Programmation 1, partie 2 : Sémantique

Louis Lemonnier
lemonnier@lsv.fr

Voici une compilation de tous les exercices que j'ai pu donner en TD de sémantique.

1 Sémantiques opérationnelles

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

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

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

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

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 $\rho \vdash \texttt{while}\ \dot{1}\ \texttt{do skip} \Rightarrow \rho'$ soit dérivable.

> **Solution :**
> On suppose qu'il en existe un. On remonte les différents arbres possibles (il en a très peu), et on tombe sur une contradiction : la seule preuve que $\rho \vdash \texttt{while}\ \dot{1}\ \texttt{do skip} \Rightarrow \rho'$ est dérivable utilise le fait que $\rho \vdash \texttt{while}\ \dot{1}\ \texttt{do skip} \Rightarrow \rho'$ est dérivable.

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 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 $C[c_1] \sim C[c_2]$.

Solution :

1. Bien faire attention à faire les deux sens de l'équivalence. Chaque sens se base sur le fait que si un jugement est dérivable et ne peut être obtenu que par une seule règle, les jugements nécessaires pour cette règle sont également dérivables.
2. Tous les programmes non dérivables.

Exercice 3 :

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. Donnez une sémantique opérationnelle à petits pas pour les expressions booléennes.
2. Modifiez 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 `True`. Même chose pour le "et".
3. Modifiez 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 4 :

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».
   Proposez une sémantique opérationnelle de ce langage.

Exercice 5 :

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. 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) 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++`.

$$\begin{aligned}
(x := e \cdot C, \rho) &\to (C, \rho[x \mapsto [\![e]\!]\rho]) \\
(\texttt{skip} \cdot C, \rho) &\to (C, \rho) \\
(c_1; c_2 \cdot C, \rho) &\to (c_1 \cdot c_2 \cdot C, \rho) \\
(\texttt{if } e \texttt{ then } c_1 \texttt{ else } c_2 \cdot C, \rho) &\to (c_1 \cdot C, \rho) \quad \text{si } [\![e]\!]\rho \neq 0 \\
(\texttt{if } e \texttt{ then } c_1 \texttt{ else } c_2 \cdot C, \rho) &\to (c_2 \cdot C, \rho) \quad \text{si } [\![e]\!]\rho = 0 \\
(\texttt{while } e \texttt{ do } c \cdot C, \rho) &\to (c \cdot \texttt{while } e \texttt{ do } c \cdot C, \rho) \quad \text{si } [\![e]\!]\rho \neq 0 \\
(\texttt{while } e \texttt{ do } c \cdot C, \rho) &\to (C, \rho) \quad \text{si } [\![e]\!]\rho = 0
\end{aligned}$$

FIGURE 2 – La sémantique opérationnelle à petits pas de IMP.

Théorèmes petit pas

Déterminisme La réduction est déterministe.

Progrès Les seules configurations ne possédant pas de successeur sont de la forme (ε, ρ).

Théorèmes grand pas

Déterminisme L'arbre de dérivation d'un jugement est unique.

Correction S'il existe une dérivation $\rho \vdash c \Rightarrow \rho_\infty$ alors il existe une dérivation $(c \cdot \varepsilon, \rho) \to^* (\varepsilon, \rho_\infty)$.

Adéquation S'il existe une dérivation $(c \cdot \varepsilon, \rho) \to^* (\varepsilon, \rho_\infty)$ alors il existe une dérivation $\rho \vdash c \Rightarrow \rho_\infty$.

Exercice 6 :

Soit c un programme et ρ un environnement. Montrer l'équivalence entre ces deux propositions :

1. Il existe une dérivation infinie de $(c \cdot \varepsilon, \rho)$;
2. Il n'existe pas de ρ_∞ tel que $\rho \vdash c \Rightarrow \rho_\infty$.

Solution :

Exécution infinie $\implies$ pas de dérivation On fait la contraposée

$$\frac{}{(x,\rho) \to_{pp} (\dot{\widehat{\rho(x)}}, \rho)}\ (\texttt{Var}) \qquad \frac{(e_1,\rho) \to_{pp} (e_1',\rho)}{(e_1 \dot{+} e_2,\rho) \to_{pp} (e_1' \dot{+} e_2,\rho)}\ (+_\ell)$$

$$\frac{(e_2,\rho) \to_{pp} (e_2',\rho)}{(\dot{n} \dot{+} e_2,\rho) \to_{pp} (\dot{n} \dot{+} e_2',\rho)}\ (+_r) \qquad \frac{}{(\dot{n} \dot{+} \dot{m},\rho) \to_{pp} (\dot{\widehat{n+m}}, \rho)}\ (+_{\text{fin}})$$

$$\frac{(e,\rho) \to_{pp} (e',\rho)}{(\dot{-}e,\rho) \to_{pp} (\dot{-}e',\rho)}\ (-) \qquad \frac{}{(\dot{-}\dot{n},\rho) \to_{pp} (\dot{\widehat{-n}}, \rho)}\ (-_{\text{fin}})$$

FIGURE 3 – Sémantique opérationnelle à petits pas des expressions arithmétiques.

— S'il existe une dérivation à grand pas de $\rho \vdash c \Rightarrow \rho_\infty$, alors par *adéquation* il y a une dérivation finie et terminale de $(c \cdot \varepsilon, \rho)$ vers $(\varepsilon, \rho_\infty)$

— Comme la réduction $\to$ est *déterministe*, les deux dérivations coïncident

— Comme $(\varepsilon, \rho_\infty)$ ne se réduit pas, il ne peut pas y avoir de run infini.

Pas de dérivation $\implies$ exécution infinie On fait la contraposée

— S'il existe une exécution finie, alors via le théorème de *progrès* nécessairement elle s'arrête sur $(\varepsilon, \rho_\infty)$.

— Le théorème de correction donne une dérivation $\rho \vdash c \Rightarrow \rho_\infty$.

Exercice 7 :

On s'intéresse à une sémantique opérationnelle à petit pas des expressions arithmétiques, en Figure 3.

1. Donnez une preuve de
$$((x \dot{+} (\dot{-}y)) \dot{+} \dot{2}, \rho[x \mapsto 3, y \mapsto 2]) \to_{pp}^* (\dot{3}, \rho[x \mapsto 3, y \mapsto 2])$$
2. Énoncer puis prouver un théorème de progrès.
3. Énoncer puis prouver un théorème de déterminisme.
4. Montrer la correction de la sémantique dénotationnelle.
5. Montrer l'adéquation de la sémantique dénotationnelle.

