de la forme 2, alors par (ii) comme $P(t)$ est maximal, t est de la forme $f(x_1, \ldots, x_p)$.

   Si C' est de la forme 1, alors t' est l'unique variable x' de C', et donc le résolvant est $C_0 \vee C'_0[x' := f(x_1, \ldots, x_p)]$, qui est clairement de la forme 2 si C'_0 est une clause non vide, et qui sinon est de la forme 1.

   Sinon, C' est de la forme 2, et par (ii) encore, t' est donc de la forme $f(x'_1, \ldots, x'_p)$. Le mgu σ envoie donc des variables vers des variables. Le résolvant $(C_0 \vee C'_0)\sigma$ est donc une disjonction de littéraux de la forme $P(x)$ où x est une variable ou $P(f(x_1, \ldots, x_p))$, ou bien leurs négations. Distinguons deux cas:

   - *C_0 contient un littéral de la forme $Q(f(y_1, \ldots, y_p))$ (ou sa négation; le cas où C'_0 en contient un aussi est similaire). Noter que ceci implique que $f(y_1, \ldots, y_p) = f(x_1, \ldots, x_p)\sigma$, puisque σ envoie des variables vers des variables.*

     On va montrer que $(C_0 \vee C'_0)\sigma$ est de la forme 2; ceci revient à montrer que tous les littéraux $Q'(t)$ du résolvant où t est non variable sont tels que $t = f(y_1, \ldots, y_p)$, et que y_1, …, y_p contient toutes les variables libres du résolvant. Or ces littéraux $Q'(t)$ viennent soit de C_0, auquel cas ils sont égaux à $f(y_1, \ldots, y_p)$ parce qu'instances par σ d'un littéral qui ne peut être que $f(x_1, \ldots, x_p)$; soit de C'_0, auquel cas ils sont de la forme $f(x'_1, \ldots, x'_p)\sigma = f(x_1, \ldots, x_p)\sigma$ (puisque σ est le mgu de t et t') $= f(y_1, \ldots, y_p)$.

- *C_0 ne contient que des littéraux de la forme $Q(x)$ où x est une variable. Mais alors C_0 est une disjonction d'ϵ-blocs. Par exemple, le résolvant ordonné de $\neg P_1(x_1) \vee P(f(x_1))$ et de $\neg P(f(y_1)) \vee Q_1(y_1)$ est $\neg P_1(x_1) \vee Q_1(x_1)$, qui est un ϵ-bloc, et celui de $\neg P_1(x_1) \vee \neg P_2(x_2) \vee P(f(x_1, x_2))$ et de $\neg P(f(y_1, y_2)) \vee Q_1(y_1)$ est $\neg P_1(x_1) \vee \neg P_2(x_2) \vee Q_1(y_1)$, qui est la disjonction des ϵ-blocs $\neg P_1(x_1) \vee Q_1(y_1)$ et $\neg P_2(x_2)$.*

4. Montrer qu'à renommage près, il n'existe qu'un nombre fini de clauses automatiques. On estimera un majorant de ce nombre en fonction du nombre n de symboles de prédicats, m de symboles de fonctions, et l'arité maximum k des symboles de fonctions. (Attention ! Les symboles de fonctions sont appliqués à des variables qui ne sont pas nécessairement disjointes deux à deux.)

   À renommage près, il y a au plus 2^{2n} clauses de type 1 (sous-ensemble de l'ensemble des $P(x)$ et des $\neg P(x)$, de cardinal $2n$).

