

Différentes sémantiques d'un langage impératif

Pierre Lescanne

dernière mise à jour: *7 février 2006 – 16: 10*

Plan

Présentation du langage

La sémantique opérationnelle

Une sémantique «petite étapes» des expressions

Points fixes, fonctionnelles et récursivité

Une sémantique dénotationnelle de IMP

Une sémantique axiomatique de IMP

- Les deux premières instructions

- Le corps de la boucle

- Les boucles

- Les affectations

Correction de la sémantique axiomatique

Complétude de la sémantique axiomatique

Les constituants d'IMP

IMP est associé aux ensembles suivants :

- ▶ les **entiers** \mathbb{N} repérés par n, m ,
- ▶ les **emplacements** (en anglais «locations») **Loc**, repérés par X, Y ,
- ▶ les **expressions arithmétiques**, **Aexp** repérées par a, a_i ,
- ▶ les **expressions booléennes**, **Bexp** repérées par b, b_i ,
- ▶ les **commandes** (ou **instructions**), **Com** repérées par c, c_j .

La syntaxe d'IMP

Les **expressions arithmétiques** **Aexp**

$$a ::= n \mid X \mid a_0 + a_1 \mid a_0 - a_1 \mid a_0 \times a_1$$

Les **expressions booléennes** **Bexp**

$$b ::= \mathbf{true} \mid \mathbf{false} \mid a_0 = a_1 \mid a_0 \leq a_1 \mid \neg b \mid b_0 \vee b_1 \mid b_0 \wedge b_1$$

Les **Commandes** **Com**

$$c ::= \mathbf{skip} \mid X := a \mid c_0; c_1 \mid \mathbf{if } b \mathbf{ then } c_0 \mathbf{ else } c_1 \mid \mathbf{while } b \mathbf{ do } c$$

Plan

Présentation du langage

La sémantique opérationnelle

Une sémantique «petite étapes» des expressions

Points fixes, fonctionnelles et récursivité

Une sémantique dénotationnelle de IMP

Une sémantique axiomatique de IMP

- Les deux premières instructions

- Le corps de la boucle

- Les boucles

- Les affectations

Correction de la sémantique axiomatique

Complétude de la sémantique axiomatique

L'ensemble des états

L'ensemble $\Sigma : \mathbf{Loc} \rightarrow \mathbb{N}$ des **états** associe un nombre à un emplacement.

Si $\sigma : \Sigma$ alors $\sigma(X)$ est le **contenu** de l'emplacement **Loc** dans l'état σ .

Une expression arithmétique a qui attend d'être évaluée dans l'état σ se note $\langle a, \sigma \rangle$

La **relation d'évaluation** entre paires associe un nombre à une paire. Elle est notée $\langle a, \sigma \rangle \rightarrow n$.

L'ensemble des états

L'ensemble $\Sigma : \mathbf{Loc} \rightarrow \mathbb{N}$ des **états** associe un nombre à un emplacement.

Si $\sigma : \Sigma$ alors $\sigma(X)$ est le **contenu** de l'emplacement **Loc** dans l'état σ .

Une expression arithmétique a qui attend d'être évaluée dans l'état σ se note $\langle a, \sigma \rangle$

La **relation d'évaluation** entre paires associe un nombre à une paire. Elle est notée $\langle a, \sigma \rangle \rightarrow n$.

C'est une sémantique «grande étape»..

Évaluation des expressions entières

Évaluation d'un nombre

$$\langle n, \sigma \rangle \rightarrow n$$

Évaluation d'un emplacement

$$\langle X, \sigma \rangle \rightarrow \sigma(X)$$

Évaluation d'une somme

$$\frac{\langle a_1, \sigma \rangle \rightarrow n_1 \quad \langle a_2, \sigma \rangle \rightarrow n_2}{\langle a_1 + a_2, \sigma \rangle \rightarrow n}$$

n est la somme de n_1 et de n_2

Évaluation des expressions entières

Évaluation d'un nombre

$$\langle n, \sigma \rangle \rightarrow n$$

Évaluation d'un emplacement

$$\langle X, \sigma \rangle \rightarrow \sigma(X)$$

Évaluation d'une somme

$$\frac{\langle a_1, \sigma \rangle \rightarrow n_1 \quad \langle a_2, \sigma \rangle \rightarrow n_2}{\langle a_1 + a_2, \sigma \rangle \rightarrow n}$$

n est la somme de n_1 et de n_2

autrement dit $n = n_1 +_{\mathbb{N}} n_2$.

Évaluation des expressions entières

Évaluation d'une soustraction

$$\frac{\langle a_1, \sigma \rangle \rightarrow n_1 \quad \langle a_2, \sigma \rangle \rightarrow n_2}{\langle a_1 - a_2, \sigma \rangle \rightarrow n} \quad n \text{ est la soustraction de } n_1 \text{ et de } n_2$$

Évaluation d'un produit

$$\frac{\langle a_1, \sigma \rangle \rightarrow n_1 \quad \langle a_2, \sigma \rangle \rightarrow n_2}{\langle a_1 \times a_2, \sigma \rangle \rightarrow n} \quad n \text{ est le produit de } n_1 \text{ et de } n_2$$

Évaluation des expressions booléennes

Évaluation d'une valeur booléenne

$$\langle \mathbf{true}, \sigma \rangle \rightarrow \mathbf{true}$$

⋮

Évaluation d'un test d'inégalité

$$\frac{\langle a_1, \sigma \rangle \rightarrow n_1 \quad \langle a_2, \sigma \rangle \rightarrow n_2}{\langle a_1 \leq a_2, \sigma \rangle \rightarrow t}$$

où t est **true** si n_1 est plus petit ou égal à n_2 et t est **false** autrement.

Évaluation des expressions booléennes

Évaluation d'une disjonction

$$\frac{\langle b_1, \sigma \rangle \rightarrow t_1 \quad \langle b_2, \sigma \rangle \rightarrow t_2}{\langle b_1 \wedge b_2, \sigma \rangle \rightarrow t}$$

où t est **true** si $t_1 \equiv \mathbf{true}$ et $t_2 \equiv \mathbf{true}$ et t est **false**
autrement dit $t = t_1 \wedge_{\mathbb{B}} t_2$

Évaluation des instructions

On suppose qu'au début de l'évaluation d'un programme, on se trouve dans l'état initial σ_0 qui tel que

$$(\forall X \in \mathbf{Loc}) \sigma_0(X) = 0$$

Comme on sait, l'évaluation d'une commande à partir d'un certain état peut se terminer ou diverger.

