Sémantique opérationelle

Samuel Mimram

École polytechnique

L'option fondements de la programmation

On couvre ici les deux points suivants du programme de l'épreuve 3b *fondements de l'informatique* (partie *fondements de la programmation*) :

1. *Sémantique des langages de programmation* : sémantiques opérationnelles. Application à un langage impératif restreint (IMP).
2. *Systèmes de types* : types simples. Application à un langage fonctionnel simple (mini-ML sans polymorphisme), sûreté du typage.

Sémantique des langages de programmation

Un programme est simplement une suite de caractères dans un fichier.

Sémantique des langages de programmation

Un programme est simplement une suite de caractères dans un fichier.

Pour décrire ce qu'il fait, il faut en donner une *sémantique*.

Sémantique des langages de programmation

Un programme est simplement une suite de caractères dans un fichier.

Pour décrire ce qu'il fait, il faut en donner une *sémantique*.

Nous nous intéressons ici à décrire la façon dont il va s'exécuter et calculer un résultat, c'est la *sémantique opérationnelle*.

Sémantique opérationnelle

Nous verrons qu'il existe deux façons de formaliser la sémantique opérationnelle :

- sémantique à *grands pas* : décrit l'évaluation du programme vers une valeur
- sémantique à *petits pas* : décrit les étapes de réductions utilisées pour calculer

Première partie I

Le langage IMP

IMP

Introduisons IMP qui est un langage de programmation *impératif* :
les instructions modifient un état (la mémoire).

Syntaxe

Les programmes sont constitués de trois catégories syntaxiques :

– les *expressions arithmétiques*

$$a \quad ::= \quad \underline{n} \quad | \quad x \quad | \quad a + a' \quad | \quad a * a'$$

avec $x \in \mathcal{V}$ (dénombrable) une *variable*

Syntaxe

Les programmes sont constitués de trois catégories syntaxiques :

- les *expressions arithmétiques*

$$a \quad ::= \quad \underline{n} \quad | \quad x \quad | \quad a + a' \quad | \quad a * a'$$

  avec $x \in \mathcal{V}$ (dénombrable) une *variable*

- les *expressions booléennes*

$$b \quad ::= \quad \texttt{true} \quad | \quad \texttt{false} \quad | \quad a < a' \quad | \quad \texttt{not}\ b$$

Syntaxe

Les programmes sont constitués de trois catégories syntaxiques :

- les *expressions arithmétiques*

$$a \quad ::= \quad \underline{n} \quad | \quad x \quad | \quad a \texttt{ + } a' \quad | \quad a \texttt{ * } a'$$

  avec $x \in \mathcal{V}$ (dénombrable) une *variable*

- les *expressions booléennes*

$$b \quad ::= \quad \texttt{true} \quad | \quad \texttt{false} \quad | \quad a \texttt{ < } a' \quad | \quad \texttt{not } b$$

- les *commandes* ou *programmes*

$$c \quad ::=$$

Syntaxe

Les programmes sont constitués de trois catégories syntaxiques :

- les *expressions arithmétiques*

$$a \quad ::= \quad \underline{n} \quad | \quad x \quad | \quad a \texttt{ + } a' \quad | \quad a \texttt{ * } a'$$

  avec $x \in \mathcal{V}$ (dénombrable) une *variable*

- les *expressions booléennes*

$$b \quad ::= \quad \texttt{true} \quad | \quad \texttt{false} \quad | \quad a \texttt{ < } a' \quad | \quad \texttt{not } b$$

- les *commandes* ou *programmes*

$$c \quad ::= \quad \texttt{skip}$$

Syntaxe

Les programmes sont constitués de trois catégories syntaxiques :

- les *expressions arithmétiques*

$$a \quad ::= \quad \underline{n} \quad | \quad x \quad | \quad a \texttt{ + } a' \quad | \quad a \texttt{ * } a'$$

  avec $x \in \mathcal{V}$ (dénombrable) une *variable*

- les *expressions booléennes*

$$b \quad ::= \quad \texttt{true} \quad | \quad \texttt{false} \quad | \quad a \texttt{ < } a' \quad | \quad \texttt{not } b$$

- les *commandes* ou *programmes*

$$c \quad ::= \quad \texttt{skip} \quad | \quad x \texttt{ := } a$$

Syntaxe

Les programmes sont constitués de trois catégories syntaxiques :

- les *expressions arithmétiques*

$$a \quad ::= \quad \underline{n} \quad | \quad x \quad | \quad a \texttt{ + } a' \quad | \quad a \texttt{ * } a'$$

  avec $x \in \mathcal{V}$ (dénombrable) une *variable*

- les *expressions booléennes*

$$b \quad ::= \quad \texttt{true} \quad | \quad \texttt{false} \quad | \quad a \texttt{ < } a' \quad | \quad \texttt{not } b$$

- les *commandes* ou *programmes*

$$c \quad ::= \quad \texttt{skip} \quad | \quad x \texttt{ := } a \quad | \quad c\texttt{; } c'$$

Syntaxe

Les programmes sont constitués de trois catégories syntaxiques :

- les *expressions arithmétiques*

$$a \quad ::= \quad \underline{n} \quad | \quad x \quad | \quad a \texttt{ + } a' \quad | \quad a \texttt{ * } a'$$

  avec $x \in \mathcal{V}$ (dénombrable) une *variable*

- les *expressions booléennes*

$$b \quad ::= \quad \texttt{true} \quad | \quad \texttt{false} \quad | \quad a \texttt{ < } a' \quad | \quad \texttt{not } b$$

- les *commandes* ou *programmes*

$$\begin{aligned} c \quad ::= \quad & \texttt{skip} \quad | \quad x \texttt{ := } a \quad | \quad c\texttt{; } c' \\ & | \quad \texttt{if } b \texttt{ then } c \texttt{ else } c' \end{aligned}$$

Syntaxe

Les programmes sont constitués de trois catégories syntaxiques :

- les *expressions arithmétiques*

$$a \quad ::= \quad \underline{n} \quad | \quad x \quad | \quad a \texttt{ + } a' \quad | \quad a \texttt{ * } a'$$

  avec $x \in \mathcal{V}$ (dénombrable) une *variable*

- les *expressions booléennes*

$$b \quad ::= \quad \texttt{true} \quad | \quad \texttt{false} \quad | \quad a \texttt{ < } a' \quad | \quad \texttt{not } b$$

- les *commandes* ou *programmes*

$$\begin{aligned} c \quad ::= \quad & \texttt{skip} \quad | \quad x \texttt{ := } a \quad | \quad c\texttt{; } c' \\ & | \quad \texttt{if } b \texttt{ then } c \texttt{ else } c' \quad | \quad \texttt{while } b \texttt{ do } c \end{aligned}$$

Syntaxe

Les programmes sont constitués de trois catégories syntaxiques :

- les *expressions arithmétiques*

$$a \quad ::= \quad \underline{n} \quad | \quad x \quad | \quad a + a' \quad | \quad a * a'$$

  avec $x \in \mathcal{V}$ (dénombrable) une *variable*

- les *expressions booléennes*

$$b \quad ::= \quad \texttt{true} \quad | \quad \texttt{false} \quad | \quad a < a' \quad | \quad \texttt{not}\ b$$

- les *commandes* ou *programmes*

$$\begin{aligned} c \quad ::= \quad & \texttt{skip} \quad | \quad x := a \quad | \quad c ; c' \\ & | \quad \texttt{if}\ b\ \texttt{then}\ c\ \texttt{else}\ c' \quad | \quad \texttt{while}\ b\ \texttt{do}\ c \end{aligned}$$

(on pourrait ajouter d'autres opérations...)

Exemple

On a par exemple le programme suivant :

```
if x < 2 * 2 then (x := 1; y := 2) else x := 0
```

Exemple

Ceci n'est pas un programme valide :

```
x := 2 + true
```

Exemple

Le programme

```
y := 1; while not (x < 1) do (y := y * x; x := x - 1)
```

calcule

Exemple

Le programme

```
y := 1; while not (x < 1) do (y := y * x; x := x - 1)
```

calcule la factorielle de `x` dans `y`.

Deuxième partie II

Sémantique à grands pas

Sémantique intuitive

On s'attend à ce que l'exécution produise les résultats suivants :

- les expressions arithmétiques vont calculer des entiers,
- les expressions booléennes vont calculer des booléens,
- les commandes vont avoir un effet sur l'état courant de la mémoire.

Nous allons définir la *sémantique à grands pas*, c'est-à-dire spécifier formellement le résultat calculé par un programme.

Les états mémoire

On modélise l'ensemble des **états mémoire** par

Les états mémoire

On modélise l'ensemble des **états mémoire** par

$$\text{Etats} \quad = \quad \mathbb{N}^{\mathcal{V}}$$

Un état σ associe à toute variable $x \in \mathcal{V}$ sa valeur $\sigma(x) \in \mathbb{N}$.

Évaluation des expressions arithmétiques

L'**évaluation** des expressions arithmétiques est donnée par une fonction :

$$[\![-]\!]^{\mathsf{Aexp}} :$$

Évaluation des expressions arithmétiques

L'**évaluation** des expressions arithmétiques est donnée par une fonction :

$$[\![-]\!]^{\mathsf{Aexp}} : \mathsf{Aexp} \times \mathsf{Etats} \to \mathbb{N}$$

qui étant données

- une expression arithmétique a,
- un état σ

donne le résultat

$$[\![a]\!]^{\mathsf{Aexp}}_{\sigma}$$

de l'évaluation de l'expression arithmétique a dans l'état σ.

Évaluation des expressions arithmétiques

L'**évaluation** des expressions arithmétiques est donnée par une fonction :

$$[\![-]\!]^{\mathsf{Aexp}} : \mathsf{Aexp} \times \mathsf{Etats} \to \mathbb{N}$$

définie par induction sur a générée par la syntaxe

$$a \quad ::= \quad \underline{n} \quad | \quad x \quad | \quad a + a' \quad | \quad a * a'$$

Évaluation des expressions arithmétiques

L'**évaluation** des expressions arithmétiques est donnée par une fonction :

$$[\![-]\!]^{\mathsf{Aexp}} : \mathsf{Aexp} \times \mathsf{Etats} \to \mathbb{N}$$

définie par induction sur a par

$$[\![\underline{n}]\!]^{\mathsf{Aexp}}_{\sigma} =$$

Évaluation des expressions arithmétiques

L'**évaluation** des expressions arithmétiques est donnée par une fonction :

$$[\![-]\!]^{\mathsf{Aexp}} : \mathsf{Aexp} \times \mathsf{Etats} \to \mathbb{N}$$

définie par induction sur a par

$$[\![\underline{n}]\!]^{\mathsf{Aexp}}_{\sigma} = n$$

Évaluation des expressions arithmétiques

L'**évaluation** des expressions arithmétiques est donnée par une fonction :

$$[\![-]\!]^{\mathsf{Aexp}} : \mathsf{Aexp} \times \mathsf{Etats} \to \mathbb{N}$$

définie par induction sur a par

$$[\![\underline{n}]\!]^{\mathsf{Aexp}}_{\sigma} = n$$
$$[\![X]\!]^{\mathsf{Aexp}}_{\sigma} =$$

Évaluation des expressions arithmétiques

L'**évaluation** des expressions arithmétiques est donnée par une fonction :

$$[\![-]\!]^{\mathsf{Aexp}} : \mathsf{Aexp} \times \mathsf{Etats} \to \mathbb{N}$$

définie par induction sur a par

$$[\![\underline{n}]\!]^{\mathsf{Aexp}}_{\sigma} = n$$
$$[\![x]\!]^{\mathsf{Aexp}}_{\sigma} = \sigma(x)$$

Évaluation des expressions arithmétiques

L'**évaluation** des expressions arithmétiques est donnée par une fonction :

$$[\![-]\!]^{\mathsf{Aexp}} : \mathsf{Aexp} \times \mathsf{Etats} \to \mathbb{N}$$

définie par induction sur a par

$$\begin{aligned} [\![\underline{n}]\!]^{\mathsf{Aexp}}_{\sigma} &= n \\ [\![x]\!]^{\mathsf{Aexp}}_{\sigma} &= \sigma(x) \\ [\![a \ + \ a']\!]^{\mathsf{Aexp}}_{\sigma} &= \end{aligned}$$

Évaluation des expressions arithmétiques

L'**évaluation** des expressions arithmétiques est donnée par une fonction :

$$[\![-]\!]^{\mathsf{Aexp}} : \mathsf{Aexp} \times \mathsf{Etats} \to \mathbb{N}$$

définie par induction sur a par

$$\begin{aligned} [\![\underline{n}]\!]^{\mathsf{Aexp}}_{\sigma} &= n \\ [\![x]\!]^{\mathsf{Aexp}}_{\sigma} &= \sigma(x) \\ [\![a \ + \ a']\!]^{\mathsf{Aexp}}_{\sigma} &= [\![a]\!]^{\mathsf{Aexp}}_{\sigma} + [\![a']\!]^{\mathsf{Aexp}}_{\sigma} \end{aligned}$$

Évaluation des expressions arithmétiques

L'**évaluation** des expressions arithmétiques est donnée par une fonction :

$$[\![-]\!]^{\mathsf{Aexp}} : \mathsf{Aexp} \times \mathsf{Etats} \to \mathbb{N}$$

définie par induction sur a par

$$\begin{aligned}
[\![\underline{n}]\!]^{\mathsf{Aexp}}_{\sigma} &= n \\
[\![x]\!]^{\mathsf{Aexp}}_{\sigma} &= \sigma(x) \\
[\![a \ + \ a']\!]^{\mathsf{Aexp}}_{\sigma} &= [\![a]\!]^{\mathsf{Aexp}}_{\sigma} + [\![a']\!]^{\mathsf{Aexp}}_{\sigma} \\
[\![a \ * \ a']\!]^{\mathsf{Aexp}}_{\sigma} &=
\end{aligned}$$

Évaluation des expressions arithmétiques

L'**évaluation** des expressions arithmétiques est donnée par une fonction :

$$[\![-]\!]^{\mathsf{Aexp}} : \mathsf{Aexp} \times \mathsf{Etats} \to \mathbb{N}$$

définie par induction sur a par

$$\begin{aligned}
[\![\underline{n}]\!]_\sigma^{\mathsf{Aexp}} &= n \\
[\![x]\!]_\sigma^{\mathsf{Aexp}} &= \sigma(x) \\
[\![a \ + \ a']\!]_\sigma^{\mathsf{Aexp}} &= [\![a]\!]_\sigma^{\mathsf{Aexp}} + [\![a']\!]_\sigma^{\mathsf{Aexp}} \\
[\![a \ * \ a']\!]_\sigma^{\mathsf{Aexp}} &= [\![a]\!]_\sigma^{\mathsf{Aexp}} \times [\![a']\!]_\sigma^{\mathsf{Aexp}}
\end{aligned}$$

Évaluation des expressions booléennes

L'**évaluation** des expressions booléennes est donnée par une fonction :

$$[\![-]\!]^{\mathsf{Bexp}} : \mathsf{Bexp} \times \mathsf{Etats} \to \mathbb{B}$$

(avec $\mathbb{B} = \{\bot, \top\}$) définie par induction sur b par

Évaluation des expressions booléennes

L'**évaluation** des expressions booléennes est donnée par une fonction :

$$[\![-]\!]^{\mathsf{Bexp}} : \mathsf{Bexp} \times \mathsf{Etats} \to \mathbb{B}$$

(avec $\mathbb{B} = \{\bot, \top\}$) définie par induction sur b par

$$[\![\mathtt{true}]\!]^{\mathsf{Bexp}}_{\sigma} =$$

Évaluation des expressions booléennes

L'**évaluation** des expressions booléennes est donnée par une fonction :

$$[\![-]\!]^{\mathsf{Bexp}} : \mathsf{Bexp} \times \mathsf{Etats} \to \mathbb{B}$$

(avec $\mathbb{B} = \{\bot, \top\}$) définie par induction sur b par

$$[\![\mathtt{true}]\!]^{\mathsf{Bexp}}_{\sigma} = \top$$

Évaluation des expressions booléennes

L'**évaluation** des expressions booléennes est donnée par une fonction :

$$[\![-]\!]^{\mathsf{Bexp}} : \mathsf{Bexp} \times \mathsf{Etats} \to \mathbb{B}$$

(avec $\mathbb{B} = \{\bot, \top\}$) définie par induction sur b par

$$[\![\texttt{true}]\!]^{\mathsf{Bexp}}_{\sigma} = \top$$
$$[\![\texttt{false}]\!]^{\mathsf{Bexp}}_{\sigma} =$$

Évaluation des expressions booléennes

L'**évaluation** des expressions booléennes est donnée par une fonction :

$$[\![-]\!]^{\mathsf{Bexp}} : \mathsf{Bexp} \times \mathsf{Etats} \to \mathbb{B}$$

(avec $\mathbb{B} = \{\bot, \top\}$) définie par induction sur b par

$$[\![\texttt{true}]\!]^{\mathsf{Bexp}}_{\sigma} = \top$$
$$[\![\texttt{false}]\!]^{\mathsf{Bexp}}_{\sigma} = \bot$$

Évaluation des expressions booléennes

L'**évaluation** des expressions booléennes est donnée par une fonction :

$$[\![-]\!]^{\mathsf{Bexp}} : \mathsf{Bexp} \times \mathsf{Etats} \to \mathbb{B}$$

