Lambda-calculus and programming language semantics

Thibaut Balabonski @ UPSay
Winter 2023
https://www.lri.fr/~blsk/LambdaCalculus/

Chapter 4: semantics of an imperative programming language

1 An imperative language: IMP

An imperative language: IMP

Core of an imperative language, with:

- arithmetic and boolean expressions
- mutable variables
- instructions (assignment, condition, loop)

Endorses the same rôle as PCF, for imperative programming

Aexp: arithmetic expressions

Integer constants: n, m — $\mathbb{N}$
Variables: X, Y — $\mathcal{V}$
Arithmetic expressions: a — Aexp

$$\begin{array}{rcl} a & ::= & n \\ & | & X \\ & | & a_1 + a_2 \\ & | & a_1 - a_2 \\ & | & a_1 \times a_2 \end{array}$$

Bexp: boolean expressions

Boolean constants: T, F — $\mathbb{B}$
Boolean expressions: b — Bexp

$$\begin{array}{rcl} b & ::= & \mathsf{T} \\ & | & \mathsf{F} \\ & | & a_1 = a_2 \\ & | & a_1 \leq a_2 \\ & | & \neg b \\ & | & b_1 \vee b_2 \\ & | & b_1 \wedge b_2 \end{array}$$

Com: commands

Commands (instructions): c — Com

$$\begin{array}{rcl} c & ::= & \mathsf{skip} \\ & | & X := a \\ & | & c_1 \mathrel{;} c_2 \\ & | & \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2 \\ & | & \mathsf{while}\ b\ \mathsf{do}\ c \end{array}$$

2 Big step operational semantics

Operational semantics

Effects of expressions and commands, depending on a *state* of the memory

States $\Sigma \quad = \quad \mathcal{V} \longrightarrow \mathbb{N}$

- functions from variables to numbers
- if $\sigma \in \Sigma$, then $\sigma(X)$ is the value of the variable X in the state σ

Note: variables only have numeric values here (no boolean variables)

Big step semantics: relation between

- expression or command
- state
- result

Semantics of arithmetic expressions

Evaluation relation

$$\langle a, \sigma \rangle \Downarrow n$$

Inference rules

$$\frac{}{\langle n, \sigma \rangle \Downarrow n} \qquad\qquad \frac{}{\langle X, \sigma \rangle \Downarrow \sigma(X)}$$

$$\frac{\langle a_1, \sigma \rangle \Downarrow n_1 \qquad \langle a_2, \sigma \rangle \Downarrow n_2 \qquad n_1 +_{\mathbb{N}} n_2 = n}{\langle a_1 + a_2, \sigma \rangle \Downarrow n}$$

Other binary operations similar

Note: the semantics being defined by a relation, some cases can be undefined Here, there are no σ, n such that $\langle 1 - 2, \sigma \rangle \Downarrow n$

Semantics of boolean expressions

Evaluation relation

$$\langle b, \sigma \rangle \Downarrow b$$

Inference rules

$$\frac{}{\langle \mathsf{T}, \sigma \rangle \Downarrow \mathsf{T}} \qquad\qquad \frac{}{\langle \mathsf{F}, \sigma \rangle \Downarrow \mathsf{F}}$$

$$\frac{\langle a_1, \sigma \rangle \Downarrow n_1 \qquad \langle a_2, \sigma \rangle \Downarrow n_2}{\langle a_1 \leq a_2, \sigma \rangle \Downarrow b}$$

where b is T if n_1 less than or equal to n_2 and F otherwise

Semantics of instructions

The relation

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

means that

- in state σ, the command c terminates
- after the execution we reach the state c'

At the beginning, we assume an initial state σ_0 such that

$$\forall X,\ \sigma_0(X) = 0$$

State evolution

Execution

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

σ' is the state such that

- $\sigma'(X)$ is $1 + \sigma(X)$
- for all $Y \neq X$, $\sigma'(Y) = \sigma(Y)$

Notation $\sigma\{X \leftarrow n\}$

$$\begin{array}{lcll} \sigma\{X \leftarrow n\}(X) & = & n & \\ \sigma\{X \leftarrow n\}(Y) & = & \sigma(Y) & \quad \text{si } Y \neq X \end{array}$$

