Master Informatique FIIL — 2016–17
Preuves Interactives et Applications — 19 octobre 2016
http://www.lri.fr/~paulin/PreuvesInteractives

Projet : étude de la procédure DPLL

1 Modalités

- Le projet sera réalisé en binôme.
- Vous avez le choix entre utiliser Coq ou Isabelle/HOL pour le réaliser.
- Vous devez rendre une archive composée d'un seul dossier portant le nom des deux binômes qui comprendra le ou les fichiers de développement ainsi qu'un rapport court (1 à 3 pages) précisant les points suivants :
  — version de logiciel utilisée ;
  — contenu des fichiers par rapport au travail demandé ;
  — modalités pour exécuter les fichiers ;
  — remarques éventuelles sur les choix réalisés.
- La date de retour impérative est le **lundi 21 novembre** par mail à Burkhart.Wolff@lri.fr (développement Isabelle) ou Christine.Paulin@lri.fr (développement Coq)
- Les fichiers de développement doivent contenir des commentaires et des séquences de test permettant de valider les définitions.
- Le projet compte pour 1/3 de la note finale en première et deuxième session.
- Le travail rendu doit correspondre à un travail personnel, une trop grande similitude entre deux projets ou avec des ressources disponibles sur Internet sera pénalisée.

2 Description du projet

Le but du projet est de formaliser et prouver l'algorithme DPLL de recherche de la satisfiabilité d'un ensemble de clauses propositionnelles.

La procédure DPLL (du nom de ses inventeurs Davis, Putnam, Logemann et Loveland) permet de dire si une formule en forme normale conjonctive (représentée par un ensemble de clauses) est satisfiable. Pour cela, l'algorithme cherche à construire une interprétation qui rend la formule vraie.

L'algorithme DPLL travaille sur des problèmes de la forme : $I \gg \Delta$ avec I une interprétation partielle des variables propositionnelles de la forme $\{x_1 \mapsto b_1; \ldots; x_n \mapsto b_n\}$ (que l'on peut aussi représenter comme un ensemble de littéraux x_i si $b_i = V$ et $\neg x_i$ si $b_i = F$) et Δ un ensemble de clauses $C_1, \ldots, C_n$. On note Δ, C le problème qui contient toutes les clauses de Δ plus la clause C. Le but de l'algorithme est de construire une interprétation qui étend I, c'est-à-dire de la forme $\{x_1 \mapsto b_1; \ldots; x_n \mapsto b_n; x_{n+1} \mapsto b_{n+1} \ldots x_p \mapsto b_p\}$ et qui rend vraies toutes les clause C_i. Les règles sont les suivantes :

Il y a deux cas triviaux. Le premier cas s'il n'y a plus de clauses, l'interprétation I répond au problème. Le second cas si l'une des clauses C_i est la clause vide alors le problème n'a pas de solution. Cela correspond aux deux règles :

$$\textsc{Success}\frac{}{I \gg \emptyset} \quad \textsc{Conflict}\frac{}{I \gg \Delta, \bot}$$

Par ailleurs on peut simplifier les problèmes : lorsque l'interprétation I fixe la valeur de la variable x, alors les clauses C qui contiennent un litteral l de la forme x ou $\neg x$ se simplifient :

- si $I(l) = V$ alors la clause est trivialement vraie dans cette interprétation et peut être supprimée ;
- si $I(l) = F$ alors on peut retirer le littéral l de la clause.

