# Modélisation et Programmation – Session 1 – 26/11/2018

Bureau d'études – 1h45 – Documents électroniques autorisés

| Prénom : | Nom : | Groupe : |
|---|---|---|

— Télécharger depuis moodle l'archive `source-etudiants.tgz`
— Désarchiver son contenu avec la commande : `tar xzvf source-etudiants.tgz`
— Vous obtenez un répertoire nommé `source-etudiants`
— Renommer ce répertoire sous la forme `source-etudiants_Nom1_Nom2` (en remplaçant Nom1 et Nom2 par le nom des deux membres du binôme par ordre alphabétique). Par exemple, si les membres sont Xavier Thirioux et Marc Pantel, vous utiliserez la commande : `mv source-etudiants source-etudiants_Pantel_Thirioux`
— 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 :

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

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)

**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)

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

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 :
— `snoc_alternative` : $\forall e \in A.\forall l \in \texttt{liste}(A).\ \texttt{snoc}(e,\ l) = \texttt{append}(l,\texttt{Cons}(e,\ \texttt{Nil}))$
— `snoc_append` : $\forall e \in A.\forall l_1 \in \texttt{liste}(A).\forall l_2 \in \texttt{liste}(A).\ \texttt{snoc}(e,\ \texttt{append}(l_1,l_2)) = \texttt{append}(l_1,\texttt{snoc}(e,\ l_2))$

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 le produit de deux entiers positifs `A` et `B`. Nous vous suggérons d'exploiter $\texttt{R} = \texttt{A} \times (\texttt{B} - \texttt{I})$ comme invariant et `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.

$\{\texttt{A} \geq 0 \wedge \texttt{B} \geq 0\}$

```
R := 0;
I := B;
while (0 < I) do
        R := R + A;
        I := I - 1
od;
```

$\{\texttt{R} = \texttt{A} \times \texttt{B}\}$