Transition Semantics

- Form of operational semantics
- Describes how each program construct transforms machine state by *transitions*
- Rules look like
  (C, *m*) --> (C' , *m'*) or (C,*m*) --> *m'*
- *C, C'* is code remaining to be executed
- m, m' represent the state/store/memory/ environment
  - Partial mapping from identifiers to values
  - Sometimes *m* (or *C*) not needed
- Indicates exactly one step of computation

Expressions and Values

- *C, C'* used for commands; *E, E' for* expressions; *U,V* for values
- Special class of expressions designated as *values*
  - Eg 2, 3 are values, but 2+3 is only an expression
- Memory only holds values
  - Other possibilities exist

Evaluation Semantics

- Transitions successfully stops when E/*C* is a value/memory
- Evaluation fails if no transition possible, but not at value/memory
- Value/memory is the final *meaning* of original expression/command (in the given state)
- Coarse semantics: final value / memory
- More fine grained: whole transition sequence

Simple Imperative Programming Language

- *I* ∈ *Identifiers*
- *N* ∈ *Numerals*
- *B* ::= true | false | *B* & *B* | *B* or *B* | not *B* | *E* < *E* | *E* = *E*
- *E::= N | I | E + E | E * E | E - E | - E*
- C::= skip | *C*;*C* | *I* ::= *E* | if *B* then *C* else *C* fi | while *B* do *C* od

Transitions for Expressions

- Numerals are values
- Boolean values = {true, false}
- Identifiers: (*I,m*) --> (*m*(*I*), *m*)

Boolean Operations:

- Operators: (short-circuit)

(false & *B*, *m*) --> (false,*m*)
(true & *B, m*) --> (*B,m*)
$$\frac{(B, m) \text{ --> } (B'', m)}{(B \,\&\, B', m) \text{ --> } (B'' \,\&\, B', m)}$$

(true or *B*, *m*) --> (true,*m*)
(false or *B, m*) --> (*B,m*)
$$\frac{(B, m) \text{ --> } (B'', m)}{(B \text{ or } B', m) \text{ --> } (B'' \text{ or } B', m)}$$

(not true, m) --> (false,*m*)
(not false, m) --> (true,*m*)
$$\frac{(B, m) \text{ --> } (B', m)}{(\text{not } B, m) \text{ --> } (\text{not } B', m)}$$

Relations

$$\frac{(E, m) \text{ --> } (E'', m)}{(E \sim E', m) \text{ --> } (E'' \sim E', m)}$$

$$\frac{(E, m) \text{ --> } (E', m)}{(V \sim E, m) \text{ --> } (V \sim E', m)}$$

$(U \sim V, m)$ --> (true,m) or (false,m) depending on whether $U \sim V$ holds or not

Arithmetic Expressions

$$\frac{(E, m) \text{ --> } (E'', m)}{(E \text{ op } E', m) \text{ --> } (E'' \text{ op } E', m)}$$

$$\frac{(E, m) \text{ --> } (E', m)}{(V \text{ op } E, m) \text{ --> } (V \text{ op } E', m)}$$

$(U \text{ op } V, m)$ --> (N,m) where N is the specified value for U op V

Commands

(skip, m) --> m

$$\frac{(E,m) \text{ --> } (E', m)}{(I ::= E, m) \text{ --> } (I ::= E', m)}$$

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

$$\frac{(C,m) \text{ --> } (C'', m')}{(C;C', m) \text{ --> } (C'';C', m')} \qquad \frac{(C,m) \text{ --> } m'}{(C;C', m) \text{ --> } (C', m')}$$

If Then Else Command

(if true then C else C' fi, m) --> (C, m)

(if false then C else C' fi, m) --> (C', m)

$$\frac{(B,m) \text{ --> } (B', m)}{(\text{if } B \text{ then } C \text{ else } C' \text{ fi}, m) \text{ --> } (\text{if } B' \text{ then } C \text{ else } C' \text{ fi}, m)}$$

While Command

(while B do C od, m) -->
(if B then C; while B do C od else skip fi, m)

In English: Expand a While into a test of the boolean guard, with the true case being to do the body and then try the while loop again, and the false case being to stop.