Universiteit Utrecht

[Faculty of **Science**
Information and Computing Sciences]

Talen en Compilers

2022 - 2023, period 2

David van Balen

Department of Information and Computing Sciences
Utrecht University

2023-01-23

15. LR parsing (2)

This lecture

LR parsing (2)

LR parsing algorithms

15.1 LR parsing algorithms

Balanced parentheses

$$
\begin{aligned}
S &\rightarrow E\$ \\
E &\rightarrow (E)E \\
E &\rightarrow \varepsilon
\end{aligned}
$$

$$
\text{start} \longrightarrow \boxed{\begin{aligned}
S &\rightarrow \bullet E\$ \\
E &\rightarrow \bullet (E)E \\
E &\rightarrow \bullet
\end{aligned}}
$$

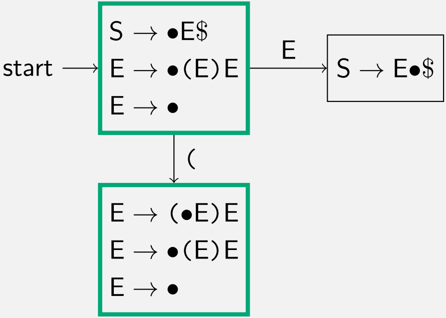
Balanced parentheses

$S \rightarrow E\$$
$E \rightarrow (E)E$
$E \rightarrow \varepsilon$

$$
\text{start} \longrightarrow \boxed{\begin{array}{l} S \rightarrow \bullet E\$ \\ E \rightarrow \bullet (E)E \\ E \rightarrow \bullet \end{array}} \xrightarrow{(} \boxed{\begin{array}{l} E \rightarrow (\bullet E)E \\ E \rightarrow \bullet (E)E \\ E \rightarrow \bullet \end{array}}
$$

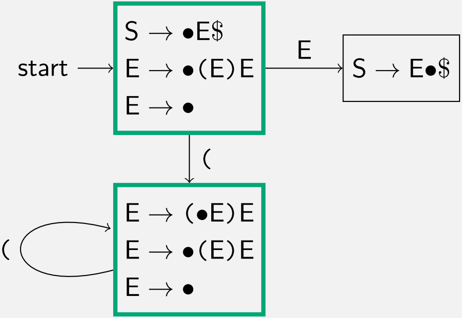
Balanced parentheses

$$
\begin{aligned}
S &\rightarrow E\$ \\
E &\rightarrow (E)E \\
E &\rightarrow \varepsilon
\end{aligned}
$$