Évaluation des instructions

Nous allons définir une relation

$$\langle c, \sigma \rangle \rightarrow \sigma'$$

qui signifie que

1. dans l'état σ , l'instruction c se termine et
2. le couple $\langle c, \sigma \rangle$ rejoint (s'évalue en) l'état σ' .

Évaluation des instructions

Par exemple, on s'imagine que

$$\langle X := X + 1, \sigma \rangle \rightarrow \sigma'$$

signifiera que σ' est l'état

- ▶ tel que $\sigma'(Y)$ est $\sigma(Y)$ pour $Y \neq X$
- ▶ et tel que $\sigma'(X)$ vaut $\sigma(X)$ augmenté de 1.

La notation $\sigma[m/X]$

Si $\sigma \in \Sigma$ est un état, $m \in \mathbb{N}$ est un naturel et $X \in \mathbf{Loc}$ est un emplacement, on écrit

$$\begin{aligned}\sigma[m/X](X) &= m \\ \sigma[m/X](Y) &= \sigma(Y) \quad \text{si } Y \neq X.\end{aligned}$$

Ainsi on pourra écrire

$$\langle X := X + 1, \sigma \rangle \rightarrow \sigma[\sigma(X) +_{\mathbb{N}} 1/X]$$

Les règles pour les instructions simples

Commande vide

$$\langle \mathbf{skip}, \sigma \rangle \rightarrow \sigma$$

Affectation

$$\frac{\langle a, \sigma \rangle \rightarrow m}{\langle X := a, \sigma \rangle \rightarrow \sigma[m/X]}$$

Les règles pour les instructions composées

Composition séquentielle

$$\frac{\langle c_0, \sigma \rangle \rightarrow \sigma'' \quad \langle c_1, \sigma'' \rangle \rightarrow \sigma'}{\langle c_0; c_1, \sigma \rangle \rightarrow \sigma'}$$

Composition conditionnelle

$$\frac{\langle b, \sigma \rangle \rightarrow \mathbf{true} \quad \langle c_0, \sigma \rangle \rightarrow \sigma'}{\langle \mathbf{if } b \mathbf{ then } c_0 \mathbf{ else } c_1, \sigma \rangle \rightarrow \sigma'}$$

$$\frac{\langle b, \sigma \rangle \rightarrow \mathbf{false} \quad \langle c_1, \sigma \rangle \rightarrow \sigma'}{\langle \mathbf{if } b \mathbf{ then } c_0 \mathbf{ else } c_1, \sigma \rangle \rightarrow \sigma'}$$

Les règles pour les instructions composées

Boucles while

$$\frac{\langle b, \sigma \rangle \rightarrow \mathbf{false}}{\langle \mathbf{while} \ b \ \mathbf{do} \ c, \sigma \rangle \rightarrow \sigma}$$
$$\frac{\langle b, \sigma \rangle \rightarrow \mathbf{true} \quad \langle c, \sigma \rangle \rightarrow \sigma'' \quad \langle \mathbf{while} \ b \ \mathbf{do} \ c, \sigma'' \rangle \rightarrow \sigma'}{\langle \mathbf{while} \ b \ \mathbf{do} \ c, \sigma \rangle \rightarrow \sigma'}$$

Les règles pour les instructions composées

Boucles while

$$\frac{\langle b, \sigma \rangle \rightarrow \mathbf{false}}{\langle \mathbf{while} \ b \ \mathbf{do} \ c, \sigma \rangle \rightarrow \sigma}$$
$$\frac{\langle b, \sigma \rangle \rightarrow \mathbf{true} \quad \langle c, \sigma \rangle \rightarrow \sigma'' \quad \langle \mathbf{while} \ b \ \mathbf{do} \ c, \sigma'' \rangle \rightarrow \sigma'}{\langle \mathbf{while} \ b \ \mathbf{do} \ c, \sigma \rangle \rightarrow \sigma'}$$

Noter que la définition ne se fait pas par induction sur la longueur de la commande.

Plan

Présentation du langage

La sémantique opérationnelle

Une sémantique «petite étapes» des expressions

Points fixes, fonctionnelles et récursivité

Une sémantique dénotationnelle de IMP

Une sémantique axiomatique de IMP

- Les deux premières instructions

- Le corps de la boucle

- Les boucles

- Les affectations

Correction de la sémantique axiomatique

Complétude de la sémantique axiomatique

Une alternative

On veut être plus proche de l'évaluation des expressions.
La relation se fait entre paires **expression-état** du type

$$\langle a, \sigma \rangle \xrightarrow{\text{pe}} \langle a', \sigma' \rangle$$

Une alternative

On aurait alors

$$\frac{\langle a_0, \sigma \rangle \dashv\text{pe}\rightarrow \langle a'_0, \sigma \rangle}{\langle a_0 + a_1, \sigma \rangle \dashv\text{pe}\rightarrow \langle a'_0 + a_1, \sigma \rangle}$$

$$\frac{\langle a_1, \sigma \rangle \dashv\text{pe}\rightarrow \langle a'_1, \sigma \rangle}{\langle n + a_1, \sigma \rangle \dashv\text{pe}\rightarrow \langle n + a'_1, \sigma \rangle}$$

$$\langle n + m, \sigma \rangle \dashv\text{pe}\rightarrow \langle n +_{\mathbb{N}} m, \sigma \rangle$$

Une alternative

On aurait alors

$$\frac{\langle a_0, \sigma \rangle \rightarrow_{pe} \langle a'_0, \sigma \rangle}{\langle a_0 + a_1, \sigma \rangle \rightarrow_{pe} \langle a'_0 + a_1, \sigma \rangle}$$
$$\frac{\langle a_1, \sigma \rangle \rightarrow_{pe} \langle a'_1, \sigma \rangle}{\langle n + a_1, \sigma \rangle \rightarrow_{pe} \langle n + a'_1, \sigma \rangle}$$
$$\langle n + m, \sigma \rangle \rightarrow_{pe} \langle n +_{\mathbb{N}} m, \sigma \rangle$$

Cela formalise une évaluation de gauche à droite des expressions.