Solution :

1. La réduction se fait comme suit
$$\begin{aligned}((x \dot{+} (\dot{-}y)) \dot{+} \dot{2}, \rho[x \mapsto 3, y \mapsto 2]) &\to_{pp}^* ((\dot{3} \dot{+} (\dot{-}y)) \dot{+} \dot{2}, \rho[x \mapsto 3, y \mapsto 2]) \\ &\to_{pp}^* ((\dot{3} \dot{+} (\dot{-}\dot{2})) \dot{+} \dot{2}, \rho[x \mapsto 3, y \mapsto 2]) \\ &\to_{pp}^* ((\dot{3} \dot{+} (\dot{-2})) \dot{+} \dot{2}, \rho[x \mapsto 3, y \mapsto 2]) \\ &\to_{pp}^* (\dot{1} \dot{+} \dot{2}, \rho[x \mapsto 3, y \mapsto 2]) \\ &\to_{pp}^* (\dot{3}, \rho[x \mapsto 3, y \mapsto 2])\end{aligned}$$
2. Les seules expressions ne pouvant pas se réduire sont de la forme $(\dot{n}, \rho)$ où n est un entier. Cela se prouve par analyse de cas sur e.
3. Si $(e,\rho) \to_{pp} (e_1,\rho)$ et $(e,\rho) \to_{pp} (e_2,\rho)$ alors $e_1 = e_2$. Par induction puis analyse de cas.

4. On doit montrer que si $[\![e]\!]_\rho = n$ alors $(e, \rho) \to_{pp}^* n$. Pour cela, on raisonne par induction sur l'expression e et analyse de cas. On a besoin d'un lemme intermédiaire qui dit

$$(e_1, \rho) \to_{pp}^* (\dot{n}, \rho) \implies (e_1 \dot{+} e_2, \rho) \to_{pp}^* (\dot{n} \dot{+} e_2, \rho)$$

   Et des variantes de cette chose. Ne faire que les cas de base et le cas de $\dot{+}$ suffit largement à faire comprendre la preuve.
5. On doit montrer que si $(e, \rho) \to_{pp}^* n$ alors $[\![e]\!]_\rho = n$. Pour cela, il suffit de montrer que toutes les règles de réécriture préservent la sémantique des expressions. C'est trivial.

2 Des graphes

Graphes

Un graphe est un couple $\langle V, E \rangle$ où V est un ensemble *fini* et $E \subseteq V \times V$.
On dit que G est un graphe sur X si $V \subseteq X$.

Expressions de graphes

On donne la grammaire abstraite suivante dont les expressions sont notées Expr_X

$$\begin{aligned} e := \ & \mathsf{Empty} \\ \mid \ & \mathsf{V}x \qquad x \in X \\ \mid \ & e \oplus e \\ \mid \ & e \otimes e \end{aligned}$$

On autorisera dans des calculs intermédiaires de la sémantique à petit pas des expressions $\bar{g}$ où g est un graphe. On notera l'ensemble des expressions intermédiaires Expr_X^+.

Contextes d'expressions

On donne la syntaxe suivante pour les contextes d'expression

$$\begin{aligned} F := \ & \square \oplus e \\ \mid \ & g \oplus \square \\ \mid \ & \square \otimes e \\ \mid \ & g \otimes \square \end{aligned}$$

Où g est un graphe.

Exercice 8 :

Nous allons étudier en détail ce langage sur les graphes.

1. Énoncer puis prouver un théorème de progrès sur la sémantique à petit pas.
2. Énoncer puis prouver un théorème de déterminisme sur la sémantique à petit pas.
3. Énoncer un théorème de terminaison, le démontrer.
4. Énoncer un résultat de correction et d'adéquation des deux sémantiques sur les graphes.
5. Démontrer la correction.

$$\frac{}{\mathsf{Empty} \to \langle \emptyset, \emptyset \rangle} \qquad \frac{}{\mathsf{V}x \to \langle \{x\}, \emptyset \rangle} \qquad \frac{e \to e'}{F[e] \to F[e']}$$

$$\frac{g_1 = \langle V_1, E_1 \rangle \quad g_2 = \langle V_2, E_2 \rangle \quad g_3 = \langle V_1 \cup V_2, E_1 \cup E_2 \rangle}{\bar{g_1} \oplus \bar{g_2} \to \bar{g_3}}$$

$$\frac{g_1 = \langle V_1, E_1 \rangle \quad g_2 = \langle V_2, E_2 \rangle \quad g_3 = \langle V_1 \cup V_2, E_1 \cup E_2 \cup V_1 \times V_2 \rangle}{\bar{g_1} \otimes \bar{g_2} \to \bar{g_3}}$$

FIGURE 4 – Sémantique à petits pas

$$[\![\mathsf{V}x]\!] \triangleq \langle \{x\}, \emptyset \rangle \tag{1}$$

$$[\![\mathsf{Empty}]\!] \triangleq \langle \emptyset, \emptyset \rangle \tag{2}$$

$$[\![e_1 \oplus e_2]\!] \triangleq \langle V_1 \cup V_2, E_1 \cup E_2 \rangle \qquad \text{si } [\![e_1]\!] = \langle V_1, E_1 \rangle \text{ et } [\![e_2]\!] = \langle V_2, E_2 \rangle \tag{3}$$

$$[\![e_1 \otimes e_2]\!] \triangleq \langle V_1 \cup V_2, E_1 \cup E_2 \cup V_1 \times V_2 \rangle \qquad \text{si } [\![e_1]\!] = \langle V_1, E_1 \rangle \text{ et } [\![e_2]\!] = \langle V_2, E_2 \rangle \tag{4}$$

Pour les expressions étendues, on ajoute la règle suivante

$$[\![\bar{g}]\!] \triangleq g \tag{5}$$

FIGURE 5 – Sémantique dénotationnelle

6. Démontrer l'adéquation.
7. Démontrer en sémantique opérationnelle l'équivalence suivante.

$$\forall g, x \otimes (y \oplus z) \to^* \bar{g} \iff (x \otimes y) \oplus (x \otimes z) \to^* \bar{g} \tag{6}$$

8. Définir une fonction $\mathsf{map} : (X \to Y) \times \mathsf{Expr}_X \to \mathsf{Expr}_Y$.
9. Calculer la sémantique dénotationnelle de map sur les graphes.
10. Quel est l'ensemble des graphes construits à partir des expressions Expr_X ?
11. En admettant l'existence d'une fonction $V : \mathsf{Expr}_X \to \mathcal{P}_f(X)$ qui à une expression de graphe associe l'ensemble de ses sommets, décrire une fonction $N_x : \mathsf{Expr}_X \to \mathcal{P}_f(X)$ qui à une expression de graphe e représentant un graphe $\langle V, E \rangle$ associe l'ensemble $\{y \in X \mid (x, y) \in E\}$.
    Justifier que pour toute expression $e \in \mathsf{Expr}_X$ telle que $[\![e]\!] = \langle V, E \rangle$ on a l'égalité $N_x(e) = \{y \mid (x, y) \in E\}$.

