Modélisation et Architecture – Session 1, 3 et 5 – 25/01/2023

Bureau d'études – 1h30 – Documents électroniques autorisés (contenu sur moodle et résultats des Travaux Pratiques)

- Télécharger depuis moodle l'archive `be_2022_S1.tgz`
- Désarchiver son contenu avec la commande : `tar xzvf be_2022_S1.tgz`
- Vous obtenez un répertoire nommé `be_2022_S1`
- Renommer ce répertoire sous la forme `be_2022_S1_Nom` (en remplaçant Nom par votre nom). Par exemple, si vous êtes Jacques Requin, vous utiliserez la commande : `mv be_2022_S1 be_2022_S1_Requin`
- Compiler la bibliothèque avec la commande : `coqc Naturelle.v`
- En fin de séance, vous rendrez sur moodle l'archive contenant le répertoire renommé.

Exercice 1 Soient A, B et C des variables propositionnelles, vous devrez montrer avec l'outil Coq (fichier `coq-exercice-1.v`) **en utilisant la déduction naturelle constructive (c'est-à-dire sans les règles du tiers-exclu (TE dans le résumé de cours) et de l'absurde classique (A dans le résumé de cours))** que la formule suivante est un théorème :

$$((\neg A) \vee (\neg B)) \rightarrow (\neg(A \wedge B))$$

1. Avec les commandes de la bibliothèque `Naturelle` utilisée en travaux pratiques
2. Avec les commandes classiques (sans la bibliothèque `Naturelle` utilisée en travaux pratiques, c'est-à-dire les commandes `intro`, `elim`, `exact`, `cut`, `split`, `left`, `right`, `destruct`, `absurd` et `apply`)

Exercice 2 Soient A, B et C des variables propositionnelles, vous devrez montrer, avec l'outil Coq (fichier `coq-exercice-2.v`) **en utilisant la déduction naturelle classique (y compris les règles du tiers-exclu ou de l'absurde classique)**, que la formule suivante est un théorème :

$$(\neg(A \wedge B)) \rightarrow ((\neg A) \vee (\neg B))$$

1. Avec les commandes de la bibliothèque `Naturelle` utilisée en travaux pratiques
2. Avec les commandes classiques (sans la bibliothèque `Naturelle` utilisée en travaux pratiques, c'est-à-dire les commandes `intro`, `elim`, `exact`, `cut`, `split`, `left`, `right`, `destruct`, `absurd`, `apply` et `classic`)

Exercice 3 Nous considérons la spécification des listes munie des équations étudiées en cours et travaux dirigés :

(a) $\forall l \in \texttt{liste}(A).\ \texttt{append}(\texttt{Nil},l) = l$
(b) $\forall t \in A. \forall l, q \in \texttt{liste}(A).\ \texttt{append}(\texttt{Cons}(t,\ q),l) = \texttt{Cons}(t,\ \texttt{append}(q,l))$
(c) $\forall l \in \texttt{liste}(A).\ \texttt{append}(l,\texttt{Nil}) = l$
(d) $\forall l_1, l_2, l_3 \in \texttt{liste}(A).\ \texttt{append}(l_1,\texttt{append}(l_2,l_3)) = \texttt{append}(\texttt{append}(l_1,l_2),l_3)$

Nous complétons cette spécification par la fonction $\texttt{snoc}(l,\ e)$ qui ajoute l'élément e à la fin de la liste l.
Le comportement de cette fonction peut être modélisé par les équations suivantes :

(e) $\forall e \in A.\ \texttt{snoc}(\texttt{Nil},\ e) = \texttt{Cons}(e,\ \texttt{Nil})$
(f) $\forall e \in A. \forall t \in A. \forall q \in \texttt{liste}(A).\ \texttt{snoc}(\texttt{Cons}(t,\ q),\ e) = \texttt{Cons}(t,\ \texttt{snoc}(q,\ e))$

Spécifier cette fonction `snoc_spec` dans l'outil Coq (fichier `coq_exercice_3.v`) sous la forme d'axiomes puis montrer que cette fonction satisfait les propriétés suivantes :

(g) $\forall e \in A. \forall l \in \texttt{liste}(A).\ \texttt{snoc}(l,\ e) = \texttt{append}(l,\texttt{Cons}(e,\ \texttt{Nil}))$
(h) $\forall e \in A. \forall l_1 \in \texttt{liste}(A). \forall l_2 \in \texttt{liste}(A).\ \texttt{snoc}(\texttt{append}(l_1,l_2),\ e) = \texttt{append}(l_1,\texttt{snoc}(l_2,\ e))$

Programmer une implantation de la fonction `snoc_impl` puis prouver que cette implantation est correcte vis-à-vis de la spécification `snoc_spec` (théorème `snoc_correctness`).

Exercice 4 Prouver la correction totale du triplet de Hoare suivant (fichier `why3_exercice_4.mlw`) pour un programme calculant la multiplication de deux entiers relatifs. Nous vous suggérons d'exploiter $\mathtt{S} = \mathtt{A} \times \mathtt{I}$ comme invariant et $\mathtt{B} - \mathtt{I}$ comme variant. Vous compléterez l'invariant si nécessaire pour construire la preuve. Vous indiquerez dans le fichier les commandes WHY3 utilisées pour construire la preuve.

$\{\}$

```
if (B < 0) then
        B := - B;
        A := - A
else
        skip
fi
S := 0;
I := 0;
while (I < B) do
        S := S + A;
        I := I + 1;
od
```

$\{\mathtt{S} = \mathtt{A} \times \mathtt{B}\}$