Un problème de point-virgule

Dominique Duval, travail en cours avec
Jean-Claude Reynaud, Jean-Guillaume Dumas, Laurent Fousse

Séminaire MAD, Grenoble, 29 mai 2008

Outline

Introduction

La syntaxe de IMP

La mémoire

Des sémantiques pour IMP

Conclusion

Une syntaxe

Un langage a une syntaxe : c'est une grammaire qui fournit les règles pour former les expressions du langage.

Nous allons nous intéresser à un langage très simple :

IMP, ou while-language.

C'est un langage impératif, comme Pascal ou C.

Exemple de commande en IMP :

$$\text{while } X \neq 0 \text{ do } X := X - 1$$

Deux sémantiques

Un langage a plusieurs sémantiques :
chacune fournit une signification pour la syntaxe.

- Signification pour le compilateur :
  une sémantique opérationnelle
  pour évaluer les expressions du langage.
- Signification pour l'utilisateur :
  une sémantique dénotationnelle
  pour interpréter les expressions du langage
  en termes mathématiques.

On espère que les sémantiques d'un langage coïncident :
pour IMP c'est vrai, mais en général c'est (un peu) FAUX !

“Interface Homme-Machine”

Les deux sémantiques forment une sorte

d’“Interface Homme-Machine”,

au niveau de la logique...

Langage	
Syntaxe	
Machine	**Homme**
Sémantique opérationnelle évaluation	Sémantique dénotationnelle interprétation

Soit c la commande $X := X - 1$

- Pour le compilateur :
  l'évaluation de c, à partir d'un état de la mémoire, consiste à décrémenter la valeur de X de 1.
- Pour l'utilisateur :
  l'interprétation de c est la fonction qui associe à chaque état de la mémoire un nouvel état, dans lequel la valeur de X a diminué de 1.

Currification :
une fonction à 2 arguments : $(c, s) \mapsto s'$
peut être vue comme une fonction à 1 argument dont le résultat est une fonction à 1 argument : $c \mapsto (s \mapsto s')$.

Sém. opérationnelle	**Sém. dénotationnelle**
$C \times S \to S$	$C \to S^S$

Soit w la commande $\text{while } X \neq 0 \text{ do } X := X - 1$

- Pour le compilateur :
  l'évaluation de w, à partir d'un état de la mémoire, consiste à tester si X est nul, si c'est le cas on s'arrête sans rien faire, sinon on décrémente la valeur de X de 1 et on recommence.
- Pour l'utilisateur :
  l'interprétation de w est la fonction partielle $[[w]]$ qui associe à chaque état de la mémoire où la valeur de X est ≥ 0 un nouvel état, où la valeur de X est mise à 0, et qui n'est pas définie si la valeur de X est < 0.

Théorème de point fixe (Knaster-Tarski).
La fonction $[[w]]$ peut être définie comme un plus petit point fixe.

Sém. opérationnelle	**Sém. dénotationnelle**
boucle	point fixe

Calculabilité ($\sim$1935)

Thèse de Church.

Toutes les définitions “raisonnables” des fonctions calculables sont équivalentes.

ou :
Tous les langages de programmation “raisonnables” ont la même puissance.

Théorème (Turing).

Le problème de l’arrêt est indécidable.

Logique - Informatique - Catégories

- 1935 : Logique pour l'informatique : Turing, Gödel, Church
- 1945 : Catégories : Eilenberg et Mac Lane
- 1963 : Logique catégorique : Lawvere
- 1965 : Esquisses : Ehresmann
- 1934...1970 : Correspondance de Curry-Howard-Lambek :
  "Logique = Informatique = Catégories"
  pour les langages fonctionnels.

Logique "traditionnelle"	symboles	$\forall, \exists, \ldots$
Logique catégorique	flèches	$X \xrightarrow{f} Y$
Logique diagrammatique [Duval et Lair 2002]	flèches en couleur	$X \xrightarrow{f} Y$

Une motivation

Famille de langages	**Langage “d’école”**	Logique “traditionnelle”	Catégories et diagrammes
fonctionnel	PCF	–	–
impératif	IMP	–	?
objet	?		?

