Universiteit Utrecht

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

Talen en Compilers

2019 - 2020, period 2

Jurriaan Hage

Department of Information and Computing Sciences
Utrecht University

2020-01-16

14. LR parsing (1)

This lecture

LR parsing (1)

- Basic idea
- The LR(0) automaton
- Conflicts

14.1 Basic idea

Setup

The LR parsing procedure is similar to that of LL parsing. An LR parser maintains a **state** consisting of

- a stack of symbols,
- and the current (remaining) input.

We write (α, x) to denote the state consisting of the stack α and input x. The stack grows to the right.

We follow the literature and ignore ϵ.

Actions

We start with state (ε, x) where x is the input string, i.e., with the empty stack. In each step, we do one of the following:

Shift The first symbol in the input moves to the stack.

$$(x, av) \rightsquigarrow (xa, v)$$

Reduce If the top symbols of the stack match the right hand side of a production, we can replace them with the corresponding left hand side.

$$(\beta\alpha, v) \rightsquigarrow (\beta N, v) \textbf{ if } N \to \alpha$$

The parser succeeds if we can reach the state (S, ε) where S is the start symbol. Otherwise, the parser fails.

Acceptance

An LR machine **accepts** a word if any sequence of choices leads to success.

- It is not always clear when to shift and when to reduce.
- As with LL, the LR approach is at first **nondeterministic**.

Example

$S \to aS \mid cS \mid b$

Let us parse aab:

stack	input	
ε	aab	initial state – we have to shift

Example

$S \to aS \mid cS \mid b$

Let us parse aab:

stack	input	
ε	aab	
a	ab	shift

Example

$S \to aS \mid cS \mid b$

Let us parse aab:

stack	input	
ε	aab	
a	ab	
aa	b	shift

Example

$S \to aS \mid cS \mid b$

Let us parse aab:

stack	input	
ε	aab	
a	ab	
aa	b	
aab	ε	reduce

Example

$S \to aS \mid cS \mid b$

Let us parse aab:

stack	input	
ε	aab	
a	ab	
aa	b	
aab	ε	
aaS	ε	reduce

Example

$S \to aS \mid cS \mid b$

Let us parse aab:

stack	input	
ε	aab	
a	ab	
aa	b	
aab	ε	
aaS	ε	
aS	ε	reduce

Example

$S \to aS \mid cS \mid b$

Let us parse aab:

stack	input	
ε	aab	
a	ab	
aa	b	
aab	ε	
aaS	ε	
aS	ε	
S	ε	parse successful

Another example

ε	ccccba	initial state, shift

$$
\begin{aligned}
S &\to cA \mid b \\
A &\to cBC \mid bSA \mid a \\
B &\to cc \mid Cb \\
C &\to aS \mid ba
\end{aligned}
$$

Let us parse ccccba.

Another example

ε	ccccba	
c	cccba	shift

$$
\begin{aligned}
S &\to cA \mid b \\
A &\to cBC \mid bSA \mid a \\
B &\to cc \mid Cb \\
C &\to aS \mid ba
\end{aligned}
$$

Let us parse ccccba.

Another example

ε	ccccba	
c	cccba	
cc	ccba	let's try reducing

$$
\begin{aligned}
S &\to cA \mid b \\
A &\to cBC \mid bSA \mid a \\
B &\to cc \mid Cb \\
C &\to aS \mid ba
\end{aligned}
$$

Let us parse ccccba.

Another example

$$
\begin{aligned}
S &\to cA \mid b \\
A &\to cBC \mid bSA \mid a \\
B &\to cc \mid Cb \\
C &\to aS \mid ba
\end{aligned}
$$

ε	ccccba	
c	cccba	
cc	ccba	
B	ccba	shift

Let us parse ccccba.

Another example

$S \to cA \mid b$
$A \to cBC \mid bSA \mid a$
$B \to cc \mid Cb$
$C \to aS \mid ba$

Let us parse ccccba.

ε	ccccba	
c	cccba	
cc	ccba	
B	ccba	
Bc	cba	shift

Another example

$$
\begin{aligned}
S &\to cA \mid b \\
A &\to cBC \mid bSA \mid a \\
B &\to cc \mid Cb \\
C &\to aS \mid ba
\end{aligned}
$$

ε	ccccba
c	cccba
cc	ccba
B	ccba
Bc	cba
Bcc	ba

let's try reducing

Let us parse ccccba.

