

Computer Science 203
 Programming Languages
 Fall 2004 - Lecture 6

Today's Challenge:
 Develop a small-step semantics for IMP

Cormac Flanagan
 University of California, Santa Cruz

Review: Big-Step vs. Small-Step Operational Semantics

- Big-Step Operational Semantics: $e \Downarrow n$
 - Judgment $e \Downarrow n$ means that e evaluates to n
 - In one, big step, all the way to a result
 - Hard to talk about
 - commands that do not terminate.
 - intermediate states.
- Small-Step Operational Semantics: $e \rightarrow e'$
 - describe a single step in the evaluation
 - many steps may be needed to get a result

Review: Small-Step Evaluation Rules for ARITH

n is the sum of n_1 and n_2 n is the product of n_1 and n_2
 $n_1 + n_2 \rightarrow n$ $n_1 * n_2 \rightarrow n$

$$\frac{e_1 \rightarrow e_1'}{e_1 + e_2 \rightarrow e_1' + e_2} \qquad \frac{e_2 \rightarrow e_2'}{n_1 + e_2 \rightarrow n_1 + e_2'}$$

$$\frac{e_1 \rightarrow e_1'}{e_1 * e_2 \rightarrow e_1' * e_2} \qquad \frac{e_2 \rightarrow e_2'}{n_1 * e_2 \rightarrow n_1 * e_2'}$$

- Example: $(3 + 4) + 5 \rightarrow 7 + 5 \rightarrow 12$

Review: Contextual semantics for ARITH

- Contextual semantics: a small-step semantics with
 - evaluation contexts - where to evaluate (bit like a prog. counter)
 - evaluation rules - how to evaluate

- Evaluation contexts for ARITH

$H ::= \bullet \mid H + e \mid n + H \mid H * e \mid n * H$

n is the sum of n_1 and n_2 n is the product of n_1 and n_2
 $H[n_1 + n_2] \rightarrow H[n]$ $H[n_1 * n_2] \rightarrow H[n]$

Refactoring the Contextual semantics for ARITH

- Refactor semantics as local + global reduction rules
- Evaluation contexts for ARITH (same)

$H ::= \bullet \mid H + e \mid n + H \mid H * e \mid n * H$

- Local reduction rules

$n_1 + n_2 \rightarrow n$ where $n = n_1$ plus n_2
 $n_1 * n_2 \rightarrow n$ where $n = n_1$ times n_2

- Global reduction rule

$H[e] \rightarrow H[e']$ provided $e \rightarrow e'$

- Example: evaluate $(3 + 4) + 5$
 - decompose: $(3 + 4) + 5$ is $H[3+4]$ where $H = \bullet + 5$
 - local rule: $3 + 4 \rightarrow 7$
 - global rule: $H[3+4] \rightarrow H[7]$
 - i.e. $(3 + 4) + 5 \rightarrow 7 + 5$

Contextual, Small-Step Semantics for IMP

Contextual, Small-Step Semantics for IMP

- Each execution step is a rewrite of program plus store
- We will define a relation $\langle c, \sigma \rangle \rightarrow \langle c', \sigma' \rangle$
 - c' is obtained from c through an atomic rewrite step.
 - $\langle x := 2+8, \sigma \rangle \rightarrow \langle x := 10, \sigma \rangle \rightarrow \langle \text{skip}, \sigma[x:=10] \rangle$
- Terminal command: "skip"
 - evaluation stops here
 - evaluation continues until we get to "skip"
 - may never reach skip - eg: while true do skip

CMPS201 Lecture 6

7

Review: IMP Syntax

```

e ::= n
    | x
    | e1 + e2    also -, *

b ::= true
    | false
    | e1 = e2    also <
    | ¬ b
    | b1 ∧ b2    also ∨

c ::= skip
    | x := e
    | c1 ; c2
    | if b then c1 else c2
    | while b do c
  
```

CMPS201 Lecture 6

8

Local Reduction Rules for IMP

- Local reduction rules specify what can be evaluated in one step: $\langle c, \sigma \rangle \rightarrow \langle c', \sigma' \rangle$ (ditto for expressions)

```

⟨x, σ⟩ → ⟨σ(x), σ⟩
⟨n1 + n2, σ⟩ → ⟨n, σ⟩           where n = n1 + n2
⟨n1 = n2, σ⟩ → ⟨true, σ⟩         if n1 = n2
⟨x := n, σ⟩ → ⟨skip, σ[x := n]⟩
⟨skip; c, σ⟩ → ⟨c, σ⟩
⟨if true then c1 else c2, σ⟩ → ⟨c1, σ⟩
⟨if false then c1 else c2, σ⟩ → ⟨c2, σ⟩
⟨while b do c, σ⟩ → ⟨if b then c; while b do c else skip, σ⟩
  
