LR Parsing

Traditional Parsing Algorithms

Eduardo Souza, Guido Wachsmuth, Eelco Visser

Recap: Traditional Parsing Algorithms

lessons learned

How can we parse context-free languages effectively?

- predictive parsing

Which grammar classes are supported by these algorithms?

- LL(k) grammars, LL(k) languages

How can we generate compiler tools from that?

- implement automaton
- generate parse tables

What are other techniques for implementing top-down parsers?

- Parser Combinators
- PEGs
- ALL(*)

Overview

today's lecture

Overview
today's lecture

efficient parsing algorithms

- LR parsing
- LR parse table generation
- SLR & LALR parse tables
- Generalized LR parsing
- Scannerless Generalized LR parsing

I

LR Parsing

LR parsing
idea

problems with LL parsing

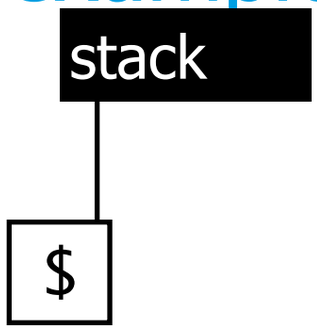
- predicting right rule
- left recursion

LR parsing

- see whole right-hand side of a rule
- look ahead
- shift or reduce

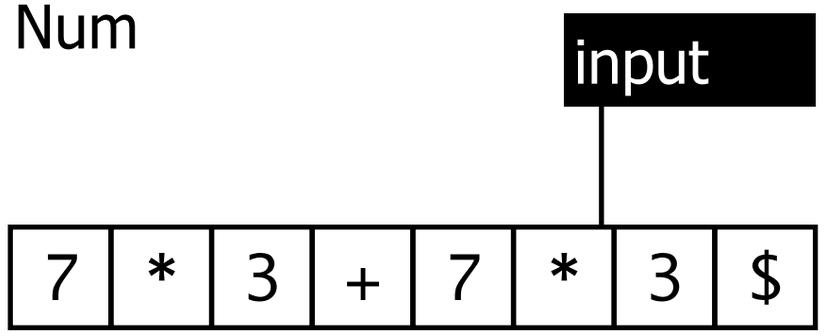
LR parsing

example



S → E $
E → E * E
E → E + E
E → Num

grammar



LR parsing

example

S → E $
E → E * E
E → E + E
E → Num

$

7	*	3	+	7	*	3	$

LR parsing

example

S → E $
E → E * E
E → E + E
E → Num

$	7

*	3	+	7	*	3	$

LR parsing

example

S → E $
E → E * E
E → E + E
E → Num

$	7
$	E

*	3	+	7	*	3	$
*	3	+	7	*	3	$

LR parsing

example

S → E $
E → E * E
E → E + E
E → Num

$	7

$	E	*

*	3	+	7	*	3	$

3	+	7	*	3	$

LR parsing

example

S → E $
E → E * E
E → E + E
E → Num

$	7

$	E	*	3

*	3	+	7	*	3	$

+	7	*	3	$

LR parsing

example

S → E $
E → E * E
E → E + E
E → Num

$	7

$	E	*	3

$	E	*	E

*	3	+	7	*	3	$

+	7	*	3	$

+	7	*	3	$

LR parsing

example

S → E $
E → E * E
E → E + E
E → Num

$	7

$	E	*	3

$	E	*	E

$	E

*	3	+	7	*	3	$

+	7	*	3	$

+	7	*	3	$

+	7	*	3	$

LR parsing

example

S → E $
E → E * E
E → E + E
E → Num

$	7

$	E	*	3

$	E	*	E

$	E	+

*	3	+	7	*	3	$

+	7	*	3	$

+	7	*	3	$

7	*	3	$

LR parsing

example

S → E $
E → E * E
E → E + E
E → Num

$	7

$	E	*	3

$	E	*	E

$	E	+	7

*	3	+	7	*	3	$

+	7	*	3	$

+	7	*	3	$

*	3	$

LR parsing

example

S → E $
E → E * E
E → E + E
E → Num

$	7

$	E	*	3

$	E	*	E

$	E	+	7

$	E	+	E

*	3	+	7	*	3	$

+	7	*	3	$

+	7	*	3	$

*	3	$

*	3	$

LR parsing

example

S → E $
E → E * E
E → E + E
E → Num

$	7

$	E	*	3

$	E	*	E

$	E	+	7

$	E	+	E	*

*	3	+	7	*	3	$

+	7	*	3	$

+	7	*	3	$

*	3	$

3	$

LR parsing

example

S → E $
E → E * E
E → E + E
E → Num

$	7

$	E	*	3

$	E	*	E

$	E	+	7

$	E	+	E	*	3

*	3	+	7	*	3	$

+	7	*	3	$

+	7	*	3	$

*	3	$

$

LR parsing

example

S → E $
E → E * E
E → E + E
E → Num

$	7

$	E	*	3

$	E	*	E

$	E	+	7

$	E	+	E	*	3

$	E	+	E	*	E

*	3	+	7	*	3	$

+	7	*	3	$

+	7	*	3	$

*	3	$

$

$

LR parsing

example

S → E $
E → E * E
E → E + E
E → Num

$	7

$	E	*	3

$	E	*	E

$	E	+	7

$	E	+	E	*	3

$	E	+	E	*	E

$	E	+	E

*	3	+	7	*	3	$

+	7	*	3	$

+	7	*	3	$

*	3	$

$

$

$

LR parsing

example

S → E $
E → E * E
E → E + E
E → Num

$	7

$	E	*	3

$	E	*	E

$	E	+	7

$	E	+	E	*	3

$	E	+	E	*	E

$	E	+	E

$	E

*	3	+	7	*	3	$

+	7	*	3	$

+	7	*	3	$

*	3	$

$

$

$

$

LR parsing

example

S → E $
E → E * E
E → E + E
E → Num

$	7

$	E	*	3

$	E	*	E

$	E	+	7

$	E	+	E	*	3

$	E	+	E	*	E

$	E	+	E

$	E	$

*	3	+	7	*	3	$

+	7	*	3	$

+	7	*	3	$

*	3	$

$

$

$

LR parsing

example

S → E $
E → E * E
E → E + E
E → Num

$	7

$	E	*	3

$	E	*	E

$	E	+	7

$	E	+	E	*	3

$	E	+	E	*	E

$	E	+	E

$	E	$

$	S

*	3	+	7	*	3	$

+	7	*	3	$

+	7	*	3	$

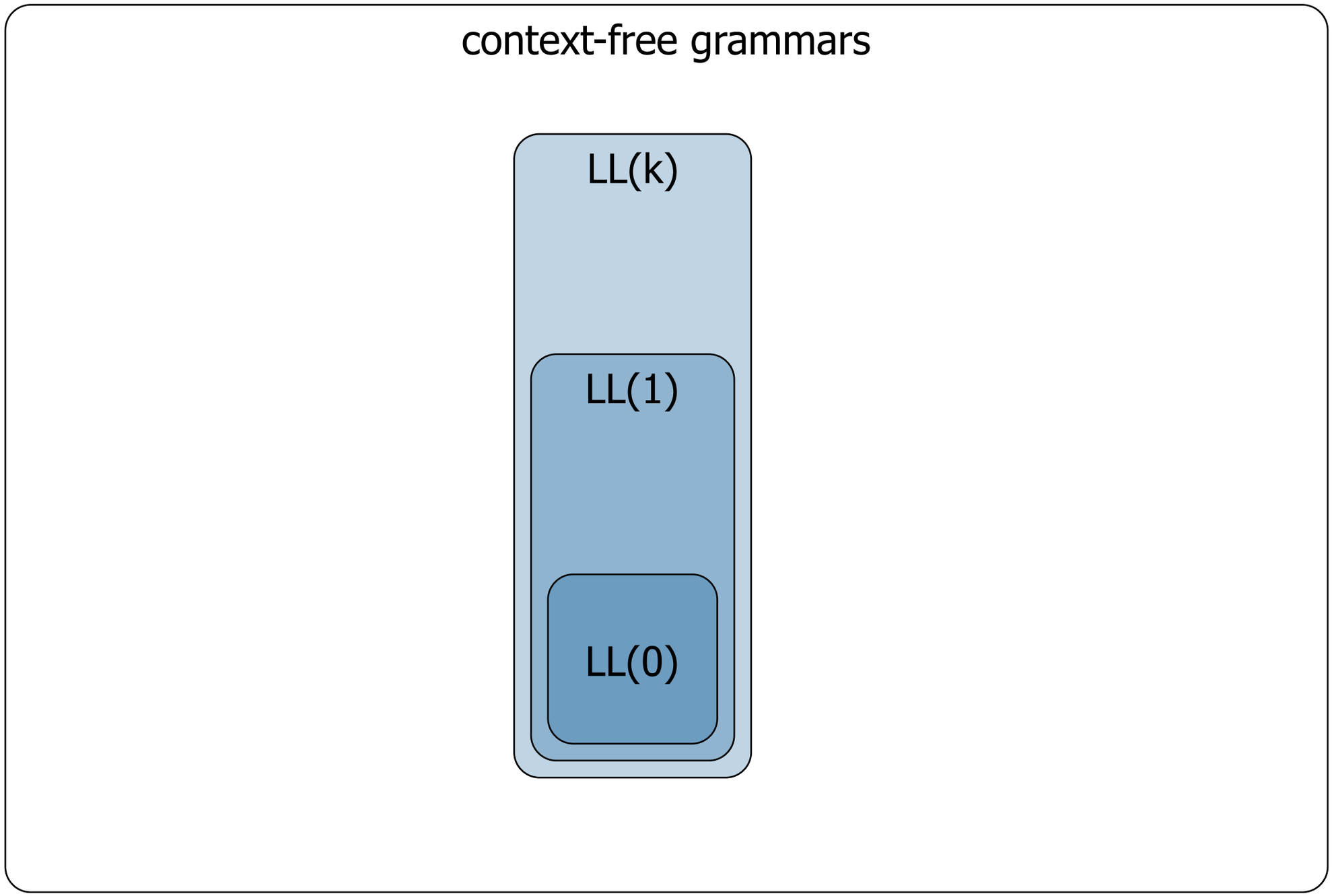
*	3	$

$

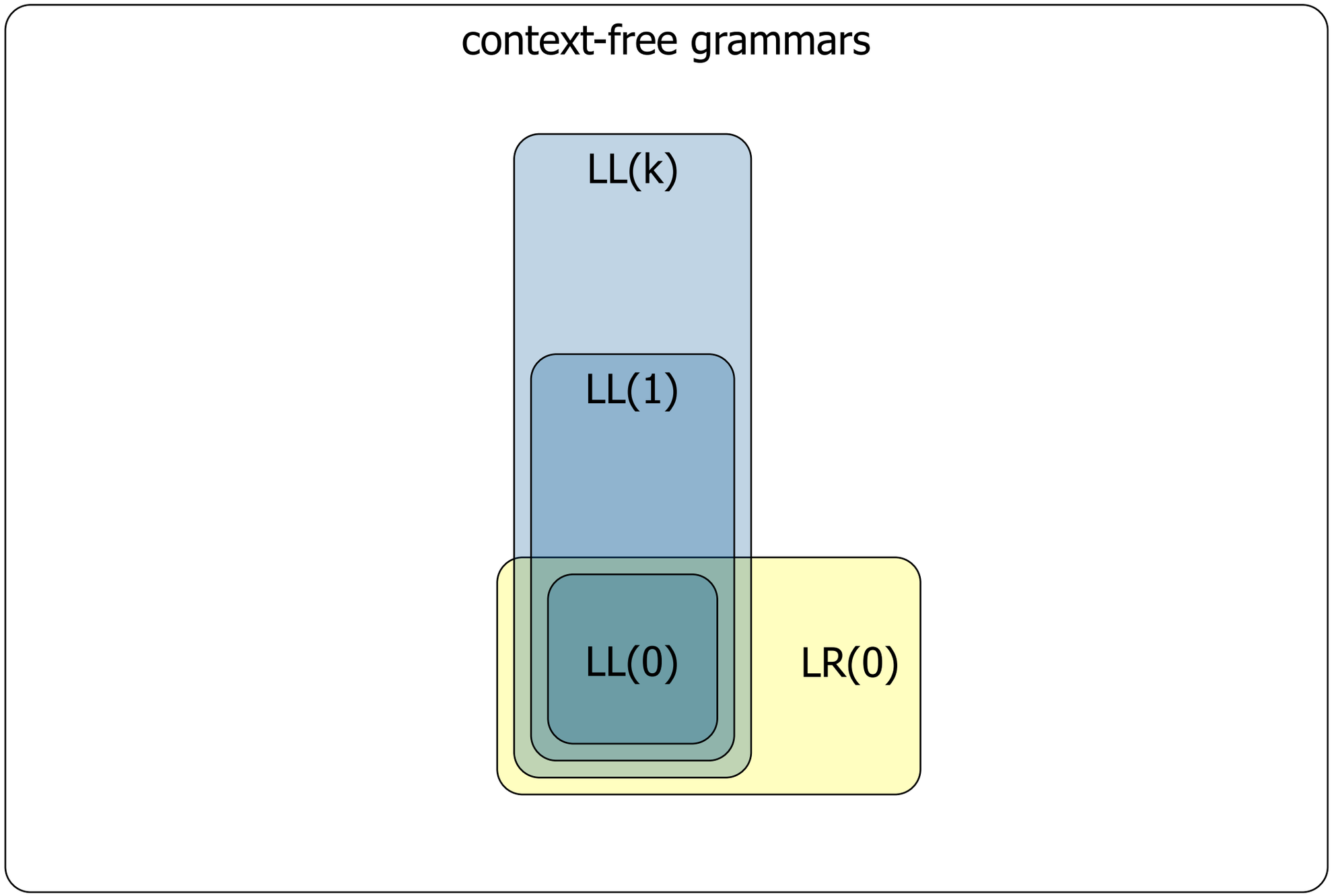
$

$

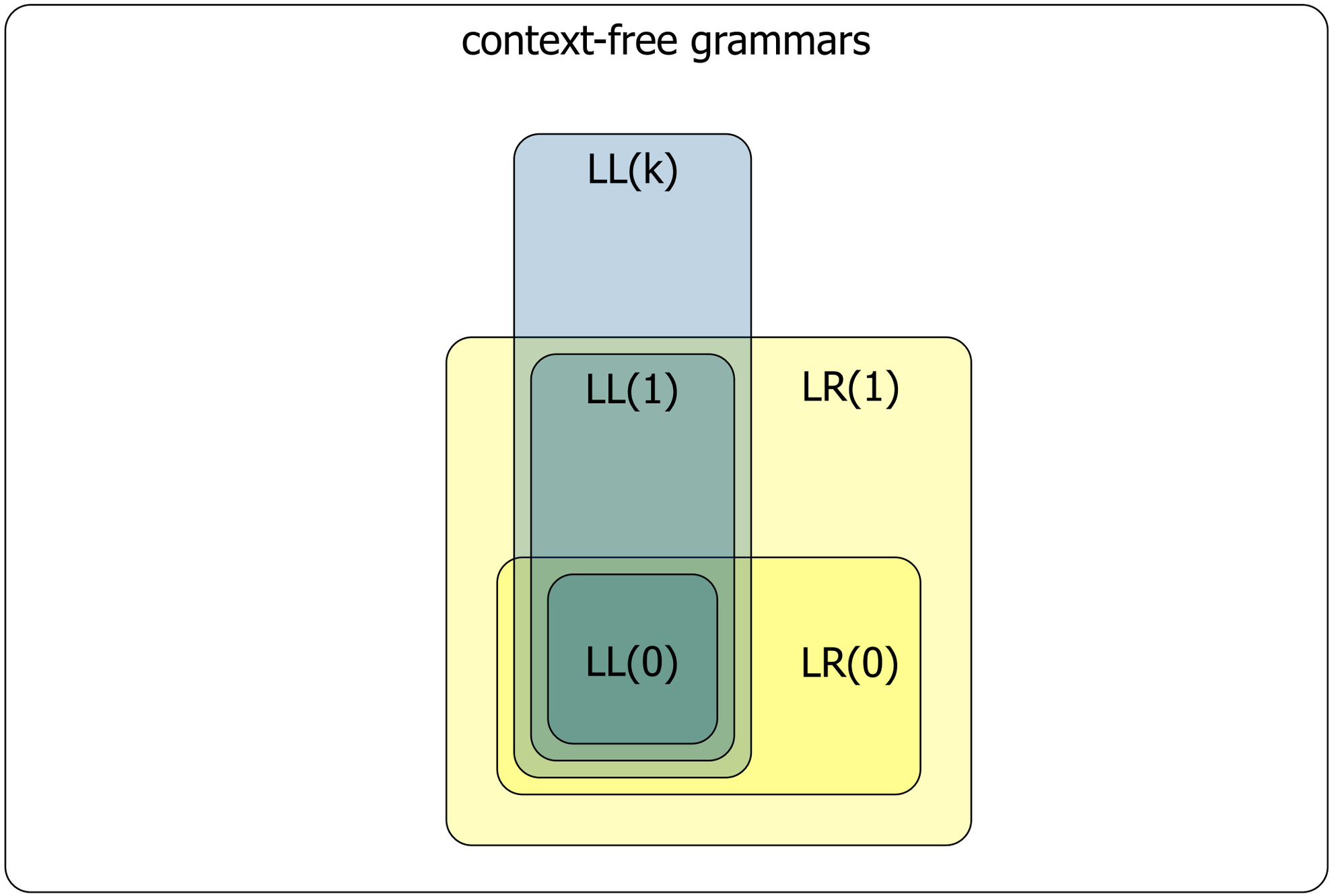
Grammar classes



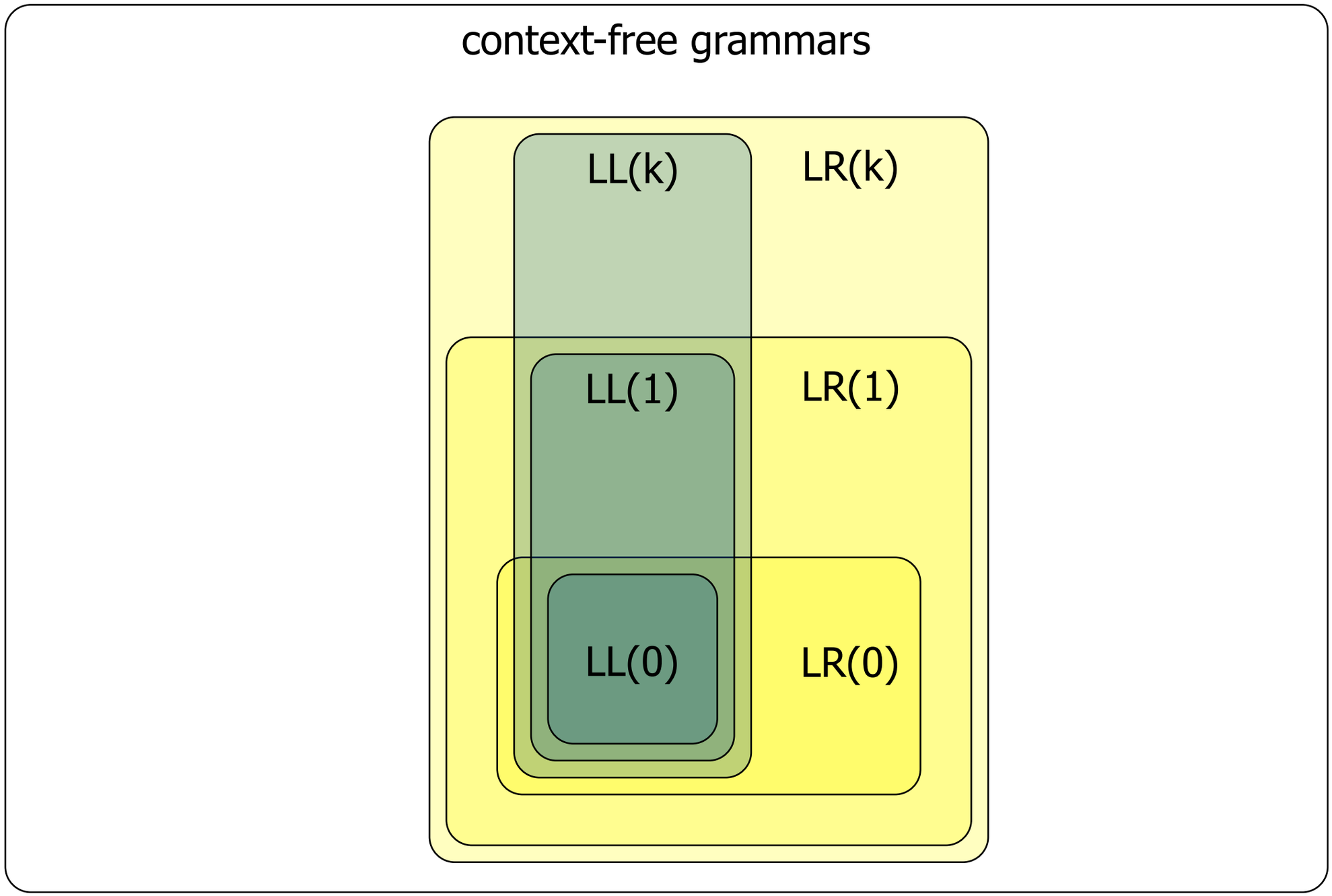
Grammar classes



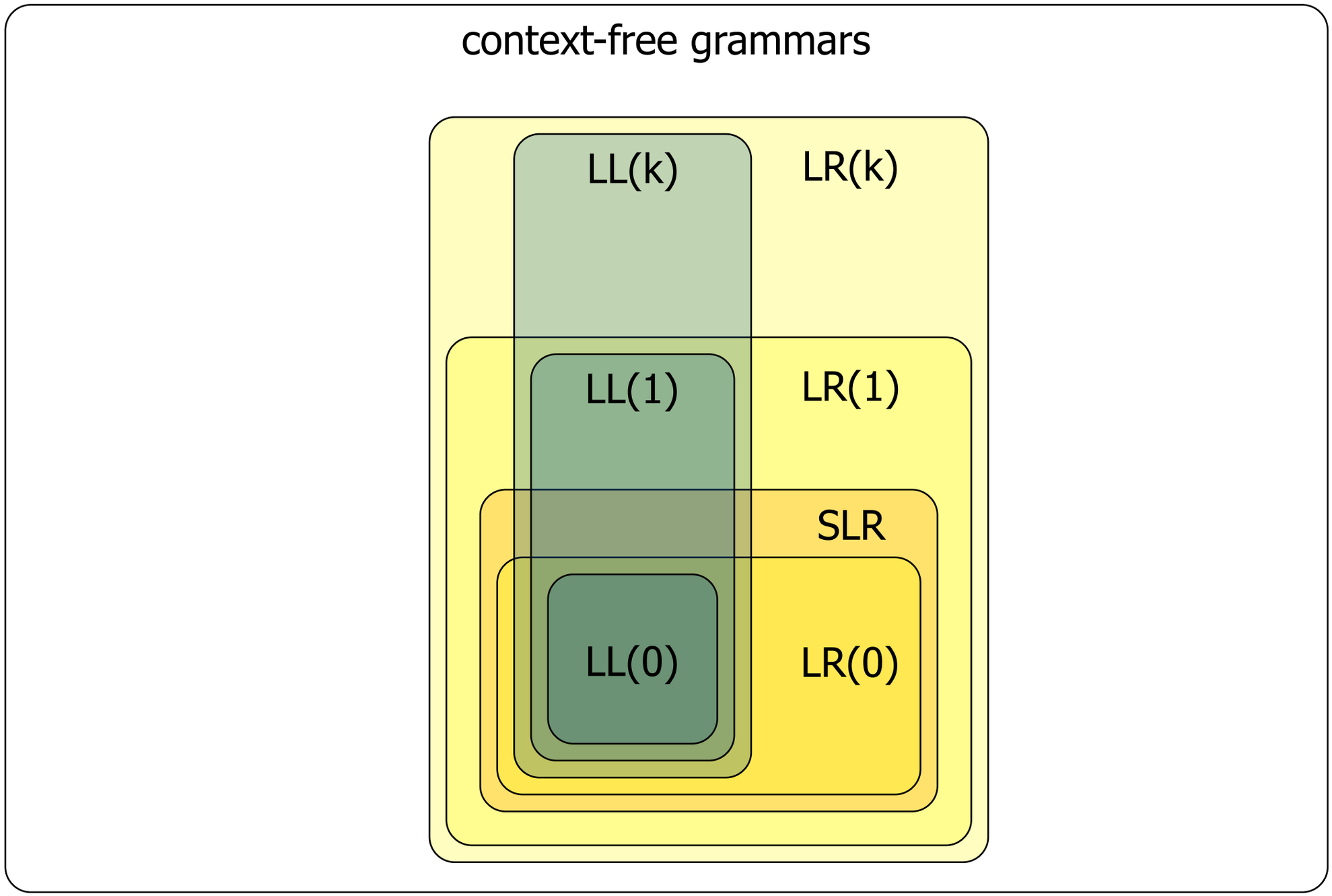
Grammar classes



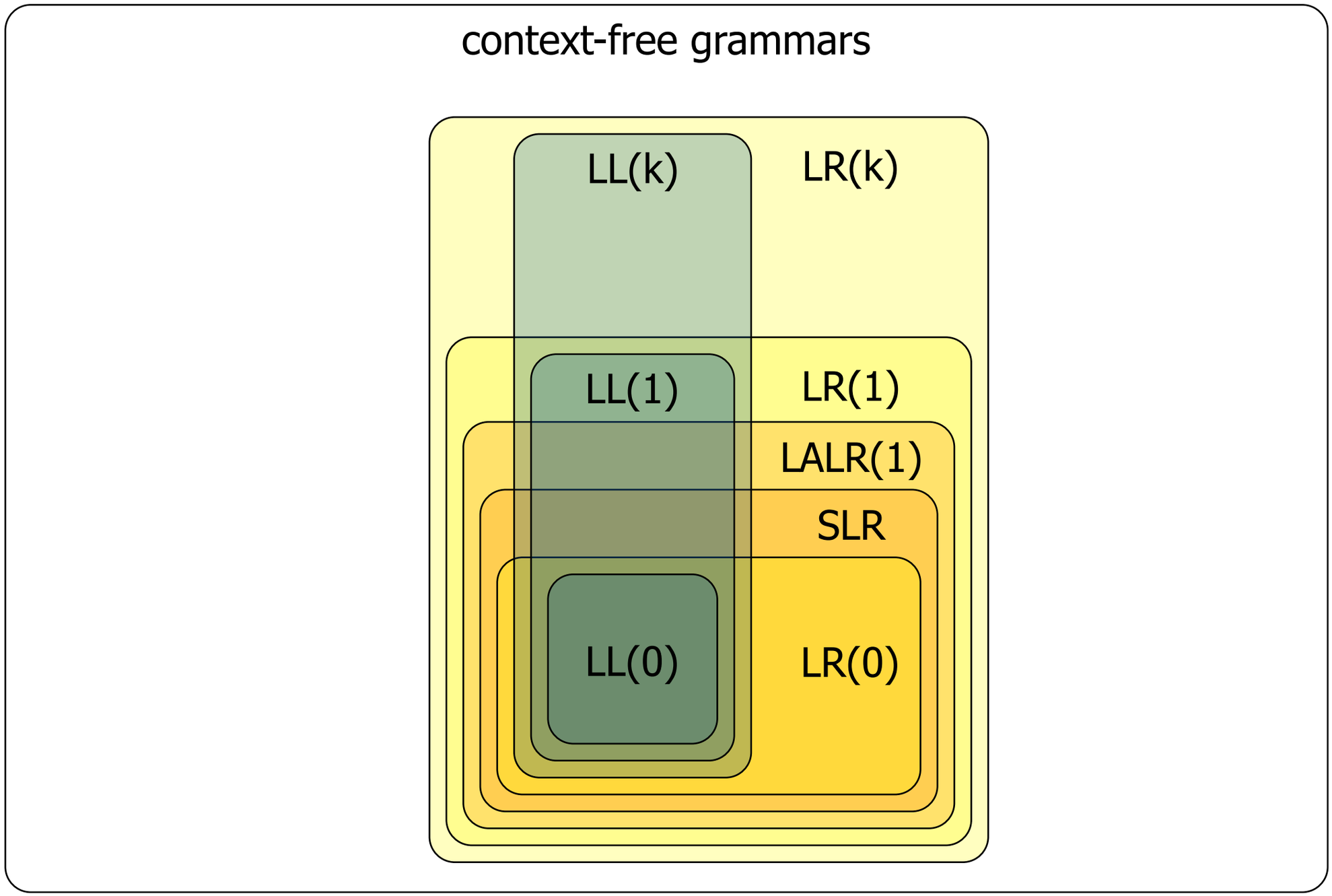
Grammar classes



Grammar classes



Grammar classes



II

LR Parse Tables

LR parsing
parse table

rows

- states of a DFA

columns

- topmost stack symbol
- Σ, N

entries

- reduce, rule number
- shift, goto state
- goto state
- accept state

	T_1	...	N_1	...
1	s 3			
2			g 5	
3	r 1			
4	r 2	a		
5				
6			g 1	
7	s 1			
8				
...				