J. C. Mitchell.
Foundations for Programming Languages.

G. Winskel.
Formal Semantics of Programming Languages.

Outline

Introduction

La syntaxe de IMP

La mémoire

Des sémantiques pour IMP

Conclusion

Expressions et commandes

Les expressions du langage IMP :

- des variables :
  $v \ ::= \ X \mid Y \mid ...$
- des expressions arithmétiques :
  $a \ ::= \ v \mid ... \mid -1 \mid 0 \mid 1 \mid ... \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 \times a_2$
- des expressions booléennes :
  $b \ ::= \ \text{true} \mid \text{false} \mid \neg b \mid b_1 \wedge b_2 \mid b_1 \vee b_2 \mid a_1 = a_2 \mid a_1 < a_2$

et des “programmes” et “procédures” :

- des commandes :

$$c \ ::= \ \text{skip} \mid v := a \mid c_1 ; c_2 \mid \text{if } b \text{ then } c_1 \text{ else } c_2 \mid \text{while } b \text{ do } c_1$$

Arbres syntaxiques

Quelques expressions arithmétiques :

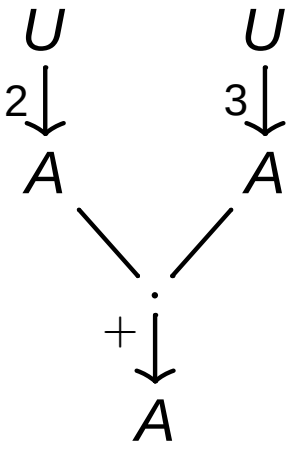
$2+3, \quad 3+2, \quad 5, \quad 4 \times X$,...

Quelques commandes :

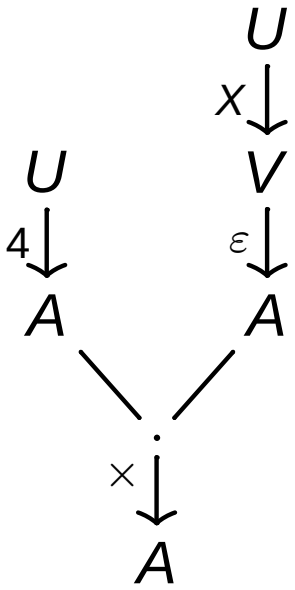
$Y := X$, $\text{while } X \neq 0 \text{ do } X := X - 1$, skip, $\text{while true do skip}$,...

Quelques arbres syntaxiques :

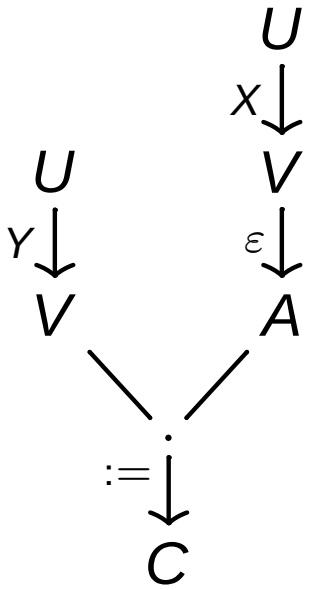
$2+3$



$4 \times X$

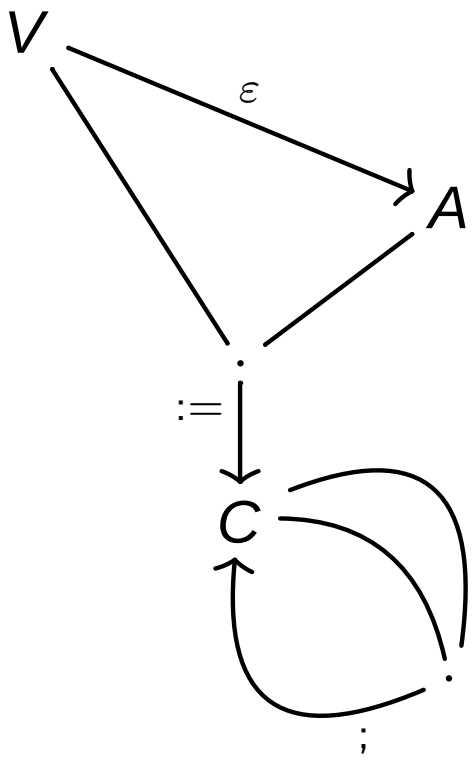


$Y := X$



La spécification de la syntaxe

C'est un "graphe", en voici une partie :



Notion de modèle :

- chaque point représente un ensemble :
  C représente l'ensemble des commandes,
- chaque flèche représente une fonction :
  $V \xrightarrow{\varepsilon} A$ représente l'inclusion de l'ensemble des variables dans l'ensemble des expressions arithmétiques,

Le point U représente un singleton : $U = \mathrm{Unit} = \mathrm{void}$.
Donc les flèches de source U représentent des constantes :
$U \xrightarrow{1} A$ pour l'expression 1, et $U \xrightarrow{X} V$ pour la variable X.

