COMS22202: 2015/16

Language Engineering

Dr Oliver Ray
(csxor@Bristol.ac.uk)

Department of Computer Science
University of Bristol

Thursday 14th April, 2016

Structural Operational Semantics p32-6

- Structural operational semantics (aka small step semantics) is more concerned *how* a program performs a computation as opposed to merely *which* computation it ultimately performs
- A structural operational semantics gives a finer level of control over the order of argument evaluation (leftmost first, rightmost first, parallel execution, lazy, strict, etc.)
- The philosophy of the structural semantics is to progressively reduce program expressions step by step in order to make them simpler and simpler while updating the state along the way
- Structural semantics specify transitions between configurations $\langle e,\sigma\rangle$ with an expression e and a state σ, that ultimately lead to a semantic value v. Such transitions are denoted by an arrow $\Rightarrow$
- The transitions are defined by axiom and rule schemata (as in the axiomatic semantics) where rule instances are obtained by replacing meta-variables by types satisfying the side conditions

Structural Semantics: Binary Numbers

- We can define a trivial operational semantics for binary numbers using an approach exactly analogous to the denotational semantics
- If we have a single digit we map it to the corresponding integer:

$$\frac{}{\langle 0, \sigma\rangle \Rightarrow 0} \qquad \frac{}{\langle 1, \sigma\rangle \Rightarrow 1}$$

- Otherwise we simply evaluate the numeral recursively:

$$\frac{\langle n , \sigma\rangle \Rightarrow k}{\langle n\ 0 , \sigma\rangle \Rightarrow 2k} \qquad \frac{\langle n , \sigma\rangle \Rightarrow k}{\langle n\ 1 , \sigma\rangle \Rightarrow 2k+1}$$

- Since numerals are atomic entities from a semantic point of view, all of these configurations evaluate directly to integers
- Notice that we could combine the last two rules as follows

$$\frac{\langle n , \sigma\rangle \Rightarrow i \quad \langle d , \sigma\rangle \Rightarrow j}{\langle n\ d , \sigma\rangle \Rightarrow k} \quad \text{where} \quad k=2i+j$$

for any state $\sigma \in$ State, any numeral $n \in$ Num, and any digit $d \in \{0,1\}$

Structural Semantics: Numerals

- It is convenient to utilise a subscript notation whereby we can write n_i to denote the numeral associated with any given integer i
- this helps avoid cluttering up our equations with many routine "boiler-plate" conversions between numerals and integers
- This works for any representation (binary, decimal, etc.) and allows the whole semantics of numerals to be simply captured by a single axiom schema as follows:

$$\textbf{NUM}\quad \frac{}{\langle n_i , \sigma \rangle \Rightarrow i}$$

- Note: as we will consider various different ways of writing structural semantic rules, we will explicitly attach a name like NUM above to the particular rules that we regard as actually being in our semantics

Structural Semantics: Variables

- If a variable v appears in a configuration we can simply reduce it to the numeral representing its value in state σ using

$$\frac{\langle n , \sigma\rangle \Rightarrow \sigma(v)}{\langle v , \sigma\rangle \Rightarrow \langle n , \sigma\rangle}$$

- Using our subscript notation, this is more conveniently written

$$\text{VAR} \quad \frac{}{\langle v , \sigma\rangle \Rightarrow \langle n_{\sigma(v)} , \sigma\rangle}$$

- Note: the new configuration $\langle n_{\sigma(v)},\sigma\rangle$ will give an integer $\sigma(v)$ in just one more step using NUM; but when evaluating a variable within an arithmetic expression, we must replace it by a numeral to give simpler arithmetic expression – which is why we don't simply use

$$\frac{}{\langle v , \sigma\rangle \Rightarrow \sigma(v)}$$

Structural Semantics: Addition

- Now we define an operational semantics for arithmetic expressions which enforces left-first argument evaluation
- Since we already have rules to deal with numerals and variables, let's consider the rules for expressions involving addition
- The simplest case is when we are adding two numerals

$$\textbf{ADD-N}\quad \frac{}{\langle n_i + n_j , \sigma \rangle \Rightarrow \langle n_{i+j} , \sigma \rangle}$$

- If the left term is not a numeral then we need to reduce it until it is

$$\textbf{ADD-L}\quad \frac{\langle a , \sigma \rangle \Rightarrow \langle a' , \sigma \rangle}{\langle a + c, \sigma \rangle \Rightarrow \langle a' + c, \sigma \rangle}$$