LR(0) parse tables

items, closure & goto

$S \rightarrow x$
$S \rightarrow (L)$
$L \rightarrow S$
$L \rightarrow L , S$

LR(0) parse tables

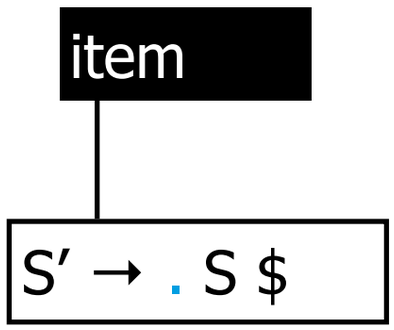
items, closure & goto

S′ → . S $

S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

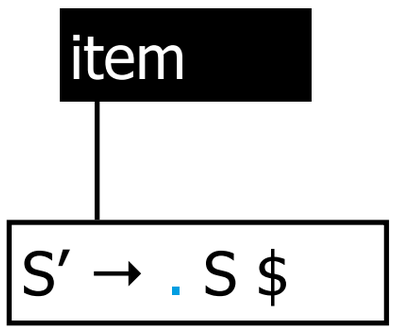
items, closure & goto



S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

items, closure & goto



closure

- for every item A → α . X β
- for every rule X → γ
- add item X → . γ

S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

items, closure & goto

S′ → . S $
S → . x
S → . (L)