Ceci est représenté par les deux règles suivantes :

$$\textsc{Bcp}_V\frac{I \gg \Delta \quad I(l) = V}{I \gg \Delta, (l \vee C)} \quad \textsc{Bcp}_F\frac{I \gg \Delta, C \quad I(l) = F}{I \gg \Delta, (l \vee C)}$$

Maintenant si aucune des règles précédentes ne s'applique, alors il faut préciser l'interprétation en choisissant une valeur pour une nouvelle variable. Un cas simple est si une clause ne contient qu'un seul littéral alors pour que la clause soit vraie il faut choisir la valeur de la variable pour rendre ce littéral vrai. Ceci s'exprime par les règles suivantes :

$$\textsc{Assume}_V\frac{I + \{x \mapsto V\} \gg \Delta \quad x \notin I}{I \gg \Delta, x} \quad \textsc{Assume}_F\frac{I + \{x \mapsto F\} \gg \Delta \quad x \notin I}{I \gg \Delta, \neg x}$$

Si maintenant, il n'y a aucune clause réduite à un littéral, alors il n'y a pas de choix évident. On va donc choisir une variable qui n'a pas encore de valeur et explorer les deux branches : celle où cette variable est interprétée par V et celle où elle est interprétée par F.

$$\textsc{Unsat}\frac{I + \{x \mapsto V\} \gg \Delta \quad I + \{x \mapsto F\} \gg \Delta \quad x \notin I}{I \gg \Delta}$$

On part d'un ensemble de clauses Δ et d'une interprétation vide. On construit un arbre en utilisant les règles précédentes et on cherche à arriver à une feuille de succès de la forme $I \gg \emptyset$. L'interprétation I est une solution de notre problème, elle rend vraie l'ensemble des clauses Δ.

Si toutes les feuilles de l'arbre correspondent à des conflits alors l'ensemble de clauses initial n'est pas satisfiable.

1. Représentation.
   - On représente les variables par des entiers (naturels ou relatifs).
   - Un litteral peut se représenter par un couple (x, b) avec x le numéro de la variable et b un booléen qui est *true* pour le littéral x et *false* pour le littéral $\neg x$.
   - Une clause peut se représenter par une liste de littéraux trié par le numéro de variable.
   - Une interprétation partielle des variables peut aussi se représenter par des listes de littéraux.
   - On pourra utiliser les bibliothèques sur les ensembles finis pour représenter les ensembles de clauses.

   Mettre en place les définitions pour représenter les données du problème.
2. Transformer les règles précédentes en une définition inductive $\texttt{DPLL}(I, \Delta, \mathit{sol})$ avec I une interprétation partielle, Δ un ensemble de clauses et *sol* l'ensemble des solutions trouvées aux feuilles de succès de l'arbre de racine $I \gg \Delta$. Cet ensemble peut être vide si toutes les feuilles sont des conflits.
3. Montrer que les règles de DPLL sont correctes. C'est-à-dire que si $\texttt{DPLL}(I, \Delta, \mathit{sol})$ est vrai alors toute interprétation $J \in \mathit{sol}$ étend l'interprétation I (c'est-à-dire que $I(x) = J(x)$ pour toutes les variables x qui apparaissent dans I) et l'interprétation J définit toutes les variables de Δ et rend vraies toutes les clauses de Δ.
4. Montrer que les règles de DPLL sont complètes. C'est-à-dire que si $\texttt{DPLL}(I, \Delta, \emptyset)$ est vrai alors toute interprétation J qui étend l'interprétation I et définit toutes les variables de Δ rend fausse une des clauses de Δ.
5. Montrer que les règles de DPLL terminent. C'est-à-dire que pour tout I et Δ, il existe *sol*, tel que $\texttt{DPLL}(I, \Delta, \mathit{sol})$ est vrai (trouver une quantité qui diminue le long des branches de l'arbre).
6. En déduire un méthode pour trouver un modèle d'une formule propositionnelle quelconque.

3 Aide

Cette section contient quelques suggestions pour bien réaliser votre projet.

3.1 Aide Coq

- Les entiers relatifs sont définis dans le module `ZArith` comme le type `Z`.
- Pour représenter les variables, on peut utiliser n'importe quel type de données infini avec une comparaison décidable et une notation en entrée commode pour les tests. On peut réutiliser les entiers, ou encore le module `Strings` de chaînes de caractères.
- Les ensembles peuvent être implantés soit en utilisant la bibliothèque `Lists.ListSet`, soit en utilisant le module plus avancé `MSets`.