(avec $\mathbb{B} = \{\bot, \top\}$) définie par induction sur b par

$$[\![\texttt{true}]\!]^{\mathsf{Bexp}}_{\sigma} = \top$$
$$[\![\texttt{false}]\!]^{\mathsf{Bexp}}_{\sigma} = \bot$$

$$[\![a < a']\!]^{\mathsf{Bexp}}_{\sigma} =$$

Évaluation des expressions booléennes

L'**évaluation** des expressions booléennes est donnée par une fonction :

$$[\![-]\!]^{\mathsf{Bexp}} : \mathsf{Bexp} \times \mathsf{Etats} \to \mathbb{B}$$

(avec $\mathbb{B} = \{\bot, \top\}$) définie par induction sur b par

$$[\![\texttt{true}]\!]^{\mathsf{Bexp}}_{\sigma} = \top$$

$$[\![\texttt{false}]\!]^{\mathsf{Bexp}}_{\sigma} = \bot$$

$$[\![a \ < \ a']\!]^{\mathsf{Bexp}}_{\sigma} = \begin{cases} \top & \text{si } [\![a]\!]^{\mathsf{Aexp}}_{\sigma} < [\![a']\!]^{\mathsf{Aexp}}_{\sigma} \\ \bot & \text{si } [\![a]\!]^{\mathsf{Aexp}}_{\sigma} \geqslant [\![a']\!]^{\mathsf{Aexp}}_{\sigma} \end{cases}$$

Évaluation des expressions booléennes

L'**évaluation** des expressions booléennes est donnée par une fonction :

$$[\![-]\!]^{\mathsf{Bexp}} : \mathsf{Bexp} \times \mathsf{Etats} \to \mathbb{B}$$

(avec $\mathbb{B} = \{\bot, \top\}$) définie par induction sur b par

$$[\![\texttt{true}]\!]_\sigma^{\mathsf{Bexp}} = \top$$

$$[\![\texttt{false}]\!]_\sigma^{\mathsf{Bexp}} = \bot$$

$$[\![a \texttt{ < } a']\!]_\sigma^{\mathsf{Bexp}} = \begin{cases} \top & \text{si } [\![a]\!]_\sigma^{\mathsf{Aexp}} < [\![a']\!]_\sigma^{\mathsf{Aexp}} \\ \bot & \text{si } [\![a]\!]_\sigma^{\mathsf{Aexp}} \geqslant [\![a']\!]_\sigma^{\mathsf{Aexp}} \end{cases}$$

$$[\![\texttt{not } b]\!]_\sigma^{\mathsf{Bexp}} =$$

Évaluation des expressions booléennes

L'**évaluation** des expressions booléennes est donnée par une fonction :

$$[\![-]\!]^{\mathsf{Bexp}} : \mathsf{Bexp} \times \mathsf{Etats} \to \mathbb{B}$$

(avec $\mathbb{B} = \{\bot, \top\}$) définie par induction sur b par

$$[\![\texttt{true}]\!]_\sigma^{\mathsf{Bexp}} = \top$$

$$[\![\texttt{false}]\!]_\sigma^{\mathsf{Bexp}} = \bot$$

$$[\![a \texttt{ < } a']\!]_\sigma^{\mathsf{Bexp}} = \begin{cases} \top & \text{si } [\![a]\!]_\sigma^{\mathsf{Aexp}} < [\![a']\!]_\sigma^{\mathsf{Aexp}} \\ \bot & \text{si } [\![a]\!]_\sigma^{\mathsf{Aexp}} \geqslant [\![a']\!]_\sigma^{\mathsf{Aexp}} \end{cases}$$

$$[\![\texttt{not } b]\!]_\sigma^{\mathsf{Bexp}} = \begin{cases} \top & \text{is } [\![b]\!]_\sigma^{\mathsf{Bexp}} = \bot \\ \bot & \text{is } [\![b]\!]_\sigma^{\mathsf{Bexp}} = \top \end{cases}$$

Évaluation des commandes

L'**évaluation** des commandes est donnée par une fonction

$$[\![-]\!]^{\mathsf{Cexp}} :$$

Évaluation des commandes

L'**évaluation** des commandes est donnée par une fonction

$$[\![-]\!]^{\mathsf{Cexp}} : \mathsf{Cexp} \times \mathsf{Etats} \to \mathsf{Etats}$$

Évaluation des commandes

L'**évaluation** des commandes est donnée par une fonction

$$[\![-]\!]^{\mathsf{Cexp}} : \mathsf{Cexp} \times \mathsf{Etats} \to \mathsf{Etats}$$

Problème :

Évaluation des commandes

L'**évaluation** des commandes est donnée par une fonction

$$[\![-]\!]^{\mathsf{Cexp}} : \mathsf{Cexp} \times \mathsf{Etats} \to \mathsf{Etats}$$

Problème : quelle va être la fonction associée au programme suivant ?

$$\texttt{while true do } x \texttt{ := } x + 1$$

Évaluation des commandes

L'**évaluation** des commandes est donnée par une fonction *partielle*

$$[\![-]\!]^{\mathsf{Cexp}} : \mathsf{Cexp} \times \mathsf{Etats} \rightharpoonup \mathsf{Etats}$$

Problème : quelle va être la fonction associée au programme suivant ?

$$\texttt{while true do } x \texttt{ := } x + 1$$

Évaluation des commandes

L'**évaluation** des commandes est donnée par une *relation*

$$\mathcal{E} \subseteq \mathrm{Cexp} \times \mathrm{Etats} \times \mathrm{Etats}$$

Problème : quelle va être la fonction associée au programme suivant ?

$$\texttt{while true do } x \texttt{ := } x + 1$$

Il sera plus naturel de la définir comme une relation puis d'en déduire qu'elle définit une fonction partielle.

Évaluation des commandes

L'**évaluation** des commandes est donnée par une *relation*

$$\mathcal{E} \subseteq \mathsf{Cexp} \times \mathsf{Etats} \times \mathsf{Etats}$$

Problème : quelle va être la fonction associée au programme suivant ?

$$\texttt{while true do } x \texttt{ := } x + 1$$

On note

$$\langle\, c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma'$$

au lieu de $(c, \sigma, \sigma') \in \mathcal{E}$, qui signifie :

la commande c, partant de l'état σ peut aboutir dans l'état σ'.

Évaluation des commandes

Nous allons définir la relation $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma'$ par

Évaluation des commandes

Nous allons définir la relation $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma'$ par ~~induction sur c~~

Évaluation des commandes

Nous allons définir la relation $\mathcal{E}$ par une liste de règles de la forme

$$\frac{\langle\, c_1 \mid \sigma_1 \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma_1' \qquad \ldots \qquad \langle\, c_n \mid \sigma_n \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma_n'}{\langle\, c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'} \text{ (NOM)}$$

Cela signifie que la relation $\mathcal{E}$ est la plus petite relation (= sous-ensemble de $\mathrm{Cexp} \times \mathrm{Etats} \times \mathrm{Etats}$) telle que si les prémisses sont dans la relation, la conclusion l'est aussi.

Évaluation des commandes

Nous allons définir la relation $\mathcal{E}$ par une liste de règles de la forme

$$\frac{\langle\, c_1 \mid \sigma_1 \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma_1' \qquad \ldots \qquad \langle\, c_n \mid \sigma_n \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma_n'}{\langle\, c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'} \text{ (NOM)}$$

Cela signifie que la relation $\mathcal{E}$ est la plus petite relation (= sous-ensemble de $\mathrm{Cexp} \times \mathrm{Etats} \times \mathrm{Etats}$) telle que si les prémisses sont dans la relation, la conclusion l'est aussi.

On s'autorise des *conditions de bord* qui sont des conditions supplémentaires que doivent vérifier les prémisses pour impliquer la conclusion.

Définitions inductives

On essaye ici de définir $\mathcal{E} \subseteq U$ avec $U = \text{Cexp} \times \text{Etats} \times \text{Etats}$ par des règles de la forme

$$\frac{E_1 \quad \ldots \quad E_n}{E}$$

qui indiquent que

$$E_1 \in \mathcal{E} \wedge \ldots \wedge E_n \in \mathcal{E} \quad \Rightarrow \quad E \in \mathcal{E}$$

Définitions inductives

On essaye ici de définir $\mathcal{E} \subseteq U$ avec $U = \text{Cexp} \times \text{Etats} \times \text{Etats}$ par des règles de la forme

$$\frac{E_1 \quad \ldots \quad E_n}{E}$$

qui indiquent que

$$E_1 \in \mathcal{E} \wedge \ldots \wedge E_n \in \mathcal{E} \quad \Rightarrow \quad E \in \mathcal{E}$$

et de plus $\mathcal{E}$ est le plus petit ensemble vérifiant cette propriété.

Définitions inductives

On essaye ici de définir $\mathcal{E} \subseteq U$ avec $U = \text{Cexp} \times \text{Etats} \times \text{Etats}$ par des règles de la forme

$$\frac{E_1 \quad \dots \quad E_n}{E}$$

qui indiquent que

$$E_1 \in \mathcal{E} \wedge \dots \wedge E_n \in \mathcal{E} \quad \Rightarrow \quad E \in \mathcal{E}$$

et de plus $\mathcal{E}$ est le plus petit ensemble vérifiant cette propriété.

On dit que $\mathcal{E}' \subseteq U$ est **clos** lorsque l'implication est vérifiée pour tout règle.

Lemme
Il existe un plus petit sous-ensemble $\mathcal{E}$ de U clos.

Définitions inductives

On essaye ici de définir $\mathcal{E} \subseteq U$ avec $U = \text{Cexp} \times \text{Etats} \times \text{Etats}$ par des règles de la forme

$$\frac{E_1 \quad \ldots \quad E_n}{E}$$

qui indiquent que

$$E_1 \in \mathcal{E} \wedge \ldots \wedge E_n \in \mathcal{E} \quad \Rightarrow \quad E \in \mathcal{E}$$

et de plus $\mathcal{E}$ est le plus petit ensemble vérifiant cette propriété.

On dit que $\mathcal{E}' \subseteq U$ est **clos** lorsque l'implication est vérifiée pour tout règle.

Lemme
Il existe un plus petit sous-ensemble $\mathcal{E}$ de U clos.

Démonstration.
Il suffit de prendre $\mathcal{E} = \bigcap_{\mathcal{E}' \subseteq U \text{ clos}} \mathcal{E}'$. □

Définitions inductives

Étant donnée une propriété P sur les éléments de U, on a le principe d'induction suivant :

Proposition
Si pour toute règle

$$\frac{E_1 \quad \dots \quad E_n}{E}$$

on a

$$P(E_1) \wedge \ldots \wedge P(E_n) \quad \Rightarrow \quad P(E)$$

alors $P(E)$ *pour tout* $E \in \mathcal{E}$.

Définitions inductives

Étant donnée une propriété P sur les éléments de U, on a le principe d'induction suivant :

Proposition
Si pour toute règle

$$\frac{E_1 \quad \dots \quad E_n}{E}$$

on a

$$P(E_1) \wedge \ldots \wedge P(E_n) \quad \Rightarrow \quad P(E)$$

alors $P(E)$ *pour tout* $E \in \mathcal{E}$.

Démonstration.
L'ensemble $\mathcal{P} = \{E \in U \mid P(E)\}$ est clos donc $\mathcal{E} = \bigcap_{\mathcal{E}' \subseteq U \text{ clos}} \mathcal{E}' \subseteq \mathcal{P}$. □

Évaluation des commandes

– la règle pour `skip` est

Évaluation des commandes

– la règle pour `skip` est

$$\frac{}{\langle\ \texttt{skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma}\ (\textsc{skip})$$

Évaluation des commandes

– la règle pour `skip` est

$$\frac{}{\langle\ \texttt{skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma}\ (\textsc{skip})$$

– la règle pour l'*assignation* est

Évaluation des commandes

– la règle pour `skip` est

$$\frac{}{\langle\ \texttt{skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma}\ (\textsc{skip})$$

– la règle pour l'*assignation* est

$$\frac{}{\langle\ x\ \texttt{:=}\ a \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma[x \mapsto [\![a]\!]_\sigma^{\mathsf{Aexp}}]}\ (\textsc{set})$$

Évaluation des commandes

– la règle pour `skip` est

$$\frac{}{\langle\ \texttt{skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma}\ (\textsc{skip})$$

– la règle pour l'*assignation* est

$$\frac{}{\langle\ x\ \texttt{:=}\ a \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma[x \mapsto [\![a]\!]_\sigma^{\mathsf{Aexp}}]}\ (\textsc{set})$$

où

$$\sigma[x \mapsto n](y) = \begin{cases} n & \text{si } y = x \\ \sigma(y) & \text{si } y \neq x \end{cases}$$

Évaluation des commandes

– la règle pour la *séquence* est

Évaluation des commandes

– la règle pour la *séquence* est

$$\frac{\langle\, c_1 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma' \qquad \langle\, c_2 \mid \sigma' \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}{\langle\, c_1\,;\ c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}\ (\textsc{seq})$$

Évaluation des commandes

– la règle pour la *séquence* est

$$\frac{\langle\, c_1 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma' \qquad \langle\, c_2 \mid \sigma' \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}{\langle\, c_1\,;\ c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}\ (\textsc{seq})$$

– les règles pour le *branchement conditionnel* sont

Évaluation des commandes

– la règle pour la *séquence* est

$$\frac{\langle\, c_1 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma' \qquad \langle\, c_2 \mid \sigma' \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}{\langle\, c_1\,;\ c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}\ (\textsc{seq})$$

– les règles pour le *branchement conditionnel* sont

$$\frac{\langle\, c_1 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}{\langle\, \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}\ (\textsc{if}_\top) \qquad \text{si } [\![b]\!]^{\mathsf{Bexp}}_\sigma = \top$$

$$\frac{\langle\, c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}{\langle\, \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}\ (\textsc{if}_\bot) \qquad \text{si } [\![b]\!]^{\mathsf{Bexp}}_\sigma = \bot$$

Évaluation des commandes

– les règles pour les *boucles* sont

Évaluation des commandes

– les règles pour les *boucles* sont

$$\frac{\langle\, c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma' \qquad \langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma' \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}{\langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}\ (\textsc{while}_{\top}) \qquad \text{si } [\![b]\!]^{\mathsf{Bexp}}_{\sigma} = \top$$

$$\frac{}{\langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma}\ (\textsc{while}_{\bot}) \qquad \text{si } [\![b]\!]^{\mathsf{Bexp}}_{\sigma} = \bot$$

Évaluation des commandes

– les règles pour les *boucles* sont

$$\frac{\langle\, c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma' \qquad \langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma' \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}{\langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}\ (\textsc{while}_\top) \qquad \text{si } [\![b]\!]^{\mathsf{Bexp}}_\sigma = \top$$

$$\frac{}{\langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma}\ (\textsc{while}_\bot) \qquad \text{si } [\![b]\!]^{\mathsf{Bexp}}_\sigma = \bot$$

Notons l'intérêt de faire une définition comme une plus petite relation plutôt que par induction sur la structure des programmes !

Évaluation des commandes

Par exemple, étant donné σ avec $\sigma(y) = 1$, montrons que le programme

$$\texttt{if } y \texttt{ < 2 then (skip; } z \texttt{ := 5) else skip}$$

aboutit à l'état

Évaluation des commandes

Par exemple, étant donné σ avec $\sigma(y) = 1$, montrons que le programme

$$\texttt{if } y \texttt{ < 2 then (skip; } z \texttt{ := 5) else skip}$$

aboutit à l'état $\sigma[z \mapsto 5]$.

Évaluation des commandes

Par exemple, étant donné σ avec $\sigma(y) = 1$, montrons que le programme

$$\texttt{if } y \texttt{ < 2 then (skip; } z \texttt{ := 5) else skip}$$

aboutit à l'état $\sigma[z \mapsto 5]$. On a en effet :

$$\langle\ \texttt{if } y \texttt{ < 2 then (skip; } z \texttt{ := 5) else skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma[z \mapsto 5]$$

Évaluation des commandes

Par exemple, étant donné σ avec $\sigma(y) = 1$, montrons que le programme

$$\texttt{if } y \texttt{ < 2 then (skip; } z \texttt{ := 5) else skip}$$

aboutit à l'état $\sigma[z \mapsto 5]$. On a en effet :

$$\frac{\langle\ \texttt{skip; } z \texttt{ := 5} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma[z \mapsto 5]}{\langle\ \texttt{if } y \texttt{ < 2 then (skip; } z \texttt{ := 5) else skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma[z \mapsto 5]}\ (\mathrm{IF}_{\top})$$

Évaluation des commandes

Par exemple, étant donné σ avec $\sigma(y) = 1$, montrons que le programme

$$\texttt{if } y \texttt{ < 2 then (skip; } z \texttt{ := 5) else skip}$$

aboutit à l'état $\sigma[z \mapsto 5]$. On a en effet :

$$\dfrac{\dfrac{\langle\, \texttt{skip} \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma \qquad \langle\, z \texttt{ := 5} \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma[z \mapsto 5]}{\langle\, \texttt{skip; } z \texttt{ := 5} \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma[z \mapsto 5]}\,(\textsc{seq})}{\langle\, \texttt{if } y \texttt{ < 2 then (skip; } z \texttt{ := 5) else skip} \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma[z \mapsto 5]}\,(\textsc{if}_{\top})$$

Évaluation des commandes

Par exemple, étant donné σ avec $\sigma(y) = 1$, montrons que le programme

$$\texttt{if } y \texttt{ < 2 then (skip; } z \texttt{ := 5) else skip}$$

aboutit à l'état $\sigma[z \mapsto 5]$. On a en effet :

$$\dfrac{\dfrac{\dfrac{}{\langle\, \texttt{skip} \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma}\,(\textsc{skip}) \qquad \langle\, z \texttt{ := 5} \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma[z \mapsto 5]}{\langle\, \texttt{skip; } z \texttt{ := 5} \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma[z \mapsto 5]}\,(\textsc{seq})}{\langle\, \texttt{if } y \texttt{ < 2 then (skip; } z \texttt{ := 5) else skip} \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma[z \mapsto 5]}\,(\textsc{if}_\top)$$

Évaluation des commandes

Par exemple, étant donné σ avec $\sigma(y) = 1$, montrons que le programme

$$\texttt{if } y \texttt{ < 2 then (skip; } z \texttt{ := 5) else skip}$$

aboutit à l'état $\sigma[z \mapsto 5]$. On a en effet :

$$\dfrac{\dfrac{\dfrac{}{\langle\, \texttt{skip} \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma}\,(\textsc{skip}) \qquad \dfrac{}{\langle\, z \texttt{ := 5} \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma[z \mapsto 5]}\,(\textsc{set})}{\langle\, \texttt{skip; } z \texttt{ := 5} \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma[z \mapsto 5]}\,(\textsc{seq})}{\langle\, \texttt{if } y \texttt{ < 2 then (skip; } z \texttt{ := 5) else skip} \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma[z \mapsto 5]}\,(\textsc{if}_\top)$$

Évaluation des commandes

Par exemple, étant donné σ avec $\sigma(y) = 1$, montrons que le programme

$$\texttt{if } y \texttt{ < 2 then (skip; } z \texttt{ := 5) else skip}$$

aboutit à l'état $\sigma[z \mapsto 5]$. On a en effet :

$$\dfrac{\dfrac{\dfrac{}{\langle\, \texttt{skip} \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma}\,(\textsc{skip}) \qquad \dfrac{}{\langle\, z \texttt{ := 5} \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma[z \mapsto 5]}\,(\textsc{set})}{\langle\, \texttt{skip; } z \texttt{ := 5} \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma[z \mapsto 5]}\,(\textsc{seq})}{\langle\, \texttt{if } y \texttt{ < 2 then (skip; } z \texttt{ := 5) else skip} \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma[z \mapsto 5]}\,(\textsc{if}_{\top})$$

De façon plus générale, on peut montrer l'évaluation d'un programme par un

arbre de dérivation.

