Évaluation de S4-UE4
« Fondement de l'informatique 2 »

igor.stephan@univ-angers.fr

À rendre par mail pour le 15 mai 2020

1 Conséquence sémantique en C

L'évaluation de « Fondement de l'informatique 2 » en première session est un projet qui a pour but de faire programmer en langage C une procédure de décision qui détermine si une formule est conséquence sémantique d'un ensemble de formules. Le programme devra être correct (quand il répond, il répond correctement) et complet (il répond toujours). De plus, les programmes seront comparés les uns aux autres sur des exemples pour certains fournis (tel que le problème 2 de coloration ou le problème 3 du Wumpus) et d'autres, non. Pour pouvoir participer à la compétition, le code doit pouvoir s'exécuter sur un Ubuntu. (Si vous n'incluez pas de librairies spécifiques à un OS particulier, le programme devrait se compiler et s'exécuter sans problème.)

2 Représentation des formules

Un ensemble de formules est représenté sous la forme d'un tableau de triplets :

```
static int ensemble_de_formules[][3]
```

Le nombre de formules de cet ensemble ainsi que le nombre de symboles propositionnels (ou variables de formule) présents dans ces formules sont donnés par

```
#define NB_SYMBOLES 3
#define NB_FORMULES 4
```

Les `NB_FORMULES` premières lignes d'un tel tableau contiennent les racines des formules. Ainsi pour un ensemble de formules

$$\{((A \vee B) \rightarrow C), \neg C, (\neg A \wedge \neg C), A\}$$

le début du tableau est (par exemple) :

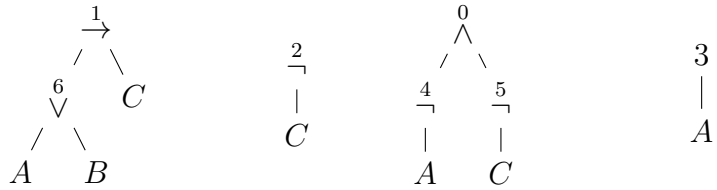
```
{{ET, 4, 5}, // 0
 {IMP, 6, _C_}, // 1
 {NEGATION, _C_, VIDE}, // 2
 {FAIT, _A_, VIDE}, // 3
...}
```

Il y a quatre formules : les nombres 0, 1, 2 et 3 représentent donc respectivement les racines des formules $(\neg A \wedge \neg C)$, $((A \vee B) \rightarrow C)$, $\neg C$ et A. Les nombres 4, 5, et 6 représentent des indices dans le tableau pour indiquer où sont les racines des sous-formules : à l'indice 4 du tableau se trouve la sous-formule $\neg A$, à l'indice 5 la sous-formule ¬C et à l'indice 6 la sous-formule $(A \vee B)$. Les symboles `_A_`, `_B_` et `_C_` font référence aux variables de formule A, B et C. L'opérateur de negation étant unaire et le tableau ayant 3 colonnes, la dernière est neutralisée par le symbole `VIDE`. Le symbole `FAIT` permet de représenter un fait (ie. une formule constituée d'un symbole propositionnel seul ou d'une variable de formule seule) ; de même que pour la négation, la dernière colonne est neutralisée par le symbole `VIDE`. Les symboles qui peuvent prendre la première position des triplets sont donc : `ET` qui représente la conjonction, `IMP` qui représente l'implication, `NEGATION` qui représente la négation (et qui a nécessairement son troisième attribut à `VIDE`) et `FAIT` lorsque la formule est réduite à un symbole propositionnel ou une variable de formule (et qui a nécessairement son troisième attribut à `VIDE`) ; auxquels sont rajoutés `OU` qui représente la disjonction et `EQU` qui représente l'équivalence logique.

Ainsi l'ensemble de formules

$$\{((A \vee B) \rightarrow C), \neg C, (\neg A \wedge \neg C), A\}$$

de représentation arborescentes



est encodé par

```
#define _A_ -1
#define _B_ -2
#define _C_ -3

#define NB_SYMBOLES 3
#define NB_FORMULES 4

static int ensemble_de_formules[][3] =
       {{ET, 4, 5},  // 0
        {IMP, 6, _C_}, // 1
        {NEGATION, _C_, VIDE}, // 2
        {FAIT, _A_, VIDE}, // 3
        {NEGATION, _A_, VIDE}, // 4
        {NEGATION, _C_, VIDE}, // 5
        {OU, _A_, _B_}}; // 6
```

Une fonction qui permet d'afficher les symboles propositionnels ou variables de formule (qui sont systématiquement strictement négatifs pour ne pas les confondre avec les indices dans le tableau) est fournie :

```
void affiche_symbole(int _symbole) {
  switch(_symbole) {
  case _A_ : printf("A"); break;
  case _B_ : printf("B"); break;
  case _C_ : printf("C"); break;
  }
}
```

Une conséquence sémantique $\Sigma \models F$ est traitée exactement comme l'ensemble de formules $\Sigma \cup F$: *la formule F a alors sa racine à l'indice 0 du tableau*. Le codage présenté ci-dessus est donc aussi le codage de

$$\{((A \vee B) \rightarrow C), \neg C, A\} \models (\neg A \wedge \neg C)$$

Le programme devra afficher 1 s'il y a conséquence sémantique et 0 sinon.

3 Un cadre technique

L'ensemble suivant de fichiers C sont fournis :

— Un fichier `examen.c` qui contient du code pour afficher sur un terminal la conséquence sémantique représentée par le code du paragraphe 2 (celui-ci étant inséré par la directive `#include "exemple.h"`).

— Un fichier `examen.h` qui définit les symboles `VIDE`, `ET`, `IMP`, `NEGATION`, `FAIT`, `OU` et `EQU`.

— Un fichier `exemple.h` qui reprend l'exemple du paragraphe 2.

— Un ensemble de fichiers de test qui sont extraits des exemples, exercices et problèmes du cours.

L'unique ligne de compilation est :

```
gcc examen.c
```

Je reste à votre disposition pour répondre à vos questions principalement par mail.