Then

$$\langle X := X + 1, \sigma\rangle \Downarrow \sigma\{X \leftarrow \sigma(X) +_{\mathbb{N}} 1\}$$

Rules for instructions

Empty command

$$\frac{}{\langle \mathsf{skip}, \sigma\rangle \Downarrow \sigma}$$

Variable assignment

$$\frac{\langle a, \sigma\rangle \Downarrow n}{\langle X := a, \sigma\rangle \Downarrow \sigma\{X \leftarrow n\}}$$

Sequential composition

$$\frac{\langle c_1, \sigma\rangle \Downarrow \sigma'' \qquad \langle c_2, \sigma''\rangle \Downarrow \sigma'}{\langle c_1 \,;\, c_2, \sigma\rangle \Downarrow \sigma'}$$

Conditional instruction

$$\frac{\langle b, \sigma\rangle \Downarrow \mathsf{T} \qquad \langle c_1, \sigma\rangle \Downarrow \sigma'}{\langle \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2, \sigma\rangle \Downarrow \sigma'} \qquad \frac{\langle b, \sigma\rangle \Downarrow \mathsf{F} \qquad \langle c_2, \sigma\rangle \Downarrow \sigma'}{\langle \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2, \sigma\rangle \Downarrow \sigma'}$$

Rules for instructions: loop

When the condition is false, nothing happens

$$\frac{\langle b, \sigma\rangle \Downarrow \mathsf{F}}{\langle \mathsf{while}\ b\ \mathsf{do}\ c, \sigma\rangle \Downarrow \sigma}$$

When the condition is true, we execute the body of the loop, and then execute the whole loop again

$$\frac{\langle b, \sigma\rangle \Downarrow \mathsf{T} \qquad \langle c, \sigma\rangle \Downarrow \sigma'' \qquad \langle \mathsf{while}\ b\ \mathsf{do}\ c, \sigma''\rangle \Downarrow \sigma'}{\langle \mathsf{while}\ b\ \mathsf{do}\ c, \sigma\rangle \Downarrow \sigma'}$$

Derivation is possible when the execution is finite

3 Denotational semantics

Denotational semantics

We want to characterize the function realized by a program

We are interested in (computable) functions rather than algorithms

We consider as *equivalent* two programs defining the same mathematical function

Denotational semantics for IMP

Base functions

- $\mathcal{A}[\![a]\!] : \Sigma \longrightarrow \mathbb{N}$
- $\mathcal{B}[\![b]\!] : \Sigma \longrightarrow \mathbb{B}$
- $\mathcal{C}[\![c]\!] : \Sigma \rightharpoonup \Sigma$

In other words

- $\mathcal{A}[\![.]\!] : \mathsf{Aexp} \longrightarrow \Sigma \longrightarrow \mathbb{N}$
- $\mathcal{B}[\![.]\!] : \mathsf{Bexp} \longrightarrow \Sigma \longrightarrow \mathbb{B}$
- $\mathcal{C}[\![.]\!] : \mathsf{Com} \longrightarrow \Sigma \rightharpoonup \Sigma$

Denotation of Aexp's

$$\begin{array}{rcl} \mathcal{A}[\![n]\!]_\sigma &=& n \\ \mathcal{A}[\![X]\!]_\sigma &=& \sigma(X) \\ \mathcal{A}[\![a_1 + a_2]\!]_\sigma &=& \mathcal{A}[\![a_1]\!]_\sigma +_{\mathbb{N}} \mathcal{A}[\![a_2]\!]_\sigma \end{array}$$

