

TP n°2

Interprétation d'une formule, satisfaisabilité

Dans ce TP on va évaluer l'interprétation d'une formule par rapport à une affectation et ensuite, on va tester la satisfaisabilité d'une formule.

En cours, on a vu qu'une *affectation* est une fonction qui à chaque variable associe une valeur booléenne. Ici, on va représenter une affectation σ par un *tableau* `S` de booléens. La valeur de `S[i]` sera la valeur de $\sigma(x_i)$. Un tableau est nécessairement fini. Pour tous les indices $i \geq S.length$ on considérera que la valeur est `false` par défaut.

Exercice 1 Dans la classe `Formule`, écrivez une méthode statique qui permet de créer une affectation à l'aide du clavier.

Exercice 2 Ajouter à la classe `Formule` une méthode `boolean eval(boolean[] S)` qui prend un tableau de booléen `S` (représentant une affectation) et qui renvoie `true` si la formule `this` est vraie par rapport à l'affectation représentée par `S`, et sinon elle renvoie `false`.

Exercice 3 Testez votre programme

Satisfaisabilité L'objectif est d'écrire un programme capable de décider si une formule est satisfaisable. Il s'agit donc, pour une formule donnée, de l'évaluer avec toutes les affectations possibles et de voir si l'une de ces évaluations renvoie 1. Évidemment, si \mathcal{V} est infini, il est impossible d'énumérer toutes les affectations possibles. Heureusement, si on s'intéresse à une formule f , le Théorème 4 du cours nous assure qu'il suffit de regarder les affectations dont le support est contenu dans l'ensemble des variables de f . Si n est le plus grand numéro d'une variable qui apparaît dans f , on pourra représenter chacune des affectations qui nous intéressent par un tableau booléen de longueur n .

(Remarque : de cette façon on pourrait décrire aussi des affectations dont le support n'est pas contenu dans l'ensemble des variables de f - vous voyez pourquoi ? En tous cas cela n'est pas grave, l'important c'est de décrire *au moins* toutes les affectations dont le support est contenu dans l'ensemble des variables de f .)

Exercice 4 Dans la classe `Formule`, créez une méthode `maxVar` qui renvoie le plus grand numéro de variable qui apparaît dans la formule.

Exercice 5 Dans la classe `Formule`, programmez une méthode `boolean[] initAffectation()` qui renvoie un tableau de booléens initialisé à `false`, dont la longueur est égale au numéro de la plus grande variable dans `this`.

Soit n le plus grand numéro de variable qui apparaît dans la formule. Nous allons juger si une formule est satisfaisable en parcourant toutes les affectations dont le support est contenu dans $\{x_0, \dots, x_n\}$, jusqu'à ce que nous en trouvions une qui satisfait la formule. Il faut énumérer tous les tableaux booléens de longueur n . Un tableau de booléen peut être vu comme un entier binaire : Le tableau `[false, true, true]` serait à lire comme le nombre binaire 011, le tableau

`[true, true, false]`, à lire comme 110, etc. Pour énumérer tous les tableau de longueur $n + 1$, il suffit donc d'énumérer les entiers entre 0 et $2^{n+1} - 1$. Dans quel ordre va-t-on les énumérer ? Mais bien sûr dans l'ordre croissant. Étant donné un tableau de booléen, pour trouver le tableau suivant, il suffit d'ajouter 1 en binaire.

Exercice 6 Programmez une méthode (éventuellement statique)
`boolean affSui vante(boolean[] t)`