Outline

Introduction

La syntaxe de IMP

La mémoire

Des sémantiques pour IMP

Conclusion

Les états de la mémoire

Comme pour tout langage impératif, les sémantiques de IMP font intervenir les états de la mémoire.
On peut supposer que la mémoire est un tableau d'entiers indexé par les variables, alors chaque état correspond à une valeur de ce tableau :

X	Y	X_1	Y_1	...
20	0	-5	46	...

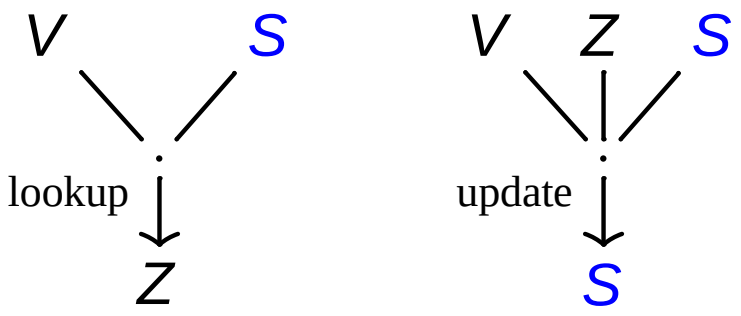
Notations : Z pour l'ensemble des entiers,
S pour l'ensemble des états de la mémoire.

Donc $S = Z^V$, ensemble des fonctions de V vers Z.

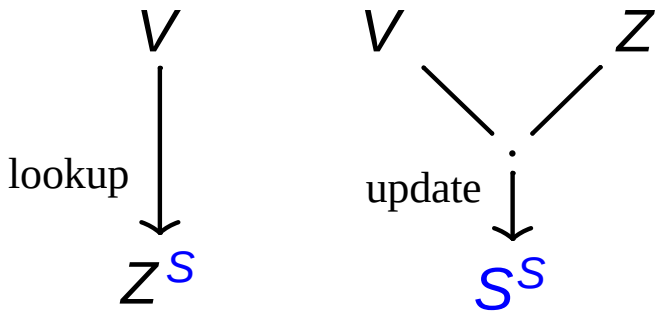
Observer et mettre à jour la mémoire

On peut observer la valeur d'une variable par lookup ou modifier la valeur d'une variable par update .

Version non currifiée :



Version currifiée :



Une équation

Les fonctions lookup et update sont liées par une équation :

Si on observe la valeur de v juste après avoir mis à jour la valeur de w à l'entier n, alors :

- ou bien $v = w$ et on obtient n,
- ou bien $v \neq w$ et on obtient la même valeur qu'avant la mise à jour.

Version non currifiée :

$$\begin{aligned} & \text{lookup}\ (v,\ \text{update}\ (w, n, s)) \\ \equiv\ & \text{si}\ (v = w)\ \text{alors}\ n\ \text{sinon}\ \ \text{lookup}\ (v, s) \end{aligned}$$

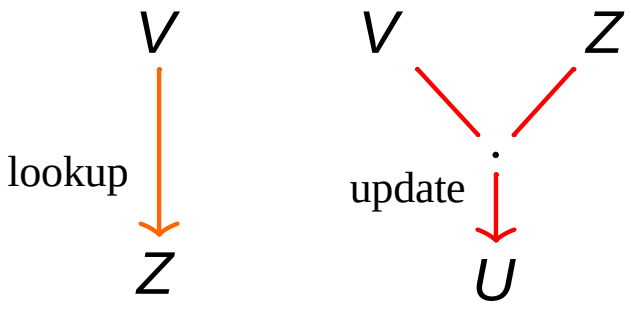
Version currifiée :

$$\begin{aligned} & \text{lookup}\ (v)(\ \text{update}\ (w, n)(s)) \\ \equiv\ & \text{si}\ (v = w)\ \text{alors}\ n\ \text{sinon}\ \ \text{lookup}\ (v)(s) \end{aligned}$$

Supprimer l'état

Plutôt que de choisir "non currifié" ou "currifié", on oublie l'état...
... mais on ajoute des couleurs :

- ROUGE : peut utiliser et modifier l'état,
- ORANGE : peut utiliser l'état, mais pas le modifier,
- VERT : ne peut ni utiliser ni modifier l'état.



$$\begin{aligned} & \text{lookup}\ (v,\ \text{update}\ (w, n)) \\ \equiv\ & \text{si}\ (v = w)\ \text{alors}\ n\ \text{sinon}\ \ \text{lookup}\ (v) \end{aligned}$$

Retrouver l'état

Il est facile de remettre l'état...

