Small Step Semantics

Dr. Mattox Beckman

University of Illinois at Urbana-Champaign
Department of Computer Science

Objectives

You should be able to ...

- Define the word "semantics."
- Determine the value of an expression using small step semantics.
- Specify the meaning of a language by writing a semantic rule.

Parts of a Formal System

To create a formal system, you must specify the following:

- A set of *symbols* or an *alphabet*
- A definition of a *valid sentence*
- A set of *transformation rules* to make new valid sentences out of old ones
- A set of *initial valid sentences*

You do **NOT** need:

- An *interpretation* of those symbols
  They are highly recommended, but the formal system can exist and do its work without one.

Example

Symbols $S, (,), Z, P, x, y$.

Definition of a furbitz

- Z is a furbitz. x and y are variables of type furbitz.
- If x is a furbitz, then $S(x)$ is a furbitz.
- If x and y are furbitzi, then $P(x, y)$ is a furbitz.

Definition of the gloppit relation

- Z has the gloppit relation with Z.
- If x and y have the gloppit relation, then $S(x)$ and $S(y)$ have the gloppit relation.
- If α and β, then we can write $\alpha g \beta$.

True sentences If $\alpha g \beta$, then also

- $P(S(\alpha), \beta) g S(P(\alpha, \beta))$, and $P(Z, \beta) g \beta$

Example

Symbols $S, (,), Z, P, x, y$.

Definition of an integer

- 0 is an integer. x and y are variables of type integer.
- If x is an integer, then $S(x)$ is an integer.
- If x and y are integers, then $P(x, y)$ is an integer.

Definition of the equality relation

- 0 has the equality relation with 0.
- If x and y have the equality relation, then $S(x)$ and $S(y)$ have the equality relation.
- If α and β, then we can write $\alpha = \beta$.

True sentences If $\alpha = \beta$, then also

- $P(S(\alpha), \beta) = S(P(\alpha, \beta))$, and $P(0, \beta) = \beta$

Grammar for Simple Imperative Programming Language

The Language

$$
\begin{array}{rcl}
S ::= & & \mathtt{skip} \\
& | & u := t \\
& | & S_1 ; S_2 \\
& | & \mathtt{if}\ B\ \mathtt{then}\ S_1\ \mathtt{else}\ S_2\ \mathtt{fi} \\
& | & \mathtt{while}\ B\ \mathtt{do}\ S_1\ \mathtt{od}
\end{array}
$$

- Let u be a possibly subscripted variable.
- Let t be an expression of some sort.
- Let B be a boolean expression.

Transitions

- There are many ways we can specify the meaning of an expression. One way is to specify the steps that the computer will take during an evaluation.
- A *transition* has the following form:

$$< S_1, \sigma > \rightarrow < S_2, \tau >$$

  where S_1 and S_2 are statements, and σ and τ represent environments. The statement could change the environment.
- Note well: $\rightarrow$ indicates *exactly one* step of evaluation. (Hence "small step semantics.")

Definition of $\rightarrow$, 1

Skip and Assignment

$$< \texttt{skip}\,, \sigma > \rightarrow < E, \sigma >$$
$$< u := t, \sigma > \rightarrow < E, \sigma[u := \sigma(t)] >$$

- σ will have the form $\{u_1 := t_1, u_2 := t_2, \ldots, u_n := t_n\}$
- If $\sigma = \{x := 5\}$, then we can say $\sigma(x) = 5$
- We can update σ.

$$\sigma[x := 20] = \{x := 20\}$$
$$\sigma[y := 20] = \{x := 5, y := 20\}$$

Definition of $\rightarrow$, 2

Sequencing

$$\frac{< S_1, \sigma > \rightarrow < S_2, \tau >}{< S_1; S, \sigma > \rightarrow < S_2; S, \tau >}$$

$$E; S \equiv S$$

- Notice how we don't talk about the second statement at all!

Definition of $\rightarrow$, 3

If

$$< \texttt{if } B \texttt{ then } S_1 \texttt{ else } S_2 \texttt{ fi}, \sigma > \rightarrow < S_1, \sigma > \text{ where } \sigma \models B$$
$$< \texttt{if } B \texttt{ then } S_1 \texttt{ else } S_2 \texttt{ fi}, \sigma > \rightarrow < S_2, \sigma > \text{ where } \sigma \models \neg B$$

- The notation $\sigma \models B$ means "B is true given variable assignments in σ."
- $\{x := 20, y := 30\} \models x < y$

Definition of $\rightarrow$, 4

While

$$< \texttt{while}\ B\ \texttt{do}\ S_1\ \texttt{od}\ , \sigma > \rightarrow < S_1; \texttt{while}\ B\ \texttt{do}\ S_1\ \texttt{od}\ , \sigma > \text{ where } \sigma \models B$$
$$< \texttt{while}\ B\ \texttt{do}\ S_1\ \texttt{od}\ , \sigma > \rightarrow < E, \sigma > \text{ where } \sigma \models \neg B$$

- Notice how the body of the while loop is copied in front of the loop!

