Advanced LL Parsing Techniques

Radim Kocman

Faculty of Information Technology
Brno University of Technology
Božetěchova 2, Brno, Czech Republic

kocman@fit.vutbr.cz

BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF INFORMATION TECHNOLOGY

LTA 2022 (December 5, 2022)

Table of Contents

Motivation

- Syntactic analysis
  - The goal is to process the input string of tokens while following the derivation of a selected grammar.
  - This process recreates a derivation tree structure of the input so that semantic actions can follow rules of the grammar.
- LL(k) parsing
  - deterministic top-down method
  - it simulates the left-most derivation of the grammar
  - deterministic prediction for the next step is done according to the left-most unprocessed symbols of the sentential form and the input
  - k represents the number of symbols on the input used for the prediction
  - if $k = 1$, (1) is often omitted from the name
  - prediction can be implemented as a table look-up in so-called parsing table

Example LL(1) grammar G_1

$$G_1 = (\{S, A\}, \{a, b, c\}, P, S)$$

where P contains:

$$S \rightarrow aAb$$
$$S \rightarrow bAa$$
$$A \rightarrow cS$$
$$A \rightarrow \varepsilon$$

Example LL(2) Grammar

Example LL(2) grammar G_2

$$G_1 = (\{S, A\}, \{a, b\}, P, S)$$

where P contains:

$$S \rightarrow aAaa$$
$$S \rightarrow bAba$$
$$A \rightarrow b$$
$$A \rightarrow \varepsilon$$

Standard LL(1) Parsing

Standard LL(1) Parsing

- This is the most common technique. However, there exist many variations of this parsing that slightly differ in details.

- The grammar can contain empty strings at the right-hand side of the rules (ε-rules).
- We are using the following auxiliary symbols:
  - $ at the end of the input
  - $\#$ at the end of the generated sentential form

- Steps to create the parsing table:
  1. create First sets
  2. create Follow sets
  3. fill parsing table cells

- We are looking for first terminal symbols that can be produced from a selected symbol.
- We iteratively compute First sets for every symbol of the grammar until the sets stabilize.

Rules of G_1

$$S \to aAb \mid bAa, \quad A \to cS \mid \varepsilon$$

Initial sets

$First(a)$	$First(b)$	$First(c)$	$First(S)$	$First(A)$
$\{a\}$	$\{b\}$	$\{c\}$	$\{\}$	$\{\}$

Final iteration

$First(a)$	$First(b)$	$First(c)$	$First(S)$	$First(A)$
$\{a\}$	$\{b\}$	$\{c\}$	$\{a, b\}$	$\{c, \varepsilon\}$

Standard LL(1) Parsing – Follow Sets

- Due to ε-rules, we need to know what follows after non-terminals.
- We iteratively compute Follow sets for every non-terminal of the grammar until the sets stabilize.
- Computing $\mathit{Follow}(X)$, for every X in $Y \rightarrow \alpha X \beta$, we add terminal symbols from $\mathit{First}(\beta)$ to the set and, if it contains ε, we also add terminal symbols from $\mathit{Follow}(Y)$ to the set.

Rules of G_1

$$S \rightarrow aAb \mid bAa, \quad A \rightarrow cS \mid \varepsilon$$

Initial sets

$Follow(S)$	$Follow(A)$
$\{\$\}$	$\{\}$

First iteration

$Follow(S)$	$Follow(A)$
$\{\$\}$	$\{a, b\}$

Second iteration

$Follow(S)$	$Follow(A)$
$\{\$, a, b\}$	$\{a, b\}$

- We are filling two-dimensional table $M[X, a]$, where X are symbols of the sentential form, and a are symbols of the input.
- For every $X \to \alpha$, we add α on the index $M[X, a]$ where a is a terminal symbol from $First(\alpha)$ and, if it contains ε, we also add terminal symbols from $Follow(X)$.