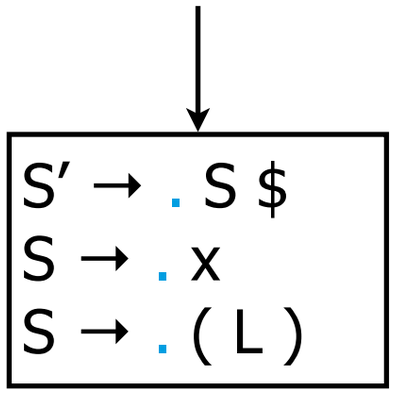
closure

- for every item A → α . X β
- for every rule X → γ
- add item X → . γ

S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

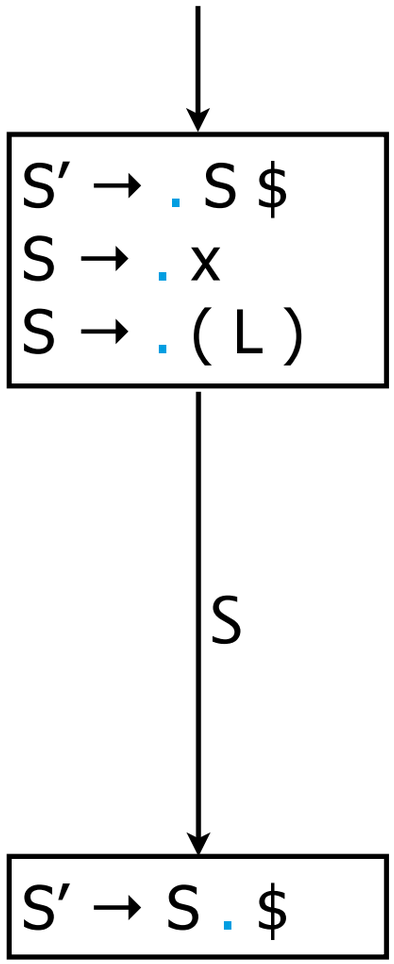
items, closure & goto



S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

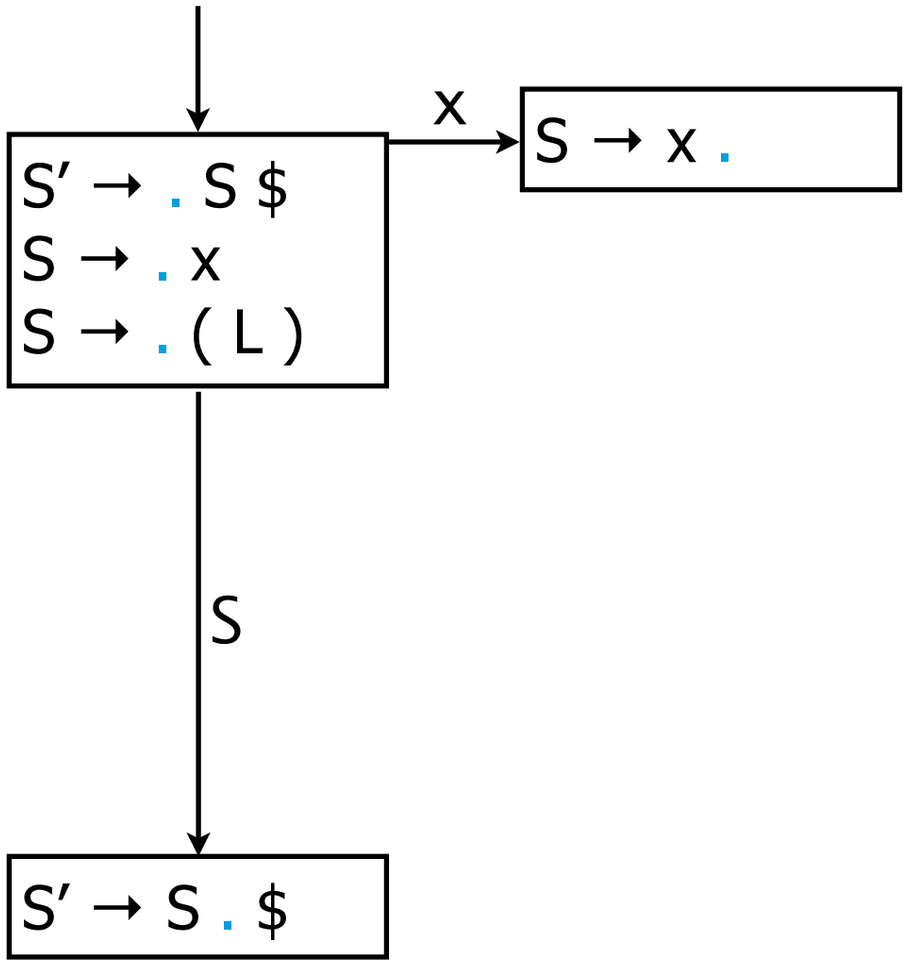
items, closure & goto



S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

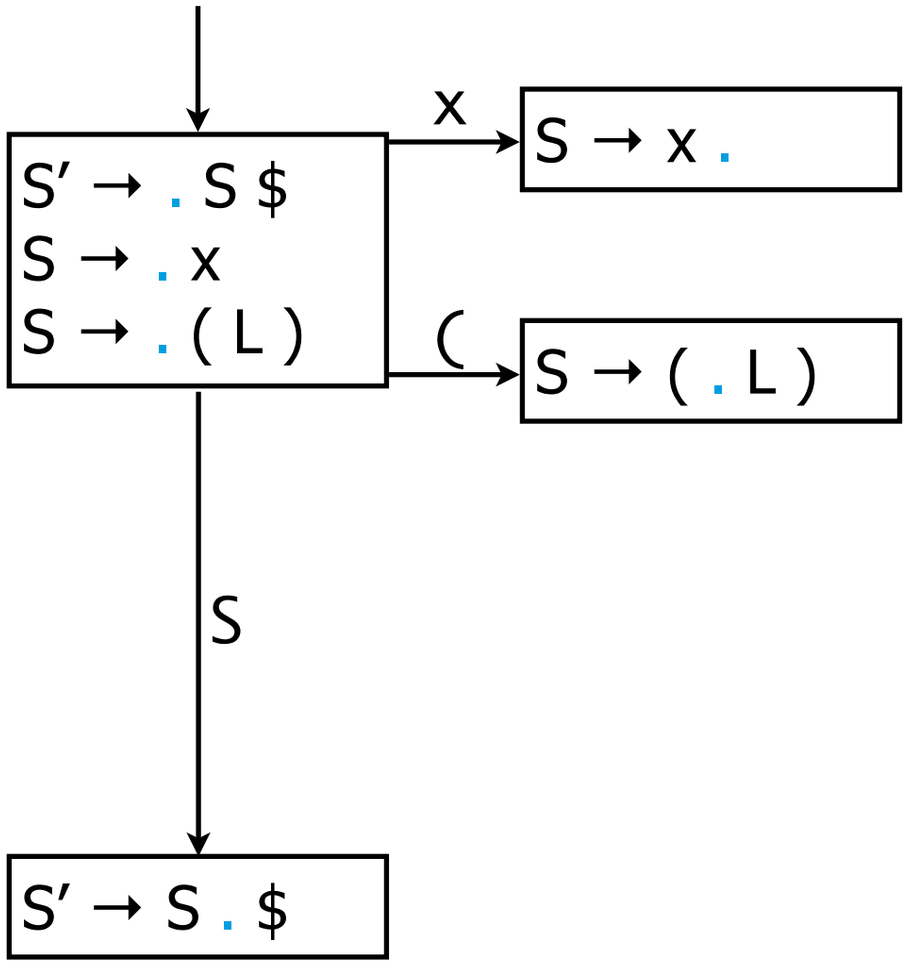
items, closure & goto



S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

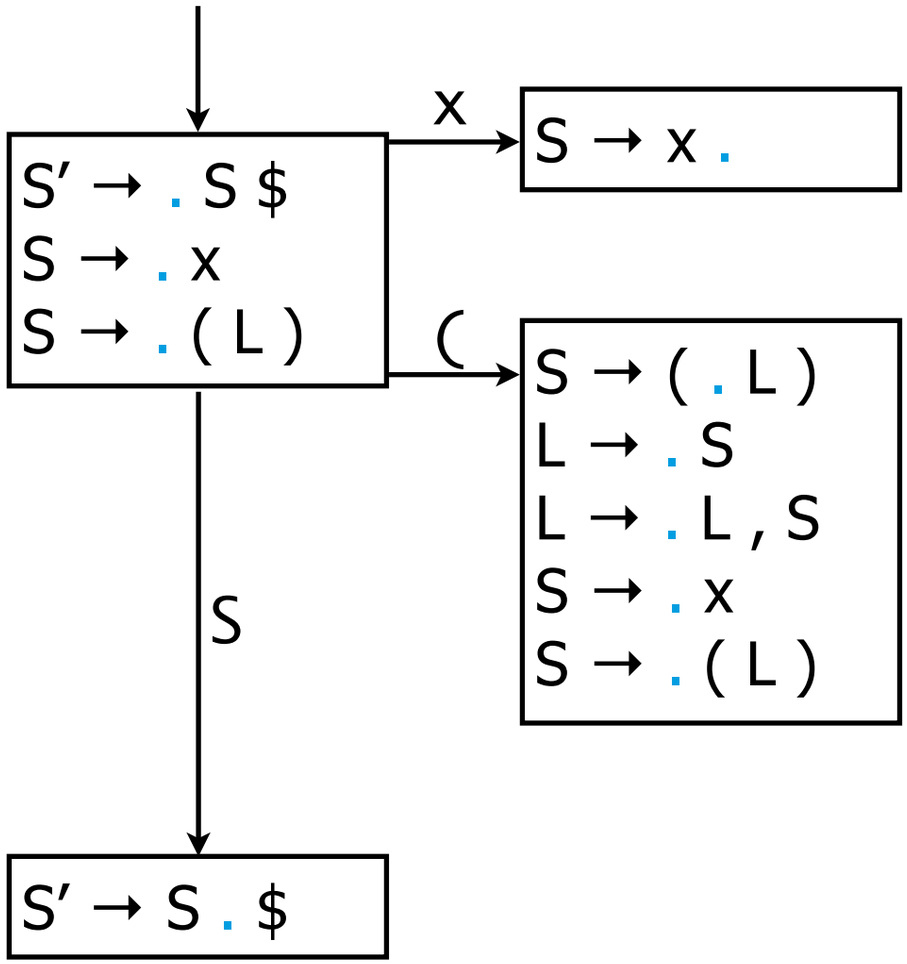
items, closure & goto



S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

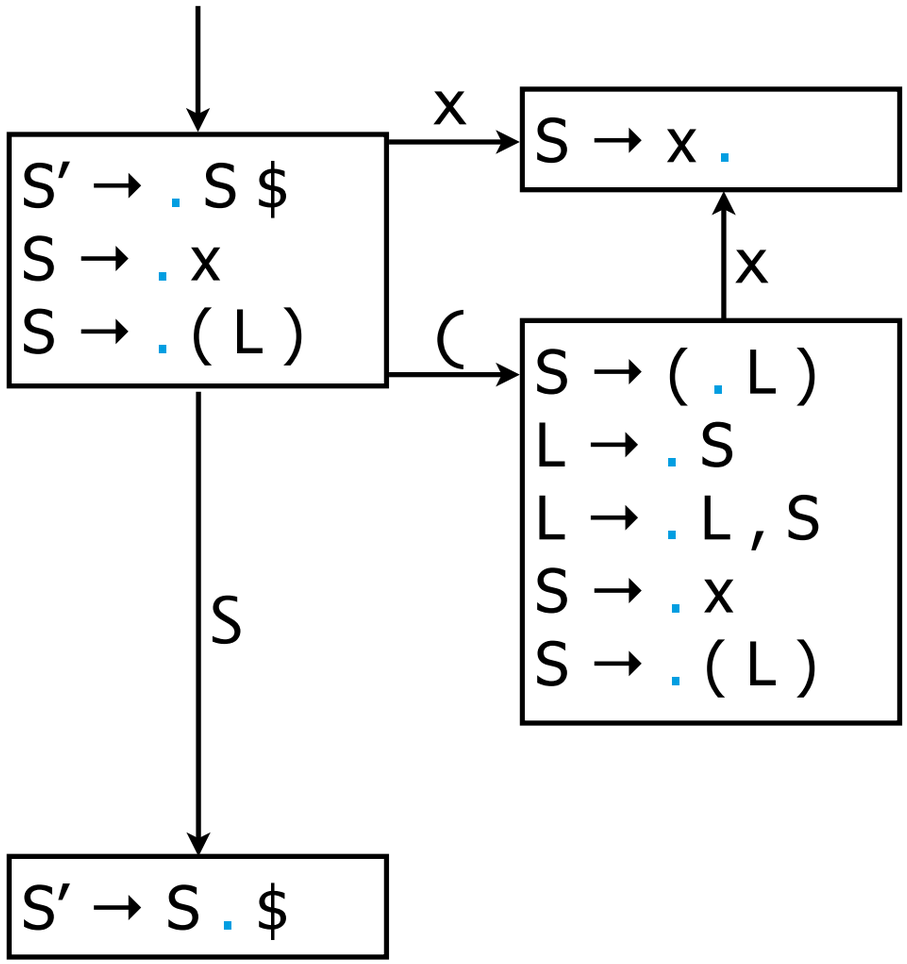
items, closure & goto



S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

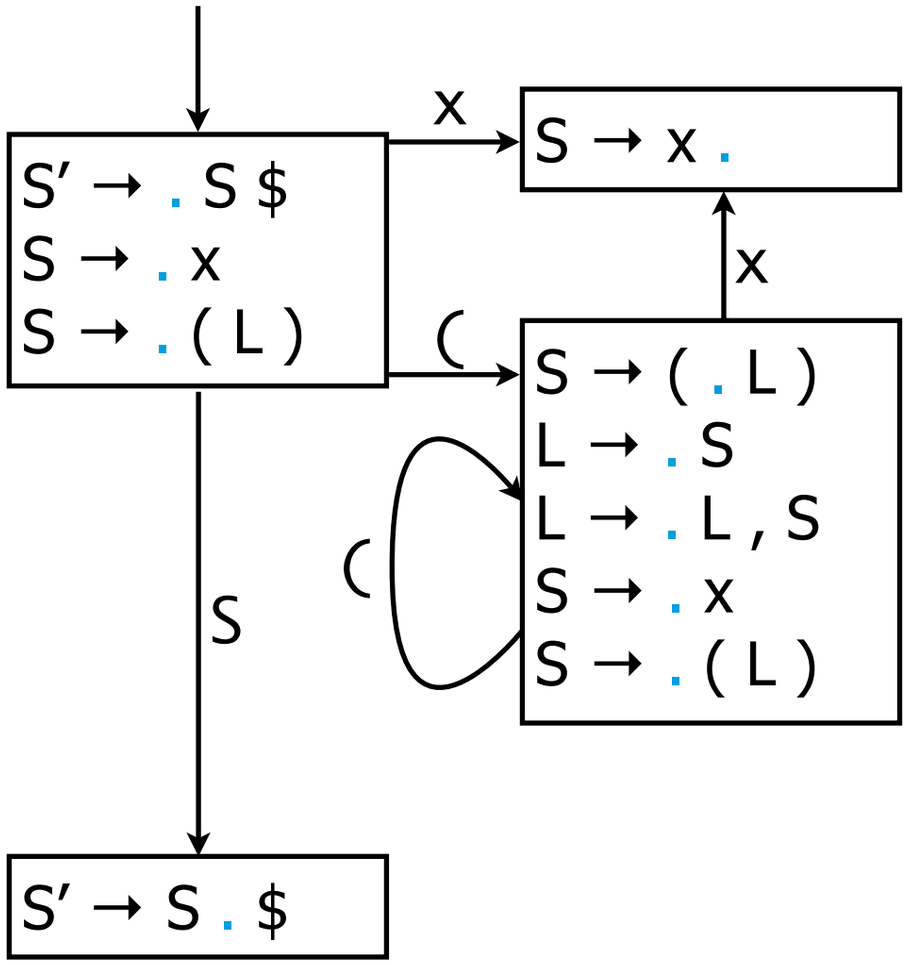
items, closure & goto



S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

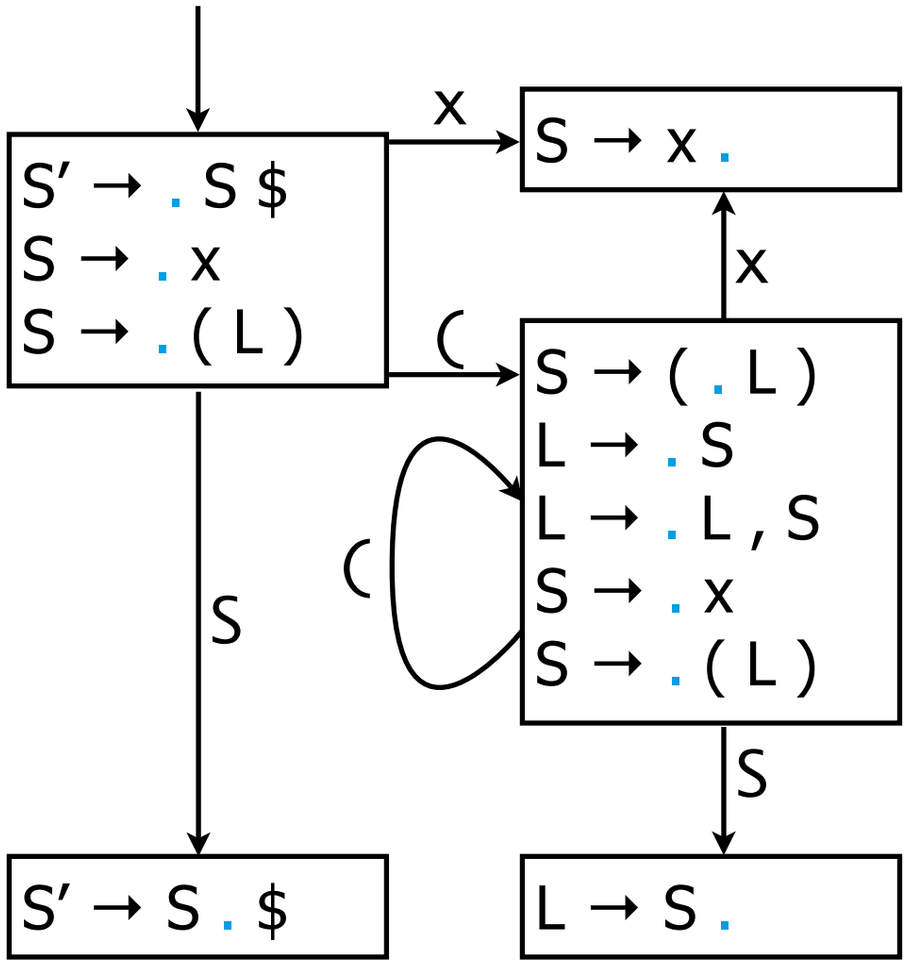
items, closure & goto



S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

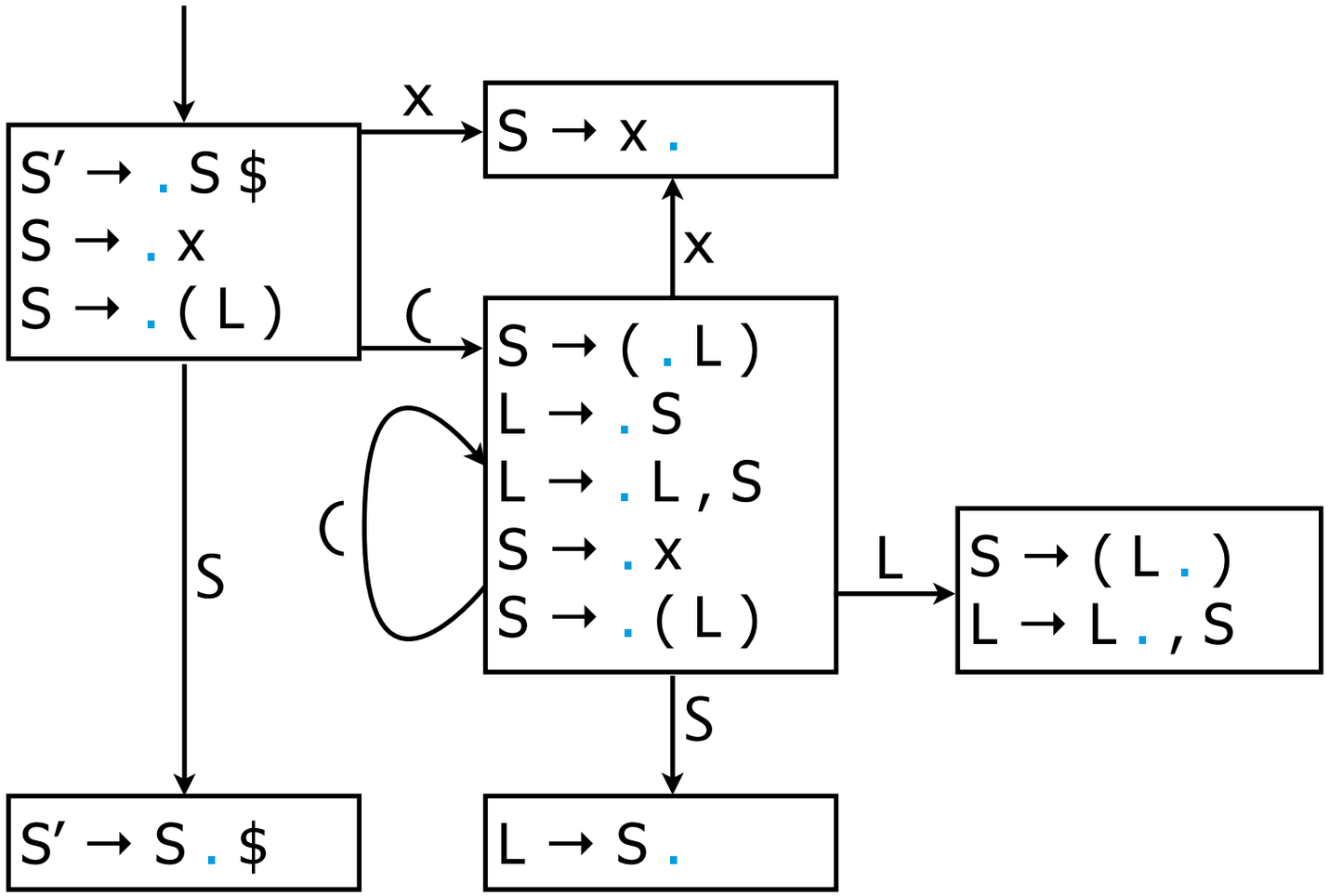
items, closure & goto



S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

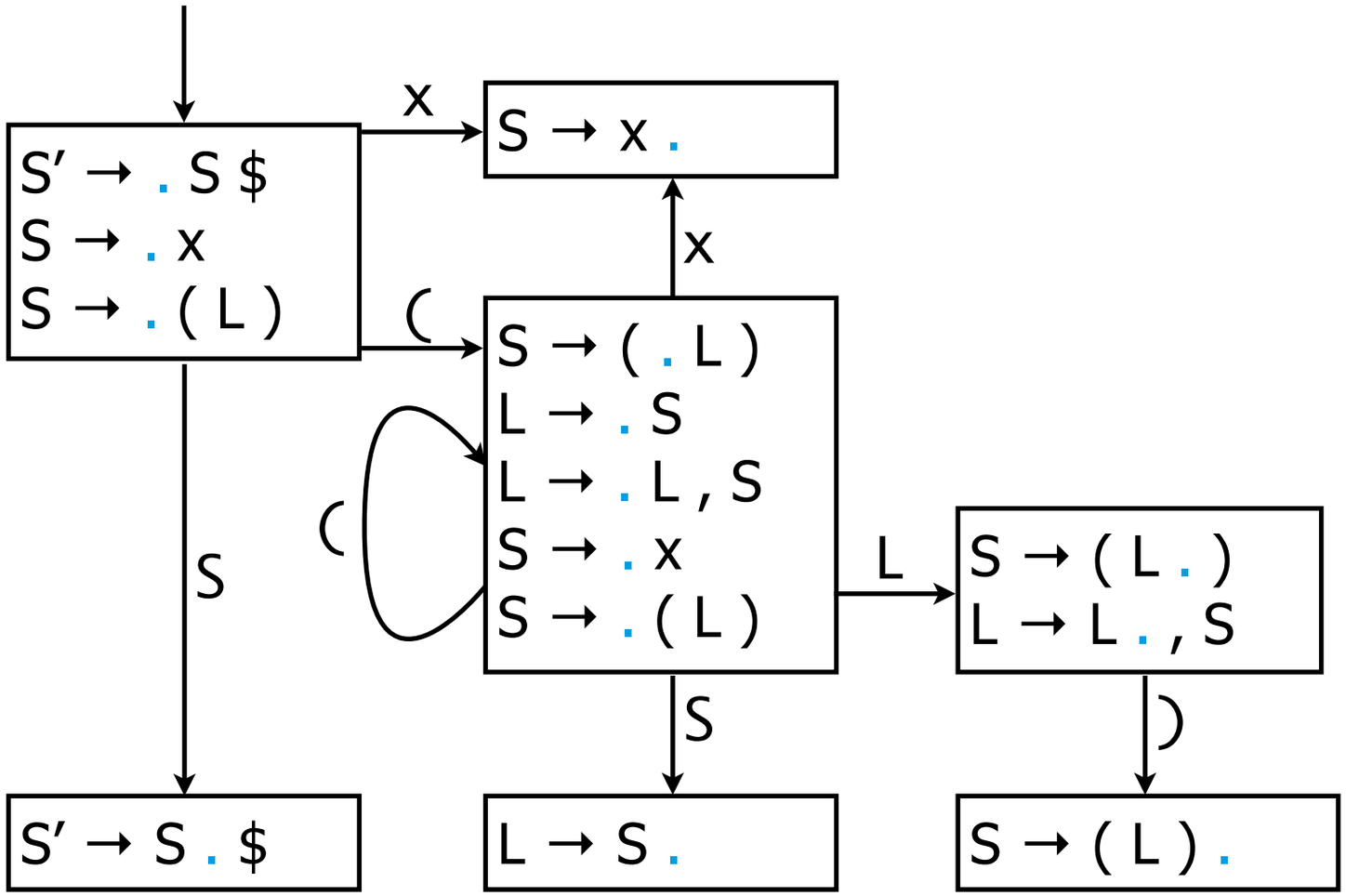
items, closure & goto



S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

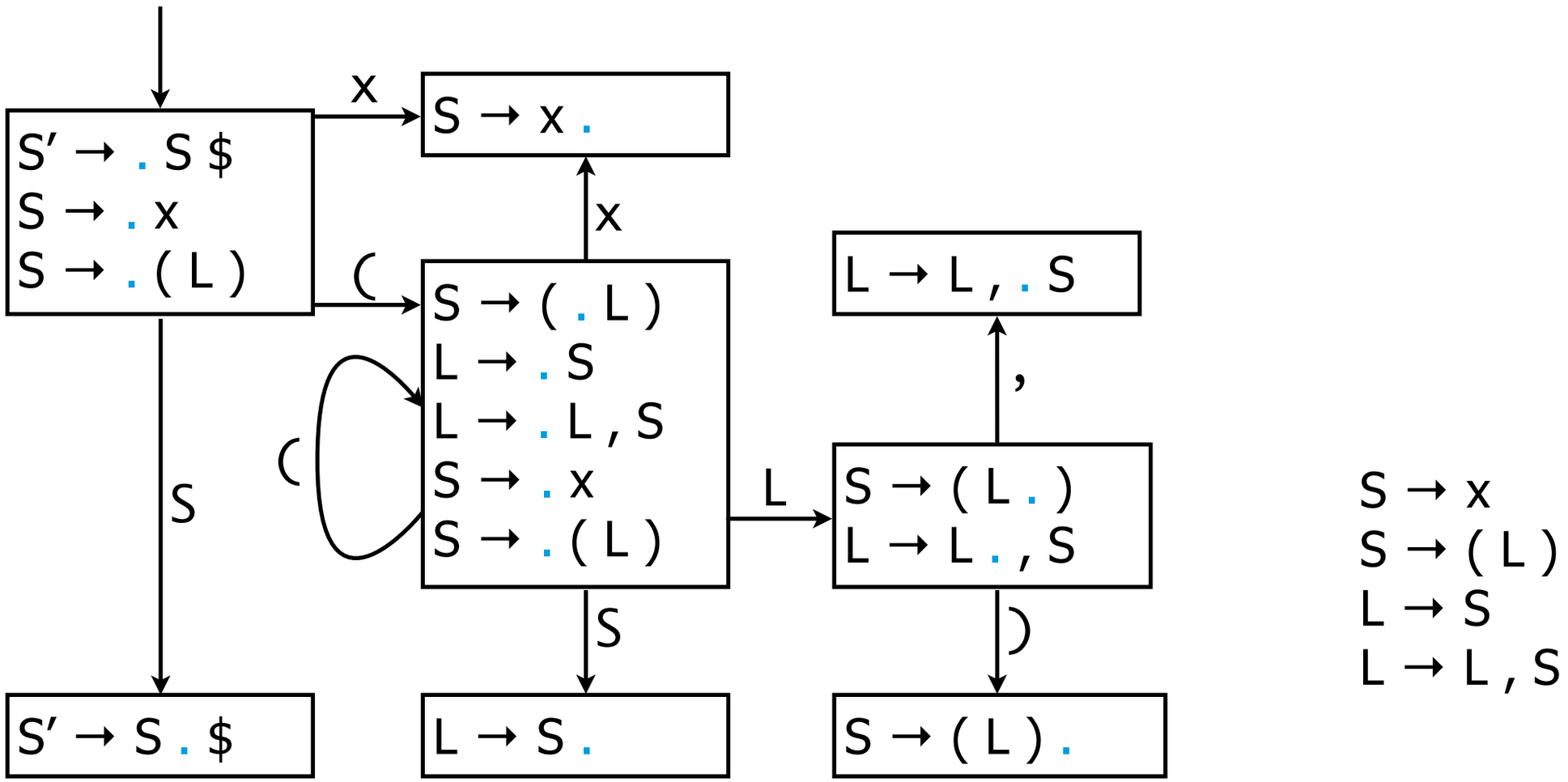
items, closure & goto



S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

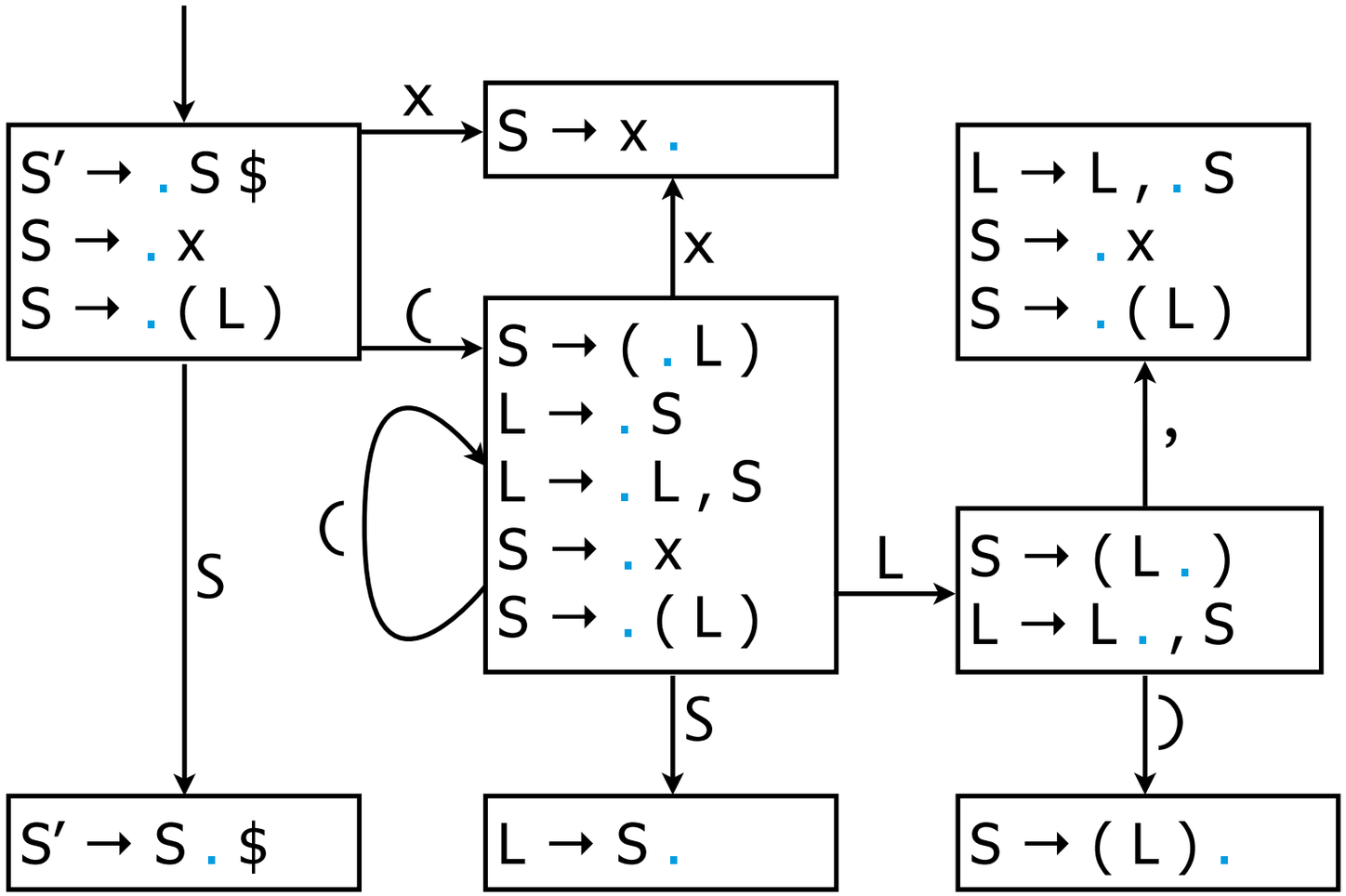
items, closure & goto



S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

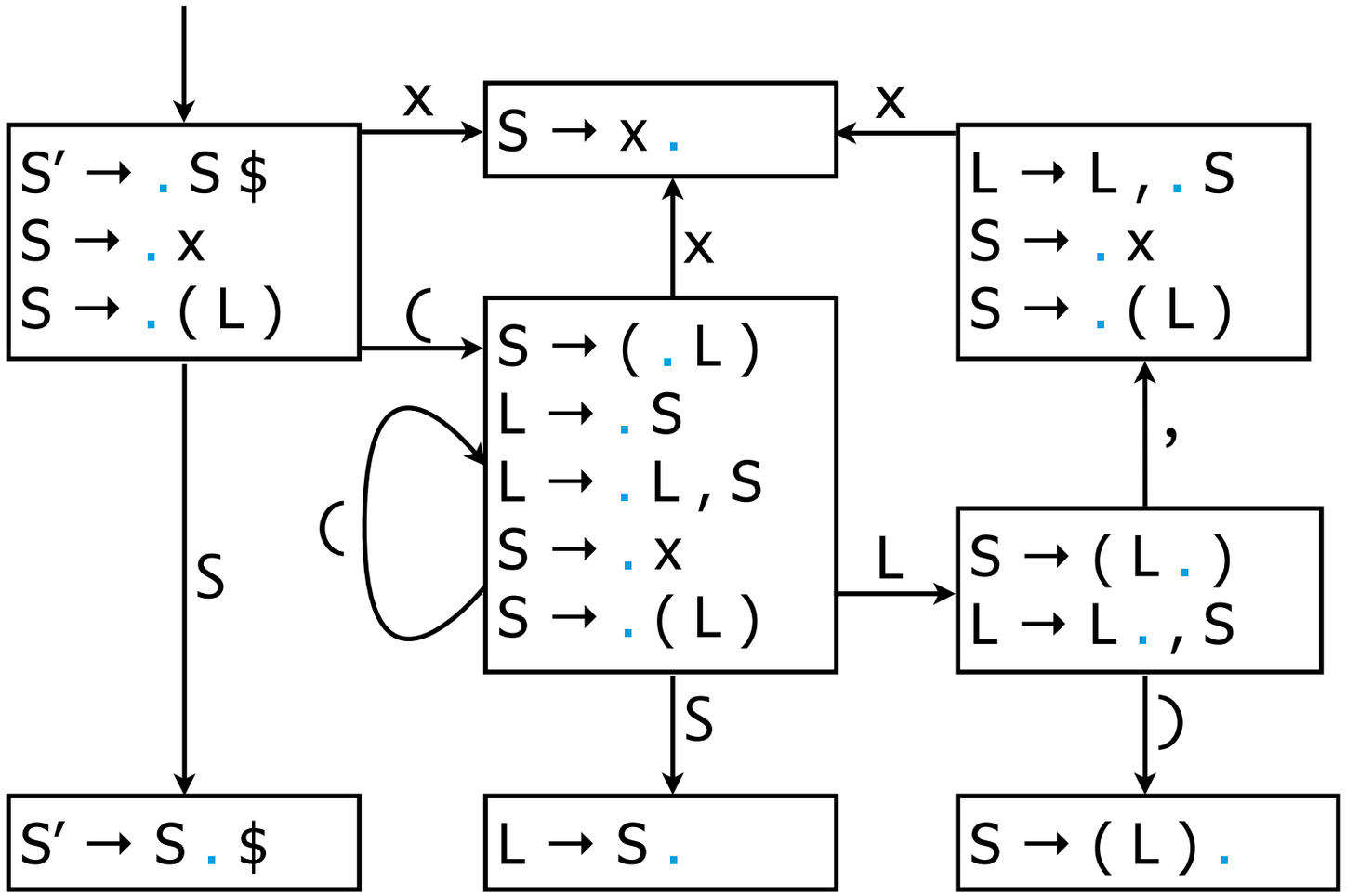
items, closure & goto



S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

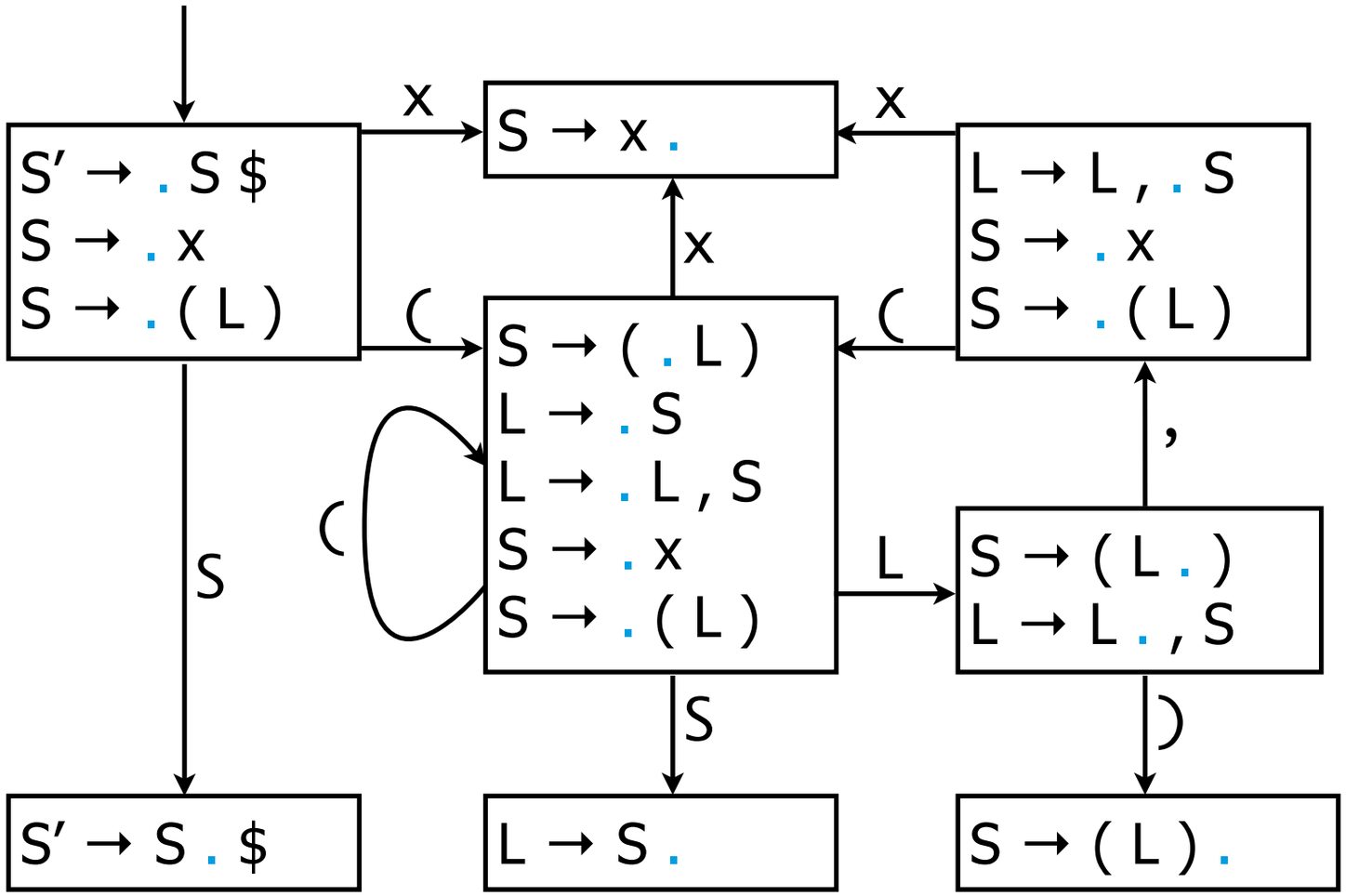
items, closure & goto



S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

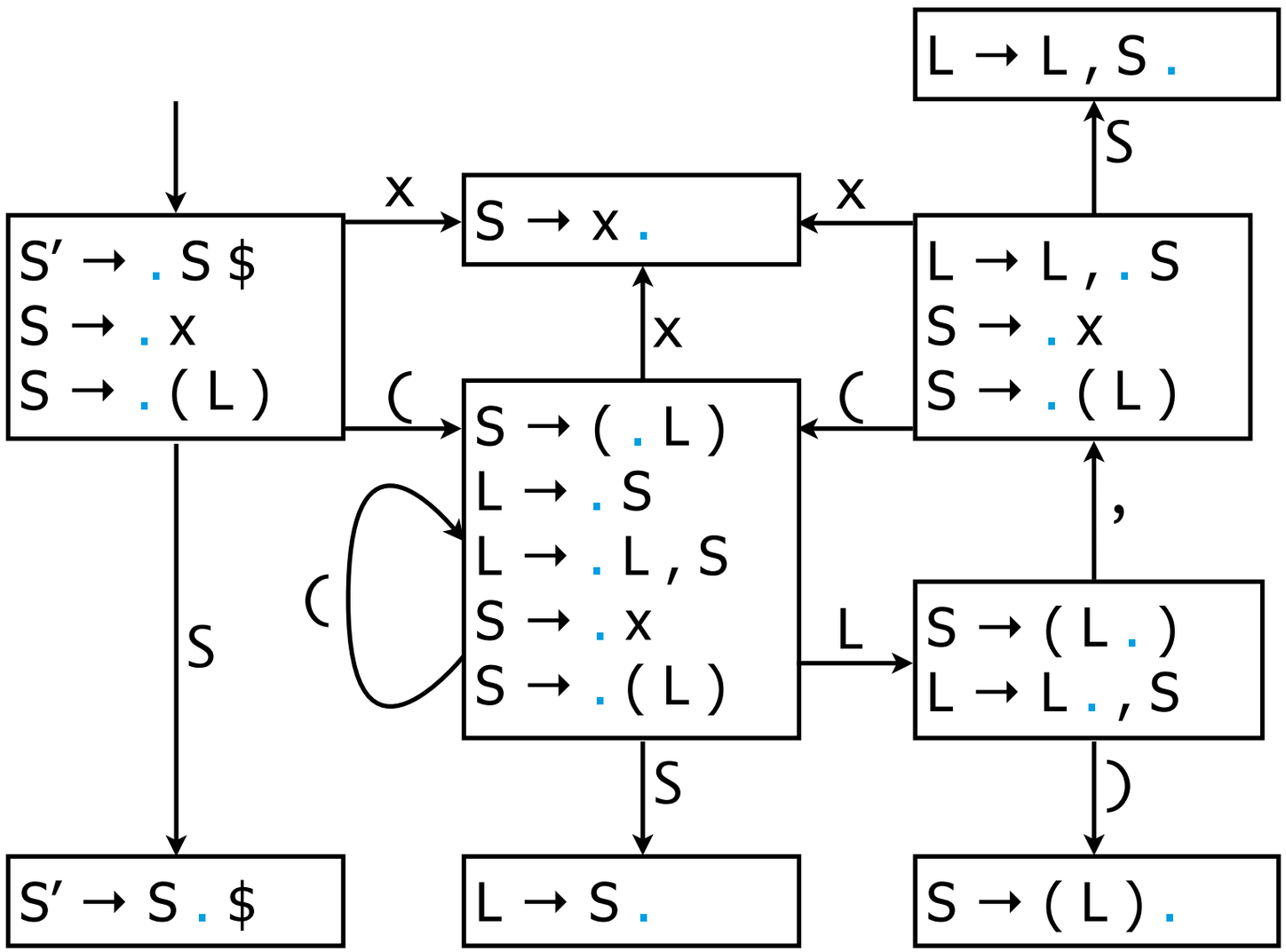
items, closure & goto



S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

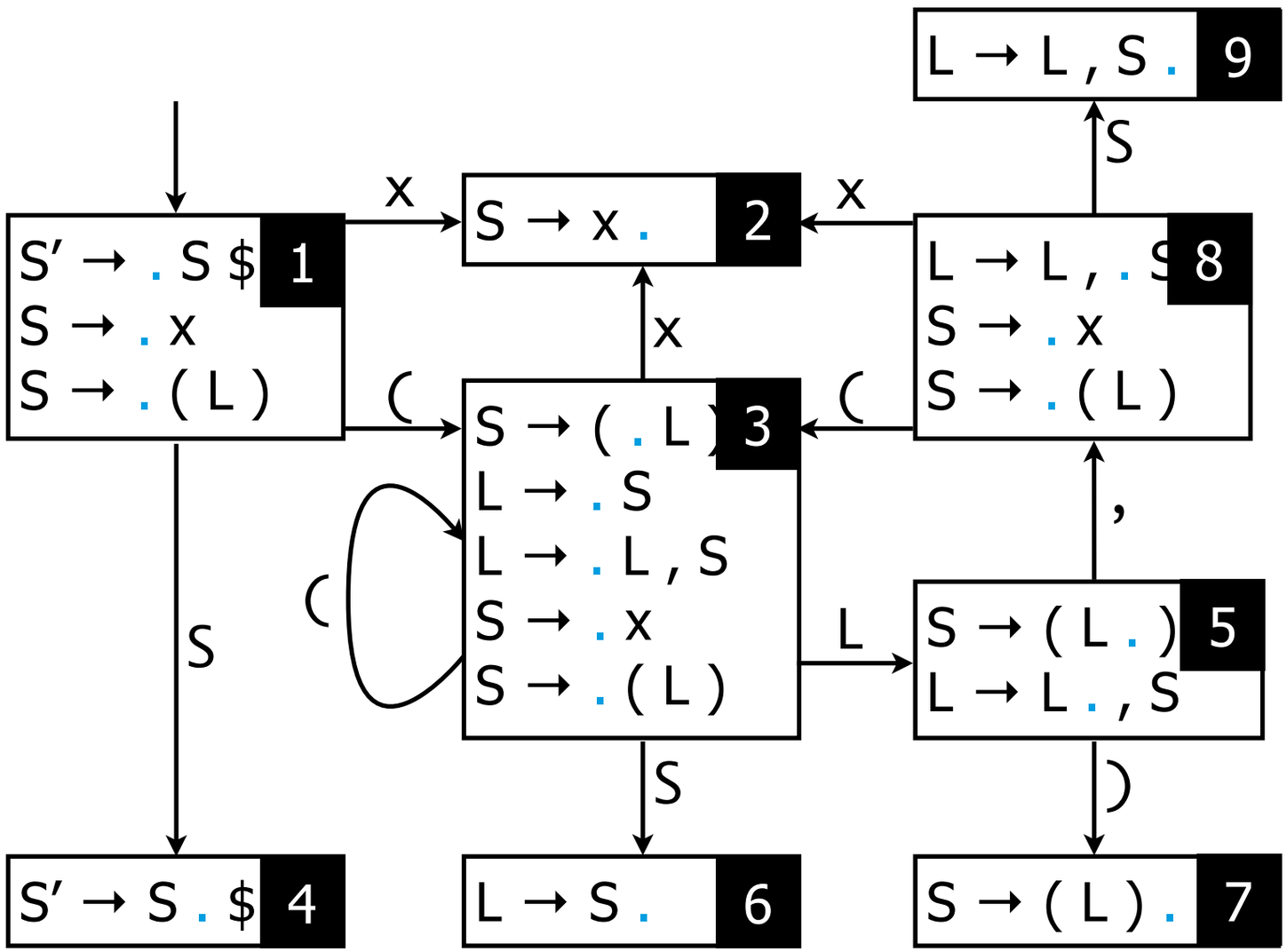
items, closure & goto



S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

items, closure & goto



S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

result

	(	)	x	,	$	S	L
1	s 3		s 2			g 4	
2	r 1	r 1	r 1	r 1	r 1		
3	s 3		s 2			g 6	g 5
4					a		
5		s 7		s 8			
6	r 3	r 3	r 3	r 3	r 3		
7	r 2	r 2	r 2	r 2	r 2		
8	s 3		s 2			g 9	
9	r 4	r 4	r 4	r 4	r 4		

