Validation d'algorithmes

Chapitre VII

Vérification formelle : Sémantique opérationnelle

Raphaël Charrondière raphael.charrondiere@inria.fr

Cadre

- Apprendre à analyser formellement un problème algorithmique
  ➺ Voir un programme comme un objet mathématique.
- L'algorithme est indépendant du langage, le programme ne l'est pas.
  ➺ Nous avons par exemple les langages impératifs et les langages fonctionnels.

- Nous utilisons IMP, un langage jouet, i.e. simplifié au maximum pour servir de support dans ce cours.
- Ce langage est *impératif*. Il gère les affectations, les conditionnelles, les boucles whiles.
- Nous avons besoin d'une grammaire pour définir correctement ce langage.

Grammaire du langage IMP

⟨*commande*⟩ ::= <ident> := <expr>
| **if** <condition> **then** <commande> **else** <commande>
| **while** <condition> **do** <commande>
| <commande> ; <commande>
| **skip**

⟨*expr*⟩ ::= <$\mathbb{N}$>
| <ident>
| $-$<expr>
| <expr> $+$ <expr>
| <expr> $-$ <expr>
| <expr> $\times$ <expr>
| <expr> $\div$ <expr>

⟨*condition*⟩ ::= <$\mathbb{B}$>
| <expr> $>$ <expr>
| <expr> $=$ <expr>
| **not** <condition>
| <condition> **or** <condition>
| <condition> **and** <condition>

Définition

Un état mémoire noté σ est une fonction ident $\rightarrow \mathbb{N}$ qui attribue une valeur à chaque variable.
L'ensemble des états mémoire est noté $\mathbb{M}$

Notation

$\sigma[k/X]$ est l'état mémoire, $\begin{array}{ll} Y \mapsto \sigma(Y) & si\ X \neq Y \\ X \mapsto k & \end{array}$

Définition

Une relation est une partie d'un produit d'ensemble.

Par exemple une relation binaire $\leqslant$ sur E est une partie de $E \times E$.
Pour $a, b \in E$, on dit que $a \leqslant b$ si et seulement si $(a, b) \in E$

Définition

Un ordre partiel $\sqsubseteq$ sur E est une relation

- binaire
- réflexive $x \sqsubseteq x$
- transitive $x \sqsubseteq y \wedge y \sqsubseteq z \implies x \sqsubseteq z$
- antisymétrique $x \sqsubseteq y \wedge y \sqsubseteq x \implies x = y$

L'ordre est dit total si de plus
$\forall x, y \in E,\ x \sqsubseteq y \vee y \sqsubseteq x$

- $\sigma, a \looparrowright k$ sur $\mathbb{M} \times \mathsf{expr} \times \mathbb{Z}$ évalue une expression arithmétique.
- $\sigma, c \looparrowright b$ sur $\mathbb{M} \times \mathsf{condition} \times \mathbb{B}$ évalue une expression booléenne.

- $\sigma, a \looparrowright k$ sur $\mathbb{M} \times \text{expr} \times \mathbb{Z}$ évalue une expression arithmétique.
- $\sigma, c \looparrowright b$ sur $\mathbb{M} \times \text{condition} \times \mathbb{B}$ évalue une expression booléenne.
- On défini une relation ternaire $\sigma, c \Downarrow \sigma'$ sur $\mathbb{M} \times \text{commande} \times \mathbb{M}$. *L'éxécution de c à partir de σ donne σ'*

Définition

$$\frac{\text{prémisse 1} \quad \ldots \quad \text{prémisse n}}{\textit{Conclusion}} \text{ condition d'application}$$

S'il n'y a pas de prémisse, alors la règle d'inférence est un *axiome*

Note : La condition d'application peut être vide.

1. skip

1. skip $\overline{\sigma, \textbf{skip} \Downarrow \sigma}$
2. séquence

1. skip $\overline{\sigma, \textbf{skip} \Downarrow \sigma}$
2. séquence $\dfrac{\sigma, c_1 \Downarrow \sigma' \qquad \sigma', c_2 \Downarrow \sigma''}{\sigma, c_1;c_2 \Downarrow \sigma''}$
3. attribution

1. skip $\overline{\sigma, \textbf{skip} \Downarrow \sigma}$
2. séquence $\dfrac{\sigma, c_1 \Downarrow \sigma' \qquad \sigma', c_2 \Downarrow \sigma''}{\sigma, c_1;c_2 \Downarrow \sigma''}$
3. attribution $\dfrac{\sigma, a \looparrowright k}{\sigma, x{:=}a \Downarrow \sigma'} \; \sigma' = \sigma[k/x]$
4. disjonction de cas

