Mini Language

Mark Boady

August 20, 2013
Mini language

- A simple programming language
- Implemented in PLY
- You will modify it for PA3
At the top level, we define a program

```
program -> stmt_list
stmt_list -> stmt SEMICOLON stmt_list | stmt
stmt ->
    assign_stmt | while_stmt | if_stmt | define_stmt
```
What kind of statements can be used to write a program

assign_stmt -> INDENT ASSIGNOP expr
while_stmt -> WHILE expr DO stmt_list OD
if_stmt -> IF expr THEN stmt_list ELSE stmt_list FI
define_stmt ->
    DEFINE IDENT PROC
    LPAREN param_list RPAREN stmt_list_list END
Expressions are commands that can actually be evaluated

\[
\begin{align*}
expr & \rightarrow expr \ Plus \ term \mid term \\
expr & \rightarrow expr \ Minus \ term \mid term \\
term & \rightarrow term \ Times \ fact \mid fact \\
fact & \rightarrow NUMBER \mid (expr) \mid IDENT \mid func\_call
\end{align*}
\]

Basic Elements are numbers or Variable Names

\[
\begin{align*}
Number & \rightarrow [0-9]+ \\
Ident & \rightarrow [a-z]+ \ except \ reserved \ words
\end{align*}
\]
Function calls and definitions need inputs

```plaintext
func_call -> IDENT LPAREN expr_list RPAREN
expr_list -> expr COMMA expr_list | expr
param_list -> IDENT COMMA param_list | IDENT
```
Parsing

- Instead of evaluating as we parse
- Step 1: Build a Parse Tree
- Step 2: Evaluate the Parse Tree

```python
def p_add( p ) :
    'expr : expr PLUS term'
    p[0] = Plus( p[1], p[3] )

def p_assn( p ) :
    'assign_stmt : IDENT ASSIGNOP expr'
    p[0] = AssignStmt( p[1], p[3] )
```
Parse Tree

- Parse Tree Objects
  - Constructor Creates a Tree
  - eval walks the tree and evaluates the expression
  - Display is for debugging

```python
class Plus( Expr ):
    def __init__( self, lhs, rhs ):
        self.lhs = lhs
        self.rhs = rhs

    def eval( self, nt, ft ):
        return self.lhs.eval( nt, ft ) + self.rhs.eval( nt, ft )

    def display( self, nt, ft, depth=0 ):
        ...
```
Evaluating the Parse Tree

def eval( self, nt, ft ) :
    return self.lhs.eval( nt, ft ) + self.rhs.eval( nt, ft )

- nt - The name table, stores the names of variables and values
- ft - function table, stores the defined functions
Parse Tree for $a := 4 \times 7$
An Example Program

- A program to find the factorial of $n$
  
  \[
  n := 0 - 5; \\
  \text{if } n \text{ then } i := n \text{ else } i := 0 - n \text{ fi; } \\
  \text{fact} := 1; \\
  \text{while } i \text{ do fact} := \text{fact} \times i; i := i - 1 \text{ od}
  \]

- For while and if, false means $\leq 0$ and true means $> 0$
Running a Program

$ python interpreterext.py < TestInput/fact.p
[42]
Running Program
Dump of Symbol Table
Name Table

   i -> 0
   fact -> 120
   n -> -5

Function Table

- The program has no print command
- We can see the final symbol table and look for answers
Mini Language Question of PA3

Add the comparison operators: < > <= >= == !=

Currently:
  - true means > 0
  - false means <= 0

You can make the operators return 0 or 1

- (4 < 5) returns 1 for true
- (5 == 7) return 0 for false