   Quant aux clauses de type 2, on a m possibilités pour le symbole f. Ce symbole peut être appliqué à au plus k variables, donc la clause a au plus k variables libres, numérotons-les $1, \ldots, k$. Les arguments de f sont des variables prises parmi ces k, donc l'unique terme $f(x_1, \ldots, x_p)$ est pris parmi mk^k. Une fois ce choix fait, la clause contient au plus $2n$ littéraux de la forme $P(f(x_1, \ldots, x_p))$ ou $\neg P(f(x_1, \ldots, x_p))$ ($mk^k 2^{2n}$ possibilités). Le reste de la clause est un sous-ensemble de littéraux de la forme $P(x_i)$ ou $\neg P(x_i)$, ce qui donne 2^{2nk} possibilités. On a donc au plus $mk^k 2^{2n(k+1)}$ clauses de type 2.

5. En déduire que, si tous les symboles de fonction de la signature sont d'arité au plus un, alors la satisfiabilité de tout ensemble de clauses automatiques est décidable.

   Si tous les symboles sont d'arité au plus un, le raisonnement de la question 3 montre que tout résolvant ordonné de clauses automatiques est encore une clause automatique, car les disjonctions d'ϵ-blocs obtenues ne contiennent qu'une variable. Comme il n'y a qu'un nombre fini de clauses automatiques modulo renommage, la résolution ordonnée s'arrête. Si la clause vide est produite, alors l'ensemble de clauses est satisfiable, sinon il est insatisfiable parce que la résolution ordonnée est complète.

6. Que peut-on dire lorsqu'il existe des symboles de fonction d'arité ≥ 2 ? (Indication : il s'agit de traiter le cas problématique des disjonctions non-triviales d'ϵ-blocs; on pourra penser à importer une règle de ramification des méthodes de tableaux.)

   On ajoute la règle suivante, qui est essentiellement une règle de tableaux :

   $$\frac{C_1 \vee C_2}{C_1 \quad | \quad C_2}\,(\vee)$$

   où C_1 et C_2 n'ont aucune variable libre en commun.

   En clair, on développe un tableau dans lequel chaque branche est un ensemble de clauses, où l'on utilise les règles de résolution binaire et de factorisation ordonnées comme des règles de prolongation (qui ajoutent des clauses à la branche courante) et la règle $(\vee)$ comme une règle de ramification qui remplace $C_1 \vee C_2$ par C_1 sur une branche, par C_2 sur l'autre.

   La règle $(\vee)$ est correcte, au sens ou tout modèle de $C_1 \vee C_2$ (quantifiée universellement) satisfait soit la clôture universelle de C_1 soit celle de C_2. Disons qu'une branche est close *si elle contient la clause vide, et qu'un tableau est* clos *si toutes ses branches sont closes. Alors tout tableau T développé à partir d'une branche S (ensemble de clauses) est tel que, si T est clos, alors S est insatisfiable.*

   La méthode de tableaux obtenue (résolution binaire ordonnée + factorisation ordonnée + $(\vee)$) est complète, d'autre part, parce que la résolution binaire + factorisation ordonnée l'est déjà.

   Pour assurer la terminaison dans le cas des clauses automatiques, il n'y a plus qu'à montrer que l'on peut appliquer $(\vee)$ en priorité par rapport aux autres règles. Autrement dit, dès que la règle de résolution binaire engendre une disjonction non-triviale $C_1 \vee \ldots \vee C_n$ ($n \geq 2$) de ϵ-blocs, on découpe immédiatement la branche courante en n branches où le résolvant est remplacé par C_i pour *chaque i, $1 \leq i \leq n$. Alors chaque branche est finie par la question 4, et par le lemme de König le tableau maximalement expansé est fini.*