```

CMPS201 Lecture 6

9

Redexes

- A redex is a syntactic expression or command that can be reduced by a local reduction rule
- For brevity, we omit some redexes
- Redexes are defined by a grammar:

```

r ::= x
    | n1 + n2
    | x := n
    | skip; c
    | if true then c1 else c2
    | if false then c1 else c2
    | while b do c
  
```

- Note that $(1 + 3) + 2$ is not a redex, but $1 + 3$ is.

CMPS201 Lecture 6

10

Review

- A *redex* is something that can be reduced in one step
 - E.g. $2+8$
- Local reduction rules reduce these redexes
 - E.g. $\langle 2+8, \sigma \rangle \rightarrow \langle 10, \sigma \rangle$
- Next: *global reduction rules*
- Consider
 - $\langle x := 1+(2+8), \sigma \rangle$
 - $\langle \text{while false do } x := 1+(2+8), \sigma \rangle$
- Should we also reduce $2+8$ in these cases?

CMPS201 Lecture 6

11

IMP Syntax

```

e ::= n
    | x
    | e1 + e2    also -, *

b ::= true
    | false
    | e1 = e2    also <
    | ¬ b
    | b1 ∧ b2    also ∨

c ::= skip
    | x := e
    | c1 ; c2
    | if b then c1 else c2
    | while b do c
  
```

IMP Evaluation Contexts

```

H ::= •
    | H + e
    | n + H
    | H = e
    | n = H
    | ¬ H
    | H ∧ b
    | p ∧ H

| x := H
| H ; c
| if H then c1 else c2
  
```

CMPS201 Lecture 6

12

Evaluation Contexts

Examples

- $x := 1 + \bullet$
 - Fill context H with $2+8$ to yield $H[2+8] = x := 1+(2+8)$
 - Or fill context with 10 to yield $H[10] = x := 1+10$
- •
- NOT: while false do $x := 1 + \bullet$
- NOT: while false do •

Contexts: Notes

- Evaluation contexts say how to find the next redex:
 - Consider $e_1 + e_2$ and its decomposition as $H[r]$.
 - If e_1 is n_1 and e_2 is n_2
 - then $H = \bullet$ and $r = n_1 + n_2$.
 - If e_1 is n_1 and e_2 is not n_2
 - then $H = n_1 + H_2$ and $e_2 = H_2[r]$.
 - If e_1 is not n_1
 - then $H = H_1 + e_2$ and $e_1 = H_1[r]$.
- In the last two cases decomposition is done recursively.

The Global Reduction Rule

- General idea of the contextual semantics:
 - Decompose the current expression into
 - the next redex r
 - and an evaluation context H (the remaining program).
 - Reduce the redex " r " to some other expression " e ".
 - Put " e " back into the original context, yielding $H[e]$.
- Formalized as a small step rule:

If $\langle r, \sigma \rangle \rightarrow \langle e, \sigma' \rangle$ then $\langle H[r], \sigma \rangle \rightarrow \langle H[e], \sigma' \rangle$

The Global Reduction Rule: Example

- Consider the command $x := 1+(2+8)$
- Split into an evaluation context H and a redex r
- Get
 - $H = x := 1 + \bullet$
 - $r = 2+8$
 - $H[r] = x := 1+(2+8)$ (original command)
- Have
 - $\langle 2+8, \sigma \rangle \rightarrow \langle 10, \sigma \rangle$ (local reduction rule)
- Define global reduction
 - $\langle H[2+8], \sigma \rangle \rightarrow \langle H[10], \sigma \rangle$ or, equivalently
 - $\langle x := 1+(2+8), \sigma \rangle \rightarrow \langle x := 1+10, \sigma \rangle$

Contextual Semantics: Example

- Consider the small-step evaluation of $x := 1; x := x + 1$ in the initial state $[x := 0]$