Plan

Présentation du langage

La sémantique opérationnelle

Une sémantique «petite étapes» des expressions

Points fixes, fonctionnelles et récursivité

Une sémantique dénotationnelle de IMP

Une sémantique axiomatique de IMP

- Les deux premières instructions

- Le corps de la boucle

- Les boucles

- Les affectations

Correction de la sémantique axiomatique

Complétude de la sémantique axiomatique

La fonction de McCarthy

Considérons la définition

```
let rec m x = if x > 100 then x - 10 else m (m (x + 11))
```

A priori cela définit une **fonction partielle** (sauf pour les valeurs de $x > 100$ pour lesquelles on est certain de la valeur).

Laquelle ?

Un ordre sur les fonctions

Nous dirons que $h \sqsubseteq g$ si h est **moins définie** que g ¹ et là où h et g sont définies ensembles elles prennent la même valeur.

Un **dirigé** F dans $(\mathbb{N} \rightarrow \mathbb{N})$ est un ensemble de fonctions partielles tel que s'il contient deux fonctions h et g

- ▶ elles sont **cohérentes**, c'est-à-dire que si $h(x)$ et $g(x)$ sont tous les deux définis, alors $h(x) = g(x)$,

et de plus il contient

- ▶ une fonction l encore plus définie que chacune d'elles, c'est-à-dire si $h(x)$ ou $g(x)$ est défini, alors $l(x)$ est défini,
- ▶ et telle que l coïncide avec chacune d'elles sur leurs domaines de définition.

¹si h est définie en x , alors g est aussi définie en x □ ▶ ◀ ☰ ▶ ◀ ☰ ▶ ◀ ☰ ▶ ☰ 🔍 ↻

Un ordre sur les fonctions

Un **majorant** de E est un élément m tel que $(\forall x \in E) x \sqsubseteq m$.

La **borne supérieure** est le plus petit des majorants.

Une fonction est **continue** si elle préserve les bornes supérieures.

$$f(\sup E) = \sup f(E).$$

Fonction continue et point fixe

On note tout d'abord que si un ensemble est tel que toute partie a une borne supérieure,

alors il a un plus petit élément qui est la borne supérieure \perp de l'ensemble vide.

On peut montrer que toute fonction continue admet un point fixe :

$$\sup\{f^n(\perp) \mid n \in \mathbb{N}\}.$$

Fonctionnelles et points fixes

Considérons la fonctionnelle F

$$F(g) = \lambda x. \text{if } x > 100 \text{ then } x - 10 \text{ else } g(g(x + 11)).$$

La fonctionnelle F est continue. En effet,

- ▶ $\lambda g. \lambda x. g(g(x + 11))$ est continue par rapport à g .
- ▶ $\lambda g. \lambda x. \text{if } b(x) \text{ then } G(g)(x) \text{ else } H(g)(x)$ est continue par rapport à g si G et H sont continues.

F a donc un point fixe $\text{Fix}(F)$ tel que $F(\text{Fix}(F)) = \text{Fix}(F)$.

Fonctionnelles et points fixes

Le point fixe de F est obtenu en itérant F à partir de la fonction \perp nulle part définie.

$$F(\perp)(x) = \text{if } x > 100 \text{ then } x - 10$$

$$F^2(\perp)(x) = \text{if } x > 100 \text{ then } x - 10 \text{ else } F(\perp) (F(\perp) (x + 11))$$

⋮

$$F^{n+1}(\perp)(x) = \text{if } x > 100 \text{ then } x - 10 \text{ else } F^n(\perp) (F^n(\perp) (x + 11))$$

⋮

Fonctionnelles et points fixes

$$\text{Fix}(F) = \sup_{n \in \mathbb{N}} F^n(\perp)$$

Comme F est continue on

$$F(\text{Fix}(F)) = \text{Fix}(F)$$

Le point fixe de F

$Fix(F)$ est la fonction m définie par

```
let rec m x = if x > 100 then x - 10 else m (m (x + 11))
```

C'est aussi la fonction

```
let rec m91 x = if x > 100 then x - 10 else 91
```

1. Encore faudrait-il le prouver !
2. En sémantique dénotationnelle on s'intéresse aux fonctions ²**dénotées** par le programme.

²Même si nous nous intéressons aux fonctions calculables plutôt qu'aux algorithmes !

Plan

Présentation du langage

La sémantique opérationnelle

Une sémantique «petite étapes» des expressions

Points fixes, fonctionnelles et récursivité

Une sémantique dénotationnelle de IMP

Une sémantique axiomatique de IMP

- Les deux premières instructions

- Le corps de la boucle

- Les boucles

- Les affectations

Correction de la sémantique axiomatique

Complétude de la sémantique axiomatique

Quelques remarques

La sémantique opérationnelle **mélange** la syntaxe et les états dans sa définition.

Elle **colle** de très (trop) **près** à la syntaxe.

Et encore pas de la meilleure manière puisque la définition ne fait **pas par induction sur la syntaxe**.

Ce qu'on cherche à définir c'est une **fonction**
(au sens mathématique du terme).

On pourra de cette manière considérer des **programmes** qui sont **équivalents** parce qu'ils définissent **la même fonction mathématique**.

Les fonctions de base

$$\mathcal{A}[a] : \Sigma \rightarrow \mathbb{N},$$

$$\mathcal{B}[b] : \Sigma \rightarrow \mathbb{T},$$

$$\mathcal{C}[c] : \Sigma \rightarrow \Sigma$$

$\Sigma \rightarrow \Sigma$ représente l'ensemble des fonctions partielles de Σ vers Σ .

On a donc les fonctions d'interprétation sémantique

$$\mathcal{A}[\] : \mathbf{Aexp} \rightarrow \Sigma \rightarrow \mathbb{N},$$

$$\mathcal{B}[\] : \mathbf{Bexp} \rightarrow \Sigma \rightarrow \mathbb{T},$$

$$\mathcal{C}[\] : \mathbf{Com} \rightarrow \Sigma \rightarrow \Sigma$$

La dénotation des **Aexp**

$$\mathcal{A}[[n]]\sigma = n$$

$$\mathcal{A}[[X]]\sigma = \sigma(X)$$

$$\mathcal{A}[[a_0 + a_1]]\sigma = \mathcal{A}[[a_0]]\sigma +_{\mathbb{N}} \mathcal{A}[[a_1]]\sigma$$

$$\mathcal{A}[[a_0 - a_1]]\sigma = \mathcal{A}[[a_0]]\sigma -_{\mathbb{N}} \mathcal{A}[[a_1]]\sigma$$

$$\mathcal{A}[[a_0 \times a_1]]\sigma = \mathcal{A}[[a_0]]\sigma \times_{\mathbb{N}} \mathcal{A}[[a_1]]\sigma$$