Example Evaluation

Evaluate: `x:=1; n:=3; while n>1 do x:=x*n; n:=n-1 od`

$$< \texttt{x:=1; n:=3; while n>1 do x:=x*n; n:=n-1 od}, \{\} >$$

Example Evaluation

Evaluate: `x:=1; n:=3; while n>1 do x:=x*n; n:=n-1 od`

$$
\begin{aligned}
& < \texttt{x:=1; n:=3; while n>1 do x:=x*n; n:=n-1 od}, \{\} > \\
\rightarrow & < \texttt{n:=3; while n>1 do x:=x*n; n:=n-1 od}, \{x := 1\} >
\end{aligned}
$$

Example Evaluation

Evaluate: `x:=1; n:=3; while n>1 do x:=x*n; n:=n-1 od`

$$
\begin{array}{l}
\quad < \texttt{x:=1; n:=3; while n>1 do x:=x*n; n:=n-1 od}, \{\} > \\
\rightarrow < \texttt{n:=3; while n>1 do x:=x*n; n:=n-1 od}, \{x := 1\} > \\
\rightarrow < \texttt{while n>1 do x:=x*n; n:=n-1 od}, \{x := 1, n := 3\} >
\end{array}
$$

Example Evaluation

Evaluate: `x:=1; n:=3; while n>1 do x:=x*n; n:=n-1 od`

$$
\begin{array}{l}
\quad < \texttt{x:=1; n:=3; while n>1 do x:=x*n; n:=n-1 od}, \{\} > \\
\to < \texttt{n:=3; while n>1 do x:=x*n; n:=n-1 od}, \{x := 1\} > \\
\to < \texttt{while n>1 do x:=x*n; n:=n-1 od}, \{x := 1, n := 3\} > \\
\to < \texttt{x:=x*n;n:=n-1;while n>1 do x:=x*n; n:=n-1 od}, \\
\qquad\qquad \{x := 1, n := 3\} >
\end{array}
$$

Example Evaluation

Evaluate: `x:=1; n:=3; while n>1 do x:=x*n; n:=n-1 od`

$$
\begin{aligned}
& < \texttt{x:=1; n:=3; while n>1 do x:=x*n; n:=n-1 od}, \{\} > \\
\rightarrow & < \texttt{n:=3; while n>1 do x:=x*n; n:=n-1 od}, \{x := 1\} > \\
\rightarrow & < \texttt{while n>1 do x:=x*n; n:=n-1 od}, \{x := 1, n := 3\} > \\
\rightarrow & < \texttt{x:=x*n;n:=n-1;while n>1 do x:=x*n; n:=n-1 od}, \\
& \qquad \{x := 1, n := 3\} > \\
\rightarrow & < \texttt{n:=n-1;while n>1 do x:=x*n; n:=n-1 od}, \{x := 3, n := 3\} >
\end{aligned}
$$

Example Evaluation

Evaluate: `x:=1; n:=3; while n>1 do x:=x*n; n:=n-1 od`

$$
\begin{aligned}
& <\texttt{x:=1; n:=3; while n>1 do x:=x*n; n:=n-1 od}, \{\} > \\
\rightarrow & <\texttt{n:=3; while n>1 do x:=x*n; n:=n-1 od}, \{x := 1\} > \\
\rightarrow & <\texttt{while n>1 do x:=x*n; n:=n-1 od}, \{x := 1, n := 3\} > \\
\rightarrow & <\texttt{x:=x*n;n:=n-1;while n>1 do x:=x*n; n:=n-1 od}, \\
& \qquad \{x := 1, n := 3\} > \\
\rightarrow & <\texttt{n:=n-1;while n>1 do x:=x*n; n:=n-1 od}, \{x := 3, n := 3\} > \\
\rightarrow & <\texttt{while n>1 do x:=x*n; n:=n-1 od}, \{x := 3, n := 2\} >
\end{aligned}
$$

Example Evaluation

Evaluate: `x:=1; n:=3; while n>1 do x:=x*n; n:=n-1 od`

$$
\begin{aligned}
& < \texttt{x:=1; n:=3; while n>1 do x:=x*n; n:=n-1 od}, \{\} > \\
\rightarrow\ & < \texttt{n:=3; while n>1 do x:=x*n; n:=n-1 od}, \{x := 1\} > \\
\rightarrow\ & < \texttt{while n>1 do x:=x*n; n:=n-1 od}, \{x := 1, n := 3\} > \\
\rightarrow\ & < \texttt{x:=x*n;n:=n-1;while n>1 do x:=x*n; n:=n-1 od}, \\
& \qquad \{x := 1, n := 3\} > \\
\rightarrow\ & < \texttt{n:=n-1;while n>1 do x:=x*n; n:=n-1 od}, \{x := 3, n := 3\} > \\
\rightarrow\ & < \texttt{while n>1 do x:=x*n; n:=n-1 od}, \{x := 3, n := 2\} > \\
\rightarrow\ & < \texttt{x:=x*n;n:=n-1;while n>1 do x:=x*n; n:=n-1 od}, \\
& \qquad \{x := 3, n := 2\} >
\end{aligned}
$$