Pour montrer qu'on peut appliquer $(\vee)$ en priorité sans détruire la complétude, on peut montrer qu'on peut les permuter par rapport aux règles de résolution binaire et factorisation ordonnées, ce qui aura l'avantage de montrer que $(\vee)$ peut être appliqué en priorité même en-dehors du cadre des clauses automatiques (exercice facile mais fastidieux). Sinon on observe que toutes les règles R du calcul sont inversibles, au sens où une branche est insatisfiable si et seulement si toutes les branches obtenues après l'application de R sont insatisfiables. Comme dans le cas des clauses automatiques l'expansion des tableaux termine, si on part d'un S insatisfiable, on aboutira en temps fini à un tableau maximalement expansé T, dont toutes les branches sont insatisfiables. (Ceci étant vrai si l'on applique $(\vee)$ en priorité notamment.) Ce tableau maximalement expansé a la propriété que si on applique n'importe quelle règle R à n'importe quelle branche B (qui ne contient que des clauses automatiques, donc $(\vee)$ n'est pas applicable), on fabrique des clauses qui sont soit déjà dans B soit sont subsumées par des clauses de B (c'est le cas lorsqu'on résout deux clauses automatiques pour obtenir une disjonction non-triviale C d'ϵ-bloc, car l'un des ϵ-blocs est nécessairement déjà dans B, et il subsume C). Comme B est insatisfiable, on en conclut immédiatement qu'elle doit contenir la clause vide, par la complétude de la résolution ordonnée. Donc T est clos.

II. À propos de S2S

On définit en $\mathcal{A}_{S2S}$ une abréviation pour dire que des ensembles sont disjoints entre eux :

$$\textit{disjoint}(X_1, \ldots, X_n) \quad :\Leftrightarrow \quad \bigwedge_{1 \leq i < j \leq n} (\neg \exists y (y \in X_i \wedge y \in X_j))$$

1. Pour quelles valeurs de x est-ce que la formule suivante est vraie dans la structure $\mathcal{A}_{S2S}$?

$$\begin{aligned} \exists X_0, X_1, X_\perp \Big(\quad & \textit{disjoint}(X_0, X_1, X_\perp)) \\ & \wedge \epsilon \in X_0 \\ & \wedge x \in X_0 \\ & \wedge \forall y (y \in X_0 \to y0 \in X_1 \wedge y1 \in X_\perp) \\ & \wedge \forall y (y \in X_1 \to y0 \in X_0 \wedge y1 \in X_\perp) \\ & \wedge \forall y (y \in X_\perp \to y0 \in X_\perp \wedge y1 \in X_\perp) \quad \Big) \end{aligned}$$

Les trois dernières lignes font penser à un automate à trois états X_0, X_1 et $X_\perp$. Les X_0, X_1, $X_\perp$ minimaux qui vérifient toutes les lignes sauf la première et la troisième sont les langages $(00)^*$ (une suite d'un nombre pair de zéros), $(00)^*0$ (un nombre impair de zéros), 0^*1A^* respectivement, où $A = \{0, 1\}$ (tout mot contenant au moins un 1).

Il est alors clair que tous les x dans $(00)^*$ vérifient la formule. (En particulier, $(00)^*$, $(00)^*0$, 0^*1A^* sont bien disjoints.) Réciproquement, notons que l'automate est déterministe et complet, autrement dit les langages $(00)^*$, $(00)^*0$, et 0^*1A^* sont non seulement disjoints mais encore leur union est exactement A^*. En particulier, si x n'est pas dans $(00)^*$ alors il est soit dans $(00)^*0$ soit dans 0^*1A^*. Dans le deux cas, montrons que x ne vérifie pas la formule. Supposons par contradiction que la formule est vraie, en particulier on a trouvé X_0, X_1 et $X_\perp$ et $x \in X_0$:

- *si $x \in (00)^*0$ alors comme $x \in X_0$, $x0 \in X_1$; mais $x0 \in (00)^*00 \subseteq X_0$, donc X_0 et X_1 ne seraient pas disjoints;*
- *si $x \in 0^*1A^*$ alors comme $x \in X_0$, $x0 \in X_1$; mais $x1 \in 0^*1A^*0 \subseteq X_\perp$, contredisant le fait que X_1 et $X_\perp$ sont disjoints.*