- Then, if the right term is not a numeral, we need to reduce that too

$$\textbf{ADD-R}\quad \frac{\langle b , \sigma \rangle \Rightarrow \langle b' , \sigma \rangle}{\langle n_i + b, \sigma \rangle \Rightarrow \langle n_i + b', \sigma \rangle}$$

- In these rules $\sigma \in \text{State}$ $n_k \in \text{Num}$ $a,b \in \text{Aexp}/\text{Num}$ $a',b',c \in \text{Aexp}$

Structural Semantics: Aexps

- Analogous rules apply to the other arithmetic operators
- The rules allow us to evaluate arithmetic expressions step by step
- For example, suppose σ_{12} is a state that maps x to 1 and y to 2
- Then we show the following evaluation sequence

$$\langle (x+5)+y, \sigma_{12}\rangle \Rightarrow \langle (1+5)+y, \sigma_{12}\rangle \Rightarrow \langle 6+y, \sigma_{12}\rangle \Rightarrow \langle 6+2, \sigma_{12}\rangle \Rightarrow \langle 8, \sigma_{12}\rangle \Rightarrow 8$$

as we can construct the following proofs

$$\dfrac{\dfrac{\dfrac{}{\langle x, \sigma_{12}\rangle \Rightarrow \langle 1, \sigma_{12}\rangle}}{\langle x+5, \sigma_{12}\rangle \Rightarrow \langle 1+5, \sigma_{12}\rangle}}{\langle (x+5)+y, \sigma_{12}\rangle \Rightarrow \langle (1+5)+y, \sigma_{12}\rangle}$$

$$\dfrac{\dfrac{}{\langle 1+5, \sigma_{12}\rangle \Rightarrow \langle 6, \sigma_{12}\rangle}}{\langle (1+5)+y, \sigma_{12}\rangle \Rightarrow \langle 6+y, \sigma_{12}\rangle}$$

$$\dfrac{\dfrac{}{\langle y, \sigma_{12}\rangle \Rightarrow \langle 2, \sigma_{12}\rangle}}{\langle 6+y, \sigma_{12}\rangle \Rightarrow \langle 6+2, \sigma_{12}\rangle}$$

$$\dfrac{}{\langle 6+2, \sigma_{12}\rangle \Rightarrow \langle 8, \sigma_{12}\rangle}$$

$$\dfrac{}{\langle 8, \sigma_{12}\rangle \Rightarrow 8}$$

Structural Semantics: Truth Values

- The rules for Booleans are similarly defined with two base cases

$$\textbf{TRUE}\quad \frac{}{\langle \text{true}, \sigma \rangle \Rightarrow tt}$$

$$\textbf{FALSE}\quad \frac{}{\langle \text{false}, \sigma \rangle \Rightarrow ff}$$

- where $\sigma \in \text{State}$

Structural Semantics: Comparison

- Comparing two numerals yields one truth value or another

$$\text{LE-T}\ \frac{}{\langle n_i \le n_j, \sigma\rangle \Rightarrow \langle \text{true}, \sigma\rangle}\ \text{if } i \le j \qquad \text{LE-F}\ \frac{}{\langle n_i \le n_j, \sigma\rangle \Rightarrow \langle \text{false}, \sigma\rangle}\ \text{if } i > j$$

- Otherwise we first reduce the LHS and then the RHS

$$\text{LE-L}\ \frac{\langle a, \sigma\rangle \Rightarrow \langle a', \sigma\rangle}{\langle a \le c, \sigma\rangle \Rightarrow \langle a' \le c, \sigma\rangle} \qquad \text{LE-R}\ \frac{\langle b, \sigma\rangle \Rightarrow \langle b', \sigma\rangle}{\langle n_i \le b, \sigma\rangle \Rightarrow \langle n_i \le b', \sigma\rangle}$$

- where $\sigma \in \text{State}$ $n_i, n_j \in \text{Num}$ $a, b \in \text{Aexp}/\text{Num}$ $a', b', c \in \text{Aexp}$

Structural Semantics: Negation

- Negating basic truth values is easy

$$\text{NEG-F}\quad \frac{}{\langle \neg\text{true}, \sigma\rangle \Rightarrow \langle\text{false}, \sigma\rangle}$$

$$\text{NEG-T}\quad \frac{}{\langle \neg\text{false}, \sigma\rangle \Rightarrow \langle\text{true}, \sigma\rangle}$$

