CS 476 – Programming Language Design

William Mansky

Questions

Nobody has responded yet.

Hang tight! Responses are coming in.

Language #1: Expressions

- Simple arithmetic and boolean operations
- Every term computes to a *value*, either int or bool

- Arithmetic operators: plus, minus, times
- Boolean operators: and, or, not, comparison, if-then-else

- 3 + 5 * 9 should compute to 48
- if 1 = 0 or 1 = 1 then 2 else 4 should compute to 2

Expressions: Syntax

```
E ::= <#>
    | E + E | E – E | E * E
    | <bool>
    | E and E | E or E
    | not E
    | E = E
    | if E then E else E
```

```
type exp = Num of int
    | Add of exp * exp | …
    | Bool of bool
    | And of exp * exp | …
    | Not of exp
    | Eq of exp * exp
    | If of exp * exp * exp
```

Expressions: Interpreter with Errors

```
let rec eval (e : exp) : retval option =
  match e with
   | …
   | Bool b -> Some (BoolVal b)
   | And (e1, e2) ->
      (match eval e1, eval e2 with
       | Some (BoolVal b1), Some (BoolVal b2) ->
                  Some (BoolVal (b1 && b2))
       | _, _ -> None)
```

Expressions: Types

- Types: int, bool
- Rules:

$$\frac{(n \text{ is a number literal})}{n : \text{int}}$$

$$\frac{e_1 : \text{int} \quad e_2 : \text{int}}{e_1 \mathbin{\textbf{+}} e_2 : \text{int}}$$

$$\frac{(b \text{ is a boolean literal})}{b : \text{bool}}$$

$$\frac{e_1 : \text{bool} \quad e_2 : \text{bool}}{e_1 \ \textbf{and} \ e_2 : \text{bool}}$$

$$\frac{e_1 : \tau \quad e_2 : \tau}{e_1 \mathbin{\textbf{=}} e_2 : \text{bool}}$$

$$\frac{e : \text{bool} \quad e_1 : \tau \quad e_2 : \tau}{\texttt{if}\ e\ \texttt{then}\ e_1\ \texttt{else}\ e_2 : \tau}$$

Structure of a language

- Syntax
  - Concrete: what do programs look like?
  - Abstract: what are the pieces of a program?
- Semantics
  - Static: which programs make sense?
  - ➢ Dynamic: what do programs do when we run them?
- Pragmatics
  - Implementation: how can we actually make the semantics happen?
  - IDE, tool support, etc.

Big-Step Operational Semantics

- Describe how expressions compute to values
- $e \Downarrow v$ means "expression e evaluates to value v"
- Roughly the same structure as an interpreter
- Defined by a system of inference rules

Expressions: Big-Step Semantics

Interpreter

Semantics

Num i -> IntVal i

$$\frac{(i \text{ is a number literal})}{i \Downarrow i}$$

Add (e1, e2) -> eval e1 + eval e2

(more or less)

$$\frac{e_1 \Downarrow i_1 \quad e_2 \Downarrow i_2 \quad (i = i_1 + i_2)}{e_1 \mathbf{+} e_2 \Downarrow i}$$

$$\left(\frac{e_1 \Downarrow i_1 \quad e_2 \Downarrow i_2}{\text{Add}\,(e_1, e_2) \Downarrow (i_1 + i_2)}\right)$$

Expressions: Big-Step Semantics

Interpreter

Num i -> IntVal i

Add (e1, e2) -> eval e1 + eval e2

(more or less)

Semantics

$$\frac{(i \text{ is a number literal})}{i \Downarrow i}$$

$$\frac{e_1 \Downarrow i_1 \quad e_2 \Downarrow i_2 \quad (i = i_1 + i_2 \land i < \text{MAX\_INT})}{e_1 \mathbf{+} e_2 \Downarrow i}$$

to say that addition is only defined when the sum doesn't overflow

Expressions: Big-Step Semantics

$$\frac{(i \text{ is a number literal})}{i \Downarrow i} \qquad \frac{e_1 \Downarrow i_1 \quad e_2 \Downarrow i_2 \quad (i = i_1 + i_2)}{e_1 \mathbin{\textbf{+}} e_2 \Downarrow i}$$

$$\frac{(b \text{ is a boolean literal})}{b \Downarrow b} \qquad \frac{e_1 \Downarrow b_1 \quad e_2 \Downarrow b_2 \quad (b = b_1 \mathbin{\&\&} b_2)}{e_1 \textbf{ and } e_2 \Downarrow b}$$

$$\frac{e_1 \Downarrow i_1 \quad e_2 \Downarrow i_2 \quad (b \text{ is true iff } i_1 \text{ is the same as } i_2)}{e_1 \mathbin{\textbf{=}} e_2 \Downarrow b}$$

- Exercise: Write one or more rules for evaluating if-expressions **`if`** e **`then`** e_1 **`else`** e_2.