Induction sur les dérivations

Rappel : la définition de l'évaluation nous donne le principe d'induction suivant.

Théorème (induction sur les dérivations)
Considérons une propriété $P(\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma')$ *sur les évaluations. Supposons que pour chacune des règles*

$$\frac{\langle\ c_1 \mid \sigma_1\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma'_1 \qquad \ldots \qquad \langle\ c_n \mid \sigma_n\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma'_n}{\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}$$

si $P(\langle\ c_i \mid \sigma_i\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma'_i)$ *est valide pour tout* i *avec* $1 \leqslant i \leqslant n$ *alors* $P(\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma')$.
Alors $P(\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma')$ *est valide pour toute évaluation* $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma'$.

(autrement dit, on raisonne par récurrence sur la taille de l'arbre de dérivation)

Déterminisme de l'évaluation

Théorème
L'évaluation des commandes est déterministe :
si $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma'$ *et* $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma''$ *alors* $\sigma' = \sigma''$.

Déterminisme de l'évaluation

Théorème
L'évaluation des commandes est déterministe :
si $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma'$ *et* $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma''$ *alors* $\sigma' = \sigma''$.

Démonstration.
On montre par induction sur la dérivation de $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma'$ la propriété
$P(\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma')$:

pour toute dérivation de $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma''$ *de la commande* c *dans l'état* σ,
on a $\sigma' = \sigma''$.

□

Déterminisme de l'évaluation

Démonstration.

– Cas de la règle

$$\frac{}{\langle\ \texttt{skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma}\ (\textsc{skip})$$

Déterminisme de l'évaluation

Démonstration.

– Cas de la règle

$$\frac{}{\langle\ \texttt{skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma}\ (\textsc{skip})$$

Considérons une évaluation $\langle\ \texttt{skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma''$.

Démonstration.

– Cas de la règle

$$\frac{}{\langle\ \texttt{skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma}\ (\textsc{skip})$$

Considérons une évaluation $\langle\ \texttt{skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma''$.

Celle-ci admet une dérivation qui se termine par une règle de la forme

$$\frac{\cdots}{\langle\ \texttt{skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}$$

Démonstration.

– Cas de la règle

$$\frac{}{\langle\ \texttt{skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma}\ (\textsc{skip})$$

Considérons une évaluation $\langle\ \texttt{skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma''$.
Celle-ci admet une dérivation qui se termine par une règle de la forme

$$\frac{\cdots}{\langle\ \texttt{skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}$$

Cette règle est nécessairement (SKIP), car c'est la seule qui permette de dériver l'évaluation d'une commande `skip`.

Démonstration.

– Cas de la règle

$$\frac{}{\langle\ \texttt{skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma}\ (\textsc{skip})$$

Considérons une évaluation $\langle\ \texttt{skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma''$.
Celle-ci admet une dérivation qui se termine par une règle de la forme

$$\frac{\cdots}{\langle\ \texttt{skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}$$

Cette règle est nécessairement (SKIP), car c'est la seule qui permette de dériver l'évaluation d'une commande `skip`.
Donc $\sigma'' = \sigma$.

Déterminisme de l'évaluation

Démonstration.

– Cas de la règle $\dfrac{\langle\, c_1 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}{\langle\, \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}\ (\mathrm{IF}_\top)$.

Déterminisme de l'évaluation

Démonstration.

– Cas de la règle $\dfrac{\langle\, c_1 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}{\langle\, \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}\ (\mathrm{IF}_\top)$. On a $[\![b]\!]_\sigma^{\mathsf{Bexp}} = \top$.

Déterminisme de l'évaluation

Démonstration.

– Cas de la règle $\dfrac{\langle\, c_1 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}{\langle\, \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}\ (\mathrm{IF}_\top)$. On a $[\![b]\!]_\sigma^{\mathsf{Bexp}} = \top$. Considérons une évaluation $\langle\, \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''$.

Démonstration.

– Cas de la règle $\dfrac{\langle\, c_1 \mid \sigma \,\rangle \longrightarrow \sigma'}{\langle\, \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}\ (\mathrm{IF}_\top)$. On a $[\![b]\!]_\sigma^{\mathrm{Bexp}} = \top$.
Considérons une évaluation $\langle\, \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''$.
Sa dérivation se termine par une règle de la forme

$$\frac{\cdots}{\langle\, \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}$$

Démonstration.

– Cas de la règle $\dfrac{\langle\, c_1 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}{\langle\, \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}\ (\mathrm{IF}_\top)$. On a $[\![b]\!]^{\mathrm{Bexp}}_\sigma = \top$.
Considérons une évaluation $\langle\, \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''$.
Sa dérivation se termine par une règle de la forme

$$\frac{\cdots}{\langle\, \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}$$

Cette règle est nécessairement $(\mathrm{IF}_\top)$ ou $(\mathrm{IF}_\bot)$,

Démonstration.

– Cas de la règle $\dfrac{\langle\, c_1 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}{\langle\, \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}\ (\mathrm{IF}_\top)$. On a $[\![b]\!]_\sigma^{\mathrm{Bexp}} = \top$.
Considérons une évaluation $\langle\, \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''$.
Sa dérivation se termine par une règle de la forme

$$\frac{\cdots}{\langle\, \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}$$

Cette règle est nécessairement $(\mathrm{IF}_\top)$ ou $(\mathrm{IF}_\bot)$, et c'est le premier car $[\![b]\!]_\sigma^{\mathrm{Bexp}} = \top$.

Démonstration.

– Cas de la règle $\dfrac{\langle\, c_1 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}{\langle\, \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}\ (\mathrm{IF}_\top)$. On a $[\![b]\!]_\sigma^{\mathrm{Bexp}} = \top$.
Considérons une évaluation $\langle\, \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''$.
Sa dérivation se termine par une règle de la forme

$$\frac{\cdots}{\langle\, \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}$$

Cette règle est nécessairement $(\mathrm{IF}_\top)$ ou $(\mathrm{IF}_\bot)$, et c'est le premier car $[\![b]\!]_\sigma^{\mathrm{Bexp}} = \top$.
La dérivation est donc de la forme

$$\dfrac{\dfrac{\vdots}{\langle\, c_1 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}}{\langle\, \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}\ (\mathrm{IF}_\top)$$

Démonstration.

– Cas de la règle $\dfrac{\langle\, c_1 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}{\langle\, \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}\ (\mathrm{IF}_\top)$. On a $[\![b]\!]^{\mathrm{Bexp}}_\sigma = \top$.
Considérons une évaluation $\langle\, \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''$.
Sa dérivation se termine par une règle de la forme

$$\frac{\cdots}{\langle\, \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}$$

Cette règle est nécessairement $(\mathrm{IF}_\top)$ ou $(\mathrm{IF}_\bot)$, et c'est le premier car $[\![b]\!]^{\mathrm{Bexp}}_\sigma = \top$.
La dérivation est donc de la forme

$$\frac{\dfrac{\vdots}{\langle\, c_1 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}}{\langle\, \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}\ (\mathrm{IF}_\top)$$

On en déduit $\sigma' = \sigma''$ par hypothèse d'induction.

Démonstration.

– Cas de la règle $\dfrac{\langle\, c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma_1' \qquad \langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma_1' \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}{\langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}\ (\textsc{while}_\top)$.

Déterminisme de l'évaluation

Démonstration.

– Cas de la règle $\dfrac{\langle\, c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma_1' \qquad \langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma_1' \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}{\langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}\ (\textsc{while}_\top)$.
On a $[\![b]\!]_\sigma^{\mathsf{Bexp}} = \top$.

Démonstration.

– Cas de la règle $\dfrac{\langle\, c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma_1' \qquad \langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma_1' \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}{\langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}\ (\textsc{while}_\top)$.
On a $[\![b]\!]_\sigma^{\mathsf{Bexp}} = \top$. Considérons une évaluation $\langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''$,

Démonstration.

– Cas de la règle $\dfrac{\langle\, c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma_1' \qquad \langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma_1' \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}{\langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}\ (\textsc{while}_\top)$.
On a $[\![b]\!]_\sigma^{\mathsf{Bexp}} = \top$. Considérons une évaluation $\langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''$, sa dérivation est de la forme

$$\dfrac{\dfrac{\vdots}{\langle\, c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma_1''} \qquad \dfrac{\vdots}{\langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma_1'' \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}}{\langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}\ (\textsc{while}_\top)$$

Démonstration.

– Cas de la règle $\dfrac{\langle\, c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma_1' \qquad \langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma_1' \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}{\langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}\ (\text{WHILE}_\top)$.
On a $[\![b]\!]_\sigma^{\mathsf{Bexp}} = \top$. Considérons une évaluation $\langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''$, sa dérivation est de la forme

$$\dfrac{\dfrac{\vdots}{\langle\, c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma_1''} \qquad \dfrac{\vdots}{\langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma_1'' \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}}{\langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}\ (\text{WHILE}_\top)$$

Par induction on a $\sigma_1' = \sigma_1''$

Démonstration.

– Cas de la règle $\dfrac{\langle\, c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma_1' \qquad \langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma_1' \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}{\langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}\ (\textsc{while}_\top)$.
On a $[\![b]\!]_\sigma^{\mathsf{Bexp}} = \top$. Considérons une évaluation $\langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''$, sa dérivation est de la forme

$$\dfrac{\dfrac{\vdots}{\langle\, c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma_1''} \qquad \dfrac{\vdots}{\langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma_1'' \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}}{\langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}\ (\textsc{while}_\top)$$

Par induction on a $\sigma_1' = \sigma_1''$ puis $\sigma' = \sigma''$ par induction.

Déterminisme de l'évaluation

Démonstration.

– Cas de la règle $\dfrac{}{\langle\ \texttt{while}\ b\ \texttt{do}\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma}\ (\textsc{while}_\perp)$.

Déterminisme de l'évaluation

Démonstration.

– Cas de la règle $\dfrac{}{\langle\ \texttt{while}\ b\ \texttt{do}\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma}\ (\text{WHILE}_\perp)$.
On a $[\![b]\!]_\sigma^{\mathsf{Bexp}} = \perp$.

Déterminisme de l'évaluation

Démonstration.

– Cas de la règle $\dfrac{}{\langle\ \texttt{while}\ b\ \texttt{do}\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma}\ (\text{WHILE}_\perp)$.
On a $[\![b]\!]^{\text{Bexp}}_\sigma = \perp$. Considérons une évaluation $\langle\ \texttt{while}\ b\ \texttt{do}\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma''$.

Démonstration.

– Cas de la règle $\dfrac{}{\langle\ \texttt{while}\ b\ \texttt{do}\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma}\ (\textsc{while}_\bot)$.
On a $[\![b]\!]_\sigma^{\mathsf{Bexp}} = \bot$. Considérons une évaluation $\langle\ \texttt{while}\ b\ \texttt{do}\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma''$.
La dérivation de celle-ci termine nécessairement par $(\textsc{while}_\top)$ ou $(\textsc{while}_\bot)$,

Déterminisme de l'évaluation

Démonstration.

– Cas de la règle $\dfrac{}{\langle\ \texttt{while}\ b\ \texttt{do}\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma}\ (\textsc{while}_\bot)$.
On a $[\![b]\!]^{\mathsf{Bexp}}_\sigma = \bot$. Considérons une évaluation $\langle\ \texttt{while}\ b\ \texttt{do}\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma''$.
La dérivation de celle-ci termine nécessairement par $(\textsc{while}_\top)$ ou $(\textsc{while}_\bot)$, et c'est le second cas car $[\![b]\!]^{\mathsf{Bexp}}_\sigma = \bot$.

Démonstration.

– Cas de la règle $\dfrac{}{\langle\ \texttt{while}\ b\ \texttt{do}\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma}\ (\textsc{while}_\bot)$.
On a $[\![b]\!]_\sigma^{\mathsf{Bexp}} = \bot$. Considérons une évaluation $\langle\ \texttt{while}\ b\ \texttt{do}\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma''$.
La dérivation de celle-ci termine nécessairement par $(\textsc{while}_\top)$ ou $(\textsc{while}_\bot)$, et c'est le second cas car $[\![b]\!]_\sigma^{\mathsf{Bexp}} = \bot$.
Donc $\sigma'' = \sigma$.

Déterminisme de l'évaluation

Démonstration.

– Les autres cas sont similaires. □

Déterminisme de l'évaluation

On a montré :

Théorème
L'évaluation des commandes est déterministe :
si $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma'$ *et* $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma''$ *alors* $\sigma' = \sigma''$.

Déterminisme de l'évaluation

On a montré :

Théorème
L'évaluation des commandes est déterministe :
si $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma'$ *et* $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma''$ *alors* $\sigma' = \sigma''$.

La relation

$$\mathcal{E} \subseteq \mathrm{Cexp} \times \mathrm{Etats} \times \mathrm{Etats}$$

définit donc une fonction partielle

$$\mathrm{Cexp} \times \mathrm{Etats} \rightharpoonup \mathrm{Etats}$$

mais l'approche précédente s'étend à des cadres non déterministes !

Équivalence observationnelle

Deux commandes c_1 et c_2 sont **observationnellement équivalentes**, ce que l'on note $c_1 \sim c_2$, si pour tous états σ et σ' on a

$$\langle\, c_1 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma' \qquad \text{si et seulement si} \qquad \langle\, c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma'$$

Équivalence observationnelle

Deux commandes c_1 et c_2 sont **observationnellement équivalentes**, ce que l'on note $c_1 \sim c_2$, si pour tous états σ et σ' on a

$$\langle\, c_1 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma' \qquad \text{si et seulement si} \qquad \langle\, c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma'$$

Lemme
$\sim$ *est une relation d'équivalence sur les commandes.*

Équivalence observationnelle

Proposition
Pour toute commande c*, on a*

$$\texttt{if true then } c \texttt{ else skip} \sim c$$

Démonstration.
La dérivation de $\langle\ \texttt{if true then } c \texttt{ else skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma'$ est nécessairement de la forme

$$\dfrac{\dfrac{\vdots}{\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma'}}{\langle\ \texttt{if } b \texttt{ then } c \texttt{ else skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma'}\ (\mathrm{IF}_{\top})$$

Équivalence observationnelle

Proposition
Pour toute commande c*, on a*

$$\texttt{if true then } c \texttt{ else skip} \sim c$$

Démonstration.
La dérivation de $\langle\ \texttt{if true then } c \texttt{ else skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma'$ est nécessairement de la forme

$$\dfrac{\dfrac{\vdots}{\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma'}}{\langle\ \texttt{if } b \texttt{ then } c \texttt{ else skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma'}\ (\mathrm{IF}_{\top})$$

ce qui est équivalent à avoir $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma'$ dérivable. □

Équivalence observationnelle

De même :

Proposition
Pour toute commande c*, on a*

$$\texttt{while } b \texttt{ do } c \sim \texttt{if } b \texttt{ then } (c\texttt{; while } b \texttt{ do } c) \texttt{ else skip}$$

Troisième partie III

Sémantique à petits pas

Sémantique operationnelle à petits pas

La sémantique à grands pas a des inconvénients :

- on ne peut pas raisonner sur les exécutions (seulement sur les résultats)
- on ne peut pas raisonner sur des programmes qui ne terminent pas (comme un serveur web)

Nous allons définir la sémantique opérationnelle à petits pas en définissant la relation de *réduction*.