S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

result

(	x	,	x	)	$

	(	)	x	,	$	S	L
1	s 3		s 2			g 4	
2	r 1	r 1	r 1	r 1	r 1		
3	s 3		s 2			g 6	g 5
4					a		
5		s 7		s 8			
6	r 3	r 3	r 3	r 3	r 3		
7	r 2	r 2	r 2	r 2	r 2		
8	s 3		s 2			g 9	
9	r 4	r 4	r 4	r 4	r 4		

1

S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

result

x	,	x	)	$

	(	)	x	,	$	S	L
1	s 3		s 2			g 4	
2	r 1	r 1	r 1	r 1	r 1		
3	s 3		s 2			g 6	g 5
4					a		
5		s 7		s 8			
6	r 3	r 3	r 3	r 3	r 3		
7	r 2	r 2	r 2	r 2	r 2		
8	s 3		s 2			g 9	
9	r 4	r 4	r 4	r 4	r 4		

3
1

S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

result

,	x	)	$

	(	)	x	,	$	S	L
1	s 3		s 2			g 4	
2	r 1	r 1	r 1	r 1	r 1		
3	s 3		s 2			g 6	g 5
4					a		
5		s 7		s 8			
6	r 3	r 3	r 3	r 3	r 3		
7	r 2	r 2	r 2	r 2	r 2		
8	s 3		s 2			g 9	
9	r 4	r 4	r 4	r 4	r 4		

2
3
1

S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

result

,	x	)	$

	(	)	x	,	$	S	L
1	s 3		s 2			g 4	
2	r 1	r 1	r 1	r 1	r 1		
3	s 3		s 2			g 6	g 5
4					a		
5		s 7		s 8			
6	r 3	r 3	r 3	r 3	r 3		
7	r 2	r 2	r 2	r 2	r 2		
8	s 3		s 2			g 9	
9	r 4	r 4	r 4	r 4	r 4		

3
1

S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

result

,	x	)	$

	(	)	x	,	$	S	L
1	s 3		s 2			g 4	
2	r 1	r 1	r 1	r 1	r 1		
3	s 3		s 2			g 6	g 5
4					a		
5		s 7		s 8			
6	r 3	r 3	r 3	r 3	r 3		
7	r 2	r 2	r 2	r 2	r 2		
8	s 3		s 2			g 9	
9	r 4	r 4	r 4	r 4	r 4		

6
3
1

S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

result

,	x	)	$

	(	)	x	,	$	S	L
1	s 3		s 2			g 4	
2	r 1	r 1	r 1	r 1	r 1		
3	s 3		s 2			g 6	g 5
4					a		
5		s 7		s 8			
6	r 3	r 3	r 3	r 3	r 3		
7	r 2	r 2	r 2	r 2	r 2		
8	s 3		s 2			g 9	
9	r 4	r 4	r 4	r 4	r 4		

3
1

S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

result

,	x	)	$

	(	)	x	,	$	S	L
1	s 3		s 2			g 4	
2	r 1	r 1	r 1	r 1	r 1		
3	s 3		s 2			g 6	g 5
4					a		
5		s 7		s 8			
6	r 3	r 3	r 3	r 3	r 3		
7	r 2	r 2	r 2	r 2	r 2		
8	s 3		s 2			g 9	
9	r 4	r 4	r 4	r 4	r 4		

5
3
1

S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

result

x	)	$

	(	)	x	,	$	S	L
1	s 3		s 2			g 4	
2	r 1	r 1	r 1	r 1	r 1		
3	s 3		s 2			g 6	g 5
4					a		
5		s 7		s 8			
6	r 3	r 3	r 3	r 3	r 3		
7	r 2	r 2	r 2	r 2	r 2		
8	s 3		s 2			g 9	
9	r 4	r 4	r 4	r 4	r 4		

8
5
3
1

S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

result

)	$

	(	)	x	,	$	S	L
1	s 3		s 2			g 4	
2	r 1	r 1	r 1	r 1	r 1		
3	s 3		s 2			g 6	g 5
4					a		
5		s 7		s 8			
6	r 3	r 3	r 3	r 3	r 3		
7	r 2	r 2	r 2	r 2	r 2		
8	s 3		s 2			g 9	
9	r 4	r 4	r 4	r 4	r 4		

2
8
5
3
1

S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

result

)	$

	(	)	x	,	$	S	L
1	s 3		s 2			g 4	
2	r 1	r 1	r 1	r 1	r 1		
3	s 3		s 2			g 6	g 5
4					a		
5		s 7		s 8			
6	r 3	r 3	r 3	r 3	r 3		
7	r 2	r 2	r 2	r 2	r 2		
8	s 3		s 2			g 9	
9	r 4	r 4	r 4	r 4	r 4		

8
5
3
1

S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

result

)	$

	(	)	x	,	$	S	L
1	s 3		s 2			g 4	
2	r 1	r 1	r 1	r 1	r 1		
3	s 3		s 2			g 6	g 5
4					a		
5		s 7		s 8			
6	r 3	r 3	r 3	r 3	r 3		
7	r 2	r 2	r 2	r 2	r 2		
8	s 3		s 2			g 9	
9	r 4	r 4	r 4	r 4	r 4		

9
8
5
3
1

S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

result

)	$

	(	)	x	,	$	S	L
1	s 3		s 2			g 4	
2	r 1	r 1	r 1	r 1	r 1		
3	s 3		s 2			g 6	g 5
4					a		
5		s 7		s 8			
6	r 3	r 3	r 3	r 3	r 3		
7	r 2	r 2	r 2	r 2	r 2		
8	s 3		s 2			g 9	
9	r 4	r 4	r 4	r 4	r 4		

3
1

S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

result

)	$

	(	)	x	,	$	S	L
1	s 3		s 2			g 4	
2	r 1	r 1	r 1	r 1	r 1		
3	s 3		s 2			g 6	g 5
4					a		
5		s 7		s 8			
6	r 3	r 3	r 3	r 3	r 3		
7	r 2	r 2	r 2	r 2	r 2		
8	s 3		s 2			g 9	
9	r 4	r 4	r 4	r 4	r 4		

5
3
1

S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

result

	(	)	x	,	$	S	L
1	s 3		s 2			g 4	
2	r 1	r 1	r 1	r 1	r 1		
3	s 3		s 2			g 6	g 5
4					a		
5		s 7		s 8			
6	r 3	r 3	r 3	r 3	r 3		
7	r 2	r 2	r 2	r 2	r 2		
8	s 3		s 2			g 9	
9	r 4	r 4	r 4	r 4	r 4		

7
5
3
1

$

S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

result

	(	)	x	,	$	S	L
1	s 3		s 2			g 4	
2	r 1	r 1	r 1	r 1	r 1		
3	s 3		s 2			g 6	g 5
4					a		
5		s 7		s 8			
6	r 3	r 3	r 3	r 3	r 3		
7	r 2	r 2	r 2	r 2	r 2		
8	s 3		s 2			g 9	
9	r 4	r 4	r 4	r 4	r 4		

$

1

S → x
S → (L)
L → S
L → L , S

LR(0) parse tables

result

	(	)	x	,	$	S	L
1	s 3		s 2			g 4	
2	r 1	r 1	r 1	r 1	r 1		
3	s 3		s 2			g 6	g 5
4					a		
5		s 7		s 8			
6	r 3	r 3	r 3	r 3	r 3		
7	r 2	r 2	r 2	r 2	r 2		
8	s 3		s 2			g 9	
9	r 4	r 4	r 4	r 4	r 4		

$

4
1

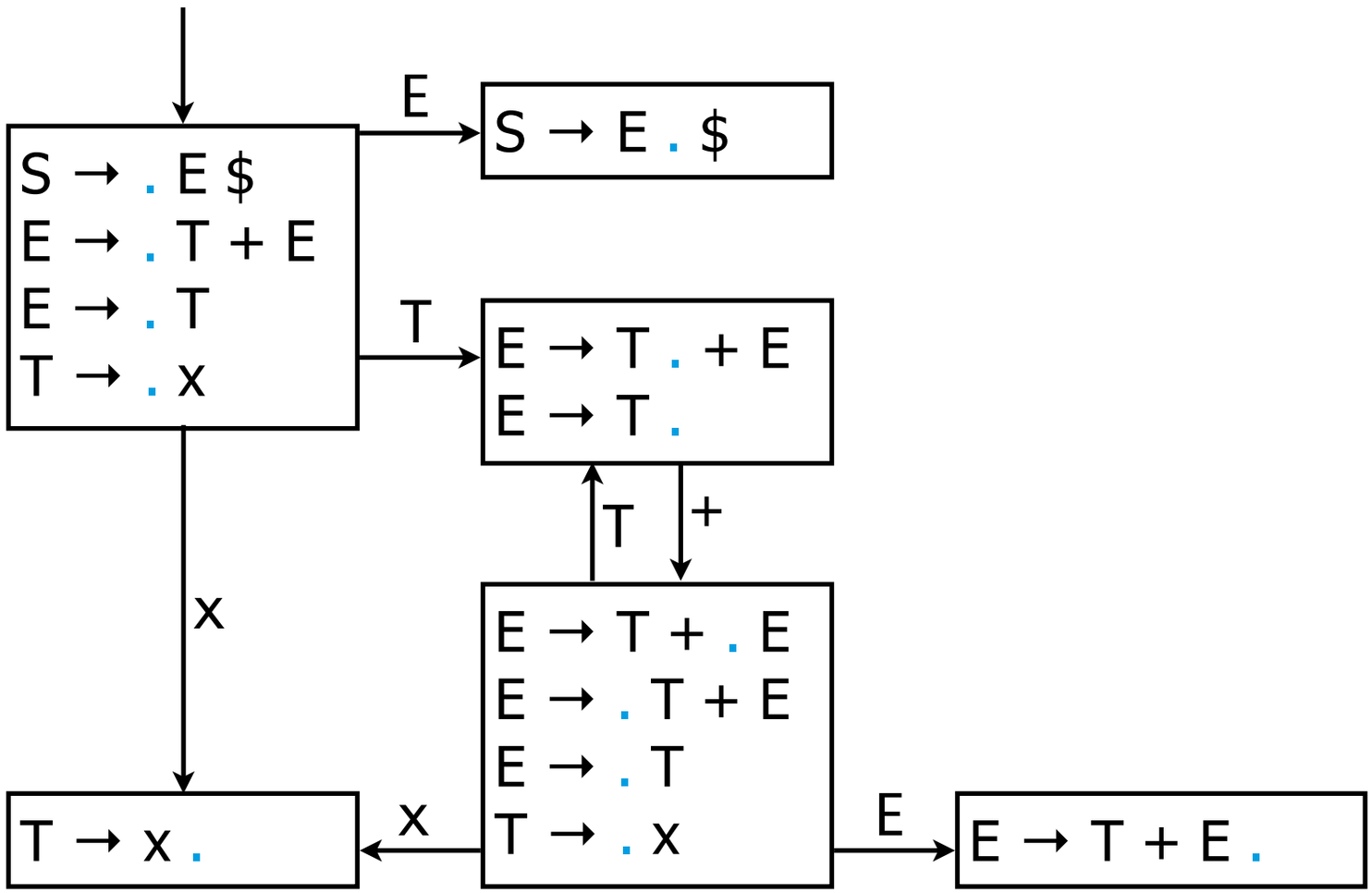
S → x
S → (L)
L → S
L → L , S

III

Conflict Resolution

SLR parse tables

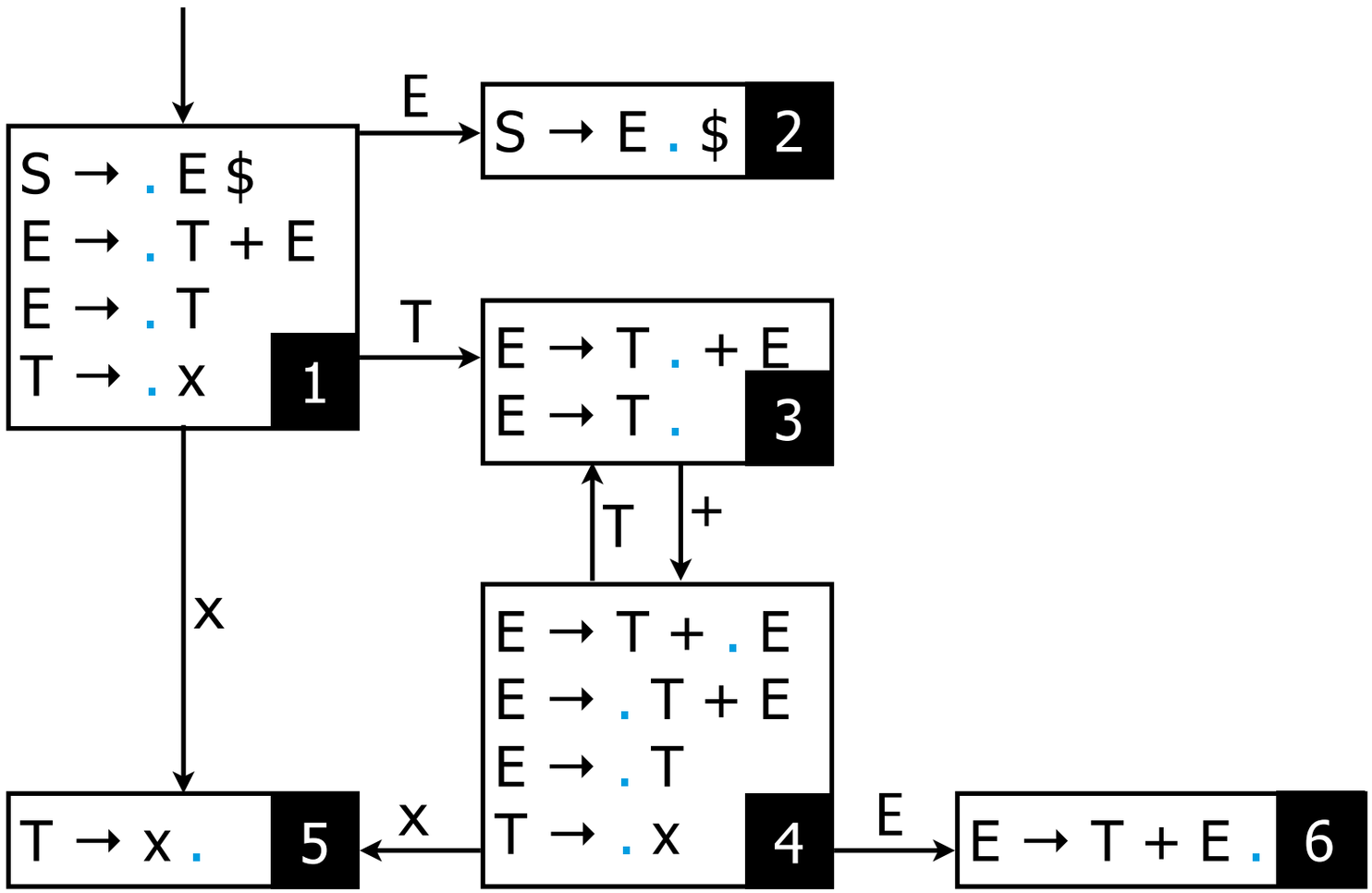
shift-reduce conflicts



E → T + E
E → T
T → x

SLR parse tables

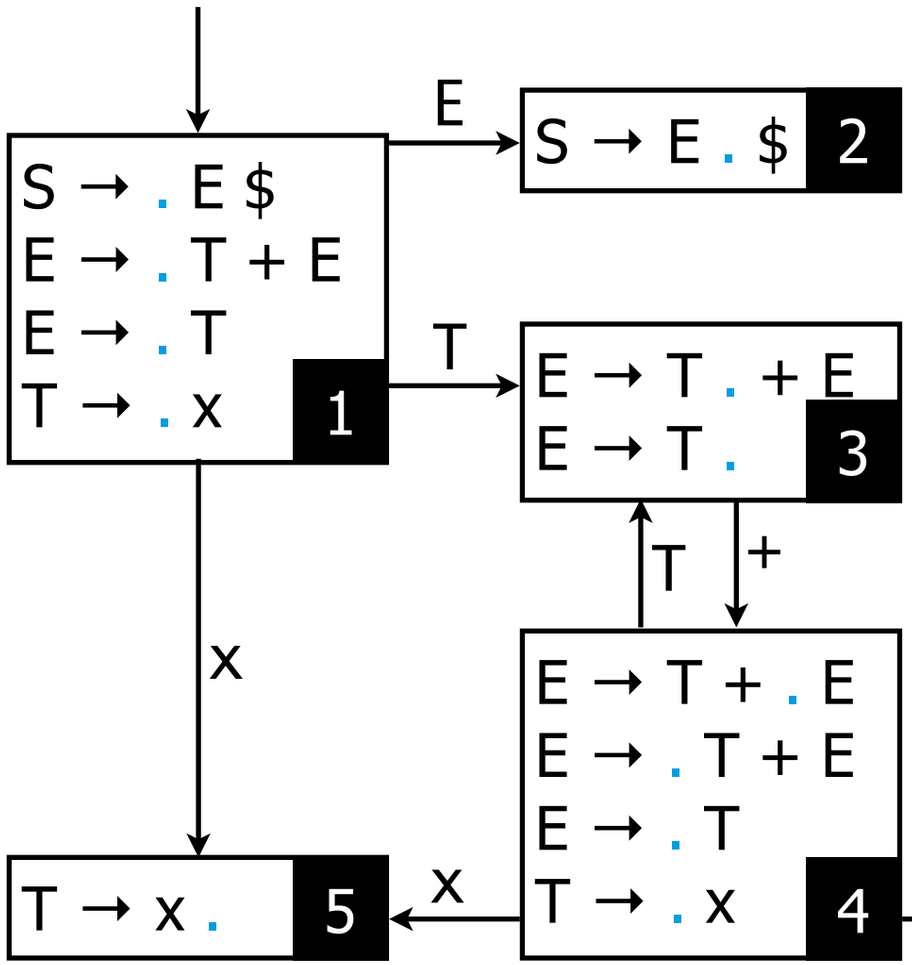
shift-reduce conflicts



E → T + E
E → T
T → x

SLR parse tables

shift-reduce conflicts



	x	+	$	E	T
1	s 5			g 2	g 3
2			a		
3	r 2	?	r 2		
4	s 5			g 6	g 3
5	r 3	r 3	r 3		
6	r 1	r 1	r 1		

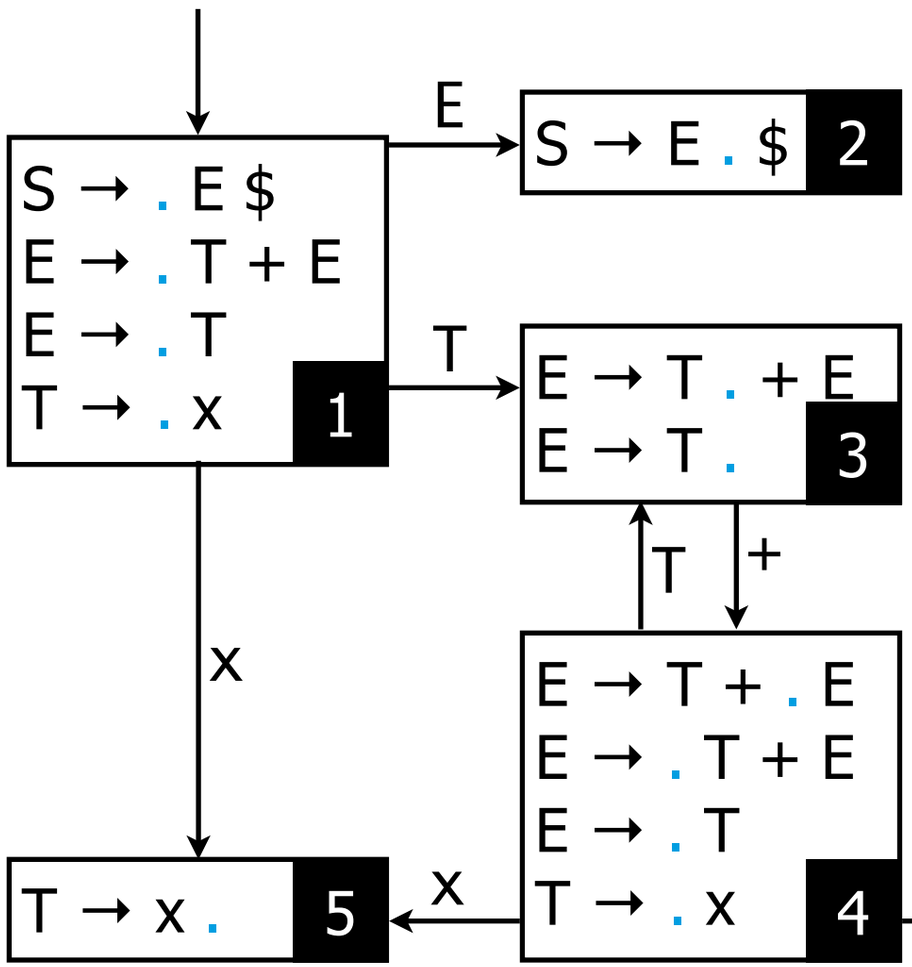
E → T + E
E → T
T → x

SLR parse tables

shift-reduce conflicts

Reduce a production S → …

on symbols k ∈ Σ, k ∈ Follow(S)



	x	+	$	E	T
1	s 5			g 2	g 3
2			a		
3		s 4	r 2		
4	s 5			g 6	g 3
5		r 3	r 3		
6			r 1		

