Une variante des automates d'arbres

Correction.

On considère des ensembles de clauses définies de la forme :

$$P(t) \Leftarrow P_1(t_1), \ldots, P_n(t_n) \tag{1}$$

obéissant aux restrictions suivantes :

(i) t est une variable ou un terme *plat linéaire*, c'est-à-dire de la forme $f(x_1, \ldots, x_m)$, où f est un symbole de fonction et $x_1, \ldots, x_m$ des variables *distinctes*.

(ii) toutes les variables libres de t_i, $1 \leq i \leq n$, sont libres dans t.

Noter qu'il n'y a aucune restriction sur la profondeur ou la taille des termes $t_1, \ldots, t_n$.

On va considérer aussi des clauses *buts* de la forme :

$$\bot \Leftarrow P_1(t_1), \ldots, P_n(t_n) \tag{2}$$

sans aucune restriction sur les t_i.

La *profondeur* $d(t)$ d'un terme t est : $d(x) = 0$ pour toute variable x, et $d(f(t_1, \ldots, t_n)) = 1 + \max(d(t_1), \ldots, d(t_n), 0)$. La profondeur $d(A)$ d'un atome $A = P(t)$ est par définition la profondeur de t.

On considère d'autre part la règle de résolution ordonnée avec sélection, pour l'ordre $\succ$ défini comme dans le cours par $P(t) \succ Q(t')$ si et seulement si t' est un sous-terme strict de t, et la fonction de sélection telle que $\mathrm{sel}\ (A \Leftarrow B_1, \ldots, B_n)$ est l'ensemble de tous les littéraux négatifs $-B_i$ tels que $d(B_i) \geq d(A)$ et B_i est de profondeur $d(B_i)$ maximale. Autrement dit, on récupère d'abord tous les B_i tels que $d(B_i) \geq d(A)$; puis on ne garde que ceux de profondeur maximale parmi ces derniers.

1. Montrer que tout résolvant ordonné avec sélection de clauses de forme (1) ou (2) est encore d'une de ces deux formes.

Si l'on résout (2) avec une clause (1), on obtient clairement encore une clause (2). Les clauses (2) ne se résolvent pas entre elles, donc il reste à considérer le cas de deux clauses (1) :

$$\frac{P(t) \Leftarrow P_1(t_1), \ldots, P_n(t_n) \qquad P_1(u) \Leftarrow Q_1(u_1), \ldots, Q_p(u_p)}{P(t\sigma) \Leftarrow Q_1(u_1\sigma), \ldots, Q_p(u_p\sigma), P_2(t_2\sigma), \ldots, P_n(t_n\sigma)}$$

On rappelle qu'aucun littéral ne peut être sélectionné dans la prémisse de droite.

Cas 1. *Si u est une variable y, il s'ensuit que $p = 0$. Le mgu σ est donc $[y := t_1]$, où y n'est pas libre dans la prémisse de gauche. Donc le résolvant est*

$$P(t) \Leftarrow P_2(t_2), \ldots, P_n(t_n) \tag{3}$$

qui est clairement de la forme (1).

Cas 2. *Si u n'est pas une variable, alors u est de la forme $f(y_1, \ldots, y_m)$, où $y_1, \ldots, y_m$ sont des variables distinctes, par (i). On a alors deux cas.*

Cas 2.1. *Si t_1 n'est pas une variable, alors comme t_1 et u s'unifient, t_1 est de la forme $f(t'_1, \ldots, t'_m)$. De plus, comme u est linéaire, σ est $[y_1 := t'_1, \ldots, y_m := t'_m]$. Comme les y_i ne sont pas libres dans la prémisse de gauche, le résolvant est de la forme*

$$P(t) \Leftarrow Q_1(u_1\sigma), \ldots, Q_p(u_p\sigma), P_2(t_2), \ldots, P_n(t_n) \tag{4}$$

