Lecture 5 – Denotational Semantics

AAA551: Programming Language Theory

Jihyeok Park

PLRG

2026 Spring

Recall

- Evaluation Contexts
- Lambda Calculus
  - α-Equivalence
  - β-Reduction
  - Call by-Value (CBV) vs. Call by-Name (CBN)
  - Substitution
  - De Bruijn Indices
- Definitional Translation
  - Church Encoding
  - Pairs and Let Bindings
  - Laziness
  - Adequacy

Contents

Semantics

Three approaches to **formal semantics**:

- **Operational** $\sigma \vdash e \Rightarrow v$
  - *How* is a program executed?
  - Useful for implementation of compilers and interpreters.

$$\frac{\vdash e_1 \Rightarrow n_1 \qquad \vdash e_2 \Rightarrow n_2}{\vdash e_1 \texttt{ + } e_2 \Rightarrow n_1 + n_2}$$

- **Denotational** $[\![e]\!]$
  - *What* is the mathematical object for a program?
  - Useful for theoretical foundations.

$$[\![e_1 \texttt{ + } e_2]\!] = [\![e_1]\!] + [\![e_2]\!]$$

- **Axiomatic** $\vdash \{\phi\}\ e\ \{\phi'\}$
  - *Which properties* does a program satisfy?
  - Useful for proving program properties and correctness.

$$\vdash \{x < n \land y < m\}\ z \texttt{ = } x \texttt{ + } y\ \{z < n + m\}$$

Contents

Simple Imperative Language – IMP

Let's redefine the semantics of IMP using denotational semantics.

$$
\begin{array}{lrl}
\text{Expressions} & e ::= & n \mid x \mid e \texttt{ + } e \mid e \texttt{ * } e \mid e \texttt{ < } e \mid \texttt{true} \mid \texttt{false} \\
\text{Statements} & s ::= & \texttt{skip} \\
 & \mid & x \texttt{ := } e \\
 & \mid & s \texttt{;} s \\
 & \mid & \texttt{if } e \texttt{ then } s \texttt{ else } s \\
 & \mid & \texttt{while } e \texttt{ do } s \\
\text{Values} & v ::= & n \mid \texttt{true} \mid \texttt{false}
\end{array}
$$

where $n \in \mathbb{Z}$, and $x \in \mathbb{X}$.

We will define two forms of denotational semantics for IMP:

$$\boxed{E[\![e]\!] : \Sigma \rightharpoonup \mathbb{V}} \qquad \boxed{S[\![s]\!] : \Sigma \rightharpoonup \Sigma}$$

where $\sigma : \mathbb{X} \rightharpoonup \mathbb{V}$.

Notational Conventions

There are two notational conventions for denotational semantics:

- Convention 1: Define $f : A \rightharpoonup B$ as sets of pairs:

$$S[\![\ldots]\!] = \{(\sigma, \sigma') \mid \ldots\}$$

- Convention 2: Define $f : A \rightharpoonup B$ point-wise:

$$S[\![\ldots]\!](\sigma) = \sigma' \text{ if } \ldots$$

We will use the second convention in most cases, but the first convention is often more intuitive and easier to read.

Expressions

$$\boxed{E[\![e]\!] : \Sigma \rightharpoonup \mathbb{V}}$$

It is a partial function because it is not defined for all environments:

- 1 `+ true` is not defined for any environment.
- x is defined only for environments that map x to some value.

Expressions

$$\boxed{E[\![e]\!] : \Sigma \rightharpoonup \mathbb{V}}$$

It is a partial function because it is not defined for all environments:

- 1 `+ true` is not defined for any environment.
- x is defined only for environments that map x to some value.

$$E[\![n]\!](\sigma) \quad \triangleq \quad n$$

Expressions

PLRG

$$\boxed{E[\![e]\!] : \Sigma \rightharpoonup \mathbb{V}}$$

It is a partial function because it is not defined for all environments:

- 1 `+` `true` is not defined for any environment.
- x is defined only for environments that map x to some value.

$$E[\![n]\!](\sigma) \triangleq n$$

$$E[\![x]\!](\sigma) \triangleq \sigma(x) \qquad \text{if } x \in \mathsf{dom}(\sigma)$$

Expressions

$$\boxed{E[\![e]\!] : \Sigma \rightharpoonup \mathbb{V}}$$

It is a partial function because it is not defined for all environments:

- 1 `+ true` is not defined for any environment.
- x is defined only for environments that map x to some value.

$$
\begin{array}{lcll}
E[\![n]\!](\sigma) & \triangleq & n & \\
E[\![x]\!](\sigma) & \triangleq & \sigma(x) & \text{if } x \in \mathsf{dom}(\sigma) \\
E[\![e_1 \texttt{ + } e_2]\!](\sigma) & \triangleq & E[\![e_1]\!](\sigma) + E[\![e_2]\!](\sigma) & \\
E[\![e_1 \texttt{ * } e_2]\!](\sigma) & \triangleq & E[\![e_1]\!](\sigma) \times E[\![e_2]\!](\sigma) &
\end{array}
$$

Expressions

$$E[\![e]\!] : \Sigma \rightharpoonup \mathbb{V}$$

It is a partial function because it is not defined for all environments:

- 1 `+` `true` is not defined for any environment.
- x is defined only for environments that map x to some value.