In other words

$$\begin{array}{rcl} \mathcal{A}[\![n]\!] &=& \sigma \mapsto n \\ \mathcal{A}[\![X]\!] &=& \sigma \mapsto \sigma(X) \\ \mathcal{A}[\![a_1 + a_2]\!] &=& \sigma \mapsto \mathcal{A}[\![a_1]\!]_\sigma +_{\mathbb{N}} \mathcal{A}[\![a_2]\!]_\sigma \end{array}$$

Denotation of Bexp's

$$\begin{array}{rcl} \mathcal{B}[\![\mathsf{T}]\!] &=& \sigma \mapsto \mathsf{T} \\ \mathcal{B}[\![a_1 = a_2]\!] &=& \sigma \mapsto \mathcal{A}[\![a_1]\!]_\sigma =_{\mathbb{N}} \mathcal{A}[\![a_2]\!]_\sigma \\ \mathcal{B}[\![b_1 \wedge b_2]\!] &=& \sigma \mapsto \mathcal{B}[\![b_1]\!]_\sigma \wedge_{\mathbb{B}} \mathcal{B}[\![b_2]\!]_\sigma \end{array}$$

Denotation of instructions

$$\begin{array}{rcl} \mathcal{C}[\![\mathsf{skip}]\!] &=& \sigma \mapsto \sigma \\ \mathcal{C}[\![X := a]\!] &=& \sigma \mapsto \sigma\{X \longleftarrow \mathcal{A}[\![a]\!]_\sigma\} \\ \mathcal{C}[\![c_1 \,;\, c_2]\!] &=& \mathcal{C}[\![c_2]\!] \circ \mathcal{C}[\![c_1]\!] \\ \mathcal{C}[\![\mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2]\!]_\sigma &=& \left\{ \begin{array}{ll} \mathcal{C}[\![c_1]\!]_\sigma & \text{si } \mathcal{B}[\![b]\!]_\sigma \\ \mathcal{C}[\![c_2]\!]_\sigma & \text{si } \neg\mathcal{B}[\![b]\!]_\sigma \end{array} \right. \end{array}$$

Loop

We are looking for a denotation of

$$\mathsf{while}\ b\ \mathsf{do}\ c \qquad (=\ w)$$

Remark: $\mathcal{C}[\![w]\!]$ and $\mathcal{C}[\![c]\!]$ are partial functions $\Sigma \rightharpoonup \Sigma$ satisfying

$$\mathcal{C}[\![w]\!]_\sigma \quad = \quad \left\{ \begin{array}{ll} (\mathcal{C}[\![w]\!] \circ \mathcal{C}[\![c]\!])(\sigma) & \text{if } \mathcal{B}[\![b]\!]_\sigma \\ \sigma & \text{if } \neg\mathcal{B}[\![b]\!]_\sigma \end{array} \right.$$

We are looking for a *fixpoint*

4 Fixpoints

McCarthy's 91 function

Is this function actually defined?

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

- if $x > 100$, the result is $x - 10$
- if $x \leqslant 100$, is there any result?

A priori: partial function, $\mathbb{N} \rightharpoonup \mathbb{N}$

An order on functions

Definition order

$$f \sqsubseteq g$$

if

- if f defined on x then g defined on x
- for any x in the shared input domain, $f(x) = g(x)$

In other words:

$$g|_{\mathsf{dom}(f)} = f$$

Directed set

Set $E \subseteq \mathbb{N} \rightharpoonup \mathbb{N}$ such that if it contains two functions f and g, then it also contains a function h which:

- is more defined than f and than g
- coïncides with f and g

In other words:

$$\forall f, g \in E,\ \exists h \in E,\ f \sqsubseteq h \wedge g \sqsubseteq h$$

Note: all functions in E are mutually consistent

Continuity

A *majorant* of E is an element m such that

$$\forall x \in E,\ x \sqsubseteq m$$

The *supremum* of E is the smallest majorant, if it exists $\qquad \sup(E)$

Note: if E is a directed set, then sup(E) *exists*

A function $f : E \longrightarrow E$ is *continuous* if it preserves supremums

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

Exercise: a continuous function is monotone

Fixpoint

Consider a directed set E

- any subset of E has a supremum
- $\sup(\emptyset)$ is the smallest element of E $\qquad \bot$

