L2/LD — TP 6 — Résolution propositionnelle

L'objectif de ce TP est d'implanter la *résolution propositionnelle* en OCaml, selon l'algorithme de Davis-Putnam (1960). Soit F une formule en CNF et S l'ensemble des variables (= propositions atomiques) de F. L'algorithme de Davis-Putnam détecte une contradiction dans l'ensemble de clauses propositionnelles de F en appliquant la résolution propositionnelle variable par variable, comme suit :

Pour toute variable v dans S
  Pour toute clause c de F contenant le littéral v
    Pour toute clause d contenant le littéral $\neg v$
      Ajouter à F la résolvante de c et d
    Fin Pour
  Fin Pour
  Retirer de F toutes les clauses contenant v ou $\neg v$
Fin Pour

La formule F est satisfaisable si et seulement si, au cours des résolutions, on n'obtient jamais la clause vide.

Lisez cet énoncé jusqu'à la fin avec de commencer à répondre aux questions. Ensuite, avec les types de données et les fonctions du TP 5, implantez et testez l'algorithme de Davis-Putnam en OCaml, en suivant les étapes des questions suivantes.

Travail à rendre

Vous devez rendre dans un unique fichier source vos réponses à tous les exercices de ce sujet. Le nom du fichier doit être composé des 8 premières lettres de votre nom de famille, sans accents. Le fichier doit être complet et exécutable sans modification (lourdes pénalités sinon). Ce fichier doit contenir un commentaire indiquant vos nom, prénom et groupe de TP, le code OCaml demandé et des commentaires explicatifs.

L'évaluation de ce travail compte pour la note de TP du module. Le travail doit être strictement personnel. Toute ressemblance excessive entre travaux rendus sera sévèrement sanctionnée.

Déposez votre fichier sur le site `http://moodle.univ-fcomte.fr/course/view.php?id=2594` du module sur moodle, dans le devoir intitulé "Rendu du TP 6 (groupes du lundi)" ou "Rendu du TP 6 (groupes du vendredi)", selon votre groupe, en respectant les délais et les autres consignes indiquées sur le site.

1 Fonctions auxiliaires

1. Définir une fonction `listVarFormula : cnf -> string list` qui calcule la liste des variables d'une formule en CNF.
2. Définir une fonction qui élimine toutes les clauses qui contiennent une variable et sa négation, dans toute formule en CNF de type `cnf`.

2 Codage de l'algorithme

1. Définir une fonction

```
addResol : string * literal list * literal list * clause list -> clause list
```

telle que `addResol(v,c,d,l)` ajoute au début de la liste de clauses `l` le résultat de la résolution des clauses `c` et `d` selon la variable v, s'il existe.

2. Définir une fonction récursive

```
dpVarLitListClauseList : string * literal list * clause list -> clause list
```

correspondant à la troisième boucle de l'algorithme de Davis-Putnam (la boucle la plus interne) : l'appel `dpVarLitListClauseList(v,c,m)` doit appeler la fonction précédente pour chaque clause de la liste de clauses `m`. Remarque : on peut se limiter aux clauses qui contiennent le littéral `Neg(v)`.

3. Définir une fonction récursive `dpVarClauseList` correspondant à la deuxième boucle de l'algorithme de Davis-Putnam : l'appel `dpVarClauseList(v,l,m)` doit parcourir la liste de clauses `l` et appeler la fonction précédente pour chacun de ses éléments (contenant le littéral `Pos(v)`).
4. Définir une fonction récursive `dpClauseList` telle que `dpClauseList(s,l,m)` applique l'algorithme de Davis-Putnam entre toutes les clauses des listes de clauses `l` et `m`, en appelant la fonction précédente pour chaque variable de la liste de variables `s`. Cette fonction correspond à la boucle externe de l'algorithme.
5. Définir une fonction récursive `dpFormula` telle que `dpFormula(f,l)` applique l'algorithme de Davis-Putnam à la formule `f : cnf` dont la liste de variables est `l`. Si cette liste est non vide, la fonction appelle la fonction précédente avec le premier élément de la liste `l`.

   Indication : Il peut être nécessaire de modifier les fonctions précédentes, pour satisfaire les aspects restants de l'algorithme de Davis-Putnam :
   — les nouvelles clauses produites doivent être ajoutées à la liste des clauses connues,
   — les clauses contenant une variable déjà traitée doivent être retirées.
6. Définir une fonction `dp : cnf -> cnf` qui calcule la liste de variables de toute formule en CNF puis applique la fonction `dpFormula`.
7. Appliquer cette fonction à $\{p_1 \vee p_2, \neg p_2 \vee \neg p_3, p_3, \neg p_1\}$, et à d'autres exemples de votre choix, pour vous assurer de sa correction.