Another example

$$
\begin{array}{l}
S \to cA \mid b \\
A \to cBC \mid bSA \mid a \\
B \to cc \mid Cb \\
C \to aS \mid ba
\end{array}
$$

ε	ccccba	
c	cccba	
cc	ccba	
B	ccba	
Bc	cba	
Bcc	ba	
BB	ba	shift

Let us parse ccccba.

Another example

$$
\begin{aligned}
S &\to cA \mid b \\
A &\to cBC \mid bSA \mid a \\
B &\to cc \mid Cb \\
C &\to aS \mid ba
\end{aligned}
$$

Let us parse ccccba.

ε	ccccba
c	cccba
cc	ccba
B	ccba
Bc	cba
Bcc	ba
BB	ba
BBb	a

let's try reducing

Another example

$$
\begin{aligned}
S &\to cA \mid b \\
A &\to cBC \mid bSA \mid a \\
B &\to cc \mid Cb \\
C &\to aS \mid ba
\end{aligned}
$$

Let us parse ccccba.

ε	ccccba	
c	cccba	
cc	ccba	
B	ccba	
Bc	cba	
Bcc	ba	
BB	ba	
BBb	a	
BBS	a	shift

Another example

$$
\begin{aligned}
S &\to cA \mid b \\
A &\to cBC \mid bSA \mid a \\
B &\to cc \mid Cb \\
C &\to aS \mid ba
\end{aligned}
$$

Let us parse ccccba.

ε	ccccba	
c	cccba	
cc	ccba	
B	ccba	
Bc	cba	
Bcc	ba	
BB	ba	
BBb	a	
BBS	a	
BBSa	ε	reduce

Another example

$S \to cA \mid b$
$A \to cBC \mid bSA \mid a$
$B \to cc \mid Cb$
$C \to aS \mid ba$

Let us parse ccccba.

ε	ccccba	
c	cccba	
cc	ccba	
B	ccba	
Bc	cba	
Bcc	ba	
BB	ba	
BBb	a	
BBS	a	
BBSa	ε	
BBSA	ε	we're stuck

Another example

$$
\begin{aligned}
S &\to cA \mid b \\
A &\to cBC \mid bSA \mid a \\
B &\to cc \mid Cb \\
C &\to aS \mid ba
\end{aligned}
$$

Let us parse ccccba.

ε	ccccba
c	cccba
cc	ccba
B	ccba
Bc	cba
Bcc	ba
BB	ba
BBb	a
BBS	a
BBSa	ε
BBSA	ε

Reducing as early as possible does not seem to be a good strategy in general.

Another example – another strategy

ε	ccccba	initial state, shift

$$
\begin{aligned}
S &\to cA \mid b \\
A &\to cBC \mid bSA \mid a \\
B &\to cc \mid Cb \\
C &\to aS \mid ba
\end{aligned}
$$

Let us parse ccccba.

Another example – another strategy

ε	ccccba	
c	cccba	shift

$$
\begin{aligned}
S &\to cA \mid b \\
A &\to cBC \mid bSA \mid a \\
B &\to cc \mid Cb \\
C &\to aS \mid ba
\end{aligned}
$$

Let us parse ccccba.

Another example – another strategy

ε	ccccba	
c	cccba	
cc	ccba	let's try shifting

$$
\begin{aligned}
S &\to cA \mid b \\
A &\to cBC \mid bSA \mid a \\
B &\to cc \mid Cb \\
C &\to aS \mid ba
\end{aligned}
$$

Let us parse ccccba.

Another example – another strategy

$$
\begin{array}{ll}
S \to cA \mid b \\
A \to cBC \mid bSA \mid a \\
B \to cc \mid Cb \\
C \to aS \mid ba
\end{array}
$$

ε	ccccba
c	cccba
cc	ccba
ccc	cba

let's try shifting

Let us parse ccccba.

Another example – another strategy

$$
\begin{aligned}
S &\to cA \mid b \\
A &\to cBC \mid bSA \mid a \\
B &\to cc \mid Cb \\
C &\to aS \mid ba
\end{aligned}
$$

ε	ccccba
c	cccba
cc	ccba
ccc	cba
cccc	ba

let's try reducing

Let us parse ccccba.

Another example – another strategy

$S \to cA \mid b$
$A \to cBC \mid bSA \mid a$
$B \to cc \mid Cb$
$C \to aS \mid ba$

ε	ccccba	
c	cccba	
cc	ccba	
ccc	cba	
cccc	ba	
ccB	ba	shift

Let us parse ccccba.

