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

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

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

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

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

Exercice 1 :
Calculer la sémantique opérationnelle à grand pas du programme :

$$c \stackrel{\text{def}}{=} x := 0\ ;\ y := 3\ ;\ \texttt{while}\ y\ \texttt{do}\ (x := x + y\ ;\ y := y + (-1)),$$

c'est-à-dire donner un jugement $\rho \vdash c \Rightarrow \rho'$ dérivable et sa dérivation.

Exercice 2 :
Prouver que pour tout environnement ρ, il n'existe pas d'environnement ρ' tel que $\rho \vdash \texttt{while}\ \dot{1}\ \texttt{do skip} \Rightarrow \rho'$ soit dérivable.

Exercice 3 :
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 tous environnements ρ, ρ', $(\rho \vdash c_1 \Rightarrow \rho')$ est dérivable ssi $(\rho \vdash c_2 \Rightarrow \rho')$ est dérivable.

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 $\texttt{while}\ \dot{1}\ \texttt{do skip}$?
3. On appelle contexte un programme qui contient une variable $\square$, correspondant à une commande. La variable peut être utilisée à de multiples reprises.
   (a) Définir formellement la notion de contexte avec une grammaire.
   (b) Soit C un contexte. Montrer que $c_1 \sim c_2$ implique $C[c_1] \sim C[c_2]$.

Exercice 4 :
Considérons les expressions booléennes suivantes :

$$b ::= \texttt{True} \mid \texttt{False} \mid e \dot{=} e \mid e \dot{<} e \mid \dot{\neg} b \mid b \dot{\vee} b \mid b \dot{\wedge} b$$

1. Donner une sémantique opérationnelle à petits pas **très naïve** pour les expressions booléennes.
2. Modifier vos règles pour qu'elles implémentent le "ou" paresseux : lors de l'évaluation de $b_1 \dot{\vee} b_2$, b_2 n'est pas évalué si b_1 s'évalue à $\texttt{True}$. Même chose pour le "et".

3. Modifier vos règles pour qu'elles implémentent l'évaluation parallèle du "ou". Même chose pour le "et".
4. Dans les trois systèmes de déduction obtenu, que peut-on dire du nombre de dérivations d'un triplet $(b, \rho) \to \_$?

Exercice 5 :

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 à grand pas, sans passer par un résultat d'équivalence avec une autre sémantique : 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 ». Proposer une sémantique opérationnelle à grand pas de ce langage.

Exercice 6 :

En cours, vous aver aperçu la distinction entre sémantique dénotationnelle et sémantique opérationnelle des programmes IMP. Cependant, vous n'aver utilisé qu'une sémantique dénotationnelle des expressions arithmétiques. 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.

   (a) Donner une sémantique opérationnelle à grands pas (inspirer vous de celle pour IMP, vue en cours et rappelée en figure 1) pour ces expressions.
   (b) Étender la sémantique dénotationnelle vue en cours pour ces expressions.
   (c) Montrer 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) Donner 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) Comprendre désormais les horreurs que permet d'écrire le C : `T[i++] = i++`.