Parsing (Part 2)

Compiler Construction

2024-11-05

Prof. Dr. Michael Kuhn
michael.kuhn@ovgu.de

Parallel Computing and I/O
Institute for Intelligent Cooperating Systems
Faculty of Computer Science
Otto von Guericke University Magdeburg
https://parcio.ovgu.de

Outline

Group Exercise

- Create a context-free grammar for regular expressions.

Group Exercise

- Create a context-free grammar for regular expressions.
  1. $S \rightarrow P$
  2. $P \rightarrow P|P$
  3. $P \rightarrow PP$
  4. $P \rightarrow P^*$
  5. $P \rightarrow (P)$
  6. $P \rightarrow \Sigma$

Group Exercise

- Create a context-free grammar for zero or more pairs of <> and {} with x symbols.

Group Exercise

- Create a context-free grammar for zero or more pairs of <> and {} with x symbols.
  1. S → P
  2. P → PP
  3. P → P<P>P
  4. P → P{P}P
  5. P → x
  6. P → ϵ

Outline

LR Grammars

- LR(1) grammars are a superset of LL(1) grammars
- They allow left recursion and common left prefixes
  - Cannot be ambiguous
  - Cannot have shift-reduce or reduce-reduce conflicts

LR(0) Automaton

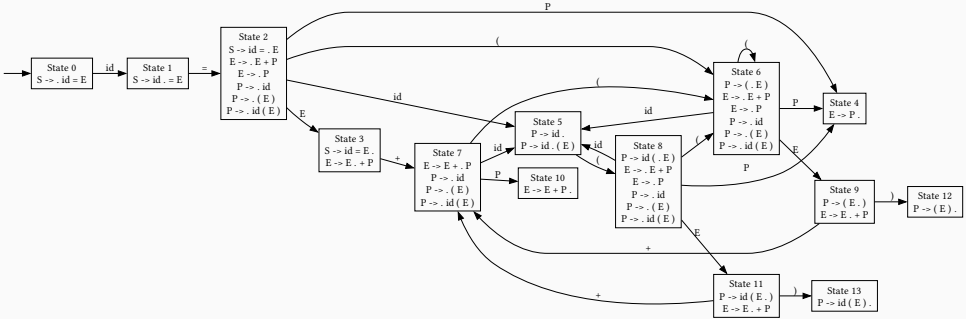
- LR(0) automaton represents all possible rules considered by a shift-reduce parser
- Each state consists of items (rules with a dot, indicating current position)
- State 0 is created using the start rule with a dot on the left

Group Exercise

- Draw the LR(0) automaton for grammar G_{13}.
  1. S $\rightarrow$ id = E
  2. E $\rightarrow$ E + P
  3. E $\rightarrow$ P
  4. P $\rightarrow$ id
  5. P $\rightarrow$ (E)
  6. P $\rightarrow$ id(E)

Group Exercise

- Draw the LR(0) automaton for grammar G_{13}.



- SLR grammars are a subset of LR(1) grammars
- Simple LR (SLR) parsing uses FOLLOW sets to resolve conflicts in LR(0) automaton
  - Reduction $A \rightarrow \alpha$ is only performed if the next token is in FOLLOW(A)

Group Exercise

- Determine the FIRST sets for grammar G_{13}.

Group Exercise

- Determine the FIRST sets for grammar G_{13}.
  - S = { id }
  - E = { id, (}
  - P = { id, (}

Group Exercise

- Determine the FOLLOW sets for grammar G_{13}.

- Determine the FOLLOW sets for grammar G_{13}.
  - S = { $ }
  - E = { $, +,) }
  - P = { $, +,) }

Group Exercise

- Write out the complete SLR parsing table for grammar G_{13}.

Group Exercise

- Write out the complete SLR parsing table for grammar G_{13}.

State	GOTO		ACTION					
	P	E	id	=	+	(	)	$
0			S1					
1				S2				
2	G4	G3	S5			S6		
3					S7			R1
4					R3		R3	R3
5					R4	S8	R4	R4
6	G4	G9	S5			S6		
7	G10		S5			S6		
8	G4	G11	S5			S6		
9					S7		S12	
10					R2		R2	R2
11					S7		S13	
12					R5		R5	R5
13					R6		R6	R6

- Is grammar G_{13} LL(1)? Why?

Group Exercise

- Is grammar G_{13} LL(1)? Why?
  - No, left recursion and common left prefixes

- Is grammar G_{13} LL(1)? Why?
  - No, left recursion and common left prefixes
- Is grammar G_{13} SLR? Why?

- Is grammar G_{13} LL(1)? Why?
  - No, left recursion and common left prefixes
- Is grammar G_{13} SLR? Why?
  - Yes, no conflicts

Group Exercise

- Perform SLR parsing using the previous parsing table for id = id (id + id)

Group Exercise

- Perform SLR parsing using the previous parsing table for id = id (id + id)

Stack	Symbols	Input	Action
0		id = id (id + id)	S1
0 1	id	= id (id + id)	S2
0 1 2	id =	id (id + id)	S5
0 1 2 5	id = id	(id + id)	S8
0 1 2 5 8	id = id (	id + id)	S5
0 1 2 5 8 5	id = id (id	+ id)	R4
0 1 2 5 8 4	id = id (P	+ id)	R3
0 1 2 5 8 11	id = id (E	+ id)	S7
0 1 2 5 8 11 7	id = id (E +	id)	S5
0 1 2 5 8 11 7 5	id = id (E + id	)	R4
0 1 2 5 8 11 7 10	id = id (E + P	)	R2
0 1 2 5 8 11	id = id (E	)	S13
0 1 2 5 8 11 13	id = id (E)		R6
0 1 2 4	id = P		R3
0 1 2 3	id = E		accept

LR(1) Automaton

- LR(1) automaton works like LR(0) automaton but adds lookahead
- Each item is annotated using tokens that could follow it
- Lookahead is a subset of the FOLLOW set of the relevant non-terminal

LALR Parsing

- LR(1) automata can become very large
- Lookahead LR (LALR) parsing alleviates this problem
- Based on LR(1) automaton and merges states with the same core

- LL(1) $\subset$ SLR $\subset$ LALR $\subset$ LR(1) $\subset$ CFG

- LL(1) $\subset$ SLR $\subset$ LALR $\subset$ LR(1) $\subset$ CFG
- Regular $\subset$ Context-Free $\subset$ Context-Sensitive $\subset$ Recursively Enumerable

- LL(1) $\subset$ SLR $\subset$ LALR $\subset$ LR(1) $\subset$ CFG
- Regular $\subset$ Context-Free $\subset$ Context-Sensitive $\subset$ Recursively Enumerable
- Less powerful languages provide stronger guarantees

Outline

Summary

- LR(0) automata tell us available steps, but do not always tell us which step to take
  - It has no lookahead and is therefore relatively limited
- SLR parsing can resolve some conflicts not possible with LR(0)
  - LALR is less powerful than LR(1) but also less complex
- LR(1) grammars can be parsed using shift-reduce techniques one lookahead token
  - Can better represent structures commonly found in programming languages

References

[Thain, 2020] Thain, D. (2020). ***Introduction to Compilers and Language Design: Second Edition.*** http://compilerbook.org/.