$$E[\![n]\!](\sigma) \triangleq n$$

$$E[\![x]\!](\sigma) \triangleq \sigma(x) \qquad \text{if } x \in \mathsf{dom}(\sigma)$$

$$E[\![e_1 \texttt{ + } e_2]\!](\sigma) \triangleq E[\![e_1]\!](\sigma) + E[\![e_2]\!](\sigma)$$

$$E[\![e_1 \texttt{ * } e_2]\!](\sigma) \triangleq E[\![e_1]\!](\sigma) \times E[\![e_2]\!](\sigma)$$

$$E[\![e_1 \texttt{ < } e_2]\!](\sigma) \triangleq \begin{cases} \texttt{true} & \text{if } E[\![e_1]\!](\sigma) < E[\![e_2]\!](\sigma) \\ \texttt{false} & \text{if } E[\![e_1]\!](\sigma) \geq E[\![e_2]\!](\sigma) \\ \text{undefined} & \text{otherwise} \end{cases}$$

Expressions

$$\boxed{E[\![e]\!] : \Sigma \rightharpoonup \mathbb{V}}$$

It is a partial function because it is not defined for all environments:

- 1 `+` `true` is not defined for any environment.
- x is defined only for environments that map x to some value.

$$
\begin{array}{lcll}
E[\![n]\!](\sigma) & \triangleq & n & \\
E[\![x]\!](\sigma) & \triangleq & \sigma(x) & \text{if } x \in \text{dom}(\sigma) \\
E[\![e_1 \texttt{ + } e_2]\!](\sigma) & \triangleq & E[\![e_1]\!](\sigma) + E[\![e_2]\!](\sigma) & \\
E[\![e_1 \texttt{ * } e_2]\!](\sigma) & \triangleq & E[\![e_1]\!](\sigma) \times E[\![e_2]\!](\sigma) & \\
E[\![e_1 \texttt{ < } e_2]\!](\sigma) & \triangleq & \begin{cases} \texttt{true} & \text{if } E[\![e_1]\!](\sigma) < E[\![e_2]\!](\sigma) \\ \texttt{false} & \text{if } E[\![e_1]\!](\sigma) \geq E[\![e_2]\!](\sigma) \\ \text{undefined} & \text{otherwise} \end{cases} & \\
E[\![\texttt{true}]\!](\sigma) & \triangleq & \texttt{true} & \\
E[\![\texttt{false}]\!](\sigma) & \triangleq & \texttt{false} &
\end{array}
$$

Statements

$$\boxed{S[\![s]\!] : \Sigma \rightharpoonup \Sigma}$$

It is a partial function for a given environment σ when:

- the statement does not terminate, or
- the statement throws an error in expressions

Statements

$$\boxed{S[\![s]\!] : \Sigma \rightharpoonup \Sigma}$$

It is a partial function for a given environment σ when:

- the statement does not terminate, or
- the statement throws an error in expressions

$$S[\![\texttt{skip}]\!](\sigma) \quad \triangleq \quad \sigma$$

Statements

PLRG

$$\boxed{S[\![s]\!] : \Sigma \rightharpoonup \Sigma}$$

It is a partial function for a given environment σ when:

- the statement does not terminate, or
- the statement throws an error in expressions

$$\begin{array}{lcll} S[\![\texttt{skip}]\!](\sigma) & \triangleq & \sigma & \\ S[\![x \texttt{ := } e]\!](\sigma) & \triangleq & \sigma[x \mapsto v] & \text{if } E[\![e]\!](\sigma) = v \end{array}$$

Statements

$$\boxed{S[\![s]\!] : \Sigma \rightharpoonup \Sigma}$$

It is a partial function for a given environment σ when:

- the statement does not terminate, or
- the statement throws an error in expressions

$$\begin{array}{lcll} S[\![\texttt{skip}]\!](\sigma) & \triangleq & \sigma & \\ S[\![x \texttt{ := } e]\!](\sigma) & \triangleq & \sigma[x \mapsto v] & \text{if } E[\![e]\!](\sigma) = v \\ S[\![s_1 \, ; s_2]\!] & \triangleq & S[\![s_2]\!] \circ S[\![s_1]\!] & \end{array}$$

Statements

$$\boxed{S[\![s]\!] : \Sigma \rightharpoonup \Sigma}$$

It is a partial function for a given environment σ when:

- the statement does not terminate, or
- the statement throws an error in expressions

$$
\begin{array}{lcll}
S[\![\texttt{skip}]\!](\sigma) & \triangleq & \sigma & \\
S[\![x\ \texttt{:=}\ e]\!](\sigma) & \triangleq & \sigma[x \mapsto v] & \text{if } E[\![e]\!](\sigma) = v \\
S[\![s_1\,\texttt{;}\,s_2]\!] & \triangleq & S[\![s_2]\!] \circ S[\![s_1]\!] &
\end{array}
$$

$$
S[\![\texttt{if}\ e\ \texttt{then}\ s_1\ \texttt{else}\ s_2]\!](\sigma) \triangleq \begin{cases} S[\![s_1]\!](\sigma) & \text{if } E[\![e]\!](\sigma) = \texttt{true} \\ S[\![s_2]\!](\sigma) & \text{if } E[\![e]\!](\sigma) = \texttt{false} \\ \text{undefined} & \text{otherwise} \end{cases}
$$

`while` Statement

How do we define the semantics of `while` statement?

