CS480

Bottom Up Parsing Ch 4 p 195-215 + handouts! Read chap 4 by Monday.

March 1, 2013

CS 480 – Spring 2013 Pacific University

The Plan

I hate wordy slides. This topic is too precise for me to get correct without lots of hints.

- Bottom Up Parsing
- Handles
- Shift/Reduce
- Operator Precedence Parsing
- Building Operator Precedence Tables
- Wednesday: Build an OPT in class

Bottom Up Parsing

- Build parse tree from leaves and work up!
 Reduce a string, w, to the start symbol, S
- Reduction: replace a substring that matches the RHS of a production with the LHS of that production
 S -> aAB
 - *Right most derivation* is run in
- S -> aABe A -> Abc | b B -> d

reverse.

abbcde

Algorithm

1) Look for a substring in *w* that matches the right side of any production.

2) Repeat step 1) with the new string *w*' until the start symbol S is produced (accept) or we run out of matching possibilities (reject)

abbcde – Problems? S -> aABe A -> Abc | b B -> d

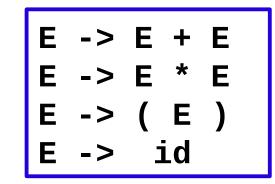
Handle

• A *handle* is a substring of a string that matches some production's right side such that a reduction to a nonterminal on the left can be done in one step along the *reverse of a rightmost derivation*.

abbcde

Practice

• page 196/198



id₃

id₂

id,

Right Most Derivation

rm

- Remember, we are $E \Rightarrow E + E$ => E + E * E => E + E * i doing bottom up parsing, so we start => E + id₂ * right here _ ►=> id, + id₂ *
- Ambiguous grammar so 1+ =>
- Handle Pruning

How to choose a Handle

• Add a restriction

We'll see an *implementable* algorithm for this later.

• Defn[Aho]: "A handle of a right-sentential form γ is a production A -> β and a position of γ where the string β may be found and replaced by A to produce the *previous right*sentential form in the rightmost derivation of y. That is, if S =>^{*}_{rm} αAw =>_{rm} $\alpha \beta w$, then A -> β in the position following α is a **handle** of $\alpha\beta w$." CS 480 - Spring 2013

Pacific University

• p199 ex4.24

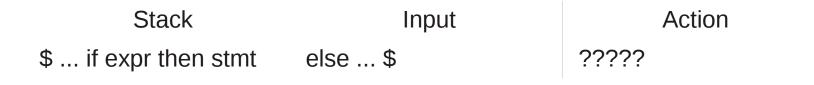


Stack	Input	Action
\$	id + id * id \$	shift
\$id	+ id * id \$	Reduce E -> id
\$E	+ id * id \$	shift
	CS 480 – Spring 2013	
	Pacific University	

CONFLICTS!

• p 201

- stmt -> if expr then stmt
 | if expr then stmt else stmt
 | other
- Some CFGs have unresolvable conflicts
 - shift/reduce
 - reduce/reduce



Operator Precedence Parsing

• Form of Shift/Reduce parsing

This allows us to find handles!

- Two important properties for these shiftreduce parsers is that ε does not appear on the right side of any production and no production has two adjacent nonterminals.
 - E -> E + E
 - T -> + T T

Precedence

• We need to define three different precedence relations between pairs of terminals

Relation	Meaning		
a <. b	a yields precedence to b		
a =. b	a has the same precedence as b		
a >. b	a takes precedence over b		

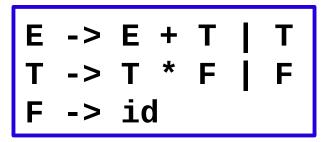
They look like >, <, and == but are very different

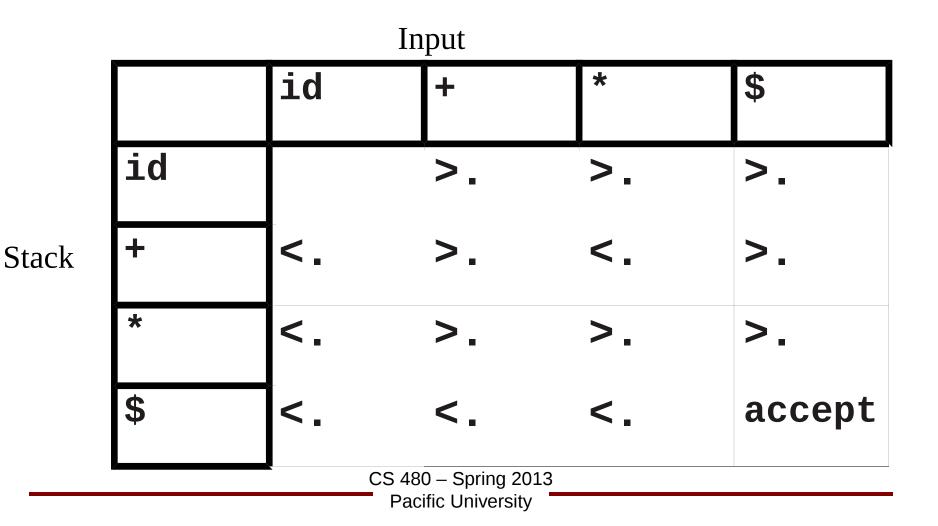
Why?

• Identify each handle using the precedence rules and reduce the right-sentential string, based on the precedence relations, to a start (accept) state.

Precedence Table

Define precedence relations between terminals.





