Programming Languages and Compilers (CS 421)

Elsa L Gunter

2112 SC, UIUC

http://www.cs.uiuc.edu/class/fa06/cs421/

Based in part on slides by Mattox Beckman, as updated by Vikram Adve and Gul Agha

Natural Semantics

- Similar to transition semantics except
  - Transition semantics is relation between individual steps of computation
  - Natural semantics is relation between computation state and final result
- Rules look like $(C, m) \Downarrow m'$
- Always want

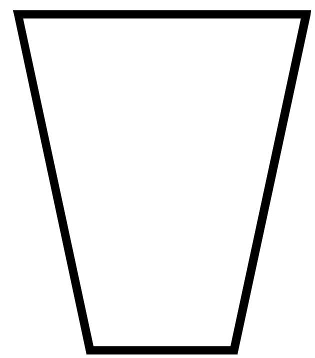
  Lemma: $(C,m) \dashrightarrow^* m'$ iff $(C, m) \Downarrow m'$

Picture

- Transition semantics

(C_1,m_1) --> (C_2,m_2) --> (C_3,m_3) --> … --> m

- Natural Semantics



$(C_1,m_1) \Downarrow m$

Natural Semantics of Atomic Expressions

- Same as Transition

- Identifiers: $(I,m) \Downarrow m(I)$
- Numerals are values: $(N,m) \Downarrow N$
- Booleans: $(\text{true},m) \Downarrow \text{true}$

  $(\text{false}\ ,m) \Downarrow \text{false}$

Booleans:

$$\frac{(B, m) \Downarrow \text{false}}{(B \,\&\, B', m) \Downarrow \text{false}} \qquad \frac{(B, m) \Downarrow \text{true} \quad (B', m) \Downarrow b}{(B \,\&\, B', m) \Downarrow b}$$

$$\frac{(B, m) \Downarrow \text{true}}{(B \text{ or } B', m) \Downarrow \text{true}} \qquad \frac{(B, m) \Downarrow \text{false} \quad (B', m) \Downarrow b}{(B \text{ or } B', m) \Downarrow b}$$

$$\frac{(B, m) \Downarrow \text{true}}{(\text{not } B, m) \Downarrow \text{false}} \qquad \frac{(B, m) \Downarrow \text{false}}{(\text{not } B, m) \Downarrow \text{true}}$$

Relations

$$\frac{(E, m) \Downarrow U \quad (E', m) \Downarrow V \quad U \sim V = b}{(E \sim E', m) \Downarrow b}$$

- By $U \sim V = b$, we mean does (the meaning of) the relation ~ hold on the meaning of *U* and *V*
- May be specified by a mathematical expression/equation or rules matching *U* and *V*

Arithmetic Expressions

$$\frac{(E, m) \Downarrow U \quad (E', m) \Downarrow V \quad U \; op \; V = N}{(E \; op \; E', m) \Downarrow N}$$

where N is the specified value for $U \; op \; V$

Commands

$$(\text{skip}, m) \Downarrow m$$

$$\frac{(E,m) \Downarrow V}{(I::=E,m) \Downarrow m[I \text{ <-- } V\,]}$$

$$\frac{(C,m) \Downarrow m' \quad (C',m') \Downarrow m''}{(C;C',\, m) \Downarrow m''}$$

If Then Else Command

$$\frac{(B,m) \Downarrow \text{true} \quad (C,m) \Downarrow m'}{(\text{if } B \text{ then } C \text{ else } C' \text{ fi}, m) \Downarrow m'}$$

$$\frac{(B,m) \Downarrow \text{false} \quad (C',m) \Downarrow m'}{(\text{if } B \text{ then } C \text{ else } C' \text{ fi}, m) \Downarrow m'}$$

While Command

$$\frac{(B,m) \Downarrow \text{false}}{(\text{while } B \text{ do } C \text{ od}, m) \Downarrow m}$$

$$\frac{(B,m)\Downarrow\text{true} \quad (C,m)\Downarrow m' \quad (\text{while } B \text{ do } C \text{ od}, m')\Downarrow m''}{(\text{while } B \text{ do C od}, m) \Downarrow m''}$$

Example

$$\frac{}{\text{(if } x > 5 \text{ then } y{:=}\ 2 + 3 \text{ else } y{:=}3 + 4 \text{ fi}, \{x \text{ -> } 7\}) \Downarrow ?}$$

Example

$$\frac{\overline{(x > 5, \{x \text{ -> } 7\}) \Downarrow ?}}{(\text{if } x > 5 \text{ then } y{:=}\ 2 + 3 \text{ else } y{:=}3 + 4 \text{ fi}, \{x \text{ -> } 7\}) \Downarrow\ ?}$$