Another example – another strategy

$$
\begin{array}{l}
S \to cA \mid b \\
A \to cBC \mid bSA \mid a \\
B \to cc \mid Cb \\
C \to aS \mid ba
\end{array}
$$

ε	ccccba
c	cccba
cc	ccba
ccc	cba
cccc	ba
ccB	ba
ccBb	a

let's try shifting

Let us parse ccccba.

Another example – another strategy

$$
\begin{aligned}
S &\to cA \mid b \\
A &\to cBC \mid bSA \mid a \\
B &\to cc \mid Cb \\
C &\to aS \mid ba
\end{aligned}
$$

Let us parse ccccba.

ε	ccccba
c	cccba
cc	ccba
ccc	cba
cccc	ba
ccB	ba
ccBb	a
ccBba	ε

reduce, but how?

Another example – another strategy

$$
\begin{array}{l}
S \to cA \mid b \\
A \to cBC \mid bSA \mid a \\
B \to cc \mid Cb \\
C \to aS \mid ba
\end{array}
$$

Let us parse ccccba.

ε	ccccba	
c	cccba	
cc	ccba	
ccc	cba	
cccc	ba	
ccB	ba	
ccBb	a	
ccBba	ε	
ccBC	ε	reduce

Another example – another strategy

$$
\begin{aligned}
S &\to cA \mid b \\
A &\to cBC \mid bSA \mid a \\
B &\to cc \mid Cb \\
C &\to aS \mid ba
\end{aligned}
$$

Let us parse ccccba.

ε	ccccba	
c	cccba	
cc	ccba	
ccc	cba	
cccc	ba	
ccB	ba	
ccBb	a	
ccBba	ε	
ccBC	ε	
cA	ε	reduce

Another example – another strategy

$S \to cA \mid b$
$A \to cBC \mid bSA \mid a$
$B \to cc \mid Cb$
$C \to aS \mid ba$

Let us parse ccccba.

ε	ccccba
c	cccba
cc	ccba
ccc	cba
cccc	ba
ccB	ba
ccBb	a
ccBba	ε
ccBC	ε
cA	ε
S	ε

parse successful

LR vs. LL

LR

ε	ccccba
c	cccba
cc	ccba
ccc	cba
cccc	ba
ccB	ba
ccBb	a
ccBba	ε
ccBC	ε
cA	ε
S	ε

LL

S	ccccba
cA	ccccba
A	cccba
cBC	cccba
BC	ccba
ccC	ccba
cC	cba
C	ba
ba	ba
a	a
ε	ε

LR vs. LL – contd.

LR derivation

$$S \Rightarrow cA \Rightarrow ccBC \Rightarrow ccBba \Rightarrow ccccba$$

This is a **rightmost** derivation.

LL derivation

$$S \Rightarrow cA \Rightarrow ccBC \Rightarrow ccccC \Rightarrow ccccba$$

This is a **leftmost** derivation.

LR vs. LL – contd.

LR derivation

$$S \Rightarrow cA \Rightarrow ccBC \Rightarrow ccBba \Rightarrow ccccba$$

This is a **rightmost** derivation.

LL derivation

$$S \Rightarrow cA \Rightarrow ccBC \Rightarrow ccccC \Rightarrow ccccba$$

This is a **leftmost** derivation.

The name LR is because

- the input is consumed starting from the **left** (the 'L'),
- and a **rightmost** derivation is returned (the 'R').

14.2 The LR(0) automaton

How to pick actions

During the parse process:

- we can only shift if there are symbols left in the input,
- we should only shift if there is an option of a later reduction,
- we can only reduce if the top of the stack matches a right hand side.

We thus have to keep track of where in the productions we might currently be while recognizing the input.

Explicit end of input

The following development is easier if we require the input grammar to have an explicit "end of input" marker (written \$):

$$
\begin{aligned}
S' &\to S\$ \\
S &\to cA \mid b \\
A &\to cBC \mid bSA \mid a \\
B &\to cc \mid Cb \\
C &\to aS \mid ba
\end{aligned}
$$

Items

Definition

An **item** (also called **LR(0)** item) is a production together with a **marked position** on the right hand side. The position can be either at the beginning, between two symbols, or at the end.

