École polytechnique — X2012 — INF423 Fondements de l'informatique

Logique, modèles, calculs: PC notée

Sujet proposé par David Monniaux
(corrigé)

Les différentes parties sont indépendantes. Vous pouvez résoudre un exercice en admettant les résultats à démontrer dans les exercices précédents de la même partie.

Il sera tenu compte dans la notation de la clarté des raisonnements (à part la question 1.4, qui demande quelques explications, toutes les solutions du corrigé tiennent en quelques lignes ; si vous partez dans quelque chose de compliqué, vous faites probablement fausse route).

Merci d'écrire lisiblement.

1 Logique du premier ordre : le fragment de Bernays, Schönfinkel et Ramsey

On sait que la logique du premier ordre est *indécidable* : il n'existe pas d'algorithme qui, étant données une signature Σ et une formule F quelconque sur cette signature, termine systématiquement en répondant si F possède ou non un modèle.

Il existe en revanche des classes restreintes de formules du premier ordre, définies par des critères syntaxiques (forme de la formule, ce que l'on peut inclure ou non dans la signature), qui sont décidables. Nous étudierons ici le fragment de Bernays, Schönfinkel et Ramsey [3] (BSR), qui met les restrictions suivantes :

1. Les seules fonctions de la signature sont des constantes (autrement dit, il n'y a de fonctions n-aires pour aucun $n > 0$).
2. Les formules acceptées sont de la forme suivante, parfois notée $\exists^*\forall^*$: d'abord un nombre quelconque (éventuellement nul) de quantificateurs existentiels, puis un nombre quelconque (éventuellement nul) de quantificateurs universels, puis une formule sans quantificateurs.

Par exemple, la formule

$$\forall x \forall y \forall z \; R(x,x) \wedge \big(R(x,y) \wedge R(y,z) \Rightarrow R(x,z)\big),$$

sur la signature où R est un prédicat binaire, qui définit que R est une relation de préordre, fait partie du fragment BSR. En revanche, $\exists x \forall y \exists z R(x,y,z)$ n'en fait pas partie, puisqu'on n'a pas le droit d'avoir un quantificateur existentiel après un universel.

Nous allons tout d'abord nous débarrasser des quantificateurs existentiels. Commençons tout d'abord par l'intuition de la méthode, sur un exemple.

Question 1.1. *Donnez un modèle de la formule $\exists x \exists y \forall z \; R(x,z) \wedge \neg R(y,z)$ sur la signature Σ consistant en un prédicat binaire R, tel que ce modèle soit également un modèle de $\forall z \; R(a,z) \wedge \neg R(b,z)$ sur la signature consistant en un prédicat binaire R et deux symboles de constante a et b.*

Solution : Prendre $R(a,a)$ et $R(a,b)$ vrais, mais $R(b,a)$ et $R(b,b)$ faux. □

Si Σ est une signature (rappel : la signature spécifie, pour chaque symbole de fonction ou de prédicat, son arité), on dit qu'une signature Σ' *étend* Σ si elle spécifie les mêmes prédicats et

arités que Σ, auxquelle elle ajoute (éventuellement) des fonctions et prédicats supplémentaires. Par exemple, si on se place sur une signature Σ spécifiant un prédicat binaire A et un symbole de constante k, la signature Σ' spécifiant un prédicat binaire A, un prédicat unaire B et deux symboles de constantes k et l étend Σ, mais la signature Σ'' spécifiant un prédicat unaire A et un symbole de constante k n'étend pas Σ, puisqu'elle ne spécifie pas la même arité pour A.

Question 1.2. *Soit F une formule et Σ une signature de la classe BSR. Montrez qu'il existe une formule F' et une signature Σ' étendant Σ, toujours de la classe BSR, tels que :*

- *F' ne comporte pas de quantificateurs existentiels,*
- *un modèle de F' (ramené à la signature Σ) est un modèle de F et un modèle de F permet d'obtenir un modèle de F'.*

Solution : Soient $x_1, \ldots, x_n$ les variables quantifiées existentiellement dans F, qu'on renomme si besoin afin d'obtenir des noms distincts de ceux des symboles de la signature Σ. On prend $\Sigma' = \Sigma \cup \{x_1, \ldots, x_n\}$ et F' la formule F sans son préfixe de quantificateurs existentiels. □

On se ramène donc, pour la suite des questions, au cas des formules $\forall^*$ (des quantificateurs universels suivis d'une formule sans quantificateurs). Nous allons tout d'abord montrer une propriété dite « de petit modèle », qui nous permet de remplacer la recherche sur des modèles de taille quelconque par la recherche d'un modèle de taille bornée.