Example

$$\frac{\dfrac{? > ? = ?}{(x,\{x\text{->}7\})\Downarrow ? \quad (5,\{x\text{->}\})\Downarrow ?}}{\dfrac{(x > 5, \{x \text{->} 7\})\Downarrow ?}{(\text{if } x > 5 \text{ then } y{:}= 2 + 3 \text{ else } y{:}=3 + 4 \text{ fi}, \{x \text{->} 7\}) \Downarrow ?}}$$

Example

7 > 5 = true

(x,{x->7})$\Downarrow$7   (5,{x->})$\Downarrow$5

(x > 5, {x -> 7})$\Downarrow$?

(if x > 5 then y:= 2 + 3 else y:=3 + 4 fi,

{x -> 7}) $\Downarrow$?

Example

$$
\dfrac{
\begin{array}{c}
7 > 5 = \text{true} \\
\dfrac{(x,\{x\text{->}7\}) \Downarrow 7 \quad (5,\{x\text{->}\}) \Downarrow 5}{(x > 5,\ \{x \text{ -> } 7\}) \Downarrow \text{true}}
\end{array}
}{
\begin{array}{c}
(\text{if } x > 5 \text{ then } y\text{:= } 2 + 3 \text{ else } y\text{:=}3 + 4 \text{ fi}, \\
\{x \text{ -> } 7\}) \Downarrow ?
\end{array}
}
$$

Example

$$\dfrac{\dfrac{7 > 5 = \text{true} \qquad (x,\{x\text{->}7\})\Downarrow 7 \quad (5,\{x\text{->}\})\Downarrow 5}{(x > 5, \{x \text{->} 7\})\Downarrow \text{true}} \qquad \dfrac{}{(y{:}= 2 + 3, \{x\text{->} 7\}) \Downarrow \text{?}}}{(\text{if } x > 5 \text{ then } y{:}= 2 + 3 \text{ else } y{:}=3 + 4 \text{ fi}, \{x \text{->} 7\}) \Downarrow \text{?}}$$

Example

7 > 5 = true

(x,{x->7})⇓7 (5,{x->})⇓5

(x > 5, {x -> 7})⇓true

(2+3, {x->7})⇓?

(y:= 2 + 3, {x-> 7}

⇓ ?

(if x > 5 then y:= 2 + 3 else y:=3 + 4 fi,

{x -> 7}) ⇓ ?

Example

? + ? = ?

(2,{x->7})⇓? (3,{x->7}) ⇓?

---

7 > 5 = true (2+3, {x->7})⇓?

(x,{x->7})⇓7 (5,{x->})⇓5 (y:= 2 + 3, {x-> 7}

(x > 5, {x -> 7})⇓true ⇓? .

(if x > 5 then y:= 2 + 3 else y:=3 + 4 fi,

{x -> 7}) ⇓ ?

Example

2 + 3 = 5

(2,{x->7})⇓2 (3,{x->7}) ⇓3

---

7 > 5 = true (2+3, {x->7})⇓?

(x,{x->7})⇓7 (5,{x->})⇓5 (y:= 2 + 3, {x-> 7}

(x > 5, {x -> 7})⇓true ⇓ ?

(if x > 5 then y:= 2 + 3 else y:=3 + 4 fi,

{x -> 7}) ⇓ ?

Example

2 + 3 = 5

(2,{x->7})⇓2   (3,{x->7}) ⇓3

---

7 > 5 = true   (2+3, {x->7})⇓5

(x,{x->7})⇓7   (5,{x->})⇓5   (y:= 2 + 3, {x-> 7}

(x > 5, {x -> 7})⇓true   ⇓?

(if x > 5 then y:= 2 + 3 else y:=3 + 4 fi,

{x -> 7}) ⇓ ?

Example

$$\frac{\dfrac{7 > 5 = \text{true} \qquad (x,\{x\text{->}7\})\Downarrow 7 \quad (5,\{x\text{->}\})\Downarrow 5}{(x > 5, \{x \text{ -> } 7\})\Downarrow \text{true}} \qquad \dfrac{\dfrac{\begin{array}{c}2 + 3 = 5 \\ (2,\{x\text{->}7\})\Downarrow 2 \quad (3,\{x\text{->}7\}) \Downarrow 3\end{array}}{(2+3, \{x\text{->}7\})\Downarrow 5}}{\begin{array}{c}(y:= 2 + 3, \{x\text{-> } 7\} \\ \Downarrow \{x\text{->}7, y\text{->}5\}\end{array}}}{\begin{array}{c}(\text{if } x > 5 \text{ then } y:= 2 + 3 \text{ else } y:=3 + 4 \text{ fi}, \\ \{x \text{ -> } 7\}) \Downarrow ?\end{array}}$$