Solution :

1. Voilà ce que dit le théorème de progrès.

$$\forall e \in \mathsf{Expr}_X^+, e \to e' \ \vee \ \exists g, e = \bar{g} \tag{7}$$

   Cela se prouve par simple analyse de cas sur l'expression, même pas besoin d'induction...

2. Voilà ce que dit le théorème de déterminisme.

$$\forall e, \forall e', \forall e'', e \to e' \wedge e \to e'' \implies e' = e'' \tag{8}$$

Cela se démontre par induction sur l'expression e. **Attention** il faut bien prendre en compte le fait que dans les expressions intermédiaires, $\bar{g}$ où g est un graphe peut apparaître.

Les cas de base sont évidents. Pour l'hérédité, il y a un cas évident, celui $F[e]$. Pour les autres, il suffit de voir que g_3 est uniquement déterminé à partir de g_1 et g_2.

3. Voilà ce que dit le théorème de terminaison.

$$\forall e, \sup\{k \mid \exists e', e \to^k e'\} < \infty \tag{9}$$

Il suffit par exemple de montrer que $e \to e' \implies |e| > |e'|$ où $|e'| = 0$ si $e' = \bar{g}$ et 1 si expression feuille, somme plus 1 quand c'est un opérateur binaire.

4. Un résultat de correction s'écrit

$$\forall e, e \to^* \bar{g} \implies [\![e]\!] = g \tag{10}$$

Un résultat d'adéquation s'écrit

$$\forall e, [\![e]\!] = g \implies e \to^* \bar{g} \tag{11}$$

5. Pour montrer la correction, il suffit de montrer que toutes les règles sont admissibles dans la sémantique donnée. C'est-à-dire $[\![e]\!] = g \wedge e \to e' \implies [\![e']\!] = g$. Il n'y a pas de difficulté particulière.
6. Pour montrer l'adéquation, on procède par induction sur l'expression e. Tout est trivial, sauf qu'il faut un petit lemme

$$\forall e, \forall e', \forall F. \quad e \to^* e' \implies F[e] \to^* F[e']$$

7. Il suffit de le vérifier sur les expressions de type $\bar{g}$ car on s'y ramène en utilisant des réductions non-déterministes.
8. Pas grand chose à dire, $\mathsf{map}\, f\, (e \oplus e') = (\mathsf{map}\, f\, e) \oplus (\mathsf{map}\, f\, e')$ etc ... La seule chose à faire est $\mathsf{map}\, f\, (\mathsf{V}x) = \mathsf{V}(fx)$.
9. La fonction $[\![\mathsf{map}]\!]$ est tout simplement la fonction qui à $\langle V, E\rangle$ associe $\langle f(V), f(E)\rangle$ où $f(E) \triangleq \{(f(u), f(v)) \mid (u, v) \in E\}$.
10. Tout graphe $g = \langle V, E\rangle$ est représentable dans les expressions Expr_X où $X \subseteq V$. Il suffit de voir qu'on peut calculer

$$\bigoplus_{(u,v)\in E} (\mathsf{V}u) \otimes (\mathsf{V}v) \tag{12}$$

11. Il suffit de définir la fonction N comme suit

$$\begin{aligned}
N_x(\mathsf{V}y) &\triangleq \emptyset \\
N_x(\mathsf{Empty}) &\triangleq \emptyset \\
N_x(e_1 \oplus e_2) &\triangleq N(e_1) \cup N(e_2) \\
N_x(e_1 \otimes e_2) &\triangleq N(e_1) \cup N(e_2) \cup \begin{cases} V(e_2) & \text{si } x \in V(e_1) \\ \emptyset & \text{sinon} \end{cases}
\end{aligned}$$

On montre par induction sur e que $N_x(e)$ correspond bien aux voisins de x dans $[\![e]\!]$. Les cas de base ne posent aucun problème. Le seul cas intéressant est celui de $\otimes$.

Notions de réécriture

Confluence Étant donné une relation $\to$ et trois termes a, b, c tels que $a \to^* b$ et $a \to^* c$, alors il existe un terme d tel que $b \to^* d$ et $c \to^* d$.

Forme normale Étant donné une relation $\to$ et un terme a, on dit que b est une forme normale de a si $a \to^* b$ et il n'existe aucun terme c tel que $b \to c$.

Exercice 9 :

On transforme les contextes pour être de la forme $F := \square \oplus e \mid e \oplus \square \mid \square \otimes e \mid e \otimes \square$.

1. Montrer que la sémantique n'est plus déterministe.
2. Énoncer un théorème de confluence.
3. Le démontrer.
4. En admettant la terminaison du nouveau système, en déduire l'existence d'une *unique* forme normale pour les expressions.

Solution :

On ne demande plus d'avoir complètement évalué les expressions à gauche pour évaluer à droite... Donc il va y avoir un non-déterminisme.

1. Il suffit de considérer l'expression suivante
$$\mathsf{V}x \oplus \mathsf{V}y \to \bar{g} \oplus Vy$$
$$\mathsf{V}x \oplus \mathsf{V}y \to \mathsf{V}x \oplus \bar{g}$$
2. Un théorème de confluence est de la forme suivante
$$\forall e, \forall e_1, \forall e_2, e \to e_1 \wedge e \to e_2 \implies \exists e_{1\wedge 2}, e_1 \to^* e_{1\wedge 2} \wedge e_2 \to^* e_{1\wedge 2} \quad (13)$$
Pour bien le comprendre, faire un petit dessin, cela permet de refermer des carrés.
3. Pour le démontrer, on fait une induction sur l'expression e, puis une analyse de cas sur la règle. **La seule règle importante est celle sur $F[e]$!**
4. Comme le système est terminant, et en utilisant le théorème de progrès, il existe un unique graphe g tel que $e \to^* g \not\to$. On voudra alors dire que le *sens* de l'expression e est l'unique graphe vers lequel il se réduit...