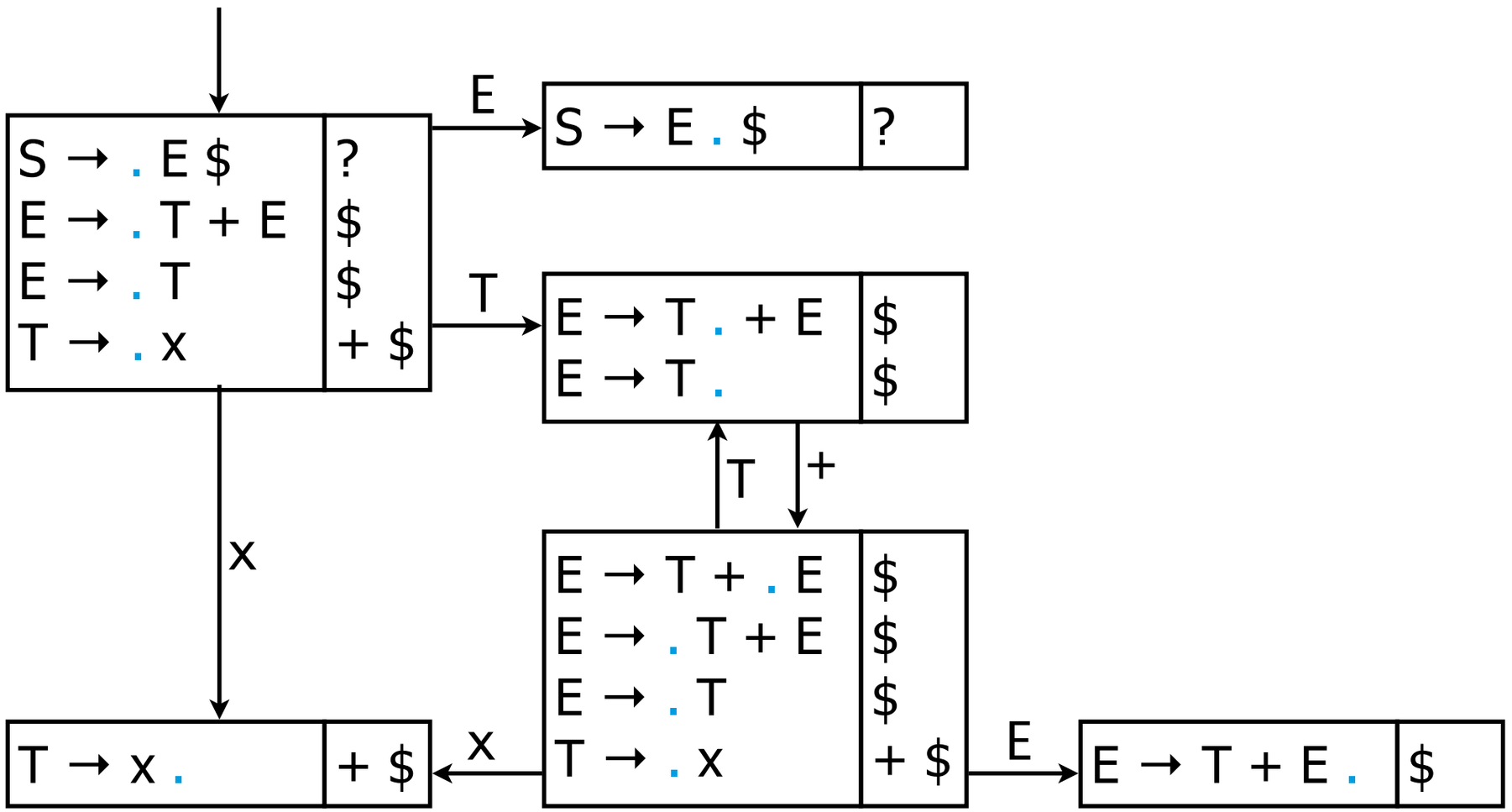
E → T + E
E → T
T → x

LR(1) parse tables

look-ahead

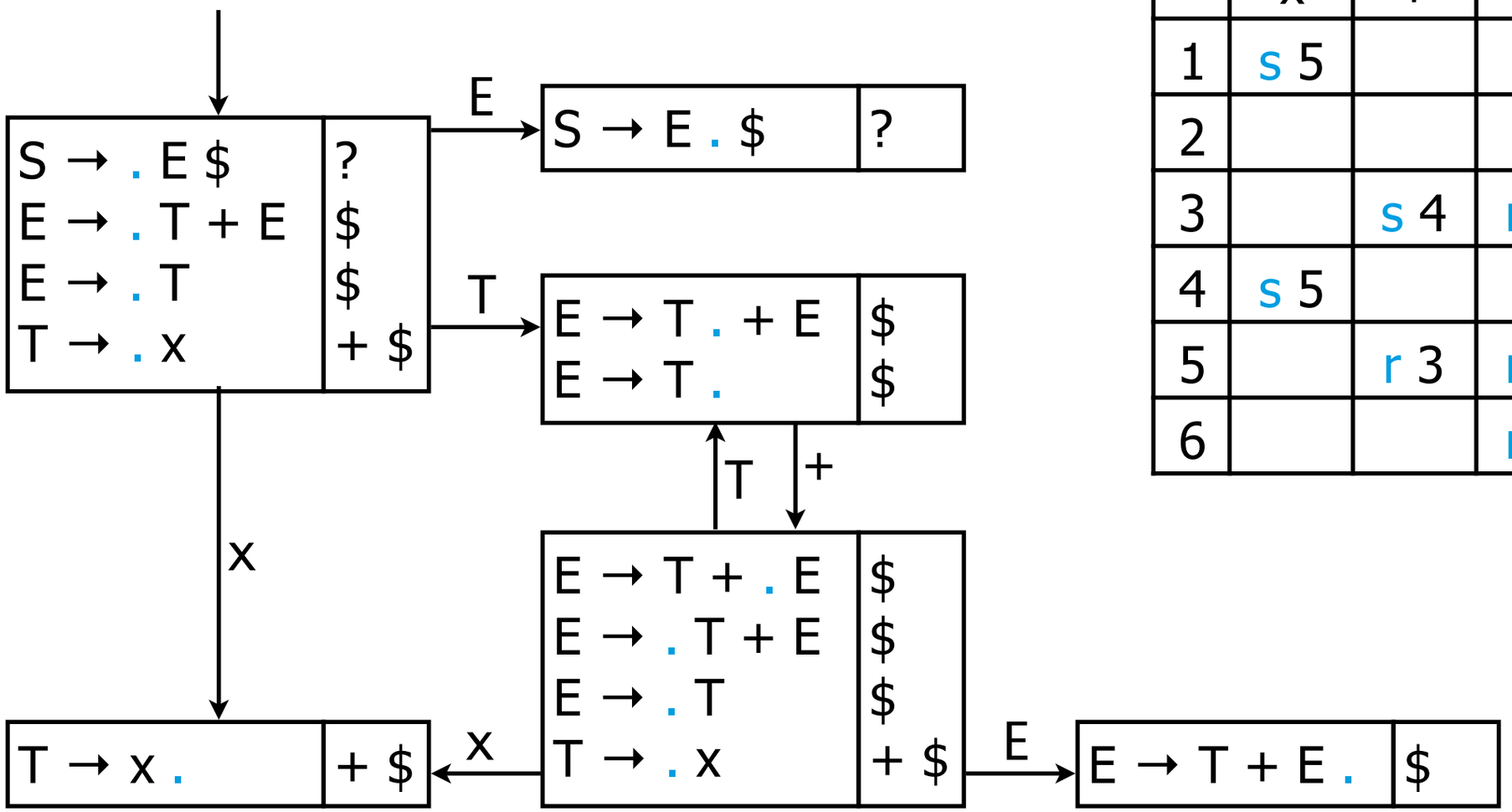
closure

- for every item A → α . X β, z
- for every rule X → γ
- for every w ∈ First(βz)
- add item X → . γ, w



LR(1) parse tables

look-ahead



closure

- for every item A → α . X β, z
- for every rule X → γ
- for every w ∈ First(βz)
- add item X → . γ, w

	x	+	$	E	T
1	s 5			g 2	g 3
2			a		
3		s 4	r 2		
4	s 5			g 6	g 3
5		r 3	r 3		
6			r 1		

E → T + E
E → T
T → x

LALR(1) parse tables
state space reduction

unify states

- with same items
- and same outgoing transitions
- but different look-ahead sets

