UNIVERSITY OF Southampton
School of Electronics and Computer Science

COMP2212
PROGRAMMING LANGUAGE CONCEPTS

Julian Rathke and Pawel Sobocinski

OPERATIONAL SEMANTICS

OPERATIONAL SEMANTICS

- An alternative approach to semantics is to build a inductive binary relation between terms of the language.
- We call this approach **operational semantics**
- There are two flavours of operational semantics: **big step** and **small step**
- In big step semantics the binary relation is between terms and values. It represents the values that a term can evaluate to.
- We typically write $E \Downarrow V$ to mean program E evaluates to value V.
  - The meaning of a program is given by the values it can evaluate to
  - Modelling the run time environment (e.g. a heap) is quite straightforward in this approach by defining the relation between run time states and values.
  - The program operators are often easily specified in terms of "what they do" rather than "what they mean".
- One disadvantage of this approach is that it still doesn't account for the effects of non-terminating programs very easily.

BIG STEP TOY SEMANTICS

- Let's give an inductive relation for the big-step semantics for the Toy language:
- The form of the relation will be $E \Downarrow V$ "expression E evaluates to value V"

$$\frac{}{\mathsf{n} \Downarrow \mathsf{n}} \qquad \frac{}{\mathsf{b} \Downarrow \mathsf{b}} \qquad \frac{}{\lambda(x:T)E \Downarrow \lambda(x:T)E}$$

$$\frac{E_1 \Downarrow \mathsf{n} \quad E_2 \Downarrow \mathsf{m} \quad n < m}{E_1 < E_2 \Downarrow \mathsf{true}} \qquad \frac{E_1 \Downarrow \mathsf{n} \quad E_2 \Downarrow \mathsf{m} \quad n \not< m}{E_1 < E_2 \Downarrow \mathsf{false}}$$

Assume that each Toy literal n corresponds to a mathematical *n*

$$\frac{E_1 \Downarrow \mathsf{n} \quad E_2 \Downarrow \mathsf{m} \quad n + m = n'}{E_1 + E_2 \Downarrow \mathsf{n}'}$$

$$\frac{E_1 \Downarrow \mathsf{true} \quad E_2 \Downarrow V}{\mathsf{if}\ E_1\ \mathsf{then}\ E_2\ \mathsf{else}\ E_3 \Downarrow V} \qquad \frac{E_1 \Downarrow \mathsf{false} \quad E_3 \Downarrow V}{\mathsf{if}\ E_1\ \mathsf{then}\ E_2\ \mathsf{else}\ E_3 \Downarrow V}$$

BIG STEP TOY SEMANTICS

IN EXAM

- Let's give an inductive relation for the big-step semantics for the Toy language:
- The form of the relation will be $E \Downarrow V$ "expression E evaluates to value V"

V replaced by x

$$\frac{E_1 \Downarrow V \quad E_2[V/x] \Downarrow V'}{\textsf{let } (x : T) \;=\; E_1 \textsf{ in } E_2 \Downarrow V'}$$

We need to define what the substitution E [V / x] means exactly

$$\frac{E_1 \Downarrow \lambda(x:T)E_1' \quad E_2 \Downarrow V_2 \quad E_1'[V_2/x] \Downarrow V'}{E_1 E_2 \Downarrow V'}$$

Big Step semantics still don't model the sequence of computation steps a program may take though - this can be problematic when modelling e.g. concurrency.

SMALL STEP OPERATIONAL SEMANTICS

- In contrast, small step operational semantics are given by an inductive relation between terms representing run time states of programs.
- Run time state includes the heap, stack, **program counter** etc.
- We typically represent changing program counters as changing terms of a language. For example we write E → E' for our reduction relation.
- This means, program state E evaluates in 'one' step of evaluation to program state E'
- By considering the evaluation of a program step-by-step then we can see clearly how a program behaves.
- To determine whether a program calculates a given return value then we repeatedly follow the single steps of evaluation until a value is reached.
- non-termination is an activity (infinite sequence of steps) rather than failure to calculate a value.
  - useful for analysing at what point programs begin to diverge and what effects they may have during divergence.
- Let's write a small-step semantics for the Toy language.

SMALL STEP TOY SEMANTICS

The form of this relation is E ➞ E' this represents a single step of computation of program state E reach the run time state E' (which can be represented as another expression).

$$\frac{n < m}{\mathsf{n} < \mathsf{m} \rightarrow \mathsf{true}} \qquad \frac{n \not< m}{\mathsf{n} < \mathsf{m} \rightarrow \mathsf{false}}$$

$$\frac{E \rightarrow E'}{\mathsf{n} < E \rightarrow \mathsf{n} < E'} \qquad \frac{E_1 \rightarrow E'}{E_1 < E_2 \rightarrow E' < E_2}$$

$$\frac{n + m = n'}{\mathsf{n} + \mathsf{m} \rightarrow \mathsf{n}'} \qquad \frac{E \rightarrow E'}{\mathsf{n} + E \rightarrow \mathsf{n} + E'} \qquad \frac{E_1 \rightarrow E'}{E_1 + E_2 \rightarrow E' + E_2}$$

$$\frac{}{\mathsf{if\ true\ then}\ E_2\ \mathsf{else}\ E_3 \rightarrow E_2} \qquad \frac{}{\mathsf{if\ false\ then}\ E_2\ \mathsf{else}\ E_3 \rightarrow E_3}$$

$$\frac{E_1 \rightarrow E'}{\mathsf{if}\ E_1\ \mathsf{then}\ E_2\ \mathsf{else}\ E_3 \rightarrow \mathsf{if}\ E'\ \mathsf{then}\ E_2\ \mathsf{else}\ E_3}$$