3 Un peu de mathématiques

Inf-demi-treillis complet

Un *inf-demi-treillis complet* est un ensemble ordonné $(X, \leq)$ non vide tel que toute famille $F \subseteq X$ a une borne inférieure $\bigwedge F$.

Exercice 10 : Treillis complets

1. Montrer qu'un inf-demi-treillis complet est en fait un treillis complet.
2. Montrer que l'ensemble des parties d'un ensemble A quelconque est un treillis complet.
3. Justifier que l'ensemble des ouverts $\mathcal{O}$ d'un espace topologique $(X, \mathcal{O})$ est un treillis complet. Quel est le sup d'une famille F d'ouverts ? Quel est son inf ?

Solution :

1. Montrons déjà que tout ensemble F possède un majorant. C'est obtenu en considérant $\inf \emptyset$.

   Considérons désormais G l'ensemble des majorants de F (non vide !), alors $\inf G = \sup F$. Il est clair que $\inf G$ est inférieur à tout majorant de F, reste à montrer $\forall x \in F, \inf G \geq x$. Néanmoins, c'est évident par définition de G.
2. L'infimum est l'intersection, le supremum l'union.
3. Les ouverts d'un espace topologique sont stable par union arbitraire, en particulier l'union va être le sup. **Attention** l'inf n'est pas l'intersection, car une intersection arbitraire d'ouverts n'est pas nécessairement ouverte ! On va tomber sur l'intérieur de l'intersection (par définition de l'intérieur).

Knaster-Tarski

Soit $(X, \leq)$ un treillis complet et $f : X \to X$ une fonction monotone. Alors l'ensemble des points fixes de f est un treillis complet non vide.

Exercice 11 : Une preuve de Knaster-Tarski

Soit f une fonction monotone de X dans X où X est un treillis complet.

1. Montrez que f possède un plus grand et un plus petit point fixe.
2. En déduire que l'ensemble des points fixes est un treillis complet.

Solution :

1. Considérons l'ensemble $A \triangleq \{x \in X \mid x \leq f(x)\}$ et $B \triangleq \{x \in X \mid f(x) \leq x\}$. Posons $a_\infty = \sup A$ et $b_\infty = \inf B$.
   - Si $a \in A$ alors $f(a) \in A$ (resp. b, B) car f est monotone.
   - Comme f est monotone,
   $$f(b_\infty) \leq f(B) \wedge f(a_\infty) \geq f(A)$$
   - Par définition de B et A on obtient alors
   $$f(b_\infty) \leq \inf f(B) \leq \inf B \wedge f(a_\infty) \geq \sup f(A) \geq \sup A$$
   - Par conséquent, $a_\infty \in A$ et $b_\infty \in B$. Donc a_∞ et b_∞ sont des points fixes.
   - Par construction, si $f(x) = x$ alors $x \in A \cap B$, donc $a_\infty \geq x \geq b_\infty$. Ce sont donc les plus grands/plus petits points fixes.
2. Prenons une famille F non vide de points fixes de f, posons $g : x \mapsto \sup F \cup \{f(x)\}$.
   - La fonction g est monotone car f est monotone. Elle possède donc un plus petit point fixe u.
   - En particulier, $f(u) \leq u$ et u majore F. Comme F est composé de points fixes de f, $f(u)$ est aussi un majorant de F.
   - Ainsi comme u est le plus petit des majorants de $F \cup \{f(u)\}$ on déduit $u \leq f(u)$. Ce qui montre que u est un point fixe de f.
   - Par construction c'est donc le plus petit point fixe de f qui majore F.
   - On peut faire la même construction pour calculer les inf.

Exercice 12 : Utilisation de Knaster-Tarski

Démontrez le théorème de Cantor-Schröder-Bernstein : si A et B sont deux ensembles tels qu'il existe deux injections f et g respectivement de A dans B et de B dans A, alors A est en bijection avec B. *Indication : faire un dessin avec deux patates, tout serait si beau si on pouvait trouver X tel que $f(X)^c$....*

Solution :

Dessin patates ! On veut utiliser f sur une partie, et g^{-1} sur l'autre pour définir une bijection entre A et B. Mais pour cela il faut que l'image de g soit disjointe de l'ensemble qu'on utilise pour f.

On cherche donc une partie X qui vérifie

$$g(f(X)^c) = X^c \tag{14}$$

Cette fonction est monotone pour l'inclusion des parties de A, elle possède donc un point fixe U. On peut alors poser

$$h(x) \triangleq \begin{cases} f(x) & \text{si } x \in U \\ g^{-1}(x) & \text{si } x \notin U \end{cases} \tag{15}$$

Et vérifier que h est bien définie et est une bijection.

Rappel sur les familles dirigées

Une famille D d'un ensemble $(X, \leq)$ est dirigée si et seulement si

1. D est non vide ;
2. pour tout $(x, y) \in D$, il existe $z \in D$, tel que $z \geq x$ et $z \geq y$.

Rappels sur les dcpos

Un dcpo est un ensemble partiellement ordonné $(X, \leq)$ tel que toute famille dirigée possède un sup. Un dcpo est *pointé* s'il existe un élément minimal.

Exercice 13 : Qui est quoi ?

Dessinez les ensembles suivants et indiquez lesquels sont des dcpos, lesquels sont des treillis complets, lesquels sont pointés, le tout *en justifiant.*

1. $\mathbf{1} = \{\bot\}$.
2. $\mathbf{Bool}_\bot = \{0, 1, \bot\}$ avec $x < y$ si et seulement si $x = \bot$ et $y \neq \bot$.
3. $\mathbb{N}$ avec l'ordre usuel.
4. $\omega + 1$ avec l'ordre usuel.
5. $\mathbb{N}^2$ avec l'ordre produit.
6. $\{[x, y] \mid x, y \in I, x \leq y\}$ avec l'ordre $\supseteq$ où $I = [0, 1]$.
7. $\{[x, y] \mid x, y \in I \cap \mathbb{Q}, x \leq y\}$ avec l'ordre $\supseteq$ où $I = [0, 1]$.

Solution :

Numéro	Treillis complet	dcpo	Pointé
1	✔	✔	✔
2	✖	✔	✔
3	✖	✖	✔
4	✔	✔	✔
5	✖	✖	✔
6	✖	✔	✔
7	✖	✖	✔

4 Topologie

Topologie

Une topologie τ sur un ensemble X est un ensemble de parties de X qui vérifie

1. τ est stable par intersection finie.
2. τ est stable par union quelconque.

On dira alors d'un élément de τ qu'il est *ouvert*. Le complémentaire d'un ouvert est par définition un ensemble *fermé*.