Expressions: Big-Step Semantics

$$\frac{(i \text{ is a number literal})}{i \Downarrow i}$$

$$\frac{e_1 \Downarrow i_1 \quad e_2 \Downarrow i_2 \quad (i = i_1 + i_2)}{e_1 \texttt{ + } e_2 \Downarrow i}$$

$$\frac{e_1 \Downarrow i_1 \quad e_2 \Downarrow i_2 \quad (b \text{ is true iff } i_1 \text{ is the same as } i_2)}{e_1 \texttt{ = } e_2 \Downarrow b}$$

$$\frac{e \Downarrow \text{true} \quad e_1 \Downarrow v_1}{\texttt{if } e \texttt{ then } e_1 \texttt{ else } e_2 \Downarrow v_1}$$

$$\frac{e \Downarrow \text{false} \quad e_2 \Downarrow v_2}{\texttt{if } e \texttt{ then } e_1 \texttt{ else } e_2 \Downarrow v_2}$$

Expressions: Big-Step Semantics

$$\frac{(i \text{ is a number literal})}{i \Downarrow i} \qquad \frac{e_1 \Downarrow i_1 \quad e_2 \Downarrow i_2 \quad (i = i_1 + i_2)}{e_1 \mathbf{+}\, e_2 \Downarrow i}$$

$$\frac{e_1 \Downarrow i_1 \quad e_2 \Downarrow i_2 \quad (b \text{ is true iff } i_1 \text{is the same as } i_2)}{e_1 \mathbf{=}\, e_2 \Downarrow b}$$

$$\frac{e \Downarrow b \quad (\text{if } b \text{ then } e_1 \text{ else } e_2) \Downarrow v}{\texttt{if}\ e\ \texttt{then}\ e_1\ \texttt{else}\ e_2 \Downarrow v}$$

Questions

Nobody has responded yet.

Hang tight! Responses are coming in.

Big-Step Inference Rules

$$\frac{e_1 : \text{int} \quad e_2 : \text{int}}{e_1 \mathbf{+} e_2 : \text{int}} \qquad \frac{e_1 \Downarrow i_1 \quad e_2 \Downarrow i_2 \quad (i = i_1 + i_2)}{e_1 \mathbf{+} e_2 \Downarrow i}$$

- *Premises* that must be true for the rule to apply
- Below the line: *conclusion* that we learn when the rule applies
- Contain both fixed symbols ($\mathbf{+}, \mathbf{int}$) and *metavariables* (e_1, e_2)
  - Rule applies for any way of filling in metavariables (e.g., for any expressions e_1, e_2)

Big-Step Inference Rules

$$\frac{}{e_1 \mathbf{+} e_2 \Downarrow}$$

1. What question are we trying to answer?
   - What kind of program does this rule evaluate?
2. What do we need to know?
   - What subexpressions need to be evaluated? What else do we need to compute?
3. What is the answer?
   - What value should the program return?

Questions

Nobody has responded yet.

Hang tight! Responses are coming in.

Using the Big-Step Semantics

$$\frac{e \Downarrow b \quad (\text{if } b \text{ then } e_1 \text{ else } e_2) \Downarrow v}{\texttt{if}\ e\ \texttt{then}\ e_1\ \texttt{else}\ e_2 \Downarrow v}$$

$$\frac{}{\texttt{if 1+2=3 then 2*2 else 7} \Downarrow 4}$$

Using the Big-Step Semantics

$$\frac{e \Downarrow b \quad (\text{if } b \text{ then } e_1 \text{ else } e_2) \Downarrow v}{\texttt{if}\ e\ \texttt{then}\ e_1\ \texttt{else}\ e_2 \Downarrow v}$$

$$\frac{\texttt{1+2=3} \Downarrow \text{true} \qquad \texttt{2*2} \Downarrow 4}{\texttt{if 1+2=3 then 2*2 else 7} \Downarrow 4}$$