avec $\sigma = [y_1 := t'_1, \ldots, y_m := t'_m]$. Or comme la prémisse de gauche vérifie (i), le résolvant (4) aussi. Toute variable libre dans le corps (à droite de $\Leftarrow$) du résolvant est libre soit dans un t_i, donc dans t puisque la prémisse de gauche vérifie (ii), soit dans un $u_i\sigma$. Comme les variables de u_i sont toutes parmi $y_1, \ldots, y_m$ puisque la prémisse de droite vérifie (ii), les variables libres de $u_i\sigma$ sont libres dans un t'_j, donc dans t_1, donc dans t puisque la prémisse de gauche vérifie (ii). Donc (4) vérifie (ii).

Cas 2.2. *Si t_1 est une variable x, on va d'abord montrer que $-P_1(t_1)$ est nécessairement sélectionné dans la prémisse de gauche. Sinon, c'est qu'aucun littéral n'est sélectionné et que t_1 est maximal parmi t, $t_1, \ldots, t_n$. Comme aucun littéral n'est sélectionné, $d(t_i) < d(t)$ pour tout i. Ceci implique que t n'est pas une variable, donc un terme plat. De plus, par (ii), $t_1 = x$ est libre dans t, ce qui implique que t_1 est un sous-terme strict de t ; mais ceci contredit la maximalité de t_1.*
Comme $t_1 = x$ est sélectionné dans la prémisse de gauche, t est donc une variable. Par (ii), $t = x$. Comme seuls les littéraux de profondeur maximale sont sélectionnés, tous les t_i sont des variables. Par (ii), $t_i = x$ pour tout i. Comme $\sigma = [x := u]$, le résolvant est donc

$$P(u) \Leftarrow Q_1(u_1), \ldots, Q_p(u_p), P_2(u), \ldots, P_n(u) \tag{5}$$

où u est un terme plat linéaire par hypothèse (donc (i) est vérifiée), et toutes les variables libres dans u_1, …, u_p ou u sont libres dans u puisque la prémisse de droite vérifie (ii). Donc (5) vérifie (ii).

2. Un *argument* d'une clause est un terme t tel que la clause contient un littéral de la forme $\pm P(t)$. Étant données deux clauses C_1 et C_2, on dira $C_1 \preceq C_2$ si et seulement s'il existe un renommage ρ tel que tout argument u_1 de $C_1\rho$ est un sous-terme d'un argument de C_2. Montrer que, si C'' est le résolvant ordonné avec sélection de deux clauses C et C' de forme (1) ou (2), et si C n'est pas splittable, alors $C'' \preceq C$ ou $C'' \preceq C'$.

   C'est évident dans le cas 1 (résolvant (3)), où l'on a $C'' \preceq C$ (C étant la prémisse de gauche et C' celle de droite) et dans le cas 2.2 (résolvant (5)), où $C'' \preceq C'$. Dans le cas 2.1 (résolvant (4)), on remarque que comme aucun littéral n'est sélectionné dans la prémisse de droite, $d(u_i) < d(u)$ pour tout i. Donc tout u_i est une variable, qui plus est libre dans u par (ii) ; donc $u_i\sigma$ est de la forme t'_{j_i} pour tout i, qui est un sous-terme de t_1 ; donc $C'' \preceq C$.
   Dans le cas où l'on résout une clause (2) avec une clause (1), l'étape de résolution est :

$$\frac{\perp \Leftarrow P_1(t_1), \ldots, P_n(t_n) \qquad P_1(u) \Leftarrow Q_1(u_1), \ldots, Q_p(u_p)}{P(t\sigma) \Leftarrow Q_1(u_1\sigma), \ldots, Q_p(u_p\sigma), P_2(t_2\sigma), \ldots, P_n(t_n\sigma)}$$

   Soit C la prémisse de gauche, C' celle de droite, C'' le résolvant. On rappelle qu'aucun littéral ne peut être sélectionné dans la prémisse de droite.