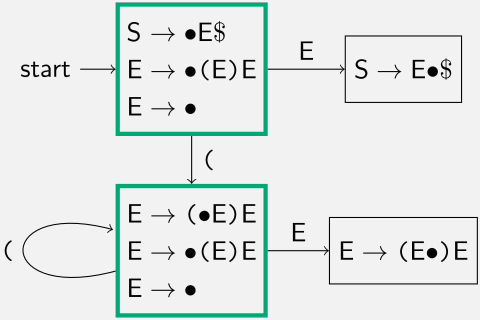
Balanced parentheses

$S \to E\$$
$E \to (E)E$
$E \to \varepsilon$



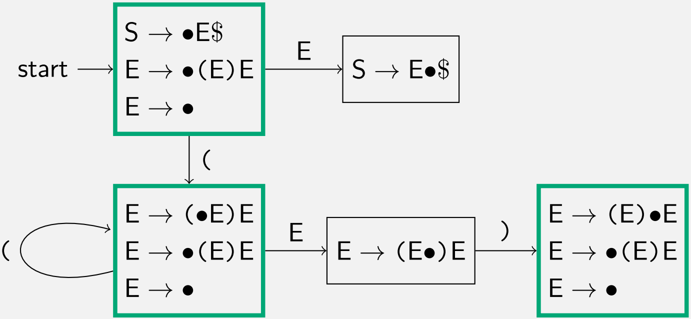
Balanced parentheses

$$S \to E\$$$
$$E \to (E)E$$
$$E \to \varepsilon$$



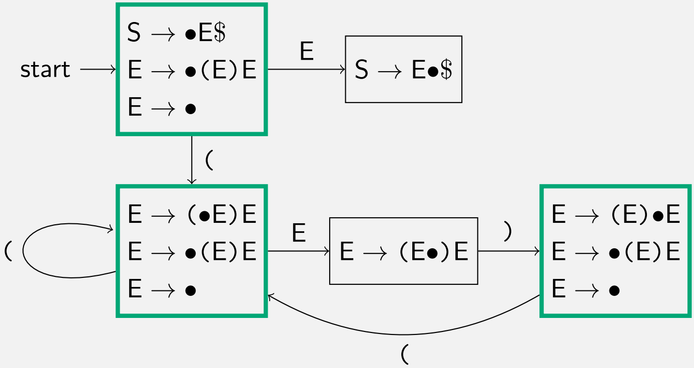
Balanced parentheses

$$S \to E\$$$
$$E \to (E)E$$
$$E \to \varepsilon$$



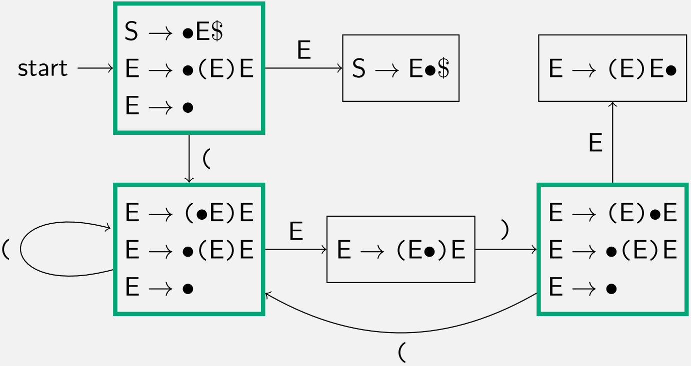
Balanced parentheses

$S \to E\$$
$E \to (E)E$
$E \to \varepsilon$



Balanced parentheses

$S \to E\$$
$E \to (E)E$
$E \to \varepsilon$



SLR(1)

The follow set (in LL sense) of E is $\{), \$\}$. We can use this information to remove the conflicts:

- Reducing by E only makes sense if the next input symbol is in that set.
- We can always shift (, resolving all conflicts in this case.

SLR(1)

The follow set (in LL sense) of E is $\{), \$\}$. We can use this information to remove the conflicts:

- Reducing by E only makes sense if the next input symbol is in that set.
- We can always shift (, resolving all conflicts in this case.

SLR(1) is LR(0) extended with **simple lookahead**:

- Resolve conflicts by looking at the next symbol.
- Use the follow sets of each nonterminal.

SLR(1) Tables

LR(0) and SLR(1) automata can be defined using a table:

- States are numbered, and correspond to item sets,
- Production rules in the grammar are also numbered,
- For every state and input symbol, you can:
  - Shift and go to state n: sn,
  - Reduce using rule n: rn,
  - Go to state n without shifting: gn.

When running an automaton from a table, we push both states and symbols to the stack, and we pop when we reduce a rule.

SLR(1) Table

$$\begin{array}{lll} S \to E\$ & & (1) \\ E \to (E)E & & (2) \\ E \to \varepsilon & & (3) \end{array}$$

	action			
state	(	)	$	E
1	s2	r3	r3	g3
2	s2	r3	r3	g4
3			acc	
4		s5		
5	s2	r3	r3	g6
6		r2	r2	

SLR(1) Table

$$\begin{array}{lll} S \to E\$ & (1) \\ E \to (E)E & (2) \\ E \to \varepsilon & (3) \end{array}$$

	action			
state	(	)	$	E
1	s2	r3	r3	g3
2	s2	r3	r3	g4
3			acc	
4		s5		
5	s2	r3	r3	g6
6		r2	r2	