Version non currifiée :

$$X \xrightarrow{f} Y \qquad X \times S \xrightarrow{f} Y \times S$$

$$X \xrightarrow{f} Y \qquad X \times S \xrightarrow{f} Y$$

$$X \xrightarrow{f} Y \qquad X \xrightarrow{f} Y$$

Version currifiée :

$$X \xrightarrow{f} Y \qquad X \xrightarrow{f} (Y \times S)^S$$

$$X \xrightarrow{f} Y \qquad X \xrightarrow{f} Y^S$$

$$X \xrightarrow{f} Y \qquad X \xrightarrow{f} Y$$

Quelques questions techniques

- Assurer la compatibilté : vert $\Rightarrow$ orange $\Rightarrow$ rouge
- Définir la composition.

$$\begin{tikzcd} X \arrow[r, "f"'] \arrow[rr, bend left, "g \circ f"] & Y \arrow[r, "g"'] & Z \end{tikzcd}$$

Version non currifiée :

$$\begin{tikzcd} X \times S \arrow[r, "f"'] \arrow[rr, bend left, "g \circ f"] & Y \times S \arrow[r, "g"'] & Z \times S \end{tikzcd}$$

Version currifiée :
utiliser la composition de Kleisli pour la monade $(- \times S)^S \ldots$

$$\begin{tikzcd} & (Y \times S)^S \arrow[r, "g*"] & (Z \times S)^S \\ X \arrow[ur, "f"'] \arrow[urr, bend left, "g \circ f"] & Y \arrow[ur, "g"'] & \end{tikzcd}$$

Pourquoi cacher l'état ?

En cachant l'état, on a un formalisme :

- plus simple,
- plus proche de la syntaxe,
- plus clair : S n'est pas de même nature que Z ou V.

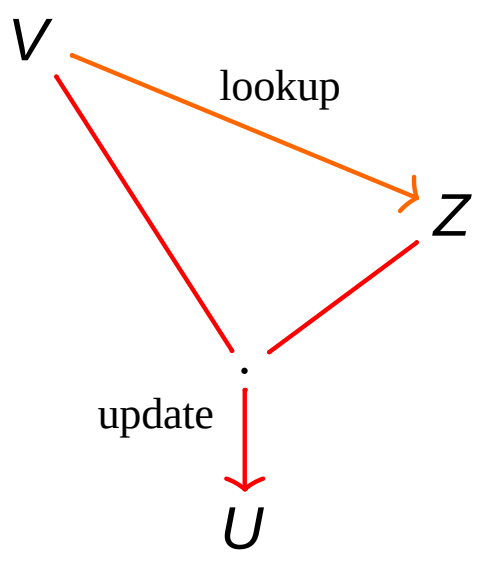
mais aussi :

- plus délicat à interpréter :
  il ne faut pas oublier les couleurs !

Cadre formel :

- Catégories et monades (cf. Haskell)
- Logique diagrammatique

La spécification de la mémoire



lookup $(v,$ update $(w,n))$

$\equiv$ si $(v=w)$ alors n

sinon lookup (v)

Notion de modèle colorié :

- chaque point représente un ensemble,
- chaque flèche rouge représente une fonction qui peut utiliser et modifier l'état,
- chaque flèche orange représente une fonction qui peut utiliser l'état mais pas le modifier,
- chaque flèche verte représente une fonction qui ne peut ni utiliser ni modifier l'état.

