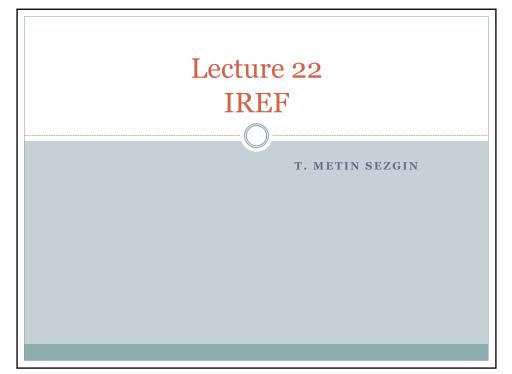
Announcements

Project deadline extended

1



Implicit references

- EREF
 - References instantiated explicitly
 - References explicitly stored in the store
 - Expressed and denoted values include references

ExpVal = Int + Bool + Proc + Ref(ExpVal)DenVal = ExpVal

3

Implicit references

- IREF
 - References are instantiated by the interpreter
 - o All denoted values are references to expressed values
 - o Each binding operation introduces a location
 - × Let
 - × letrec
 - × proc
 - o Pointers to stores are saved in the environment

ExpVal = Int + Bool + ProcDenVal = Ref(ExpVal)

New grammar

• A set operation for assignment

```
Expression ::= \mathtt{set} \ \textit{Identifier} = \textit{Expression} \\ \boxed{\mathtt{assign-exp} \ (\mathtt{var} \ \mathtt{exp1})}
```

5

```
Examples
let x = 0
in letrec even(dummy)
            = if zero?(x)
               then 1
                                               let g = let count = 0
           in proc (dummy)
               else begin
                     set x = -(x, 1);
                                                            begin
                     (odd 888)
                                                              set count = -(count,-1);
                    end
                                                              count
           odd (dummy)
                                                            end
            = if zero?(x)
                                               in let a = (g 11)
in let b = (g 11)
               then 0
               else begin
                                                     in -(a,b)
                     set x = -(x,1);
                     (even 888)
   end in begin set x = 13; (odd -888) end
```

Behavior specification

• var-exp

(value-of (var-exp var) ρ σ) = ($\sigma(\rho(var)), \sigma$)

assign-exp

```
\frac{(\text{value-of } exp_1 \ \rho \ \sigma_0) = (val_1, \sigma_1)}{(\text{value-of } (\text{assign-exp } var \ exp_1) \ \rho \ \sigma_0) = (\lceil 27 \rceil, [\rho(var) = val_1]\sigma_1)}
```

• apply-procedure

```
(apply-procedure (procedure var\ body\ \rho) val\ \sigma) = (value-of body\ [var=l]\rho\ [l=val]\sigma)
```

7

Implementation

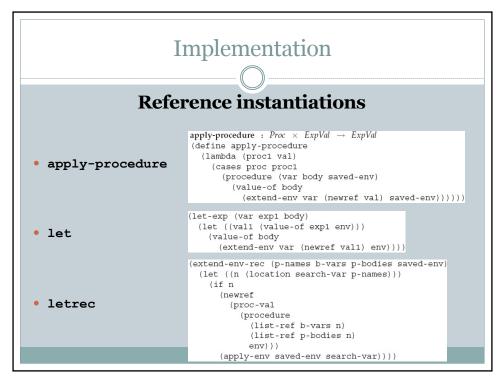
• var-exp

(var-exp (var) (deref (apply-env env var)))

• assign-exp

```
(assign-exp (var exp1)
  (begin
      (setref!
            (apply-env env var)
            (value-of exp1 env))
      (num-val 27)))
```

apply-procedure



С