3.2 The Front End

- Want to avoid having to specify structure explicitly:

```
> (a-program          ; add1(+(3,x))
    (primapp-exp
      (incr-prim)
        ((primap-exp
          (add-prim)
            ((lit-exp 3) (var-exp
x))))))
```

- Instead, want to enter programs directly into interpreter:

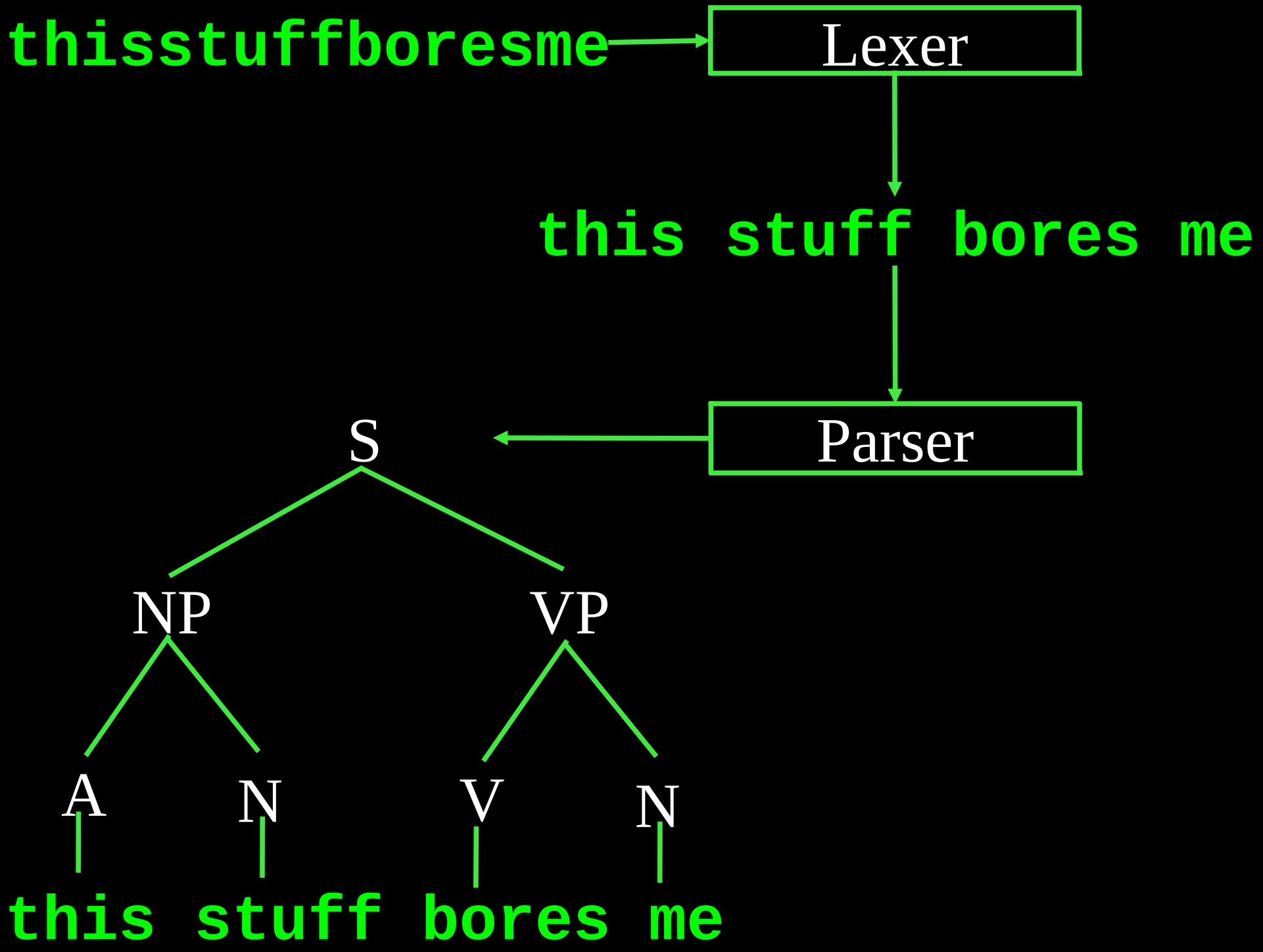
```
--> add1(+(3,x))
```

3.2 The Front End

This will require two steps:

- Break input string into words (*tokens*, *lexemes*): *scanning* (*lexing*)
- Group tokens into constituents (trees and sub-trees):

parsing


thisstuffboresme
Lexer
this stuff bores me
Parser
S
NP
VP
A
N
V
N
this stuff bores me

- Want to provide grammar and have parser built for us automatically:

```
S    ::=  NP  VP

NP   ::=  A   N

VP   ::=  V   N

A    ::=  this

N    ::=  me | stuff

V    ::=  bores
```

- A program that does this is called a *parser generator*.
- For C, use lex/yacc.
- For Scheme, use SLLGEN:

Scanner specification:

```
(define scanner-spec-3-1
  '((white-sp
       (whitespace)                          skip)
    (comment
       ("%" (arbno (not #\newline)))         skip)
    (identifier
      (letter (arbno (or letter
                         digit
                         "_"
                         "-"
                         "?")))              symbol)
    (number (digit (arbno digit))            number)))
```

Parser specification:

```
(define grammar-3-1
  '((program (expression) a-program)
    (expression (number) lit-exp)
    (expression (identifier) var-exp)
    (expression
      (primitive "("
          (separated-list expression ",") ")")
      primapp-exp)
    (primitive ("+")     add-prim)
    (primitive ("-")     subtract-prim)
    (primitive ("*")     mult-prim)
    (primitive ("add1")  incr-prim)
    (primitive ("sub1")  decr-prim)))
```

Usage:

```
(define read-eval-print
  (sllgen:make-rep-loop
        "--> "
        eval-program
        (sllgen:make-stream-parser
           scanner-spec-3-1
           grammar-3-1)))

  > (read-eval-print)
  --> 5
  5
  --> add1(2)
  3
  --> +(add1(2), -(6,4))
  5
```