$$S[\![\texttt{while}\ e\ \texttt{do}\ s]\!](\sigma) \triangleq$$

`while` Statement

How do we define the semantics of `while` statement?

$$S[\![\texttt{while}\ e\ \texttt{do}\ s]\!](\sigma) \triangleq \begin{cases} \sigma & \text{if } E[\![e]\!](\sigma) = \texttt{false} \\ S[\![\texttt{while}\ e\ \texttt{do}\ s]\!](S[\![s]\!](\sigma)) & \text{if } E[\![e]\!](\sigma) = \texttt{true} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Is it a valid definition?

while Statement

How do we define the semantics of `while` statement?

$$S[\![\texttt{while}\ e\ \texttt{do}\ s]\!](\sigma) \triangleq \begin{cases} \sigma & \text{if } E[\![e]\!](\sigma) = \texttt{false} \\ S[\![\texttt{while}\ e\ \texttt{do}\ s]\!](S[\![s]\!](\sigma)) & \text{if } E[\![e]\!](\sigma) = \texttt{true} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Is it a valid definition? **No**, because it is **recursive equation** not a definition.

`while` Statement

How do we define the semantics of `while` statement?

$$S[\![\texttt{while}\ e\ \texttt{do}\ s]\!](\sigma) \triangleq \begin{cases} \sigma & \text{if } E[\![e]\!](\sigma) = \texttt{false} \\ S[\![\texttt{while}\ e\ \texttt{do}\ s]\!](S[\![s]\!](\sigma)) & \text{if } E[\![e]\!](\sigma) = \texttt{true} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Is it a valid definition? **No**, because it is **recursive equation** not a definition.

We need to prove that:

- there exists a solution for the equation, and
- the solution is unique.

`while` Statement

How do we define the semantics of `while` statement?

$$S[\![\texttt{while}\ e\ \texttt{do}\ s]\!](\sigma) \triangleq \begin{cases} \sigma & \text{if } E[\![e]\!](\sigma) = \texttt{false} \\ S[\![\texttt{while}\ e\ \texttt{do}\ s]\!](S[\![s]\!](\sigma)) & \text{if } E[\![e]\!](\sigma) = \texttt{true} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Is it a valid definition? **No**, because it is **recursive equation** not a definition.

We need to prove that:

- there exists a solution for the equation, and
- the solution is unique.

Let's try to find a solution for the equation using **fixed-point theory**.

Solving Recursive Equations

Solve the following recursive equations on natural number functions:

$$
\begin{aligned}
f(x) &= \begin{cases} 0 & \text{if } x = 0 \\ f(x-1) + 2x - 1 & \text{otherwise} \end{cases} \\
f(x) &=
\end{aligned}
$$

Solving Recursive Equations

Solve the following recursive equations on natural number functions:

$$
\begin{aligned}
f(x) &= \begin{cases} 0 & \text{if } x = 0 \\ f(x-1) + 2x - 1 & \text{otherwise} \end{cases} \\
f(x) &= x^2
\end{aligned}
$$

Solving Recursive Equations

Solve the following recursive equations on natural number functions:

$$
\begin{aligned}
f(x) &= \begin{cases} 0 & \text{if } x = 0 \\ f(x-1) + 2x - 1 & \text{otherwise} \end{cases} \\
f(x) &= x^2
\end{aligned}
$$

$$
\begin{aligned}
g(x) &= g(x) + 1 \\
g(x) &=
\end{aligned}
$$

Solving Recursive Equations

PLRG

Solve the following recursive equations on natural number functions:

$$
\begin{aligned}
f(x) &= \begin{cases} 0 & \text{if } x = 0 \\ f(x-1) + 2x - 1 & \text{otherwise} \end{cases} \\
f(x) &= x^2 \\
\\
g(x) &= g(x) + 1 \\
g(x) &= \text{(no solution)}
\end{aligned}
$$

Solving Recursive Equations

Solve the following recursive equations on natural number functions:

$$
\begin{aligned}
f(x) &= \begin{cases} 0 & \text{if } x = 0 \\ f(x-1) + 2x - 1 & \text{otherwise} \end{cases} \\
f(x) &= x^2 \\
\\
g(x) &= g(x) + 1 \\
g(x) &= \text{(no solution)} \\
\\
h(x) &= 2 \cdot h(x-1) \\
h(x) &=
\end{aligned}
$$

Solving Recursive Equations

Solve the following recursive equations on natural number functions:

$$
\begin{aligned}
f(x) &= \begin{cases} 0 & \text{if } x = 0 \\ f(x-1) + 2x - 1 & \text{otherwise} \end{cases} \\
f(x) &= x^2 \\
\\
g(x) &= g(x) + 1 \\
g(x) &= \text{(no solution)} \\
\\
h(x) &= 2 \cdot h(x-1) \\
h(x) &= k \cdot 2^x \text{ for some } k \qquad \text{(infinitely many solutions)}
\end{aligned}
$$

Solving Recursive Equations

Solve the following recursive equations on natural number functions:

$$
\begin{aligned}
f(x) &= \begin{cases} 0 & \text{if } x = 0 \\ f(x-1) + 2x - 1 & \text{otherwise} \end{cases} \\
f(x) &= x^2 \\
\\
g(x) &= g(x) + 1 \\
g(x) &= \text{(no solution)} \\
\\
h(x) &= 2 \cdot h(x-1) \\
h(x) &= k \cdot 2^x \text{ for some } k \qquad \text{(infinitely many solutions)}
\end{aligned}
$$

Only the first equation has a unique solution, which is $f(x) = x^2$.

