SAP — TD6 — *Bounded model checking, procédures de décision*

31 mars 2009

1 Procédure de décision pour l'arithmétique linéaire

Nous allons proposer un algorithme qui, étant donné une formule logique composée de $\wedge$, $\vee$, $\neg$, variables propositionnelles et inégalités linéaires, dit si cette formule est satisfiable.

1.1 Mise en forme normale négative

Nous allons d'abord utiliser le fait que $\neg(a \vee b) \equiv (\neg a) \wedge (\neg b)$ et $\neg(a \wedge b) \equiv (\neg a) \vee (\neg b)$ en remplaçant toute sous-formule de la forme de gauche vers la forme de droite. Montrez qu'à force de remplacements, on obtient une formule où les seules négations sont aux feuilles de l'arbre syntaxique (dite *negative normal form* ou NNF).

Proposez une façon économique d'implémenter cette transformation et donnez sa complexité.

1.2 Mise en forme normale disjonctive

Nous allons utiliser le fait que $(a \vee b) \wedge c \equiv (a \wedge c) \vee (b \wedge c)$ en remplaçant toute sous-formule de la forme de gauche vers la forme de droite. Montrez qu'à force de remplacements, on obtient une formule de la forme

$$(\cdots \wedge \cdots \wedge \dots) \vee \cdots \vee (\cdots \wedge \cdots \wedge \dots) \tag{1}$$

dont les feuilles de l'arbre syntaxique (les atomes) sont les atomes de la formule d'origine. Cette formule est dite *forme normale disjonctive* (DNF).

Quelle est la taille maximale de la formule résultante ?

Si on veut vérifier si $\exists b_1, \dots, b_m \in \mathbb{B} \exists x_1, \dots, x_n \in \mathbb{R}$ F, F une formule comprenant les variables booléennes $b_1, \dots, b_m$ et les variables réelles $x_1, \dots, x_n$, on peut commencer par mettre F en forme normale disjonctive, et on se ramène au cas de décider si $\exists x_1, \dots, x_n$ C où C est une conjonction d'inégalités linéaires (larges ou strictes).

1.3 Élimination de Fourier-Motzkin

On veut maintenant décider si un système (une conjonction) d'inégalités linéaires larges a une solution ou pas. On pratique une *élimination de quantificateurs* : étant donné une formule $\exists x\ F$, on produit une formule équivalente F'.

On divise les inégalités en celles I_+ imposant une borne supérieure sur x (équivalentes à $x \leq \dots$ où x n'intervient pas dans $\dots$), celles I_- imposant une borne inférieure, et celles I_0 où x n'intervient pas.

On propose de garder les inégalités de I_0, et de remplacer les autres par l'ensemble des inégalités de la forme $L_- \leq L_+$ où $x \leq L_+$ est dans I_+ et $x \geq L_-$ est dans I_-. L_- et L_+ sont des expressions linéaires affines en les autres variables que x.

Montrez que ce que l'on obtient est équivalent au problème d'origine.

Montrez que si on poursuit le procédé jusqu'à éliminer toutes les variables, on peut décider si le système d'origine a une solution (et en fournir une le cas échéant). Donnez une borne de complexité.

Comment traiter les inégalités strictes ?

1.4 NP-complétude

On admettra qu'il est possible de résoudre en temps polynomial le problème de *programmation linéaire* : étant donné des inégalités linéaires à coefficients entiers, et une forme linéaire, dire si ces inégalités ont une solution et si oui, fournir la solution maximisant la forme linéaire.

Montrer que le problème de décider $\exists b_1, \dots, b_m \exists x_1, \dots, x_n\ F$ où F est une formule à base de $\wedge$, $\vee$, $\neg$, variables booléennes b_i et inégalités linéaires sur les x_i est NP-complet.

2 Fonctions non interprétées

Pour modéliser les tableaux, la mémoire, on les assimile à des fonctions des entiers (ou d'un sous-ensemble des entiers) vers les entiers, avec une construction $t[i \mapsto x]$ qui vaut $t(j)$ partout sauf pour i où elle vaut x (permet de modéliser l'affectation).

On a donc des formules du style $f(i) = x{+}4 \wedge f(j{+}2) = 3 \wedge j \geq i{-}2 \wedge j{+}2 \leq i$, et on cherche si elles ont une solution.

Comment (sans chercher à optimiser) vous ramèneriez-vous au cas sans ces fonctions ?