Lecture 16
State – Effects

Languages considered so far

- LET
- PROC
- LETREC
- EXPLICIT-REFS (EREF)
Computational Effects

- So far we have considered
  - Expressions generating values
  - Everything local
  - No notion of global state
  - No global storage
- We want to be able to
  - Read memory locations
  - Print values in the memory
  - Write to the memory
  - Have global variables
  - Share values across separate computations
- We need
  - A model for memory
    - Access memory locations
    - Modify memory contents

New concepts

- Storable values
  - What sorts of things can we store?
- Memory stores
  - Where do we store things?
- Memory references (pointers)
  - How do we access the stores?
The new design

- Denotable and Expressed values

\[
\begin{align*}
ExpVal & = Int + Bool + Proc + Ref(ExpVal) \\
DenVal & = ExpVal
\end{align*}
\]

- Three new operations
  - `newref`
  - `deref`
  - `setref`

Example: references help us share variables

```plaintext
let x = newref(0)
in letrec even(dummy) = if zero?(deref(x))
  then 1
  else begin
    setref(x, +(deref(x),1));
    (odd 888)
  end
odd(dummy) = if zero?(deref(x))
  then 0
  else begin
    setref(x, -(deref(x),1));
    (even 888)
  end
in begin setref(x,13); (odd 888) end
```
Example: references help us create hidden state

\[
\text{let } g = \text{let } \text{counter } = \text{newref}(0) \\
\quad \text{in proc (dummy)} \\
\quad \begin{align*}
\text{begin} \\
& \text{setref(counter, -(deref(counter), -1))}; \\
& \text{deref(counter)} \\
\text{end} \\
\text{in let } a = (g \ 1) \\
\text{in let } b = (g \ 1) \\
\text{in } -(a, b)
\end{align*}
\]

The entire expression evaluates to -1

Behind the scenes...
Example: reference to a reference

```
let x = newref(newref(0))
in begin
  setref(deref(x), 11);
  deref(deref(x))
end
```

What does this evaluate to?