Then any continuous function $f : E \longrightarrow E$ admits the following has a fixpoint

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

Function 91 defined as a fixpoint

Consider the function $F : (\mathbb{N} \rightharpoonup \mathbb{N}) \longrightarrow (\mathbb{N} \rightharpoonup \mathbb{N})$ defined by

$$F(f) \quad = \quad x \longmapsto \mathsf{if}\ x > 100\ \mathsf{then}\ x - 10\ \mathsf{else}\ f(f(x + 11))$$

F is continuous

- $f \longmapsto (x \longmapsto f(f(x + 11)))$ continuous with respect to f
- $f \longmapsto (x \longmapsto \mathsf{if}\ b(x)\ \mathsf{then}\ G(f)(x)\ \mathsf{else}\ H(f)(x))$ continuous with respect to f if G and H are

F has a fixpoint $\mathsf{Fix}(F)$ such that $F(\mathsf{Fix}(F)) = \mathsf{Fix}(F)$ This fixpoint of F can be defined starting from the partial function $\bot$ which is undefined on every possible input

The denotation of while b do c defined as a fixpoint

For any functions $g : \Sigma \rightharpoonup \Sigma$ and $h : \Sigma \longrightarrow \mathbb{B}$, consider the function $F_{g,h} : (\Sigma \rightharpoonup \Sigma) \longrightarrow (\Sigma \rightharpoonup \Sigma)$ defined by

$$F_{g,h}(f)(\sigma) \quad = \quad \begin{cases} (f \circ g)(\sigma) & \text{if } h(\sigma) \\ \sigma & \text{if } \neg h(\sigma) \end{cases}$$

$F_{g,h}$ is continuous for $\sqsubseteq$

$$\sup_{f \in E}(F_{g,h}(f))(\sigma) \quad = \quad \begin{cases} (\sup_{f \in E}(f) \circ g)(\sigma) & \text{if } h(\sigma) \\ \sigma & \text{otherwise} \end{cases}$$

$F_{g,h}$ has a fixpoint $\mathsf{Fix}(F_{g,h})$, which is a partial function $\Sigma \rightharpoonup \Sigma$

Analysis of $F_{g,h}$

Meaning of the iterates $F_{g,h}^k(\bot)$

- $F_{g,h}(\bot)$ is defined only on states σ such that $\neg h(\sigma)$, and then $F_{g,h}(\bot)(\sigma) = \sigma$
- $F_{g,h}^{n+1}(\bot)$ is defined on states σ such that
  - $g^i(\sigma)$ is defined for all $i \leqslant n + 1$
  - $h(g^i(\sigma))$ for $i \leqslant n$
  - $\neg h(g^{n+1}(\sigma))$

  and then $F_{g,h}^{n+1}(\bot)(\sigma) = g^{n+1}(\sigma)$

The fixpoint of $F_{g,h}$ is thus defined for all σ such that there is n with

- $g^i(\sigma)$ defined and $h(g^i(\sigma))$ for $i < n$
- $g^n(\sigma)$ defined and $\neg h(g^n(\sigma))$

We thus define

$$\mathcal{C}[\![\mathsf{while}\ b\ \mathsf{do}\ c]\!] \quad = \quad \mathsf{Fix}(F_{\mathcal{C}[\![c]\!],\mathcal{B}[\![b]\!]})$$

5 Soundness and completeness

Soundness of the operational semantics

Soundness: the values given by the operational semantics are correct with respect to the denotational semantics

Theorem

$$\text{If} \quad \langle c, \sigma \rangle \Downarrow \sigma' \quad \text{then} \quad \mathcal{C}[\![c]\!]_\sigma = \sigma'$$

Proof by induction on the derivation of $\langle c, \sigma \rangle \Downarrow \sigma'$, with lemmas on the semantics of the expressions

Completeness of the operational semantics

Completeness: the operational semantics allows the derivation of all the values specified by the denotational semantics