Solving Recursive Equations

Solve the following recursive equations on natural number functions:

$$
\begin{aligned}
f(x) &= \begin{cases} 0 & \text{if } x = 0 \\ f(x-1) + 2x - 1 & \text{otherwise} \end{cases} \\
f(x) &= x^2 \\[1em]
g(x) &= g(x) + 1 \\
g(x) &= \text{(no solution)} \\[1em]
h(x) &= 2 \cdot h(x-1) \\
h(x) &= k \cdot 2^x \text{ for some } k \qquad \text{(infinitely many solutions)}
\end{aligned}
$$

Only the first equation has a unique solution, which is $f(x) = x^2$.

How do we find the solution for the first equation?

Solving Recursive Equations

$$f(x) = \begin{cases} 0 & \text{if } x = 0 \\ f(x-1) + 2x - 1 & \text{otherwise} \end{cases}$$

Consider the following sequence of partial functions:

$$f_0 \triangleq \varnothing$$

Solving Recursive Equations

$$f(x) = \begin{cases} 0 & \text{if } x = 0 \\ f(x-1) + 2x - 1 & \text{otherwise} \end{cases}$$

Consider the following sequence of partial functions:

$$\begin{aligned} f_0 &\triangleq \varnothing \\ f_1 &\triangleq \{(0, 0)\} \end{aligned}$$

Solving Recursive Equations

$$f(x) = \begin{cases} 0 & \text{if } x = 0 \\ f(x-1) + 2x - 1 & \text{otherwise} \end{cases}$$

Consider the following sequence of partial functions:

$$\begin{aligned} f_0 &\triangleq \varnothing \\ f_1 &\triangleq \{(0,0)\} \\ f_2(x) &\triangleq \begin{cases} 0 & \text{if } x = 0 \\ f_1(x-1) + 2x - 1 & \text{if } x - 1 \in \mathsf{dom}(f_1) \end{cases} \\ &= \{(0,0), (1,1)\} \end{aligned}$$

Solving Recursive Equations

$$f(x) = \begin{cases} 0 & \text{if } x = 0 \\ f(x-1) + 2x - 1 & \text{otherwise} \end{cases}$$

Consider the following sequence of partial functions:

$$\begin{aligned}
f_0 &\triangleq \varnothing \\
f_1 &\triangleq \{(0,0)\} \\
f_2(x) &\triangleq \begin{cases} 0 & \text{if } x = 0 \\ f_1(x-1) + 2x - 1 & \text{if } x - 1 \in \mathsf{dom}(f_1) \end{cases} \\
&= \{(0,0), (1,1)\} \\
&\vdots \\
f_n(x) &\triangleq \begin{cases} 0 & \text{if } x = 0 \\ f_{n-1}(x-1) + 2x - 1 & \text{if } x - 1 \in \mathsf{dom}(f_{n-1}) \end{cases} \\
&= \{(0,0), (1,1), (2,4), \ldots, (n-1, (n-1)^2)\}
\end{aligned}$$

Solving Recursive Equations

$$f(x) = \begin{cases} 0 & \text{if } x = 0 \\ f(x-1) + 2x - 1 & \text{otherwise} \end{cases}$$

Consider the following sequence of partial functions:

$$\begin{aligned}
f_0 &\triangleq \varnothing \\
f_1 &\triangleq \{(0,0)\} \\
f_2(x) &\triangleq \begin{cases} 0 & \text{if } x = 0 \\ f_1(x-1) + 2x - 1 & \text{if } x - 1 \in \mathsf{dom}(f_1) \end{cases} \\
&= \{(0,0), (1,1)\} \\
&\vdots \\
f_n(x) &\triangleq \begin{cases} 0 & \text{if } x = 0 \\ f_{n-1}(x-1) + 2x - 1 & \text{if } x - 1 \in \mathsf{dom}(f_{n-1}) \end{cases} \\
&= \{(0,0), (1,1), (2,4), \ldots, (n-1, (n-1)^2)\}
\end{aligned}$$

If we continue this process, we can find the solution for the equation.

Solving Recursive Equations

We can redefine this sequence of partial functions using the n-th application of the transfer function $F : (\mathbb{N} \rightharpoonup \mathbb{N}) \to (\mathbb{N} \rightharpoonup \mathbb{N})$:

$$\forall n \in \mathbb{N}.\ f_n = F^n(\varnothing)$$

where $F(g)(x) \triangleq \begin{cases} 0 & \text{if } x = 0 \\ g(x-1) + 2x - 1 & \text{if } x - 1 \in \mathsf{dom}(g) \end{cases}$

Solving Recursive Equations

We can redefine this sequence of partial functions using the n-th application of the transfer function $F : (\mathbb{N} \rightharpoonup \mathbb{N}) \to (\mathbb{N} \rightharpoonup \mathbb{N})$:

$$\forall n \in \mathbb{N}.\ f_n = F^n(\varnothing)$$

where $F(g)(x) \triangleq \begin{cases} 0 & \text{if } x = 0 \\ g(x-1) + 2x - 1 & \text{if } x - 1 \in \mathsf{dom}(g) \end{cases}$

We can define the solution as follows:

$$\begin{aligned} f &= f_0 \cup f_1 \cup f_2 \cup \ldots \\ &= F^0(\varnothing) \cup F^1(\varnothing) \cup F^2(\varnothing) \cup \ldots \\ &= \textstyle\bigcup_{n \geq 0} F^n(\varnothing) \end{aligned}$$

Fixed-Points

Definition (Fixed-Points)

An element $x \in \text{dom}(F)$ is a **fixed-point** of F if $F(x) = x$.