1. skip $\dfrac{}{\sigma, \textbf{skip} \Downarrow \sigma}$
2. séquence $\dfrac{\sigma, c_1 \Downarrow \sigma' \qquad \sigma', c_2 \Downarrow \sigma''}{\sigma, c_1 ; c_2 \Downarrow \sigma''}$
3. attribution $\dfrac{\sigma, a \hookrightarrow k}{\sigma, x{:=}a \Downarrow \sigma'} \; \sigma' = \sigma[k/x]$
4. disjonction de cas $\dfrac{\sigma, b \hookrightarrow true \qquad \sigma, c_1 \Downarrow \sigma'}{\sigma, \textbf{if } \mathsf{b} \textbf{ then } c_1 \textbf{ else } c_2 \Downarrow \sigma'}$

   $\dfrac{\sigma, b \hookrightarrow false \qquad \sigma, c_2 \Downarrow \sigma'}{\sigma, \textbf{if } \mathsf{b} \textbf{ then } c_1 \textbf{ else } c_2 \Downarrow \sigma'}$
5. boucle

1. skip $\dfrac{}{\sigma, \textbf{skip} \Downarrow \sigma}$

2. séquence $\dfrac{\sigma, c_1 \Downarrow \sigma' \qquad \sigma', c_2 \Downarrow \sigma''}{\sigma, c_1 ; c_2 \Downarrow \sigma''}$

3. attribution $\dfrac{\sigma, a \hookrightarrow k}{\sigma, x := a \Downarrow \sigma'} \quad \sigma' = \sigma[k/x]$

4. disjonction de cas $\dfrac{\sigma, b \hookrightarrow \mathit{true} \qquad \sigma, c_1 \Downarrow \sigma'}{\sigma, \textbf{if}\ \mathrm{b}\ \textbf{then}\ c_1\ \textbf{else}\ c_2 \Downarrow \sigma'}$

   $\dfrac{\sigma, b \hookrightarrow \mathit{false} \qquad \sigma, c_2 \Downarrow \sigma'}{\sigma, \textbf{if}\ \mathrm{b}\ \textbf{then}\ c_1\ \textbf{else}\ c_2 \Downarrow \sigma'}$

5. boucle $\dfrac{\sigma, b \hookrightarrow \mathit{false}}{\sigma, \textbf{while}\ \mathrm{b}\ \textbf{do}\ c \Downarrow \sigma}$

   $\dfrac{\sigma, b \hookrightarrow \mathit{true} \qquad \sigma, c \Downarrow \sigma' \qquad \sigma', \textbf{while}\ \mathrm{b}\ \textbf{do}\ c \Downarrow \sigma''}{\sigma, \textbf{while}\ \mathrm{b}\ \textbf{do}\ c \Downarrow \sigma''}$

C: R:=1; **while** $K > 1$ **do** (R:=$R \times K$; K:=$K - 1$)

C: $R{:=}1$; **while** $K > 1$ **do** $(R{:=}R \times K;\ K{:=}K-1)$
CI: $R{:=}1$
CA: $R{:=}R \times K;\ K{:=}K-1$
CB: **while** $K > 1$ **do** $(R{:=}R \times K;\ K{:=}K-1)$

	σ_0	σ_1	σ_2	σ_3
K	2	2	2	1
R	0	1	2	2

C: $R{:=}1$; **while** $K > 1$ **do** $(R{:=}R \times K;\ K{:=}K - 1)$
CI: $R{:=}1$
CA: $R{:=}R \times K;\ K{:=}K - 1$
CB: **while** $K > 1$ **do** $(R{:=}R \times K;\ K{:=}K - 1)$