Initial parsing table

	a	b	c	$
S				
A				
#				

Rules of G_1

$$S \to aAb \mid bAa, \quad A \to cS \mid \varepsilon$$

First sets

$First(a)$	$First(b)$	$First(c)$	$First(S)$	$First(A)$
$\{a\}$	$\{b\}$	$\{c\}$	$\{a, b\}$	$\{c, \varepsilon\}$

Follow sets

$Follow(S)$	$Follow(A)$
$\{\$, a, b\}$	$\{a, b\}$

Final parsing table

	a	b	c	$\$$
S	$a\,A\,b$	$b\,A\,a$		
A	ε	ε	$c\,S$	
$\#$				$accept$

Standard LL(1) Parsing – Parsing Table

- For the use with push-down automata we add pop rules for terminal symobls on the stack to the parsing table.

Parsing table

	a	b	c	$
S	$a\,A\,b$	$b\,A\,a$		
A	ε	ε	$c\,S$	
a	*pop*			
b		*pop*		
c			*pop*	
#				*accept*

Rules of G_2

$$S \to aAaa \mid bAba, \quad A \to b \mid \varepsilon$$

Parsing table

	a	b	$\$$
S	$a\,A\,a\,a$	$b\,A\,b\,a$	
A	ε	$b \mid \varepsilon$	
$\#$			*accept*

Full LL(1) Parsing

Full LL(1) Parsing

- The Follow sets from the standard LL(1) parsing only approximate the possible follow-up symbols, and the predictions can thus be wrong.
- We will compute precise sets of follow-up terminals according to the current context.
- We will use auxiliary LL(1) tables to compute new non-terminal symbols that hold information about possible follow-up terminals.
- Steps to create the parsing table:
  1. create First sets (same as before)
  2. create an auxiliary LL(1) table for the new start non-terminal $[S, \{\$\}]$
  3. create auxiliary LL(1) tables for other new non-terminals until their set stabilizes
  4. fill parsing table cells

Full LL(1) Parsing – Auxiliary Tables

- Computing a table for non-terminal $[X, N]$, for every $X \to \alpha$ we add a row into the table with following parts:
  - Next – set of possible first terminals computed from $First(\alpha)$ and N
  - Production – α
  - Follow – for every $\alpha = \beta Y \gamma$, add $[Y, M]$ where M is a set of possible first terminals computed from $First(\gamma)$ and N

Table $[S, \{\$\}]$

Next	Production	Follow

Rules of G_1

$$S \rightarrow aAb \mid bAa, \quad A \rightarrow cS \mid \varepsilon$$

First sets

$First(a)$	$First(b)$	$First(c)$	$First(S)$	$First(A)$
$\{a\}$	$\{b\}$	$\{c\}$	$\{a, b\}$	$\{c, \varepsilon\}$

Initial auxiliary table

Table $[S, \{\$\}]$

Next	Production	Follow
$\{a\}$	$a\ A\ b$	$[A, \{b\}]$
$\{b\}$	$b\ A\ a$	$[A, \{a\}]$

- We create remaining tables according to new non-nonterminals from previous Follow columns.

Other auxiliary table

Table $[A, \{b\}]$

Next	Production	Follow
$\{c\}$	$c\,S$	$[S, \{b\}]$
$\{b\}$	ε	–

Table $[A, \{a\}]$

Next	Production	Follow
$\{c\}$	$c\,S$	$[S, \{a\}]$
$\{a\}$	ε	–

Table $[S, \{b\}]$

Next	Production	Follow
$\{a\}$	$a\,A\,b$	$[A, \{b\}]$
$\{b\}$	$b\,A\,a$	$[A, \{a\}]$

Table $[S, \{a\}]$

Next	Production	Follow
$\{a\}$	$a\,A\,b$	$[A, \{b\}]$
$\{b\}$	$b\,A\,a$	$[A, \{a\}]$

Full LL(1) Parsing – Parsing Table

- The parsing table contains the new non-terminals instead of the original non-terminals of the grammar. We also replace non-terminals in the right-hand sides of rules.