Fixed-Points

PLRG

Definition (Fixed-Points)

An element $x \in \text{dom}(F)$ is a **fixed-point** of F if $F(x) = x$.

The following is a **fixed-point** of F:

$$f = \bigcup_{n \geq 0} F^n(\varnothing)$$

where $F(g)(x) \triangleq \begin{cases} 0 & \text{if } x = 0 \\ g(x-1) + 2x - 1 & \text{if } x - 1 \in \text{dom}(g) \end{cases}$

because $F(f) = F(\bigcup_{n \geq 0} F^n(\varnothing)) = \bigcup_{n \geq 0} F^{n+1}(\varnothing) = f$.

Fixed-Points

PLRG

Definition (Fixed-Points)

An element $x \in \mathsf{dom}(F)$ is a **fixed-point** of F if $F(x) = x$.

The following is a **fixed-point** of F:

$$f = \bigcup_{n \geq 0} F^n(\varnothing)$$

where $F(g)(x) \triangleq \begin{cases} 0 & \text{if } x = 0 \\ g(x-1) + 2x - 1 & \text{if } x - 1 \in \mathsf{dom}(g) \end{cases}$
because $F(f) = F(\bigcup_{n \geq 0} F^n(\varnothing)) = \bigcup_{n \geq 0} F^{n+1}(\varnothing) = f$.

We can generalize this idea to solve the recursive equation by finding the **least fixed-point** of the transfer function F.

$$\mathbf{lfp}(F) \triangleq \bigcup_{n \geq 0} F^n(\varnothing)$$

`while` Statement Revisited

Similarly, let's define the semantics of `while` statement using the least fixed-point of the transfer function $F : (\Sigma \rightharpoonup \Sigma) \to (\Sigma \rightharpoonup \Sigma)$:

$$S[\![\texttt{while } e \texttt{ do } s]\!] \triangleq \mathbf{lfp}(F) = \bigcup_{n \geq 0} F^n(\varnothing)$$

where

$$F(f)(\sigma) = \begin{cases} \sigma & \text{if } E[\![e]\!](\sigma) = \texttt{false} \\ f(S[\![s]\!](\sigma)) & \text{if } E[\![e]\!](\sigma) = \texttt{true} \\ \text{undefined} & \text{otherwise} \end{cases}$$

`while` Statement Revisited

Similarly, let's define the semantics of `while` statement using the least fixed-point of the transfer function $F : (\Sigma \rightharpoonup \Sigma) \to (\Sigma \rightharpoonup \Sigma)$:

$$S[\![\texttt{while } e \texttt{ do } s]\!] \triangleq \mathbf{lfp}(F) = \bigcup_{n \geq 0} F^n(\varnothing)$$

where

$$F(f)(\sigma) = \begin{cases} \sigma & \text{if } E[\![e]\!](\sigma) = \texttt{false} \\ f(S[\![s]\!](\sigma)) & \text{if } E[\![e]\!](\sigma) = \texttt{true} \\ \text{undefined} & \text{otherwise} \end{cases}$$

We will learn why it is the least fixed-point in the next lecture with the **fixed-point theory**.

Explicit Errors

$$\boxed{\mathcal{E}[\![e]\!] : \Sigma \rightharpoonup \mathbb{V}} \qquad \boxed{\mathcal{S}[\![s]\!] : \Sigma \rightharpoonup \Sigma}$$

The statement semantics is a partial function and not defined if:

- the statement does not terminate, or
- the statement throws an error in expressions

Explicit Errors

PLRG

$$\boxed{\mathbb{E}[\![e]\!] : \Sigma \rightharpoonup \mathbb{V}} \qquad \boxed{\mathbb{S}[\![s]\!] : \Sigma \rightharpoonup \Sigma}$$

The statement semantics is a partial function and not defined if:

- the statement does not terminate, or
- the statement throws an error in expressions

To explicitly distinguish between error and non-termination, we can redefine the semantics with the error value $\perp$:

$$\boxed{\mathbb{E}[\![e]\!] : \Sigma_\perp \to \mathbb{V}_\perp} \qquad \boxed{\mathbb{S}[\![s]\!] : \Sigma_\perp \rightharpoonup \Sigma_\perp}$$

where $X_\perp \triangleq X \cup \{\perp\}$ for any set X.

Explicit Errors

PLRG

$$\boxed{E[\![e]\!] : \Sigma \rightharpoonup \mathbb{V}} \qquad \boxed{S[\![s]\!] : \Sigma \rightharpoonup \Sigma}$$

The statement semantics is a partial function and not defined if:

- the statement does not terminate, or
- the statement throws an error in expressions

To explicitly distinguish between error and non-termination, we can redefine the semantics with the error value $\perp$:

$$\boxed{E[\![e]\!] : \Sigma_\perp \to \mathbb{V}_\perp} \qquad \boxed{S[\![s]\!] : \Sigma_\perp \rightharpoonup \Sigma_\perp}$$

where $X_\perp \triangleq X \cup \{\perp\}$ for any set X.

