CSE401: Parsing (C)

David Notkin
Autumn 2000

UW CSE401 AQ 2000 • D. Notkin • All rights reserved • Parsing B • Slide 2

Objectives: today

- Recap and clarify PREDICT table
- Describe computation of FIRST and FOLLOW
  - And the relationship to PREDICT
- Recursive descent parsing
  - High-level issues and
  - (time-permitting) a walk through the PL/0 parser

UW CSE401 AQ 2000 • D. Notkin • All rights reserved • Parsing B • Slide 3

Next week

- Monday: no class
- Wednesday: Mark Seigle
- Friday: Matthai Philipose

UW CSE401 AQ 2000 • D. Notkin • All rights reserved • Parsing B • Slide 4

Example

```
Stmt  ::= 1 if expr then Stmt else Stmt |
          2 while Expr do Stmt |
          3 begin Stmts end
Stmts ::= 4 Stmt ; Stmts | 5 ε
Expr  ::= 6 id
```

	if	then	else	while	do	begin	end	id	;
Stmt	1			2		3			
Stmts	4			4		4	5		
Expr								6	

UW CSE401 AQ 2000 • D. Notkin • All rights reserved • Parsing B • Slide 5

```
if id then
  begin end
else
  while id do
    begin end
  end
```

```
➢   Stmt
1.  if Expr then Stmt else Stmt end
6.  if id then Stmt else Stmt end
3.  if id then begin Stmts end
          else Stmt end
5.  if id then begin end
          else Stmt end
2.  if id then begin end
          else while Expr do Stmt end
6.  if id then begin end
          else while id do Stmt end
3.  if id then begin end
          else while id do
                 begin Stmts end end
5.  if id then begin end
          else while id do
                 begin end end
```

Production applied

UW CSE401 AQ 2000 • D. Notkin • All rights reserved • Parsing B • Slide 6

Compute PREDICT

In terms of FIRST and FOLLOW

- FIRST(α)
  - ✓ α is any string of tokens
  - ✓ FIRST(α) is the set of terminals that being strings derived from α
  - ✓ If α can derive ε, then include ε in FIRST
- FOLLOW(A)
  - ✓ A is a non-terminal
  - ✓ FOLLOW(A) is the set of terminals that can appear immediately after A in some derivation in the language
  - ✓ If A can be the end of such a derivation, then include $ in FOLLOW
    - ✓ $ represents "end of input"

UW CSE401 AQ 2000 • D. Notkin • All rights reserved • Parsing B • Slide 7

Computing FIRST(X)

for all grammar symbols X

- If X is a terminal, then FIRST(X) is {X}
- If X ::= ε, then add ε to FIRST(X)
- If X is a non-terminal and X ::= $Y_1Y_2…Y_k$
  - FIRST(X) is initialized to FIRST(Y_1)
  - If ε ∈ FIRST(Y_1) then union in FIRST(Y_2)
  - If ε ∈ FIRST(Y_1) and ε ∈ FIRST(Y_2) then union in FIRST(Y_3)
  - Etc.
  - If ε ∈ FIRST(Y_i) for all i, then add ε to FIRST(X)

UW CSE401 AQ 2000 • D. Notkin • All rights reserved • Parsing B • Slide 8

Computing FIRST($X_1X_2…X_n$)

- Start with all the non-ε symbols in FIRST(X_1)
- If ε ∈ FIRST(X_1), then union in all the non -ε symbols in FIRST(X_2)
- If ε ∈ FIRST(X_1) and ε ∈ FIRST(X_2), then union in all the non -ε symbols in FIRST(X_3)
- Etc.
- If ε ∈ FIRST(X_i) for all i, then add ε to FIRST($X_1X_2…X_n$)

UW CSE401 AQ 2000 • D. Notkin • All rights reserved • Parsing B • Slide 9

Computing FOLLOW(A)

for all non-terminals A

- Place $ in FOLLOW(S), where S is the start symbol and $ is the end-of-input marker
- If there is a production A ::= αBβ then everything in FIRST(β) except for ε is placed in FOLLOW(B)
- If there is a production A ::= αB (or A ::= αBβ where ε ∈ FIRST(β)) then everything in FOLLOW(A) is placed in FOLLOW(B)
- Apply until a fixpoint is reached

UW CSE401 AQ 2000 • D. Notkin • All rights reserved • Parsing B • Slide 10

	FIRST	FOLLOW
S ::= **if** E **then** S **else** S		
| **while** E **do** S		
| **begin** Ss **end**		
Ss ::= S **;** Ss		
| ε		
E ::= **id**		

UW CSE401 AQ 2000 • D. Notkin • All rights reserved • Parsing B • Slide 11

Another example

```
E  ::= T { + T }
T  ::= F { * F }
F  ::= - F | id | ( E )
```

Sugared

```
E  ::= 1 T E'
E' ::= 2 + T E' | 3 ε
T  ::= 4 F T'
T' ::= 5 * F T' | 6 ε
F  ::= 7 - F | 8 id | 9 ( E )
```

Unsugared

UW CSE401 AQ 2000 • D. Notkin • All rights reserved • Parsing B • Slide 12

Construct predictive parsing table

- Input is a grammar G
- Output is a table M indexed by non-terminals (rows) and terminals (columns)

```
foreach production A ::= α in G do
   foreach terminal a in FIRST(α)
      add the production to M[A,a];
   if ε ∈ FIRST(α)
      add the production to M[A,b] for
        each terminal b in FOLLOW(A)
   if ε ∈ FIRST(α) and $ ∈ FOLLOW(A)
      add the production to M[A,$]
Every remaining undefined entry is an error
```

UW CSE401 AQ 2000 • D. Notkin • All rights reserved • Parsing B • Slide 13

PREDICT

	id	+	-	*	/	(	)
E							
E′							
T							
T′							
F							

UW CSE401 AQ 2000 • D. Notkin • All rights reserved • Parsing B • Slide 14

PREDICT and LL(1)

- If the PREDICT table has at most one entry in each cell, then the grammar is LL(1)
  - And there is only one choice (it's predictive) , making it fast to parse and easy to implement
- Multiple entries in a cell
  - Arise with left recursion, ambiguity, common prefixes, etc.
  - Can patch by hand
  - Or use more powerful parsing techniques

UW CSE401 AQ 2000 • D. Notkin • All rights reserved • Parsing B • Slide 15

Recursive descent parsers

- Write procedure for each non-terminal
- Each procedure selects the correct right-hand side by peeking at the input tokens
- Then the r.h.s. is consumed
  - If it's a terminal symbol, verify it is next and then advance through the token stream
  - If it's a non-terminal, call corresponding procedure
- Construct and return AST representing the r.h.s.

UW CSE401 AQ 2000 • D. Notkin • All rights reserved • Parsing B • Slide 16

It's demo time…

- Let's look at some of the PL/0 code to see how the recursive descent parsing works in practice