

14. PROC 2

[lecture 14 -- Proc.pdf](#)

Implementation

```
proc? : SchemeVal → Bool
(define proc?
  (lambda (val)
    (procedure? val)))

procedure : Var × Exp × Env → Proc
(define procedure
  (lambda (var body env)
    (lambda (val)
      (value-of body (extend-env var val env))))))

apply-procedure : Proc × ExpVal → ExpVal
(define apply-procedure
  (lambda (proc1 val)
    (proc1 val)))
```

Alternative Implementation

```
proc? : SchemeVal → Bool
procedure : Var × Exp × Env → Proc
(define-datatype proc proc?
  (procedure
    (var identifier?)
    (body expression?)
    (saved-env environment?)))

apply-procedure : Proc × ExpVal → ExpVal
(define apply-procedure
  (lambda (proc1 val)
    (cases proc proc1
      (procedure (var body saved-env)
        (value-of body (extend-env var val saved-env)))))))
```

Changes To The Interpreter

```
(define-datatype expval expval?
  (num-val
    (num number?))
  (bool-val
    (bool boolean?))
  (proc-val
    (proc proc?)))

(proc-exp (var body)
  (proc-val (procedure var body env)))

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