The expression semantics is total but the statement semantics is partial; if it throws an error, it returns $\perp$; if it does not terminate, it is undefined.

Explicit Errors

$$\boxed{E[\![e]\!] : \Sigma \rightharpoonup \mathbb{V}} \qquad \boxed{S[\![s]\!] : \Sigma \rightharpoonup \Sigma}$$

The statement semantics is a partial function and not defined if:

- the statement does not terminate, or
- the statement throws an error in expressions

To explicitly distinguish between error and non-termination, we can redefine the semantics with the error value $\perp$:

$$\boxed{E[\![e]\!] : \Sigma_\perp \to \mathbb{V}_\perp} \qquad \boxed{S[\![s]\!] : \Sigma_\perp \rightharpoonup \Sigma_\perp}$$

where $X_\perp \triangleq X \cup \{\perp\}$ for any set X.

The expression semantics is total but the statement semantics is partial; if it throws an error, it returns $\perp$; if it does not terminate, it is undefined.

The following represents the error propagation:

$$\forall e.\ E[\![e]\!](\perp) = \perp \qquad \text{and} \qquad \forall s.\ S[\![s]\!](\perp) = \perp$$

Explicit Errors

$$\boxed{E[\![e]\!] : \Sigma_\perp \to \mathbb{V}_\perp}$$

$$E[\![x]\!](\sigma) \triangleq \begin{cases} \sigma(x) & \text{if } x \in \mathsf{dom}(\sigma) \\ \perp & \text{otherwise} \end{cases}$$

$$E[\![e_1 \texttt{ + } e_2]\!](\sigma) \triangleq \begin{cases} n_1 + n_2 & \text{if } E[\![e_1]\!](\sigma) = n_1 \in \mathbb{Z} \wedge E[\![e_2]\!](\sigma) = n_2 \in \mathbb{Z} \\ \perp & \text{otherwise} \end{cases}$$

...

Explicit Errors

PLRG

$$\boxed{E[\![e]\!] : \Sigma_\perp \to \mathbb{V}_\perp}$$

$$E[\![x]\!](\sigma) \triangleq \begin{cases} \sigma(x) & \text{if } x \in \mathsf{dom}(\sigma) \\ \perp & \text{otherwise} \end{cases}$$

$$E[\![e_1 \texttt{ + } e_2]\!](\sigma) \triangleq \begin{cases} n_1 + n_2 & \text{if } E[\![e_1]\!](\sigma) = n_1 \in \mathbb{Z} \land E[\![e_2]\!](\sigma) = n_2 \in \mathbb{Z} \\ \perp & \text{otherwise} \end{cases}$$

...

$$\boxed{S[\![s]\!] : \Sigma_\perp \rightharpoonup \Sigma_\perp}$$

$$S[\![x \texttt{ := } e]\!](\sigma) \triangleq \begin{cases} \perp & \text{if } E[\![e]\!](\sigma) = \perp \\ \sigma[x \mapsto E[\![e]\!](\sigma)] & \text{otherwise} \end{cases}$$

$$S[\![\texttt{if } e \texttt{ then } s_1 \texttt{ else } s_2]\!](\sigma) \triangleq \begin{cases} S[\![s_1]\!](\sigma) & \text{if } E[\![e]\!](\sigma) = \texttt{true} \\ S[\![s_2]\!](\sigma) & \text{if } E[\![e]\!](\sigma) = \texttt{false} \\ \perp & \text{otherwise} \end{cases}$$

...

Contents

Non-Deterministic Imperative Language – NIMP

Let's define the semantics of NIMP, which is a non-deterministic extension of IMP with the following syntax:

$$
\begin{array}{lrl}
\text{Expressions} & e & ::= \ldots \mid [c_0, c_1] \\
\text{Statements} & s & ::= \ldots \\
\text{Values} & v & ::= \ldots
\end{array}
$$

where $n \in \mathbb{Z}$, $c_0 \in \mathbb{Z} \cup \{-\infty\}$, $c_1 \in \mathbb{Z} \cup \{+\infty\}$, and $x \in \mathbb{X}$.

The expression $[c_0, c_1]$ non-deterministically returns an integer between c_0 and c_1 (inclusive).

We will define two forms of denotational semantics for NIMP:

$$\boxed{E[\![e]\!] : \Sigma \to \mathcal{P}(\mathbb{V})} \qquad \boxed{S[\![s]\!] \in \mathcal{P}(\Sigma) \to \mathcal{P}(\Sigma)}$$

where $\sigma : \mathbb{X} \rightharpoonup \mathbb{V}$.