Outline

Introduction

La syntaxe de IMP

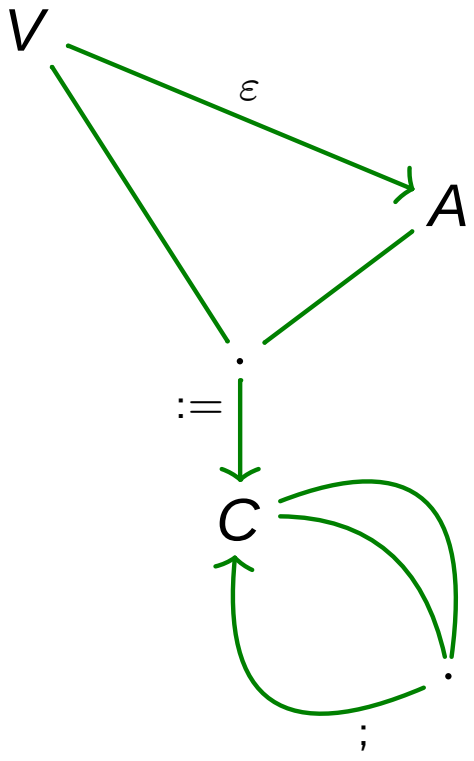
La mémoire

Des sémantiques pour IMP

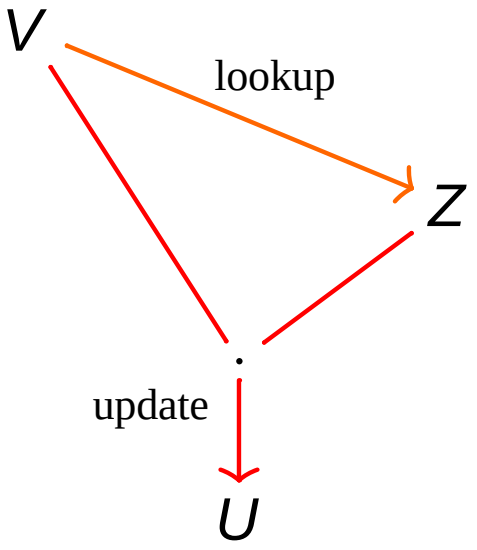
Conclusion

Rappel : les spécifications

Syntaxe

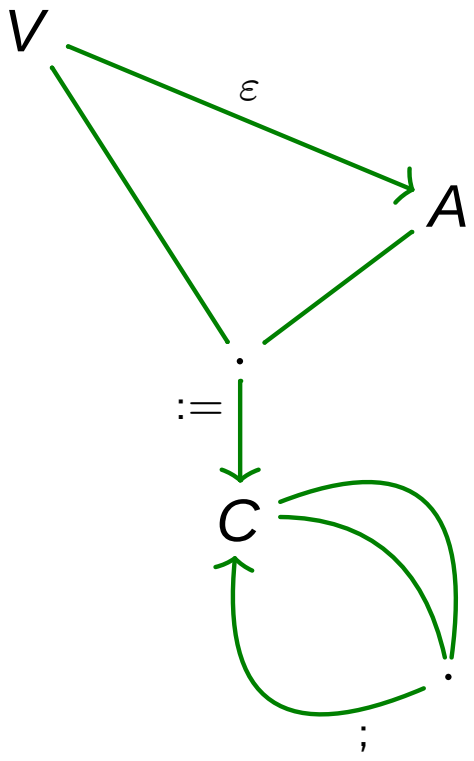


Mémoire

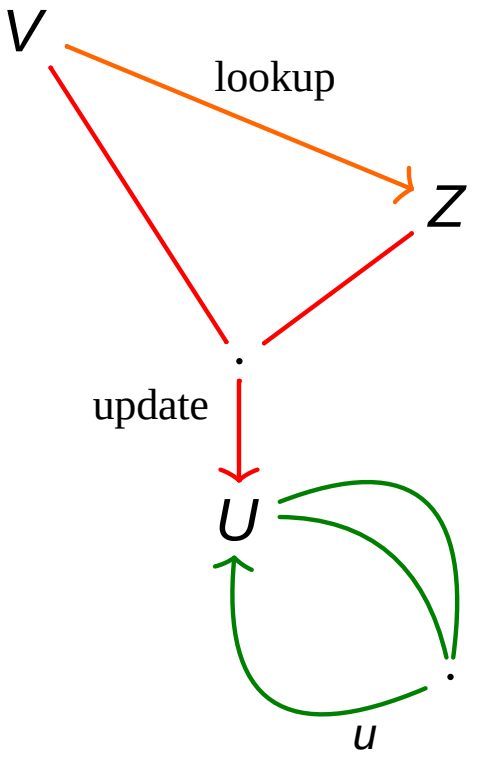


De plus, puisque U représente un singleton,
il existe une unique fonction $U^2 \xrightarrow{u} U$.

