Projet IPR

Sujet 1

4 décembre 2007

L'algorithme DPLL, nommé d'après ses inventeurs Davis, Putnam, Logemann et Loveland, est la procédure la plus classique pour décider si une formule propositionnelle en forme normale conjonctive[1] (FNC) est satisfaisable ou non. Pour cela, l'algorithme essaye de construire une instanciation des variables propositionnelles qui rende la formule vraie. En principe, l'algorithme verifie toutes les 2^n possibilités d'instancier les variables, mais il utilise deux heuristiques intelligentes qui lui permettent d'aller plus vite :

- la *propagation des contraintes booléennes* : à chaque fois qu'une valeur de vérité est choisie pour une variable, la formule est simplifiée en conséquence : les littéraux devenus faux sont supprimés des clauses, et si une clause contient un littéral devenu vrai, toute la clause est supprimée ;
- la règle de la *clause unitaire* : si la formule contient une clause formée d'un seul littéral, la valeur de vérité de la variable de ce littéral est choisie de sorte qu'il soit vrai.

De cette manière, l'algorithme procède en attribuant une valeur de vérité à chaque variable propositionnelle appartenant à la formule, jusqu'à ce que l'un des deux évenements suivants arrive :

- la formule simplifée est la conjonction vide $\emptyset$; dans ce cas, une instanciation des variables a été trouvée qui rend vraie la formule de départ : elle est donc satisfaisable et l'algorithme s'arrête ;
- l'algorithme a atteint un état dans lequel la formule simplifée contient une clause vide (un *conflit*) : dans ce cas, l'algorithme effectue un retour en arrière à l'endroit le plus récent où il peut affecter une autre valeur de vérité à une variable.

Pour la notation des formules en FNC, nous adoptons les conventions suivantes :

- l'ordre des littéraux dans une clause, ou des clauses dans une conjonction, est sans importance ; nous écrirons $l \vee C$ pour décrire une clause qui contient au moins le littéral l et $\{l_1, l_2, l_3\}$ pour la clause formée des littéraux l_1, l_2 et l_3 ;
- une formule en FNC est écrite $C_1, \ldots, C_n$ où les C_i sont les clauses de la formule. Parfois, nous omettrons les accolades des clauses singletons : si

[1] *i.e.* une formule de la forme $\bigwedge_{i=1}^{n}(l_1 \vee \cdots \vee l_{k_i})$, où chaque l_j est un littéral, c'est-à-dire une variable propositionnelle ou sa négation.

Δ est un ensemble de clauses, Δ, l est une formule avec une clause qui ne contient que le littéral l.

$$\textsc{Conflict}\ \frac{}{\Gamma \vdash \Delta, \emptyset} \qquad \textsc{Assume}\ \frac{\Gamma, l \vdash \Delta}{\Gamma \vdash \Delta, l} \qquad \textsc{Bcp}\ \begin{cases} \dfrac{\Gamma, l \vdash \Delta, C}{\Gamma, l \vdash \Delta, \bar{l} \vee C} \\ \dfrac{\Gamma, l \vdash \Delta}{\Gamma, l \vdash \Delta, l \vee C} \end{cases}$$

$$\textsc{Unsat}\ \frac{\Gamma, l \vdash \Delta \qquad \Gamma, \bar{l} \vdash \Delta}{\Gamma \vdash \Delta}$$

FIG. 1 – Une version abstraite de DPLL

La figure 1 montre le fonctionnement de l'algorithme DPLL, à l'aide de cinq règles d'inférences. Elles décrivent l'état de l'algorithme par le *séquent* $\Gamma \vdash \Delta$, où Γ est l'ensemble des littéraux supposés vrais, et Δ est la formule courante. Ces règles doivent être lues de bas en haut : l'état sous le trait est l'état *avant* l'application de la règle d'inférence. La règle CONFLICT correspond au cas où l'algorithme a trouvé la clause vide dans la formule. Cette branche de l'arbre de recherche étant terminée, l'algorithme doit revenir en arrière pour trouver une autre instanciation des variables. La règle ASSUME décrit l'heuristique des clauses unitaires : un littéral dans une clause unitaire doit être supposé vrai, sans quoi la formule serait immédiatement insatisfaisable.

Les règles BCP décrivent la propagation des contraintes propositionnelles imposées par les variables dans Γ. Si un littéral est supposé faux (sa négation est dans Γ), il peut être supprimé de la clause (première règle). Si une clause contient un littéral qui est supposé vrai, la clause peut être entièrement supprimée (deuxième règle).

Finalement, la règle UNSAT, qui doit se lire de bas en haut et de gauche à droite, décrit le choix d'une valeur de vérité pour un littéral de la formule. Il est d'abord supposé vrai (partie gauche), et si aucune instanciation satisfaisable n'a été trouvée (toutes les branches de l'arbre de gauche terminent avec la règle CONFLICT), l'algorithme revient par backtracking à cette règle UNSAT et essaye la branche droite, en supposant cette fois le littéral faux.