- Otherwise we first need to reduce the argument to a truth value

$$\text{NEG-R}\quad \frac{\langle b, \sigma\rangle \Rightarrow \langle b', \sigma\rangle}{\langle \neg b, \sigma\rangle \Rightarrow \langle \neg b', \sigma\rangle}$$

- In these rules $\sigma \in \text{State}$ $b \in \text{Bexp}/\{\text{true},\text{false}\}$ $b' \in \text{Bexp}$

Structural Semantics: Conjunction

- Conjunction is easy once we have reduced the LHS argument

$$\text{AND-F}\quad \frac{}{\langle \text{false} \wedge c, \sigma\rangle \Rightarrow \langle \text{false}, \sigma\rangle}$$

n.b. Lazy evaluation!

$$\text{AND-T}\quad \frac{}{\langle \text{true} \wedge c, \sigma\rangle \Rightarrow \langle c, \sigma\rangle}$$

- Otherwise we first need to reduce the LHS argument

$$\text{AND-L}\quad \frac{\langle a, \sigma\rangle \Rightarrow \langle a', \sigma\rangle}{\langle a \wedge c, \sigma\rangle \Rightarrow \langle a' \wedge c, \sigma\rangle}$$

- In these rules $\sigma \in \text{State}$ $a \in \text{Bexp}/\{\text{true},\text{false}\}$ $a', c \in \text{Bexp}$

Structural Semantics: Bexps

- Analogous rules apply to the other Boolean operators
- The rules allow us to evaluate Boolean expressions step by step
- For example, suppose σ_{12} is a state that maps x to 1 and y to 2
- Then we show the following derivation sequence

$$\langle \neg(x \leq 5), \sigma_{12} \rangle \Rightarrow \langle \neg \mathrm{true}, \sigma_{12} \rangle \Rightarrow \langle \mathrm{false}, \sigma_{12} \rangle \Rightarrow \mathrm{ff}$$

- As we can construct the following proofs

$$\dfrac{\dfrac{\dfrac{}{\langle x, \sigma_{12} \rangle \Rightarrow \langle 1, \sigma_{12} \rangle}}{\langle x \leq 5, \sigma_{12} \rangle \Rightarrow \langle 1 \leq 5, \sigma_{12} \rangle}}{\langle \neg(x \leq 5), \sigma_{12} \rangle \Rightarrow \langle \neg \mathrm{true}, \sigma_{12} \rangle} \qquad \dfrac{}{\langle \neg \mathrm{true}, \sigma_{12} \rangle \Rightarrow \langle \mathrm{false}, \sigma_{12} \rangle} \qquad \dfrac{}{\langle \mathrm{false}, \sigma_{12} \rangle \Rightarrow \mathrm{ff}}$$

Structural Semantics: Skip p33

- If we are left with a skip command then simply return the state

$$\textbf{SKIP}\quad \frac{}{\langle \text{skip}, \sigma \rangle \Rightarrow \sigma}$$

Structural Semantics: Assign p33

- For assignment we could use denotation of Aexps (as in the book)

$$\frac{}{\langle x:=a, \sigma \rangle \Rightarrow \sigma[\, x \mapsto \mathcal{A}[[a]]\, \sigma \,]}$$

- But here we prefer to avoid mixing up the different semantics by using our own structural semantics of Aexps instead

$$\textbf{ASS-N}\ \frac{\langle n_i , \sigma \rangle \Rightarrow i}{\langle x:=n_i , \sigma \rangle \Rightarrow \sigma[x \mapsto i]} \qquad \textbf{ASS-R}\ \frac{\langle a , \sigma \rangle \Rightarrow \langle a' , \sigma \rangle}{\langle x:=a, \sigma \rangle \Rightarrow \langle x:=a', \sigma \rangle}$$

- Note that the premises indicate if we get a complete configuration (left rule) or an incomplete configuration (right rule) so the types are implicit here ($n_i \in$ Num and $a \in$ Aexp/Num)
- Note that we could have reduced numeral assignment to skip in **ASS-N** to simplify later rules by ensuring only a skip can yield a final state:

$$\frac{}{\langle x:=n_i, \sigma \rangle \Rightarrow \langle \text{skip}, \sigma[x \mapsto i] \rangle}$$