Syntaxe



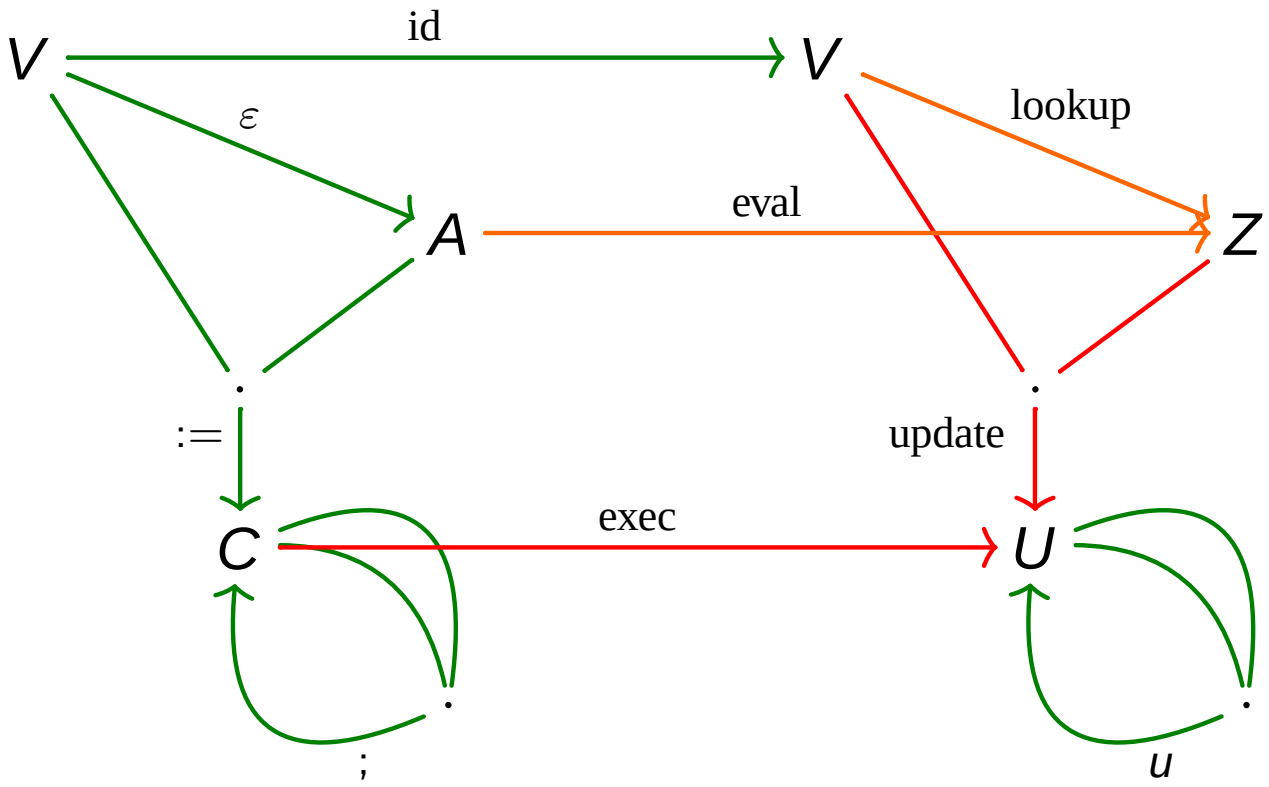
Mémoire



Sémantique coloriée

Chaque sémantique “traduit” la syntaxe dans la mémoire.

Syntaxe $\longrightarrow$ Mémoire



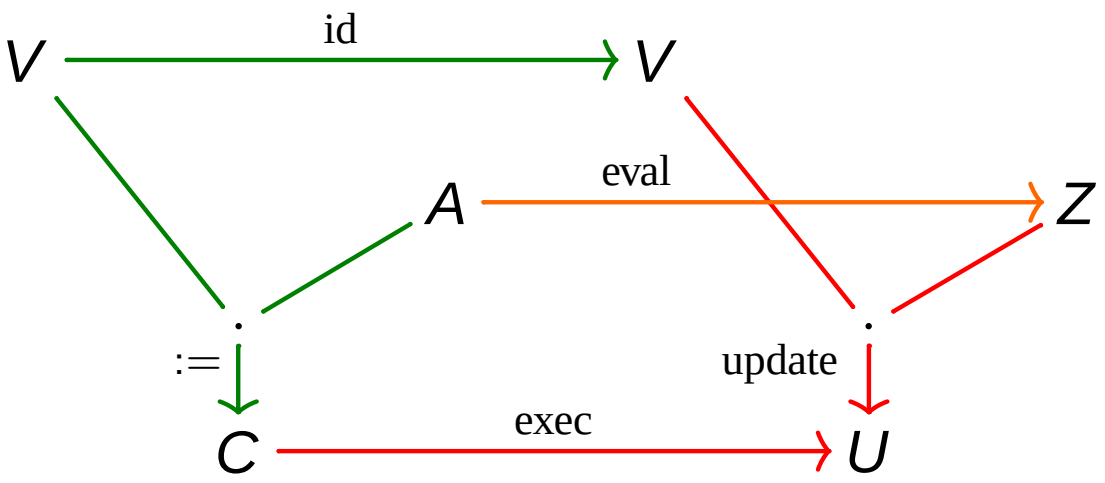
Conversion des variables

$$\begin{tikzcd} V \arrow[r, "\mathrm{id}"] \arrow[d, "\varepsilon"'] & V \arrow[d, "\mathrm{lookup}"] \\ A \arrow[r, "\mathrm{eval}"] & Z \end{tikzcd}$$