La dénotation des **Aexp**

$$\mathcal{A}[[n]] = \lambda\sigma.n$$

$$\mathcal{A}[[X]] = \lambda\sigma.\sigma(X)$$

$$\mathcal{A}[[a_0 + a_1]] = \lambda\sigma.(\mathcal{A}[[a_0]]\sigma +_{\mathbb{N}} \mathcal{A}[[a_1]]\sigma)$$

$$\mathcal{A}[[a_0 - a_1]] = \lambda\sigma.(\mathcal{A}[[a_0]]\sigma -_{\mathbb{N}} \mathcal{A}[[a_1]]\sigma)$$

$$\mathcal{A}[[a_0 \times a_1]] = \lambda\sigma.(\mathcal{A}[[a_0]]\sigma \times_{\mathbb{N}} \mathcal{A}[[a_1]]\sigma)$$

La dénotation des **Bexp**

$$\mathcal{B}[\mathbf{true}] = \lambda\sigma.\mathbf{true}$$

$$\mathcal{B}[\mathbf{false}] = \lambda\sigma.\mathbf{false}$$

$$\mathcal{B}[a_0 = a_1] = \lambda\sigma.(\mathcal{A}[a_0]\sigma =_{\mathbb{N}} \mathcal{A}[a_1]\sigma)$$

⋮

$$\mathcal{B}[b_0 \wedge b_1] = \lambda\sigma.(\mathcal{B}[b_0]\sigma \wedge_{\mathbb{T}} \mathcal{B}[b_1]\sigma)$$

⋮

La dénotation des instructions

$$\mathcal{C}[\mathbf{skip}] = \lambda\sigma.\sigma$$

$$\mathcal{C}[X := a] = \lambda\sigma.\sigma[\mathcal{A}[a]\sigma/X]$$

$$\mathcal{C}[c_0; c_1] = \mathcal{C}[c_1] \circ \mathcal{C}[c_0]$$

$$\mathcal{C}[\mathbf{if } b \mathbf{ then } c_0 \mathbf{ else } c_1]\sigma = \mathcal{C}[c_0]\sigma \quad \text{si } \mathcal{B}[b]\sigma$$

$$\mathcal{C}[\mathbf{if } b \mathbf{ then } c_0 \mathbf{ else } c_1]\sigma = \mathcal{C}[c_1]\sigma \quad \text{si } \neg\mathcal{B}[b]\sigma$$

La dénotation de la boucle while

On cherche à dénoter la commande **while** b **do** c .

Soit $g \in \Sigma \rightarrow \Sigma$.

Considérons la «fonctionnelle» $\Gamma_g : (\Sigma \rightarrow \Sigma) \rightarrow (\Sigma \rightarrow \Sigma)$.

$$\Gamma_g(f)\sigma = (f \circ g)\sigma \quad \text{si } \mathcal{B}[[b]]\sigma$$

$$\Gamma_g(f)\sigma = \sigma \quad \text{si } \neg \mathcal{B}[[b]]\sigma$$

La fonctionnelle Γ_g

La fonctionnelle Γ_g est continue pour la relation \sqsubseteq .

$$\begin{aligned}(\sup_{h \in F} \Gamma_g(h))\sigma &= ((\sup_{h \in F} h) \circ g)\sigma && \text{si } \mathcal{B}[b]\sigma \\(\sup_{h \in F} \Gamma_g(h))\sigma &= \sigma && \text{si } \neg \mathcal{B}[b]\sigma\end{aligned}$$

La fonctionnelle Γ_g a un point fixe $Fix(\Gamma_g)$
qui est une fonction partielle de Σ vers Σ .

$$Fix(\Gamma_g) : \Sigma \rightarrow \Sigma.$$

La dénotation de la boucle while

Remarquons que $\Gamma_g(\perp)$ n'est défini que sur les états σ tels que $\neg \mathcal{B}[[b]]\sigma$ et qu'alors

$$\Gamma_g(\perp)\sigma = \sigma$$

et que $\Gamma_g^{n+1}(\perp)$ est défini sur les états σ tels que

- ▶ $g^i\sigma$ est défini pour $i \leq n+1$
- ▶ et $\mathcal{B}[[b]]g^i\sigma$ pour $i \leq n$, tandis que $\neg \mathcal{B}[[b]]g^{n+1}\sigma$.

Et on a

$$\Gamma_g^{n+1}(\perp)(\sigma) = g^{n+1}(\sigma).$$

La dénotation de la boucle while

Le point fixe de Γ_g est donc défini pour tous les σ tels qu'il existe un n avec

- ▶ $g^i(\sigma)$ défini et $\mathcal{B}[[b]]g^i(\sigma)$ pour $i < n$
- ▶ et $g^n(\sigma)$ défini et $\neg\mathcal{B}[[b]]g^n(\sigma)$.

La dénotation de la boucle while

En fait, on s'intéresse à la fonctionnelle

$$\begin{aligned}\Gamma_{\mathcal{C}[[c]]}(f)\sigma &= (f \circ \mathcal{C}[[c]])\sigma && \text{si } \mathcal{B}[[b]]\sigma \\ \Gamma_{\mathcal{C}[[c]]}(f)\sigma &= \sigma && \text{si } \neg\mathcal{B}[[b]]\sigma\end{aligned}$$

On pose

$$\mathcal{C}[[\mathbf{while} \ b \ \mathbf{do} \ c]] = \text{Fix}(\Gamma_{\mathcal{C}[[c]]}).$$

Plan

Présentation du langage

La sémantique opérationnelle

Une sémantique «petite étapes» des expressions

Points fixes, fonctionnelles et récursivité

Une sémantique dénotationnelle de IMP

Une sémantique axiomatique de IMP

- Les deux premières instructions

- Le corps de la boucle

- Les boucles

- Les affectations

Correction de la sémantique axiomatique

Complétude de la sémantique axiomatique

Qu'est-ce que l'annotation d'un programme par des propositions ?

A départ, il s'agit de prouver la correction des programmes en examinant des **instantanés**³ successifs de l'état pendant le déroulement du programme.

Les concepteurs voulaient seulement prouver des propriétés des programmes.

Aujourd'hui c'est utilisé *aussi* pour décrire leur signification.

³L'expression **snapshot** et due à Floyd.

