TP 9 - Formules du calcul propositionnel

5 décembre 2022

1 Formules Logiques

Dans cette section on se propose d'implémenter les formules du calcul propositionnel. On peut utiliser pour cela le type suivant :

```
type formula = Var of int | Not of formula | Or of formula * formula
```

1. Étendez le type formula avec les connecteurs `And` et `Imply` et les valeurs $\top$ et $\bot$.
2. Définissez les formules $f_1 = p_4 \to ((p_3 \to p_2) \to p_1)$ et $f_2 = (p_1 \wedge (p_2 \to p_3)) \vee \bot$.
3. Implémentez une fonction `formula_height: formula -> in` qui calcule la hauteur d'une formule passée en argument.
4. Implémentez une fonction `size: formula -> int` qui calcule la taille d'une formule passée en argument.
5. Définissez un type `valuation` qui est une liste de couples d'entiers, à l'identifiant d'une variable propositionnelle et de booléens correspondant à la valeur de vérité prise par la variable propositionnelle.
6. Implémentez une fonction `get_value: valuation -> int -> bool` qui renvoie la valeur de vérité d'une variable étant donné une valuation et l'identifiant de cette variable. On pourra choisir de renvoyer `false` par défaut quand la valuation ne précise pas la valeur de la variable.
7. Implémentez une fonction `evaluate` qui prend une formule et une valuation et renvoie la valeur de vérité de cette formule.
8. Évaluez f_1 et f_2 avec la valuation $p_1 = 1, p_2 = 0, p_3 = 0$
9. Implémentez une fonction `variable_list: formula -> int list` qui liste toutes les variables d'une formule sans doublon.
10. Implémentez une fonction `generate_all_valuations: int list -> valuation list` qui génère toutes les valuations pour un ensemble de variables.
11. Implémentez une fonction `is_satisfiable: formula -> bool` qui vérifie si une formule est satisfiable
12. Implémentez une fonction `is_contradiction: formula -> bool` qui vérifie si une formule est une antilogie. Vous pourrez vous servir de la fonction `is_satisfiable`.
13. Implémentez une fonction `is_tautology: formula -> bool` qui vérifie si une formule est une tautologie. Vous pourrez vous servir de la fonction `is_satisfiable`.
14. Testez ces fonctions sur f_1 et f_2
15. Implémentez une fonction `is_equivalent: formula -> formula -> bool` qui teste si deux formules sont sémantiquement équivalentes.
16. (Bonus) Implémentez une fonction `print_truth_table: formula -> unit` qui affiche la table de vérité d'une formule (unit est le type de retour de `Printf.printf`). On pourra s'appuyer sur plusieurs fonctions intermédiaires pour ceci. Testez cette fonction sur f_1 et f_2 et $f_3 = (p_6 \to p_5) \vee (p_3 \wedge (p_4 \vee p_2)) \vee (p_1 \wedge p_5)$.
17. Implémentez une fonction `subformulas -> formula -> formula list` qui extrait les sous formules d'une formule.
18. Testez cette fonction sur f_1 et f_2.
19. Implémentez une fonction `substitute: formula -> int -> formula -> formula` qui substitue une variable dans une formule par une autre formule.
20. Substituez p_4 par $\bot$ dans f_1.
21. (Bonus) Implémentez une fonction `instant_substitute: formula -> int list -> formula list -> formula` qui substitue une liste de variables sans doublons dans une formule chacune par une autre formule. Substituez dans f_3 p_i par $p_{i=1 mod 7}$.
22. Implémentez une fonction `simplify : formula -> formula` qui simplifie une formule en remplaçant les sous formules sans variables par $\bot$ ou $\top$ selon la valeur de vérité de la sous-formule.
23. (Bonus) Implémentez l'algorithme de Quine. Comparez le temps d'exécution de celui-ci avec la fonction `is_satisfiable` pour des formules de plus en plus grandes.
24. (Bonus) Implémentez une fonction `to_CNF : formula -> formula` qui transforme une formule quelconque en formule en forme normale conjonctive équivalente.
25. (Bonus) Implémentez une fonction `to_DNF : formula -> formula` qui transforme une formule quelconque en formule en forme normale disjonctive équivalente.