Using the Big-Step Semantics

$$\frac{e_1 \Downarrow v_1 \quad e_2 \Downarrow v_2 \quad (b \text{ is true iff } v_1 = v_2)}{e_1 \texttt{ = } e_2 \Downarrow b}$$

$$\frac{\dfrac{\texttt{1+2} \Downarrow 3 \quad \texttt{3} \Downarrow 3}{\texttt{1+2=3} \Downarrow \text{true}} \quad \texttt{2*2} \Downarrow 4}{\texttt{if 1+2=3 then 2*2 else 7} \Downarrow 4}$$

Using the Big-Step Semantics

$$\frac{e_1 \Downarrow i_1 \quad e_2 \Downarrow i_2 \quad (i = i_1 + i_2)}{e_1 \texttt{+} e_2 \Downarrow i}$$

$$\dfrac{\dfrac{\dfrac{\texttt{1} \Downarrow 1 \quad \texttt{2} \Downarrow 2}{\texttt{1+2} \Downarrow 3} \quad \texttt{3} \Downarrow 3}{\texttt{1+2=3} \Downarrow \text{true}} \quad \texttt{2*2} \Downarrow 4}{\texttt{if 1+2=3 then 2*2 else 7} \Downarrow 4}$$

Using the Big-Step Semantics

$$\frac{(i \text{ is a number literal})}{i \Downarrow i}$$

$$\cfrac{\cfrac{\cfrac{\overline{\texttt{1} \Downarrow 1} \quad \overline{\texttt{2} \Downarrow 2}}{\texttt{1+2} \Downarrow 3} \quad \texttt{3} \Downarrow 3}{\texttt{1+2=3} \Downarrow \text{true}} \quad \texttt{2*2} \Downarrow 4}{\texttt{if 1+2=3 then 2*2 else 7} \Downarrow 4}$$

Using the Big-Step Semantics

$$\frac{\dfrac{\dfrac{\dfrac{}{1 \Downarrow 1} \qquad \dfrac{}{2 \Downarrow 2}}{1+2 \Downarrow 3} \qquad \dfrac{}{3 \Downarrow 3}}{1+2=3 \Downarrow \text{true}} \qquad \dfrac{\dfrac{}{2 \Downarrow 2} \qquad \dfrac{}{2 \Downarrow 2}}{2*2 \Downarrow 4}}{\texttt{if 1+2=3 then 2*2 else 7} \Downarrow 4}$$

Questions

Nobody has responded yet.

Hang tight! Responses are coming in.

Expressions: Big-Step Semantics

$$\frac{(i \text{ is a number literal})}{i \Downarrow i}$$

$$\frac{e_1 \Downarrow i_1 \quad e_2 \Downarrow i_2 \quad (i = i_1 + i_2)}{e_1 \textbf{ + } e_2 \Downarrow i}$$

$$\frac{(b \text{ is a boolean literal})}{b \Downarrow b}$$

$$\frac{e_1 \Downarrow b_1 \quad e_2 \Downarrow b_2 \quad (b = b_1 \mathbin{\&\&} b_2)}{e_1 \textbf{ and } e_2 \Downarrow b}$$

$$\frac{e_1 \Downarrow i_1 \quad e_2 \Downarrow i_2 \quad (b \text{ is true iff } i_1 \text{ is the same as } i_2)}{e_1 \textbf{ = } e_2 \Downarrow b}$$

Expressions: Big-Step Semantics

$$\frac{(i \text{ is a number literal})}{i \Downarrow i} \qquad \frac{e_1 \Downarrow i_1 \quad e_2 \Downarrow i_2}{e_1 \mathbf{+} e_2 \Downarrow (i_1 + i_2)}$$

```
match (e : exp) with
| Num i -> IntVal i
| Add (e1, e2) -> (match eval e1, eval e2 with
                   | Some (IntVal i1), Some (IntVal i2) ->
                       Some (IntVal (i1 + i2))
                   | ...)
```

Exercise: Case Expression

- Like a 3-way if, or a switch statement in C

- **case** 5 **pos:** 1 **neg:** 2 **zero:** 3 should return 1
- **case** -1 **pos:** 1 **neg:** true **zero:** 3 should return true
- **case** 3–3 **pos:** 1 **neg:** 2 **zero:** 2+3 should return 5

- Step 1: Describe its behavior in English.