La notation des assertions

On va énoncer des **assertions de corrections** sur des instructions.
Puis des des règles d'inférence sur ces assertions de correction.
Les assertions de corrections s'écriront

$$\{A\} c \{B\}$$

A est la **précondition** de **c**.

B est la **postcondition de correction partielle** de **c**.

L'assertion doit se lire

*«Si **A** est satisfaite avant l'exécution de **c** et si l'exécution de **c se termine** alors **B** est satisfaite après l'exécution de **c**».*

L'annotation d'un programme

Si l'on veut annoter un programme par des propriétés, on utilise une notation comme celle-ci

Pre : *Condition₁*

Texte ... du ... programme

Post : *Condition₂*

C'est très commode pour la mise au point.

L'annotation d'un programme

Pre : *Condition₁*

Texte ... du ... programme

Post : *Condition₂*

est équivalent à

{*Condition₁*} Texte ... du ... programme {*Condition₂*}

Un exemple

Considérons

```
q := 0; r := x; while r >= y do r := r - y; q := q + 1
```

Un exemple

On annote le programme

Pre : $x \geq 0$

```
q := 0; r := x; while r >= y do r := r - y; q := q + 1
```

Post : $0 \leq r < y \wedge x = q \times y + r$

Pre Post

Pre représente l'instantané avant l'exécution du programme.

Post représente l'instantané après.

Un exemple

Compte tenu de ce qu'on a dit, on a aussi

Pre : $x \geq 0$

```
q := 0 ; r := x ; while r >= y do r := r + y ; q := q + 1
```

Post : $0 \leq r < y \quad \wedge \quad x = q \times y + r$

Car comme la boucle ne va jamais s'arrêter, la définition est satisfaite.

Collection de quelques propriétés

Puisque la condition d'exécution de la boucle est $r \geq y$
après la boucle on a $\neg(r \geq y)$ donc $r < y$.

Comme on exécute la boucle sous la condition $r \geq y$, la soustraction $r - y$ produira toujours une valeur positive et r sera toujours positif ⁴.

On introduit donc naturellement la notion d'**invariant** de boucle, c'est-à-dire une propriété qui **est satisfaite** au début et à la fin de l'exécution du «corps» de la boucle.

- ▶ $r \geq 0$ est un invariant de la boucle.
- ▶ $x = q \times y + r$ est un autre invariant.

⁴A condition que ?

Les deux premières instructions

Pre : $x \geq 0$

$q := 0$; $r := x$;

Post : $x \geq 0 \wedge q = 0 \wedge r = x$

Les deux premières instructions

Pre : $x \geq 0$

$q := 0$; $r := x$;

Post : $x \geq 0 \wedge q = 0 \wedge r = x$

Duquel on déduit qu'après ces deux instructions on a

$r \geq 0 \wedge x = q \times y + r$

Le corps de la boucle

On souhaite

Pre : $r \geq 0 \wedge x = q \times y + r$

$r := r - y ; q := q + 1$

Post : $r \geq 0 \wedge x = q \times y + r$

car c'est l'**invariant** de la boucle.

Ça ne suffit pas.

Le corps de la boucle

Il faut

Pre : $r \geq y \wedge r \geq 0 \wedge x = q \times y + r$

$r := r - y ; q := q + 1$

Post : $r \geq 0 \wedge x = q \times y + r$

Mais en fait $r \geq y$ est la **condition** de la boucle,
donc elle est satisfaite, comme précondition.

Les boucles

La règle concernant les boucles peut s'énoncer ainsi

On rentre dans le corps avec $Pre : b \wedge I$.

On sort de toute la boucle⁵ avec $Post : \neg b \wedge I$.

où b est la condition de la boucle
et I est l'invariant de la boucle

⁵Si toutefois on en sort !

Les boucles

Plus formellement on écrit

$$\frac{\{b \wedge I\} c \{I\}}{\{I\} \mathbf{while} \ b \ \mathbf{do} \ c \ \{\neg b \wedge I\}}$$

et l'on espère avoir tout dit !

Les boucles

$$\frac{\{b \wedge I\} c \{I\}}{\{I\} \text{ while } b \text{ do } c \{\neg b \wedge I\}}$$

dit que

- ▶ si I est satisfaite avant l'exécution de la boucle
- ▶ et si la boucle se termine

alors $\neg b \wedge I$ est satisfaite.

Les affectations

On a

Pre : ?

$q := q + 1$

Post : $r \geq 0 \wedge x = q \times y + r$

Les affectations

En fait

Pre : $r \geq 0 \wedge x = (q + 1) \times y + r$

$q := q + 1$

Post : $r \geq 0 \wedge x = q \times y + r$

Et

Les affectations

Pre : $r - y \geq 0 \quad \wedge \quad x = (q + 1) \times y + r - y$

$r := r - y$

Post : $r \geq 0 \quad \wedge \quad x = (q + 1) \times y + r$

Pre : $r \geq 0 \quad \wedge \quad x = (q + 1) \times y + r$

$q := q + 1$

Post : $r \geq 0 \quad \wedge \quad x = q \times y + r$

Les affectations

Pre : $r - y \geq 0 \quad \wedge \quad x = (q + 1) \times y + r - y$

r := **r** - **y**

Post : $r \geq 0 \quad \wedge \quad x = (q + 1) \times y + r$

Pre : $r \geq 0 \quad \wedge \quad x = (q + 1) \times y + r$

q := **q** + 1

Post : $r \geq 0 \quad \wedge \quad x = q \times y + r$

ce qui est bien ce qu'on veut parce que

$(q + 1) \times y + r - y = q \times y + r$ et $r - y \geq 0 \Leftrightarrow r \geq y$.

Les affectations

La règle de l'affectation est donc

$$\{B[X \leftarrow a]\} X := a \{B\}$$

Les affectations

La règle de l'affectation est donc

$$\{B[X \leftarrow a]\} X := a \{B\}$$

Remarquer l'intérêt d'aller «en marche arrière»
plutôt qu'«en marche avant».

Les affectations

La règle de l'affectation est donc

$$\{B[X \leftarrow a]\} X := a \{B\}$$

On définit $B[X \leftarrow a]$ comme d'habitude par induction sur B .

La signification d'une assertion de correction

$\sigma \models A$ signifie que A est satisfaite dans l'état σ .