La conversion $V \xrightarrow{\varepsilon} A$ est traduite en $\mathrm{lookup} : V \to Z$, donc elle est :

- évaluée comme $\mathrm{lookup} : V \times S \to Z$
- interprétée par $\mathrm{lookup} : V \to Z^S$

Affectation



L'affectation $V \times A \overset{:=}{\to} C$ est traduite en $\text{update} : V \times Z \to U$, donc elle est :

- évaluée comme $\text{update} : V \times Z \times S \to S$
- interprétée par $\text{update} : V \times Z \to S^S$

Semi-produit de fonctions

On définit $\mathrm{id} \times \mathrm{eval}$ par :

$$\begin{tikzcd}
V \arrow[r, "\mathrm{id}"] & V \\
V \times A \arrow[u] \arrow[d] \arrow[r, "\mathrm{id} \times \mathrm{eval}"] \arrow[ur, phantom, "\sim"] \arrow[dr, phantom, "="] & V \times Z \arrow[u] \arrow[d] \\
A \arrow[r, "\mathrm{eval}"] & Z
\end{tikzcd}$$

C'est justifié par :

$$\begin{tikzcd}
V \arrow[r, "\mathrm{id}"] & V \\
V \times A \times S \arrow[u] \arrow[d] \arrow[r, "\mathrm{id} \times \mathrm{eval}"] \arrow[ur, phantom, "="] \arrow[dr, phantom, "="] & V \times Z \times S \arrow[u] \arrow[d] \\
A \times S \arrow[r, "\mathrm{eval}"] & Z \times S
\end{tikzcd}$$

Semi-congruence

On a utilisé parfois $\sim$ à la place de $=$.
Signification : $f \sim g$ ssi f et g retournent toujours la même valeur, mais ils peuvent modifier l'état de façon différente.

$$\begin{tikzcd} X \arrow[r, bend left, "f"] \arrow[r, "\sim" description] \arrow[r, bend right, "g"'] & Y \end{tikzcd}$$

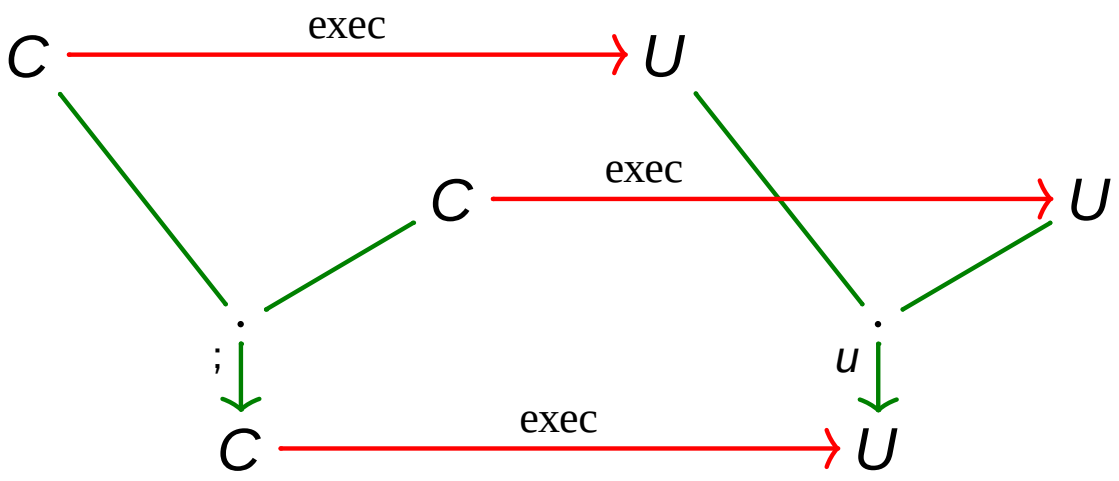
$$\begin{tikzcd} & Y \times S \arrow[dr, bend left, "\pi"] \arrow[dd, phantom, "="] & \\ X \times S \arrow[ur, bend left, "f"] \arrow[dr, bend right, "g"'] & & Y \\ & Y \times S \arrow[ur, bend right, "\pi"'] & \end{tikzcd}$$

C'est une relation d'équivalence, qui vérifie $f \sim g \Rightarrow f \circ a \sim g \circ a$ mais $f \sim g \not\Rightarrow h \circ f \sim h \circ g$ sauf si h est vert.