Réduction

La **reduction** est la plus petite relation sur

$$\mathcal{R} \subseteq \text{Cexp} \times \text{Etats} \times \text{Cexp} \times \text{Etats}$$

closes par des règles qui seront données par la suite.

Réduction

La **reduction** est la plus petite relation sur

$$\mathcal{R} \subseteq \mathsf{Cexp} \times \mathsf{Etats} \times \mathsf{Cexp} \times \mathsf{Etats}$$

closes par des règles qui seront données par la suite.

Un élément $(c, \sigma, c', \sigma') \in \mathcal{R}$ sera noté

$$\langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c' \mid \sigma'\ \rangle$$

Réduction

La **reduction** est la plus petite relation sur

$$\mathcal{R} \subseteq \mathrm{Cexp} \times \mathrm{Etats} \times \mathrm{Cexp} \times \mathrm{Etats}$$

closes par des règles qui seront données par la suite.

Un élément $(c, \sigma, c', \sigma') \in \mathcal{R}$ sera noté

$$\langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c' \mid \sigma'\ \rangle$$

ce qu'on peut lire :

dans l'état σ,
la commande c va commencer par exécuter une instruction qui aboutit à l'état σ',
après quoi la commande c' reste encore à exécuter.

Réduction des expressions arithmétiques

Nous allons commencer par définir la **réduction** des expressions arithmétiques comme un sous-ensemble de

$$\text{Aexp} \times \text{Etats} \times \text{Aexp} \times \text{Etats}$$

clos par les règles suivantes dont les éléments seront notés

$$\langle\ a \mid \sigma\ \rangle \longrightarrow \langle\ a' \mid \sigma'\ \rangle$$

Réduction des expressions arithmétiques

– variables :

Réduction des expressions arithmétiques

– variables :

$$\frac{}{\langle\ x \mid \sigma\ \rangle \longrightarrow \langle\ \underline{\sigma(x)} \mid \sigma\ \rangle}$$

Réduction des expressions arithmétiques

– variables :

$$\frac{}{\langle\, x \mid \sigma \,\rangle \longrightarrow \langle\, \underline{\sigma(x)} \mid \sigma \,\rangle}$$

– addition :

Réduction des expressions arithmétiques

– variables :

$$\frac{}{\langle\, x \mid \sigma \,\rangle \longrightarrow \langle\, \underline{\sigma(x)} \mid \sigma \,\rangle}$$

– addition :

$$\frac{\langle\, a_1 \mid \sigma \,\rangle \longrightarrow \langle\, a_1' \mid \sigma' \,\rangle}{\langle\, a_1 \;+\; a_2 \mid \sigma \,\rangle \longrightarrow \langle\, a_1' \;+\; a_2 \mid \sigma' \,\rangle}$$

Réduction des expressions arithmétiques

– variables :

$$\frac{}{\langle\, x \mid \sigma \,\rangle \longrightarrow \langle\, \underline{\sigma(x)} \mid \sigma \,\rangle}$$

– addition :

$$\frac{\langle\, a_1 \mid \sigma \,\rangle \longrightarrow \langle\, a_1' \mid \sigma' \,\rangle}{\langle\, a_1 \;+\; a_2 \mid \sigma \,\rangle \longrightarrow \langle\, a_1' \;+\; a_2 \mid \sigma' \,\rangle}$$

$$\frac{\langle\, a \mid \sigma \,\rangle \longrightarrow \langle\, a' \mid \sigma' \,\rangle}{\langle\, \underline{n} \;+\; a \mid \sigma \,\rangle \longrightarrow \langle\, \underline{n} \;+\; a' \mid \sigma' \,\rangle}$$

Réduction des expressions arithmétiques

– variables :

$$\frac{}{\langle\, x \mid \sigma \,\rangle \longrightarrow \langle\, \underline{\sigma(x)} \mid \sigma \,\rangle}$$

– addition :

$$\frac{\langle\, a_1 \mid \sigma \,\rangle \longrightarrow \langle\, a_1' \mid \sigma' \,\rangle}{\langle\, a_1 \;+\; a_2 \mid \sigma \,\rangle \longrightarrow \langle\, a_1' \;+\; a_2 \mid \sigma' \,\rangle}$$

$$\frac{\langle\, a \mid \sigma \,\rangle \longrightarrow \langle\, a' \mid \sigma' \,\rangle}{\langle\, \underline{n} \;+\; a \mid \sigma \,\rangle \longrightarrow \langle\, \underline{n} \;+\; a' \mid \sigma' \,\rangle}$$

$$\frac{}{\langle\, \underline{m} \;+\; \underline{n} \mid \sigma \,\rangle \longrightarrow \langle\, \underline{m+n} \mid \sigma \,\rangle}$$

Réduction des expressions arithmétiques

– variables :

$$\frac{}{\langle\ x \mid \sigma\ \rangle \longrightarrow \langle\ \underline{\sigma(x)} \mid \sigma\ \rangle}$$

– addition :

$$\frac{\langle\ a_1 \mid \sigma\ \rangle \longrightarrow \langle\ a_1' \mid \sigma'\ \rangle}{\langle\ a_1\ +\ a_2 \mid \sigma\ \rangle \longrightarrow \langle\ a_1'\ +\ a_2 \mid \sigma'\ \rangle}$$

$$\frac{\langle\ a \mid \sigma\ \rangle \longrightarrow \langle\ a' \mid \sigma'\ \rangle}{\langle\ \underline{n}\ +\ a \mid \sigma\ \rangle \longrightarrow \langle\ \underline{n}\ +\ a' \mid \sigma'\ \rangle}$$

$$\frac{}{\langle\ \underline{m}\ +\ \underline{n} \mid \sigma\ \rangle \longrightarrow \langle\ \underline{m+n} \mid \sigma\ \rangle}$$

– multiplication : similaire

Réduction des expressions booléennes

On définit la **réduction** des opérations booléennes comme un sous-ensemble de

$$\text{Bexp} \times \text{Etats} \times \text{Bexp} \times \text{Etats}$$

clos par les règles suivantes.

Réduction des expressions booléennes

– comparaison :

Réduction des expressions booléennes

– comparaison :

$$\frac{\langle\ a_1 \mid \sigma\ \rangle \longrightarrow \langle\ a_1' \mid \sigma'\ \rangle}{\langle\ a_1 < a_2 \mid \sigma\ \rangle \longrightarrow \langle\ a_1' < a_2 \mid \sigma'\ \rangle}$$

Réduction des expressions booléennes

– comparaison :

$$\frac{\langle\ a_1 \mid \sigma\ \rangle \longrightarrow \langle\ a_1' \mid \sigma'\ \rangle}{\langle\ a_1 < a_2 \mid \sigma\ \rangle \longrightarrow \langle\ a_1' < a_2 \mid \sigma'\ \rangle}$$

$$\frac{\langle\ a \mid \sigma\ \rangle \longrightarrow \langle\ a' \mid \sigma'\ \rangle}{\langle\ \underline{n} < a \mid \sigma\ \rangle \longrightarrow \langle\ \underline{n} < a' \mid \sigma'\ \rangle}$$

Réduction des expressions booléennes

– comparaison :

$$\frac{\langle\ a_1 \mid \sigma\ \rangle \longrightarrow \langle\ a_1' \mid \sigma'\ \rangle}{\langle\ a_1 < a_2 \mid \sigma\ \rangle \longrightarrow \langle\ a_1' < a_2 \mid \sigma'\ \rangle}$$

$$\frac{\langle\ a \mid \sigma\ \rangle \longrightarrow \langle\ a' \mid \sigma'\ \rangle}{\langle\ \underline{n} < a \mid \sigma\ \rangle \longrightarrow \langle\ \underline{n} < a' \mid \sigma'\ \rangle}$$

$$\frac{}{\langle\ \underline{m} < \underline{n} \mid \sigma\ \rangle \longrightarrow \langle\ \texttt{true} \mid \sigma\ \rangle} \qquad \text{si } m < n$$

Réduction des expressions booléennes

– comparaison :

$$\frac{\langle\ a_1 \mid \sigma\ \rangle \longrightarrow \langle\ a_1' \mid \sigma'\ \rangle}{\langle\ a_1 < a_2 \mid \sigma\ \rangle \longrightarrow \langle\ a_1' < a_2 \mid \sigma'\ \rangle}$$

$$\frac{\langle\ a \mid \sigma\ \rangle \longrightarrow \langle\ a' \mid \sigma'\ \rangle}{\langle\ \underline{n} < a \mid \sigma\ \rangle \longrightarrow \langle\ \underline{n} < a' \mid \sigma'\ \rangle}$$

$$\frac{}{\langle\ \underline{m} < \underline{n} \mid \sigma\ \rangle \longrightarrow \langle\ \texttt{true} \mid \sigma\ \rangle} \quad \text{si } m < n$$

$$\frac{}{\langle\ \underline{m} < \underline{n} \mid \sigma\ \rangle \longrightarrow \langle\ \texttt{false} \mid \sigma\ \rangle} \quad \text{si } m \geqslant n$$

Réduction des expressions booléennes

– négation :

Réduction des expressions booléennes

– négation :

$$\frac{\langle\ b \mid \sigma\ \rangle \longrightarrow \langle\ b' \mid \sigma'\ \rangle}{\langle\ \texttt{not}\ \ b \mid \sigma\ \rangle \longrightarrow \langle\ \texttt{not}\ \ b' \mid \sigma'\ \rangle}$$

Réduction des expressions booléennes

– négation :

$$\frac{\langle\ b \mid \sigma\ \rangle \longrightarrow \langle\ b' \mid \sigma'\ \rangle}{\langle\ \texttt{not}\ \ b \mid \sigma\ \rangle \longrightarrow \langle\ \texttt{not}\ \ b' \mid \sigma'\ \rangle}$$

$$\frac{}{\langle\ \texttt{not false} \mid \sigma\ \rangle \longrightarrow \langle\ \texttt{true} \mid \sigma\ \rangle}$$

Réduction des expressions booléennes

– négation :

$$\frac{\langle\ b \mid \sigma\ \rangle \longrightarrow \langle\ b' \mid \sigma'\ \rangle}{\langle\ \texttt{not}\ \ b \mid \sigma\ \rangle \longrightarrow \langle\ \texttt{not}\ \ b' \mid \sigma'\ \rangle}$$

$$\frac{}{\langle\ \texttt{not false} \mid \sigma\ \rangle \longrightarrow \langle\ \texttt{true} \mid \sigma\ \rangle}$$

$$\frac{}{\langle\ \texttt{not true} \mid \sigma\ \rangle \longrightarrow \langle\ \texttt{false} \mid \sigma\ \rangle}$$

Réduction des commandes

Nous pouvons finalement définir la réduction des commandes comme le plus petit sous-ensemble de

$$\text{Cexp} \times \text{Etats} \times \text{Cexp} \times \text{Etats}$$

clos par les règles suivantes.

Réduction des commandes

- variables :

Réduction des commandes

– variables :

$$\frac{\langle\, a \mid \sigma \,\rangle \longrightarrow \langle\, a' \mid \sigma' \,\rangle}{\langle\, x \;:=\; a \mid \sigma \,\rangle \longrightarrow \langle\, x \;:=\; a' \mid \sigma' \,\rangle}\ (\textsc{set})$$

Réduction des commandes

– variables :

$$\frac{\langle\, a \mid \sigma \,\rangle \longrightarrow \langle\, a' \mid \sigma' \,\rangle}{\langle\, x \;:=\; a \mid \sigma \,\rangle \longrightarrow \langle\, x \;:=\; a' \mid \sigma' \,\rangle}\ (\textsc{set})$$

$$\frac{}{\langle\, x \;:=\; \underline{n} \mid \sigma \,\rangle \longrightarrow \langle\, \mathtt{skip} \mid \sigma[x \mapsto n] \,\rangle}\ (\textsc{set}_n)$$

Réduction des commandes

– variables :

$$\frac{\langle\, a \mid \sigma \,\rangle \longrightarrow \langle\, a' \mid \sigma' \,\rangle}{\langle\, x \;:=\; a \mid \sigma \,\rangle \longrightarrow \langle\, x \;:=\; a' \mid \sigma' \,\rangle}\ (\textsc{set})$$

$$\frac{}{\langle\, x \;:=\; \underline{n} \mid \sigma \,\rangle \longrightarrow \langle\, \texttt{skip} \mid \sigma[x \mapsto n] \,\rangle}\ (\textsc{set}_n)$$

– séquence :

Réduction des commandes

– variables :

$$\frac{\langle\, a \mid \sigma \,\rangle \longrightarrow \langle\, a' \mid \sigma' \,\rangle}{\langle\, x \;:=\; a \mid \sigma \,\rangle \longrightarrow \langle\, x \;:=\; a' \mid \sigma' \,\rangle}\ (\textsc{set})$$

$$\frac{}{\langle\, x \;:=\; \underline{n} \mid \sigma \,\rangle \longrightarrow \langle\, \mathtt{skip} \mid \sigma[x \mapsto n] \,\rangle}\ (\textsc{set}_n)$$

– séquence :

$$\frac{\langle\, c_1 \mid \sigma \,\rangle \longrightarrow \langle\, c_1' \mid \sigma' \,\rangle}{\langle\, c_1 ;\; c_2 \mid \sigma \,\rangle \longrightarrow \langle\, c_1' ;\; c_2 \mid \sigma' \,\rangle}\ (\textsc{seq}_{\textsc{g}})$$

Réduction des commandes

– variables :

$$\frac{\langle\, a \mid \sigma \,\rangle \longrightarrow \langle\, a' \mid \sigma' \,\rangle}{\langle\, x \;:=\; a \mid \sigma \,\rangle \longrightarrow \langle\, x \;:=\; a' \mid \sigma' \,\rangle}\ (\textsc{set})$$

$$\frac{}{\langle\, x \;:=\; \underline{n} \mid \sigma \,\rangle \longrightarrow \langle\, \texttt{skip} \mid \sigma[x \mapsto n] \,\rangle}\ (\textsc{set}_n)$$

– séquence :

$$\frac{\langle\, c_1 \mid \sigma \,\rangle \longrightarrow \langle\, c_1' \mid \sigma' \,\rangle}{\langle\, c_1;\; c_2 \mid \sigma \,\rangle \longrightarrow \langle\, c_1';\; c_2 \mid \sigma' \,\rangle}\ (\textsc{seq}_{\textsc{g}})$$

$$\frac{}{\langle\, \texttt{skip};\; c \mid \sigma \,\rangle \longrightarrow \langle\, c \mid \sigma \,\rangle}\ (\textsc{seq}_{\textsc{d}})$$

Réduction des commandes

– branchement conditionnel :

Réduction des commandes

– branchement conditionnel :

$$\frac{\langle\, b \mid \sigma \,\rangle \longrightarrow \langle\, b' \mid \sigma' \,\rangle}{\langle\, \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow \langle\, \texttt{if}\ b'\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma' \,\rangle}\ (\textsc{if})$$

Réduction des commandes

– branchement conditionnel :

$$\frac{\langle\, b \mid \sigma \,\rangle \longrightarrow \langle\, b' \mid \sigma' \,\rangle}{\langle\, \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow \langle\, \texttt{if}\ b'\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma' \,\rangle}\ (\mathrm{IF})$$

$$\frac{}{\langle\, \texttt{if true then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow \langle\, c_1 \mid \sigma \,\rangle}\ (\mathrm{IF}_\top)$$

Réduction des commandes

– branchement conditionnel :

$$\frac{\langle\, b \mid \sigma \,\rangle \longrightarrow \langle\, b' \mid \sigma' \,\rangle}{\langle\, \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow \langle\, \texttt{if}\ b'\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma' \,\rangle}\ (\mathrm{IF})$$

$$\frac{}{\langle\, \texttt{if true then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow \langle\, c_1 \mid \sigma \,\rangle}\ (\mathrm{IF}_\top)$$

$$\frac{}{\langle\, \texttt{if false then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow \langle\, c_2 \mid \sigma \,\rangle}\ (\mathrm{IF}_\bot)$$

Réduction des commandes

– branchement conditionnel :

$$\frac{\langle\, b \mid \sigma \,\rangle \longrightarrow \langle\, b' \mid \sigma' \,\rangle}{\langle\, \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow \langle\, \texttt{if}\ b'\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma' \,\rangle}\ (\textsc{if})$$

$$\frac{}{\langle\, \texttt{if true then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow \langle\, c_1 \mid \sigma \,\rangle}\ (\textsc{if}_\top)$$

$$\frac{}{\langle\, \texttt{if false then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow \langle\, c_2 \mid \sigma \,\rangle}\ (\textsc{if}_\bot)$$

– boucles :

Réduction des commandes

– branchement conditionnel :

$$\frac{\langle\, b \mid \sigma \,\rangle \longrightarrow \langle\, b' \mid \sigma' \,\rangle}{\langle\, \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow \langle\, \texttt{if}\ b'\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma' \,\rangle}\ (\textsc{if})$$

$$\frac{}{\langle\, \texttt{if true then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow \langle\, c_1 \mid \sigma \,\rangle}\ (\textsc{if}_\top)$$

$$\frac{}{\langle\, \texttt{if false then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow \langle\, c_2 \mid \sigma \,\rangle}\ (\textsc{if}_\bot)$$

– boucles :

$$\frac{}{\langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma \,\rangle \longrightarrow \langle\, \texttt{if}\ b\ \texttt{then}\ (c\texttt{;}\ \texttt{while}\ b\ \texttt{do}\ c)\ \texttt{else skip} \mid \sigma \,\rangle}\ (\textsc{while}$$

Réduction de commandes

Notons qu'une règle de la forme

$$\frac{\langle\, b \mid \sigma \,\rangle \longrightarrow \langle\, b' \mid \sigma' \,\rangle}{\langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma \,\rangle \longrightarrow \langle\, \texttt{while}\ b'\ \texttt{do}\ c \mid \sigma \,\rangle}$$

ne conviendrait pas pour les boucles : on veut évaluer b à chaque tour !

