15. LETREC

Lecture 15 -- LETREC.pdf

PROC is dead! Long live LETREC!

LETREC = PROC + RECURSION (via `LETREC`)

What we want to implement:

```
letrec double(x)
        = if zero?(x) then 0
                else -((double - (x,1)), -2)
in (double 6)
```

Steps

1. Define the grammar

```
Expression ::= letrec Identifier (Identifier) = Expression in Expression
                           letrec-exp (p-name b-var p-body letrec-body)
```

1st line is defined through the `the-lexical-spec` part of the lang.rkt, the ssl gen library parses the string and returns the proper data.
2nd line is defined in the `the-grammar` part of the lang.rkt.

2. The environment must be updated to be extended recursively.

Extend the environment recursivly

```
(value-of
  (letrec-exp proc-name bound-var proc-body letrec-body)
  ρ) → initial env
= (value-of
    letrec-body
    (extend-env-rec proc-name bound-var proc-body ρ)) → extended env
```

extend the environment for recursive calls
recursivly

if the search variable matches a recursive Procedure

```
(apply-env ρ1 proc-name) → search for proc in environment
= (proc-val (procedure bound-var proc-body ρ1))
```

return the proc-val found in the environment

if there is no match

```
(apply-env ρ1 var) = (apply-env ρ var)
```

search for var in env in procedure