   Cas 1. *Si u est une variable y, il s'ensuit que $p = 0$. Le mgu σ est donc $[y := t_1]$, où y n'est pas libre dans la prémisse de gauche. Donc C'' est*

$$\perp \Leftarrow P_2(t_2), \ldots, P_n(t_n) \tag{6}$$

   et donc $C'' \preceq C$.

Cas 2. *Si u n'est pas une variable, alors u est de la forme $f(y_1, \ldots, y_m)$, où y_1, …, y_m sont des variables distinctes, par (i). On a alors deux cas.*

Cas 2.1. *Si t_1 n'est pas une variable, alors comme t_1 et u s'unifient, t_1 est de la forme $f(t'_1, \ldots, t'_m)$. De plus, comme u est linéaire, σ est $[y_1 := t'_1, \ldots, y_m := t'_m]$. Comme les y_i ne sont pas libres dans la prémisse de gauche, le résolvant C'' est de la forme*

$$\perp \Leftarrow Q_1(u_1\sigma), \ldots, Q_p(u_p\sigma), P_2(t_2), \ldots, P_n(t_n) \tag{7}$$

avec $\sigma = [y_1 := t'_1, \ldots, y_m := t'_m]$. Comme sel $(C') = \emptyset$*, tous les u_i sont des variables, qui sont parmi y_1, …, y_m par (ii). Donc $C'' \preceq C'$.*

Cas 2.2. *Si t_1 est une variable x, on va d'abord montrer que $-P_1(t_1)$ est nécessairement sélectionné dans la prémisse de gauche C. En effet, comme C est une clause négative non vide,* sel $(C) \neq \emptyset$*, et dans ce cas on doit résoudre sur un littéral sélectionné.*
Comme t_1 est une variable, tous les t_i sont des variables. Par hypothèse, C n'est pas splittable, donc tous les t_i sont égaux à la variable x. Comme $\sigma = [x := u]$, le résolvant C'' est donc

$$\perp \Leftarrow Q_1(u_1), \ldots, Q_p(u_p), P_2(u), \ldots, P_n(u) \tag{8}$$

donc $C'' \preceq C'$.

3. En déduire que la satisfiabilité d'ensembles de clauses de forme (1) ou (2) est soluble en NEXPTIME (temps non-déterministe exponentiel).

   Soit S un tel ensemble de clauses. Par récurrence sur la longueur de la dérivation par résolution ordonnée avec sélection, en utilisant les questions précédentes, toute clause C' fabriquée par résolution ordonnée avec sélection et splitting à partir de S est de la forme (1) ou (2), et est telle que $C' \preceq C$ pour une certaine clause C de S.
   Fixons C, et le renommage ρ montrant que $C' \preceq C$. Chaque argument de $C'\rho$ est un sous-terme d'un argument de C, il y en a donc au plus $|C|$, où $|C|$ est la taille de C. Si l'on a p symboles de prédicats, on a donc au plus $2^{p|C|}$ choix possibles pour le corps de la clause. Donc on a au plus $p|C|2^{p|C|}$ clauses (1) et au plus $2^{p|C|}$ clauses (2) telles que $C' \preceq C$. Il n'y a donc qu'un nombre au plus simplement exponentiel de clauses déduites par résolution ordonnée avec sélection et splitting. Comme chaque étape de résolution prend un temps polynomial (l'unification prenant un temps $O(a) = O(1)$), et de même pour le splitting, la résolution ordonnée avec sélection et splitting termine en NEXPTIME. Une fois tous les résolvants calculés, S est insatisfiable si et seulement si la clause vide a été dérivée, d'où le résultat.

4. Montrer que ce problème est DEXPTIME-difficile, en réduisant un autre problème DEXPTIME-complet (trouvé dans le cours, par ex.) à celui de la satisfiabilité des clauses de forme (1) ou (2).

