13. PROC

lecture 13 -- Proc.pdf

We can now define procedures!

```
Expression ::= proc (Identifier) Expression
                          proc-exp (var body)

Expression ::= (Expression Expression)
                          call-exp (rator rand)
```

rator: operator
rand: Actual parameter

Example:

```
let f = proc (x) - (x,11)
        in (f (f 77))

(proc (f) (f (f 77))
        proc (x) -(x,11))
```

Expressed and Denoted values

Before:

```
ExpVal = Int + Bool
DenVal = Int + Bool
```

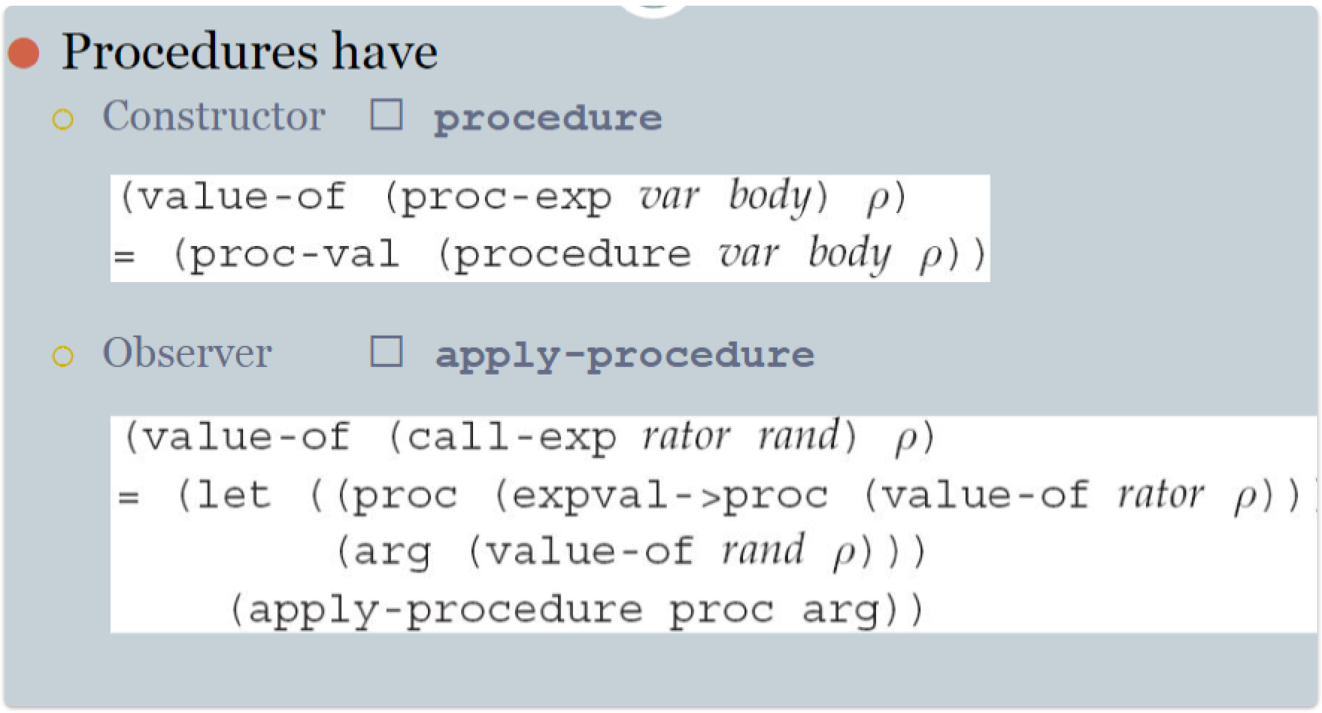
After:

```
ExpVal = Int + Bool + Proc
DenVal = Int + Bool + Proc
```

What?

Constructors And Observers

Constructor: Similar to builders in OOP languages.
Observer: Retrieves values without modifying the object, similar to getter methods in OOP.



- Procedures have
  - Constructor □ **procedure**

```
(value-of (proc-exp var body) ρ)
= (proc-val (procedure var body ρ))
```

  - Observer □ **apply-procedure**

```
(value-of (call-exp rator rand) ρ)
= (let ((proc (expval->proc (value-of rator ρ))
        (arg (value-of rand ρ)))
     (apply-procedure proc arg))
```

We construct with *procedure* and observe with *apply-procedure*.