Theorem

$$\text{If} \quad \mathcal{C}[\![c]\!]_\sigma \quad \text{is defined and is equal to } \sigma' \text{ the one can derive} \quad \langle c, \sigma \rangle \Downarrow \sigma'$$

Proof by inducton on c, with lemmas on the semantics of expressions

6 Axiomatic semantics

Axiomatic semantics for IMP

Hoare triples

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

- A: precondition of c
- B: postcondition of partial correctness of c

Interpretation

If A is satisfied before execution of c and if the execution of c terminates, then B is satisfied after the execution of c

Rules for partial correctness

$$\frac{}{\vdash \{A\}\ \mathsf{skip}\ \{A\}} \qquad \frac{}{\vdash \{B\{X \leftarrow a\}\}\ X := a\ \{B\}} \qquad \frac{\vdash \{A\}\ c_1\ \{C\} \qquad \vdash \{C\}\ c_2\ \{B\}}{\vdash \{A\}\ c_1\ ;\ c_2\ \{B\}}$$

$$\frac{\vdash \{b \wedge A\}\ c_1\ \{B\} \qquad \vdash \{(\neg b) \wedge A\}\ c_2\ \{B\}}{\vdash \{A\}\ \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2\ \{B\}} \qquad \frac{\vdash \{b \wedge I\}\ c\ \{I\}}{\vdash \{I\}\ \mathsf{while}\ b\ \mathsf{do}\ c\ \{(\neg b) \wedge I\}}$$

$$\frac{A \implies A' \qquad \vdash \{A'\}\ c\ \{B'\} \qquad B' \implies B}{\vdash \{A\}\ c\ \{B\}}$$

Meaning of an assertion

Notation

- $\sigma \models A$: A is satisfied by the state σ

The triple $\{A\}\ c\ \{B\}$ then means

$$\forall \sigma \in \Sigma,\ (\sigma \models A \wedge \mathcal{C}[\![c]\!]_\sigma \text{ défini}) \implies \mathcal{C}[\![c]\!]_\sigma \models B$$

Simplification: $\mathcal{C}[\![c]\!]$ is extended as a total function returning the value $\perp$ where it should not be defined. We define $\perp \models A$ for all assertion A. Then $\{A\}\ c\ \{B\}$ means

$$\forall \sigma \in \Sigma,\ \sigma \models A \implies \mathcal{C}[\![c]\!]_\sigma \models B$$

Definition of $\sigma \models A$: a semantics for assertions

To keep the formalism compact, we use as assertions the boolean expressions of IMP

$$
\begin{array}{ll}
\bot \models A & \\
\sigma \models \mathsf{T} & \\
\sigma \models a_1 = a_2 & \text{if } \mathcal{A}[\![a_1]\!]_\sigma =_{\mathbb{N}} \mathcal{A}[\![a_2]\!]_\sigma \\
\sigma \models A \wedge B & \text{if } \sigma \models A \text{ and } \sigma \models B \\
\sigma \models \neg A & \text{if } \sigma \not\models A
\end{array}
$$

Extended assertions

We could add: quantifications and logical variables

- extend Aexp with special variables i
- extend Bexp with the assertions $\forall i.A$ and $\exists i.A$
- parameterize the semantics by a *valuation* $\rho : \mathcal{I} \longrightarrow \mathbb{N}$

$$\mathcal{A}[\![i]\!]_{\rho,\sigma} \quad = \quad \rho(i)$$

$$
\begin{array}{ll}
\sigma \models_\rho \forall i.A & \text{if } \sigma \models_{\rho\{i \leftarrow n\}} A \text{ for all } n \in \mathbb{N} \\
\sigma \models_\rho \exists i.A & \text{if } \sigma \models_{\rho\{i \leftarrow n\}} A \text{ for at least one } n \in \mathbb{N}
\end{array}
$$

