TD 6

$$\frac{}{\rho \vdash x := e \Rightarrow \rho[x \mapsto [\![e]\!]\rho]}\ (:=) \qquad \frac{}{\rho \vdash \texttt{skip} \Rightarrow \rho}\ (\texttt{Skip})$$

$$\frac{\rho \vdash c_1 \Rightarrow \rho' \quad \rho' \vdash c_2 \Rightarrow \rho''}{\rho \vdash c_1; c_2 \Rightarrow \rho''}\ (\texttt{Seq})$$

$$\frac{\rho \vdash c_1 \Rightarrow \rho'}{\rho \vdash \texttt{if}\ e\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \Rightarrow \rho'}\ (\texttt{if}_1) \quad \text{si } [\![e]\!]\rho \neq 0 \qquad \frac{\rho \vdash c_2 \Rightarrow \rho''}{\rho \vdash \texttt{if}\ e\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \Rightarrow \rho'}\ (\texttt{if}_2) \quad \text{si } [\![e]\!]\rho = 0$$

$$\frac{\rho \vdash c \Rightarrow \rho' \quad \rho' \vdash \texttt{while}\ e\ \texttt{do}\ c \Rightarrow \rho''}{\rho \vdash \texttt{while}\ e\ \texttt{do}\ c \Rightarrow \rho''}\ (\texttt{while}) \quad \text{si } [\![e]\!]\rho \neq 0 \qquad \frac{}{\rho \vdash \texttt{while}\ e\ \texttt{do}\ c \Rightarrow \rho}\ (\texttt{while}_{\text{fin}}) \quad \text{si } [\![e]\!]\rho = 0$$

FIGURE 1 – La sémantique opérationnelle à grands pas de IMP.

Exercice 1. Prouvez que pour tout environnement ρ, il n'existe pas d'environnement ρ' tel que $\rho \vdash \texttt{while}\ \dot{1}\ \texttt{do skip} \Rightarrow \rho'$.

Exercice 2. Étant donnés deux programmes c_1, c_2, on dit que c_1 et c_2 sont équivalents, noté $c_1 \sim c_2$ ssi pour tout environnements ρ, ρ', $(\rho \vdash c_1 \Rightarrow \rho') \Leftrightarrow (\rho \vdash c_2 \Rightarrow \rho')$

1. Montrer que $\texttt{while}\ e\ \texttt{do}\ c \sim \texttt{if}\ e\ \texttt{then}\ (c;\ \texttt{while}\ e\ \texttt{do}\ c)\ \texttt{else skip}$
2. Quels sont les programmes équivalents à $\texttt{while}\ \dot{1}\ \texttt{do skip}$?

Exercice 3. L'un des objectifs des cours suivants sera d'introduire les outils mathématiques nécessaires pour pouvoir définir une sémantique dénotationnelle de IMP : $[\![c]\!]\rho = \rho'$, ce qui signifie que l'exécution du programme c dans l'environnement ρ mène à l'environnement ρ'. Cette écriture «fonctionnelle» suggère que l'environnement ρ' est entièrement déterminé par c et ρ (bien sûr, puisque nos programmes sont «déterministes»).

1. Montrer cette propriété pour la sémantique opérationnelle : pour tout c, ρ, ρ_1, ρ_2, si $\rho \vdash c \Rightarrow \rho_1$ et $\rho \vdash c \Rightarrow \rho_2$ alors $\rho_1 = \rho_2$.
2. On considère le langage non déterministe donné par la syntaxe suivante :

   $$c ::= \texttt{skip} \mid x := e \mid c; c \mid \texttt{if}\ e\ \texttt{then}\ c\ \texttt{else}\ c \mid \texttt{while}\ e\ \texttt{do}\ c \mid c \vee c$$

   où l'instruction $c_1 \vee c_2$ signifie «exécute c_1 ou exécute c_2, de manière non déterministe».

   Proposez une sémantique opérationnelle de ce langage.

Exercice 4. En cours, vous avez aperçu la distinction entre sémantique dénotationnelle et sémantique opérationnelle des programmes IMP. Cependant, vous n'avez utilisé qu'une sémantique dénotationnelle des expressions arithmétiques (rappelée ci dessous). Dans cet exercice, on s'étudie le cas de deux extensions des opérations arithmétiques :

1. On étend nos expressions arithmétiques par la syntaxe suivante :

   $$e ::= x \mid \dot{n} \mid e \dot{+} e \mid \dot{-}e \mid \dot{f}(e)$$

   Pour interpréter le symbole $\dot{f}$, on suppose disposer d'une fonction **partielle** f des entiers dans les entiers.

