Programmation 1

TD n°10

25 novembre 2020

1 Semantics and verification

Imp

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

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

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.

$$\begin{aligned} t &:= x \mid 0 \mid 1 \mid t+t \mid -t \mid t \times t \mid i \\ \phi &:= (t \sim t) \mid t \leq t \mid \neg\phi \mid \phi \wedge \phi \mid \exists i.\phi \end{aligned}$$

Exercise 1 : Warmup

1. Give denotational semantics for Boolean expressions.
2. Give semantics for logical formulae. We write $\rho \models^I \phi$ when the formula ϕ is valid in the environment ρ for the program variables and I for the logical variables.
3. Notice that the syntax of Boolean expressions of Imp is a subset of the syntax of the formulas. Do the two semantics then coincide ? Show that for all I, $\rho \models^I b \iff [\![b]\!]_\rho \neq 0$.
4. Show that we can assume that $x < y$ is a valid boolean expression.

Solution:

1. $\mathcal{B}[\![\textbf{true}]\!]\sigma = \mathit{true}$
   $\mathcal{B}[\![\textbf{false}]\!]\sigma = \mathit{false}$
   $\mathcal{B}[\![a_0 \sim a_1]\!]\sigma = (\mathcal{A}[\![a_0]\!]\sigma = \mathcal{A}[\![a_1]\!]\sigma)$
   $\mathcal{B}[\![a_0 \leq a_1]\!]\sigma = (\mathcal{A}[\![a_0]\!]\sigma \leq \mathcal{A}[\![a_1]\!]\sigma)$
   $\mathcal{B}[\![\neg b]\!]\sigma = \neg\mathcal{B}[\![b]\!]\sigma$
   $\mathcal{B}[\![b_0 \wedge b_1]\!]\sigma = \mathcal{B}[\![b_0]\!]\sigma \wedge \mathcal{B}[\![b_1]\!]\sigma$

2.

$$\begin{array}{ll}
\rho \models t_1 \sim t_2 & \text{if } [\![t_1]\!]\rho \sim [\![t_2]\!]\rho\,; \\
\rho \models \neg\varphi & \text{if } \rho \not\models \varphi\,; \\
\rho \models \phi \wedge \psi & \text{if } \rho \models \varphi \text{ and } \rho \models \psi\,; \\
\rho \models \exists i.\varphi & \text{if } \rho \models \varphi[i := n] \text{ for some } n.
\end{array}$$

3. We can use straightforward structural induction for each instance to show equivalence.
4. A simple structural induction shows that each denotation is a function. For example,

$$\mathcal{B}[\![a_0 < a_1]\!]\sigma = \begin{cases} \textbf{true} & \text{if } \mathcal{A}[\![a_0]\!]\sigma < \mathcal{A}[\![a_1]\!]\sigma \\ \textbf{false} & \text{if } \mathcal{A}[\![a_1]\!]\sigma \leq \mathcal{A}[\![a_0]\!]\sigma \end{cases}$$

for all $\sigma \in \Sigma$.

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 2 : Hoare on a toy language

1. Show that for all ρ, I, for all terms u, v and variable x

$$\rho \models^I \phi[x \mapsto u] \iff \rho[x \mapsto [\![u]\!]_\rho] \models^I \phi$$

2. Show that every Hoare triple is valid. In other words, show that the system is correct.
3. Suggest a rule for the while condition. Show that it is correct.
4. With the help of the axiomatic system, prove the following triple

$$\{x \sim 0 \wedge y \sim 0 \wedge z \sim 0 \wedge n \geq 0\}\ w\ \{x \sim n^3\}$$

where

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

Solution:

1. Once again, we can use structural induction on ϕ in order to show this.
2. This is proved by induction on the derivation tree of the considered triple. Thus, for each rule, assuming that the triples in premises are valid, we need to show that the triple in conclusion is valid too. The proofs are straightforward for the all the rules given in the table. (The sequence rule is omitted in the table - that is the only non-trivial case apart from the while condition given below. Send me an email if you want the proof for that !)
3. We show the following is the rule for the while-condition.

$$\frac{\{A \wedge b\}\ c\ \{A\}}{\{A\}\ \textbf{while}\ b\ \textbf{do}\ c\ \{A \wedge \neg b\}}$$

   Note : The assertion A is called the *invariant* because the premise, that $\{A \wedge b\}\ c\ \{A\}$ is valid, says that the assertion A is preserved by a full execution of the body of the loop, and in a while loop such executions only take place from states satisfying b. From a state satisfying A either the execution of the while-loop diverges or a finite number of executions of the body are performed, each beginning in a state satisfying b. In the latter case, as A is an invariant the final state satisfies A and also $\neg b$ on exiting the loop.

   Proof. Let us assume that the premise $\{A \wedge b\}\ c\ \{A\}$ is valid. To show that $\{A\}$ **while** b **do** c $\{A \wedge \neg b\}$ is valid, let's consider some state Σ such that $[\![I]\!]_\Sigma$ holds and some execution $\Sigma, \textbf{while}\ b\ \textbf{do}\ c \rightsquigarrow^* \Sigma', \textbf{skip}$.