Fonction continue

Une fonction $f : X \to Y$ est *continue* de (X, τ) vers (Y, θ) si et seulement si

$$\text{Pour tout } U \in \theta,\ f^{-1}(U) \in \tau \tag{16}$$

Topologie de Scott

Soit $(D, \leq)$ un dcpo. Une partie $U \subseteq D$ est appellée un *ouvert de Scott* si et seulement si elle vérifie

1. U est clos vers le haut :

$$\text{Pour tous } x, y, \quad \text{Si } x \in U \wedge x \leq y, \text{ alors } y \in U. \tag{17}$$

2. U est inacessible par le bas :

$$\text{Pour tout } E \text{ dirigée}, \quad \text{Si } \sup E \in U, \text{ alors } E \cap U \neq \emptyset. \tag{18}$$

Fonction Scott-continue

Soit $(X, \leq)$ un ensemble ordonné. Une famille $D \subseteq X$ est dite dirigée si :

— D est non vide ;

— et pour tous $x, y \in D$, x et y ont un majorant dans D.

Une fonction $f : X \to Y$ est Scott-continue si :

— f est monotone ;

— et pour toute famille dirigée D qui admet un sup $\vee D$, la famille $f(D)$ a un sup et $f(\vee D) = \vee f(D)$.

Exercice 14 : Un gentil début

Soit τ une topologie sur un ensemble X.

1. Montrer que τ contient l'ensemble X.
2. Montrer que τ contient l'ensemble $\emptyset$.

Exercice 15 : Topologie de Scott

1. Montrer que la topologie de Scott est une topologie.
2. Montrer qu'un fermé de D est clos par le bas et par suprema de famille dirigée.
3. Montrer que $\downarrow x \triangleq \{y \in D \mid y \leq x\}$ est un fermé de D pour la topologie de Scott.
4. Montrer que les fonctions continues pour la topologie de Scott sont les fonctions Scott-continues.

Solution :

1. La stabilité par union arbitraire est triviale. L'intersection finie se fait juste en remarquant qu'on prend un élément au dessus de tous les éléments qui sont dans chacun des U_i.
2. Le complémentaire d'un clos par le haut est clos par le bas, et il est facile de voir que inaccessible par le bas veut dire que son complémentaire contient les sup.
3. Son complémentaire est ouvert : clairement il est clos par le haut, et inaccessible par le bas...
4. On fait les deux sens
   (a) Supposons f Scott-continue, montrons que $f^{-1}(U)$ est ouvert quand U est un ouvert. Comme f est monotone, on a $f^{-1}(U)$ clos par le haut. Supposons $\sup F \in f^{-1}(U)$, alors $f(\sup F) \in U$, donc $\sup f(F) \in U$ donc $f(F) \cap U \neq \emptyset$ donc $F \cap f^{-1}(U) \neq \emptyset$.
   (b) Réciproquement, considérons f continue. Montrons que f est monotone. Si $x \leq y$, considérons $F = \downarrow f(y)$ qui est fermé. On a $y \in f^{-1}(F)$ et comme c'est un clos par le bas, x aussi, donc $f(x) \in F$, donc $f(x) \leq f(y)$.
   Montrons déormais que f préserve les sup. Soit F une famille dirigée, on sait déjà comme f est monotone que

$$f(\sup x) \geq \sup f(x) \tag{19}$$

   Reste à montrer que $f(\sup F) \leq \sup f(x)$. Pour cela, considérons $G = \downarrow \sup f(F)$ c'est un fermé. On a donc $f^{-1}(G)$ qui est aussi fermé. Par construction, $F \subseteq f^{-1}(G)$. Comme c'est un fermé, il est stable par sup de famille dirigée donc $\sup F \in f^{-1}(G)$, cela prouve

$$f(\sup F) \leq \sup f(F) \tag{20}$$

   Et on a donc conclu.

Exercice 16 : Booléens

On considère $\mathbf{Bool}_\perp = \{0, 1, \perp\}$ ordonné tel que avec $x < y$ si et seulement si $x = \perp$ et $y \neq \perp$.

1. Quels sont les ouverts de Scott de $\mathbf{Bool}_\perp$? Les fermés ?
2. Exhibez toutes les fonctions monotones de $\mathbf{Bool}_\perp$ dans $\mathbf{Bool}_\perp$.
3. Soit D un dcpo, et f une fonction monotone de $\mathbf{Bool}_\perp$ dans D. Montrez que f est Scott-continue.

4. Dessinez $\mathbf{Bool}_\perp \times \mathbf{Bool}_\perp$ (ordre produit).
5. Énumérez les fonctions Scott continues f telle que f restreinte à $\{0,1\}$ définit la fonction booléenne « ou ».
6. En voyant $\perp$ comme « un calcul divergent », donnez une interprétation calculatoire de chacun des prolongements à $\mathbf{Bool}_\perp$ de la fonction booléenne « ou ».

Solution :

1. Les ouverts de Scott sont $\{0\}, \{1\}, \emptyset, \{0,1\}, \{0,1,\perp\}$. Les fermés sont les complémentaires.
2. Les fonctions monotones de $\mathbf{Bool}_\perp$ dans lui-même sont
   (a) Toutes les opérations booléennes usuelles (id, négation, confondre 0, etc) liftées.
   (b) Les opérations qui envoient $\perp$ sur $\perp$, et qui envoient font n'importe quoi sur 0 et 1.
   (c) Les fonctions qui envoient tout le monde sur $\{0\}$ ou sur $\{1\}$.
3. Faire un joli dessin
4. Soit f une fonction monotone, comme $\mathbf{Bool}_\perp$ est plat, il est assez facile de calculer les sup...
5. On peut faire du $\vee$ *eager*, paresseux dans deux sens, ou bien parallèle.
6. CF la réponse précédente.

Exercice 17 : Réels à précision arbitraire

Soit $I = \mathbb{R}$ et $J = \{[x,y] \mid x,y \in I, x \leq y\}$ avec l'ordre $\supseteq$.

1. Montrez que J est un dcpo. Est-ce un treillis ? Complet ?
2. Donnez une fonction monotone de J dans $\mathbf{Bool}_\perp$ qui n'est pas Scott-continue.
3. Quels sont les éléments maximaux de J ? Notons M l'ensemble des éléments maximaux.
4. Soit f une fonction continue de J dans $\mathbf{Bool}_\perp$. Montrez que $f^{-1}(\{1\})$ est ouvert.
5. Considérons l'application $I : x \mapsto \{x\}$ qui va de $[0,1]$ dans J. Montrer que I est continue.