"Théorème" [Dumas-Duval-Reynaud 2007].
On peut faire des preuves en utilisant $\sim$ "à la place de" $=$.

Le problème du point-virgule

On serait tenté de dessiner :



Mais si on pouvait définir $\mathrm{exec} \times \mathrm{exec}$, qui est symétrique, cela signifierait “évaluer c_1 et c_2 en parallèle”.

“Corollaire” [Dumas-Duval-Reynaud 2007].
On peut définir un produit séquentiel $\mathrm{exec} \ltimes \mathrm{exec}$, qui signifie “évaluer c_1 puis évaluer c_2”.

Produit séquentiel de fonctions

On définit $\text{exec} \ltimes \text{exec} \;=\; (\,\text{exec} \times \text{id}\,) \circ (\,\text{id} \times \text{exec}\,)$:

$$\begin{tikzcd}
C \arrow[r, "\text{exec}"] \arrow[dr, phantom, "="] & U \arrow[r, "\text{id}"] \arrow[dr, phantom, "\sim"] & U \\
C^2 \arrow[u] \arrow[r, "\text{exec} \times \text{id}"] \arrow[d] \arrow[dr, phantom, "\sim"] & U \times C \arrow[u] \arrow[r, "\text{id} \times \text{exec}"] \arrow[d] \arrow[dr, phantom, "="] & U^2 \arrow[u] \arrow[d] \\
C \arrow[r, "\text{id}"] & C \arrow[r, "\text{exec}"] & U
\end{tikzcd}$$

C'est justifié par :

$$\begin{tikzcd}
C \times S \arrow[r, "\text{exec}"] \arrow[dr, phantom, "="] & U \times S \\
C^2 \times S \arrow[u] \arrow[r, "\text{exec} \times \text{id}"] \arrow[d] \arrow[dr, phantom, "="] & U \times C \times S \arrow[u] \arrow[d] \\
C \arrow[r, "\text{id}"] & C
\end{tikzcd}
\quad
\begin{tikzcd}
U \arrow[r, "\text{id}"] \arrow[dr, phantom, "="] & U \\
U \times C \times S \arrow[u] \arrow[r, "\text{id} \times \text{exec}"] \arrow[d] \arrow[dr, phantom, "="] & U^2 \times S \arrow[u] \arrow[d] \\
C \times S \arrow[r, "\text{exec}"] & U \times S
\end{tikzcd}$$

$$\xrightarrow{\text{id}}$$

Outline

Introduction

La syntaxe de IMP

La mémoire

Des sémantiques pour IMP

Conclusion

Logiques

Toute logique comporte un aspect “modèles”
et un aspect “preuves”.

[Duval-Lair 2002] fournit une définition “abstraite” de la notion de logique : les logiques diagrammatiques.

Cela permet de définir des logiques “exotiques” et de les traduire vers des logiques usuelles.

[Dumas-Duval-Reynaud 2007] applique cela pour définir le produit séquentiel, qui est fondamental en cas d’effets de bord en informatique.

Logiques

[Dumas-Duval-Fousse-Reynaud, *en cours*] applique cela à IMP et précise la traduction vers la sémantique opérationnelle et vers la sémantique dénotationnelle de IMP.

Langage	
Syntaxe	
Sémantique “coloriée”	
Machine	**Homme**
Sémantique opérationnelle	Sémantique dénotationnelle

Un beau discours pour terminer

... L'informatique est une discipline jeune, elle a à peine 50 ans. Elle ne dispose pas aujourd'hui de bases théoriques similaires à celles des autres disciplines scientifiques. On construit encore les systèmes complexes comme on bâtissait les cathédrales au Moyen Âge...
... Le grand défi qui se pose alors, est le développement d'une discipline des logiciels et des systèmes qui permette de mieux comprendre, maîtriser et construire le monde qui nous entoure...
... J'espère que le caractère fondamental de notre discipline sera enfin reconnu et le soutien apporté à la mesure de son importance ...

Extraits du discours de Joseph Sifakis
Médaille d'argent du CNRS, 13 décembre 2001