1	(()())()$
1(2	()())()$
1(2(2	)())()$

SLR(1) Table

$$
\begin{array}{lll}
S \to E\$ & & (1) \\
E \to (E)E & & (2) \\
E \to \varepsilon & & (3)
\end{array}
$$

	action			
state	(	)	$	E
1	s2	r3	r3	g3
2	s2	r3	r3	g4
3			acc	
4		s5		
5	s2	r3	r3	g6
6		r2	r2	

1	(()())()$
1(2	()())()$
1(2(2	)())()$
1(2(2E4	)())()$

SLR(1) Table

$$\begin{array}{lll} S \to E\$ & & (1) \\ E \to (E)E & & (2) \\ E \to \varepsilon & & (3) \end{array}$$

	action			
state	(	)	$	E
1	s2	r3	r3	g3
2	s2	r3	r3	g4
3			acc	
4		s5		
5	s2	r3	r3	g6
6		r2	r2	

1	(()())()$
1(2	()())()$
1(2(2	)())()$
1(2(2E4	)())()$
1(2(2E4)5	())()$
1(2(2E4)5(2	))()$
1(2(2E4)5(2E4	))()$
1(2(2E4)5(2E4)5	)()$

Universiteit Utrecht
[Faculty of Science
Information and Computing Sciences]

SLR(1) Table

$$\begin{array}{lll} S & \to E\$ & (1) \\ E & \to (E)E & (2) \\ E & \to \varepsilon & (3) \end{array}$$

	action			
state	(	)	$	E
1	s2	r3	r3	g3
2	s2	r3	r3	g4
3			acc	
4		s5		
5	s2	r3	r3	g6
6		r2	r2	

1	(()())()$
1(2	()())()$
1(2(2	)())()$
1(2(2E4	)())()$
1(2(2E4)5	())()$
1(2(2E4)5(2	))()$
1(2(2E4)5(2E4	))()$
1(2(2E4)5(2E4)5	)()$
1(2(2E4)5(2E4)5E6	)()$

SLR(1) Table

$$
\begin{array}{lll}
S \to E\$ & & (1) \\
E \to (E)E & & (2) \\
E \to \varepsilon & & (3)
\end{array}
$$

	action			
state	(	)	$	E
1	s2	r3	r3	g3
2	s2	r3	r3	g4
3			acc	
4		s5		
5	s2	r3	r3	g6
6		r2	r2	

1	(()())()$
1(2	()())()$
1(2(2	)())()$
1(2(2E4	)())()$
1(2(2E4)5	())()$
1(2(2E4)5(2	))()$
1(2(2E4)5(2E4	))()$
1(2(2E4)5(2E4)5	)()$
1(2(2E4)5(2E4)5E6	)()$
1(2(2E4)5E6	)()$

Universiteit Utrecht

[Faculty of Science
Information and Computing Sciences]

SLR(1) Table

$$\begin{array}{lll} S & \rightarrow E\$ & (1) \\ E & \rightarrow (E)E & (2) \\ E & \rightarrow \varepsilon & (3) \end{array}$$

	action			
state	(	)	$	E
1	s2	r3	r3	g3
2	s2	r3	r3	g4
3			acc	
4		s5		
5	s2	r3	r3	g6
6		r2	r2	

1	(()())()$
1(2	()())()$
1(2(2	)())()$
1(2(2E4	)())()$
1(2(2E4)5	())()$
1(2(2E4)5(2	))()$
1(2(2E4)5(2E4	))()$
1(2(2E4)5(2E4)5	)()$
1(2(2E4)5(2E4)5E6	)()$
1(2(2E4)5E6	)()$
1(2E4	)()$