Bonus Qu'est-ce que la topologie de Scott restreinte à l'ensemble des éléments maximaux ?

6. Montrer que M est connexe. C'est-à-dire qu'il n'existe pas d'ouverts U, V disjoints tous deux non vides tels que
$$U \cap M \uplus V \cap M = M \tag{21}$$
7. Soit g une fonction continue de J dans $\mathbf{Bool}_\perp$ telle que pour tout $x \in I$, $g(\{x\}) \neq \perp$. Montrez que g est constante sur I.
8. Imaginons un langage de programmation qui implémente l'arithmétique réelle de précision arbitraire en utilisant des intervalles. Comment se compotera la fonction d'égalité de ce langage ?

Solution :

1. Non I n'est pas un treillis, déjà il n'a même pas d'élément $\top$. En revanche c'est bien un dcpo avec
$$\sup F = \bigcap F \tag{22}$$
2. Il suffit de faire une fonction qui produit tout le temps $\perp$ excepté sur $\{0\}$. C'est une fonction monotone, mais pas Scott-continue.
3. Les éléments maximaux de J sont les singletons.

4. C'est trivial, $\{1\}$ est un ouvert.
5. Il suffit de considérer $I^{-1}(U)$ et de voir que l'inacessible par le bas implique l'existence d'un voisinage ouvert autour de chaque point.

Bonus La topologie de Scott restreinte aux éléments maximaux est la topologie usuelle de $\mathbb{R}$... Pour cela, il suffit de montrer que I est en fait un homéomorphisme vers M.

6. Comme M est l'image de $[0,1]$ par une application continue, M est connexe.
7. L'image d'un *connexe* par une fonction continue est *connexe*. Or, les seuls connexes de $\mathbf{Bool}_\perp$ ne contenant pas $\perp$ sont $\{0\}$ et $\{1\}$.
8. Une fonction raisonnable d'égalité doit être réflexive et différencier 0 et 1. Si elle était Scott-continue, elle le serait en ses deux arguments et donc la fonction $(0 ==)$ serait constante au vu des questions précédentes.

   La fonction d'égalité n'est donc pas Scott-continue, en particulier, elle ne va pas correspondre à une fonction exprimable dans le langage qui nous intéresse.

Exercice 18 : Mots finis ...ou infinis

Soit $S = \{0,1\}^\infty = \{0,1\}^* \cup \{0,1\}^\omega$, avec l'ordre préfixe.

1. Montrez que S est un dcpo. Est-ce un treillis ?
2. Quels sont les éléments maximaux de S ?
3. Soit f une fonction de S dans $\mathbf{Bool}_\perp$ telle que :

$$\text{Pour tout } s \in \{0,1\}^\omega, \begin{cases} f(s) = 1 & \text{si } s \text{ contient le facteur } 0 \cdot 1 \\ f(s) = 0 & \text{sinon} \end{cases}$$

   Montrez que f n'est pas Scott-continue. Intuition ?
4. On considère la fonction $v : S \to J$ définie par :

$$v(b_1 \cdot b_2 \cdots b_n) = \left[\sum_{i=1}^{n} 2^{-i} b_i, \sum_{i=1}^{n} 2^{-i} b_i + 2^{-n}\right]$$
$$v(b_1 \cdot b_2 \cdots) = \left\{\sum_{i=1}^{\infty} 2^{-i} b_i\right\}$$

   A quoi sert v ? Montrez que v est Scott-continue. Est-elle injective ?
5. Soit g une fonction Scott-continue de S dans $\mathbf{Bool}_\perp$ qui est compatible avec v :

$$\text{Pour tous } x, y \in S, \text{ si } v(x) = v(y), \text{ alors } g(x) = g(y)$$

   Montrez que si pour tout $x \in \{0,1\}^\omega$, $g(x) \neq \perp$, alors g est constante sur $\{0,1\}^\infty$. Quel espoir pour l'arithmétique de précision arbitraire ?

Solution :

1. Considérons une famille dirigée F pour l'ordre préfixe. Soit elle stationne et son sup est trivial, soit le sup des taille est infini et alors on peut construire un mot infini qui correspond. Attention ce n'est **pas** un inf-demi-treillis : en effet, l'ensemble vide n'a pas de minorant.
2. Les éléments maximaux de S sont les mots infinis.
3. Si la fonction f était Scott-continue, alors $f^{-1}(\{0\})$ est un ouvert de Scott. Donc quand on considère la suite 0^k il existe un k fini pour lequel $f(0^k) = 0$ mais alors $f(0^k \cdot 1) = 0$ par monotonie, ce qui est absurde.

4. La fonction v permet d'interpréter les mots infinis dans les réels à précision variable dans $[0,1]$. La fonction est clairement monotone à cause du 2^n qui mange le reste. Les sommes étant de réels positifs, tout commute aux sup et cela se passe bien. Attention, la fonction n'est pas injective, car le mot 01^ω est égal au mot 1.
5. Si g n'est pas constante, on peut construire $\hat{g}$ qui correspond au lifting de g selon v, et qui n'est pas non plus constante. Via l'exercice précédent, c'est absurde.

Exercice 19 : Topologie et séparation

1. Montrer que si la topologie de Scott sur $(X, \leq)$ est séparée (i.e. pour tout $x \neq x'$ il existe deux voisinages ouverts U et U' respectivement de x et de x' dont l'intersection est vide) alors $\leq$ est en réalité l'égalité sur X.
2. Montrer que la topologie de Scott est T_0, c'est-à-dire que pour tout $x \neq x'$, il existe un voisinage ouvert de x qui ne contient pas x'.

Solution :

1. Si $x \leq x'$ alors en particulier ils ne sont pas séparables, donc nécessairement $x \leq x' \to x = x'$.
2. C'est évident, soit ils sont incomparables, soit ils sont comparables. Dans tous les cas cela marche.

Exercice 20 : Catégorie Cartésienne Close

Montrez que la catégorie des dcpos est cartésienne close, c'est-à-dire passez par les étapes suivantes :

1. Montrer qu'il existe un dcpo $\mathbf{1}$ tel que pour tout dcpo D il existe une unique fonction continue de $\mathbf{1}$ vers D.
2. Montrer que si D_1 et D_2 sont deux dcpos alors $D_1 \times D_2$ avec l'ordre produit est un dcpo.
3. Montrer que $D_1 \times D_2$ vérifie une propriété universelle du produit (où toutes les quantifications sont sur des fonctions continues).