	σ_0	σ_1	σ_2	σ_3
K	2	2	2	1
R	0	1	2	2

$$\mathcal{A}_2 \ : \quad \dfrac{\dfrac{\sigma_1, R \times K \looparrowright 2}{\sigma_1, R := R \times K \Downarrow \sigma_2}\ \ \sigma_2 = \sigma_1[2/R] \qquad \dfrac{\sigma_2, K - 1 \looparrowright 1}{\sigma_2, K := K - 1 \Downarrow \sigma_3}\ \ \sigma_3 = \sigma_2[1/K]}{\sigma_1, CA \Downarrow \sigma_3}$$

$$\mathcal{A}_1 \ : \quad \dfrac{\dfrac{\sigma_0, 1 \looparrowright 1}{\sigma_0, CI \Downarrow \sigma_1}\ \ \sigma_1 = \sigma_0[1/R] \qquad \dfrac{\sigma_1, K > 1 \looparrowright true \qquad \mathcal{A}_2 \qquad \dfrac{\sigma_3, K > 1 \looparrowright false}{\sigma_3, CB \Downarrow \sigma_3}}{\sigma_1, CB \Downarrow \sigma_3}}{\sigma_0, C \Downarrow \sigma_3}$$

Soit C: **while** *true* **do skip**

Montrer que $\forall \sigma, \sigma' \in \mathbb{M}$ il n'y a pas de moyen de déduire $\sigma, C \Downarrow \sigma'$

Nous voulons prouver une propriété P sur l'exécution d'un programme. Formellement nous voulons montrer
$\forall \sigma, C, \sigma' \ (\sigma, C \Downarrow \sigma') \implies P(\sigma, C, \sigma')$
Une preuve par induction sur la dérivation étudie la preuve sur les 7 règles définies précédemment.

1. skip : $\forall \sigma \in \mathbb{M},\ P(\sigma, \textbf{skip}, \sigma)$
2. séquence :

1. skip : $\forall \sigma \in \mathbb{M},\ P(\sigma, \textbf{skip}, \sigma)$
2. séquence :$\forall \sigma, \sigma', \sigma'', C_1, C_2,\ (\sigma, C_1 \Downarrow \sigma') \wedge (\sigma', C_2 \Downarrow \sigma'') \wedge P(\sigma, C_1, \sigma') \wedge P(\sigma', C_2, \sigma'') \implies P(\sigma, C_1; C_2, \sigma'')$
3. attribution :

1. skip : $\forall \sigma \in \mathbb{M},\ P(\sigma, \textbf{skip}, \sigma)$
2. séquence :$\forall \sigma, \sigma', \sigma'', C_1, C_2,\ (\sigma, C_1 \Downarrow \sigma') \wedge (\sigma', C_2 \Downarrow \sigma'') \wedge P(\sigma, C_1, \sigma') \wedge P(\sigma', C_2, \sigma'') \implies P(\sigma, C_1; C_2, \sigma'')$
3. attribution :$\forall \sigma, \sigma', a, k,\ (\sigma, a \looparrowright k) \wedge \sigma' = \sigma[k/x] \implies P(\sigma, x := a, \sigma')$
4. disjonction de cas
(true) :

1. skip : $\forall \sigma \in \mathbb{M},\ P(\sigma, \textbf{skip}, \sigma)$
2. séquence : $\forall \sigma, \sigma', \sigma'', C_1, C_2,\ (\sigma, C_1 \Downarrow \sigma') \wedge (\sigma', C_2 \Downarrow \sigma'') \wedge P(\sigma, C_1, \sigma') \wedge P(\sigma', C_2, \sigma'') \implies P(\sigma, C_1; C_2, \sigma'')$
3. attribution : $\forall \sigma, \sigma', a, k,\ (\sigma, a \looparrowright k) \wedge \sigma' = \sigma[k/x] \implies P(\sigma, x := a, \sigma')$
4. disjonction de cas (true) : $\forall \sigma, \sigma', b, C_1, C_2\, (\sigma, b \looparrowright \mathit{true}) \wedge (\sigma, C_1 \Downarrow \sigma') \wedge P(\sigma, C_1, \sigma') \implies P(\sigma, \textbf{if}\ \mathrm{b}\ \textbf{then}\ C_1\ \textbf{else}\ C_2, \sigma')$
5. disjonction de cas (false) : $\forall \sigma, \sigma', b, C_1, C_2\, (\sigma, b \looparrowright \mathit{false}) \wedge (\sigma, C_2 \Downarrow \sigma') \wedge P(\sigma, C_2, \sigma') \implies P(\sigma, \textbf{if}\ \mathrm{b}\ \textbf{then}\ C_1\ \textbf{else}\ C_2, \sigma')$
6. boucle (non exécutée) :