SLR(1) Table

$$\begin{array}{lll} S \to E\$ & (1) \\ E \to (E)E & (2) \\ E \to \varepsilon & (3) \end{array}$$

	action			
state	(	)	$	E
1	s2	r3	r3	g3
2	s2	r3	r3	g4
3			acc	
4		s5		
5	s2	r3	r3	g6
6		r2	r2	

1	(()())()$
1(2	()())()$
1(2(2	)())()$
1(2(2E4	)())()$
1(2(2E4)5	())()$
1(2(2E4)5(2	))()$
1(2(2E4)5(2E4	))()$
1(2(2E4)5(2E4)5	)()$
1(2(2E4)5(2E4)5E6	)()$
1(2(2E4)5E6	)()$
1(2E4	)()$
1(2E4)5	()$
1(2E4)5(2	)$
1(2E4)5(2E4	)$
1(2E4)5(2E4)5	$
1(2E4)5(2E4)5E6	$
1(2E4)5E6	$

SLR(1) Table

$$
\begin{array}{lll}
S \to E\$ & & (1) \\
E \to (E)E & & (2) \\
E \to \varepsilon & & (3)
\end{array}
$$

	action			
state	(	)	$	E
1	s2	r3	r3	g3
2	s2	r3	r3	g4
3			acc	
4		s5		
5	s2	r3	r3	g6
6		r2	r2	

1	(()())()$
1(2	()())()$
1(2(2	)())()$
1(2(2E4	)())()$
1(2(2E4)5	())()$
1(2(2E4)5(2	))()$
1(2(2E4)5(2E4	))()$
1(2(2E4)5(2E4)5	)()$
1(2(2E4)5(2E4)5E6	)()$
1(2(2E4)5E6	)()$
1(2E4	)()$
1(2E4)5	()$
1(2E4)5(2	)$
1(2E4)5(2E4	)$
1(2E4)5(2E4)5	$
1(2E4)5(2E4)5E6	$
1(2E4)5E6	$
1E3	$

Full (canonical) LR(1)

SLR(1) uses the follow set of nonterminals, but one could be more specific.

Full (canonical) LR(1)

SLR(1) uses the follow set of nonterminals, but one could be more specific.

Definition

An LR(1) item is a pair of an LR(0) item and a single terminal.

An item encodes where in the parsing process we are, and what is the first symbol that is expected after the current production is finished.

- The closure operation is adapted to keep track of lookahead symbols.
- As a result, the automaton is usually significantly larger. Therefore, canonical LR(1) parsers are rarely used in practice.

Using the LR(1) automaton

Once computed, we can use the lookahead symbols to resolve situations that would be conflicts in the LR(0) situation:

- A reduction applies only if the next input symbol matches the lookahead symbol of the item.
- In a state with multiple reduce options, the lookahead symbol can be compared with the next input symbol in order to choose one.
- In a state with shift and reduce options, there is only a conflict if there is both a shift and reduce option for the next input symbol.

Balanced parentheses using LR(1)

$$\begin{array}{l} S \to \bullet E \quad , \$ \\ E \to \bullet(E)E, \$ \\ E \to \bullet \quad , \$ \end{array}$$

$$\begin{array}{l} E \to (\bullet E)E, \$ \\ E \to \bullet(E)E,) \\ E \to \bullet \quad ,) \end{array}$$

$$\begin{array}{l} E \to (\bullet E)E,) \\ E \to \bullet(E)E,) \\ E \to \bullet \quad ,) \end{array}$$

$$S \to E\bullet, \$$$

$$E \to (E\bullet)E, \$$$

$$E \to (E\bullet)E,)$$

$$\begin{array}{l} E \to (E)\bullet E, \$ \\ E \to \bullet(E)E, \$ \\ E \to \bullet \quad , \$ \end{array}$$

$$\begin{array}{l} E \to (E)\bullet E,) \\ E \to \bullet(E)E,) \\ E \to \bullet \quad ,) \end{array}$$

$$E \to (E)E\bullet, \$$$

$$E \to (E)E\bullet,)$$

LALR(1)

The LALR(1) algorithm tries to find a compromise between the power of LR(1) and the simplicity of SLR(1):

- LR(1) items are used for computing fine-grained lookahead information,
- only the LR(0) automaton is used.