Structural Semantics: Composition p33

- If the first command in a sequence reduces directly to a complete configuration (through assignment or skip) then simply evaluate it and run the second command from that resulting state

$$\textbf{SEQ-R}\quad \frac{\langle S_1, \sigma\rangle \Rightarrow \sigma'}{\langle S_1 ; S_2 , \sigma\rangle \Rightarrow \langle S_2 , \sigma'\rangle}$$

- Otherwise keep reducing the first command until it completes

$$\textbf{SEQ-L}\quad \frac{\langle S_1, \sigma\rangle \Rightarrow \langle S'_1 , \sigma'\rangle}{\langle S_1 ; S_2 , \sigma\rangle \Rightarrow \langle S'_1 ; S_2 , \sigma'\rangle}$$

- Note that if we used the alternative reduction of numeral assignment to skip (at the bottom of the last slide) then the SEQ-R rule above could be replaced by the simpler

$$\frac{}{\langle \mathrm{skip} ; S_2 , \sigma\rangle \Rightarrow \langle S_2 , \sigma\rangle}$$

Structural Semantics: Conditionals p33

- To evaluate a conditional we first reduce the condition

$$\textbf{COND-B}\quad \frac{\langle b, \sigma\rangle \Rightarrow \langle b', \sigma\rangle}{\langle \text{if } b \text{ then } S_1 \text{ else } S_2\ , \sigma\rangle \Rightarrow \langle \text{if } b' \text{ then } S_1 \text{ else } S_2\ , \sigma\rangle}$$

- As when the condition finally yields a truth value, then reducing the conditional is easy

$$\textbf{COND-F}\quad \frac{}{\langle \text{if false then } S_1 \text{ else } S_2\ , \sigma\rangle \Rightarrow \langle S_2 , \sigma\rangle}$$

$$\textbf{COND-T}\quad \frac{}{\langle \text{if true then } S_1 \text{ else } S_2\ , \sigma\rangle \Rightarrow \langle S_1 , \sigma\rangle}$$

Structural Semantics: Loops! p33

- For loops, we cannot follow the same approach as for conditionals (or we will ned up defining an infinite loop!)

- The solution is to unroll the loop one step

$$\text{LOOP}\quad \frac{}{\langle \text{while}\ b\ \text{do}\ S, \sigma\rangle \Rightarrow \langle \text{if}\ b\ \text{then}\ (S; \text{while}\ b\ \text{do}\ S)\ \text{else}\ \text{skip}, \sigma\rangle}$$

Structural Semantics: Stms

- We can show the derivation sequence

$$\langle (z{:=}x;\ x{:=}y);\ y{:=}z, \sigma_{570}\rangle \Rightarrow \langle (z{:=}5;\ x{:=}y);\ y{:=}z, \sigma_{570}\rangle \Rightarrow \langle x{:=}y;\ y{:=}z, \sigma_{575}\rangle$$
$$\Rightarrow \langle x{:=}7;\ y{:=}z, \sigma_{575}\rangle \Rightarrow \langle y{:=}z, \sigma_{775}\rangle \Rightarrow \langle y{:=}5, \sigma_{775}\rangle \Rightarrow \sigma_{755}$$

- using the following proofs

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\langle x, \sigma_{570}\rangle \Rightarrow \langle 5, \sigma_{570}\rangle}}{\langle z{:=}x, \sigma_{570}\rangle \Rightarrow \langle z{:=}5, \sigma_{570}\rangle}}{\langle (z{:=}x;\ x{:=}y), \sigma_{570}\rangle \Rightarrow \langle (z{:=}5;\ x{:=}y, \sigma_{570}\rangle}}{\langle (z{:=}x;\ x{:=}y);\ y{:=}z, \sigma_{570}\rangle \Rightarrow \langle (z{:=}5;\ x{:=}y);\ y{:=}z, \sigma_{570}\rangle}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\langle 5, \sigma_{570}\rangle \Rightarrow 5}}{\langle z{:=}5, \sigma_{570}\rangle \Rightarrow \sigma_{575}}}{\langle (z{:=}5;\ x{:=}y), \sigma_{570}\rangle \Rightarrow \langle x{:=}y, \sigma_{575}\rangle}}{\langle (z{:=}5;\ x{:=}y);\ y{:=}z, \sigma_{570}\rangle \Rightarrow \langle x{:=}y;\ y{:=}z, \sigma_{575}\rangle}$$