1. skip : $\forall \sigma \in \mathbb{M},\ P(\sigma, \textbf{skip}, \sigma)$
2. séquence :$\forall \sigma, \sigma', \sigma'', C_1, C_2,\ (\sigma, C_1 \Downarrow \sigma') \wedge (\sigma', C_2 \Downarrow \sigma'') \wedge P(\sigma, C_1, \sigma') \wedge P(\sigma', C_2, \sigma'') \implies P(\sigma, C_1; C_2, \sigma'')$
3. attribution :$\forall \sigma, \sigma', a, k,\ (\sigma, a \looparrowright k) \wedge \sigma' = \sigma[k/x] \implies P(\sigma, x := a, \sigma')$
4. disjonction de cas
(true) :$\forall \sigma, \sigma', b, C_1, C_2\ (\sigma, b \looparrowright \mathit{true}) \wedge (\sigma, C_1 \Downarrow \sigma') \wedge P(\sigma, C_1, \sigma') \implies P(\sigma, \textbf{if}\ \mathrm{b}\ \textbf{then}\ C_1\ \textbf{else}\ C_2, \sigma')$
5. disjonction de cas (false) :
$\forall \sigma, \sigma', b, C_1, C_2\ (\sigma, b \looparrowright \mathit{false}) \wedge (\sigma, C_2 \Downarrow \sigma') \wedge P(\sigma, C_2, \sigma') \implies P(\sigma, \textbf{if}\ \mathrm{b}\ \textbf{then}\ C_1\ \textbf{else}\ C_2, \sigma')$
6. boucle (non exécutée) :$\forall \sigma, b, C,\ (\sigma, b \looparrowright \mathit{false}) \implies P(\sigma, \textbf{while}\ \mathrm{b}\ \textbf{do}\ c, \sigma)$
7. boucle (exécutée) :

1. skip : $\forall \sigma \in \mathbb{M},\ P(\sigma, \textbf{skip}, \sigma)$
2. séquence :$\forall \sigma, \sigma', \sigma'', C_1, C_2,\ (\sigma, C_1 \Downarrow \sigma') \land (\sigma', C_2 \Downarrow \sigma'') \land P(\sigma, C_1, \sigma') \land P(\sigma', C_2, \sigma'') \implies P(\sigma, C_1; C_2, \sigma'')$
3. attribution :$\forall \sigma, \sigma', a, k,\ (\sigma, a \looparrowright k) \land \sigma' = \sigma[k/x] \implies P(\sigma, x := a, \sigma')$
4. disjonction de cas
   (true) :$\forall \sigma, \sigma', b, C_1, C_2\ (\sigma, b \looparrowright \mathit{true}) \land (\sigma, C_1 \Downarrow \sigma') \land P(\sigma, C_1, \sigma') \implies P(\sigma, \textbf{if}\ \mathrm{b}\ \textbf{then}\ C_1\ \textbf{else}\ C_2, \sigma')$
5. disjonction de cas (false) :
   $\forall \sigma, \sigma', b, C_1, C_2\ (\sigma, b \looparrowright \mathit{false}) \land (\sigma, C_2 \Downarrow \sigma') \land P(\sigma, C_2, \sigma') \implies P(\sigma, \textbf{if}\ \mathrm{b}\ \textbf{then}\ C_1\ \textbf{else}\ C_2, \sigma')$
6. boucle (non exécutée) :$\forall \sigma, b, C,\ (\sigma, b \looparrowright \mathit{false}) \implies P(\sigma, \textbf{while}\ \mathrm{b}\ \textbf{do}\ c, \sigma)$
7. boucle (exécutée) :$\forall \sigma, \sigma', \sigma'', b, C,\ (\sigma, b \looparrowright \mathit{true}) \land (\sigma, C \Downarrow \sigma') \land P(\sigma, C, \sigma') \land (\sigma', \textbf{while}\ \mathrm{b}\ \textbf{do}\ C \Downarrow \sigma'') \land P(\sigma', \textbf{while}\ \mathrm{b}\ \textbf{do}\ C, \sigma'') \implies P(\sigma, \textbf{while}\ \mathrm{b}\ \textbf{do}\ C, \sigma'')$

Lemme

Déterminisme des expressions arithmétiques :

$\forall \sigma, a, k_1, k_2,\ (\sigma, a \looparrowright k_1) \wedge (\sigma, a \looparrowright k_2) \implies k_1 = k_2$

Idem pour les conditions (expressions booléennes).