Items

Definition

An **item** (also called **LR(0)** item) is a production together with a **marked position** on the right hand side. The position can be either at the beginning, between two symbols, or at the end.

For example, for the production

$$A \to cBC$$

we obtain the following items:

$$A \to \bullet cBC$$
$$A \to c\bullet BC$$
$$A \to cB\bullet C$$
$$A \to cBC\bullet$$

Start

Our example grammar:

$$
\begin{aligned}
S' &\to S\$ \\
S &\to cA \mid b \\
A &\to cBC \mid bSA \mid a \\
B &\to cc \mid Cb \\
C &\to aS \mid ba
\end{aligned}
$$

At the beginning of the parse process, we are at the beginning of recognizing the right hand side of the start symbol.

This corresponds to the item:

$$S' \to \bullet S\$$$

Closure of an item (set)

If the marker in an item is in front of a nonterminal, we can in fact be at the beginning of recognizing the right hand side of that nonterminal, too.

Closure of an item (set)

If the marker in an item is in front of a nonterminal, we can in fact be at the beginning of recognizing the right hand side of that nonterminal, too.

We add these items and repeat the procedure until we reach a fixed point, the **closure** of the item set:

$$
\begin{aligned}
S' &\to \bullet S\$ \\
S &\to \bullet cA \\
S &\to \bullet b
\end{aligned}
$$

These three items describe a **state** in an **automaton** augmenting the parser.

Building the automaton

A state such as

$$
\begin{aligned}
S' &\rightarrow \bullet S\$ \\
S &\rightarrow \bullet cA \\
S &\rightarrow \bullet b
\end{aligned}
$$

tells us a few things:

- We cannot reduce in this state, because no items in the current state have position markers at the end of a right hand side.
- We should shift on seeing a c or a b.
- There cannot be a valid parse if the next input symbol is an a, so we can signal an error.

Transitions

Transitions in the automaton are labelled with **symbols**, i.e., transitions can be labelled with both terminals and nonterminals.

Transitions

Transitions in the automaton are labelled with **symbols**, i.e., transitions can be labelled with both terminals and nonterminals.

In any case, to determine the new state, we move the position marker where applicable and compute the closure of the remaining items.

Transition example

From state 0,

$$
\begin{aligned}
S' &\rightarrow \bullet S\$ \\
S &\rightarrow \bullet cA \\
S &\rightarrow \bullet b
\end{aligned}
$$

on seeing a c we move to state 1:

$$
\begin{aligned}
S &\rightarrow c\bullet A \\
A &\rightarrow \bullet cBC \\
A &\rightarrow \bullet bSA \\
A &\rightarrow \bullet a
\end{aligned}
$$

This is another **shift state**.

Transition example – contd.

From state 1,

$$
\begin{aligned}
S &\rightarrow c \bullet A \\
A &\rightarrow \bullet cBC \\
A &\rightarrow \bullet bSA \\
A &\rightarrow \bullet a
\end{aligned}
$$

on seeing another c we move to state 2:

$$
\begin{aligned}
A &\rightarrow c \bullet BC \\
B &\rightarrow \bullet cc \\
B &\rightarrow \bullet Cb \\
C &\rightarrow \bullet aS \\
C &\rightarrow \bullet ba
\end{aligned}
$$

Still a **shift state**.

Transition example – contd.

From state 2,

$$
\begin{aligned}
A &\rightarrow c \bullet BC \\
B &\rightarrow \bullet cc \\
B &\rightarrow \bullet Cb \\
C &\rightarrow \bullet aS \\
C &\rightarrow \bullet ba
\end{aligned}
$$

on seeing yet another c we move to state 3:

$$
B \rightarrow c \bullet c
$$

Still a **shift state**.

Transition example – contd.

From state 3,

> $B \rightarrow c \bullet c$

on seeing yet another c we move to state 4:

> $B \rightarrow cc \bullet$

A state with items that are marked in end position is called a **reduce state**.

Transition example – contd.

From state 3,

> $B \rightarrow c \bullet c$

on seeing yet another c we move to state 4:

> $B \rightarrow cc \bullet$

A state with items that are marked in end position is called a **reduce state**.

If there are no other items in the state (as here), the automaton has no outgoing transitions in the state.

How to reduce?

On the stack, we remember not only the symbols we push, but also the states of the automaton we went through.

How to reduce?

On the stack, we remember not only the symbols we push, but also the states of the automaton we went through.