might introduce new conflicts

LALR(1) parse tables

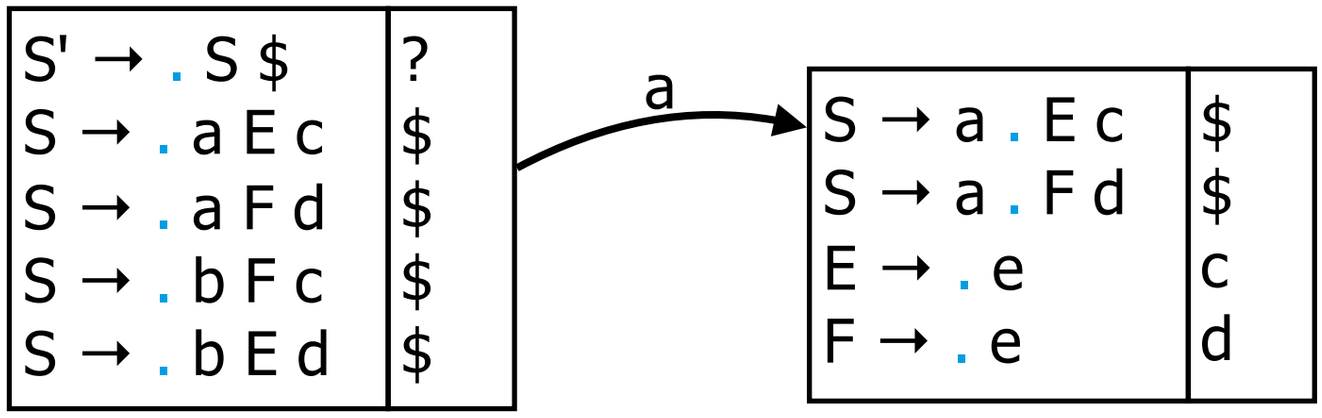
state space reduction

S' → . S $	?
S → . a E c	$
S → . a F d	$
S → . b F c	$
S → . b E d	$

S′ → S $
S → a E c
S → a F d
S → b F c
S → b E d
E → e
F → e

LALR(1) parse tables

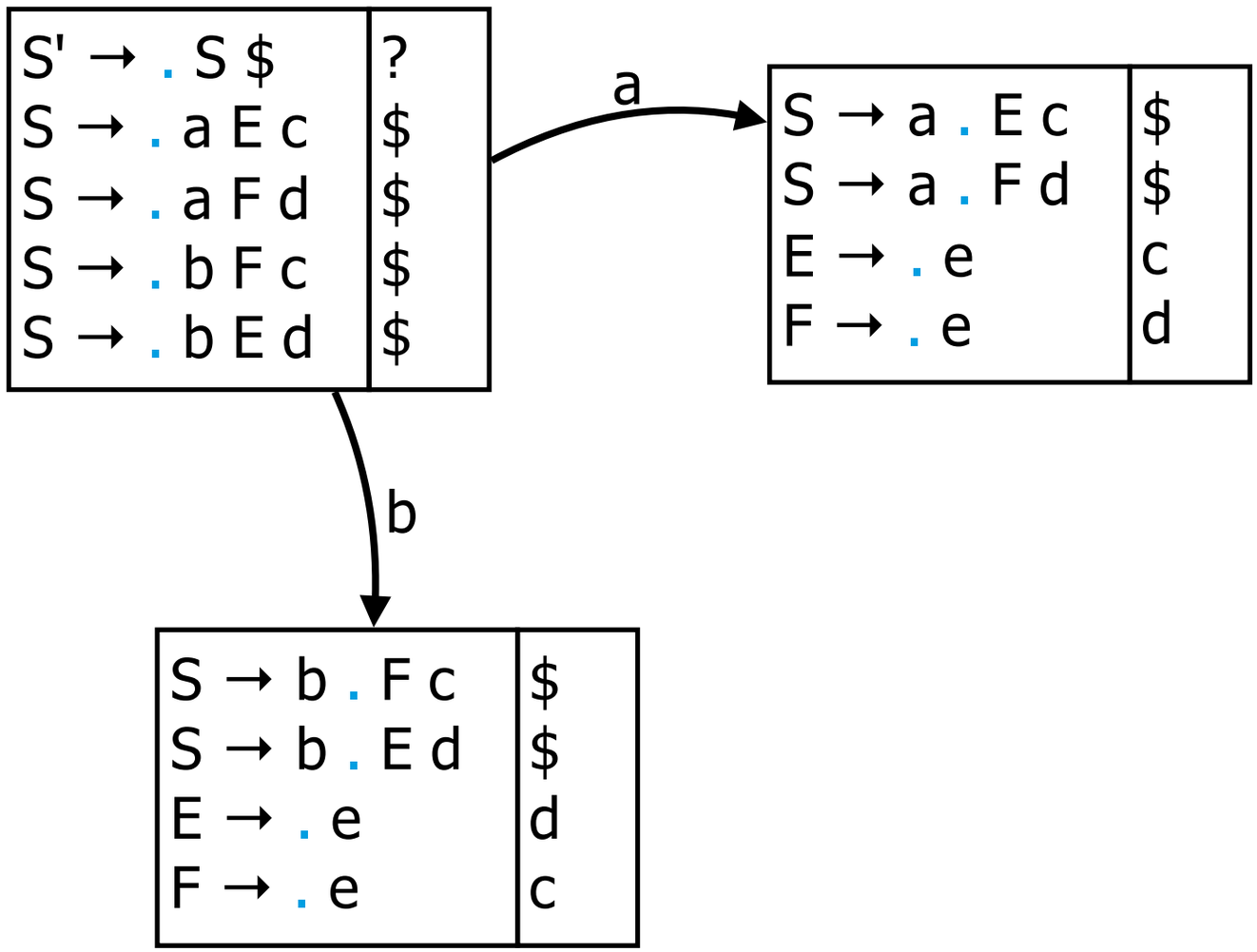
state space reduction



S′ → S $
S → a E c
S → a F d
S → b F c
S → b E d
E → e
F → e

LALR(1) parse tables

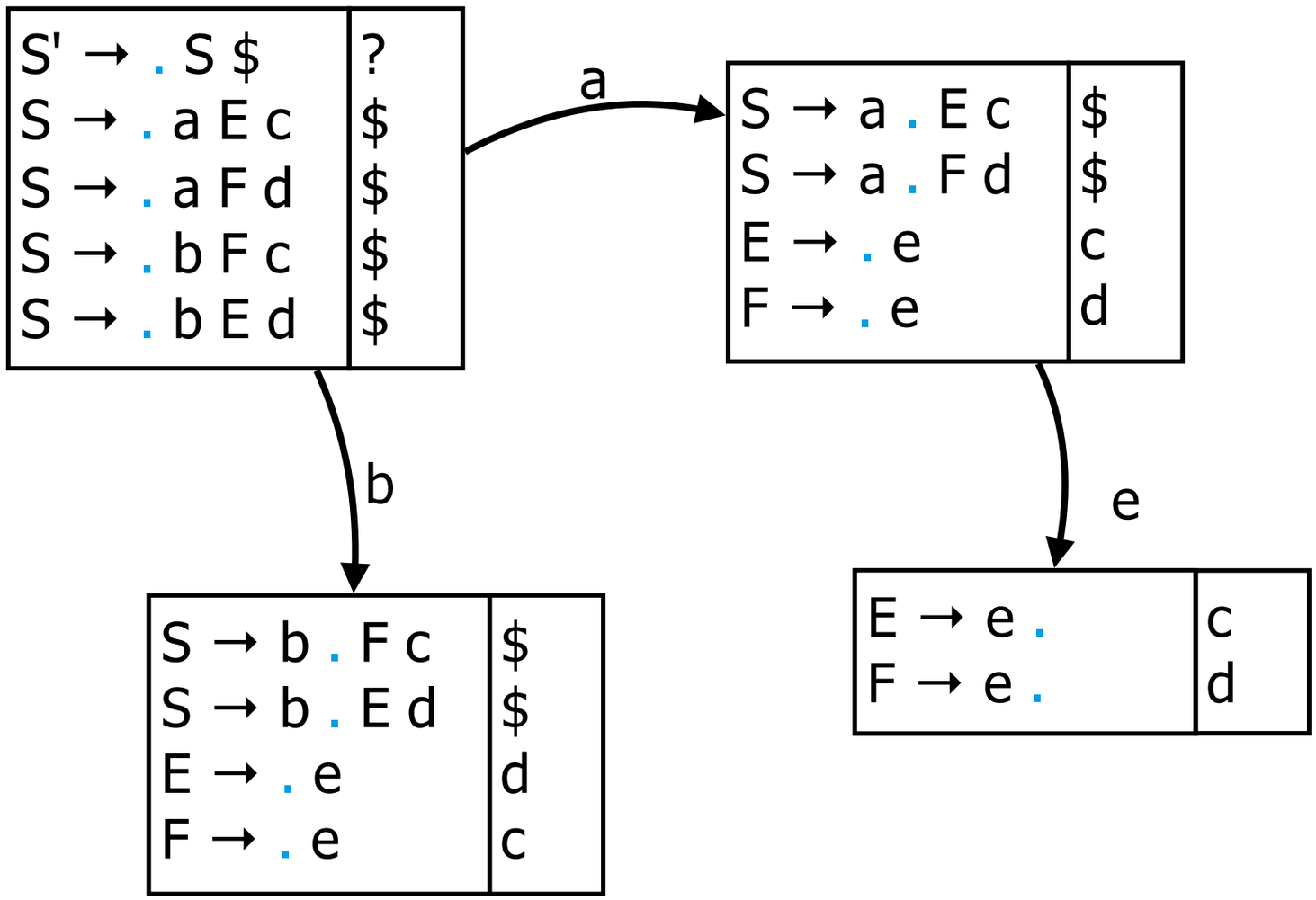
state space reduction



S′ → S $
S → a E c
S → a F d
S → b F c
S → b E d
E → e
F → e

LALR(1) parse tables

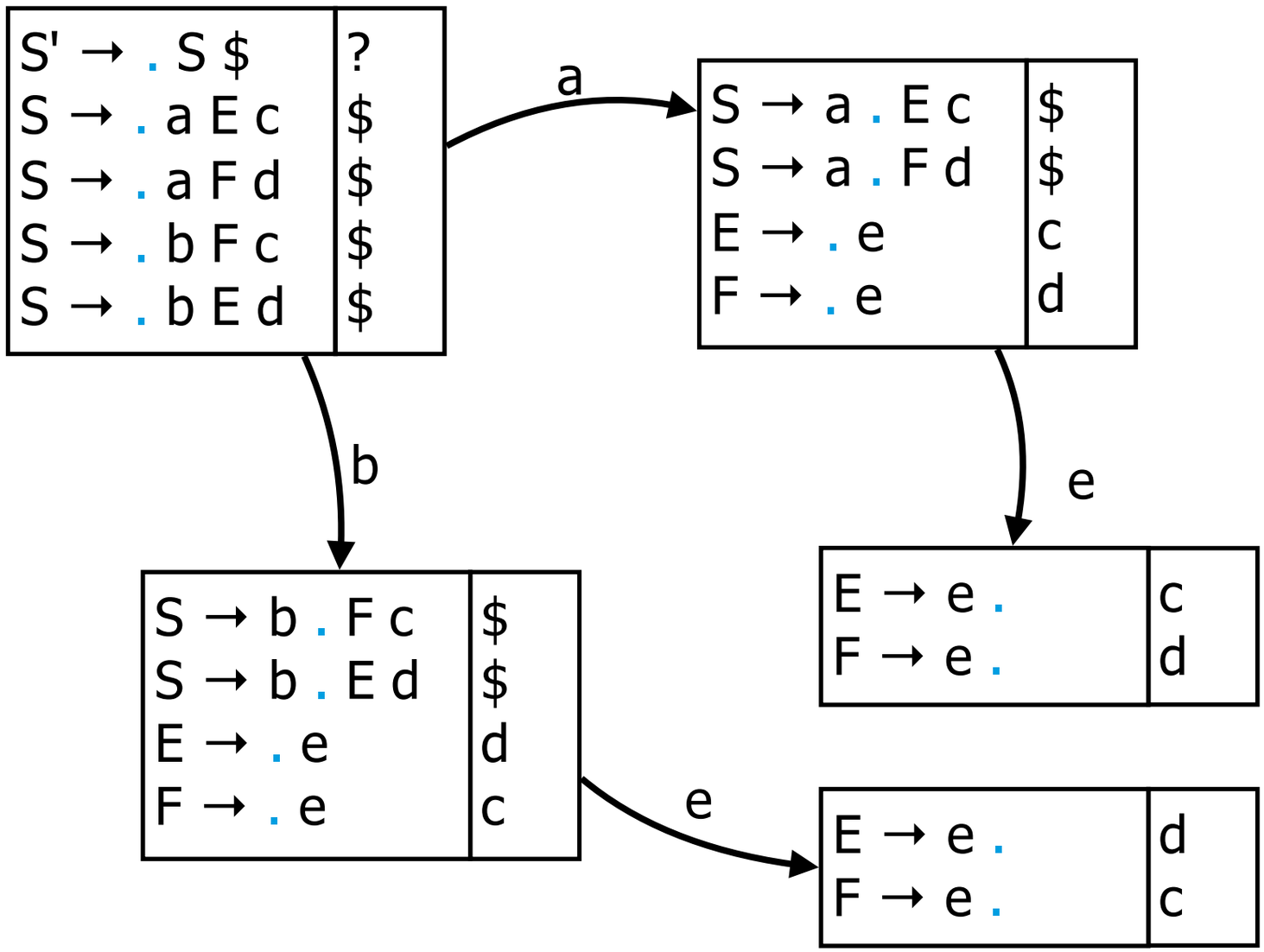
state space reduction



S′ → S $
S → a E c
S → a F d
S → b F c
S → b E d
E → e
F → e

LALR(1) parse tables

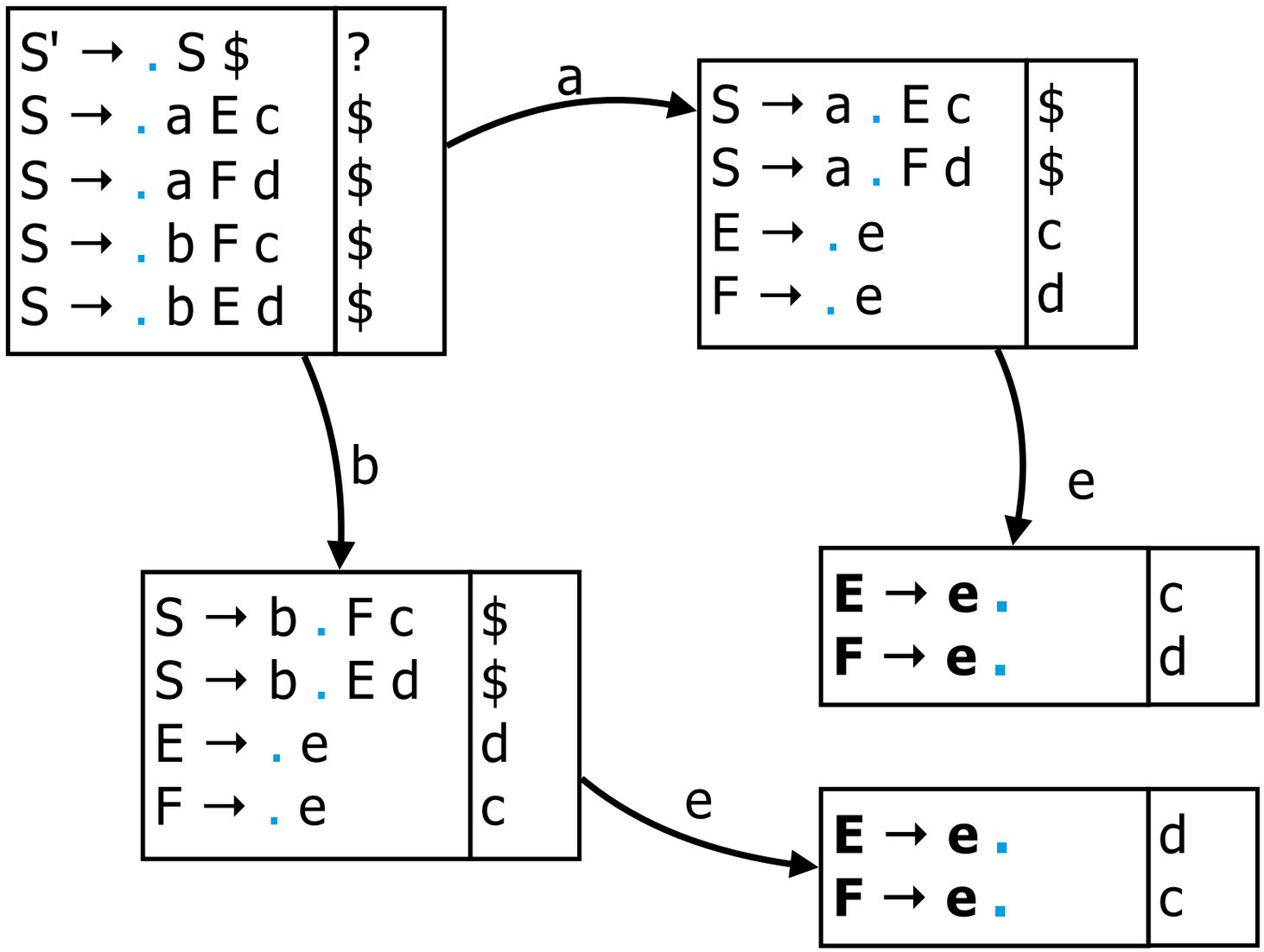
state space reduction



S′ → S $
S → a E c
S → a F d
S → b F c
S → b E d
E → e
F → e

LALR(1) parse tables

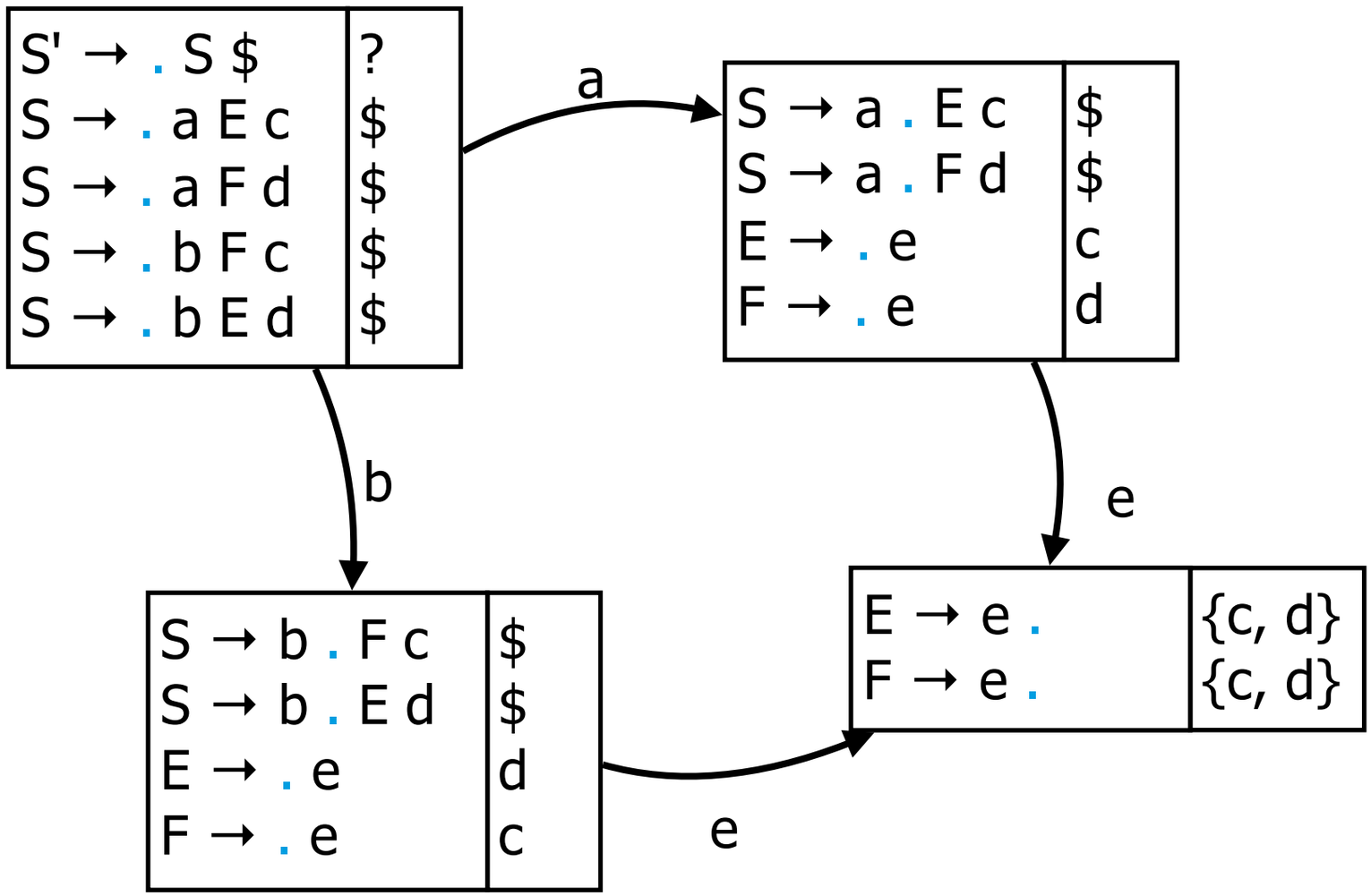
state space reduction



S′ → S $
S → a E c
S → a F d
S → b F c
S → b E d
E → e
F → e

LALR(1) parse tables

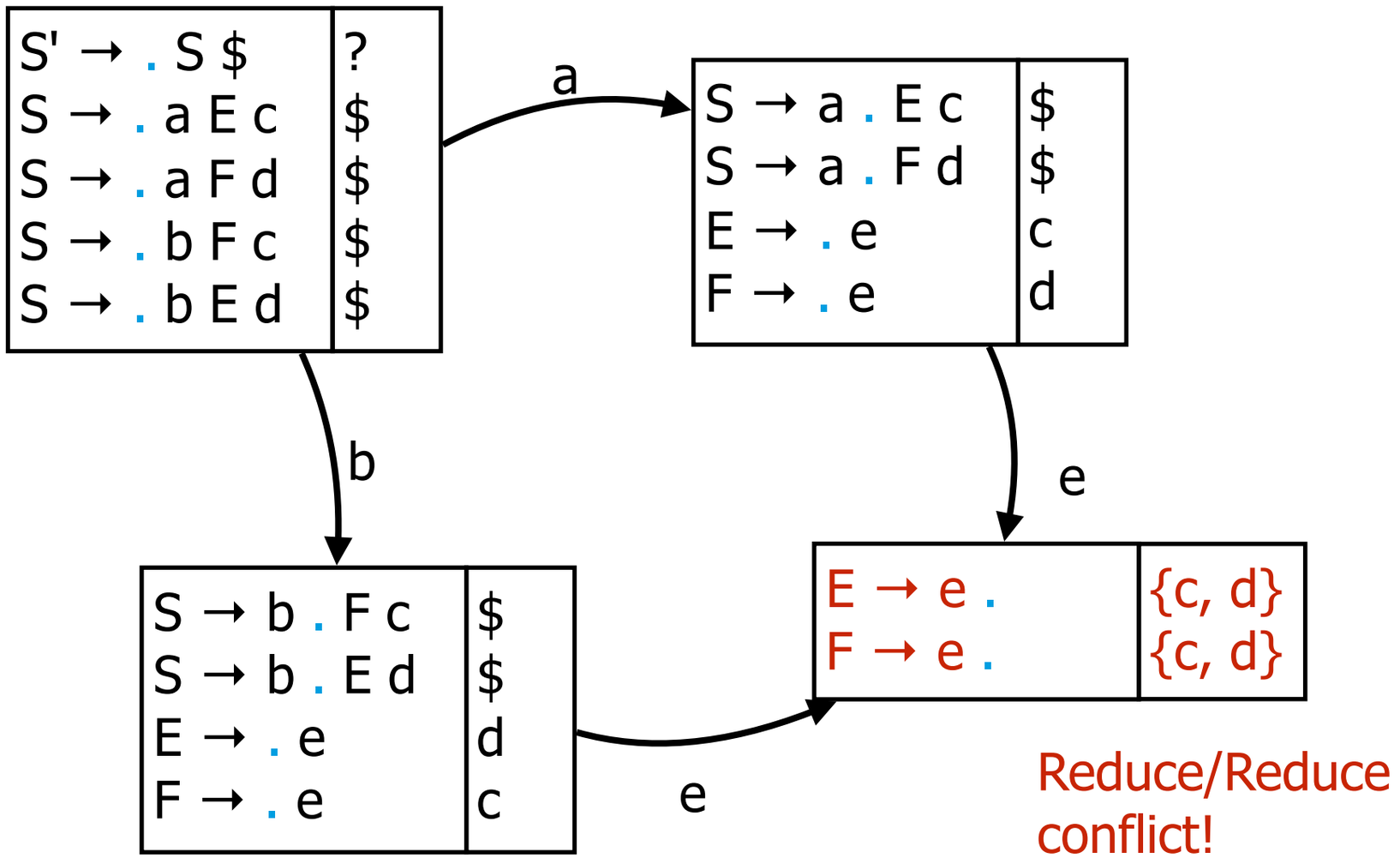
state space reduction



S′ → S $
S → a E c
S → a F d
S → b F c
S → b E d
E → e
F → e

LALR(1) parse tables

state space reduction



S′ → S $
S → a E c
S → a F d
S → b F c
S → b E d
E → e
F → e

IV

Generalized-LR Parsing

Generalized Parsing

- Parse all interpretations of the input, therefore it can handle ambiguous grammars.
- Parsers split whenever finding an ambiguous interpretation and act in (pseudo) parallel.
- Multiple parsers can join whenever they finish parsing an ambiguous fragment of the input.
- Some parsers may "die", if the ambiguity was caused by a lack of lookahead.

Generalized LR

- Multiple parsers are synchronized on shift actions.
- Each parser has its own stack, and as they share states, the overall structure becomes a graph (GSS).
- If two parsers have the same state on top of their stack, they are joined into a single parser.
- Reduce actions affect all possible paths from the top of the stack.

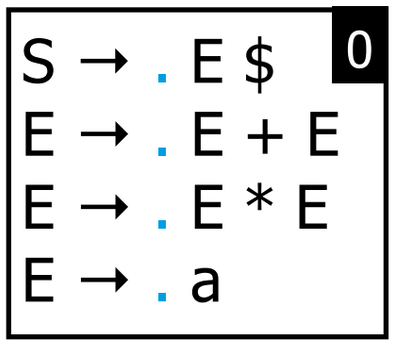
Generalized LR

SLR table

$S \rightarrow E\ \$$
$E \rightarrow E + E$
$E \rightarrow E * E$
$E \rightarrow a$