Question 1.3. *Soient $x_1, \ldots, x_n$ les symboles de constantes. Montrez que si une formule F de type $\forall^*$ admet un modèle, alors elle admet un modèle de cardinal au plus n (par cardinal du modèle nous entendons celui de l'ensemble de base).*

Solution : Soit $\mathcal{M}$ un modèle de F. À chaque symbole de constante x_i, ce modèle associe un objet $\mathcal{M}[x_i] \in \mathcal{M}$, et pour chaque prédicat n-aire π ce modèle donne une fonction $\mathcal{M}[\pi] \in \mathcal{M}^n \to \{0,1\}$. Posons maintenant $\mathcal{M}'$ la restriction de $\mathcal{M}$ à l'ensemble $\{\mathcal{M}[x_1], \ldots, \mathcal{M}[x_n]\}$. C'est également un modèle de F, de cardinal au maximum n. □

Question 1.4. *Soit F une formule $\forall^*$, de taille $|F|$, sur une signature à n symboles de constantes, avec m quantificateurs universels. Montrez qu'il existe une formule F' purement propositionnelle et sans quantificateurs, de taille $O(n^m|F|)$, qui est satisfiable si et seulement si F l'est. Par taille de la formule, nous entendons son nombre de symboles (chaque constante, prédicat, variable, quantificateur, connecteur logique compte pour un).*

Solution : Commençons par un exemple pour donner l'intuition. Considérons une formule $F : \forall x \forall y \ \neg R(x,y) \vee \neg R(y,x)$ sur une signature avec trois symboles de constantes a, b, c, un prédicat binaire R. Nous construisons un modèle dont l'ensemble de base a au plus autant d'éléments qu'il y a de constantes dans la signature.

Sur un tel modèle *fini*, la quantification universelle peut se « déplier » en une conjonction : la formule F est, sur un modèle à trois éléments a, b, c, équivalente à la conjonction où l'on énumère tous les cas pour le couple $(x, y) \in \{a, b, c\}^2$:

$$\begin{aligned}(\neg R(a,a) \vee \neg R(a,a)) \wedge (\neg R(a,b) \vee \neg R(b,a)) \wedge (\neg R(a,c) \vee \neg R(c,a)) \wedge \\ (\neg R(b,a) \vee \neg R(a,b)) \wedge (\neg R(b,b) \vee \neg R(b,b)) \wedge (\neg R(b,c) \vee \neg R(c,b)) \wedge \\ (\neg R(c,a) \vee \neg R(a,c)) \wedge (\neg R(c,b) \vee \neg R(b,c)) \wedge (\neg R(c,c) \vee \neg R(c,c))\end{aligned}$$

Il suffit alors de déclarer des variables propositionnelles $R_{a,a}, R_{a,b}, R_{a,c}, R_{b,a}, R_{b,b}, R_{b,c}, R_{c,a}, R_{c,b}, R_{c,c}$ et de transformer la formule ci-dessus en formule purement propositionnelle sur ces 9 variables.

Maintenant, la preuve. Soient $x_1, \ldots, x_m$ les variables quantifiées universellement dans F. Pour $\langle \alpha_1, \ldots, \alpha_m \rangle \in \{1, \ldots, n\}^m$, notons $G[\alpha_1, \ldots, \alpha_m]$ la formule F privée de ses quantificateurs

universels, et dans laquelle on a remplacé chaque prédicat k-aire $P(x_{i_1}, \ldots, x_{i,k})$ par une variable propositionnelle $P_{\alpha_{i_1},\ldots,\alpha_{i_k}}$. On remarquera que pour chaque prédicat k-aire, on utilisera au plus m^k variables propositionnelles.

La formule F' est alors

$$\bigwedge_{\alpha_1,\ldots,\alpha_m\rangle \in \{1,\ldots,n\}^m} G[\alpha_1, \ldots, \alpha_m],$$

À un modèle $\mathcal{M}$ de F correspond un modèle de F' de cardinal au plus n : à chaque variable propositionnelle $P_{i_1,\ldots,i_k}$ de F', correspondant à un prédicat k-aire de F, on associe la valeur booléenne $\mathcal{M}[x_{i_1}, \ldots, x_{i_k}]$.