Since we reduce the two cs by a B, we go back to state 2,

$$
\begin{aligned}
A &\to c \bullet BC \\
B &\to \bullet cc \\
B &\to \bullet Cb \\
C &\to \bullet aS \\
C &\to \bullet ba
\end{aligned}
$$

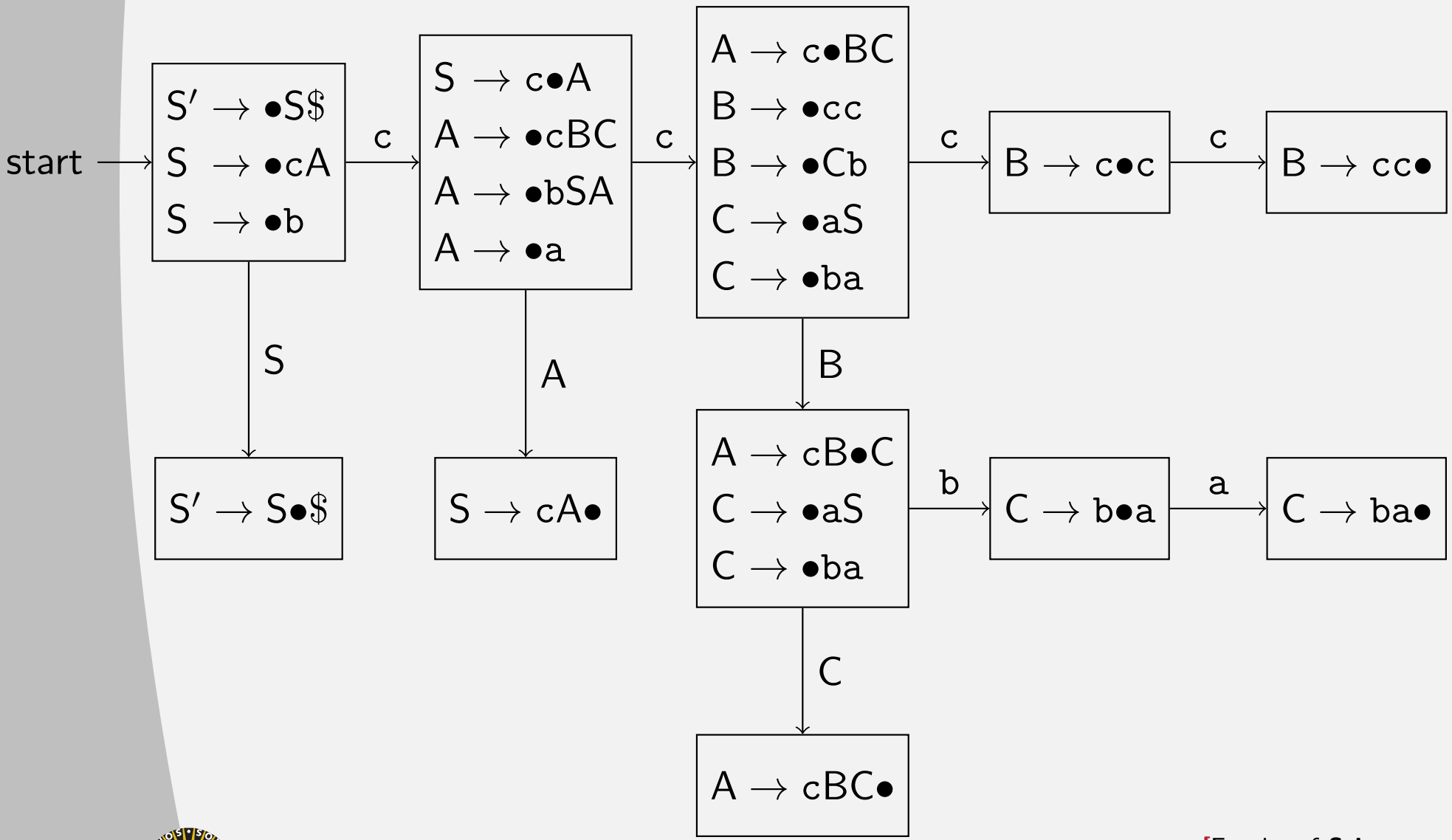
and assume we have read a B.

Summary

The automaton tells us what to do:

- In a shift state, we shift the state and the next symbol if it is among the transitions, or signal an error otherwise.
- In a reduce state, we remove the right hand side from the stack, look at the state, look where we're going when reading the left hand side, put the left hand side on the stack, and go on.