Example Evaluation

Evaluate: `x:=1; n:=3; while n>1 do x:=x*n; n:=n-1 od`

$$
\begin{aligned}
& < \texttt{x:=1; n:=3; while n>1 do x:=x*n; n:=n-1 od}, \{\} > \\
\rightarrow & < \texttt{n:=3; while n>1 do x:=x*n; n:=n-1 od}, \{x := 1\} > \\
\rightarrow & < \texttt{while n>1 do x:=x*n; n:=n-1 od}, \{x := 1, n := 3\} > \\
\rightarrow & < \texttt{x:=x*n;n:=n-1;while n>1 do x:=x*n; n:=n-1 od}, \\
& \qquad \{x := 1, n := 3\} > \\
\rightarrow & < \texttt{n:=n-1;while n>1 do x:=x*n; n:=n-1 od}, \{x := 3, n := 3\} > \\
\rightarrow & < \texttt{while n>1 do x:=x*n; n:=n-1 od}, \{x := 3, n := 2\} > \\
\rightarrow & < \texttt{x:=x*n;n:=n-1;while n>1 do x:=x*n; n:=n-1 od}, \\
& \qquad \{x := 3, n := 2\} > \\
\rightarrow & < \texttt{n:=n-1;while n>1 do x:=x*n; n:=n-1 od}, \{x := 6, n := 2\} >
\end{aligned}
$$

Example Evaluation

Evaluate: `x:=1; n:=3; while n>1 do x:=x*n; n:=n-1 od`

$$< \texttt{x:=1; n:=3; while n>1 do x:=x*n; n:=n-1 od}, \{\} >$$
$$\to < \texttt{n:=3; while n>1 do x:=x*n; n:=n-1 od}, \{x := 1\} >$$
$$\to < \texttt{while n>1 do x:=x*n; n:=n-1 od}, \{x := 1, n := 3\} >$$
$$\to < \texttt{x:=x*n;n:=n-1;while n>1 do x:=x*n; n:=n-1 od}, \{x := 1, n := 3\} >$$
$$\to < \texttt{n:=n-1;while n>1 do x:=x*n; n:=n-1 od}, \{x := 3, n := 3\} >$$
$$\to < \texttt{while n>1 do x:=x*n; n:=n-1 od}, \{x := 3, n := 2\} >$$
$$\to < \texttt{x:=x*n;n:=n-1;while n>1 do x:=x*n; n:=n-1 od}, \{x := 3, n := 2\} >$$
$$\to < \texttt{n:=n-1;while n>1 do x:=x*n; n:=n-1 od}, \{x := 6, n := 2\} >$$
$$\to < \texttt{while n>1 do x:=x*n; n:=n-1 od}, \{x := 6, n := 1\} >$$

Example Evaluation

Evaluate: `x:=1; n:=3; while n>1 do x:=x*n; n:=n-1 od`

$$
\begin{aligned}
& < \texttt{x:=1; n:=3; while n>1 do x:=x*n; n:=n-1 od}, \{\} > \\
\rightarrow & < \texttt{n:=3; while n>1 do x:=x*n; n:=n-1 od}, \{x := 1\} > \\
\rightarrow & < \texttt{while n>1 do x:=x*n; n:=n-1 od}, \{x := 1, n := 3\} > \\
\rightarrow & < \texttt{x:=x*n;n:=n-1;while n>1 do x:=x*n; n:=n-1 od}, \\
& \qquad \{x := 1, n := 3\} > \\
\rightarrow & < \texttt{n:=n-1;while n>1 do x:=x*n; n:=n-1 od}, \{x := 3, n := 3\} > \\
\rightarrow & < \texttt{while n>1 do x:=x*n; n:=n-1 od}, \{x := 3, n := 2\} > \\
\rightarrow & < \texttt{x:=x*n;n:=n-1;while n>1 do x:=x*n; n:=n-1 od}, \\
& \qquad \{x := 3, n := 2\} > \\
\rightarrow & < \texttt{n:=n-1;while n>1 do x:=x*n; n:=n-1 od}, \{x := 6, n := 2\} > \\
\rightarrow & < \texttt{while n>1 do x:=x*n; n:=n-1 od}, \{x := 6, n := 1\} > \\
\rightarrow & < E, \{x := 6, n := 1\} >
\end{aligned}
$$