2. Montrer le théorème suivant : pour tout langage régulier L sur l'alphabet $\{0,1\}$ il existe une formule $F_L(x)$ du langage de la structure $\mathcal{A}_{S2S}$ telle que pour tout mot $w \in \{0,1\}^*$:

$$\mathcal{A}_{S2S}, [x \mapsto w] \models F_L(x) \qquad \Leftrightarrow \qquad w \in L$$

Soit $\mathcal{A}$ un automate déterministe complet (pour toute lettre et tout état, il existe exactement une transition sortant de l'état étiquetée par la lettre) reconnaissant L. Pour chaque état q, créer une variable d'ensemble X_q. Soit Q l'ensemble des états, q_0 l'état initial, et F l'ensemble des états finaux de $\mathcal{A}$. On crée alors la formule :

$$\begin{aligned}\exists (X_q)_{q\in Q} \Big(\quad & \textit{disjoint}((X_q)_{q\in Q}) \\ & \wedge \epsilon \in X_{q_0} \\ & \wedge \bigvee_{q \in F} x \in X_q \\ & \wedge \bigwedge_{q \xrightarrow{a} q'} \forall y (y \in X_q \to ya \in X_{q'}) \quad \Big)\end{aligned}$$

La démonstration de correction est essentiellement la même que dans la question précédente : les ensembles X_q minimaux vérifiant la deuxième et la quatrième ligne sont exactement les langages reconnus en q dans $\mathcal{A}$. Comme $\mathcal{A}$ est déterministe et complet, ces ensembles sont de plus disjoints deux à deux et leur union vaut A^*. On peut alors rejouer l'argument de la question précédente.

3. On considère la structure $\mathcal{A}'_{S2S}$ qui a le même langage que $\mathcal{A}_{S2S}$. La seule différence dans l'interprétation est que le symbole de fonction 0 est interprété par $0(w) := 0w$, tandis que le symbole de fonction 1 garde son ancienne interprétation $1(w) := w1$. Montrer que la théorie de $\mathcal{A}'_{S2S}$ est indécidable.

   Indication: Réduire le problème indécidable suivant :

   Étant donnée une machine de Turing sur l'alphabet $\{a, b\}$ à ruban borné gauche et qui ne s'arrête jamais, et un état q, est-ce que l'état q est atteignable à partir de l'état initial et ruban infini (à droite) rempli avec 0 ?

   L'idée est de représenter un calcul d'une telle machine de Turing par une grille où chaque configuration est représentée par la partie du ruban à gauche de la tête, puis l'état, puis la partie du ruban à droite de la tête. Les cases de la grille sont donc remplies soit par a, b, ou par un symbole qui représente un état de la machine. Noter qu'une coordonnée (n, m) de la grille peut être représentée par le mot $0^n 1^m$. On utilisera des variables d'ensembles pour dénoter les cases qui sont remplies par a, b, et les états q_0, q_1, etc.

   Par hypothèse, la grille est $\mathbb{N}^2$. À létape i de l'exécution de la machine de Turing, on représente sa bande et son état par un mot de A^*QA^ω, où $A = \{a, b\}$ et Q est l'alphabet des états de la machine (la tête étant positionnée sur la lettre à droite de l'état). La grille mentionnée est l'empilement de ces bandes les unes sur les autres, verticalement.

   On va utiliser des variables d'ensemble X_0, X_a, X_b, X_{q_1}, ..., X_{q_k} pour dénoter les positions où l'on trouve un 0, un a, un b ou un état respectivement.

   Comme il ne doit y avoir qu'une lettre par case, on impose :

$$\textit{disjoint}(X_0, X_a, X_b, X_{q_1}, \ldots, X_{q_k}) \tag{1}$$

   On a ensuite besoin de dire quelle est la configuration initiale (le mot $q_{init}0^\omega$, où q_{init} est l'état initial) :

$$\epsilon \in X_{q_{init}} \wedge \forall w \cdot F_{1.1^*}(w) \Rightarrow w \in X_0 \tag{2}$$

   où $F_L(w)$ est la formule de la question précédente.