A part of the automaton



The complete automaton

The automaton can be rather large.

- In our example case, the automaton has 20 states.

Suitable for a generator, not really suitable for manual generation.

14.3 Conflicts

Types of conflicts

There are two types of conflicts that can arise:

- If a state allows both shifting and reducing, there is a **shift-reduce conflict**.
- If a state allows reducing by more than one production, there is a **reduce-reduce conflict**.

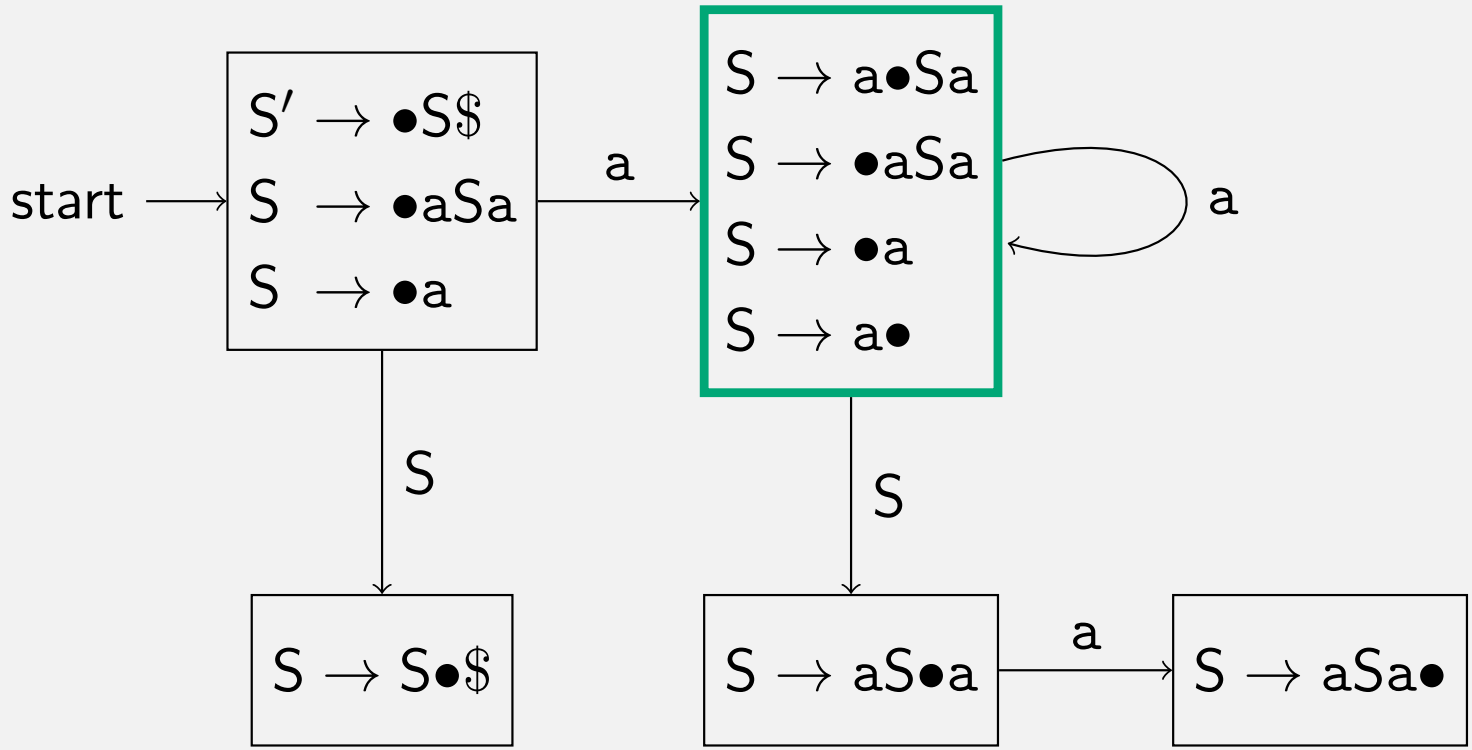
Example of a shift-reduce conflict

Consider the following grammar:

$$
\begin{aligned}
S' &\to S\$ \\
S &\to aSa \mid a
\end{aligned}
$$

Example of a shift-reduce conflict – contd.