Suite de réductions

On note

$$\langle\ c_1 \mid \sigma_1\ \rangle \xrightarrow{*} \langle\ c_n \mid \sigma_n\ \rangle$$

lorsqu'il existe une **suite de réductions**

$$\langle\ c_1 \mid \sigma_1\ \rangle \to \langle\ c_2 \mid \sigma_2\ \rangle \to \langle\ c_3 \mid \sigma_3\ \rangle \to \ldots \to \langle\ c_n \mid \sigma_n\ \rangle$$

Suite de réductions

On note

$$\langle\ c_1 \mid \sigma_1\ \rangle \xrightarrow{*} \langle\ c_n \mid \sigma_n\ \rangle$$

lorsqu'il existe une **suite de réductions**

$$\langle\ c_1 \mid \sigma_1\ \rangle \to \langle\ c_2 \mid \sigma_2\ \rangle \to \langle\ c_3 \mid \sigma_3\ \rangle \to \ldots \to \langle\ c_n \mid \sigma_n\ \rangle$$

On appelle n sa *longueur*.

Voyons un exemple.

Suite de réductions

$$\langle \texttt{y := 1; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_0 \rangle$$

Suite de réductions

$$\langle\ \texttt{y := 1; while not (x < 1) do (y := y * x; x := x - 1)}\ |\ \sigma_0\ \rangle$$
$$\longrightarrow\ \langle\ \texttt{skip; while not (x < 1) do (y := y * x; x := x - 1)}\ |\ \sigma_1\ \rangle \quad \text{avec } \sigma_1 = \sigma_0$$

Suite de réductions

$$
\begin{aligned}
& \langle\, \texttt{y := 1; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_0 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{skip; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1 \,\rangle \quad \text{avec } \sigma_1 = \sigma_0 \\
\longrightarrow\ & \langle\, \texttt{while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1 \,\rangle
\end{aligned}
$$

Suite de réductions

$$
\begin{aligned}
& \langle\, \texttt{y := 1; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_0 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{skip; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1 \,\rangle \quad \text{avec } \sigma_1 = \sigma_0 \\
\longrightarrow\ & \langle\, \texttt{while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if not (x < 1) then (y := y * x; x := x - 1);} \ W \ \texttt{else skip} \mid \sigma_1 \,\rangle
\end{aligned}
$$

Suite de réductions

$$
\begin{aligned}
& \langle\, \texttt{y := 1; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_0 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{skip; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1 \,\rangle \quad \text{avec } \sigma_1 = \sigma_0 \\
\longrightarrow\ & \langle\, \texttt{while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if not (x < 1) then (y := y * x; x := x - 1);} \ W \ \texttt{else skip} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if not (3 < 1) then (y := y * x; x := x - 1);} \ W \ \texttt{else skip} \mid \sigma_1 \,\rangle
\end{aligned}
$$

Suite de réductions

$$
\begin{aligned}
& \langle\ \texttt{y := 1; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_0\ \rangle \\
\longrightarrow\ & \langle\ \texttt{skip; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1\ \rangle \quad \text{avec } \sigma_1 = \sigma_0 \\
\longrightarrow\ & \langle\ \texttt{while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1\ \rangle \\
\longrightarrow\ & \langle\ \texttt{if not (x < 1) then (y := y * x; x := x - 1);}\ W\ \texttt{else skip} \mid \sigma_1\ \rangle \\
\longrightarrow\ & \langle\ \texttt{if not (3 < 1) then (y := y * x; x := x - 1);}\ W\ \texttt{else skip} \mid \sigma_1\ \rangle \\
\longrightarrow\ & \langle\ \texttt{if not false then (y := y * x; x := x - 1);}\ W\ \texttt{else skip} \mid \sigma_1\ \rangle
\end{aligned}
$$

Suite de réductions

$$
\begin{aligned}
& \langle\ \texttt{y := 1; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_0\ \rangle \\
\longrightarrow\ & \langle\ \texttt{skip; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1\ \rangle \quad \text{avec } \sigma_1 = \sigma_0 \\
\longrightarrow\ & \langle\ \texttt{while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1\ \rangle \\
\longrightarrow\ & \langle\ \texttt{if not (x < 1) then (y := y * x; x := x - 1);}\ W\ \texttt{else skip} \mid \sigma_1\ \rangle \\
\longrightarrow\ & \langle\ \texttt{if not (3 < 1) then (y := y * x; x := x - 1);}\ W\ \texttt{else skip} \mid \sigma_1\ \rangle \\
\longrightarrow\ & \langle\ \texttt{if not false then (y := y * x; x := x - 1);}\ W\ \texttt{else skip} \mid \sigma_1\ \rangle \\
\longrightarrow\ & \langle\ \texttt{if true then (y := y * x; x := x - 1);}\ W\ \texttt{else skip} \mid \sigma_1\ \rangle
\end{aligned}
$$

Suite de réductions

$$
\begin{aligned}
& \langle\ \texttt{y := 1; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_0\ \rangle \\
\longrightarrow\ & \langle\ \texttt{skip; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1\ \rangle \quad \text{avec } \sigma_1 = \sigma_0 \\
\longrightarrow\ & \langle\ \texttt{while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1\ \rangle \\
\longrightarrow\ & \langle\ \texttt{if not (x < 1) then (y := y * x; x := x - 1);}\ W\ \texttt{else skip} \mid \sigma_1\ \rangle \\
\longrightarrow\ & \langle\ \texttt{if not (3 < 1) then (y := y * x; x := x - 1);}\ W\ \texttt{else skip} \mid \sigma_1\ \rangle \\
\longrightarrow\ & \langle\ \texttt{if not false then (y := y * x; x := x - 1);}\ W\ \texttt{else skip} \mid \sigma_1\ \rangle \\
\longrightarrow\ & \langle\ \texttt{if true then (y := y * x; x := x - 1);}\ W\ \texttt{else skip} \mid \sigma_1\ \rangle \\
\longrightarrow\ & \langle\ \texttt{(y := y * x; x := x - 1);}\ W \mid \sigma_1\ \rangle
\end{aligned}
$$

Suite de réductions

$$
\begin{aligned}
& \langle\, \texttt{y := 1; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_0 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{skip; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1 \,\rangle \quad \text{avec } \sigma_1 = \sigma_0 \\
\longrightarrow\ & \langle\, \texttt{while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if not (x < 1) then (y := y * x; x := x - 1);} \ W \ \texttt{else skip} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if not (3 < 1) then (y := y * x; x := x - 1);} \ W \ \texttt{else skip} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if not false then (y := y * x; x := x - 1);} \ W \ \texttt{else skip} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if true then (y := y * x; x := x - 1);} \ W \ \texttt{else skip} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(y := y * x; x := x - 1);} \ W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(y := 1 * x; x := x - 1);} \ W \mid \sigma_1 \,\rangle
\end{aligned}
$$

Suite de réductions

$$
\begin{aligned}
& \langle\ \texttt{y := 1; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_0\ \rangle \\
\longrightarrow\ & \langle\ \texttt{skip; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1\ \rangle \quad \text{avec } \sigma_1 = \sigma_0 \\
\longrightarrow\ & \langle\ \texttt{while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1\ \rangle \\
\longrightarrow\ & \langle\ \texttt{if not (x < 1) then (y := y * x; x := x - 1); } W \texttt{ else skip} \mid \sigma_1\ \rangle \\
\longrightarrow\ & \langle\ \texttt{if not (3 < 1) then (y := y * x; x := x - 1); } W \texttt{ else skip} \mid \sigma_1\ \rangle \\
\longrightarrow\ & \langle\ \texttt{if not false then (y := y * x; x := x - 1); } W \texttt{ else skip} \mid \sigma_1\ \rangle \\
\longrightarrow\ & \langle\ \texttt{if true then (y := y * x; x := x - 1); } W \texttt{ else skip} \mid \sigma_1\ \rangle \\
\longrightarrow\ & \langle\ \texttt{(y := y * x; x := x - 1); } W \mid \sigma_1\ \rangle \\
\longrightarrow\ & \langle\ \texttt{(y := 1 * x; x := x - 1); } W \mid \sigma_1\ \rangle \\
\longrightarrow\ & \langle\ \texttt{(y := 1 * 3; x := x - 1); } W \mid \sigma_1\ \rangle
\end{aligned}
$$

Suite de réductions

$$
\begin{aligned}
& \langle\ \texttt{y := 1; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_0\ \rangle \\
\longrightarrow\ & \langle\ \texttt{skip; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1\ \rangle \quad \text{avec } \sigma_1 = \sigma_0 \\
\longrightarrow\ & \langle\ \texttt{while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1\ \rangle \\
\longrightarrow\ & \langle\ \texttt{if not (x < 1) then (y := y * x; x := x - 1);}\ W\ \texttt{else skip} \mid \sigma_1\ \rangle \\
\longrightarrow\ & \langle\ \texttt{if not (3 < 1) then (y := y * x; x := x - 1);}\ W\ \texttt{else skip} \mid \sigma_1\ \rangle \\
\longrightarrow\ & \langle\ \texttt{if not false then (y := y * x; x := x - 1);}\ W\ \texttt{else skip} \mid \sigma_1\ \rangle \\
\longrightarrow\ & \langle\ \texttt{if true then (y := y * x; x := x - 1);}\ W\ \texttt{else skip} \mid \sigma_1\ \rangle \\
\longrightarrow\ & \langle\ \texttt{(y := y * x; x := x - 1);}\ W \mid \sigma_1\ \rangle \\
\longrightarrow\ & \langle\ \texttt{(y := 1 * x; x := x - 1);}\ W \mid \sigma_1\ \rangle \\
\longrightarrow\ & \langle\ \texttt{(y := 1 * 3; x := x - 1);}\ W \mid \sigma_1\ \rangle \\
\longrightarrow\ & \langle\ \texttt{(y := 3; x := x - 1);}\ W \mid \sigma_1\ \rangle
\end{aligned}
$$

Suite de réductions

$$
\begin{aligned}
& \langle\, \texttt{y := 1; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_0 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{skip; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1 \,\rangle \quad \text{avec } \sigma_1 = \sigma_0 \\
\longrightarrow\ & \langle\, \texttt{while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if not (x < 1) then (y := y * x; x := x - 1);}\ W\ \texttt{else skip} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if not (3 < 1) then (y := y * x; x := x - 1);}\ W\ \texttt{else skip} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if not false then (y := y * x; x := x - 1);}\ W\ \texttt{else skip} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if true then (y := y * x; x := x - 1);}\ W\ \texttt{else skip} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(y := y * x; x := x - 1);}\ W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(y := 1 * x; x := x - 1);}\ W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(y := 1 * 3; x := x - 1);}\ W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(y := 3; x := x - 1);}\ W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(skip; x := x - 1);}\ W \mid \sigma_1 \,\rangle
\end{aligned}
$$

Suite de réductions

$$
\begin{aligned}
& \langle\, \texttt{y := 1; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_0 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{skip; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1 \,\rangle \quad \text{avec } \sigma_1 = \sigma_0 \\
\longrightarrow\ & \langle\, \texttt{while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if not (x < 1) then (y := y * x; x := x - 1);}\ W\ \texttt{else skip} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if not (3 < 1) then (y := y * x; x := x - 1);}\ W\ \texttt{else skip} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if not false then (y := y * x; x := x - 1);}\ W\ \texttt{else skip} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if true then (y := y * x; x := x - 1);}\ W\ \texttt{else skip} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(y := y * x; x := x - 1);}\ W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(y := 1 * x; x := x - 1);}\ W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(y := 1 * 3; x := x - 1);}\ W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(y := 3; x := x - 1);}\ W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(skip; x := x - 1);}\ W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{x := x - 1;}\ W \mid \sigma_2 \,\rangle \qquad \text{avec } \sigma_2 = \sigma_1[\texttt{y} \mapsto 3]
\end{aligned}
$$

Suite de réductions

$$
\begin{aligned}
& \langle\, \texttt{y := 1; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_0 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{skip; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1 \,\rangle \quad \text{avec } \sigma_1 = \sigma_0 \\
\longrightarrow\ & \langle\, \texttt{while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if not (x < 1) then (y := y * x; x := x - 1); } W \texttt{ else skip} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if not (3 < 1) then (y := y * x; x := x - 1); } W \texttt{ else skip} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if not false then (y := y * x; x := x - 1); } W \texttt{ else skip} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if true then (y := y * x; x := x - 1); } W \texttt{ else skip} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(y := y * x; x := x - 1); } W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(y := 1 * x; x := x - 1); } W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(y := 1 * 3; x := x - 1); } W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(y := 3; x := x - 1); } W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(skip; x := x - 1); } W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{x := x - 1; } W \mid \sigma_2 \,\rangle \qquad \text{avec } \sigma_2 = \sigma_1[\texttt{y} \mapsto 3] \\
\longrightarrow\ & \langle\, \texttt{x := 3 - 1; } W \mid \sigma_2 \,\rangle
\end{aligned}
$$

Suite de réductions

$$
\begin{aligned}
& \langle\, \texttt{y := 1; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_0 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{skip; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1 \,\rangle \quad \text{avec } \sigma_1 = \sigma_0 \\
\longrightarrow\ & \langle\, \texttt{while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if not (x < 1) then (y := y * x; x := x - 1); } W \texttt{ else skip} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if not (3 < 1) then (y := y * x; x := x - 1); } W \texttt{ else skip} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if not false then (y := y * x; x := x - 1); } W \texttt{ else skip} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if true then (y := y * x; x := x - 1); } W \texttt{ else skip} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(y := y * x; x := x - 1); } W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(y := 1 * x; x := x - 1); } W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(y := 1 * 3; x := x - 1); } W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(y := 3; x := x - 1); } W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(skip; x := x - 1); } W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{x := x - 1; } W \mid \sigma_2 \,\rangle \qquad \text{avec } \sigma_2 = \sigma_1[\texttt{y} \mapsto 3] \\
\longrightarrow\ & \langle\, \texttt{x := 3 - 1; } W \mid \sigma_2 \,\rangle
\end{aligned}
$$

Suite de réductions

$$
\begin{aligned}
& \langle\, \texttt{y := 1; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_0 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{skip; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1 \,\rangle \quad \text{avec } \sigma_1 = \sigma_0 \\
\longrightarrow\ & \langle\, \texttt{while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if not (x < 1) then (y := y * x; x := x - 1); } W \texttt{ else skip} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if not (3 < 1) then (y := y * x; x := x - 1); } W \texttt{ else skip} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if not false then (y := y * x; x := x - 1); } W \texttt{ else skip} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if true then (y := y * x; x := x - 1); } W \texttt{ else skip} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(y := y * x; x := x - 1); } W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(y := 1 * x; x := x - 1); } W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(y := 1 * 3; x := x - 1); } W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(y := 3; x := x - 1); } W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(skip; x := x - 1); } W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{x := x - 1; } W \mid \sigma_2 \,\rangle \qquad \text{avec } \sigma_2 = \sigma_1[\texttt{y} \mapsto 3] \\
\longrightarrow\ & \langle\, \texttt{x := 3 - 1; } W \mid \sigma_2 \,\rangle
\end{aligned}
$$

Suite de réductions

$$
\begin{aligned}
& \langle\, \texttt{y := 1; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_0 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{skip; while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1 \,\rangle \quad \text{avec } \sigma_1 = \sigma_0 \\
\longrightarrow\ & \langle\, \texttt{while not (x < 1) do (y := y * x; x := x - 1)} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if not (x < 1) then (y := y * x; x := x - 1); } W \texttt{ else skip} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if not (3 < 1) then (y := y * x; x := x - 1); } W \texttt{ else skip} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if not false then (y := y * x; x := x - 1); } W \texttt{ else skip} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{if true then (y := y * x; x := x - 1); } W \texttt{ else skip} \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(y := y * x; x := x - 1); } W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(y := 1 * x; x := x - 1); } W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(y := 1 * 3; x := x - 1); } W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(y := 3; x := x - 1); } W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{(skip; x := x - 1); } W \mid \sigma_1 \,\rangle \\
\longrightarrow\ & \langle\, \texttt{x := x - 1; } W \mid \sigma_2 \,\rangle \qquad \text{avec } \sigma_2 = \sigma_1[\texttt{y} \mapsto 3] \\
\longrightarrow\ & \langle\, \texttt{x := 3 - 1; } W \mid \sigma_2 \,\rangle
\end{aligned}
$$

Déterminisme de la réduction

Théorème
La réduction est déterministe : si

$$\langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c_1 \mid \sigma_1\ \rangle \qquad \textit{et} \qquad \langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c_2 \mid \sigma_2\ \rangle$$

alors $c_1 = c_2$ *et* $\sigma_1 = \sigma_2$.

Déterminisme de la réduction

Théorème
La réduction est déterministe : si

$$\langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c_1 \mid \sigma_1\ \rangle \qquad \textit{et} \qquad \langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c_2 \mid \sigma_2\ \rangle$$

alors $c_1 = c_2$ *et* $\sigma_1 = \sigma_2$.

Démonstration.
Par induction sur la dérivation de la réduction $\langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c_1 \mid \sigma_1\ \rangle$. □

Déterminisme de la réduction

Il est important que l'évaluation soit déterministe, mais on pourrait faire sans pour la réduction.

Déterminisme de la réduction

Il est important que l'évaluation soit déterministe, mais on pourrait faire sans pour la réduction.