Rules of G_1

$$S \to aAb \mid bAa, \quad A \to cS \mid \varepsilon$$

Final parsing table

	a	b	c	$\$$
$[S, \{\$\}]$	$a\,[A, \{b\}]\,b$	$b\,[A, \{a\}]\,a$		
$[S, \{a\}]$	$a\,[A, \{b\}]\,b$	$b\,[A, \{a\}]\,a$		
$[S, \{b\}]$	$a\,[A, \{b\}]\,b$	$b\,[A, \{a\}]\,a$		
$[A, \{a\}]$	ε		$c\,[S, \{a\}]$	
$[A, \{b\}]$		ε	$c\,[S, \{b\}]$	
$\#$				$accept$

Standard LL(1) parsing table

	a	b	c	$\$$
S	$a\,A\,b$	$b\,A\,a$		
A	ε	ε	$c\,S$	
$\#$				*accept*

Full LL(1) parsing table

	a	b	c	$\$$
$[S, \{\$\}]$	$a\,[A, \{b\}]\,b$	$b\,[A, \{a\}]\,a$		
$[S, \{a\}]$	$a\,[A, \{b\}]\,b$	$b\,[A, \{a\}]\,a$		
$[S, \{b\}]$	$a\,[A, \{b\}]\,b$	$b\,[A, \{a\}]\,a$		
$[A, \{a\}]$	ε		$c\,[S, \{a\}]$	
$[A, \{b\}]$		ε	$c\,[S, \{b\}]$	
$\#$				*accept*

General LL(k) Parsing

General LL(k) Parsing

- This technique generalizes full LL(1) parsing so that we can use more than one symbol on the input for the prediction.
- k has to be selected at the start
- this method works with sets of strings (not symbols)
- we are using k auxiliary symbols \$ at the end of the input
- Steps to create the parsing table:
  1. create First sets
  2. create a auxiliary LL(k) table for the new start non-terminal $[S, \{\$^k\}]$
  3. create auxiliary LL(k) tables for other new non-terminals until their set stabilizes
  4. fill parsing table cells

New string operation $\oplus_k$

$$a \oplus_2 bc = ab$$

$$\{a, ab, \varepsilon\} \oplus_2 \{aa, b\} = \{aa, ab, b\}$$

Rules of G_2

$$S \rightarrow aAaa \mid bAba, \quad A \rightarrow b \mid \varepsilon$$

$First_2$ Sets

$First_2(a)$	$First_2(b)$	$First_2(S)$	$First_2(A)$
$\{a\}$	$\{b\}$	$\{aa, ab, bb\}$	$\{b, \varepsilon\}$

Rules of G_2

$$S \rightarrow aAaa \mid bAba, \quad A \rightarrow b \mid \varepsilon$$

$First_2$ Sets

$First_2(a)$	$First_2(b)$	$First_2(S)$	$First_2(A)$
$\{a\}$	$\{b\}$	$\{aa, ab, bb\}$	$\{b, \varepsilon\}$

Initial auxiliary table

Table $[S, \{\$\$\}]$

Next	Production	Follow
$\{aa, ab\}$	$a\ A\ a\ a$	$[A, \{aa\}]$
$\{bb\}$	$b\ A\ b\ a$	$[A, \{ba\}]$

General LL(k) Parsing – Auxiliary Tables

- We create remaining tables according to new non-nonterminals from previous Follow columns.

Other auxiliary table

Table $[A, \{aa\}]$

Next	Production	Follow
$\{ba\}$	b	–
$\{aa\}$	ε	–

Table $[A, \{ba\}]$

Next	Production	Follow
$\{bb\}$	b	–
$\{ba\}$	ε	–

- The parsing table is indexed as $M[X, a]$, where X are symbols of the sentential form, and a are all possible k-length strings of input symbols (padded with $).

Rules of G_2

$$S \rightarrow aAaa \mid bAba, \quad A \rightarrow b \mid \varepsilon$$

Parsing table

	aa	ab	$a\$$	ba	bb	$b\$$	$\$\$$
$[S, \{\$\$\}]$	$a[A, \{aa\}]aa$	$a[A, \{aa\}]aa$			$b[A, \{aa\}]ba$		
$[A, \{aa\}]$	ε			b			
$[A, \{ba\}]$				ε	b		
$\#$							*accept*

- The position of pop rules depends on the first unprocessed symbol of the input.