SMALL STEP TOY SEMANTICS CONTINUED

The form of this relation is E ➞ E' this represents a single step of computation of program state E reach the run time state E' (which can be represented as another expression).

$$\frac{}{\textsf{let}\ (x : T)\ =\ V\ \textsf{in}\ E_2 \rightarrow E_2[V/x]}$$

$$\frac{E_1 \rightarrow E'}{\textsf{let}\ (x : T)\ =\ E_1\ \textsf{in}\ E_2 \rightarrow \textsf{let}\ (x : T)\ =\ E'\ \textsf{in}\ E_2}$$

$$\frac{}{\lambda(x : T)E_1\ V \rightarrow E_1[V/x]}$$

$$\frac{E_2 \rightarrow E'}{\lambda(x : T)E_1\ E_2 \rightarrow \lambda(x : T)E_1\ E'}$$

$$\frac{E_1 \rightarrow E'}{E_1\ E_2 \rightarrow E'\ E_2}$$

EXAMPLE AND BIG STEP PROOF TREE

- We'll take an example Toy program and consider how to evaluate it using big step semantics, and then with small step semantics.
- The example is :

```
let (x : Int) =
  if (10 < 3) then 0 else (10 + 1)
in x + 42
```

- The inductive rules in the big step semantics form a proof tree to justify the final conclusion.
- This tree is as follows

$$\dfrac{\dfrac{\dfrac{\dfrac{}{10 \Downarrow 10} \quad \dfrac{}{3 \Downarrow 3}}{(10 < 3) \Downarrow \mathsf{false}} \qquad \dfrac{\dfrac{}{10 \Downarrow 10} \quad \dfrac{}{1 \Downarrow 1}}{(10 + 1) \Downarrow 11}}{\mathsf{if}\ (10 < 3)\ \mathsf{then}\ 0\ \mathsf{else}\ (10 + 1)\ \Downarrow 11} \qquad \dfrac{\dfrac{}{11 \Downarrow 11} \quad \dfrac{}{42 \Downarrow 42}}{(x + 42)[11/x] \Downarrow 53}}{\mathsf{let}\ (x : T)\ =\ \mathsf{if}\ (10 < 3)\ \mathsf{then}\ 0\ \mathsf{else}\ (10 + 1)\ \mathsf{in}\ x + 42 \Downarrow 53}$$

SMALL STEP PROOF TREES

For our example, small step semantics requires five evaluation steps to reach the value 53. **Each** single step is given by a proof tree that justifies it.

$$\dfrac{\dfrac{\overline{10 < 3 \to \mathsf{false}}}{\mathsf{if}\ (10 < 3)\ \mathsf{then}\ 0\ \mathsf{else}\ (10+1) \to \mathsf{if\ false\ then}\ 0\ \mathsf{else}\ (10+1)}}{\mathsf{let}\ (x : T) = \mathsf{if}\ (10 < 3)\ \mathsf{then}\ 0\ \mathsf{else}\ (10+1)\ \mathsf{in}\ x + 42 \to \mathsf{let}\ (x : T) = \mathsf{if\ false\ then}\ 0\ \mathsf{else}\ (10+1)\ \mathsf{in}\ x + 42}$$

Tree for Step 1

$$\dfrac{\overline{\mathsf{if\ false\ then}\ 0\ \mathsf{else}\ (10+1) \to (10+1)}}{\mathsf{let}\ (x : T) = \mathsf{if\ false\ then}\ 0\ \mathsf{else}\ (10+1)\ \mathsf{in}\ x + 42 \to \mathsf{let}\ (x : T) = (10+1)\ \mathsf{in}\ x + 42}$$

Tree for Step 2

$$\dfrac{\overline{10 + 1 \to 11}}{\mathsf{let}\ (x : T) = (10+1)\ \mathsf{in}\ x + 42 \to \mathsf{let}\ (x : T) = 11\ \mathsf{in}\ x + 42}$$

Tree for Step 3

$$\overline{\mathsf{let}\ (x : T) = 11\ \mathsf{in}\ x + 42 \to (x + 42)[11/x]}$$

Step 4

$$\overline{(x + 42)[11/x] \to 53}$$

Step 5

SMALL STEP PROOF TREES

For our example, small step semantics requires five evaluation steps to reach the value 53. **Each** single step is given by a proof tree that justifies it.

We sometimes write this sequence of steps without showing the proof trees..

$$\textsf{let } (x : T) \; = \; \textsf{if } (10 < 3) \textsf{ then } 0 \textsf{ else } (10 + 1) \textsf{ in } x + 42$$

$$\rightarrow$$

$$\textsf{let } (x : T) \; = \; \textsf{if false then } 0 \textsf{ else } (10 + 1) \textsf{ in } x + 42$$

$$\rightarrow$$

$$\textsf{let } (x : T) \; = \; (10 + 1) \textsf{ in } x + 42$$

$$\rightarrow$$

$$\textsf{let } (x : T) \; = \; 11 \textsf{ in } x + 42$$

$$\rightarrow$$

$$11 + 42$$

$$\rightarrow$$

$$53$$

RELATING SMALL STEP AND BIG STEP SEMANTICS

- Ideally, if we have defined our semantics correctly, then we should have a strong relationship between the big step and small step semantics.
- They should specify the same behaviours - albeit in different ways.
- To formalise this we first define the following: $E \rightarrow^{*} E'$
- if and only if there exists a (possibly empty) sequence
- $$E = E_1 \rightarrow E_2 \rightarrow \ \ldots \rightarrow E_n = E'$$
- We can prove the following Theorem for the Toy language semantics.

Theorem For all E ,V :

$$E \Downarrow V \text{ if and only if } E \rightarrow^{*} V$$

The proof of this is a straightforward proof by induction over the big step and small step derivation trees.

NEXT LECTURE: TYPE SAFETY