Syntax Analysis - LR(0) Parsing

66.648 Compiler Design Lecture (02/04/98)

Computer Science

Rensselaer Polytechnic

Lecture Outline

- LR(0) Parsing Algorithm
- Parse Tables
- Examples
- Administration

LR(k) Parsing Algorithms

This is an efficient class of Bottom-up parsing algorithms. The other bottom-up parsers include operator precedence parsers.

The name LR(k) means:

L - Left-to-right scanning of the input

R - Constructing rightmost derivation in reverse

k - number of input symbols to select a parser action

Yet Another Example

Consider a grammar to generate all palindromes.

1) S--> P

2) P --> a Pa

3) P --> b P b

4) P --> c

LR parsers work with an augmented grammar in which the start symbol never appears in the right side of a production. In a given grammar, if the start symbol appears in the RHS, we can add a production S' --> S (S' is the new start symbol and S was the old start symbol)

Example Cont...

STACK	INPUT BUFFER	ACTION
$	abcba$	shift
$a	bcba$	shift
$ab	cba$	shift
$abc	ba$	reduce
$abP	ba$	shift
$abPb	a$	reduce
$aP	a$	shift
$aPa	$	reduce
$P	$	reduce

LR(0) Parsers

Qn: How to select parser actions (namely shift, reduce, accept and error)?

Ans:

1) By constructing a DFA that encodes all parser states, and transitions on terminals and nonterminals. The transitions on terminals are the parser actions(also called the action table) and transitions on nonterminals resulting in a new state (also called the goto table).

2) Keeping a stack to simulate the PDA. This stack maintains the list of states.

LR(0) Items and Closure

LR(0) parser state needs to capture how much of a given production we have scanned . LR(0) parser (like a FSA) needs to know how much the production (on the rhs) we have scanned so far.

For example: in the production:

P --> a P a

An LR(0) item is a production with a mark/dot on the RHS. SO the items for this production will be P--> . a P a , P --> a . P a, P --> a P. a, P--> aPa.

Items and Closure Contd

Intuitively, there is a derivation (or we have seen the input symbols) to the left of dot.

Two kinds of items, kernel items and nonkernel items - Kernel and nonkernel items.

Kernel Items - Includes initial item S' --> .S and all items in which dot does not appear at the left most position.

Nonkernel Items- All other items which have dots at the leftmost position.

Closure of Items

Let I be the set of items. Then Closure (I) consists of the set of items that are constructed as follows:

1) Every item I is also in the Closure(I) - reflexive

2 If A --. alpha . B beta is in Closure(I), and B--> gamma is production, then add the item B--> .gamma also in the Closure(I), if it is not already a member. Repeat this until no more items can be added.

Intuition

Closure represents an equivalent state - all the possible ways that you could have reached that state.

Example: I = { S--> .P}

Closure (I) = { S-->.P,P-->.aPa,P-->.bPb,P-->.c}

In Arithmetic Expression: S'-->.E

closure(I)={ }

GOTO Operation

Let I be the set of items and let X be a grammar symbol (nonterminal/terminal). Then

GOTO(I,X) = Closure({A--> alpha X.beta| A--> alpha . X beta is in I})

It is a new set of items moving a dot over X. Intuitively, we have seen either an input symbol (terminal symbol) or seen a derivation starting with that nonterminal.

Canonical set of Items (states)

Enumerate possible states for an LR(0) parser. Each state is a canonical set of items.

Algorithm:

1) Start with a canonical set, Closure({S'-->.S})

2) If I is a canonical set and X is a grammar symbol such that I'=goto(I,X) is nonempty, then make I' a new canonical set (if it is not already a canonical set). Keep repeating this until no more canonical sets can be created.

The algorithm terminates!!.

Example

S0: S--> .P , P --> .a P a, P--> .bP b, P-->.c

S1: S--> P.

S2: P --> a.Pa, P-->.aPa,P-->.bPb,P-->.c

S3:P--> b.P b, P-->.aPa,P-->.bPb,P-->.c

S4: P--> c.

S5: P--> aP.a

S6:P--> bP.b

S7: P--> aPa.

S8: P--> bP b.

Finite State Machine

Draw the FSA. The major difference is that transitions can be both terminal and nonterminal symbols.

Key Idea in Canonical states

If a state contains an item of the form A--> beta ., then state prompts a reduce action (provided the correct symbols follow).

If a state contains A--> alpha . delta, then the state prompts the parser to perform a shift action (of course on the right symbols).

If a state contains S'--> S. and there are no more input symbols left, then the parser is prompted to accept.

Else an error message is prompted.

Prasing Table

state	Input symbol				goto
	a	b	c	$	P
0	s2	s3	s4		2
1.				acc	
2.	s2	s3	s4		5
3.	s2	s3	s4		6
4.	r3	r3			
5.	s7				
6.		s8			
7.	r1	r1	r1	r1	

Parsing Table Contd

si means shift the input symbol and goto state I.

rj means reduce by jth production. Note that we are not storing all the items in the state in our table.

example: abcba$

if we go thru, parsing algorithm, we get

Example Contd

State	input	action
$S0	abcba$	shift
$S0aS2	bcba$	shift
$S0aS2bS3	cba$	shist
$S0aS2bS3cS4	ba$	reduce

Shift/Reduce Conflicts

An LR(0) state contains aconflict if its canonical set has two items that recommend conflicting actions.

shift/reduce conflict - when one item prompts a shift action, the other prompts a reduce action.

reduce/reduce conflict - when two items prompt for reduce actions by different production.

A grammar is said be to be LR(0) grammar, if the table does not have any conflicts.

LALR Grammar

Programming languages cannot be generated by

LR(0) grammar. We usually have a look ahead symbol, to deteremine what kind of action parser will be prompted for.

These lookaheads refine states and actions.

Comments and Feedback

Project 2 will be in the web by Friday (this).

Please keep reading chapter 4 and understand the material. Work out as many exercises as you can.