Théorème

Déterminisme des programmes IMP :

$\forall \sigma, \sigma_1, \sigma_2, C,\ (\sigma, C \Downarrow \sigma_1) \wedge (\sigma, C \Downarrow \sigma_2) \implies \sigma_1 = \sigma_2$

Au lieu de considérer la transformation de l'état d'entrée en 1 étape, on peut décomposer et suivre le programme pas à pas.
On conserve toujours l'évaluation des expressions arithmétiques et booléennes ($\sigma, a \looparrowright k$).
On définit un jugement noté $\sigma, c \to \sigma', c'$, où $\sigma, \sigma' \in \mathbb{M}$ sont des états mémoire, et c, c' sont des commandes.
σ, c se réduit en σ', c' par les règles d'inférences suivantes :

1. attribution :

1. attribution : $\dfrac{\sigma, a \looparrowright k}{\sigma, x{:=}a \rightarrow \sigma', \textbf{skip}}\ \sigma' = \sigma[k/x]$
2. séquence :

1. attribution : $$\frac{\sigma, a \looparrowright k}{\sigma, x := a \rightarrow \sigma', \textbf{skip}} \quad \sigma' = \sigma[k/x]$$
2. séquence : $$\frac{\sigma, c_1 \rightarrow \sigma', c_1'}{\sigma, c_1 ; c_2 \rightarrow \sigma', c_1' ; c_2} \qquad \frac{}{\sigma, \textbf{skip} ; c_2 \rightarrow \sigma, c_2}$$
3. disjonction de cas :

1. attribution : $\dfrac{\sigma, a \looparrowright k}{\sigma, x{:=}a \rightarrow \sigma', \textbf{skip}}\ \sigma' = \sigma[k/x]$
2. séquence : $\dfrac{\sigma, c_1 \rightarrow \sigma', c_1'}{\sigma, c_1;c_2 \rightarrow \sigma', c_1';c_2}$ $\dfrac{}{\sigma, \textbf{skip};c_2 \rightarrow \sigma, c_2}$
3. disjonction de cas : $\dfrac{\sigma, b \looparrowright true}{\sigma, \textbf{if}\ \mathrm{b}\ \textbf{then}\ c_1\ \textbf{else}\ c_2 \rightarrow \sigma, c_1}$
$\dfrac{\sigma, b \looparrowright false}{\sigma, \textbf{if}\ \mathrm{b}\ \textbf{then}\ c_1\ \textbf{else}\ c_2 \rightarrow \sigma, c_2}$
4. boucle :

1. attribution : $\dfrac{\sigma, a \hookrightarrow k}{\sigma, x{:=}a \rightarrow \sigma', \textbf{skip}} \quad \sigma' = \sigma[k/x]$

2. séquence : $\dfrac{\sigma, c_1 \rightarrow \sigma', c_1'}{\sigma, c_1;c_2 \rightarrow \sigma', c_1';c_2} \qquad \dfrac{}{\sigma, \textbf{skip};c_2 \rightarrow \sigma, c_2}$

3. disjonction de cas : $\dfrac{\sigma, b \hookrightarrow \mathit{true}}{\sigma, \textbf{if}\ \mathrm{b}\ \textbf{then}\ c_1\ \textbf{else}\ c_2 \rightarrow \sigma, c_1}$

$$\dfrac{\sigma, b \hookrightarrow \mathit{false}}{\sigma, \textbf{if}\ \mathrm{b}\ \textbf{then}\ c_1\ \textbf{else}\ c_2 \rightarrow \sigma, c_2}$$

4. boucle : $\dfrac{\sigma, b \hookrightarrow \mathit{false}}{\sigma, \textbf{while}\ \mathrm{b}\ \textbf{do}\ c \rightarrow \sigma, \textbf{skip}}$

$$\dfrac{\sigma, b \hookrightarrow \mathit{true}}{\sigma, \textbf{while}\ \mathrm{b}\ \textbf{do}\ c \rightarrow \sigma, c;\ \textbf{while}\ \mathrm{b}\ \textbf{do}\ c}$$

On sera souvent intéressé par la clôture réflexive et transitive de $\rightarrow$ notée $\rightarrow^*$

Définition

$\rightarrow^*$ est définie par induction par les règles d'inférences

$$\frac{}{\sigma, c \rightarrow^* \sigma, c} \quad \text{et} \quad \frac{\sigma, c \rightarrow \sigma', c' \qquad \sigma', c' \rightarrow^* \sigma'', c''}{\sigma, c \rightarrow^* \sigma'', c''}$$

$\rightarrow^*$ peut s'interpréter en zéro ou plusieurs petits pas.