Expressions

$$\boxed{E[\![e]\!] : \Sigma \rightarrow \mathcal{P}(\mathbb{V})}$$

$$E[\![n]\!](\sigma) \quad \triangleq \quad \{n\}$$

Expressions

$$\boxed{E[\![e]\!] : \Sigma \to \mathcal{P}(\mathbb{V})}$$

$$E[\![n]\!](\sigma) \triangleq \{n\}$$

$$E[\![x]\!](\sigma) \triangleq \begin{cases} \{\sigma(x)\} & \text{if } x \in \text{dom}(\sigma) \\ \varnothing & \text{otherwise} \end{cases}$$

Expressions

$$\boxed{E[\![e]\!] : \Sigma \to \mathcal{P}(\mathbb{V})}$$

$$
\begin{array}{lcl}
E[\![n]\!](\sigma) & \triangleq & \{n\} \\
E[\![x]\!](\sigma) & \triangleq & \begin{cases} \{\sigma(x)\} & \text{if } x \in \mathsf{dom}(\sigma) \\ \varnothing & \text{otherwise} \end{cases} \\
E[\![[c_0, c_1]]\!](\sigma) & \triangleq & \{n \in \mathbb{Z} \mid c_0 \le n \le c_1\}
\end{array}
$$

Expressions

$$\boxed{E[\![e]\!] : \Sigma \to \mathcal{P}(\mathbb{V})}$$

$$
\begin{aligned}
E[\![n]\!](\sigma) &\triangleq \{n\} \\
E[\![x]\!](\sigma) &\triangleq \begin{cases} \{\sigma(x)\} & \text{if } x \in \mathsf{dom}(\sigma) \\ \varnothing & \text{otherwise} \end{cases} \\
E[\![[c_0, c_1]]\!](\sigma) &\triangleq \{n \in \mathbb{Z} \mid c_0 \leq n \leq c_1\} \\
E[\![e_1 \texttt{ + } e_2]\!](\sigma) &\triangleq \{n_1 + n_2 \mid n_1 \in E[\![e_1]\!](\sigma) \land n_2 \in E[\![e_2]\!](\sigma)\} \\
E[\![e_1 \texttt{ * } e_2]\!](\sigma) &\triangleq \{n_1 \times n_2 \mid n_1 \in E[\![e_1]\!](\sigma) \land n_2 \in E[\![e_2]\!](\sigma)\}
\end{aligned}
$$

Expressions

PLRG

$$\boxed{E[\![e]\!] : \Sigma \to \mathcal{P}(\mathbb{V})}$$

$$
\begin{aligned}
E[\![n]\!](\sigma) &\triangleq \{n\} \\
E[\![x]\!](\sigma) &\triangleq \begin{cases} \{\sigma(x)\} & \text{if } x \in \mathsf{dom}(\sigma) \\ \varnothing & \text{otherwise} \end{cases} \\
E[\![[c_0, c_1]]\!](\sigma) &\triangleq \{n \in \mathbb{Z} \mid c_0 \leq n \leq c_1\} \\
E[\![e_1 \texttt{ + } e_2]\!](\sigma) &\triangleq \{n_1 + n_2 \mid n_1 \in E[\![e_1]\!](\sigma) \wedge n_2 \in E[\![e_2]\!](\sigma)\} \\
E[\![e_1 \texttt{ * } e_2]\!](\sigma) &\triangleq \{n_1 \times n_2 \mid n_1 \in E[\![e_1]\!](\sigma) \wedge n_2 \in E[\![e_2]\!](\sigma)\} \\
E[\![e_1 \texttt{ < } e_2]\!](\sigma) &\triangleq \{\texttt{true} \mid \exists n_1 \in E[\![e_1]\!](\sigma).\ \exists n_2 \in E[\![e_2]\!](\sigma).\ n_1 < n_2\} \cup \\
&\ \{\texttt{false} \mid \exists n_1 \in E[\![e_1]\!](\sigma).\ \exists n_2 \in E[\![e_2]\!](\sigma).\ n_1 \geq n_2\}
\end{aligned}
$$

Expressions

$$\boxed{E[\![e]\!] : \Sigma \to \mathcal{P}(\mathbb{V})}$$

$$
\begin{array}{lcl}
E[\![n]\!](\sigma) & \triangleq & \{n\} \\
E[\![x]\!](\sigma) & \triangleq & \begin{cases} \{\sigma(x)\} & \text{if } x \in \mathsf{dom}(\sigma) \\ \varnothing & \text{otherwise} \end{cases} \\
E[\![[c_0, c_1]]\!](\sigma) & \triangleq & \{n \in \mathbb{Z} \mid c_0 \leq n \leq c_1\} \\
E[\![e_1 \texttt{ + } e_2]\!](\sigma) & \triangleq & \{n_1 + n_2 \mid n_1 \in E[\![e_1]\!](\sigma) \land n_2 \in E[\![e_2]\!](\sigma)\} \\
E[\![e_1 \texttt{ * } e_2]\!](\sigma) & \triangleq & \{n_1 \times n_2 \mid n_1 \in E[\![e_1]\!](\sigma) \land n_2 \in E[\![e_2]\!](\sigma)\} \\
E[\![e_1 \texttt{ < } e_2]\!](\sigma) & \triangleq & \{\texttt{true} \mid \exists n_1 \in E[\![e_1]\!](\sigma).\ \exists n_2 \in E[\![e_2]\!](\sigma).\ n_1 < n_2\} \cup \\
& & \{\texttt{false} \mid \exists n_1 \in E[\![e_1]\!](\sigma).\ \exists n_2 \in E[\![e_2]\!](\sigma).\ n_1 \geq n_2\} \\
E[\![\texttt{true}]\!](\sigma) & \triangleq & \{\texttt{true}\} \\
E[\![\texttt{false}]\!](\sigma) & \triangleq & \{\texttt{false}\}
\end{array}
$$