Exercise: Case Expression

- Step 1: Describe its behavior in English.

Evaluates e_1 if e is positive, e_2 if e is negative, e_3 if e is zero

- Step 2: Write big-step semantic rule(s) for it. (in-class exercise)

$$\frac{?}{\textbf{case}\ e\ \textbf{pos:}\ e_1\ \textbf{neg:}\ e_2\ \textbf{zero:}\ e_3 \Downarrow ?}$$

Exercise: Case Expression

- Step 2: Write big-step semantic rule(s) for it.

$$\frac{?}{\texttt{case}\ e\ \texttt{pos:}\ e_1\ \texttt{neg:}\ e_2\ \texttt{zero:}\ e_3 \Downarrow ?}$$

$$\frac{e \Downarrow \text{true} \quad e_1 \Downarrow v}{\texttt{if}\ e\ \texttt{then}\ e_1\ \texttt{else}\ e_2 \Downarrow v} \qquad \frac{e \Downarrow \text{false} \quad e_2 \Downarrow v}{\texttt{if}\ e\ \texttt{then}\ e_1\ \texttt{else}\ e_2 \Downarrow v}$$

Exercise: Case Expression

- Step 2: Write big-step semantic rule(s) for it.

$$\frac{e \Downarrow i \quad (i > 0) \quad e_1 \Downarrow v}{\texttt{case}\ e\ \texttt{pos:}\ e_1\ \texttt{neg:}\ e_2\ \texttt{zero:}\ e_3 \Downarrow v}$$

$$\frac{e \Downarrow i \quad (i < 0) \quad e_2 \Downarrow v}{\texttt{case}\ e\ \texttt{pos:}\ e_1\ \texttt{neg:}\ e_2\ \texttt{zero:}\ e_3 \Downarrow v}$$

$$\frac{e \Downarrow \mathrm{true} \quad e_1 \Downarrow v}{\texttt{if}\ e\ \texttt{then}\ e_1\ \texttt{else}\ e_2 \Downarrow v} \qquad \frac{e \Downarrow \mathrm{false} \quad e_2 \Downarrow v}{\texttt{if}\ e\ \texttt{then}\ e_1\ \texttt{else}\ e_2 \Downarrow v}$$

Exercise: Case Expression

- Step 2: Write big-step semantic rule(s) for it.

$$\frac{e \Downarrow i \quad (i > 0) \quad e_1 \Downarrow v}{\textbf{case}\ e\ \textbf{pos}:\ e_1\ \textbf{neg}:\ e_2\ \textbf{zero}:\ e_3 \Downarrow v}$$

$$\frac{e \Downarrow i \quad (i < 0) \quad e_2 \Downarrow v}{\textbf{case}\ e\ \textbf{pos}:\ e_1\ \textbf{neg}:\ e_2\ \textbf{zero}:\ e_3 \Downarrow v}$$

$$\frac{e \Downarrow 0 \quad e_3 \Downarrow v}{\textbf{case}\ e\ \textbf{pos}:\ e_1\ \textbf{neg}:\ e_2\ \textbf{zero}:\ e_3 \Downarrow v}$$

Exercise: Case Expression

- Step 3: Implement it in the interpreter.

$$\frac{e \Downarrow i \quad (i > 0) \quad e_1 \Downarrow v}{\textbf{case}\ e\ \textbf{pos:}\ e_1\ \textbf{neg:}\ e_2\ \textbf{zero:}\ e_3 \Downarrow v} \qquad \ldots$$

```
let rec eval (e : exp) : value option =
  match e with
  | Case (cond, e1, e2, e3) -> (match eval cond with …)
```

Exercise: Case Expression

- Step 3: Implement it in the interpreter.

```
let rec eval (e : exp) : value option =
  match e with
  | Case (cond, e1, e2, e3) ->
    (match eval cond with
     | Some (IntVal i) -> if i > 0 then eval e1
                          else if i < 0 then eval e2 else eval e3
     | _ -> None)
```

Exercise: Case Expression

- Step 4: Run test cases.

```
let rec eval (e : exp) : value option = ...

eval (Case (Num 5, Num 1, Num 2, Num 3));;
(* should return Some (IntVal 1)) *)
```

Questions

Nobody has responded yet.

Hang tight! Responses are coming in.