State	Context	Redex
$\langle x := 1; x := x + 1, [x := 0] \rangle$	•; $x := x + 1$	$x := 1$
$\langle \text{skip}; x := x + 1, [x := 1] \rangle$	•	$\text{skip}; x := x + 1$
$\langle x := x + 1, [x := 1] \rangle$	$x := \bullet + 1$	x
$\langle x := 1 + 1, [x := 1] \rangle$	$x := \bullet$	$1 + 1$
$\langle x := 2, [x := 1] \rangle$	•	$x := 2$
$\langle \text{skip}, [x := 2] \rangle$		

Normal vs Short-Circuit Boolean Operators

- What if we want normal evaluation of \wedge ?
 - Define the following contexts, redexes, and local rules:
 - $H ::= \dots \mid H \wedge b_2 \mid p_1 \wedge H$
 - $r ::= \dots \mid p_1 \wedge p_2$
 - $\langle p_1 \wedge p_2, \sigma \rangle \rightarrow \langle p, \sigma \rangle$ where $p = p_1 \wedge p_2$
- What if we want short-circuit evaluation of \wedge ?
 - Define the following contexts, redexes, and local rules:
 - $H ::= \dots \mid H \wedge b_2$
 - $r ::= \dots \mid \text{true} \wedge b_2 \mid \text{false} \wedge b_2$
 - $\langle \text{true} \wedge b_2, \sigma \rangle \rightarrow \langle b_2, \sigma \rangle$
 - $\langle \text{false} \wedge b_2, \sigma \rangle \rightarrow \langle \text{false}, \sigma \rangle$
 - The local reduction kicks in before b_2 is evaluated.

Some Further Topics

- Treatment of errors in operational semantics
 - with an explicit "error" result, as in $(3/0) \rightarrow \text{error}$,
 - with an "error" expression, as in $(3 + \text{error})$,
 - with "stuck" computations, so $(3/0) \rightarrow r$ for no r .

CMPS201 Lecture 6

19

Contextual Semantics: Notes

- For example: $c = c_1; c_2$
 - either $c_1 = \text{skip}$ and then $c = H[\text{skip}; c_2]$ with $H = \bullet$
 - or $c_1 \neq \text{skip}$ and then $c_1 = H[r_1]$;
so $c = c_1; c_2 = H[r_1]; c_2 = H'[r_1]$ where $H' = H; c_2$
- For example: $c = \text{if } b \text{ then } c_1 \text{ else } c_2$
 - either $b = \text{true}$ or $b = \text{false}$ and then $c = H[r]$ with $H = \bullet$
 - or b is not a value and $b = H[r]$;
so $c = H'[r]$ where $H' = \text{if } H \text{ then } c_1 \text{ else } c_2$
- Decomposition theorem: If c is not "skip" then there exist unique H and r such that c is $H[r]$.
 \Rightarrow Progress and determinism.

CMPS201 Lecture 6

20

Summary of Operational Semantics

- Precise specification of dynamic semantics:
 - order of evaluation (or that it doesn't matter)
 - error conditions (sometimes implicitly, by rule applicability)
- Simple and abstract (cf. implementations)
 - no low-level details such as stack and memory management, data layout, etc.
- Often not compositional (as for while)
- Basis for some important proofs about languages
 - e.g. type soundness
- Basis for some reasoning about particular programs
- Point of reference for other semantics

CMPS201 Lecture 6

21

Homework 4 (for October 18)

Consider a variant of ARITH where integers can be between 0 and $2^{64}-1$ only, and the arithmetic operations yield an "OVERFLOW" error rather than results larger than $2^{64}-1$ (and otherwise behave as usual).

1. Revise the abstract syntax of the language.
Note that the "overflow" error is not an expression of the language.
2. Revise the big-step operational semantics, using judgments of the form $e \Downarrow v$ where e is an expression and v is either an integer n or the new result OVERFLOW.
3. Write a contextual semantics for this variant of ARITH.
In other words, define redexes, local reduction rules, contexts, and global reduction rules.
Hint: think about the "type" of \rightarrow .
What kinds of things should be on the left and right of this relation?
4. Turn in on paper at start of class Tuesday October 18th.

Please attempt before Thursday, so we can discuss in class if need be.

CMPS201 Lecture 6

22

Project Proposals

- Project proposals due Thursday October 20th
- One (or two) page document describing
 - your proposed project
 - time budget (how much work is it?)
 - timeline (when will you do which parts?)
 - risks (always consider - and mitigate - risks!)
- Office hours
 - next week Jessica has office hours at 11am instead of 10am.
- Next class
 - axiomatic semantics
 - read Winskel Chapter 6

CMPS201 Lecture 6

23