   Le problème de la vacuité de l'intersection de langages réguliers d'arbres est DEXPTIME-complet. Donnons-nous n automates d'arbres $S_1, \ldots, S_n$, et supposons sans perte de généralité qu'ils portent sur des ensembles de symboles de prédicats disjoints. Chaque clause d'automate non déterministe est de la forme (1) avec tous les t_i qui sont des variables. Pour tester si $L_{P_1}(S_1) \cap \ldots \cap L_{P_n}(S_n) = \emptyset$, il suffit d'ajouter une clause de la forme

   $$\perp \Leftarrow P_1(x), \ldots, P_n(x)$$

   par le lemme 5 du cours. Cette clause est clairement de la forme (2).

---

5. Montrer que la condition imposant que les variables x_1, …, x_n sont distinctes dans *(i)* est indispensable, au sens où l'ignorer produit un format de clauses dont la satisfiabilité est indécidable. (On garde la contrainte que t est une variable ou un terme plat.) On pourra s'inspirer des notes de cours et recoder par exemple le problème de correspondance de Post.

On va coder le problème de correspondance de Post dans le sens inverse des notes de cours. Autrement dit, on va d'abord construire l'ensemble des paires de mots identiques non vides, puis on va saturer cet ensemble sous la règle que si $(wu_i, w'v_i)$ est dans l'ensemble, où (u_i, v_i) est une des paires de mots en entrée, alors (w, w') est dans l'ensemble. Il ne restera plus qu'à demander si (ϵ, ϵ) est dans l'ensemble.
Il suffit donc d'écrirer :

$$\begin{aligned} & Q(a_j(\epsilon)) \qquad (1 \leq j \leq n) \\ Q(a_j(x)) \quad \Leftarrow \quad & Q(x) \qquad (1 \leq j \leq n) \end{aligned}$$

alors Q reconnaît tous les mots sur $a_1, \ldots, a_n$. On écrit maintenant l'unique clause qui brise la condition de linéarité :

$$P(c(x,x)) \quad \Leftarrow \quad Q(x)$$

qui exprime que P reconnaît toutes les paires $c(w,w)$ de mots w identiques non vides.
On écrit maintenant les règles de décomposition annoncées plus haut :

$$P(c(x,y)) \quad \Leftarrow \quad P(c(u_i(x), v_i(y))) \qquad (1 \leq i \leq p)$$

Il ne reste plus qu'à se demander si la paire (ϵ, ϵ) est reconnue par P :

$$\perp \quad \Leftarrow \quad P(c(\epsilon, \epsilon))$$

L'ensemble de clauses ainsi obtenu est alors insatisfiable si et seulement si le problème de correspondance de Post, sur les entrées (u_i, v_i), $1 \leq i \leq p$, a une solution.
Une autre (quasi-)solution, proposée par un élève, est de reprendre le codage du cours :

$$\begin{aligned} & P(\epsilon, \epsilon) && (9) \\ P(u_i(x), v_i(y)) \quad \Leftarrow \quad & P(x,y) \qquad (1 \leq i \leq p) && (10) \\ & Q(a_j(\epsilon)) \qquad (1 \leq j \leq n) && (11) \\ Q(a_j(x)) \quad \Leftarrow \quad & Q(x) \qquad (1 \leq j \leq n) && (12) \\ \perp \quad \Leftarrow \quad & P(x,x), Q(x) && (13) \end{aligned}$$

et de modifier le codage de la clause (10) pour qu'il rentre dans notre format. (Le fait que les prédicats ne soient pas d'arité 1 *est, on le sait, inessentiel, voir le cours.) L'idée à ce point est de créer un prédicat frais E à deux arguments, de le forcer à coder l'égalité de termes en écrivant l'unique clause suivante pour E :*

$$E(x,x)$$

et de remplacer la clause (10) par la clause :

$$P(x', y') \quad \Leftarrow \quad P(x,y), E(x', u_i(x)), E(y', v_i(y))$$

Cette clause ne vérifie pas la condition (ii), mais la question suivante montre que cette condition est en fait superflue pour la décidabilité.