Properties of the semantics of assertions

Some results

- $\mathcal{B}[\![b]\!]_\sigma = \mathsf{T}$ if and only if $\sigma \models_\rho b$
- $\mathcal{B}[\![b]\!]_\sigma = \mathsf{F}$ if and only if $\sigma \not\models_\rho b$
- $\mathcal{A}[\![a]\!]_{\rho\{i \leftarrow n\},\sigma} = \mathcal{A}[\![a\{i \leftarrow n\}]\!]_{\rho,\sigma}$

(by induction)

Validity of a Hoare triple

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

if and only if **for any valuation ρ and any state σ**

$$\sigma \models_\rho A \implies \mathcal{C}[\![c]\!]_\sigma \models_\rho B$$

7 Soundness of the axiomatic semantics

Soundness of the axiomatic semantics

Theorem

$$\text{If} \quad \vdash \{A\}\ c\ \{B\} \quad \text{then} \quad \models \{A\}\ c\ \{B\}$$

By induction on the derivation of $\vdash \{A\}\ c\ \{B\}$

Substitution lemmas

Arithmetic expressions

$$\mathcal{A}[\![a_1\{X \leftarrow a_2\}]\!]_{\rho,\sigma} \quad = \quad \mathcal{A}[\![a_1]\!]_{\rho,\sigma\{X \leftarrow \mathcal{A}[\![a_2]\!]_{\rho,\sigma}\}}$$

Boolean expressions

$$\sigma \vDash_\rho B\{X \leftarrow a\} \iff \sigma\{X \leftarrow \mathcal{A}[\![a]\!]_\sigma\} \vDash_\rho B$$

Proof of soundness $\vdash \{A\}\ c\ \{B\} \implies \vDash \{A\}\ c\ \{B\}$

By induction on the derivation of $\vdash \{A\}\ c\ \{B\}$

- Case $\vdash \{A\}$ skip $\{A\}$ Since $\mathcal{C}[\![\text{skip}]\!]_\sigma = \sigma$ we have $\sigma \vDash_\rho A \implies \mathcal{C}[\![\text{skip}]\!]_\sigma \vDash_\rho A$ and then $\vDash \{A\}$ skip $\{A\}$
- Case $\vdash \{B\{X \leftarrow a\}\}\ X := a\ \{B\}$ By substitution lemma we have $\sigma \vDash_\rho B\{X \leftarrow a\}$ if and only if $\sigma\{X \leftarrow \mathcal{A}[\![a]\!]_\sigma\} \vDash_\rho B$ Since $\sigma\{X \leftarrow \mathcal{A}[\![a]\!]_\sigma\} = \mathcal{C}[\![X := a]\!]_\sigma$ we deduce $\sigma \vDash_\rho B\{X \leftarrow a\} \implies \mathcal{C}[\![X := a]\!]_\sigma \vDash_\rho B$ and therefore $\vDash \{B\{X \leftarrow a\}\}\ X := a\ \{B\}$
- sequence
- conditional
- consequence
- Case $\vdash \{I\}$ while b do c' $\{I \wedge (\neg b)\}$ with $\vdash \{b \wedge I\}\ c'\ \{I\}$ Induction hypothesis: $\vDash \{b \wedge I\}\ c'\ \{I\}$

  We show the following by recurrence over n:

  $$P(n) \quad = \quad \forall \sigma \in \Sigma, \sigma \vDash_\rho I \implies F^n_{\mathcal{C}[\![c']\!],\mathcal{B}[\![b]\!]}(\bot)(\sigma) \vDash_\rho I \wedge (\neg b)$$

  We deduce $\sigma \vDash_\rho I \implies \mathcal{C}[\![\text{while } b \text{ do } c']\!] \vDash_\rho I \wedge (\neg b)$ and $\vDash \{I\}$ while b do c' $\{I \wedge (\neg b)\}$

Completeness of the axiomatic semantics

Theorem

$$\text{If} \quad \vDash \{A\}\ c\ \{B\} \quad \text{then} \quad \vdash \{A\}\ c\ \{B\}$$

The proof is based on the algorithm computing the weakest preconditions!