   Pour tous $f : A \to D_1, g : A \to D_2$,
   il existe un unique $h : A \to D_1 \times D_2$ tel que $\pi_1 \circ h = f$ et $\pi_2 \circ h = g$

4. Montrer que $[A \to B]$ l'ensemble des fonctions continues de A vers B ordonnées point à point est un dcpo.
5. Montrer que si A, B, C sont des dcpos, alors toute fonction continue $f : A \times B \to C$ se transforme en une fonction $\Gamma f : A \to [B \to C]$ qui est bien définie et continue.
6. Montrer qu'une fonction $f : A \times B \to C$ est continue si et seulement si elle est continue en ses deux arguments.
7. Montrer que l'application d'évaluation $\Delta : A \times [A \to B] \to B$ est continue.

Solution :

1. Le dcpo est l'ensemble vide.
2. Considérons une famille dirigée du produit $F \subseteq D_1 \times D_2$. Montrons que $\sup F = (\sup \pi_1 F, \sup \pi_2 F)$.

(a) La famille $\pi_1 F$ est dirigée car π_1 est croissante. De même pour $\pi_2 F$, on a donc bien l'existence des sup écrits plus haut.

(b) Il est clair que pour tout $(x, y) \in F$, on a $x \leq \sup \pi_1 F$ et $y \leq \sup \pi_2 F$. Donc cette paire est bien un majorant.

(c) Considérons (u, v) majorant de F, clairement $u \leq \sup \pi_1 F$ et $v \leq \sup \pi_2 F$, donc $(u, v) \geq (\sup \pi_1 F, \sup \pi_2 F)$. C'est donc le plus petit des majorants.

3. Considérons deux fonctions f et g comme décrites dans l'énoncé. On pose $h(x) \triangleq (f(x), g(x))$. Par construction on a bien l'équation désirée.

   (a) La fonction h est unique, en effet l'équation détermine les projections et donc la paire...

   (b) La fonction h est monotone, en effet $x \leq y$ implique $f(x) \leq f(y)$ et $g(x) \leq g(y)$ donc $h(x) \leq h(y)$. Elle préserve les sup parce que celui-ci se calcule composante par composante.

4. Considérons une famille dirigée de fonctions F de A vers B. Montrons que $\sup F \triangleq x \mapsto \sup\{f(x) \mid f \in F\}$.

   (a) Comme la famille F est dirigée, on déduit que pour x fixé, la famille des $\{f(x) \mid f \in F\}$ est dirigée. Ainsi, la fonction est bien définie.

   (b) Cette fonction est trivialement supérieure point à point à toutes les fonctions de F, et est tout aussi trivialement la plus petite point à point vérifiant cette propriété.

   (c) La fonction est monotone, car $x \leq y$ implique $f(x) \leq f(y)$ pour tout $f \in F$, ainsi les sup sont comparables.

   (d) La fonction préserve les sup, car si G est une famille dirigée de points

$$\sup_{x \in G} \sup_{f \in F} f(x) = \sup_{f \in F} \sup_{x \in G} f(x) = \sup_{f \in F} f(\sup_{x \in G} x) \tag{23}$$

5. Considérons une fonction $f : A \times B \to C$. Posons $\Gamma f : A \to (B \to C)$ la fonction définie par $\Gamma f : x \mapsto (y \mapsto f(x, y))$.

   (a) Pour tout $x \in A$, la fonction $y \mapsto f(x, y)$ est Scott-continue. C'est trivial.

   (b) La fonction Γf est monotone (évident)

   (c) La fonction Γf préserve les sup. Soit G une famille dirigée de A par construction on a l'égalité suivante

$$\sup_{x \in G} (\Gamma f)(x) \triangleq y \mapsto \sup_{x \in G} f(x, y) \tag{24}$$

   Donc on a bien

$$\sup_{x \in G} (\Gamma f)(x) = y \mapsto f(\sup G, y) = (\Gamma f)(\sup G) \tag{25}$$

6. Soit f une fonction continue, alors en fixant un argument elle reste trivialement continue. Réciproquement, supposons f continue en chacun de ses arguments.

   (a) La fonction f est monotone, car monotone en ses deux arguments, par construction du produit.

(b) La fonction préserve les sup car

$$\begin{aligned} f\left(\sup_{(a,b)\in F}(a,b)\right) &= f\left(\sup \pi_1 F, \sup \pi_2 F\right) \\ &= \sup_{x\in\pi_1 F} f\left(x, \sup \pi_2 F\right) \\ &= \sup_{x\in\pi_1 F}\sup_{y\in\pi_2 F} f(x,y) \\ &= \sup_{(x,y)\in F} f(x,y) \end{aligned}$$

Attention, la dernière égalité ne tient que parce que la famille est *Dirigée* !!! À bien justifier.

7. On fait comme toujours les deux étapes
   (a) Montrons que la fonction d'évaluation est croissante. Soit $(x, f) \leq (y, g)$.

$$f(x) \leq g(x) \leq g(y) \tag{26}$$

   (b) Montrons que la fonction préserve les sup. En utilisant la question précédente, on a seulement besoin de vérifier en fixant f ou en fixant x, ce qui rend la preuve triviale.

Exercice 21 : Un petit langage

On considère le langage $\{\star, \bullet\}$ équipé de la sémantique à petits pas suivante :

$$X \star \bullet Y \to XY \text{ s'il existe } n \geq 0, \text{ tel que } X = \star^n$$
$$X \bullet \star Y \to XY \text{ s'il existe } n \geq 0, \text{ tel que } X = \bullet^n$$

1. Énoncer puis prouver un théorème de déterminisme.
2. On considère le DCPO $\{0, 1\}$ équipé de l'ordre plat et la sémantique suivante :

$$\begin{aligned} [\![\varepsilon]\!]_2 &= 0 \\ [\![aX]\!]_2 &= 1 - [\![X]\!]_2 \qquad\qquad \text{si } a \in \{\star, \bullet\} \end{aligned}$$

   Montrer que cette sémantique est correcte par rapport à la sémantique à petits pas.
3. Même question pour le DCPO des entiers relatifs équipé de l'ordre plat et la sémantique suivante :

$$\begin{aligned} [\![\varepsilon]\!]_{\mathbb{Z}} &= 0 \\ [\![\star X]\!]_{\mathbb{Z}} &= 1 + [\![X]\!]_{\mathbb{Z}} \\ [\![\bullet X]\!]_{\mathbb{Z}} &= -1 + [\![X]\!]_{\mathbb{Z}} \end{aligned}$$

4. On se donne la notion d'équivalence observationnelle suivante :

$$A \equiv B \text{ est défini par : pour tout } C[\cdot], C[A] \to^* \varepsilon \text{ si et seulement si } C[B] \to^* \varepsilon$$

   Montrer que c'est une relation d'équivalence.