Réciproquement, si on a un modèle $\mathcal{M}'$ de F', alors on en déduit un modèle $\mathcal{M}$ de F sur l'ensemble $\{1, \ldots, n\}$ où l'on impose la valeur de $P(i_1, \ldots, i_k)$ dans $\mathcal{M}$ à celle de $P_{i_1,\ldots,i_k}$ dans $\mathcal{M}'$. Pour les $P(i_1, \ldots, i_k)$ pour lesquels il n'existe aucune variable propositionnelle $P_{i_1,\ldots,i_k}$ associée dans F', on met n'importe quelle valeur. □

Ainsi, on peut convertir un problème du fragment BSR en un problème purement propositionnel en quelque sorte équivalent (mais, certes, de taille exponentiellement plus grande), d'où la terminologie parfois utilisée pour ce fragment de « effectivement propositionnel » (EPR). Des logiciels de décision de formules comme Z3 [1] ont d'ailleurs des procédures spécialisées pour les formules EPR, car de nombreux problèmes peuvent se mettre sous cette forme avec un peu de travail [2]. À noter que la décision de ce fragment est coûteuse [1].

Question 1.5. *Montrez que pour toute conjonction de formules $\forall^*$ sur une signature il existe une formule $\forall^*$ équivalente sur la même signature, de taille au plus linéaire en la taille de la conjonction.*

Solution : On montre la propriété pour une conjonction de deux formules ; le résultat général se déduit par récurrence. On a deux formules $\forall x_m \ldots \forall x_m\ A$ et $\forall y_1 \ldots \forall y_n\ B$. Sans perte de généralité, on suppose $m \leq n$. On note A' la formule A où l'on a remplacé chaque x_i par y_i. Alors $(\forall x_m \ldots \forall x_m\ A) \wedge (\forall y_1 \ldots \forall y_n\ B)$ est équivalente à $\forall y_1 \ldots \forall y_n\ A' \wedge B$. □

Nous voulons maintenant aborder des théories « égalitaires, », c'est-à-dire des théories dont la signature comporte un prédicat $=$ qui devra être interprété par l'égalité dans les modèles. Si on applique tels quels les raisonnements précédents, en considérant $=$ comme un prédicat binaire associatif et commutatif comme les autres, on peut se retrouver avec des solutions indésirables, comme $a = b \wedge R(a) \wedge \neg R(b)$. Il faut donc prendre quelques précautions.

Question 1.6. *Soit une formule F de type $\forall^*$ sur la signature Σ, comportant le prédicat $=$. Montrez qu'on peut construire une formule F', également de type $\forall^*$, sur la signature Σ' où l'on a remplacé $=$ par un symbole de prédicat binaire $\equiv$, telle que tout modèle de F est un modèle de F' si on remplace $=$ par $\equiv$, et que tout modèle de F' se transforme en un modèle de F.*

Solution : Comme dans la PC « premier ordre » et dans le « poly » : on contraint le symbole $\equiv$ à être une congruence

- relation d'équivalence : réflexivité ($\forall x\ x \equiv x$), transitivité ($\forall x \forall y \forall z\ x \equiv y \wedge y \equiv z \Rightarrow x \equiv z$), symétrie ($\forall x \forall y\ x \equiv y \Rightarrow y \equiv x$) ;
- congruence : pour tout prédicat k-aire P, on ajouté la contrainte

$$\forall x_1 \ldots \forall x_k \forall y_1 \ldots \forall y_k\ x_1 \equiv y_1 \wedge \cdots \wedge x_k \equiv y_k \wedge P(x_1, \ldots, x_k) \Rightarrow P(y_1, \ldots, y_k)$$

La conjonction de F dans laquelle on a substitué $=$ par $\equiv$ et de toutes ces contraintes n'est pas directement une formule $\forall^*$: c'est une conjonction de formules $\forall^*$. On conclut par la question précédente. □

1. `http://z3.codeplex.com/`

2 Calculabilité : unions et intersections de langages

Question 2.1. *Soient A et B deux langages semidécidables (= récursivement énumérables). Montrez que leur union et leur intersection le sont aussi (et même que l'on peut calculer les codages des machines de Turing qui décident celles-ci en fonction de ceux de celles qui décident A et B).*

Solution : Notons a et b les machines de Turing qui reconnaissent A et B.