$\{A\} c \{B\}$ signifie

$$\forall \sigma : \Sigma. (\sigma \models A \ \& \ C[[c]]\sigma \text{ est défini}) \Rightarrow C[[c]]\sigma \models B$$

La signification d'une assertion de correction

En fait au lieu de **est défini** on étend $\mathcal{C}[[c]]$ en une fonction totale qui vaut \perp là où $\mathcal{C}[[c]]$ devrait ne pas être définie et l'on pose

$$\perp \vDash A$$

pour toute proposition A .

Avec cette convention on peut dire que $\{A\} c \{B\}$ signifie

$$\forall \sigma : \Sigma. \quad \sigma \vDash A \Rightarrow \mathcal{C}[[c]]\sigma \vDash B$$

Le langage des propositions

Pour pouvoir affirmer des propriétés de l'état,
il nous faut **un langage assez riche**.

On veut

- ▶ tous les prédicats booléens du langage IMP,
- ▶ des quantifications sur des valeurs entières,
- ▶ ce qui induit qu'il faut introduire des variables libres destinées à être quantifiées.

Le langage enrichi des expressions **Aexpv**

On définit tout d'abord le langage enrichi de **variables libres**
Aexpv :

$$a ::= n \mid X \mid i \mid a_0 + a_1 \mid a_0 - a_1 \mid a_0 \times a_1$$

où i sont des variables entières formant **Intvar**.

Le langage enrichi de propositions **Bexpv**

Les propositions comportent des expressions arithmétiques enrichies et on y ajoute des implications et des quantifications sur les variables de **Intvar**.

On définit **Bexpv** :

$$A ::= \text{true} \mid \text{false} \mid a_0 = a_1 \mid a_0 \leq a_1 \mid \neg A \mid A_0 \vee A_1 \mid A_0 \wedge A_1 \\ \mid A_0 \Rightarrow A_1 \mid \forall i. A \mid \exists i. A$$

Substitutions dans les propositions

On peut faire des substitutions dans les propositions

On les note $A[i \leftarrow a]$.

Une **valuation** est une fonction $\rho : \mathbf{Intvar} \rightarrow \mathbb{N}$.

Sémantique des expressions arithmétiques enrichies

On définit une fonction

$\mathcal{A}v[\] : \mathbf{Aexpv} \rightarrow (\mathbf{Intvar} \rightarrow \mathbb{N}) \rightarrow \Sigma \rightarrow \mathbb{N} :$

$$\mathcal{A}v[n]\rho\sigma = n$$

$$\mathcal{A}v[X]\rho\sigma = \sigma(X)$$

$$\mathcal{A}v[i]\rho\sigma = \rho(i)$$

$$\mathcal{A}v[a_0 + a_1]\rho\sigma = \mathcal{A}v[a_0]\rho\sigma +_{\mathbb{N}} \mathcal{A}v[a_1]\rho\sigma$$

$$\mathcal{A}v[a_0 - a_1]\rho\sigma = \mathcal{A}v[a_0]\rho\sigma -_{\mathbb{N}} \mathcal{A}v[a_1]\rho\sigma$$

$$\mathcal{A}v[a_0 \times a_1]\rho\sigma = \mathcal{A}v[a_0]\rho\sigma \times_{\mathbb{N}} \mathcal{A}v[a_1]\rho\sigma$$

Sémantique des assertions

La sémantique des expressions booléennes

$\sigma \models^{\rho} \mathbf{true}$	if <i>rien</i>
$\sigma \models^{\rho} a_0 = a_1$	if $\mathcal{A}v[a_0]\rho\sigma =_{\mathbb{N}} \mathcal{A}v[a_1]\rho\sigma$
$\sigma \models^{\rho} a_0 \leq a_1$	if $\mathcal{A}v[a_0]\rho\sigma \leq_{\mathbb{N}} \mathcal{A}v[a_1]\rho\sigma$
$\sigma \models^{\rho} A \wedge B$	if $\sigma \models^{\rho} A$ et $\sigma \models^{\rho} B$
$\sigma \models^{\rho} A \vee B$	if $\sigma \models^{\rho} A$ ou $\sigma \models^{\rho} B$
$\sigma \models^{\rho} \neg A$	if $\sigma \not\models^{\rho} A$

Sémantique des assertions

La sémantique des expressions booléennes

$\sigma \models^\rho A \Rightarrow B$	if de $\sigma \models^\rho A$ on peut déduire $\sigma \models^\rho B$
$\sigma \models^\rho \forall i.A$	if $\sigma \models^{\rho(i:=n)} A$ pour tout $n \in \mathbb{N}$
$\sigma \models^\rho \exists i.A$	if $\sigma \models^{\rho(i:=n)} A$ pour un $n \in \mathbb{N}$
$\perp \models^\rho A$	if <i>rien</i>

Quelques résultats

On peut prouver que pour n'importe quelle valuation ρ

- ▶ $\mathcal{B}[[b]]\sigma = \mathbf{true}$ si et seulement si $\sigma \models^{\rho} b$
- ▶ $\mathcal{B}[[b]]\sigma = \mathbf{false}$ si et seulement si $\sigma \not\models^{\rho} b$
- ▶ $\mathcal{A}v[[a]]\rho(i:=n)\sigma = \mathcal{A}v[[a[i \leftarrow n]]]\rho\sigma$

Satisfaction d'une assertion de correction

$$\models^{\rho} \{A\} c \{B\} \quad \text{ssi} \quad \forall \sigma : \Sigma. \sigma \models^{\rho} A \Rightarrow C[[c]]\sigma \models^{\rho} B$$

Validité d'une assertion de correction

$\models \{A\} c \{B\}$ ssi pour toute valuation ρ et tout état σ
 $\sigma \models^\rho \{A\} c \{B\}$

Les règles de correction partielle 1/4

Règle de l'instruction vide :

$$\{A\} \text{ skip } \{A\}$$

Règle de l'affectation :

$$\{B[X \leftarrow a]\} X := a \{B\}$$

Les règles de correction partielle 2/4

Règle de la composition séquentielle

$$\frac{\{A\} c_0 \{C\} \quad \{C\} c_1 \{B\}}{\{A\} c_0; c_1 \{B\}}$$

Règle de la composition conditionnelle :

$$\frac{\{A \wedge b\} c_0 \{B\} \quad \{\neg b \wedge A\} c_1 \{B\}}{\{A\} \text{ if } b \text{ then } c_0 \text{ else } c_1 \{B\}}$$

Les règles de correction partielle 3/4

Règle de la boucle :

$$\frac{\{b \wedge I\} c \{I\}}{\{I\} \text{ while } b \text{ do } c \{\neg b \wedge I\}}$$

Règle de conséquence :

$$\frac{\vDash A \Rightarrow A' \quad \{A'\} c \{B'\} \quad \vDash B' \Rightarrow B}{\{A\} c \{B\}}$$

Les règles de correction partielle 4/4

Remarquer que la règle de conséquence s'appuie sur des propriétés «valides».