$$\dfrac{\dfrac{\dfrac{}{\langle y, \sigma_{575}\rangle \Rightarrow \langle 7, \sigma_{575}\rangle}}{\langle x{:=}y, \sigma_{575}\rangle \Rightarrow \langle x{:=}7, \sigma_{575}\rangle}}{\langle x{:=}y;\ y{:=}z, \sigma_{575}\rangle \Rightarrow \langle x{:=}7;\ y{:=}z, \sigma_{575}\rangle}$$

$$\dfrac{\dfrac{\dfrac{}{\langle 7, \sigma_{575}\rangle \Rightarrow 7}}{\langle x{:=}7, \sigma_{575}\rangle \Rightarrow \sigma_{775}}}{\langle x{:=}7;\ y{:=}z, \sigma_{575}\rangle \Rightarrow \langle y{:=}z, \sigma_{775}\rangle}$$

$$\dfrac{\dfrac{}{\langle z, \sigma_{775}\rangle \Rightarrow \langle 5, \sigma_{775}\rangle}}{\langle y{:=}z, \sigma_{775}\rangle \Rightarrow \langle y{:=}5, \sigma_{775}\rangle}$$

$$\dfrac{\dfrac{}{\langle 5, \sigma_{775}\rangle \Rightarrow 5}}{\langle y{:=}5, \sigma_{775}\rangle \Rightarrow \sigma_{755}}$$

Derivation Sequences p32-3

- A configuration γ can have one of two forms:
  - Either it is an *incomplete* (or *intermediate*) configuration $\gamma = \langle S , \sigma \rangle$
  - Or it is a *terminal* (or *complete*) configuration of the form $\gamma = \sigma$
- An incomplete configuration γ can have one of two properties:
  - Either it is *stuck* if there is no γ' such that $\gamma \Rightarrow \gamma'$
  - Or it is *unstuck* if there is some γ' such that $\gamma \Rightarrow \gamma'$
- A derivation sequence from $\langle S,\sigma \rangle$ can have one of two forms
  - Either it is a *finite* sequence $\gamma_0, \gamma_1, \ldots, \gamma_n$ such that $\gamma_0=\langle S,\sigma \rangle$ and $\gamma_i \Rightarrow \gamma_{i+1}$ for all $0 \leq i \leq n-1$ and γ_n is a terminal or stuck configuration
  - Or it is an *infinite* sequence $\gamma_0, \gamma_1, \gamma_2, \ldots$ such that $\gamma_0=\langle S,\sigma \rangle$ and $\gamma_i \Rightarrow \gamma_{i+1}$ for all $0 \leq i$
- We write $\gamma \Rightarrow^k \gamma'$ to denote that γ' can be obtained from γ in *exactly* k steps using the transition relation $\Rightarrow$
- We write $\gamma \Rightarrow^* \gamma'$ to denote that γ' can be obtained from γ in some *finite* number of steps using the transition relation $\Rightarrow$

Termination and Looping p36

- The execution of statement S in state σ *terminates* iff there exists a finite derivation sequence from $\langle S , \sigma \rangle$
- The execution of statement S in state σ *loops* iff there exists an infinite derivation sequence from $\langle S , \sigma \rangle$
- A statement S always *terminates* iff its execution terminates in all states σ
- A statement S always *loops* iff its execution loops in all states σ
- An execution terminates *successfully* iff it ends with a terminal configuration

Determinism and Equivalence p38-9

- A structural operational semantics is strongly deterministic iff $\langle S,\sigma\rangle \Rightarrow \gamma$ and $\langle S,\sigma\rangle \Rightarrow \gamma'$ imply that $\gamma = \gamma'$ for all S, σ, γ, γ'

- A structural operational semantics is weakly deterministic iff $\langle S,\sigma\rangle \Rightarrow^* \sigma'$ and $\langle S,\sigma\rangle \Rightarrow^* \sigma''$ imply that $\sigma' = \sigma''$ for all S, σ, σ', σ''

- Two statements S_1 and S_2 are *semantically equivalent* (under the structural semantics) whenever it holds that for all states σ
  - $\langle S_1, \sigma\rangle \Rightarrow^* \gamma$ iff $\langle S_2, \sigma\rangle \Rightarrow^* \gamma$ whenever is γ stuck or terminal
  - There is an infinite derivation sequence from $\langle S_1, \sigma\rangle$ iff there is an infinite derivation from $\langle S_2, \sigma\rangle$