How does this work? (high level) p205 \$ id + id * id \$

\$ <. id >. + <. id >. * <. id >. \$

- Scan from left (to right) until the first >. is found
- Then, scan backwards (left) until a <. is found
- The handle is everything to the left of the >. and right of the <.
 - Including surrounding nonterminals

In Code, p206 Algo 4.5

- How to find a handle
- Use a stack
- If the precedence relation <. or =. holds between the topmost terminal on the stack and the next input symbol, SHIFT
- If the relation >. holds, REDUCE
- No relation, SYNTAX ERROR

This is the solution to the bottom up assignment!

Example

Handle/Output	Stack	Input	Reason
	\$	id + id * id \$	Start State
	\$ id	+id*id\$	\$ <. id

Unary Operators

• In your grammar: *, &, -

Example: Unary prefix operator
 ~ (not operator. Is not also a binary operator)
 X <. ~ for any op X
 ~>. X if ~ has higher precedence than X,
 and ~ <. X otherwise

Unary op is also a binary op

- * is dereference and multiplication
- - is negation and subtraction
- & is not also binary
- Use lexer to return different token for
 - Dereference/Multiplication
 - Negation/Subtraction
- How?
 - Lexer needs to remember the previous token!
 - Cannot look ahead

Define

• Operator-precedence grammar is an ε-free operator grammar in which the precedence

relations <.,=.,>. constructed as previous are disjoint. For any pair of terminals, a and b, never more than one of the relations a <.b, a=.b, a >. b is true.

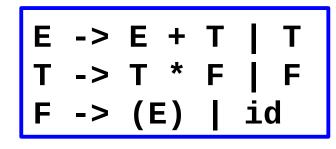
Create Table

- Let G be an ε-free operator grammar
- For each two terminals a and b we say:
- a =. b if there exists a RHS: $\alpha a\beta b\gamma$ where β is either ϵ or a single nonterminal.
- a <. b if for some NT A, a RHS αaAβ exists, and A=>⁺ γbδ, where γ is either ε or a single NT
- a >. b if for some NT A, a RHS $\alpha Ab\beta$ exists, and A=>⁺ $\gamma a\delta$, where δ is either ϵ or a single NT

LEADING, TRAILING

- LEADING: for each NT, those terminals that can be the first terminal in a string derived from that NT
- TRAILING: for each NT, those terminals that can be the last terminal in a string derived from that NT

Leading/Trailing



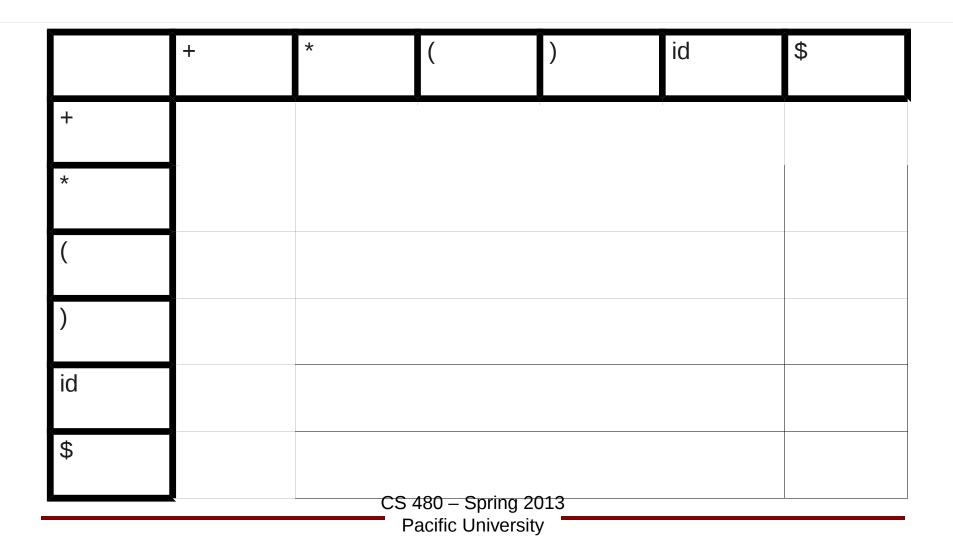
Nonterminal	First terminal	Last terminal
E		
Т		
F		

Compute Precedence

- For =. look for RHS with two terminals separated by nothing or a NT
- E -> E + T | T T -> T * F | F F -> (E) | id
- <. Look for RHS with a terminal immediately to the left of a NT (a, A in rule above) For each, a is
 <. to any terminal LEADING(A)
- >. Look for a RHS with a nonterminal immediately to the left of a terminal (A, b rule above). Every terminal TRAILING(A) >. b

Compute Precedence

• Algo 5.2 on handout!





CS 480 – Spring 2013 Pacific University

Create Operator Precedence Table

- Page 207: heuristic for arithmetic expressions
- Precedence & Associativity
- If op X has higher precedence than op Y, make X >. Y and Y <. X
- If op X and op Y have equal precedence, make X >. Y and Y >. X if they are left assoc.
 X <. Y and Y <. X if they are right assoc.
- X <. id, id >. X, X <. (, (<. X,) >.X, X >.), X
 >. \$, \$<.X, for all op X
- More on page 207

Build the table!

	Operators	Associativity		
High	Λ	right		
	* /	left		
Low	+ -	left		

	+ -	*	/	^	id	()	\$
+								
-								
*								
/								
٨								
id								
(
)								
\$								
CS 480 – Spring 2013 Pacific University								
Pacific University								