Programmation 1

TD n°11

Aliaume Lopez

9 décembre 2019

♻ : reprise d'un exercice ! : exercice de compréhension
👍 : exercice fondamental de cours 🏆 : une solution complète par mail = un gâteau

Exercise 1 : Quelques aides

1. Sortir une feuille pour noter la correction.
2. Ne pas attendre la correction pour réfléchir sur une feuille.
3. Ne pas hésiter à demander à son voisin, ou mieux, au chargé de TD.
4. Rédiger et ne pas se contenter d'avoir une idée.

! **Exercise 2 : Back to Luc's answer**

Posons $J \triangleq \{[a,b] \mid 0 \leq a \leq b \leq 1\}$ où a et b sont des nombres réels. Rappelons que $(J, \supseteq)$ est un DCPO et que cela donne lieu à une topologie τ sur J, la topologie de Scott.

1. Considérons M l'ensemble des éléments maximaux de J. Déterminer la topologie induite par τ sur M.
2. Montrer que toute propriété P continue sur M est constante.

1 Sémantique et vérification

Imp

On donne une version de Imp possédant non seulement des expressions arithmétiques, mais aussi des expressions booléennes.

$$e := x \mid 0 \mid 1 \mid e + e \mid -e \mid e \times e$$
$$b := (e \sim e) \mid e \leq e \mid \neg b \mid b \wedge b$$
$$c := \textbf{skip} \mid \textbf{while } b \textbf{ do } c \mid x := e \mid \textbf{if } b \textbf{ then } c \textbf{ else } c$$

Formules arithmétiques au premier ordre

Voici la construction des formules au premier ordre que nous autoriserons, leur ensemble est noté $\mathsf{FO}[0, 1, +, \times, \leq]$. Dans la suite i est une variable logique à valeur entière.

$$t := x \mid 0 \mid 1 \mid t + t \mid -t \mid t \times t \mid i$$
$$\phi := (t \sim t) \mid t \leq t \mid \neg\phi \mid \phi \wedge \phi \mid \exists i.\phi$$

Triplets de Hoare

On appelle triplet de Hoare $\{\phi\}\ c\ \{\psi\}$. On dit que ce triplet est *valide* sous I, ce qui est noté $\models^I \{\phi\}\ c\ \{\psi\}$ quand

$$\forall \rho, \rho \models^I \phi \wedge [\![c]\!]_\rho \neq \bot \implies [\![c]\!]_\rho \models^I \psi$$

Une autre manière de présenter cela est d'étendre la sémantique des formules en posant $\bot \models^I \phi$ quelque soit la formule ϕ et l'environnement σ.
On notera $\models \{\phi\}\ c\ \{\psi\}$ quand pour tout I on a $\models^I \{\phi\}\ c\ \{\psi\}$.

Axiomatique de Hoare

On donne des règles de Hoare pour toutes les constructions excepté le while.

$$\frac{}{\{\phi\}\ \textbf{skip}\ \{\phi\}} \qquad \frac{\{\phi \wedge b\}\ c_1\ \{\psi\} \qquad \{\phi \wedge \neg b\}\ c_2\ \{\psi\}}{\{\phi\}\ \textbf{if}\ b\ \textbf{then}\ c_1\ \textbf{else}\ c_2\ \{\psi\}}$$

$$\frac{}{\{\phi[x := e]\}\ x := e\ \{\phi\}} \qquad \frac{\phi \implies \phi' \qquad \{\phi'\}\ c\ \{\psi'\} \qquad \psi' \implies \psi}{\{\phi\}\ c\ \{\psi\}}$$

♻ Exercise 3 : Hoare sur un langage jouet

1. Montrer que pour tout ρ, I, pour tout u termes et x variable
$$\rho \models^I \phi[x \mapsto u] \iff \rho[x \mapsto [\![u]\!]_\rho] \models^I \phi$$
2. Donner le triplet de Hoare correspondant à la séquence.
3. Montrer que tout triplet de Hoare est valide. C'est-à-dire que le système est correct.
4. Proposer une règle pour le while. Montrer que celle-ci est correcte.
5. À l'aide de ce système axiomatique, prouver le triplet suivant
$$\{x \sim 0 \wedge y \sim 0 \wedge z \sim 0 \wedge n \geq 0\}\ c\ \{x \sim n^3\} \tag{1}$$

Où
$$c \triangleq \textbf{while}\ z < 3n\ \textbf{do}\ z := z + 3; y := y + 2z - 3; x := x + y - z + 1$$

Plus faible précondition libérale

On note $\mathsf{wlp}^I\ (c, \phi) \triangleq \{\rho \mid [\![c]\!]_\rho \models^I \phi\}$.

! Exercise 4 : Plus faible précondition libérale

1. Soit I une interprétation des variables logiques. Pour tout programme c sans boucle while et formule ψ, construire une formule $\phi_{c,\psi}$ telle que $\rho \models^I \phi_{c,\psi}$ si et seulement si $\rho \in \mathsf{wlp}^I\ (c, \psi)$.
2. Soit ϕ une formule définissant $\mathsf{wlp}^I\ (\textbf{while}\ b\ \textbf{do}\ c, \psi)$. Donnez une équation $\models^I \phi \iff \phi'$ où ϕ' est une formule faisant intervenir ϕ.
3. À l'aide de disjonctions et conjonctions infinies, écrire deux solutions à cette équation.
4. Laquelle correspond à ϕ ?

Exercise 5 : Complétude

On admet qu'il existe une formule exprimant la plus faible précondition libérale pour la boucle while.

1. Montrer que l'axiomatique définie est complète. C'est-à-dire, prouver que pour tout triplet valide $\models \{\phi\}\ c\ \{\psi\}$ il existe une dérivation de $\{\phi\}\ c\ \{\psi\}$. *Indication : On pourra commencer par le démontrer pour les plus faibles préconditions libérales.*
2. Que dire d'un système S de preuve sur les triplets de Hoare qui est correct et vérifiable ?
3. Pourquoi la logique de Hoare est-elle malgré tout complète ?
4. On admet que les plus faibles préconditions libérales sont calculables. En déduire que le problème suivant n'est pas récursivement énnumérable.

   Entrée Une formule close $\phi \in \mathsf{FO}[0, 1, +, \times, \leq]$

   Sortie Est-ce que ϕ est valide ?

2 Probably Correct Functions

Pour l'exercice suivant, se munir du poly de cours ou bien des transparents pour avoir accès aux sémantiques dénotationnelles et opérationnelles.

Exercise 6 : 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 $E \vdash u \Downarrow v$ où E est un P-environnement, et v une valeur.

Exercise 7 : Jouons avec les types

Pour chaque expression OCaml ci-dessous, donner s'il existe le type de l'expression. Justifier.

1. `let f x = x in (f 3, f "trois")`
2. `(fun f -> (f 3, f "trois")) (fun x -> x)`
3. `let f x = x in leg g = ref f in (!g 3, !g "trois")`

Exercise 8 : 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.

Exercise 9 : 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* !

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 ?