Intersection : sur l'entrée x, simuler $a(x)$ puis simuler $b(x)$.

Union : simuler simultanément les calculs de $a(x)$ et $b(x)$ (faire $0, 1, 2 \ldots$ pas des deux calculs), terminer dès que l'un des deux calculs termine. □

Nous pouvons nous poser la même question quant à des unions et intersections infinies.

Question 2.2. *Soit A_k une famille de langages semidécidables, donnés par une fonction calculable a telle que $a(k)$ désigne le codage de la machine de Turing qui reconnaît A_k. Montrez que leur union est également semidécidable (et même que l'on peut calculer le codage d'une machine de Turing qui semidécide cette union en fonction de celui d'une machine qui calcule a).*

Solution : $x \in \bigcup_k A_k$ si et seulement s'il existe (y, z) tels que le calcul « calculer $a(y)$ puis exécuter la machine $a(y)$ sur l'entrée x » termine en au maximum z pas, ce qui se teste effectivement. Il suffit donc d'énumérer tous les couples (y, z). □

On admettra que l'ensemble suivant n'est pas semidécidable (ce résultat a été démontré dans la PC5) :

$$T = \{x \mid \forall y \text{ La machine de codage } x \text{ termine sur l'entrée } y\}.$$

Question 2.3. *Qu'en est-il d'une intersection infinie de langages semidécidables ?*

Solution : Soit A_k l'ensemble des machines de Turing qui terminent sur l'entrée k. Cet ensemble est clairement semidécidable. L'intersection de tous ces ensembles est l'ensemble T des machines de Turing qui terminent sur toute entrée, qui n'est pas semidécidable, comme admis. □

3 Application de la calculabilité aux programmes informatiques

Nous étudierons ici un langage dérivé du langage Java, où nous supposerons que les types entiers ne sont pas bornés (autrement dit, **int** correspond aux entiers mathématiques $\mathbb{Z}$). Nous admettrons qu'un tel langage peut représenter les mêmes programmes que les machines de Turing, et réciproquement.

Le compilateur pour ce langage, comme celui du langage Java, refuse les programmes dont il ne peut établir par des moyens simples (et d'ailleurs décrits dans le standard Java) que toutes les variables locales sont initialisées avant d'être utilisées. Ce test peut refuser des programmes qui n'utilisent en aucun cas de variable non initialisée. Bref, il s'agit d'un test *pessimiste*, qui vise à refuser tous les programmes posant problème, quitte à en refuser certains qui ne posent pas de problème.

Par exemple, le compilateur refuse le programme suivant au motif que la variable min peut être utilisée sans être initialisée aux lignes 7 et 12 ; or cela est impossible, parce que la variable min est forcément initialisée lors de l'itération $\mathsf{k} = 0$ avant qu'on ne l'utilise.

```
1  class Uninitialized {
2      int minimum(double[] x) {
3          if (x.length < 1) return -1;
4          int min;
```

```
5           boolean initialized = false;
6           for(int k=0; k<x.length; k++) {
7               if (!initialized || x[k] < x[min]) {
8                   min = k;
9                   initialized = true;
10              }
11          }
12          return min;
13      }
14  }
```

Question 3.1. *Pourquoi le compilateur ne détecte-t-il pas exactement les cas où les variables sont réellement utilisées sans initialisation ? On pourra notamment s'intéresser à la condition d'utilisation de* x *dans le programme suivant :*

```
int x, y;
P();
y=2*x;
```

Solution : La variable x est lue si et seulement si P() termine, donc l'analyse devrait déterminer si P() termine. □

Références

[1] Harry R. Lewis. Complexity results for classes of quantificational formulas. *Journal of Computer and System Sciences*, 21(3) :317–353, 1980. doi: 10.1016/0022-0000(80)90027-6.

[2] Ruzica Piskac, Leonardo de Moura, and Nikolaj Bjørner. Deciding effectively propositional logic with equality. Technical report 181, Microsoft Research, 2008. URL `http://research.microsoft.com/pubs/76532/tr-2008-181.pdf`.

[3] F. P. Ramsey. On a problem of formal logic. *Proceedings of the London Mathematical Society*, 30 :264–286, 1930. doi: 10.1112/plms/s2-30.1.264.