Par exemple, si on changeait les règles de l'addition en

$$\frac{\langle\, a_1 \mid \sigma \,\rangle \longrightarrow \langle\, a_1' \mid \sigma' \,\rangle}{\langle\, a_1 \;+\; a_2 \mid \sigma \,\rangle \longrightarrow \langle\, a_1' \;+\; a_2 \mid \sigma' \,\rangle} \qquad \frac{\langle\, a_2 \mid \sigma \,\rangle \longrightarrow \langle\, a_2' \mid \sigma' \,\rangle}{\langle\, a_1 \;+\; a_2 \mid \sigma \,\rangle \longrightarrow \langle\, a_1 \;+\; a_2' \mid \sigma' \,\rangle}$$

Déterminisme de la réduction

Il est important que l'évaluation soit déterministe, mais on pourrait faire sans pour la réduction.

Par exemple, si on changeait les règles de l'addition en

$$\frac{\langle\ a_1 \mid \sigma\ \rangle \longrightarrow \langle\ a_1' \mid \sigma'\ \rangle}{\langle\ a_1 + a_2 \mid \sigma\ \rangle \longrightarrow \langle\ a_1' + a_2 \mid \sigma'\ \rangle} \qquad \frac{\langle\ a_2 \mid \sigma\ \rangle \longrightarrow \langle\ a_2' \mid \sigma'\ \rangle}{\langle\ a_1 + a_2 \mid \sigma\ \rangle \longrightarrow \langle\ a_1 + a_2' \mid \sigma'\ \rangle}$$

on aurait

$$\langle\ (2+2)+(3+3) \mid \sigma\ \rangle \longrightarrow \langle\ 4+(3+3) \mid \sigma\ \rangle$$
$$\langle\ (2+2)+(3+3) \mid \sigma\ \rangle \longrightarrow \langle\ (2+2)+6 \mid \sigma\ \rangle$$

Déterminisme de la réduction

Par exemple, en C, le résultat du programme suivant dépend du compilateur :

```
int a;

int f(int i) {
  a = i;
  return i;
}

int main(void) {
  int b = f(1) + f(2);
  printf("a = %d\n", a);
  return 0;
}
```

Quatrième partie IV

Équivalence entre les sémantiques

Commandes irréductibles

Une commande c est **irréductible** dans un état σ s'il n'existe pas de réduction

$$\langle\, c \mid \sigma \,\rangle \longrightarrow \langle\, c' \mid \sigma' \,\rangle$$

(c'est-à-dire qu'on a fini le calcul)

Commandes irréductibles

Une commande c est **irréductible** dans un état σ s'il n'existe pas de réduction

$$\langle\, c \mid \sigma \,\rangle \longrightarrow \langle\, c' \mid \sigma' \,\rangle$$

(c'est-à-dire qu'on a fini le calcul)

Quelles sont les commandes irréductibles ?

Commandes irréductibles

Lemme
Pour tout état σ, une commande c est irréductible ssi $c = \texttt{skip}$.

Démonstration.

Commandes irréductibles

Lemme
Pour tout état σ, une commande c est irréductible ssi $c = \texttt{skip}$.

Démonstration.

$\Leftarrow$: il n'y a pas de règle permettant de réduire `skip`.

Commandes irréductibles

Lemme
Pour tout état σ, une commande c est irréductible ssi $c = \texttt{skip}$.

Démonstration.

$\Leftarrow$: il n'y a pas de règle permettant de réduire `skip`.

$\Rightarrow$: par induction sur c, pour tout $c \neq \texttt{skip}$ est réductible.

Commandes irréductibles

Lemme
Pour tout état σ, une commande c est irréductible ssi $c = \texttt{skip}$.

Démonstration.

$\Leftarrow$: il n'y a pas de règle permettant de réduire `skip`.

$\Rightarrow$: par induction sur c, pour tout $c \neq \texttt{skip}$ est réductible.

- Si $c = x \texttt{ := } a$, alors c est réductible par (SET).

Commandes irréductibles

Lemme
Pour tout état σ, une commande c est irréductible ssi $c = \texttt{skip}$.

Démonstration.

$\Leftarrow$: il n'y a pas de règle permettant de réduire `skip`.

$\Rightarrow$: par induction sur c, pour tout $c \neq \texttt{skip}$ est réductible.

- Si $c = x \texttt{ := } a$, alors c est réductible par (SET).
- Si $c = c_1 \texttt{ ; } c_2$, alors on distingue deux cas.

Commandes irréductibles

Lemme
Pour tout état σ, une commande c est irréductible ssi $c = \texttt{skip}$.

Démonstration.

$\Leftarrow$: il n'y a pas de règle permettant de réduire `skip`.

$\Rightarrow$: par induction sur c, pour tout $c \neq \texttt{skip}$ est réductible.

- Si $c = x \texttt{ := } a$, alors c est réductible par (SET).
- Si $c = c_1 \texttt{;} \ c_2$, alors on distingue deux cas.
  - Si $c_1 \neq \texttt{skip}$ alors c_1 est réductible et donc c est réductible par (SEQ_G).

Commandes irréductibles

Lemme
Pour tout état σ, une commande c est irréductible ssi $c = \texttt{skip}$.

Démonstration.

$\Leftarrow$: il n'y a pas de règle permettant de réduire `skip`.

$\Rightarrow$: par induction sur c, pour tout $c \neq \texttt{skip}$ est réductible.

- Si $c = x \texttt{ := } a$, alors c est réductible par (SET).
- Si $c = c_1 \texttt{;} \ c_2$, alors on distingue deux cas.
  - Si $c_1 \neq \texttt{skip}$ alors c_1 est réductible et donc c est réductible par (SEQ_G).
  - Si $c_1 = \texttt{skip}$ alors c est réductible par (SEQ_D).

Commandes irréductibles

Lemme
Pour tout état σ, une commande c est irréductible ssi $c = \texttt{skip}$.

Démonstration.

$\Leftarrow$: il n'y a pas de règle permettant de réduire `skip`.

$\Rightarrow$: par induction sur c, pour tout $c \neq \texttt{skip}$ est réductible.

- Si $c = x \texttt{ := } a$, alors c est réductible par (SET).
- Si $c = c_1 \texttt{;} \ c_2$, alors on distingue deux cas.
  - Si $c_1 \neq \texttt{skip}$ alors c_1 est réductible et donc c est réductible par (SEQ_G).
  - Si $c_1 = \texttt{skip}$ alors c est réductible par (SEQ_D).
- Autres cas en exercice, en utilisant les lemmes :

Commandes irréductibles

Lemme
Pour tout état σ, une commande c est irréductible ssi $c = \texttt{skip}$.

Démonstration.

$\Leftarrow$: il n'y a pas de règle permettant de réduire `skip`.

$\Rightarrow$: par induction sur c, pour tout $c \neq \texttt{skip}$ est réductible.

- Si $c = x \texttt{ := } a$, alors c est réductible par (SET).
- Si $c = c_1 \texttt{;} \ c_2$, alors on distingue deux cas.
  - Si $c_1 \neq \texttt{skip}$ alors c_1 est réductible et donc c est réductible par (SEQ_G).
  - Si $c_1 = \texttt{skip}$ alors c est réductible par (SEQ_D).
- Autres cas en exercice, en utilisant les lemmes :
  - une expression arithmétique est irréductible ssi elle est de la forme $\underline{n}$,

Commandes irréductibles

Lemme
Pour tout état σ, une commande c est irréductible ssi $c = \texttt{skip}$.

Démonstration.

$\Leftarrow$: il n'y a pas de règle permettant de réduire `skip`.

$\Rightarrow$: par induction sur c, pour tout $c \neq \texttt{skip}$ est réductible.

- Si $c = x \texttt{ := } a$, alors c est réductible par (SET).
- Si $c = c_1 \texttt{;} \ c_2$, alors on distingue deux cas.
  - Si $c_1 \neq \texttt{skip}$ alors c_1 est réductible et donc c est réductible par (SEQ_G).
  - Si $c_1 = \texttt{skip}$ alors c est réductible par (SEQ_D).
- Autres cas en exercice, en utilisant les lemmes :
  - une expression arithmétique est irréductible ssi elle est de la forme $\underline{n}$,
  - une expression booléenne est réductible ssi elle est de le forme `true` ou `false`. □

Équivalence entre les sémantiques

Les sémantiques à grands et petits pas coïncident :

Équivalence entre les sémantiques

Les sémantiques à grands et petits pas coïncident :

Théorème
On a équivalence entre

(i) $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma'$ *(dans la sémantique à grands pas),*

(ii) $\langle\ c \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma'\ \rangle$ *(dans la sémantique à petits pas).*

Équivalence entre les sémantiques

Nous allons avoir besoin de lemmes auxiliaires.

Équivalence entre les sémantiques

Lemme
Les expressions se réduisent vers leur valeur :

- *Étant donnée une expression arithmétique* a*, on a*

$$\langle\ a \mid \sigma\ \rangle \longrightarrow^{*} \langle\ \underline{[\![a]\!]_\sigma^{\mathrm{Aexp}}} \mid \sigma\ \rangle$$

- *Étant donnée une expression booléenne* b*, on a*

$$\langle\ b \mid \sigma\ \rangle \longrightarrow^{*} \langle\ \underline{[\![b]\!]_\sigma^{\mathrm{Bexp}}} \mid \sigma\ \rangle$$

avec $\underline{\top} = \mathtt{true}$ *et* $\underline{\bot} = \mathtt{false}$.

Équivalence entre les sémantiques

Lemme
Les expressions se réduisent vers leur valeur :

- *Étant donnée une expression arithmétique* a*, on a*

$$\langle\ a \mid \sigma\ \rangle \xrightarrow{*} \langle\ \underline{[\![a]\!]_\sigma^{\mathrm{Aexp}}} \mid \sigma\ \rangle$$

- *Étant donnée une expression booléenne* b*, on a*

$$\langle\ b \mid \sigma\ \rangle \xrightarrow{*} \langle\ \underline{[\![b]\!]_\sigma^{\mathrm{Bexp}}} \mid \sigma\ \rangle$$

avec $\underline{\top} = \mathtt{true}$ *et* $\underline{\bot} = \mathtt{false}$*.*

Par exemple, $\langle\ (3\ +\ 1)\ *\ (x\ +\ 5) \mid \sigma\ \rangle$

Équivalence entre les sémantiques

Lemme
Les expressions se réduisent vers leur valeur :

- *Étant donnée une expression arithmétique* a*, on a*

$$\langle\ a \mid \sigma\ \rangle \longrightarrow^{*} \langle\ \underline{[\![a]\!]_{\sigma}^{\mathrm{Aexp}}} \mid \sigma\ \rangle$$

- *Étant donnée une expression booléenne* b*, on a*

$$\langle\ b \mid \sigma\ \rangle \longrightarrow^{*} \langle\ \underline{[\![b]\!]_{\sigma}^{\mathrm{Bexp}}} \mid \sigma\ \rangle$$

avec $\underline{\top} = \mathtt{true}$ *et* $\underline{\bot} = \mathtt{false}$.

Par exemple, $\langle\ (3\ +\ 1)\ *\ (x\ +\ 5) \mid \sigma\ \rangle \longrightarrow \langle\ 4\ *\ (x\ +\ 5) \mid \sigma\ \rangle$

Équivalence entre les sémantiques

Lemme
Les expressions se réduisent vers leur valeur :

- *Étant donnée une expression arithmétique a, on a*

$$\langle\ a \mid \sigma\ \rangle \xrightarrow{*} \langle\ \underline{[\![a]\!]_\sigma^{\mathsf{Aexp}}} \mid \sigma\ \rangle$$

- *Étant donnée une expression booléenne b, on a*

$$\langle\ b \mid \sigma\ \rangle \xrightarrow{*} \langle\ \underline{[\![b]\!]_\sigma^{\mathsf{Bexp}}} \mid \sigma\ \rangle$$

avec $\underline{\top} = \mathtt{true}$ et $\underline{\bot} = \mathtt{false}$.

Par exemple, $\langle\ (3\ +\ 1)\ *\ (x\ +\ 5) \mid \sigma\ \rangle \longrightarrow \langle\ 4\ *\ (x\ +\ 5) \mid \sigma\ \rangle$
$\longrightarrow \langle\ 4\ *\ (6\ +\ 5) \mid \sigma\ \rangle$

Équivalence entre les sémantiques

Lemme
Les expressions se réduisent vers leur valeur :

- *Étant donnée une expression arithmétique a, on a*

$$\langle\ a \mid \sigma\ \rangle \xrightarrow{*} \langle\ \underline{[\![a]\!]_\sigma^{\mathsf{Aexp}}} \mid \sigma\ \rangle$$

- *Étant donnée une expression booléenne b, on a*

$$\langle\ b \mid \sigma\ \rangle \xrightarrow{*} \langle\ \underline{[\![b]\!]_\sigma^{\mathsf{Bexp}}} \mid \sigma\ \rangle$$

avec $\underline{\top} = \mathtt{true}$ et $\underline{\bot} = \mathtt{false}$.

Par exemple,
$$\begin{aligned}\langle\ (3\ +\ 1)\ *\ (x\ +\ 5) \mid \sigma\ \rangle &\longrightarrow \langle\ 4\ *\ (x\ +\ 5) \mid \sigma\ \rangle \\ &\longrightarrow \langle\ 4\ *\ (6\ +\ 5) \mid \sigma\ \rangle \\ &\longrightarrow \langle\ 4\ *\ 11 \mid \sigma\ \rangle\end{aligned}$$

Équivalence entre les sémantiques

Lemme
Les expressions se réduisent vers leur valeur :

- *Étant donnée une expression arithmétique* a*, on a*

$$\langle\ a \mid \sigma\ \rangle \xrightarrow{*} \langle\ \underline{[\![a]\!]_\sigma^{\mathsf{Aexp}}} \mid \sigma\ \rangle$$

- *Étant donnée une expression booléenne* b*, on a*

$$\langle\ b \mid \sigma\ \rangle \xrightarrow{*} \langle\ \underline{[\![b]\!]_\sigma^{\mathsf{Bexp}}} \mid \sigma\ \rangle$$

avec $\underline{\top} = \mathtt{true}$ *et* $\underline{\bot} = \mathtt{false}$*.*

Par exemple,
$$\begin{aligned}\langle\ (3\ +\ 1)\ *\ (x\ +\ 5) \mid \sigma\ \rangle &\longrightarrow \langle\ 4\ *\ (x\ +\ 5) \mid \sigma\ \rangle \\ &\longrightarrow \langle\ 4\ *\ (6\ +\ 5) \mid \sigma\ \rangle \\ &\longrightarrow \langle\ 4\ *\ 11 \mid \sigma\ \rangle \\ &\longrightarrow \langle\ 44 \mid \sigma\ \rangle\end{aligned}$$

Équivalence entre les sémantiques

Lemme
Les expressions se réduisent vers leur valeur :

- *Étant donnée une expression arithmétique a, on a*

$$\langle\ a \mid \sigma\ \rangle \longrightarrow^{*} \langle\ \underline{[\![a]\!]_\sigma^{\mathrm{Aexp}}} \mid \sigma\ \rangle$$

- *Étant donnée une expression booléenne b, on a*

$$\langle\ b \mid \sigma\ \rangle \longrightarrow^{*} \langle\ \underline{[\![b]\!]_\sigma^{\mathrm{Bexp}}} \mid \sigma\ \rangle$$

avec $\underline{\top} = \mathtt{true}$ et $\underline{\bot} = \mathtt{false}$.

Démonstration.
Par induction sur a (ou b). □

Équivalence entre les sémantiques

Lemme
La réduction préserve les valeurs :

- *si* $\langle\ a \mid \sigma\ \rangle \longrightarrow \langle\ a' \mid \sigma'\ \rangle$ *alors* $\sigma = \sigma'$ *et* $[\![a]\!]_{\sigma}^{\mathrm{Aexp}} = [\![a']\!]_{\sigma'}^{\mathrm{Aexp}}$,
- *si* $\langle\ b \mid \sigma\ \rangle \longrightarrow \langle\ b' \mid \sigma'\ \rangle$ *alors* $\sigma = \sigma'$ *et* $[\![b]\!]_{\sigma}^{\mathrm{Bexp}} = [\![b']\!]_{\sigma'}^{\mathrm{Bexp}}$.

Équivalence entre les sémantiques

Lemme
La réduction préserve les valeurs :

- *si* $\langle\ a \mid \sigma\ \rangle \longrightarrow \langle\ a' \mid \sigma'\ \rangle$ *alors* $\sigma = \sigma'$ *et* $[\![a]\!]_{\sigma}^{\mathrm{Aexp}} = [\![a']\!]_{\sigma'}^{\mathrm{Aexp}}$,
- *si* $\langle\ b \mid \sigma\ \rangle \longrightarrow \langle\ b' \mid \sigma'\ \rangle$ *alors* $\sigma = \sigma'$ *et* $[\![b]\!]_{\sigma}^{\mathrm{Bexp}} = [\![b']\!]_{\sigma'}^{\mathrm{Bexp}}$.

Démonstration.
On a

$$\langle\ a \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\!\longrightarrow \langle\ a' \mid \sigma'\ \rangle$$

Équivalence entre les sémantiques

Lemme
La réduction préserve les valeurs :

- *si* $\langle\ a \mid \sigma\ \rangle \longrightarrow \langle\ a' \mid \sigma'\ \rangle$ *alors* $\sigma = \sigma'$ *et* $[\![a]\!]^{\mathrm{Aexp}}_{\sigma} = [\![a']\!]^{\mathrm{Aexp}}_{\sigma'}$,
- *si* $\langle\ b \mid \sigma\ \rangle \longrightarrow \langle\ b' \mid \sigma'\ \rangle$ *alors* $\sigma = \sigma'$ *et* $[\![b]\!]^{\mathrm{Bexp}}_{\sigma} = [\![b']\!]^{\mathrm{Bexp}}_{\sigma'}$.