$$\frac{}{in(c,x)\cdot P \xRightarrow{in(c,x)} P}\ (\textit{in}) \qquad \frac{}{out(c,e)\cdot P \xRightarrow{out(c,e)} P}\ (\textit{out})$$

$$\frac{P \xRightarrow{\alpha} P'}{P\|Q \xRightarrow{\alpha} P'\|Q}\ (\|_\ell) \qquad \frac{Q \xRightarrow{\alpha} Q'}{P\|Q \xRightarrow{\alpha} P\|Q'}\ (\|_r)$$
$$\text{pour } \alpha \in \{\tau, in(c,x), out(c,x) \mid c \in \mathcal{C}, x \in \mathcal{X}\}$$

$$\frac{P \xRightarrow{in(c,x)} P' \quad Q \xRightarrow{out(c,e)} Q'}{P\|Q \xRightarrow{\tau} P'[x \leftarrow e]\|Q'}\ (\tau_\ell)$$

$$\frac{P \xRightarrow{out(c,e)} P' \quad Q \xRightarrow{in(c,x)} Q'}{P\|Q \xRightarrow{\tau} P'\|Q'[x \leftarrow e]}\ (\tau_r)$$

FIGURE 2 – Une sémantique du π-calcul

(a) Donnez une sémantique opérationnelle à grands pas (inspirez vous de celle pour IMP, vue en cours et rappelée en figure 1) pour ces expressions.

(b) Étendez la sémantique dénotationnelle vue en cours pour ces expressions.

(c) Montrez l'équivalence des deux sémantiques.

(d) Quelle différence essentielle rend la sémantique dénotationnelle pour les expressions arithmétiques beaucoup plus simple que celle pour les programmes ?

2. Expressions arithmétiques avec effets de bords : on étend nos expressions arithmétiques par la syntaxe suivante :

$$e ::= x \mid \dot{n} \mid e \dot{+} e \mid \dot{-}e \mid c\ \texttt{resultis}\ e$$

Intuitivement, pour évaluer l'expression c `resultis` e, on évalue d'abord la commande c, puis on évalue e dans l'environnement obtenu.

(a) Donnez des sémantiques opérationnelle et dénotationnelle pour ces expressions (on supposera disposer d'une sémantique pour les programmes c).

(b) Comment s'évalue le terme $((x := x \dot{+} \dot{1})\ \texttt{resultis}\ x) \dot{+} ((y := x \dot{+} x)\ \texttt{resultis}\ x)$ dans l'environnement $\rho = [x \mapsto 3]$.

(c) Vous comprenez désormais les horreurs que permet d'écrire le C : `T[i++] = i++`.

Exercice 5. Une entrevue avec le π-calcul :
Le π-calcul est un langage théorique utilisé pour décrire des problématiques de concurrence. On suppose disposer d'un ensemble dénombrable de variables $\mathcal{X}$, un ensemble d'expressions $\mathcal{E}$ contenant l'ensemble de variables, et un ensemble fini de canaux de communication $\mathcal{C}$. On s'intéresse à un fragment du π-calcul dont la syntaxe est donnée par la grammaire suivante :

$$P ::= \texttt{skip} \mid in(c,x)\cdot P \mid out(c,e)\cdot P \mid P\|Q$$

où $c \in \mathcal{C}$, $x \in \mathcal{X}$ et $e \in \mathcal{E}$.

Nous gardons ici l'ensemble des expressions parfaitement abstrait, mais on pourrait imaginer que $\mathcal{E}$ est l'ensemble des expressions arithmétiques, comme rencontré jusque là.

L'objectif de cet exercice est de montrer l'équivalence de deux sémantiques du π-calcul. La première est présentée en figure 2. La seconde est donnée ci dessous. Un *multi-ensemble* d'éléments de X est intuitivement un ensemble dans lequel un même élément peut intervenir plusieurs fois. Formellement, cela se représente comme une fonction de X dans $\mathbb{N}$, où à chaque élement $x \in X$ est associé le nombre de copies de x présentes dans le multi-ensemble. Nous travaillons avec des multi-ensembles finis, c'est à dire

que l'ensemble des éléments présents dans le multi-ensemble est fini. Une autre façon de se le représenter : un multi-ensemble est une liste dans laquelle l'ordre des éléments n'importe pas. Nous les noterons comme des ensembles. Ainsi, les multi-ensembles $\{2; 2; 1\}$ et $\{2; 1; 2\}$ sont égaux, mais sont différents du multi-ensemble $\{2; 1\}$.

Étant donnés deux multi-ensembles S et T, l'*union disjointe*, notée $S+T$ correspond au multi-ensemble M tel que pour tout $x \in X$, $M(x) = S(x) + T(x)$. Les règles de la seconde sémantique du π-calcul sont les suivantes :

$$\{P \| Q\} + S \to \{P; Q\} + S$$
$$\{in(c, x) \cdot P; out(c, e) \cdot Q\} + S \to \{P[x \leftarrow e]; Q\} + S$$

1. Énoncer et prouver la propriété d'équivalence des deux sémantiques.