LALR(1)

The LALR(1) algorithm tries to find a compromise between the power of LR(1) and the simplicity of SLR(1):

- LR(1) items are used for computing fine-grained lookahead information,
- only the LR(0) automaton is used.

This can be achieved by merging states of the LR(1) algorithm that differ only in their lookahead symbols, while still using the lookahead information to resolve conflicts.

LALR(1)

The LALR(1) algorithm tries to find a compromise between the power of LR(1) and the simplicity of SLR(1):

- LR(1) items are used for computing fine-grained lookahead information,
- only the LR(0) automaton is used.

This can be achieved by merging states of the LR(1) algorithm that differ only in their lookahead symbols, while still using the lookahead information to resolve conflicts.

There are several algorithms for computing LALR(1) lookahead information more efficiently, so that the full LR(1) automaton is not required.

LR(1) vs. LALR(1) vs. SLR(1)

Unfortunately, merging the LR(1) states can introduce conflicts.

Therefore:

- there are more LR(1) grammars than LALR(1) grammars,
- but there are more LALR(1) grammars than SLR(1) grammars.

The LALR(1) algorithm is the one that is used in most parser generators (like Yacc, Bison, or Happy).

Generalized LR (GLR)

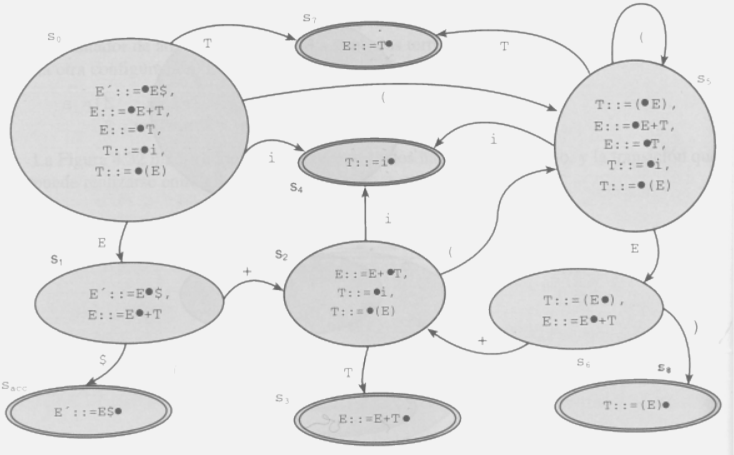
Another option to deal with conflicts is to accept that locally, there are multiple options:

- on a conflict, we split the stack and try all options in a breadth-first manner;
- if the common parts of the stack are shared and the resulting nondeterminism can be resolved quickly, there is not much performance overhead.

Let's do an example

Compute the LR (0) automaton for

$$
\begin{aligned}
\mathsf{E}' &\rightarrow \mathsf{E}\$ \\
\mathsf{E} &\rightarrow \mathsf{T} \mid \mathsf{E} + \mathsf{T} \\
\mathsf{T} &\rightarrow \mathtt{i} \mid (\mathsf{E})
\end{aligned}
$$


S0
E´::=●E$,
E::=●E+T,
E::=●T,
T::=●i,
T::=●(E)
S7
E::=T●
S5
T::=(●E),
E::=●E+T,
E::=●T,
T::=●i,
T::=●(E)
S4
T::=i●
S1
E´::=E●$,
E::=E●+T
S2
E::=E+●T,
T::=●i,
T::=●(E)
S6
T::=(E●),
E::=E●+T
Sacc
E´::=E$●
S3
E::=E+T●
S8
T::=(E)●
T
(
i
E
+
$
)