TP n°3 - Révisions de logique

Exercice 1 CCINP 2015

De nombreux travaux sont réalisés en Intelligence Artificielle pour construire un programme qui imite le raisonnement humain et soit capable de réussir le test de Turing, c'est-à-dire qu'il ne puisse pas être distingué d'un être humain dans une conversation à l'aveugle. Vous êtes chargé(e)s de vérifier la correction des réponses données par un tel programme lors des tests de bon fonctionnement.

Dans le scénario de test considéré, **le comportement attendu** est le respect de la règle suivante : pour chaque question, le programme répondra par trois affirmations dont une seule sera correcte.

Nous noterons A_1, A_2, A_3 les formules propositionnelles associées aux affirmations répondues par le programme (on explicite leur écriture plus tard).

1. Représenter le comportement attendu sous la forme d'une formule du calcul des propositions qui dépend de A_1, A_2 et A_3.

Dans un premier temps on demande au programme : Quels éléments doivent contenir les aliments que je dois consommer pour préserver ma santé ?

Il répond les affirmations suivantes :

A_1 : Consommez au moins des aliments qui contiennent des glucides, mais pas de lipides !

A_2 : Si vous consommez des aliments qui contiennent des glucides, alors ne consommez pas d'aliments qui contiennent des lipides !

A_3 : Ne consommez aucun aliment qui contient des lipides !

Nous noterons G, respectivement L, les variables propositionnelles qui correspondent au fait de consommer des aliments qui contiennent des glucides, respectivement des lipides.

2. Exprimer A_1, A_2 et A_3 sous la forme de formules du calcul des propositions. Ces formules peuvent dépendre des variables G et L.
3. À partir de la question 1 et en utilisant le calcul des propositions, déterminer ce que doivent contenir les aliments que vous devrez consommer pour préserver votre santé.

   Remarque : cette question signifie qu'il faut utiliser des équivalents sémantiques pour simplifier la formule, comme par exmeple les lois de De Morgan, les propriétés de distributivité et les propriétés de $\top$ et $\bot$.

Dans un second temps, on demande au programme : Quelles activités dois-je pratiquer si je veux préserver ma santé ?

Les réponses sont les suivantes, mais suite à une coupure de courant, la dernière affirmation est interrompue.

A_1 : Ne faites des activités sportives que si vous prenez également du repos !

A_2 : Si vous ne faites pas d'activité intellectuelle, alors ne prenez pas de repos !

A_3 : Prenez du repos ou faites des activités ...

Nous noterons S, I et R les variables propositionnelles qui correspondent au fait de faire des activités sportives, des activités intellectuelles et de prendre du repos.

4. Exprimer A_1, A_2 et A_3 sous la forme de formule du calcul des propositions. Ces formules peuvent dépendre de S, I et R.
5. En utilisant une table de vérité, déterminer quelle(s) activité(s) vous devez pratiquer pour préserver votre santé.

Exercice 2 Formes normales

On considère la formule $\varphi = ((a \Rightarrow \neg b) \Rightarrow \neg a) \wedge c$.

1. Établir la table de vérité de φ.
2. Proposer une forme normale disjonctive de φ.
3. Proposer une forme normale conjonctive de φ.

Exercice 3 En Ocaml

1. Proposer un type OCaml pour représenter les formules logiques (avec les constructeurs et, ou, non, implique et équivaut). Les variables seront représentées par un nom de type `string`.
2. Représenter les arbres syntaxiques des formules logiques suivantes, puis les coder en OCaml.

(a) $f_1 : a \vee (b \wedge c)$,
(b) $f_2 : (a \wedge \neg b) \vee (b \wedge \neg(c \vee a))$,
(c) $f_3 : \neg a \vee (a \Rightarrow b)$,

On va représenter les valuations de nos variables par une liste de couples (nom de variable, valeur booléenne).

3. Écrire une fonction `evalue` qui, étant données une valuation et une formule, évalue la formule pour cette valuation.
4. Écrire une fonction `variables` qui, étant donné une formule, renvoie la liste des variables qui apparaissent à l'intérieur. Attention : on ne veut pas de doublons !

Pour énumérer les valuations, on définit un ordre dessus par analogie aux nombres binaires. On considère **`False`** comme 0 et **`True`** comme 1. La première valuation à 3 variables a, b, c est donc **`[“a”, false; “b”, false; “c”, false]`** (comme 000), la deuxième **`[“a”, true; “b”, false; “c”, false]`** (comme 001), la troisième **`[“a”, false; “b”, true; “c”, false]`** (comme 010) et la dernière **`[“a”, true; “b”, true; “c”, true]`** (comme 111).

5. Écrire une fonction `next` qui, étant donné une valuation, renvoie la prochaine valuation.
6. Écrire une fonction `bruteforce` (d'une formule) qui trouve une valuation qui satisfait la formule.
7. Écrire une fonction `bruteforce_nb` (d'une formule) qui compte le nombre de valuations qui satisfont la formule.
8. Écrire une fonction `tautologie` (d'une formule), qui décide si la formule est une tautologie
9. Écrire une fonction `pretty_print` (d'une valuation), qui affiche la valuation de manière jolie. Exemple d'exécution de la fonction :

```
pretty_print [``a'', false; "b", true]
a --> 0  b --> 1
-: unit=()
```

10. Améliorer la fonction `tautologie` pour qu'elle renvoie un contre-exemple, c'est à dire une valuation telle que la formule ne s'évalue pas à vrai. Indice : on peut utiliser une exception.
11. Écrire une fonction `forme_normale` qui met une formule sous forme normale conjonctive.