Generalized LR

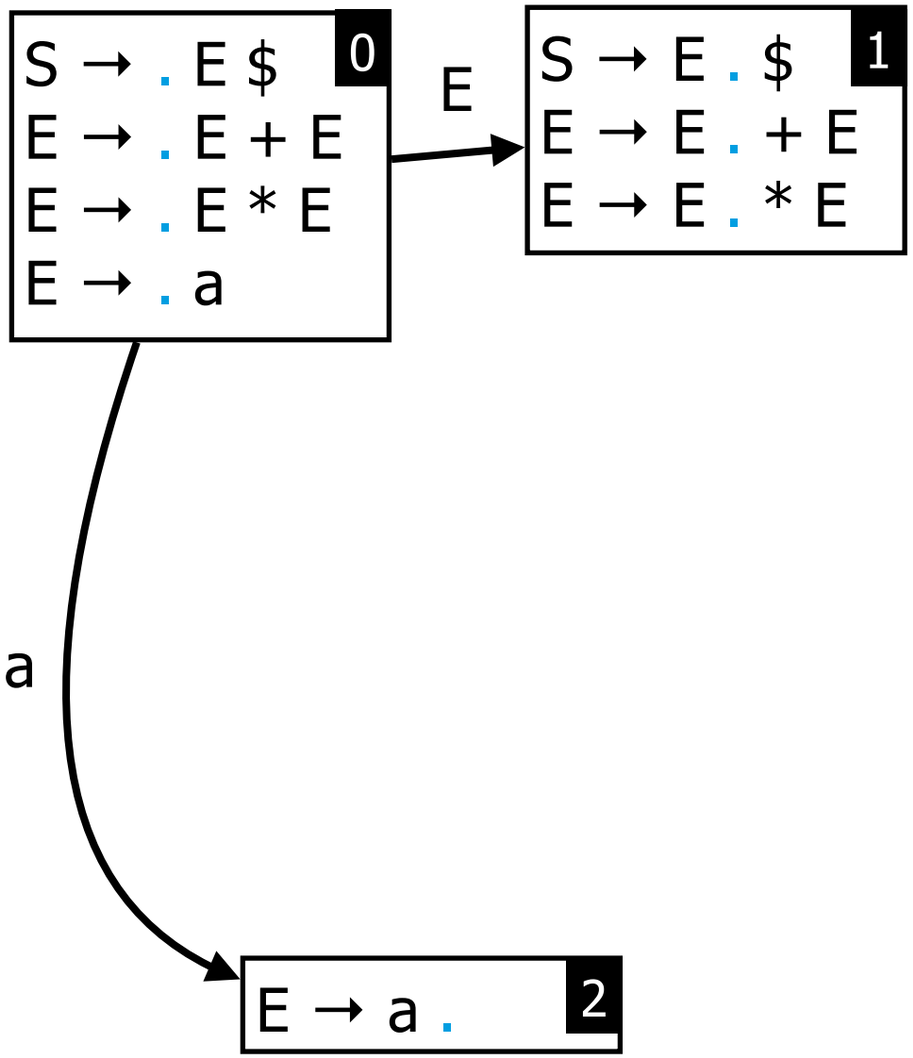
SLR table



S → E $
E → E + E
E → E * E
E → a

Generalized LR

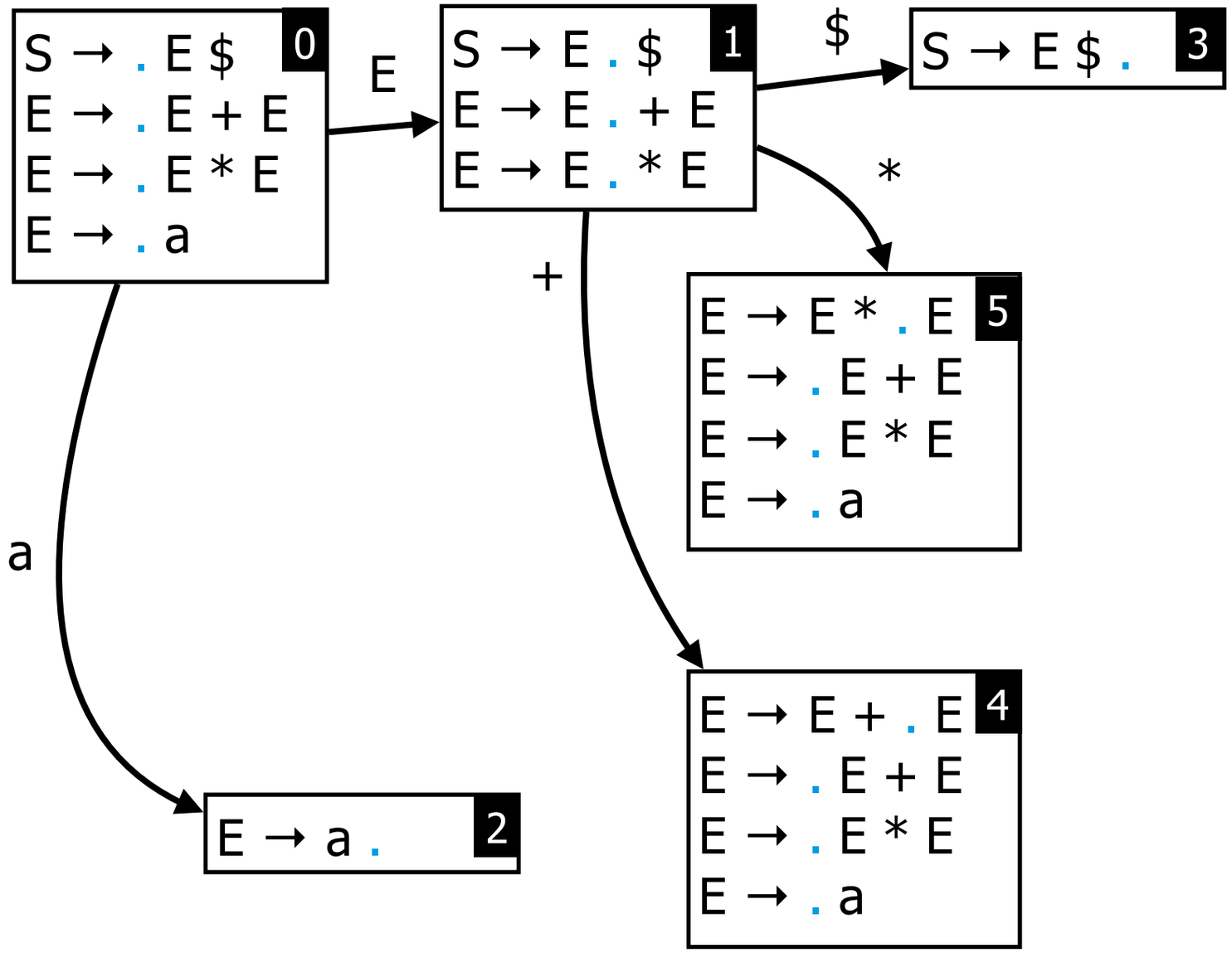
SLR table



S → E $
E → E + E
E → E * E
E → a

Generalized LR

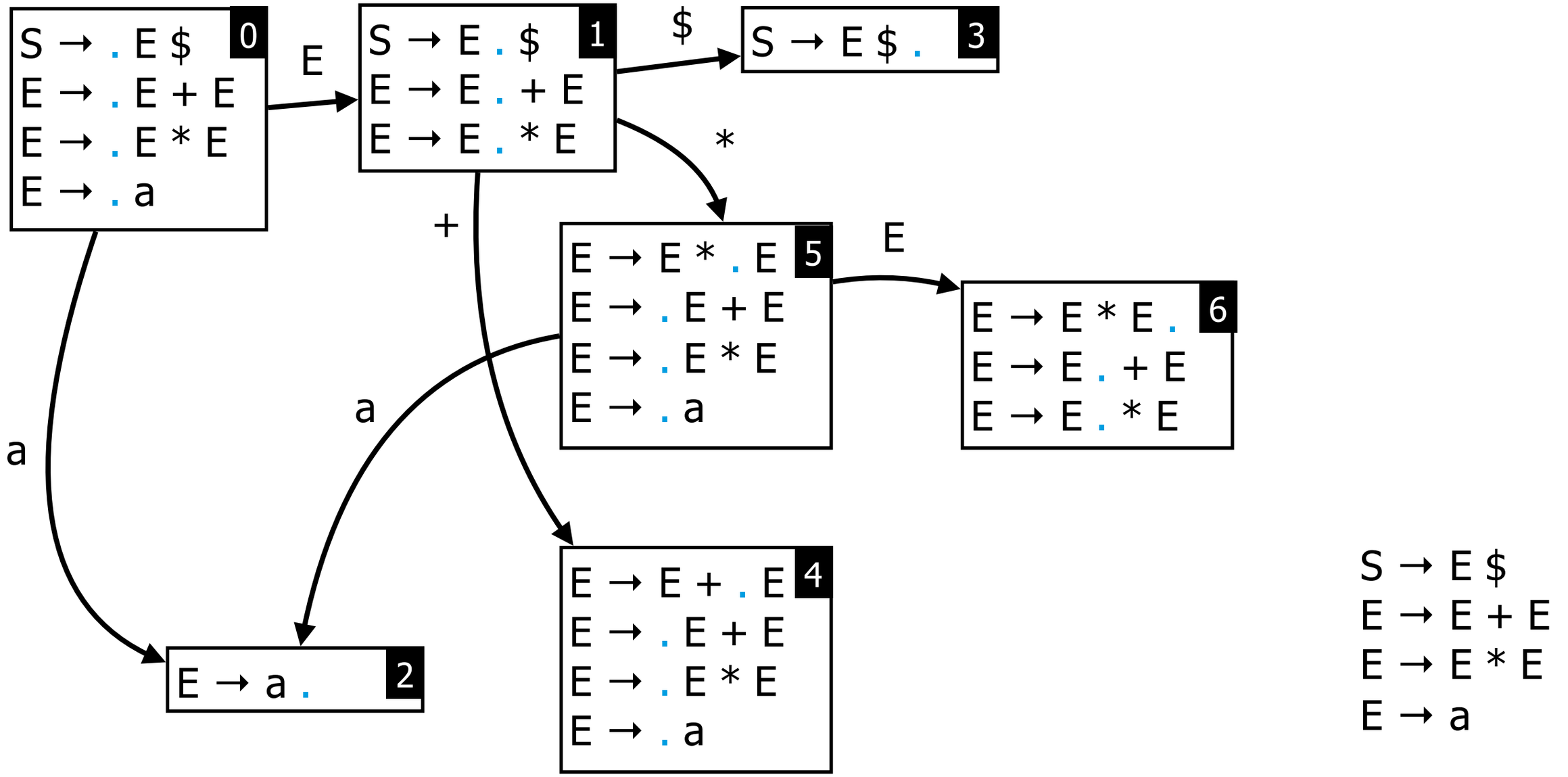
SLR table



S → E $
E → E + E
E → E * E
E → a

Generalized LR

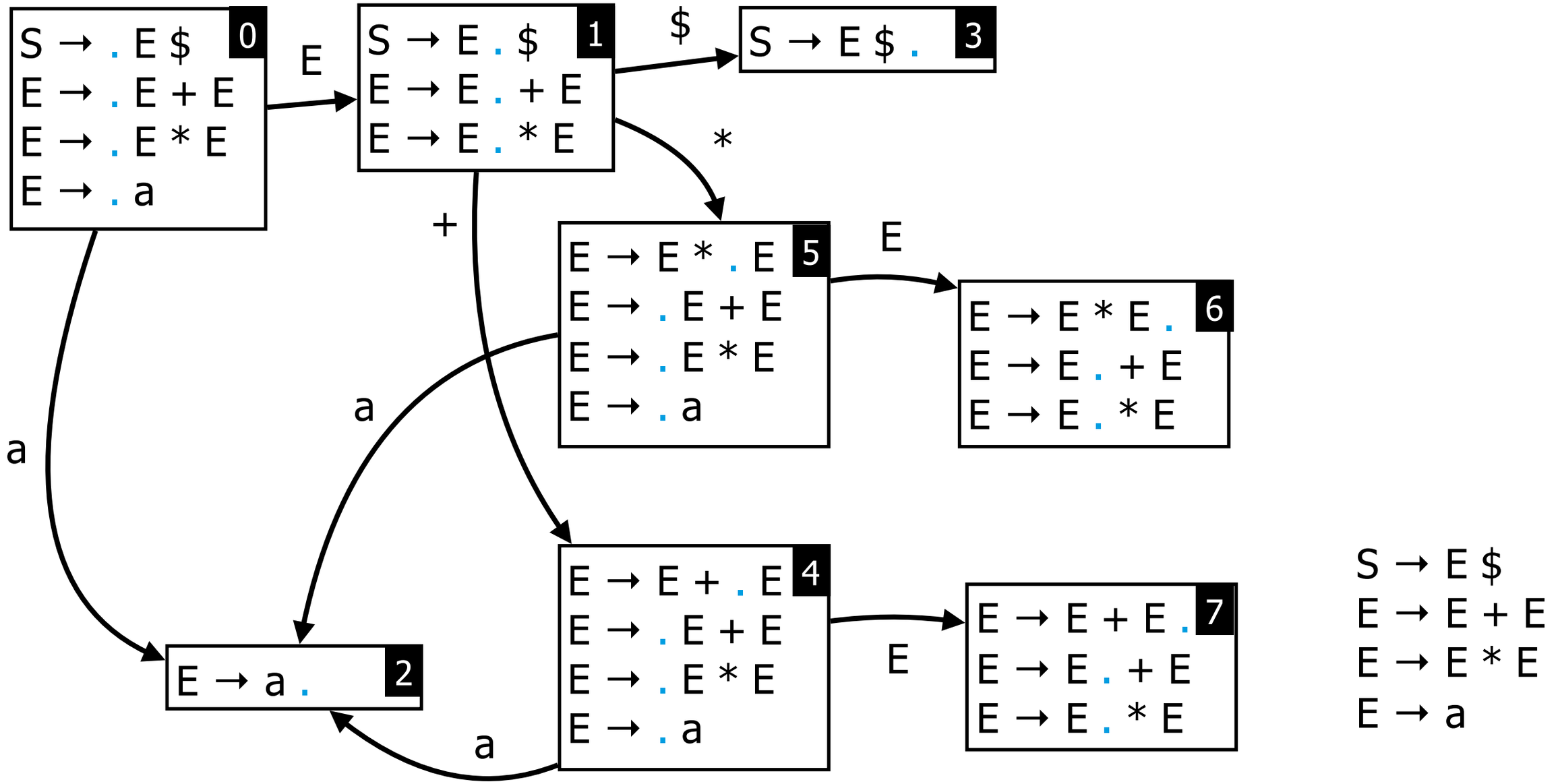
SLR table



S → E $
E → E + E
E → E * E
E → a

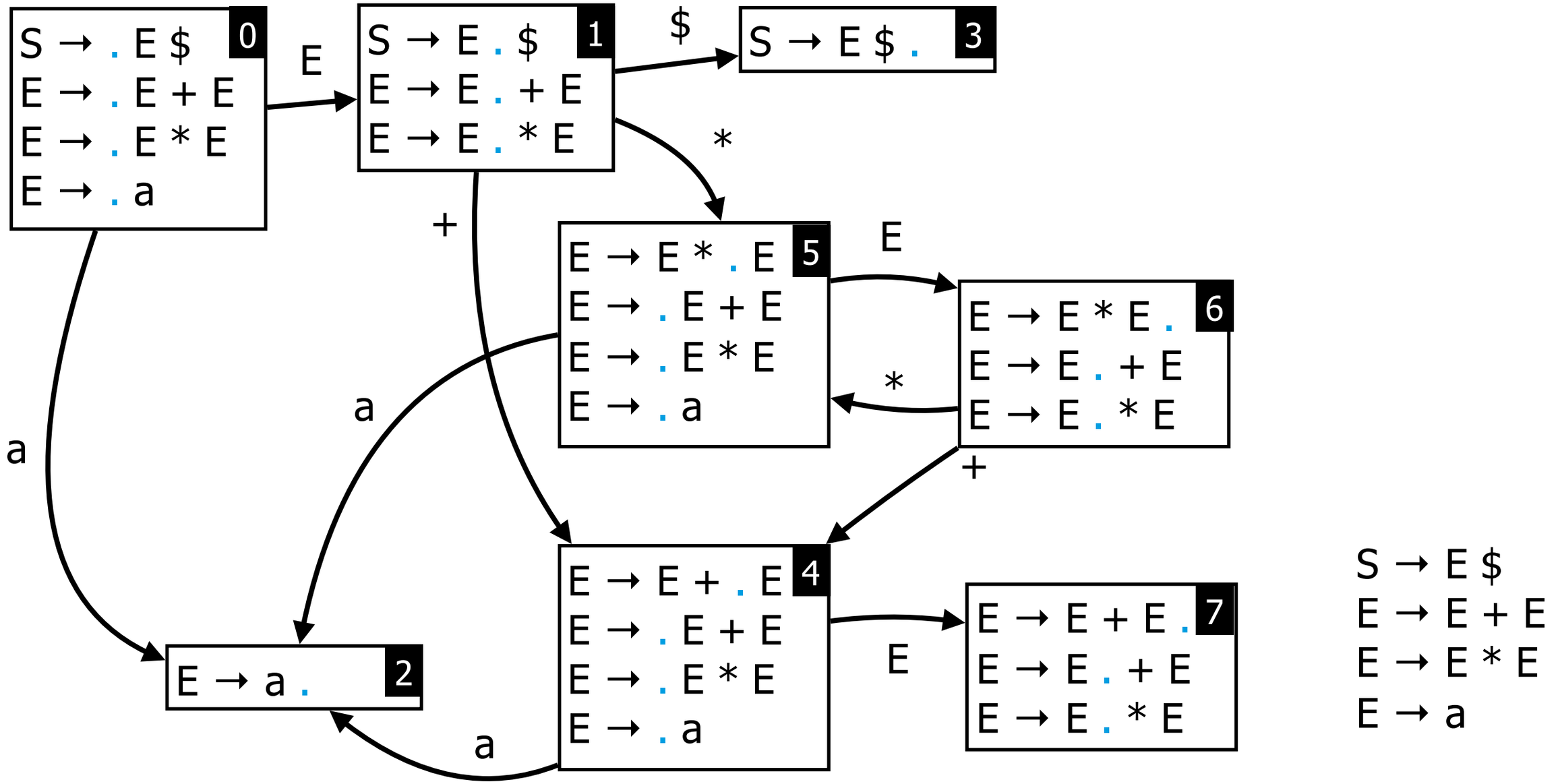
Generalized LR

SLR table



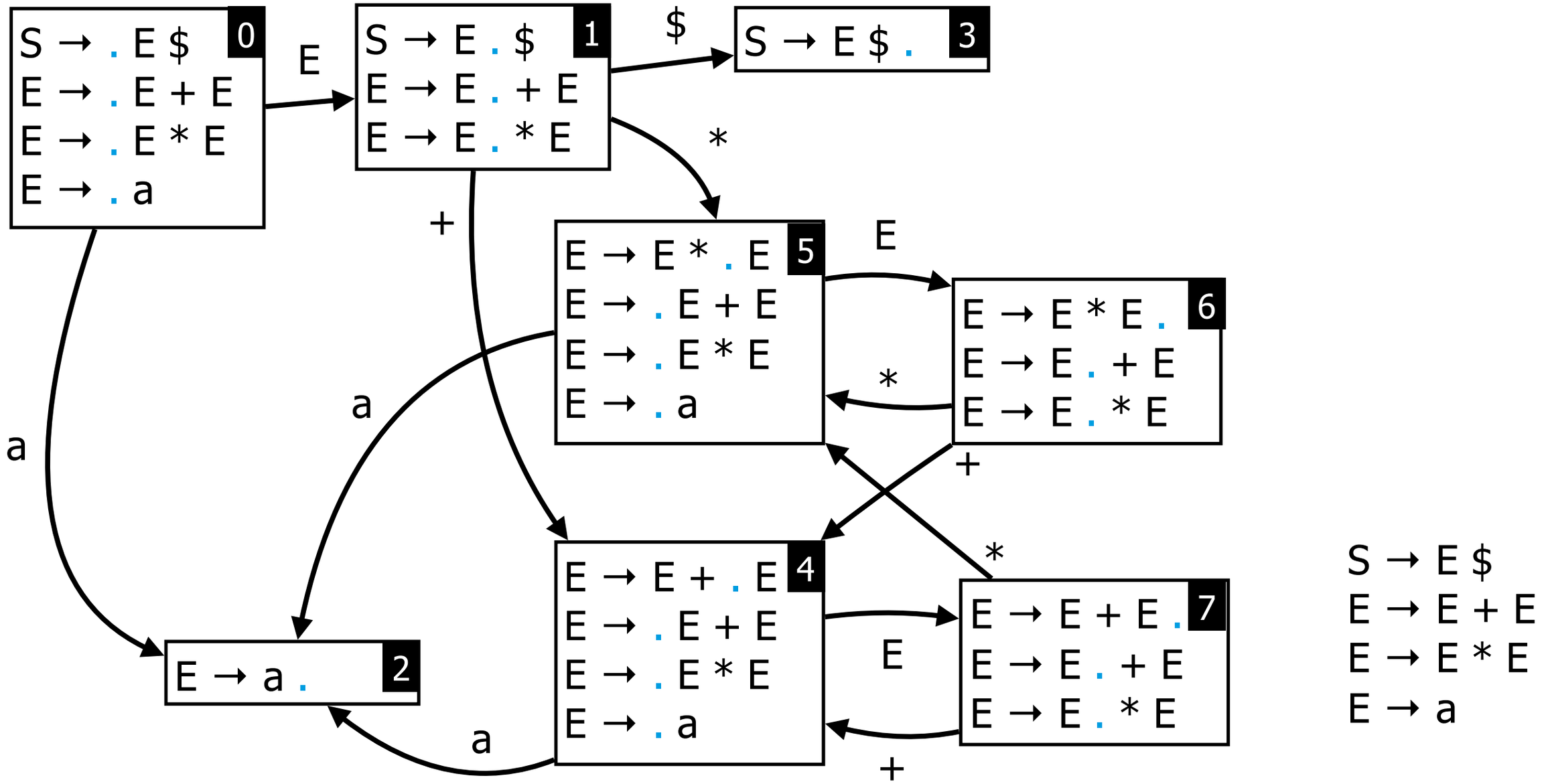
Generalized LR

SLR table



Generalized LR

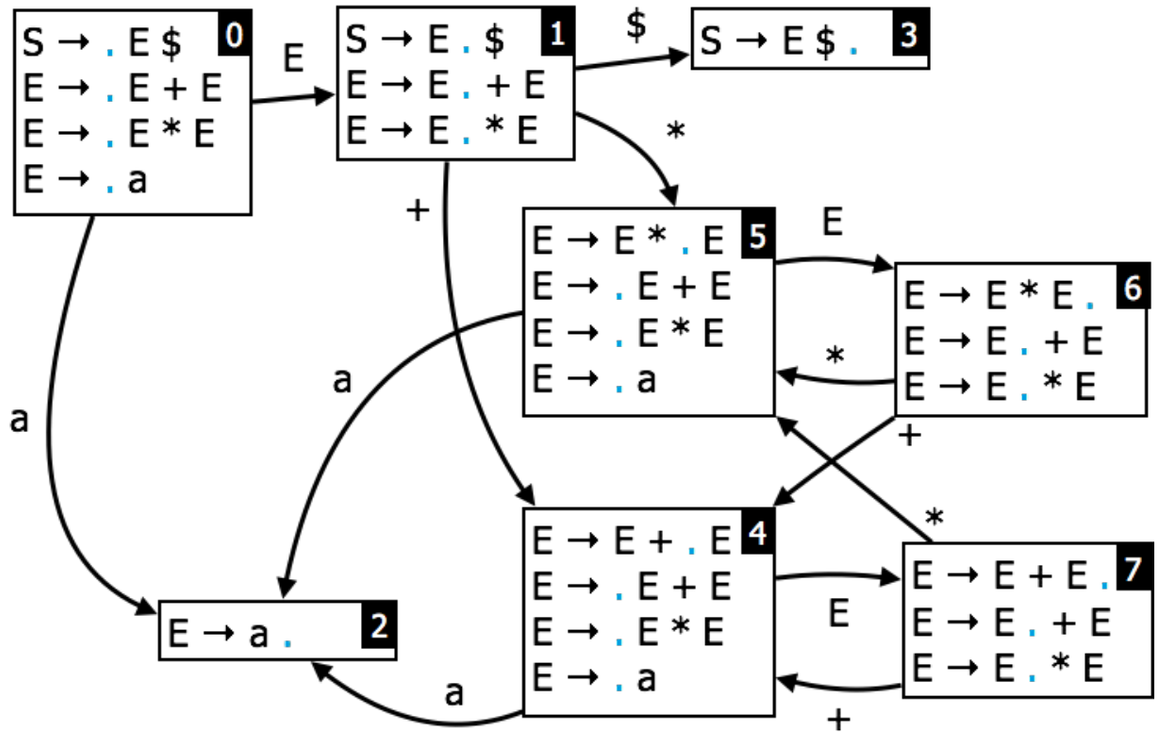
SLR table



Generalized LR

SLR table

Nonter	Nullable	First	Follow
S			
E			



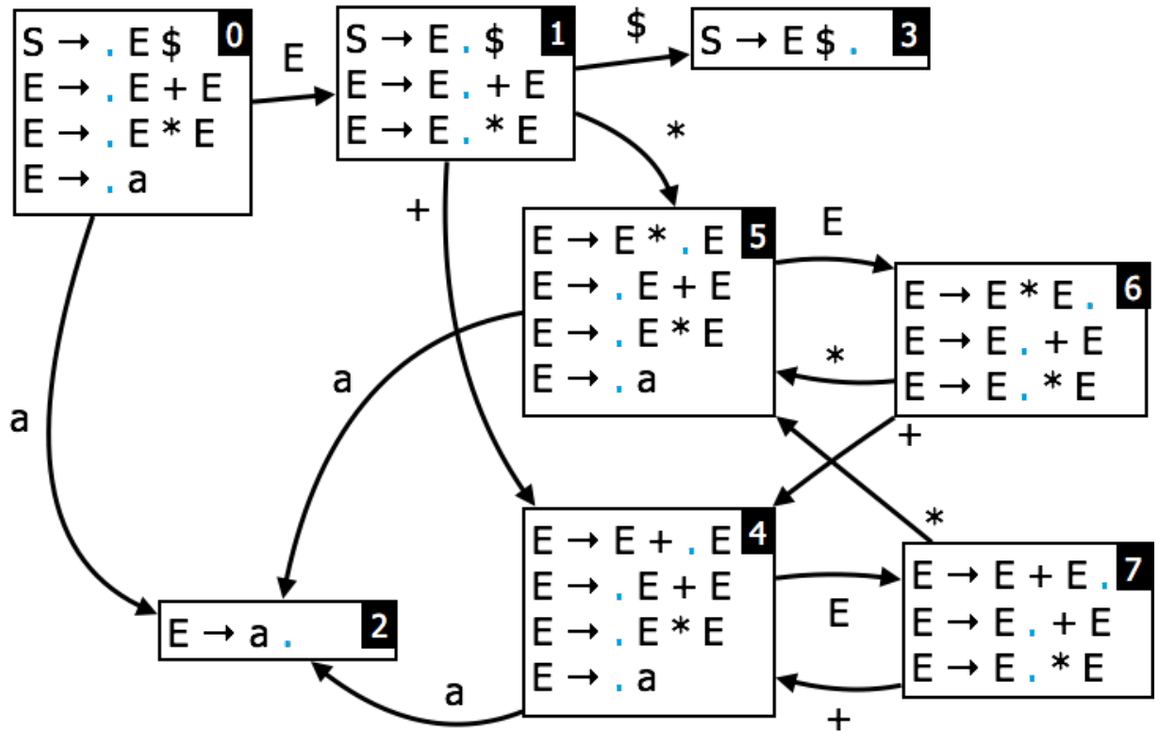
State	Action				Goto	
	a	+	*	$	S	E
0						
1						
2						
3						
4						
5						
6						
7						

(0) S → E $
(1) E → E + E
(2) E → E * E
(3) E → a

Generalized LR

SLR table

Nonter	Nullable	First	Follow
S	no	a	-
E	no	a	+, *, $



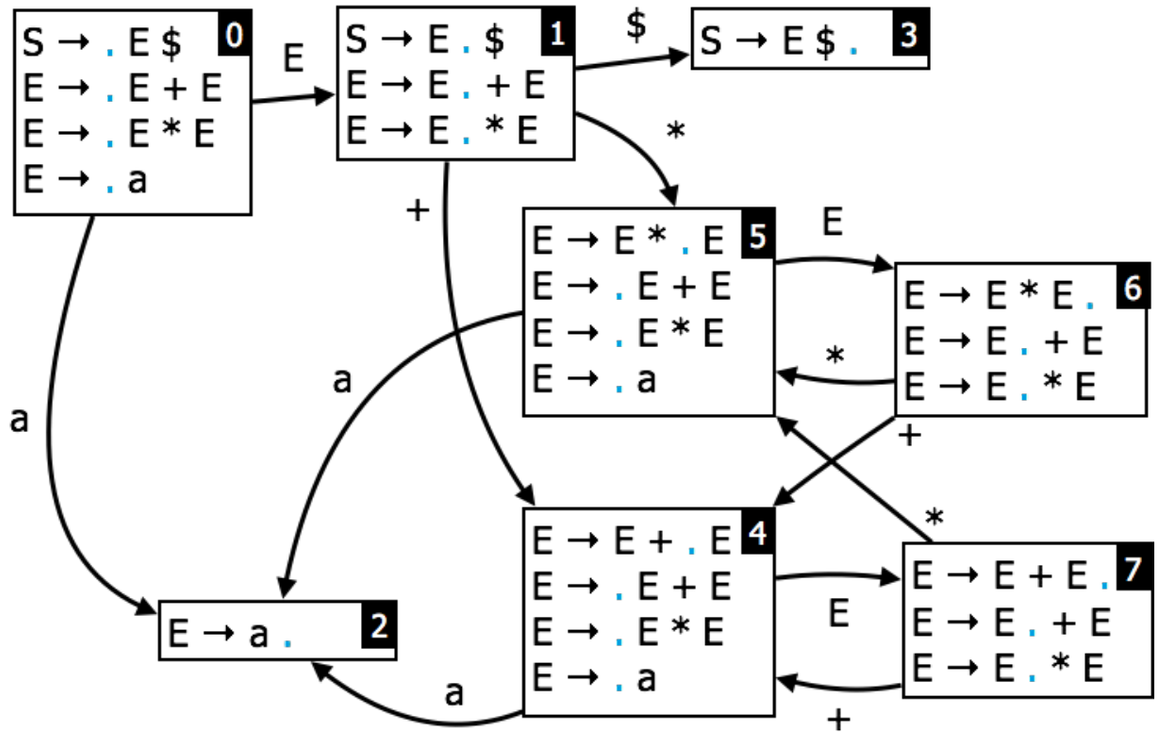
State	Action				Goto	
	a	+	*	$	S	E
0						
1						
2						
3						
4						
5						
6						
7						

(0) S → E $
(1) E → E + E
(2) E → E * E
(3) E → a

Generalized LR

SLR table

Nonter	Nullable	First	Follow
S	no	a	-
E	no	a	+, *, $



State	Action				Goto	
	a	+	*	$	S	E
0	s2					1
1		s4	s5	s3		
2		r3	r3	r3		
3				acc		
4	s2					7
5	s2					6
6		s4/r2	s5/r2	r2		
7		s4/r1	s5/r1	r1		

(0) S → E $
(1) E → E + E
(2) E → E * E
(3) E → a

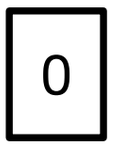
Generalized LR

Parsing

a + a * a

(0) S → E $
(1) E → E + E
(2) E → E * E
(3) E → a

State	Action				Goto	
	a	+	*	$	S	E
0	s2					1
1		s4	s5	s3		
2		r3	r3	r3		
3				acc		
4	s2					7
5	s2					6
6		s4/r2	s5/r2	r2		
7		s4/r1	s5/r1	r1		



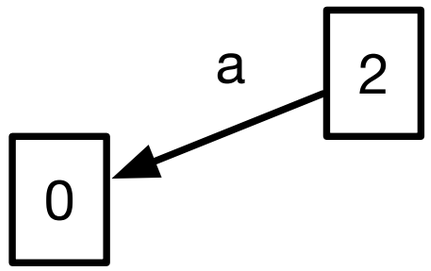
Generalized LR

Parsing

+ a * a

(0) S → E $
(1) E → E + E
(2) E → E * E
(3) E → a

State	Action				Goto	
	a	+	*	$	S	E
0	s2					1
1		s4	s5	s3		
2		r3	r3	r3		
3				acc		
4	s2					7
5	s2					6
6		s4/r2	s5/r2	r2		
7		s4/r1	s5/r1	r1		



synchronize on shifts

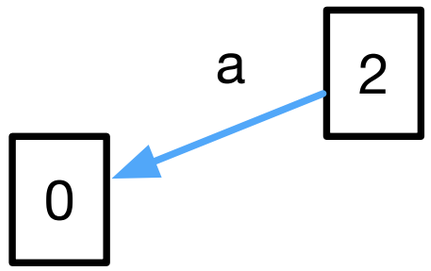
Generalized LR

Parsing

+ a * a

(0) S → E $
(1) E → E + E
(2) E → E * E
(3) E → a

State	Action				Goto	
	a	+	*	$	S	E
0	s2					1
1		s4	s5	s3		
2		r3	r3	r3		
3				acc		
4	s2					7
5	s2					6
6		s4/r2	s5/r2	r2		
7		s4/r1	s5/r1	r1		



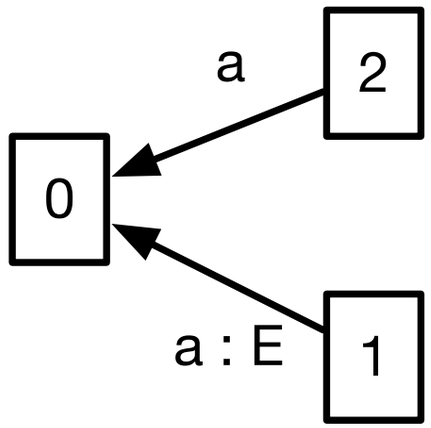
Generalized LR

Parsing

+ a * a

(0) S → E $
(1) E → E + E
(2) E → E * E
(3) E → a

State	Action				Goto	
	a	+	*	$	S	E
0	s2					1
1		s4	s5	s3		
2		r3	r3	r3		
3				acc		
4	s2					7
5	s2					6
6		s4/r2	s5/r2	r2		
7		s4/r1	s5/r1	r1		



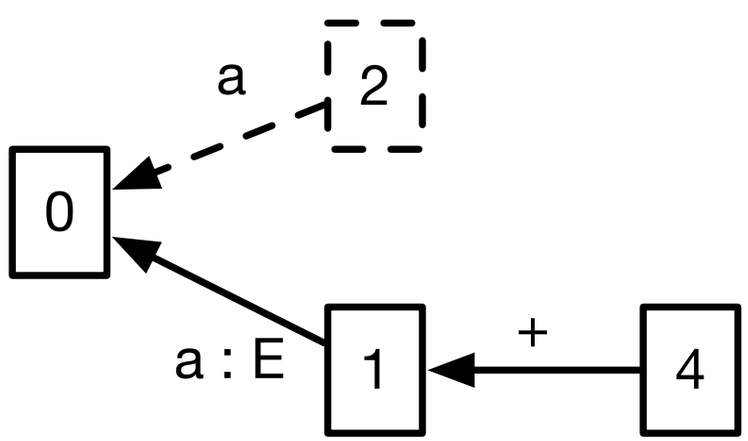
Generalized LR

Parsing

a * a

(0) S → E $
(1) E → E + E
(2) E → E * E
(3) E → a

State	Action				Goto	
	a	+	*	$	S	E
0	s2					1
1		s4	s5	s3		
2		r3	r3	r3		
3				acc		
4	s2					7
5	s2					6
6		s4/r2	s5/r2	r2		
7		s4/r1	s5/r1	r1		



synchronize

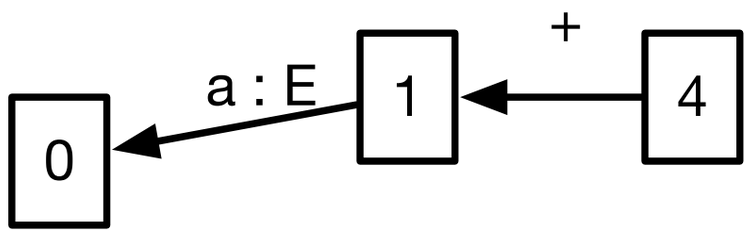
Generalized LR

Parsing

a * a

(0) S → E $
(1) E → E + E
(2) E → E * E
(3) E → a

State	Action				Goto	
	a	+	*	$	S	E
0	s2					1
1		s4	s5	s3		
2		r3	r3	r3		
3				acc		
4	s2					7
5	s2					6
6		s4/r2	s5/r2	r2		
7		s4/r1	s5/r1	r1		



Generalized LR

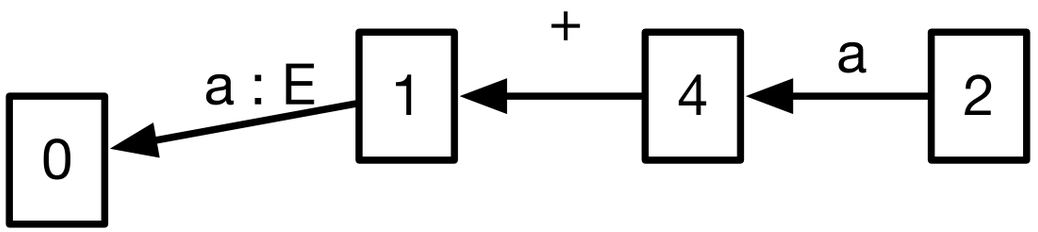
Parsing

(0) S → E $
(1) E → E + E
(2) E → E * E
(3) E → a

State	Action				Goto	
	a	+	*	$	S	E
0	s2					1
1		s4	s5	s3		
2		r3	r3	r3		
3				acc		
4	s2					7
5	s2					6
6		s4/r2	s5/r2	r2		
7		s4/r1	s5/r1	r1		

* a

synchronize



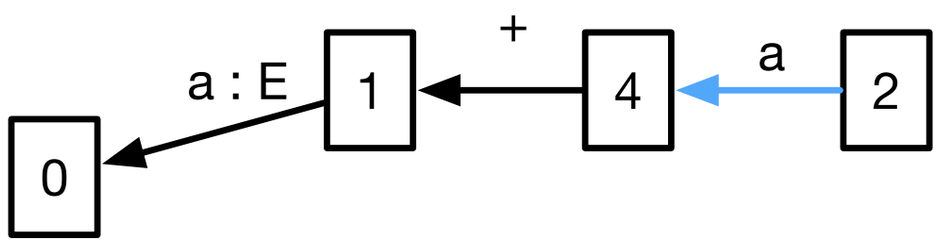
Generalized LR

Parsing

(0) S → E $
(1) E → E + E
(2) E → E * E
(3) E → a

State	Action				Goto	
	a	+	*	$	S	E
0	s2					1
1		s4	s5	s3		
2		r3	r3	r3		
3				acc		
4	s2					7
5	s2					6
6		s4/r2	s5/r2	r2		
7		s4/r1	s5/r1	r1		

* a



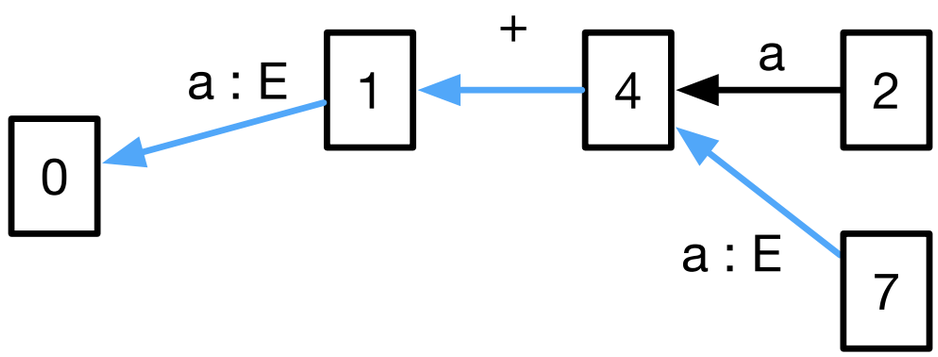
Generalized LR

Parsing

(0) S → E $
(1) E → E + E
(2) E → E * E
(3) E → a

State	Action				Goto	
	a	+	*	$	S	E
0	s2					1
1		s4	s5	s3		
2		r3	r3	r3		
3				acc		
4	s2					7
5	s2					6
6		s4/r2	s5/r2	r2		
7		s4/r1	s5/r1	r1		

* a



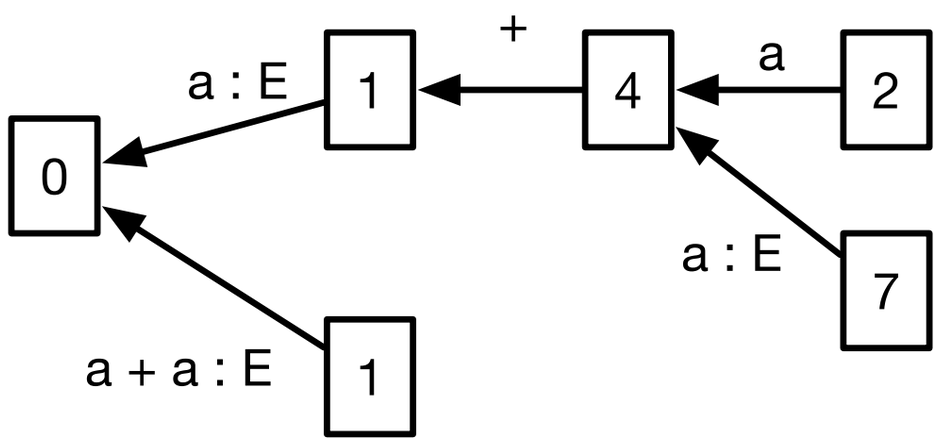
Generalized LR

Parsing

(0) S → E $
(1) E → E + E
(2) E → E * E
(3) E → a

* a

State	Action				Goto	
	a	+	*	$	S	E
0	s2					1
1		s4	s5	s3		
2		r3	r3	r3		
3				acc		
4	s2					7
5	s2					6
6		s4/r2	s5/r2	r2		
7		s4/r1	s5/r1	r1		



Generalized LR

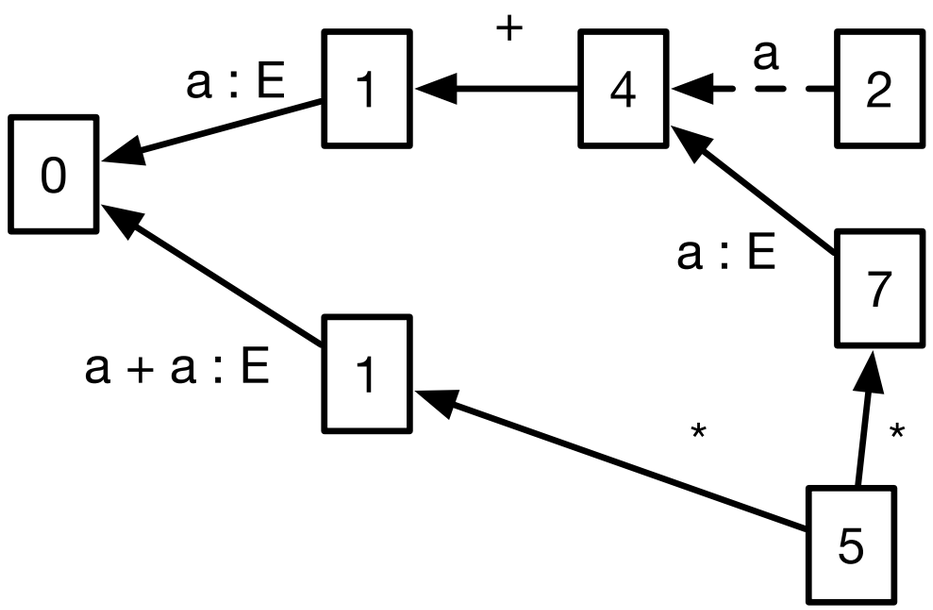
Parsing

a

(0) S → E $
(1) E → E + E
(2) E → E * E
(3) E → a

State	Action				Goto	
	a	+	*	$	S	E
0	s2					1
1		s4	s5	s3		
2		r3	r3	r3		
3				acc		
4	s2					7
5	s2					6
6		s4/r2	s5/r2	r2		
7		s4/r1	s5/r1	r1		

synchronize



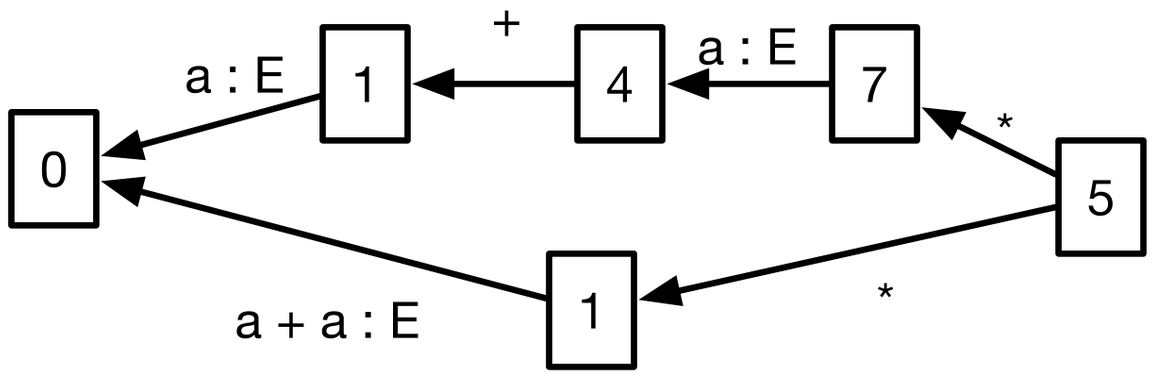
Generalized LR

Parsing

a

(0) S → E $
(1) E → E + E
(2) E → E * E
(3) E → a

State	Action				Goto	
	a	+	*	$	S	E
0	s2					1
1		s4	s5	s3		
2		r3	r3	r3		
3				acc		
4	s2					7
5	s2					6
6		s4/r2	s5/r2	r2		
7		s4/r1	s5/r1	r1		



