Projet de raisonnement propositionnel

Edouard.H Théo.R.V

2022–2023

Résumé

Ce document constitue notre compte rendu du projet de raisonnement propositionnel traitent d'une implémentation de la méthode des tableaux et de fonction de test. Il s'agit d'une méthode permettant de déterminer la satisfesabiliter d'une formule booléenne. Les fonctions de test permettent de vérifier empiriquement l'implémentation de la méthode.

Dans un premier temps, nous présenterons notre implémentation de la méthode des tableaux et des divers choix présent dans celle-ci. Pour finir, nous aborderons les difficultés liées à la production de ce projet.

Table des matières

I Implémentation de la méthode des tableaux

1 Formule

Tout d'abord, nous avons dès le début de la conception de ce projet mis en place les opérateurs étendus vus en TD. On peut notamment citée les opérateurs, **xor**, **nor**, **nand**, **diff** et **equiv** de symbole respectif $\oplus$, $\uparrow$, $\downarrow$, $\setminus$ et $\Leftrightarrow$.

La fonction **string_of_formule** est une fonction récursive non terminale, ce choix est dû à plusieurs raisons. La première étant que sa complexité espace sera logarithmique par la division de la formule par deux a chaque appel récursif. La deuxième raison réside dans l'hypothèse que les formule afficher ne devrais pas être d'une taille démesurée enfaite une formule trop grande ne pourrait pas être affichée sur un terminal de taille standard. De plus ce choix favorise la compréhension de cette fonction.

En ce qui concerne la fonction **eval** ayant déjà implémenté la partie de 'base' en TP tout le travail, c'est focaliser sur les opérateurs étendus. Ces opérations n'étant pas implémenté de façon standard en Ocaml, nous avons donc à partir des tables de vérités mises en place des formules équivalente permettant alors leur implémentation, voir table 1 pour les formules équivalente résultant des tables de vérités.

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0
$(a \setminus b) \vee (b \setminus a)$		

a	b	$a \uparrow b$
0	0	1
0	1	1
1	0	1
1	1	0
$\neg(a \wedge b)$		

a	b	$a \downarrow b$
0	0	1
0	1	0
1	0	0
1	1	0
$\neg(a \vee b)$		

a	b	$a \setminus b$
0	0	0
0	1	1
1	0	1
1	1	0
$a \vee b$		

a	b	$a \Leftrightarrow b$
0	0	0
0	1	1
1	0	1
1	1	0
$(a \rightarrow b) \wedge (b \rightarrow a)$		

TABLE 1 – Tables de vérités et formule équivalente.

2 RandomFormule

Pour la fonction **random_form**, nous avons fais le choix de la divisé en plusieur sous-fonction. Un tel choix permet une compréhension plus aisé du traitement effectué dans la fonction. Les quatres fonctions **random_atome**, **random_n_operator**, **random_u_operator** et **random_b_operator** correponde chacune à l'obtention d'un élément de façon pseduo-aléatoire, un atome, un opérateurs nullaire, unaire et binaire respectivement au fonction sité plus haut. Une telle division permet aussi de facilité un possible futur ajout d'opérateurs, d'où la création d'une fonction pour obtenir un opérateur unaire de façon alétoire bien qu'il n'aille qu'un seul. Toutes les autres fonction obtients l'element cible en tirant un nombre pseudo-aléatoire à l'aide du module **Random**.

3

4 Test

La fonction **to_alea_inter**, a été implémentée à l'aide d'une fonction auxiliaire qui sera renvoyer à l'utilisateur partiellement exécuter (sans tous ces paramètres, pour renvoyer une fonction de type interprétation).

II Les difficultés rencontrées

III Conclusion