Example

$$\dfrac{\dfrac{7 > 5 = \text{true}}{(x,\{x\text{->}7\})\Downarrow 7 \quad (5,\{x\text{->}\})\Downarrow 5}}{(x > 5,\ \{x \text{->} 7\})\Downarrow \text{true}} \qquad \dfrac{\dfrac{\dfrac{2 + 3 = 5}{(2,\{x\text{->}7\})\Downarrow 2 \quad (3,\{x\text{->}7\})\Downarrow 3}}{(2{+}3,\ \{x\text{->}7\})\Downarrow 5}}{(y{:=}\ 2 + 3,\ \{x\text{->}\ 7\} \Downarrow \{x\text{->}7,\ y\text{->}5\}}$$

$$\overline{(\text{if } x > 5 \text{ then } y{:=}\ 2 + 3 \text{ else } y{:=}3 + 4 \text{ fi},\ \{x \text{->} 7\}) \Downarrow \{x\text{->}7,\ y\text{->}5\}}$$

Let in Command

$$\frac{(E,m) \Downarrow V \quad (C,m[x \text{<-} V]) \Downarrow m'}{(\text{let } x = E \text{ in } C,\ m) \Downarrow m''}$$

Where $m''(y) = m'(y)$ for $y \neq x$ and $m''(x) = m(x)$ if $m(x)$ is defined, and $m''(x)$ is undefined otherwisei

Example

$$\frac{(5,\{x->17\}) \Downarrow 5 \qquad \frac{\dfrac{(x,\{x->5\}) \Downarrow 5 \quad (3,\{x->5\}) \Downarrow 3}{(x+3,\{x->5\}) \Downarrow 8}}{(x:=x+3,\{x->5\}) \Downarrow \{x->8\}}}{(\text{let } x = 5 \text{ in } (x:=x+3),\ \{x -> 17\}) \Downarrow ?}$$

Example

$$\dfrac{(5,\{x\text{->}17\}) \Downarrow 5 \qquad \dfrac{\dfrac{(x,\{x\text{->}5\}) \Downarrow 5 \quad (3,\{x\text{->}5\}) \Downarrow 3}{(x+3,\{x\text{->}5\}) \Downarrow 8}}{(x:=x+3,\{x\text{->}5\}) \Downarrow \{x\text{->}8\}}}{(\text{let } x = 5 \text{ in } (x:=x+3),\ \{x \text{ -> } 17\}) \Downarrow \{x \text{ -> } 17\}}$$

Comment

- Simple Imperative Programming Language introduces variables *implicitly* through assignment
- The let-in command introduces scoped variables *explictly*
- Clash of constructs apparent in awkward semantics

Interpretation Versus Compilation

- A **compiler** from language L1 to language L2 is a program that takes an L1 program and for each piece of code in L1 generates a piece of code in L2 of same meaning
- An **interpreter** of L1 in L2 is an L2 program that executes the meaning of a given L1 program
- Compiler would examine the body of a loop once; an interpreter would examine it every time the loop was executed

Interpreter

- An *Interpreter* represents the operational semantics of a language L1 (source language) in the language of implementation L2 (target language)
- Built incrementally
  - Start with literals
  - Variables
  - Primitive operations
  - Evaluation of expressions
  - Evaluation of commands/declarations

Interpreter

- Takes abstract syntax trees as input
  - In simple cases could be just strings
- One procedure for each syntactic category (nonterminal)
  - eg one for expressions, another for commands
- If Natural semantics used, tells how to compute final value from code
- If Transition semantics used, tells how to compute next "state"
  - To get final value, put in a loop

Natural Semantics Example

- compute_exp (Var(v), m) = look_up v m
- compute_exp (Int(n), _) = Num (n)
- …
- compute_com(IfExp(b,c1,c2),m) =

  if compute_exp (b,m) = Bool(true)

  then compute_com (c1,m)

  else compute_com (c2,m)

Natural Semantics Example

- compute_com(While(b,c), m) =
  if compute_exp (b,m) = Bool(false)
  then m
  else compute_com
  (While(b,c), compute_com(c,m))

- May fail to terminate - exceed stack limits
- Returns no useful information then

Why Have Both Semantics?

- Natural Semantics corresponds to a recursive program for evaluating
- Transition Semantics corresponds to iterative program for evaluating one step at a time
- Natural Semantics more concise but can't express nonterminating computation