   We proceed by induction on the number of steps of this execution. We have two cases depending on whether the condition $[\![b]\!]_\Sigma$ is 0 or not. If it is 0 then the execution terminates in a single step, and $\Sigma' = \Sigma$, hence, $[\![A \wedge \neg b]\!]_{\Sigma'}$ holds. If the condition is not 0, then the execution has the form $\Sigma, \textbf{while}\ b\ \textbf{do}\ c \rightsquigarrow \Sigma, (c; \textbf{while}\ b\ \textbf{do}\ c) \rightsquigarrow^* \Sigma', \textbf{skip}$.

   Since the sequence is terminating, we know that there is a state Σ'' such that $\Sigma, c \rightsquigarrow \Sigma''$ and $\Sigma'', \textbf{while}\ b\ \textbf{do}\ c \rightsquigarrow^* \Sigma', \textbf{skip}$. Since $[\![A \wedge b]\!]_\Sigma$ holds and $\{A \wedge b\}\ c\ \{A\}$ is valid, $[\![A]\!]_{\Sigma''}$ holds.

   By induction, since $\Sigma'', \textbf{while}\ b\ \textbf{do}\ c \rightsquigarrow^* \Sigma', \textbf{skip}$ has fewer steps than the original execution, we get that $[\![A \wedge \neg b]\!]_{\Sigma'}$ holds. Hence, it is valid.
4. We have, $\{x = 0 \wedge y = 0 \wedge z = 0 \wedge n \geq 0\}$ while $z < 3n$ do $z := z + 3; y := y + 2z - 3; x := x + y - z + 1 \{x = n^3\}$.

   We write the while statement as **while** b **do** c for simplicity where b is $z < 3n$ and c is $z := z + 3; y := y + 2z - 3; x := x + y - z + 1$.
   The first step is to identify the invariant, which is as follows :

$$I \equiv 27x = z^3 \wedge 3y = z^2 \wedge z \leq 3n$$

   In order to show that it is an invariant, we need to show

$$\{I \wedge z < 3n\}\ c\ \{I\}$$

   We apply the assignment rules (for each variable) followed by the sequencing rule in order to obtain :

   $\{27(x + (y + 2(z + 3) - 3) - (z + 3) - 1) = (z + 3)^3 \wedge 3(y + 2(z + 3) - 3) = (z + 3)^2 \wedge (z + 3) \leq 3n\}\ c\ \{I\}$

Furthermore, since

$$z + 3 \leq 3n \implies z \leq 3(n+1) \implies z < 3n$$

we have our loop invariant. Then, using the while property, we have the inference

$$\{I\}\ \textbf{while}\ b\ \textbf{do}\ c\ \{I \land \neg b\}$$

Clearly $x \sim 0 \land y \sim 0 \land z \sim 0 \land n \geq 0 \implies I$ and,

$$\begin{aligned} I \land z \not< 3n &\implies 27x = z^3 \land 3y = z^2 \land z \leq 3n \land z \not< 3n \\ &\implies 27x = z^3 \land 3y = z^2 \land z = 3n \\ &\implies 27x = (3n)^3 \\ &\implies x = n^3 \end{aligned}$$

Finally, by the consequence rule, we have $\{x \sim 0 \land y \sim 0 \land z \sim 0 \land n \geq 0\}\ w\ \{x \sim n^3\}$.

Plus faible précondition libérale

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

Exercise 3 : Weakest liberal precondition

1. Let I be an interpretation of logical variables. For all programs c without a while loop and formulas ψ, construct a formula $\phi_{c,\psi}$ such that $\rho \models^I \phi_{c,\psi}$ if and only if $\rho \in \mathsf{wlp}^I(c, \psi)$.
2. Let ϕ be a formula defining $\mathsf{wlp}^I(\textbf{while}\ b\ \textbf{do}\ c, \psi)$. Give an equation $\models^I \phi \iff \phi'$ where ϕ' is a formula involving ϕ.
3. Using infinite disjunction and conjunction, write two solutions to this equation.
4. Which one does ϕ correspond to ?

Solution:
(Sketch)

1. For all the commands c except while, we have the following
   (a) $wlp(skip, \psi) = \psi$
   (b) $wlp(x := a, \psi) = \psi[a/x]$
   (c) $wlp((c_1; c_2), \psi) = wlp(c_1, wlp(c_2, \psi))$
   (d) $wlp(\textbf{if}\ b\ \textbf{then}\ c_1\ \textbf{else}\ c_2, \psi) = (b \implies wlp(c_1, \psi)) \land (\neg b \implies wlp(c_2, \psi)$
2. (All subsections follow) The definition of $\mathsf{wlp}^I(\textbf{while}\ b\ \textbf{do}\ c, \psi)$ is not straightforward. We encode the WLP for every iteration of the loop. For a loop that terminates in i steps, we have the following definition

$$\begin{aligned} F_0(\psi) &= \textbf{True} \\ F_{i+1}(\psi) &= (\neg b \implies \psi) \land (b \implies \mathsf{wlp}^I(c, F_i(\psi))) \end{aligned}$$

Using infinite conjuction, we obtain

$$\mathsf{wlp}^I(\textbf{while}\ b\ \textbf{do}\ c, \psi) = \bigwedge_i F_i(\psi)$$