L'exemple

Remarquons que l'on utilise la règle de conséquence, car

$$\begin{aligned} \models r \geq y \wedge r \geq 0 \wedge x = q \times y + r &\Rightarrow r - y \geq 0 \wedge x = (q + 1) \times y + r - y \\ \{r - y \geq 0 \wedge x = (q + 1) \times y + r - y\} r := r - y; q := q + 1 \{r \geq 0 \wedge x = q \times y + r\} \\ \models r \geq 0 \wedge x = q \times y + r &\Rightarrow r \geq 0 \wedge x = q \times y + r \end{aligned}$$

$$\{r \geq y \wedge r \geq 0 \wedge x = q \times y + r\} r := r - y; q := q + 1 \{r \geq 0 \wedge x = q \times y + r\}$$

Plan

Présentation du langage

La sémantique opérationnelle

Une sémantique «petite étapes» des expressions

Points fixes, fonctionnelles et récursivité

Une sémantique dénotationnelle de IMP

Une sémantique axiomatique de IMP

- Les deux premières instructions

- Le corps de la boucle

- Les boucles

- Les affectations

Correction de la sémantique axiomatique

Complétude de la sémantique axiomatique

Correction (cas des expressions arithmétiques)

Lemme : Soient ρ une valuation, $a \in \mathbf{Aexpv}$, $a_0 \in \mathbf{Aexpv}$,
 $X \in \mathbf{Loc}$, $\sigma \in \Sigma$

$$\mathcal{A}v[[a_0[X \leftarrow a]]]\rho\sigma = \mathcal{A}v[[a_0]]\rho(\sigma[\mathcal{A}v[[a]]\rho\sigma/X])$$

Correction (cas des expressions arithmétiques)

Lemme : Soient ρ une valuation, $a \in \mathbf{Aexpv}$, $a_0 \in \mathbf{Aexpv}$, $X \in \mathbf{Loc}$, $\sigma \in \Sigma$

$$\mathcal{A}v[[a_0[X \leftarrow a]]]\rho\sigma = \mathcal{A}v[[a_0]]\rho(\sigma[\mathcal{A}v[[a]]\rho\sigma/X])$$

$a[X \leftarrow a_0]$ est défini comme d'habitude.

Correction (cas des expressions booléennes)

Lemme : Soient ρ une valuation, $a \in \mathbf{Aexp}$, $X \in \mathbf{Loc}$, $\sigma \in \Sigma$ et B une expression booléenne de \mathbf{Bexpv}

$$\sigma \models^\rho B[X \leftarrow a] \quad \text{ssi} \quad \sigma[\mathcal{A}[a]\sigma/X] \models^\rho B$$

Correction (cas des instructions)

Théorème : Si $\vdash \{A\} c \{B\}$ alors $\models \{A\} c \{B\}$.

Correction (cas des instructions)

Théorème : Si $\vdash \{A\} c \{B\}$ alors $\models \{A\} c \{B\}$.

Démonstration :

La démonstration se fait par induction sur la preuve de $\vdash \{A\} c \{B\}$.

- ▶ **Correction de skip**. Dans ce cas, $A \equiv B$
et on a $\vdash \{A\} \text{skip} \{A\}$.
Comme $\mathcal{C}[\text{skip}]\sigma = \sigma$, clairement

$$\sigma \models^p A \quad \Rightarrow \quad \mathcal{C}[\text{skip}]\sigma \models^p A$$

donc $\models \{A\} \text{skip} \{A\}$.

Correction (cas des instructions)

- *Correction de l'affectation.* Si c est $X := a$ alors
 $A \equiv B[X \leftarrow a]$.

Comme on a $\sigma \vDash^{\rho} B[X \leftarrow a]$ ssi $\sigma[\mathcal{A}[[a]]\sigma/X] \vDash^{\rho} B$,
(par le lemme précédent).

alors clairement

$$\sigma \vDash^{\rho} B[X \leftarrow a] \Rightarrow \mathcal{C}[[X := a]]\sigma \vDash^{\rho} B$$

donc $\vDash \{B[X \leftarrow a]\} X := a \{B\}$.

Correction (cas des instructions)

- *Correction de la composition séquentielle.*

Supposons $\vdash \{A\} c_0; c_1 \{B\}$, alors il existe C

telle que $\vdash \{A\} c_0 \{C\}$ et $\vdash \{C\} c_1 \{B\}$

et par induction, $\models \{A\} c_0 \{C\}$ et $\models \{C\} c_1 \{B\}$.

Soit ρ une valuation quelconque et σ un état tel que $\sigma \models^\rho A$.

Parce que $\models^\rho \{A\} c_0 \{C\}$, on a $\mathcal{C}[\![c_0]\!] \sigma \models^\rho C$

duquel associé à $\models^\rho \{C\} c_1 \{B\}$ on tire $\mathcal{C}[\![c_1]\!](\mathcal{C}[\![c_0]\!] \sigma) \models^\rho B$,

c'est-à-dire $\mathcal{C}[\![c_0; c_1]\!] \sigma \models^\rho B$ (par définition de $\mathcal{C}[\![_]\!]$).

Donc $\models \{A\} c_0; c_1 \{B\}$.

Correction (cas des instructions)

- ▶ *Correction de la composition conditionnelle.* En exercice!

Correction (cas des instructions)

- *Correction de la boucle.*

Supposons que $\vdash \{A\} \text{ while } b \text{ do } c' \{B\}$.

On a forcément $B \equiv A \wedge \neg b$.

On a aussi $\vdash \{A \wedge b\} c' \{A\}$.

Donc $\models \{A \wedge b\} c' \{A\}$.