Statements

$$\boxed{S[\![s]\!] \in \mathcal{P}(\Sigma) \to \mathcal{P}(\Sigma)}$$

$$S[\![\texttt{skip}]\!](X) \quad \triangleq \quad X$$

Statements

$$\boxed{S[\![s]\!] \in \mathcal{P}(\Sigma) \to \mathcal{P}(\Sigma)}$$

$$S[\![\texttt{skip}]\!](X) \triangleq X$$

$$S[\![x \texttt{ := } e]\!](X) \triangleq \{\sigma[x \mapsto v] \mid \sigma \in X \land v \in E[\![e]\!](\sigma)\}$$

Statements

$$\boxed{S[\![s]\!] \in \mathcal{P}(\Sigma) \rightarrow \mathcal{P}(\Sigma)}$$

$$
\begin{aligned}
S[\![\texttt{skip}]\!](X) &\triangleq X \\
S[\![x \texttt{ := } e]\!](X) &\triangleq \{\sigma[x \mapsto v] \mid \sigma \in X \land v \in E[\![e]\!](\sigma)\} \\
S[\![s_1 \,\texttt{;}\, s_2]\!] &\triangleq S[\![s_2]\!] \circ S[\![s_1]\!]
\end{aligned}
$$

Statements

$$\boxed{S[\![s]\!] \in \mathcal{P}(\Sigma) \to \mathcal{P}(\Sigma)}$$

$$
\begin{array}{lcl}
S[\![\texttt{skip}]\!](X) & \triangleq & X \\
S[\![x\ \texttt{:=}\ e]\!](X) & \triangleq & \{\sigma[x \mapsto v] \mid \sigma \in X \land v \in E[\![e]\!](\sigma)\} \\
S[\![s_1\,\texttt{;}\,s_2]\!] & \triangleq & S[\![s_2]\!] \circ S[\![s_1]\!] \\
S[\![\texttt{if}\ e\ \texttt{then}\ s_1\ \texttt{else}\ s_2]\!](X) & \triangleq & S[\![s_1]\!](\{\sigma \in X \mid \texttt{true} \in E[\![e]\!](\sigma)\}) \cup \\
 & & S[\![s_2]\!](\{\sigma \in X \mid \texttt{false} \in E[\![e]\!](\sigma)\})
\end{array}
$$

Statements

$$\boxed{S[\![s]\!] \in \mathcal{P}(\Sigma) \to \mathcal{P}(\Sigma)}$$

$$\begin{array}{lcl}
S[\![\texttt{skip}]\!](X) & \triangleq & X \\
S[\![x \texttt{ := } e]\!](X) & \triangleq & \{\sigma[x \mapsto v] \mid \sigma \in X \land v \in E[\![e]\!](\sigma)\} \\
S[\![s_1 \texttt{;} s_2]\!] & \triangleq & S[\![s_2]\!] \circ S[\![s_1]\!] \\
S[\![\texttt{if } e \texttt{ then } s_1 \texttt{ else } s_2]\!](X) & \triangleq & S[\![s_1]\!](\{\sigma \in X \mid \texttt{true} \in E[\![e]\!](\sigma)\}) \cup \\
& & S[\![s_2]\!](\{\sigma \in X \mid \texttt{false} \in E[\![e]\!](\sigma)\}) \\
S[\![\texttt{while } e \texttt{ do } s]\!] & \triangleq & \mathbf{lfp}(F)
\end{array}$$

where

$$\begin{array}{lcl}
F(g)(X) & \triangleq & g(S[\![s]\!](\{\sigma \in X \mid \texttt{true} \in E[\![e]\!](\sigma)\})) \cup \\
& & \{\sigma \in X \mid \texttt{false} \in E[\![e]\!](\sigma)\}
\end{array}$$

Statements

$$\boxed{S[\![s]\!] \in \mathcal{P}(\Sigma) \to \mathcal{P}(\Sigma)}$$

$$\begin{array}{lcl}
S[\![\texttt{skip}]\!](X) & \triangleq & X \\
S[\![x \ \texttt{:=}\ e]\!](X) & \triangleq & \{\sigma[x \mapsto v] \mid \sigma \in X \land v \in E[\![e]\!](\sigma)\} \\
S[\![s_1\,\texttt{;}\,s_2]\!] & \triangleq & S[\![s_2]\!] \circ S[\![s_1]\!] \\
S[\![\texttt{if}\ e\ \texttt{then}\ s_1\ \texttt{else}\ s_2]\!](X) & \triangleq & S[\![s_1]\!](\{\sigma \in X \mid \texttt{true} \in E[\![e]\!](\sigma)\}) \cup \\
 & & S[\![s_2]\!](\{\sigma \in X \mid \texttt{false} \in E[\![e]\!](\sigma)\}) \\
S[\![\texttt{while}\ e\ \texttt{do}\ s]\!] & \triangleq & \mathbf{lfp}(F)
\end{array}$$

where

$$\begin{array}{lcl}
F(g)(X) & \triangleq & g(S[\![s]\!](\{\sigma \in X \mid \texttt{true} \in E[\![e]\!](\sigma)\})) \cup \\
 & & \{\sigma \in X \mid \texttt{false} \in E[\![e]\!](\sigma)\}
\end{array}$$

We will talk about why the least fixed-point of F exists in the next lecture with the **fixed-point theory**.

Next Lecture

PLRG

- Fixed-Point Theory

Jihyeok Park
jihyeok_park@korea.ac.kr
https://plrg.korea.ac.kr