Ensuite on a besoin d'exprimer la bande $i+1$ en fonction de la bande i; c'est facile parce que les opérations $w \mapsto 0w$ et $w \mapsto w1$ nous permettent de monter d'un cran et de se déplacer d'un cran à droite respectivement.

Traitons des mouvements à droite d'abord. Pour représenter les mouvements de la machine de Turing à l'état q' et regardant la lettre α consistant en réécrire α en β, passer à l'état q'' et se déplacer à droite, on veut transformer tout mot à la ligne i de la forme $wq'\alpha w'$ en le mot $w\beta q''w'$ à la ligne $i+1$. On va donc écrire en particulier que si on voit un $q'\alpha$ sur n'importe quelle ligne i, alors juste au-dessus on doit voir un $\beta q''$:

$$\forall w \cdot w \in X_{q'} \wedge w1 \in X_\alpha \Rightarrow 0w \in X_\beta \wedge 0w1 \in X_{q''} \tag{3}$$

La partie un peu plus subtile est qu'il faut montrer que w et w' ne bougent pas. Or les lettres dans w et w' sont celles sur lesquelles la tête n'est pas positionnée ni en ligne i ni en ligne $i+1$. Posons $tete(w) = X_{q_1}(w) \vee \ldots \vee X_{q_k}(w)$ pour dire que la tête est en w. Posons $idem(w) = (X_0(w) \Leftrightarrow X_0(0w)) \wedge (X_a(w) \Leftrightarrow X_a(0w)) \wedge (X_b(w) \Leftrightarrow X_b(0w))$ pour dire que les lettres en w et juste au-dessus sont les mêmes (ou bien sont des lettres d'état). On écrit donc la formule :

$$\forall w \quad \cdot \quad \neg tete(w) \wedge \neg tete(0w) \Rightarrow idem(w) \tag{4}$$

Pour les mouvements à gauche, on doit réécrire $w\gamma q'\alpha w'$ à la ligne i en $wq''\gamma\beta w'$ à la ligne $i+1$:

$$\forall w \cdot w \in X_\gamma \wedge w1 \in X_{q'} \wedge w11 \in X_\alpha \Rightarrow 0w \in X_{q''} \wedge 0w1 \in X_\gamma \wedge 0w11 \in X_\beta \tag{5}$$

Il faut de plus montrer que w et w' ne bougent pas. Leurs lettres sont aux positions ou la tête n'est pas ni en ligne i ni en ligne $i+1$, et où la tête n'est pas non plus immédiatement à gauche en ligne i. Cette dernière condition revient à dire que la position w est soit de la forme 0^* soit de la forme $w'1$ avec la tête qui n'est pas en w'. On écrit donc deux formules correspondant à ces deux cas :

$$\begin{aligned} \forall w \quad &\cdot \quad F_{0^*}(w) \wedge \neg tete(w) \wedge \neg tete(0w) \Rightarrow idem(w) && (6) \\ \forall w \quad &\cdot \quad \neg tete(w') \wedge \neg tete(w'1) \wedge \neg tete(0w'1) \Rightarrow idem(w'1) && (7) \end{aligned}$$

Le fait que l'état q est atteignable est $\exists w \cdot w \in X_q$. Il ne reste donc qu'à écrire la formule :

$$\begin{aligned} \exists X_0, X_a, X_b, X_{q_1}, \ldots, X_{q_k} \quad \cdot \quad & (1) \wedge (2) \\ \wedge \quad & \bigwedge_{q',\alpha \text{ mvt. à droite } \beta,q''} ((3) \wedge (4)) \\ \wedge \quad & \bigwedge_{\gamma,q',\alpha \text{ mvt. à gauche } q'',\gamma,\beta} ((5) \wedge (6) \wedge (7)) \\ \wedge \quad & \exists w \cdot w \in X_{q_{fin}} \end{aligned}$$