On va démontrer $P(n)$ par induction sur n , où

$$P(n) \triangleq (\forall \sigma \in \Sigma) \quad \sigma \models^{\rho} A \Rightarrow \Gamma_{\mathcal{C}[[c']]}^n(\perp)(\sigma) \models^{\rho} A \wedge \neg b.$$

Il s'ensuivra que

$$\sigma \models^{\rho} A \Rightarrow \mathcal{C}[[\mathbf{while} \ b \ \mathbf{do} \ c']](\sigma) \models^{\rho} A \wedge \neg b.$$

et donc $\models \{A\} \ \mathbf{while} \ b \ \mathbf{do} \ c' \ \{A \wedge \neg b\}$

Correction (cas des instructions)

Preuve de $(\forall \sigma \in \Sigma) \quad \sigma \models^\rho A \Rightarrow \Gamma_{\mathcal{C}[[c']]}^n(\perp)(\sigma) \models^\rho A \wedge \neg b.$

- Si $n = 0$ alors $\Gamma_{\mathcal{C}[[c']]}^n(\perp)(\sigma) = \perp(\sigma) = \perp$

donc $\Gamma_{\mathcal{C}[[c']]}^n(\perp)(\sigma) \models^\rho A \wedge \neg b.$

- Rappelons que $\Gamma_{\mathcal{C}[[c']]}^{n+1}(\perp)$ est défini sur les états σ tels que

- ▶ $\mathcal{C}[[c']]^i \sigma$ est défini pour $i \leq n + 1$

- ▶ et $\mathcal{B}[[b]](\mathcal{C}[[c']]^i \sigma) = \mathbf{true}$ pour $i \leq n$, tandis que $\mathcal{B}[[b]](\mathcal{C}[[c']]^{n+1} \sigma) = \mathbf{false}.$

Et on a

$$\Gamma_{\mathcal{C}[[c']]}^{n+1}(\perp)(\sigma) = \mathcal{C}[[c']]^{n+1} \sigma.$$

Correction (cas des instructions)

D'autre part par induction

$$\mathcal{C}[[c']^n(\sigma) \models^{\rho} A \wedge \neg b \Rightarrow \mathcal{C}[[c']^{n+1}(\sigma) \models^{\rho} A$$

parce que par hypothèse $\models \{A \wedge b\} c' \{A\}$,
donc $\sigma' \models^{\rho} A$ et $\mathcal{B}[[b]\sigma'$ impliquent $\mathcal{C}[[c'](\sigma') \models^{\rho} A$.

Donc $\sigma \models^{\rho} A \Rightarrow \mathcal{C}[[c']^{n+1}(\sigma) \models^{\rho} A$.

D'autre part, $\neg \mathcal{B}[[b]\mathcal{C}[[c']^{n+1}\sigma$ signifie $\mathcal{C}[[c']^{n+1}(\sigma) \models^{\rho} \neg b$.

Donc $\sigma \models^{\rho} A \Rightarrow \mathcal{C}[[c']^{n+1}(\sigma) \models^{\rho} A \wedge \neg b$.

Correction (cas des instructions)

- ▶ *Correction de la règle de conséquence.* Supposons qu'on ait
 - ▶ $\models A \Rightarrow A'$,
 - ▶ $\vdash \{A'\} c \{B'\}$ donc par induction $\models \{A'\} c \{B'\}$
 - ▶ et $\models B' \Rightarrow B$.

Soit une valuation ρ . Supposons qu'on ait $\sigma \models^\rho A$, alors $\sigma \models^\rho A'$.

Donc $C[[c]]\sigma \models^\rho B'$, donc aussi $C[[c]]\sigma \models^\rho B$.

Par conséquent $\models \{A\} c \{B\}$.

Plan

Présentation du langage

La sémantique opérationnelle

Une sémantique «petite étapes» des expressions

Points fixes, fonctionnelles et récursivité

Une sémantique dénotationnelle de IMP

Une sémantique axiomatique de IMP

- Les deux premières instructions

- Le corps de la boucle

- Les boucles

- Les affectations

Correction de la sémantique axiomatique

Complétude de la sémantique axiomatique

Précondition la plus faible

Quand on veut prouver $\{A\} c_0; c_1 \{B\}$, la question se pose de savoir si on peut trouver une proposition C à mettre au milieu c'est-à-dire tel que $\{A\} c_0 \{C\}$ et $\{C\} c_1 \{B\}$.

L'idée c'est de la construire comme la **précondition la plus faible** associée à B et à l'instruction c_1 .

Précondition la plus faible

Tout d'abord on définit un ensemble d'états (pas une proposition !) :

$$wp^\rho \llbracket c, B \rrbracket =^{def} \{ \sigma \in \Sigma_\perp \mid C \llbracket c \rrbracket \sigma \models^\rho B \}.$$

C'est-à-dire l'image inverse par la fonction $C \llbracket c \rrbracket$ de l'ensemble

$$\{ \sigma' \in \Sigma_\perp \mid \sigma' \models^\rho B \}.$$

Précondition la plus faible

Supposons qu'il existe une proposition A_0 telle que pour toute valuation ρ , on ait $\{\sigma \in \Sigma \mid \sigma \models^\rho A_0\} = wp^\rho[[c, B]]$.

Alors

$$\models^\rho \{A\} c \{B\} \quad \text{ssi} \quad \models^\rho A \Rightarrow A_0$$

autrement dit

$$\models \{A\} c \{B\} \quad \text{ssi} \quad \models A \Rightarrow A_0$$

Avec la règle de conséquence on devrait pouvoir construire les preuves de correction partielle d'un programme.

Expressivité

Un langage de propositions est **expressif** si pour toute instruction c et toute proposition B il existe une proposition, notée $w[[c, B]]$, telle que pour toute valuation ρ on ait

$$\{\sigma \in \Sigma \mid \sigma \models^\rho w[[c, B]]\} = wp^\rho[[c, B]]$$

Attention : $w[[c, B]]$ est une expression du langage, alors que $wp^\rho[[c, B]]$ est un sous-ensemble de Σ .

Expressivité

Théorème : **Aexpv** est expressif.

Démonstration :

On l'obtient au prix de codages à la Gödel.

Voyez le livre de Glynn Winskel **The Formal semantics of programming language**.

Complétude

Les règles de la sémantique axiomatique forment un système complet.

Théorème : (Cook) Si $\models \{A\} c \{B\}$ alors $\vdash \{A\} c \{B\}$.

Complétude

Démonstration : En gros on arrive à montrer que

$$\models \{A\} c \{B\}$$

implique

$$\vdash \{w[c, B]\} c \{B\}$$

où $w[c, B]$ est une proposition qui exprime la précondition la plus faible, associée à B et à c .

On peut aussi montrer $\models A \Rightarrow w[c, B]$.

Et donc on conclut par la règle de conséquence.