Parsing table

	aa	ab	$a\$$	ba	bb	$b\$$	$\$\$$
$[S, \{\$\$\}]$	$a[A, \{aa\}]aa$	$a[A, \{aa\}]aa$			$b[A, \{aa\}]ba$		
$[A, \{aa\}]$	ε			b			
$[A, \{ba\}]$				ε	b		
a	*pop*	*pop*	*pop*				
b				*pop*	*pop*	*pop*	
$\#$							*accept*

LL(k) Parsing for Automaton with One-Symbol Reading Head

LL(k) Parsing for PDA

- We can modify the general LL(k) parsing table so that it is suitable for a standard push-down automaton with a one-symbol reading head.
- In the original concept of LL(k) parsing, states of the automaton are almost not utilized. Therefore, we can use states to create a symbol buffer.
- We always use only one $ at the end of the input.
- Steps to create the parsing table:
  1. create general LL(k) parsing table
  2. augment it with automaton states

Notation

- we use the standard notation for symbols on the stack
- we denote any terminal x of the input as $[x]$
- state buffer containing α is denoted as $:\alpha:$

Non-terminals of the LL(2) parsing table for G_2

For better readability, we set $[S, \{\$\$\}] = S$, $[A, \{aa\}] = A_1$, and $[A, \{ba\}] = A_2$.

Parsing table layout

	states of length $< k$	states of length $= k$
stack symbols	empty	parsing actions
input symbols	input reading	empty

Parsing table layout

	:0:	:a:	:b:	:aa:	:ab:	:$a\$$:	:ba:	:bb:	:$b\$$:	:$\$\$$:
S										
A_1										
A_2										
a										
b										
#										
[a]										
[b]										
[$]										

- State transitions depend only on k and terminals of the grammar. We need to fill the table in a way so that the states behave as a buffer.

Input reading part of the parsing table

	:0:	:a:	:b:
[a]	:a:	:aa:	:ba:
[b]	:b:	:ab:	:bb:
[\$]	:\$\$:	:a\$:	:b\$:

- Pop rules read symbols from the stack and the state buffer.

Rules of G_2

$$S \rightarrow aAaa \mid bAba, \quad A \rightarrow b \mid \varepsilon$$

Final parsing table

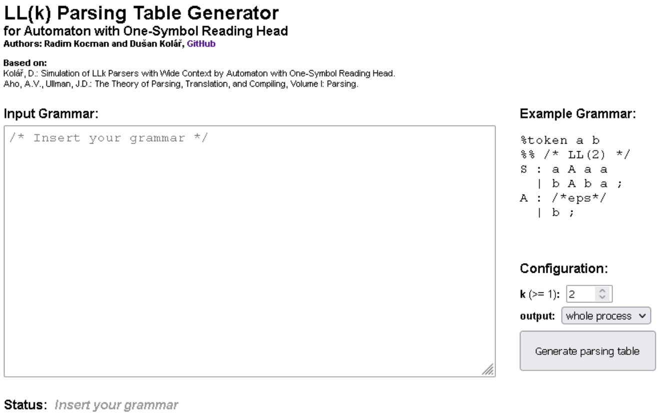
	:0:	:a:	:b:	:aa:	:ab:	:a$:	:ba:	:bb:	:b$:	:$$:
S				aA_1aa	aA_1aa			bA_2ba		
A_1				ε			b			
A_2							ε	b		
a				*pop* :a:	*pop* :b:	*pop* :$$:				
b							*pop* :a:	*pop* :b:	*pop* :$$:	
#										*accept*
[a]	:a:	:aa:	:ba:							
[b]	:b:	:ab:	:bb:							
[$]	:$$:	:a$:	:b$:							

LL(k) Parsing Table Generator

LL(k) Parsing Table Generator

https://www.fit.vutbr.cz/~kocman/llkptg/

https://github.com/rkocman/LLk-Parsing-Table-Generator



References

- Dick Grune and Ceriel J.H. Jacobs
  Parsing Techniques: A Practical Guide
  Springer, 2nd edition (2008)

- Alfred V. Aho and Jeffrey D. Ullman
  The Theory of Parsing, Translation, and Compiling,
  Volume I: Parsing
  Prentice Hall, Inc. (1972)

- Dušan Kolář
  Simulation of LLk Parsers with Wide Context
  by Automaton with One-Symbol Reading Head
  MOSIS 2004

And that’s it!