6. Montrer que la condition *(ii)* est superflue. Autrement dit, montrer que la satisfiabilité d'ensembles de clauses (1) satisfaisant *(i)* mais pas nécessairement *(ii)*, et de clauses (2), est décidable en NEXPTIME. On pourra se contenter de montrer ce qui change dans ce cas par rapport à la démonstration que vous avez faite pour les questions 1 à 3, en justifiant les étapes nouvelles de raisonnement.

   On reprend la démonstration des questions 1 à 3.
   Question 1 : on montre que tout résolvant ordonné avec sélection de deux clauses (1) ou (2) non splittables *est encore de l'une de ces deux formes. Les cas 1 et 2.1 sont similaires.*

Dans le cas 2.2, où t_1 est une variable x, on commence de nouveau par montrer que $-P_1(t_1)$ est sélectionné dans la prémisse de gauche. Sinon, c'est qu'aucun littéral n'est sélectionné et que t_1 est maximal. Comme aucun littéral n'est sélectionné, $d(t_i) < d(t)$ pour tout i, donc t est un terme plat et tous les t_i sont des variables. Comme la prémisse de droite est supposée non splittable, tous les t_i sont des variables qui apparaissent comme argument du terme plat t, donc t_1 est un sous-terme strict de t, contredisant la maximalité de t_1.

Donc $-P_1(t_1)$ est sélectionné, donc t et tous les t_i sont des variables. Comme la clause est non splittable, tous ces termes sont la même variable x. Donc le résolvant est de la forme (5).

Question 2 : on montre que si ni C ni C' n'est splittable, alors $C'' \preceq C$ ou $C'' \preceq C'$. Le seul cas à changer est le cas 2.1, lorsque l'on résout deux clauses (1). On ne peut plus invoquer (ii) pour justifier que u_i est une variable libre de u. En revanche, C' n'étant pas splittable et u_i étant une variable, nécessairement u_i est une variable libre de u, et le raisonnement se poursuit comme en question 2.

Question 3 : l'argument est inchangé.

7. (Optionnel) Suggérer une technique permettant de montrer que le problème est en fait soluble en DEXPTIME, donc DEXPTIME-complet. On ne demande pas d'effectuer la preuve, seulement de définir une autre stratégie plausible de démonstration automatique qui termine en temps déterministe exponentiel.

   Au lieu d'utiliser le splitting, on va utiliser le splitting sans splitting, comme en section 6.5 du cours. Comme dans le cours, on laisse l'ordre $\succ$ inchangé, et on modifie la fonction de sélection de sorte qu'elle retourne l'ensemble des littéraux de la forme $-q(t)$ s'il y en a, et sinon retourne comme plus haut l'ensemble des littéraux négatifs de profondeur maximale supérieure ou égale à celle de la tête de la clause. Comme il n'y a qu'un nombre exponentiel de blocs splittés n'utilisant pas de symbole de la forme $\ulcorner C \urcorner$, il n'y a qu'un nombre exponentiel de symboles $q = \ulcorner C \urcorner$. De plus, on peut montrer comme dans le cours que toute clause obtenue ne contient qu'un $+q(*)$, ou au plus m $-q_i(*)$, où m est une borne sur l'arité maximale des symboles de fonctions et sur le nombre de blocs dans l'ensemble initial de clauses. Les détails sont fastidieux, et n'étaient pas demandés.

En utilisant le théorème 15, et en modifiant sa preuve, on peut montrer que, bien que les ensembles de clauses (1) satisfaisant (i) définissent exactement les langages réguliers d'arbres. Ceci peut paraître surprenant, étant donné l'aspect relativement général des clauses (1).

On peut aussi déduire de cet exercice un théorème dû à Sophie Tison : étant donné un terme t du premier ordre quelconque (en particulier pas spécialement linéaire), l'existence d'une instance $t\sigma$ de t reconnue par un automate d'arbres S donné est décidable, en temps exponentiel. Il suffit en effet d'écrire l'automate d'arbres S sous forme de clauses, et si son état final est P, de lui adjoindre la clause but $\perp \Leftarrow P(t)$.