Pour les exercices suivants, se munir du poly de cours ou bien des diapositives pour avoir accès aux sémantiques dénotationnelles et opérationnelles.

5 Une partie fonctionnelle

Exercice 22 : La sémantique n'est pas adéquate

Considérons l'expression PCF u suivante :

```
letrec f (x) = 3 in
letrec g (x) = g (x) in
f (g 0)
```

1. Ce n'est pas une expression valide, car il manque les annotations de type. Les ajouter.
2. Calculer la sémantique dénotationnelle de u.
3. Montrer qu'il n'y a aucune dérivation $E \vdash u \Rightarrow v$ où E est un P-environnement, et v une valeur.

Exercice 23 : Un exercice nécessaire

Considérons l'expression PCF u suivante :

```
letrec f x = if x=0 then 0 else x + f (x-1) in
        f 6
```

Calculer la sémantique dénotationnelle de u.

Exercice 24 : Boolean Function Calculus

On considère le langage suivant

$$M := x \mid \lambda x : \tau.M \mid MN \mid \textbf{let } x : \tau = M \textbf{ in } N \mid \textbf{ff} \mid \textbf{tt} \mid \textbf{if } M \textbf{ then } N \textbf{ else } P$$

1. Proposer un système de typage adapté.
2. Donner une dérivation de $\vdash (\lambda x.\textbf{if } x \textbf{ then ff else } x)\textbf{tt} : \textbf{bool}$
3. Montrer que le système de type proposé est déterministe.
4. Quel élément de la syntaxe du langage de programmation est crucial pour garantir le déterminisme du typage ? Expliciter avec un exemple.
5. Montrer que la construction `let` s'encode à l'aide des autres constructions de manière bien typée.

Solution :

1. $$\frac{}{\Gamma \vdash \textbf{tt} : \textbf{bool}} \qquad \frac{}{\Gamma \vdash \textbf{ff} : \textbf{bool}} \qquad \frac{}{\Gamma, x : \tau \vdash x : \tau}$$

   $$\frac{\Gamma \vdash M : \sigma \to \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau} \qquad \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x : \sigma.M : \sigma \to \tau}$$

   $$\frac{\Gamma \vdash P : \textbf{bool} \qquad \Gamma \vdash M : \tau \qquad \Gamma \vdash N : \tau}{\Gamma \vdash \textbf{if } P \textbf{ then } M \textbf{ else } N : \tau}$$

   $$\frac{\Gamma \vdash M : \sigma \qquad \Gamma, x : \sigma \vdash N : \tau}{\Gamma \vdash \textbf{let } x = M \textbf{ in } N : \tau}$$

2. Trivial
3. Le fait que les types sont dans la syntaxe. C'est-à-dire, l'inférence de type n'est pas déterministe, la type *erasure* perd de l'information. Il suffit de voir $\lambda x.x$.
4. Il suffit d'écrire $\textbf{let } x = M \textbf{ in } N \triangleq (\lambda x.N)M$.

Exercice 25 : Relations logiques et terminaison

L'objectif de cet exercice est de montrer que tout programme bien typé en BoolPCF sans `let` termine. Pour cela, on donne la sémantique à petit pas suivante :

$$\begin{aligned}(\lambda x.M)N &\to M[x \mapsto N]\\ \textbf{if tt then } M \textbf{ else } N &\to M\\ \textbf{if ff then } M \textbf{ else } N &\to N\\ C[M] &\to C[M'] \qquad \text{si } M \to M'\end{aligned}$$

$$C := \square \mid CM \mid \textbf{if } C \textbf{ then } M \textbf{ else } M$$

On utilise la notation $M \Downarrow$ pour signifier que M termine.

On découpe la preuve en deux en utilisant une construction intermédiaire (une relation logique). Formellement, on prouve $\Gamma \vdash M : \tau \implies M \in \|\tau\|_C$ puis $M \in \|\tau\|_C \implies M \Downarrow$.

On donne ci-après la définition inductive de $\|\tau\|_V$ (pour valeurs) et $\|\tau\|_C$ (pour calculs).

$$\|\textbf{bool}\|_V \triangleq \{\textbf{tt}, \textbf{ff}\} \qquad \|\tau\|_C \triangleq \{M \mid M \to^* v \in \|\tau\|_V\}$$
$$\|\sigma \to \tau\|_V \triangleq \{\lambda x : \sigma.M \mid \forall v \in \|\sigma\|_V, M[x \mapsto v] \in \|\tau\|_C\}$$

La définition inductive sur τ ne prend en compte que des programmes sans variables libres. Pour pallier cela, on définit

$$\|\tau\|_X^f \triangleq \left\{ M \mid \exists \Gamma, \Gamma \vdash M : \tau \wedge \forall \rho : \prod_{x:\sigma \in \Gamma} \|\sigma\|_V, M\rho \in \|\tau\|_X \right\}$$

Ce qui se lit, « un terme avec des variables libres $x_i : \sigma_i$ est dans $\|\tau\|_X^f$ si et seulement s'il est bien typé et pour toute instanciation des x_i avec des éléments de $\|\sigma_i\|_V$, le terme est dans $\|\tau\|_X$ ». Remarquons que les variables sont instanciées avec des termes *clos* (c'est-à-dire, sans variable libre) !

1. Montrer que la sémantique opérationnelle est déterministe.
2. Montrer que le typage est préservé par réduction.
3. Montrer que $\|\tau\|_V \subseteq \|\tau\|_C$, en déduire la même inclusion pour les termes ouverts.
4. Montrer que $\|\tau\|_C$ est clos par réduction et anti-réduction. En déduire que $\|\tau\|_C^f$ est clos par les mêmes opérations.
5. Montrer par induction sur τ la propriété suivante :
$$\forall M, \forall \Gamma, \forall \tau, \Gamma \vdash M : \tau \implies M \in \|\tau\|_C^f$$
6. Montrer par induction sur τ la propriété suivante :
$$\forall M, \forall \tau, M \in \|\tau\|_C^f \implies M \Downarrow$$
7. En déduire que tout terme typé termine.
8. En particulier, l'expression suivante n'est pas typable : $\Delta \triangleq \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$. Que "calcule" cette expression ?
9. Si on ajoute Δ comme une construction primitive, quel serait son type ?
10. Cet ajout change-t-il le théorème de terminaison ?