The LR(0) automaton looks as follows:



The marked state is both a shift and a reduce state.

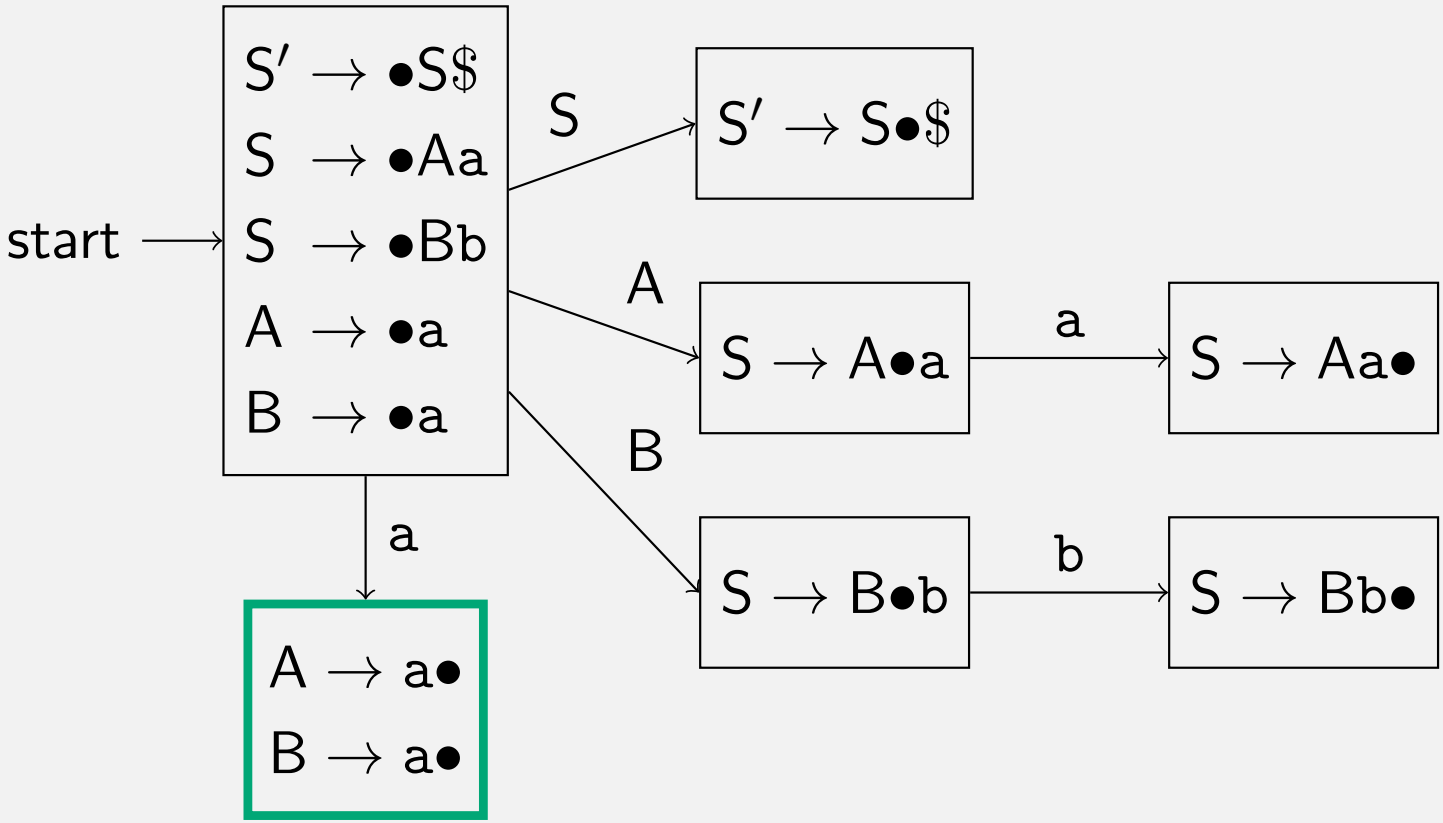
Example of a reduce-reduce conflict

Consider the following grammar:

$$
\begin{aligned}
S' &\to S\$ \\
S &\to Aa \mid Bb \\
A &\to a \\
B &\to a
\end{aligned}
$$

Example of a reduce-reduce conflict – contd.

The LR(0) automaton looks as follows:



The marked state contains multiple reducible productions.

Next lecture

More about conflicts