Démonstration.
On a

$$\begin{array}{ccc} \langle\ a \mid \sigma\ \rangle & \longrightarrow & \langle\ a' \mid \sigma'\ \rangle \\ \Big\downarrow{\scriptstyle *} & & \Big\downarrow{\scriptstyle *} \\ \langle\ \underline{[\![a]\!]^{\mathrm{Aexp}}_{\sigma}} \mid \sigma\ \rangle & & \langle\ \underline{[\![a']\!]^{\mathrm{Aexp}}_{\sigma'}} \mid \sigma'\ \rangle \end{array}$$

Équivalence entre les sémantiques

Lemme
La réduction préserve les valeurs :

- *si* $\langle\ a \mid \sigma\ \rangle \longrightarrow \langle\ a' \mid \sigma'\ \rangle$ *alors* $\sigma = \sigma'$ *et* $[\![a]\!]^{\mathrm{Aexp}}_{\sigma} = [\![a']\!]^{\mathrm{Aexp}}_{\sigma'}$,
- *si* $\langle\ b \mid \sigma\ \rangle \longrightarrow \langle\ b' \mid \sigma'\ \rangle$ *alors* $\sigma = \sigma'$ *et* $[\![b]\!]^{\mathrm{Bexp}}_{\sigma} = [\![b']\!]^{\mathrm{Bexp}}_{\sigma'}$.

Démonstration.
On a

$$\begin{array}{ccc} \langle\ a \mid \sigma\ \rangle & \longrightarrow & \langle\ a' \mid \sigma'\ \rangle \\ {\scriptstyle *}\downarrow & & \downarrow{\scriptstyle *} \\ \langle\ \underline{[\![a]\!]^{\mathrm{Aexp}}_{\sigma}} \mid \sigma\ \rangle & \underset{*}{\longleftrightarrow} & \langle\ \underline{[\![a']\!]^{\mathrm{Aexp}}_{\sigma'}} \mid \sigma'\ \rangle \end{array}$$

par déterminisme

Équivalence entre les sémantiques

Lemme
La réduction préserve les valeurs :

- *si* $\langle\ a \mid \sigma\ \rangle \longrightarrow \langle\ a' \mid \sigma'\ \rangle$ *alors* $\sigma = \sigma'$ *et* $[\![a]\!]^{\mathrm{Aexp}}_{\sigma} = [\![a']\!]^{\mathrm{Aexp}}_{\sigma'}$,
- *si* $\langle\ b \mid \sigma\ \rangle \longrightarrow \langle\ b' \mid \sigma'\ \rangle$ *alors* $\sigma = \sigma'$ *et* $[\![b]\!]^{\mathrm{Bexp}}_{\sigma} = [\![b']\!]^{\mathrm{Bexp}}_{\sigma'}$.

Démonstration.
On a

$$\begin{array}{ccc} \langle\ a \mid \sigma\ \rangle & \longrightarrow & \langle\ a' \mid \sigma'\ \rangle \\ {\scriptstyle *}\downarrow & & \downarrow{\scriptstyle *} \\ \langle\ \underline{[\![a]\!]^{\mathrm{Aexp}}_{\sigma}} \mid \sigma\ \rangle & = & \langle\ \underline{[\![a']\!]^{\mathrm{Aexp}}_{\sigma'}} \mid \sigma'\ \rangle \end{array}$$

car les entiers sont irréductibles.

□

Équivalence entre les sémantiques

Lemme
La réduction préserve les valeurs :

- *si* $\langle\ a \mid \sigma\ \rangle \longrightarrow \langle\ a' \mid \sigma'\ \rangle$ *alors* $\sigma = \sigma'$ *et* $[\![a]\!]^{\mathrm{Aexp}}_{\sigma} = [\![a']\!]^{\mathrm{Aexp}}_{\sigma'}$,
- *si* $\langle\ b \mid \sigma\ \rangle \longrightarrow \langle\ b' \mid \sigma'\ \rangle$ *alors* $\sigma = \sigma'$ *et* $[\![b]\!]^{\mathrm{Bexp}}_{\sigma} = [\![b']\!]^{\mathrm{Bexp}}_{\sigma'}$.

Démonstration.
On a

$$\begin{array}{ccc} \langle\ a \mid \sigma\ \rangle & \longrightarrow & \langle\ a' \mid \sigma'\ \rangle \\ {\scriptstyle *}\downarrow & & \downarrow{\scriptstyle *} \\ \langle\ \underline{[\![a]\!]^{\mathrm{Aexp}}_{\sigma}} \mid \sigma\ \rangle & = & \langle\ \underline{[\![a']\!]^{\mathrm{Aexp}}_{\sigma'}} \mid \sigma'\ \rangle \end{array}$$

car les entiers sont irréductibles. [mais on pourrait faire une preuve qui fonctionne dans un cas non-déterministe] □

Équivalence entre les sémantiques

Lemme
Les suites de réductions sont compatibles avec le contexte :

$$\langle\, a \mid \sigma \,\rangle \xrightarrow{*} \langle\, a' \mid \sigma' \,\rangle \quad \textit{implique} \quad \langle\, x \texttt{ := } a \mid \sigma \,\rangle \xrightarrow{*} \langle\, x \texttt{ := } a' \mid \sigma' \,\rangle$$

$$\langle\, c_1 \mid \sigma \,\rangle \xrightarrow{*} \langle\, c_1' \mid \sigma' \,\rangle \quad \textit{implique} \quad \langle\, c_1 \texttt{ ; } c_2 \mid \sigma \,\rangle \xrightarrow{*} \langle\, c_1' \texttt{ ; } c_2 \mid \sigma' \,\rangle$$

$$\langle\, b \mid \sigma \,\rangle \xrightarrow{*} \langle\, b' \mid \sigma' \,\rangle \quad \textit{implique}$$

$$\langle\, \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2 \mid \sigma \,\rangle \xrightarrow{*} \langle\, \texttt{if } b' \texttt{ then } c_1 \texttt{ else } c_2 \mid \sigma' \,\rangle$$

Démonstration.
Par récurrence sur la longueur de la suite de réductions. □

Équivalence entre les sémantiques

On peut finalement montrer l'équivalence entre les sémantiques à grands et à petits pas.

Équivalence entre les sémantiques

Démonstration.
$\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma' \Rightarrow \langle\ c \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma'\ \rangle$ par induction sur la dérivation.

– Si la dernière règle est

$$\frac{}{\langle\ \texttt{skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma}\ (\textsc{skip})$$

on conclut immédiatement par

$$\langle\ \texttt{skip} \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma\ \rangle$$

en 0 pas de réduction.

Équivalence entre les sémantiques

Démonstration.
$\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma' \Rightarrow \langle\ c \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma'\ \rangle$ par induction sur la dérivation.

– Si la dernière règle est

$$\frac{}{\langle\ x\ \texttt{:=}\ a \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma[x \mapsto [\![a]\!]_\sigma^{\mathsf{Aexp}}]}\ (\textsc{set})$$

Équivalence entre les sémantiques

Démonstration.
$\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma' \Rightarrow \langle\ c \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma'\ \rangle$ par induction sur la dérivation.

– Si la dernière règle est

$$\frac{}{\langle\ x\ \texttt{:=}\ a \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma[x \mapsto [\![a]\!]_\sigma^{\mathsf{Aexp}}]}\ (\textsc{set})$$

alors on a

$$\langle\ a \mid \sigma\ \rangle \xrightarrow{*} \langle\ \underline{[\![a]\!]_\sigma^{\mathsf{Aexp}}} \mid \sigma\ \rangle$$

Équivalence entre les sémantiques

Démonstration.
$\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma' \Rightarrow \langle\ c \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma'\ \rangle$ par induction sur la dérivation.

– Si la dernière règle est

$$\frac{}{\langle\ x\ \texttt{:=}\ a \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma[x \mapsto [\![a]\!]_\sigma^{\mathsf{Aexp}}]}\ (\textsc{set})$$

alors on a

$$\langle\ a \mid \sigma\ \rangle \xrightarrow{*} \langle\ \underline{[\![a]\!]_\sigma^{\mathsf{Aexp}}} \mid \sigma\ \rangle$$

et donc

$$\langle\ x\ \texttt{:=}\ a \mid \sigma\ \rangle \xrightarrow{*} \langle\ x\ \texttt{:=}\ \underline{[\![a]\!]_\sigma^{\mathsf{Aexp}}} \mid \sigma\ \rangle$$

Équivalence entre les sémantiques

Démonstration.
$\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma' \Rightarrow \langle\ c \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma'\ \rangle$ par induction sur la dérivation.

– Si la dernière règle est

$$\frac{}{\langle\ x\ \texttt{:=}\ a \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma[x \mapsto [\![a]\!]^{\mathsf{Aexp}}_{\sigma}]}\ (\textsc{set})$$

alors on a

$$\langle\ a \mid \sigma\ \rangle \xrightarrow{*} \langle\ \underline{[\![a]\!]^{\mathsf{Aexp}}_{\sigma}} \mid \sigma\ \rangle$$

et donc

$$\langle\ x\ \texttt{:=}\ a \mid \sigma\ \rangle \xrightarrow{*} \langle\ x\ \texttt{:=}\ \underline{[\![a]\!]^{\mathsf{Aexp}}_{\sigma}} \mid \sigma\ \rangle$$

et on conclut avec la règle

$$\frac{}{\langle\ x\ \texttt{:=}\ \underline{n} \mid \sigma\ \rangle \longrightarrow \langle\ \texttt{skip} \mid \sigma[x \mapsto n]\ \rangle}\ (\textsc{set}_n)$$

Équivalence entre les sémantiques

Démonstration.
$\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma' \Rightarrow \langle\ c \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma'\ \rangle$ par induction sur la dérivation.

– Si la dernière règle est

$$\frac{\langle\ c_1 \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma' \qquad \langle\ c_2 \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma''}{\langle\ c_1\,;\ c_2 \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma''}\ (\textsc{seq})$$

Équivalence entre les sémantiques

Démonstration.
$\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma' \Rightarrow \langle\ c \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma'\ \rangle$ par induction sur la dérivation.

– Si la dernière règle est

$$\frac{\langle\ c_1 \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma' \qquad \langle\ c_2 \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma''}{\langle\ c_1 ;\ c_2 \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma''}\ (\textsc{seq})$$

Par hypothèse d'induction, on a

$$\langle\ c_1 \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma'\ \rangle \qquad\qquad \langle\ c_2 \mid \sigma'\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma''\ \rangle$$

Équivalence entre les sémantiques

Démonstration.
$\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma' \Rightarrow \langle\ c \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma'\ \rangle$ par induction sur la dérivation.

– Si la dernière règle est

$$\frac{\langle\ c_1 \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma' \qquad \langle\ c_2 \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}{\langle\ c_1\,;\ c_2 \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma''}\ (\textsc{seq})$$

Par hypothèse d'induction, on a

$$\langle\ c_1 \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma'\ \rangle \qquad\qquad \langle\ c_2 \mid \sigma'\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma''\ \rangle$$

D'où

$$\langle\ c_1\,;\ c_2 \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip};\ c_2 \mid \sigma'\ \rangle$$

Équivalence entre les sémantiques

Démonstration.
$\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma' \Rightarrow \langle\ c \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma'\ \rangle$ par induction sur la dérivation.

– Si la dernière règle est

$$\frac{\langle\ c_1 \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma' \qquad \langle\ c_2 \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma''}{\langle\ c_1\,;\ c_2 \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma''}\ (\textsc{seq})$$

Par hypothèse d'induction, on a

$$\langle\ c_1 \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma'\ \rangle \qquad\qquad \langle\ c_2 \mid \sigma'\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma''\ \rangle$$

D'où

$$\langle\ c_1\,;\ c_2 \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip};\ c_2 \mid \sigma'\ \rangle$$

et l'on conclut avec la suite de réductions

$$\langle\ c_1\,;\ c_2 \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip};\ c_2 \mid \sigma'\ \rangle \xrightarrow{(\textsc{seq}_\textsc{d})} \langle\ c_2 \mid \sigma'\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma''\ \rangle$$

Équivalence entre les sémantiques

Démonstration.
$\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma' \Rightarrow \langle\ c \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma'\ \rangle$ par induction sur la dérivation.

– Si la dernière règle est

$$\frac{\langle\ c_1 \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma'}{\langle\ \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma'}\ (\mathrm{IF}_\top)$$

où $[\![b]\!]^{\mathsf{Bexp}}_\sigma = \top$.

Équivalence entre les sémantiques

Démonstration.
$\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma' \Rightarrow \langle\ c \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma'\ \rangle$ par induction sur la dérivation.

– Si la dernière règle est

$$\frac{\langle\ c_1 \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma'}{\langle\ \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma'}\ (\mathrm{IF}_\top)$$

où $[\![b]\!]_\sigma^{\mathsf{Bexp}} = \top$. Par hypothèse d'induction, on a

$$\langle\ c_1 \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma'\ \rangle$$

Équivalence entre les sémantiques

Démonstration.
$\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma' \Rightarrow \langle\ c \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma'\ \rangle$ par induction sur la dérivation.

– Si la dernière règle est

$$\frac{\langle\ c_1 \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}{\langle\ \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}\ (\mathrm{IF}_\top)$$

où $[\![b]\!]_\sigma^{\mathrm{Bexp}} = \top$. Par hypothèse d'induction, on a

$$\langle\ c_1 \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma'\ \rangle$$

De plus, $\langle\ b \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{true} \mid \sigma\ \rangle$

Équivalence entre les sémantiques

Démonstration.
$\langle\, c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma' \Rightarrow \langle\, c \mid \sigma \,\rangle \xrightarrow{*} \langle\, \texttt{skip} \mid \sigma' \,\rangle$ par induction sur la dérivation.

– Si la dernière règle est

$$\frac{\langle\, c_1 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma'}{\langle\, \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma'}\ (\mathrm{IF}_\top)$$

où $[\![b]\!]_\sigma^{\mathrm{Bexp}} = \top$. Par hypothèse d'induction, on a

$$\langle\, c_1 \mid \sigma \,\rangle \xrightarrow{*} \langle\, \texttt{skip} \mid \sigma' \,\rangle$$

De plus, $\langle\, b \mid \sigma \,\rangle \xrightarrow{*} \langle\, \texttt{true} \mid \sigma \,\rangle$
d'où $\langle\, \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle \xrightarrow{*} \langle\, \texttt{if true then}\ c_1\ \texttt{else}\ c_2 \mid \sigma \,\rangle$

Équivalence entre les sémantiques

Démonstration.
$\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma' \Rightarrow \langle\ c \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma'\ \rangle$ par induction sur la dérivation.

– Si la dernière règle est

$$\frac{\langle\ c_1 \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}{\langle\ \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma'}\ (\mathrm{IF}_\top)$$

où $[\![b]\!]_\sigma^{\mathsf{Bexp}} = \top$. Par hypothèse d'induction, on a

$$\langle\ c_1 \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma'\ \rangle$$

De plus, $\langle\ b \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{true} \mid \sigma\ \rangle$
d'où $\langle\ \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{if true then}\ c_1\ \texttt{else}\ c_2 \mid \sigma\ \rangle$
On a donc la suite de réductions

$$\langle\ \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{if true then}\ c_1\ \texttt{else}\ c_2 \mid \sigma\ \rangle$$
$$\xrightarrow{(\mathrm{SEQ_G})} \langle\ c_1 \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma'\ \rangle$$

Équivalence entre les sémantiques

Démonstration.
$\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\!\rightarrow \sigma' \Rightarrow \langle\ c \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma'\ \rangle$ par induction sur la dérivation.

– Les autres cas sont laissés au lecteur. □

Équivalence entre les sémantiques

Démonstration.
Réciproquement, $\langle\ c \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma_\infty\ \rangle \Rightarrow \langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty$ par récurrence sur la longueur de $\xrightarrow{*}$.

– Si la longueur est 0, on a $c = \texttt{skip}$ et $\sigma_\infty = \sigma$ et on conclut par $(\textsc{skip})$:

$$\langle\ \texttt{skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma$$

Équivalence entre les sémantiques

Démonstration.
Réciproquement, $\langle\ c \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma_\infty\ \rangle \Rightarrow \langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty$ par récurrence sur la longueur de $\xrightarrow{*}$.

- Si la longueur est 0, on a $c = \texttt{skip}$ et $\sigma_\infty = \sigma$ et on conclut par $(\textsc{skip})$:

$$\langle\ \texttt{skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma$$

- Sinon, on a

$$\langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c' \mid \sigma'\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma_\infty\ \rangle$$

Équivalence entre les sémantiques

Démonstration.
Réciproquement, $\langle\ c \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma_\infty\ \rangle \Rightarrow \langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty$ par récurrence sur la longueur de $\xrightarrow{*}$.

- Si la longueur est 0, on a $c = \texttt{skip}$ et $\sigma_\infty = \sigma$ et on conclut par $(\textsc{skip})$:

$$\langle\ \texttt{skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma$$

- Sinon, on a

$$\langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c' \mid \sigma'\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma_\infty\ \rangle$$

avec $\langle\ c' \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty$ par hypothèse de récurrence

Équivalence entre les sémantiques

Démonstration.
Réciproquement, $\langle\ c \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma_\infty\ \rangle \Rightarrow \langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty$ par récurrence sur la longueur de $\xrightarrow{*}$.

- Si la longueur est 0, on a $c = \texttt{skip}$ et $\sigma_\infty = \sigma$ et on conclut par $(\textsc{skip})$:

$$\langle\ \texttt{skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma$$

- Sinon, on a

$$\langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c' \mid \sigma'\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma_\infty\ \rangle$$

avec $\langle\ c' \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty$ par hypothèse de récurrence et il faut montrer

$$\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty$$

Équivalence entre les sémantiques

Lemme
Si $\langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c' \mid \sigma'\ \rangle$ *et* $\langle\ c' \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty$ *alors* $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty$.

Démonstration.
Par induction sur la dérivation de $\langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c' \mid \sigma'\ \rangle$.

Équivalence entre les sémantiques

Lemme
Si $\langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c' \mid \sigma'\ \rangle$ *et* $\langle\ c' \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty$ *alors* $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty$.

Démonstration.
– Si c'est

$$\frac{\langle\ a \mid \sigma\ \rangle \longrightarrow \langle\ a' \mid \sigma'\ \rangle}{\langle\ x\ :=\ a \mid \sigma\ \rangle \longrightarrow \langle\ x\ :=\ a' \mid \sigma'\ \rangle}\ (\textsc{set})$$

Équivalence entre les sémantiques

Lemme
Si $\langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c' \mid \sigma'\ \rangle$ *et* $\langle\ c' \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty$ *alors* $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty$.

Démonstration.
– Si c'est

$$\frac{\langle\ a \mid \sigma\ \rangle \longrightarrow \langle\ a' \mid \sigma'\ \rangle}{\langle\ x\ :=\ a \mid \sigma\ \rangle \longrightarrow \langle\ x\ :=\ a' \mid \sigma'\ \rangle}\ (\textsc{set})$$

alors on a

$$\langle\ x\ :=\ a \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma[x \mapsto [\![a]\!]_\sigma^{\mathsf{Aexp}}] \qquad \langle\ x\ :=\ a' \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma'[x \mapsto [\![a']\!]_{\sigma'}^{\mathsf{Aexp}}]$$

Équivalence entre les sémantiques

Lemme
Si $\langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c' \mid \sigma'\ \rangle$ *et* $\langle\ c' \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty$ *alors* $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty$.

Démonstration.
– Si c'est

$$\frac{\langle\ a \mid \sigma\ \rangle \longrightarrow \langle\ a' \mid \sigma'\ \rangle}{\langle\ x\ :=\ a \mid \sigma\ \rangle \longrightarrow \langle\ x\ :=\ a' \mid \sigma'\ \rangle}\ (\textsc{set})$$

alors on a

$$\langle\ x\ :=\ a \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma[x \mapsto [\![a]\!]_\sigma^{\mathsf{Aexp}}] \qquad \langle\ x\ :=\ a' \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma'[x \mapsto [\![a']\!]_{\sigma'}^{\mathsf{Aexp}}]$$

avec $\sigma = \sigma'$ et $[\![a]\!]_\sigma^{\mathsf{Aexp}} = [\![a']\!]_{\sigma'}^{\mathsf{Aexp}}$ (lemme sur les expressions arithmétiques).

Équivalence entre les sémantiques

Lemme
Si $\langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c' \mid \sigma'\ \rangle$ *et* $\langle\ c' \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty$ *alors* $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty$.

Démonstration.
– Si c'est

$$\frac{}{\langle\ x\ \texttt{:=}\ \underline{n} \mid \sigma\ \rangle \longrightarrow \langle\ \texttt{skip} \mid \sigma[x \mapsto n]\ \rangle}\ (\text{SET}_n)$$

Équivalence entre les sémantiques

Lemme
Si $\langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c' \mid \sigma'\ \rangle$ *et* $\langle\ c' \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty$ *alors* $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty$.

Démonstration.
– Si c'est

$$\frac{}{\langle\ x\ :=\ \underline{n} \mid \sigma\ \rangle \longrightarrow \langle\ \mathtt{skip} \mid \sigma[x \mapsto n]\ \rangle}\ (\mathrm{SET}_n)$$

c'est immédiat par

$$\frac{}{\langle\ x\ :=\ \underline{n} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma[x \mapsto [\![\underline{n}]\!]^{\mathsf{Aexp}}_\sigma]}\ (\mathrm{SET})$$

Équivalence entre les sémantiques

Lemme
Si $\langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c' \mid \sigma'\ \rangle$ *et* $\langle\ c' \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty$ *alors* $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty$.

Démonstration.
– Si c'est

$$\frac{\langle\ c_1 \mid \sigma\ \rangle \longrightarrow \langle\ c_1' \mid \sigma'\ \rangle}{\langle\ c_1\,;\ c_2 \mid \sigma\ \rangle \longrightarrow \langle\ c_1';\ c_2 \mid \sigma'\ \rangle}\ (\mathrm{SEQ_G})$$

Équivalence entre les sémantiques

Lemme
Si $\langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c' \mid \sigma'\ \rangle$ *et* $\langle\ c' \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty$ *alors* $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty$.

Démonstration.
– Si c'est

$$\frac{\langle\ c_1 \mid \sigma\ \rangle \longrightarrow \langle\ c_1' \mid \sigma'\ \rangle}{\langle\ c_1;\ c_2 \mid \sigma\ \rangle \longrightarrow \langle\ c_1';\ c_2 \mid \sigma'\ \rangle}\ (\mathrm{SEQ_G})$$

L'évaluation $\langle\ c_1';\ c_2 \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty$ est de la forme

$$\frac{\langle\ c_1' \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma'' \qquad \langle\ c_2 \mid \sigma''\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty}{\langle\ c_1';\ c_2 \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty}\ (\mathrm{SEQ})$$

Équivalence entre les sémantiques

Lemme
Si $\langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c' \mid \sigma'\ \rangle$ *et* $\langle\ c' \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty$ *alors* $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty$.

Démonstration.
– Si c'est

$$\frac{\langle\ c_1 \mid \sigma\ \rangle \longrightarrow \langle\ c_1' \mid \sigma'\ \rangle}{\langle\ c_1;\ c_2 \mid \sigma\ \rangle \longrightarrow \langle\ c_1';\ c_2 \mid \sigma'\ \rangle}\ (\mathrm{SEQ_G})$$

L'évaluation $\langle\ c_1';\ c_2 \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty$ est de la forme

$$\frac{\langle\ c_1' \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma'' \qquad \langle\ c_2 \mid \sigma''\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty}{\langle\ c_1';\ c_2 \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty}\ (\mathrm{SEQ})$$

L'hypothèse d'induction sur $\langle\ c_1' \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma''$ donne $\langle\ c_1 \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma''$

Équivalence entre les sémantiques

Lemme
Si $\langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c' \mid \sigma'\ \rangle$ *et* $\langle\ c' \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty$ *alors* $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty$.

Démonstration.
– Si c'est

$$\frac{\langle\ c_1 \mid \sigma\ \rangle \longrightarrow \langle\ c_1' \mid \sigma'\ \rangle}{\langle\ c_1;\ c_2 \mid \sigma\ \rangle \longrightarrow \langle\ c_1';\ c_2 \mid \sigma'\ \rangle}\ (\mathrm{SEQ_G})$$

L'évaluation $\langle\ c_1';\ c_2 \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty$ est de la forme

$$\frac{\langle\ c_1' \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma'' \qquad \langle\ c_2 \mid \sigma''\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty}{\langle\ c_1';\ c_2 \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty}\ (\mathrm{SEQ})$$

L'hypothèse d'induction sur $\langle\ c_1' \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma''$ donne $\langle\ c_1 \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma''$ d'où

$$\frac{\langle\ c_1 \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma'' \qquad \langle\ c_2 \mid \sigma''\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty}{\langle\ c_1;\ c_2 \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty}\ (\mathrm{SEQ})$$

Équivalence entre les sémantiques

Lemme
Si $\langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c' \mid \sigma'\ \rangle$ *et* $\langle\ c' \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty$ *alors* $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty$.

Démonstration.
– Si c'est

$$\frac{}{\langle\ \texttt{skip;}\ c \mid \sigma\ \rangle \longrightarrow \langle\ c \mid \sigma\ \rangle}\ (\mathrm{SEQ_D})$$

Équivalence entre les sémantiques

Lemme
Si $\langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c' \mid \sigma'\ \rangle$ *et* $\langle\ c' \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty$ *alors* $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty$.

Démonstration.
– Si c'est

$$\frac{}{\langle\ \texttt{skip;}\ c \mid \sigma\ \rangle \longrightarrow \langle\ c \mid \sigma\ \rangle}\ (\mathrm{SEQ_D})$$

alors on a immédiatement

$$\frac{\langle\ \texttt{skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma \qquad \langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty}{\langle\ \texttt{skip;}\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty}\ (\mathrm{SEQ})$$

Équivalence entre les sémantiques

Lemme
Si $\langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c' \mid \sigma'\ \rangle$ *et* $\langle\ c' \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty$ *alors* $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty$.

Démonstration.
– Cas $(\textsc{if})$, $(\textsc{if}_\top)$ et $(\textsc{if}_\bot)$: laissés au lecteur.

Équivalence entre les sémantiques

Lemme
Si $\langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c' \mid \sigma'\ \rangle$ *et* $\langle\ c' \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty$ *alors* $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty$.

Démonstration.

– Si c'est

$$\frac{}{\langle\ \texttt{while}\ b\ \texttt{do}\ c \mid \sigma\ \rangle \longrightarrow \langle\ \texttt{if}\ b\ \texttt{then}\ (c\texttt{;}\ \texttt{while}\ b\ \texttt{do}\ c)\ \texttt{else skip} \mid \sigma\ \rangle}\ (\textsc{while})$$

Équivalence entre les sémantiques

Lemme
Si $\langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c' \mid \sigma'\ \rangle$ *et* $\langle\ c' \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty$ *alors* $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty$.

Démonstration.

– Si c'est

$$\frac{}{\langle\ \texttt{while}\ b\ \texttt{do}\ c \mid \sigma\ \rangle \longrightarrow \langle\ \texttt{if}\ b\ \texttt{then}\ (c;\ \texttt{while}\ b\ \texttt{do}\ c)\ \texttt{else skip} \mid \sigma\ \rangle}\ (\textsc{while})$$

Si $[\![b]\!]^{\mathsf{Bexp}}_{\sigma} = \top$.

Équivalence entre les sémantiques

Lemme
Si $\langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c' \mid \sigma'\ \rangle$ *et* $\langle\ c' \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty$ *alors* $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty$.

Démonstration.

– Si c'est

$$\frac{}{\langle\ \texttt{while}\ b\ \texttt{do}\ c \mid \sigma\ \rangle \longrightarrow \langle\ \texttt{if}\ b\ \texttt{then}\ (c;\ \texttt{while}\ b\ \texttt{do}\ c)\ \texttt{else skip} \mid \sigma\ \rangle}\ (\textsc{while})$$

Si $[\![b]\!]^{\mathsf{Bexp}}_{\sigma} = \top$. L'évaluation de la droite est de a forme

$$\frac{\dfrac{\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma' \qquad \langle\ \texttt{while}\ b\ \texttt{do}\ c \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty}{\langle\ c;\ \texttt{while}\ b\ \texttt{do}\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty}\ (\textsc{seq})}{\langle\ \texttt{if}\ b\ \texttt{then}\ (c;\ \texttt{while}\ b\ \texttt{do}\ c)\ \texttt{else skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty}\ (\textsc{if}_\top)$$

Équivalence entre les sémantiques

Lemme
Si $\langle\, c \mid \sigma \,\rangle \longrightarrow \langle\, c' \mid \sigma' \,\rangle$ *et* $\langle\, c' \mid \sigma' \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty$ *alors* $\langle\, c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty$.

Démonstration.

– Si c'est

$$\frac{}{\langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma \,\rangle \longrightarrow \langle\, \texttt{if}\ b\ \texttt{then}\ (c;\ \texttt{while}\ b\ \texttt{do}\ c)\ \texttt{else skip} \mid \sigma \,\rangle}\ (\textsc{while})$$

Si $[\![b]\!]^{\mathsf{Bexp}}_{\sigma} = \top$. L'évaluation de la droite est de a forme

$$\frac{\dfrac{\langle\, c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma' \qquad \langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma' \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty}{\langle\, c;\ \texttt{while}\ b\ \texttt{do}\ c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty}\ (\textsc{seq})}{\langle\, \texttt{if}\ b\ \texttt{then}\ (c;\ \texttt{while}\ b\ \texttt{do}\ c)\ \texttt{else skip} \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty}\ (\textsc{if}_\top)$$

et on conclut par

$$\frac{\langle\, c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma' \qquad \langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma' \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty}{\langle\, \texttt{while}\ b\ \texttt{do}\ c \mid \sigma \,\rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty}\ (\textsc{while}_\top)$$

Équivalence entre les sémantiques

Lemme
Si $\langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c' \mid \sigma'\ \rangle$ *et* $\langle\ c' \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty$ *alors* $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma_\infty$.

Démonstration.

– Si c'est

$$\frac{}{\langle\ \texttt{while}\ b\ \texttt{do}\ c \mid \sigma\ \rangle \longrightarrow \langle\ \texttt{if}\ b\ \texttt{then}\ (c;\ \texttt{while}\ b\ \texttt{do}\ c)\ \texttt{else skip} \mid \sigma\ \rangle}\ (\textsc{while})$$

Si $[\![b]\!]^{\mathsf{Bexp}}_\sigma = \bot$.

Équivalence entre les sémantiques

Lemme
Si $\langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c' \mid \sigma'\ \rangle$ *et* $\langle\ c' \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty$ *alors* $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty$.

Démonstration.

– Si c'est

$$\frac{}{\langle\ \texttt{while}\ b\ \texttt{do}\ c \mid \sigma\ \rangle \longrightarrow \langle\ \texttt{if}\ b\ \texttt{then}\ (c;\ \texttt{while}\ b\ \texttt{do}\ c)\ \texttt{else skip} \mid \sigma\ \rangle}\ (\textsc{while})$$

Si $[\![b]\!]_\sigma^{\mathsf{Bexp}} = \bot$. L'évaluation de la droite est de la forme

$$\frac{\langle\ \texttt{skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma}{\langle\ \texttt{if}\ b\ \texttt{then}\ (c;\ \texttt{while}\ b\ \texttt{do}\ c)\ \texttt{else skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma}\ (\textsc{if}_\bot)$$

Équivalence entre les sémantiques

Lemme
Si $\langle\ c \mid \sigma\ \rangle \longrightarrow \langle\ c' \mid \sigma'\ \rangle$ *et* $\langle\ c' \mid \sigma'\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty$ *alors* $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma_\infty$.

Démonstration.

– Si c'est

$$\frac{}{\langle\ \texttt{while}\ b\ \texttt{do}\ c \mid \sigma\ \rangle \longrightarrow \langle\ \texttt{if}\ b\ \texttt{then}\ (c;\ \texttt{while}\ b\ \texttt{do}\ c)\ \texttt{else skip} \mid \sigma\ \rangle}\ (\textsc{while})$$

Si $[\![b]\!]_\sigma^{\mathsf{Bexp}} = \bot$. L'évaluation de la droite est de la forme

$$\frac{\langle\ \texttt{skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma}{\langle\ \texttt{if}\ b\ \texttt{then}\ (c;\ \texttt{while}\ b\ \texttt{do}\ c)\ \texttt{else skip} \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma}\ (\textsc{if}_\bot)$$

et on conclut par

$$\frac{}{\langle\ \texttt{while}\ b\ \texttt{do}\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow\ \sigma}\ (\textsc{while}_\bot)$$

Équivalence entre les sémantiques

On a ainsi défini les sémantiques à grands pas (évaluation) et à petits pas (réduction) et montré :

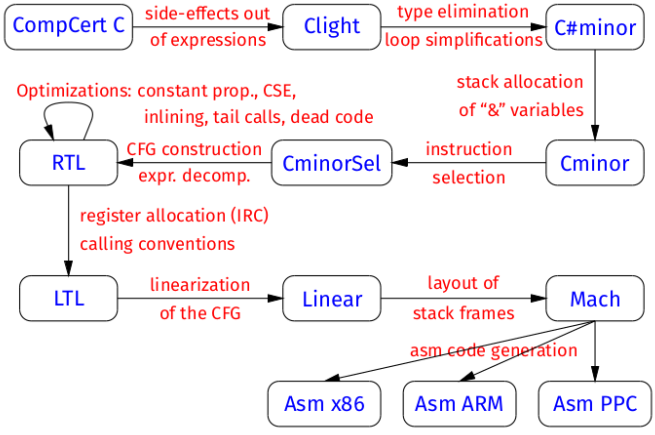
Théorème
On a équivalence entre

(i) $\langle\ c \mid \sigma\ \rangle \longrightarrow\!\!\!\!\rightarrow \sigma'$ *(dans la sémantique à grands pas),*

(ii) $\langle\ c \mid \sigma\ \rangle \xrightarrow{*} \langle\ \texttt{skip} \mid \sigma'\ \rangle$ *(dans la sémantique à petits pas).*

En pratique

Tout ceci est utile en pratique, par exemple pour certifier des compilateur :



(CompCert)

Le cours et les TD sont disponibles sur

https://agreg.mimram.fr/