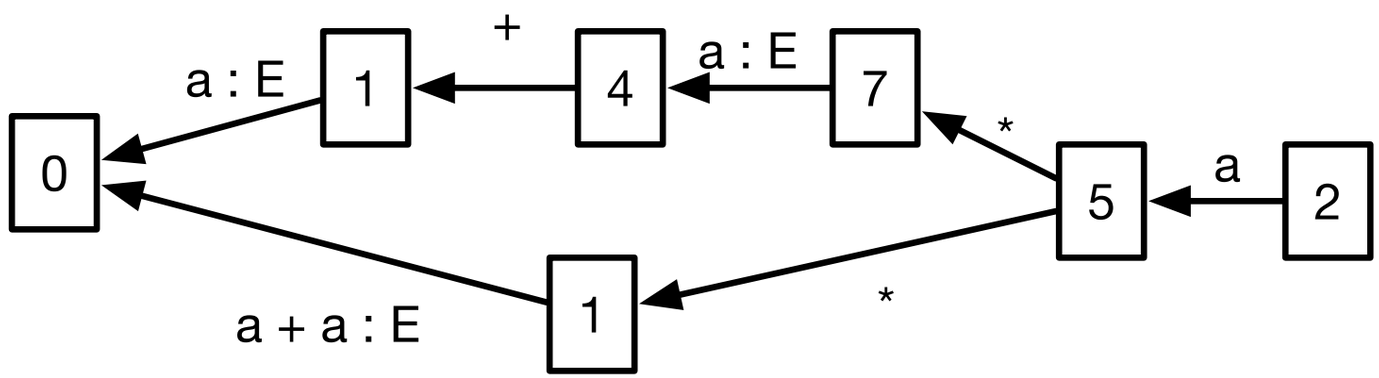
Generalized LR

Parsing

(0) S → E $
(1) E → E + E
(2) E → E * E
(3) E → a

State	Action				Goto	
	a	+	*	$	S	E
0	s2					1
1		s4	s5	s3		
2		r3	r3	r3		
3				acc		
4	s2					7
5	s2					6
6		s4/r2	s5/r2	r2		
7		s4/r1	s5/r1	r1		

synchronize

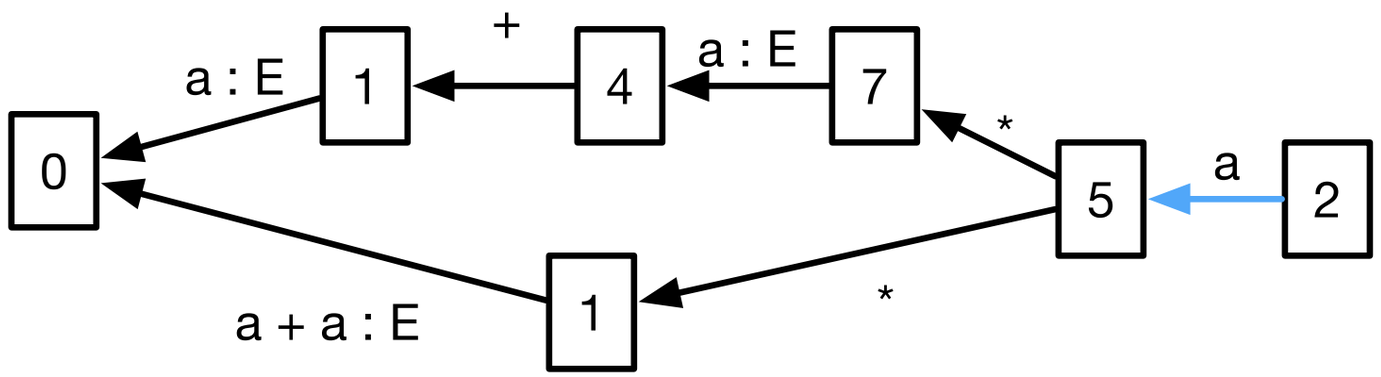


Generalized LR

Parsing

(0) S → E $
(1) E → E + E
(2) E → E * E
(3) E → a

State	Action				Goto	
	a	+	*	$	S	E
0	s2					1
1		s4	s5	s3		
2		r3	r3	r3		
3				acc		
4	s2					7
5	s2					6
6		s4/r2	s5/r2	r2		
7		s4/r1	s5/r1	r1		

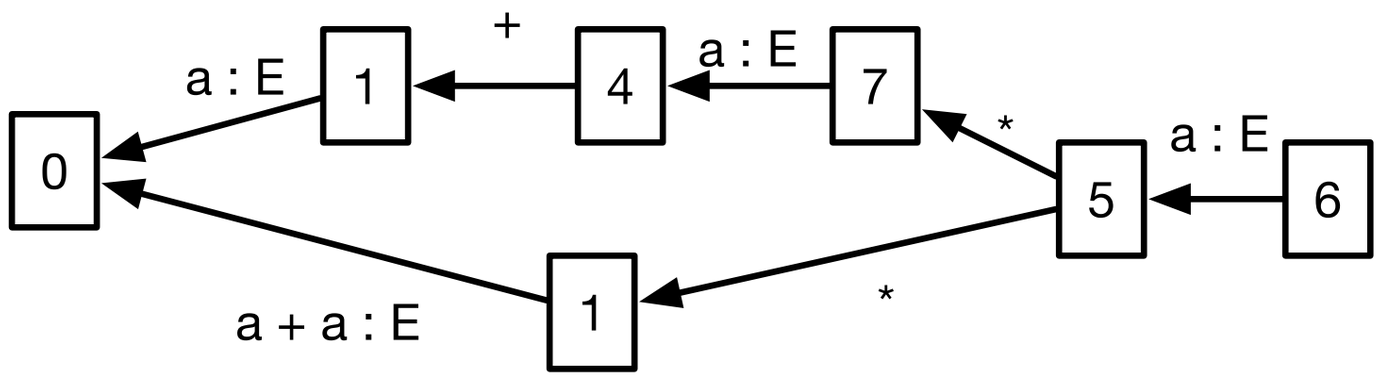


Generalized LR

Parsing

(0) S → E $
(1) E → E + E
(2) E → E * E
(3) E → a

State	Action				Goto	
	a	+	*	$	S	E
0	s2					1
1		s4	s5	s3		
2		r3	r3	r3		
3				acc		
4	s2					7
5	s2					6
6		s4/r2	s5/r2	r2		
7		s4/r1	s5/r1	r1		

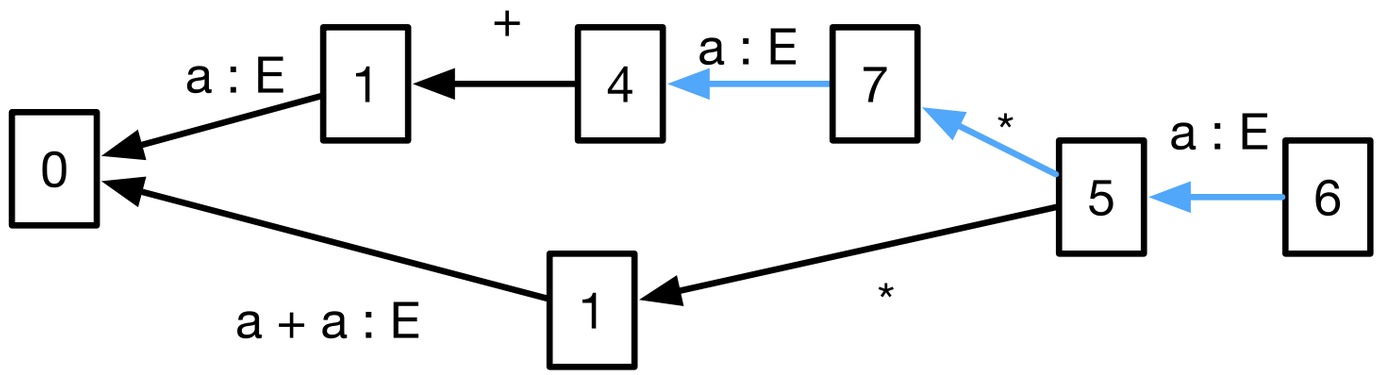


Generalized LR

Parsing

(0) S → E $
(1) E → E + E
(2) E → E * E
(3) E → a

State	Action				Goto	
	a	+	*	$	S	E
0	s2					1
1		s4	s5	s3		
2		r3	r3	r3		
3				acc		
4	s2					7
5	s2					6
6		s4/r2	s5/r2	r2		
7		s4/r1	s5/r1	r1		

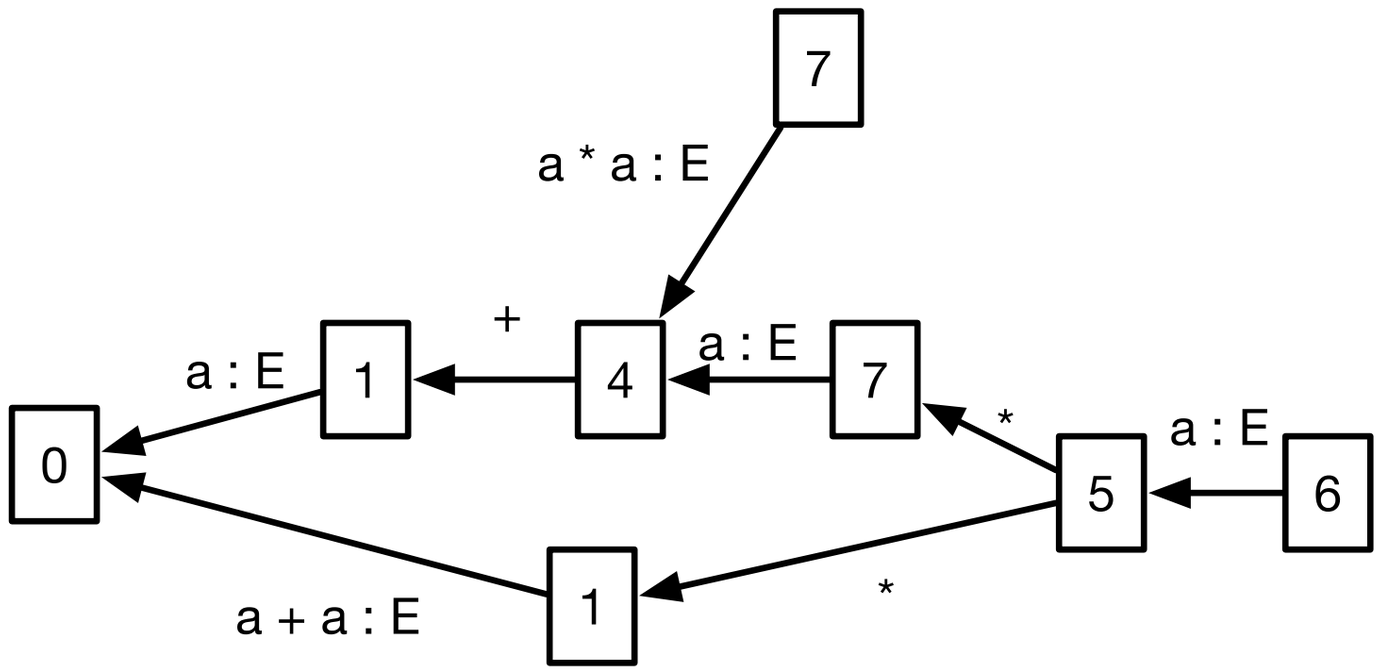


Generalized LR

Parsing

(0) S → E $
(1) E → E + E
(2) E → E * E
(3) E → a

State	Action				Goto	
	a	+	*	$	S	E
0	s2					1
1		s4	s5	s3		
2		r3	r3	r3		
3				acc		
4	s2					7
5	s2					6
6		s4/r2	s5/r2	r2		
7		s4/r1	s5/r1	r1		

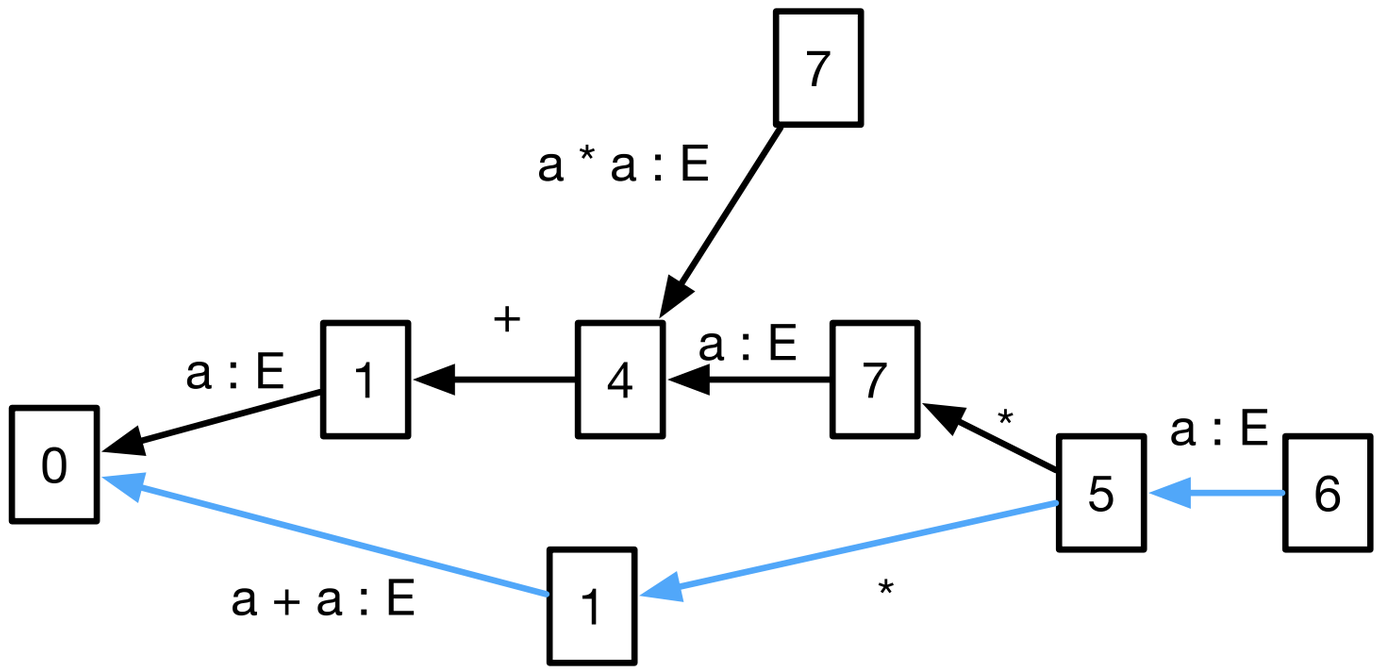


Generalized LR

Parsing

(0) S → E $
(1) E → E + E
(2) E → E * E
(3) E → a

State	Action				Goto	
	a	+	*	$	S	E
0	s2					1
1		s4	s5	s3		
2		r3	r3	r3		
3				acc		
4	s2					7
5	s2					6
6		s4/r2	s5/r2	r2		
7		s4/r1	s5/r1	r1		

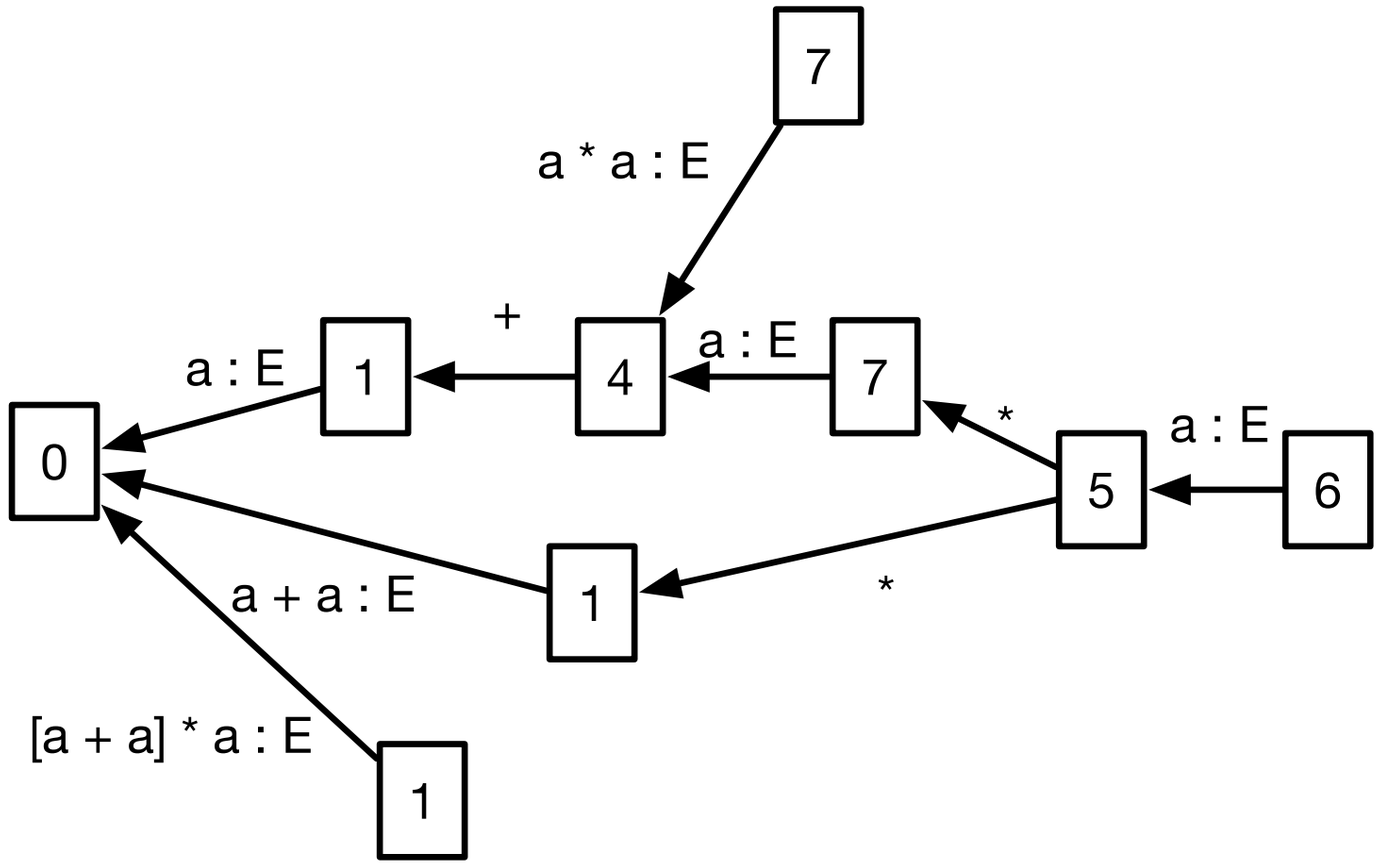


Generalized LR

Parsing

(0) S → E $
(1) E → E + E
(2) E → E * E
(3) E → a

State	Action				Goto	
	a	+	*	$	S	E
0	s2					1
1		s4	s5	s3		
2		r3	r3	r3		
3				acc		
4	s2					7
5	s2					6
6		s4/r2	s5/r2	r2		
7		s4/r1	s5/r1	r1		

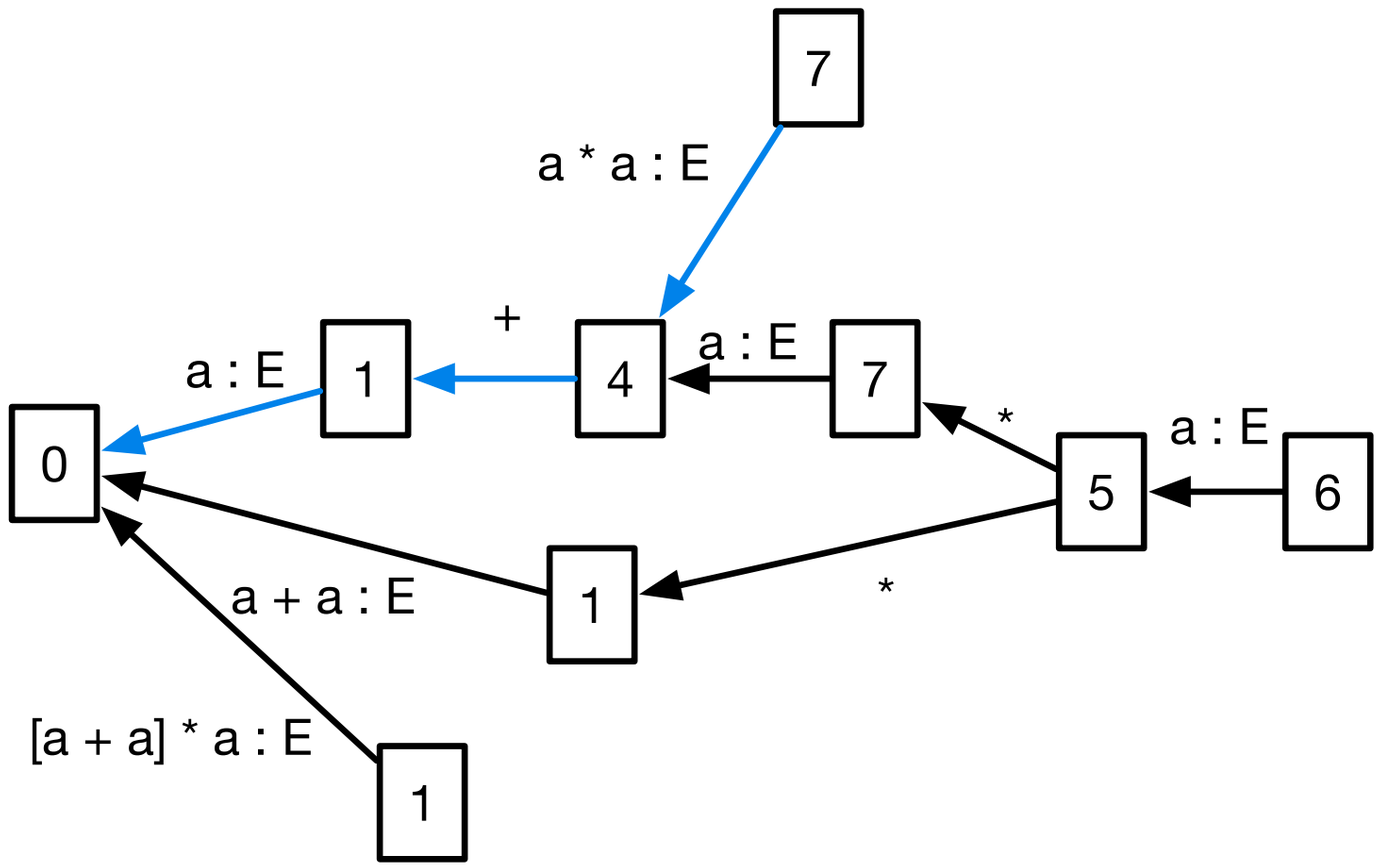


Generalized LR

Parsing

(0) S → E $
(1) E → E + E
(2) E → E * E
(3) E → a

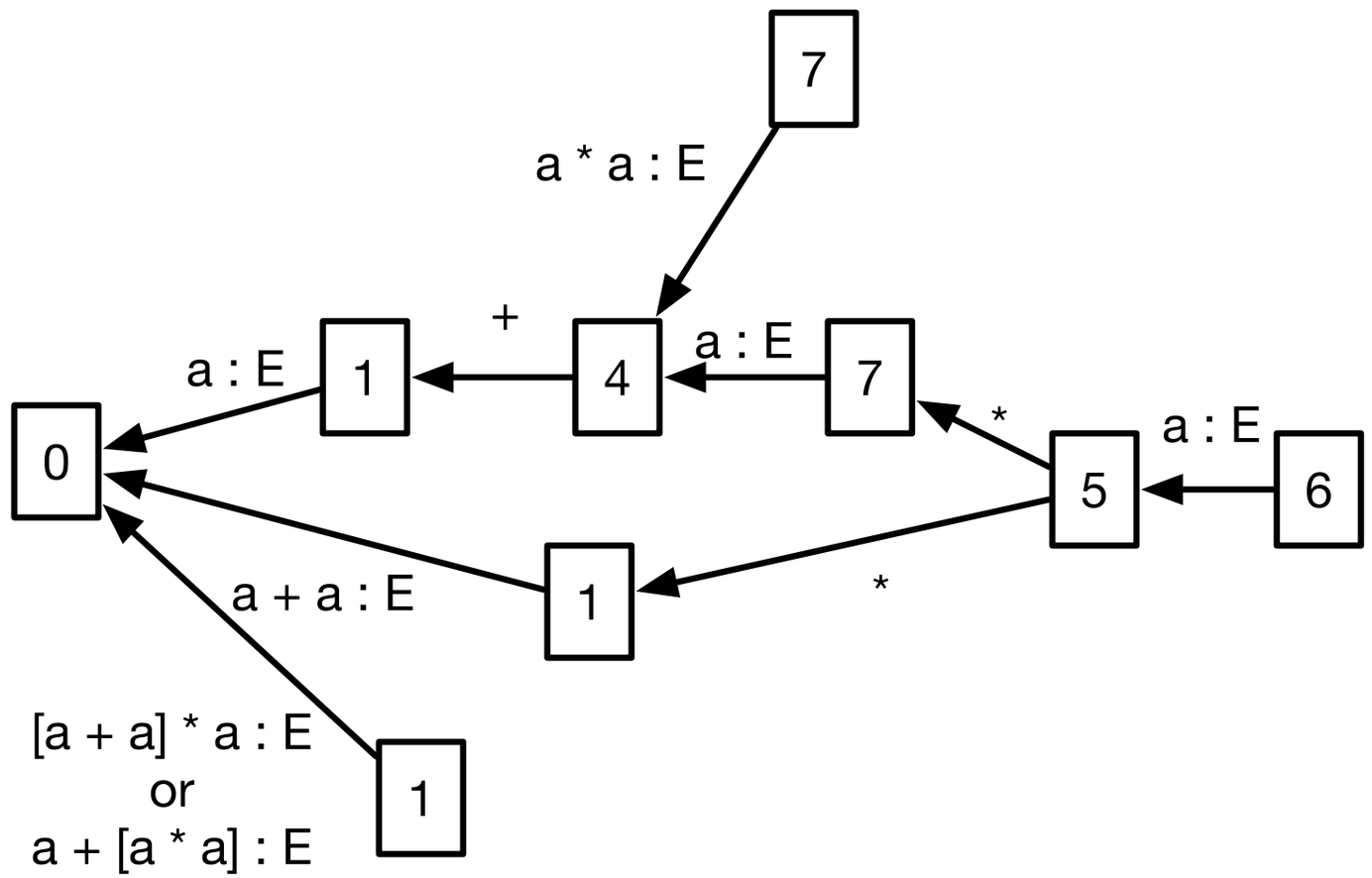
State	Action				Goto	
	a	+	*	$	S	E
0	s2					1
1		s4	s5	s3		
2		r3	r3	r3		
3				acc		
4	s2					7
5	s2					6
6		s4/r2	s5/r2	r2		
7		s4/r1	s5/r1	r1		



Generalized LR
Parsing

(0) S → E $
(1) E → E + E
(2) E → E * E
(3) E → a

State	Action				Goto	
	a	+	*	$	S	E
0	s2					1
1		s4	s5	s3		
2		r3	r3	r3		
3				acc		
4	s2					7
5	s2					6
6		s4/r2	s5/r2	r2		
7		s4/r1	s5/r1	r1		

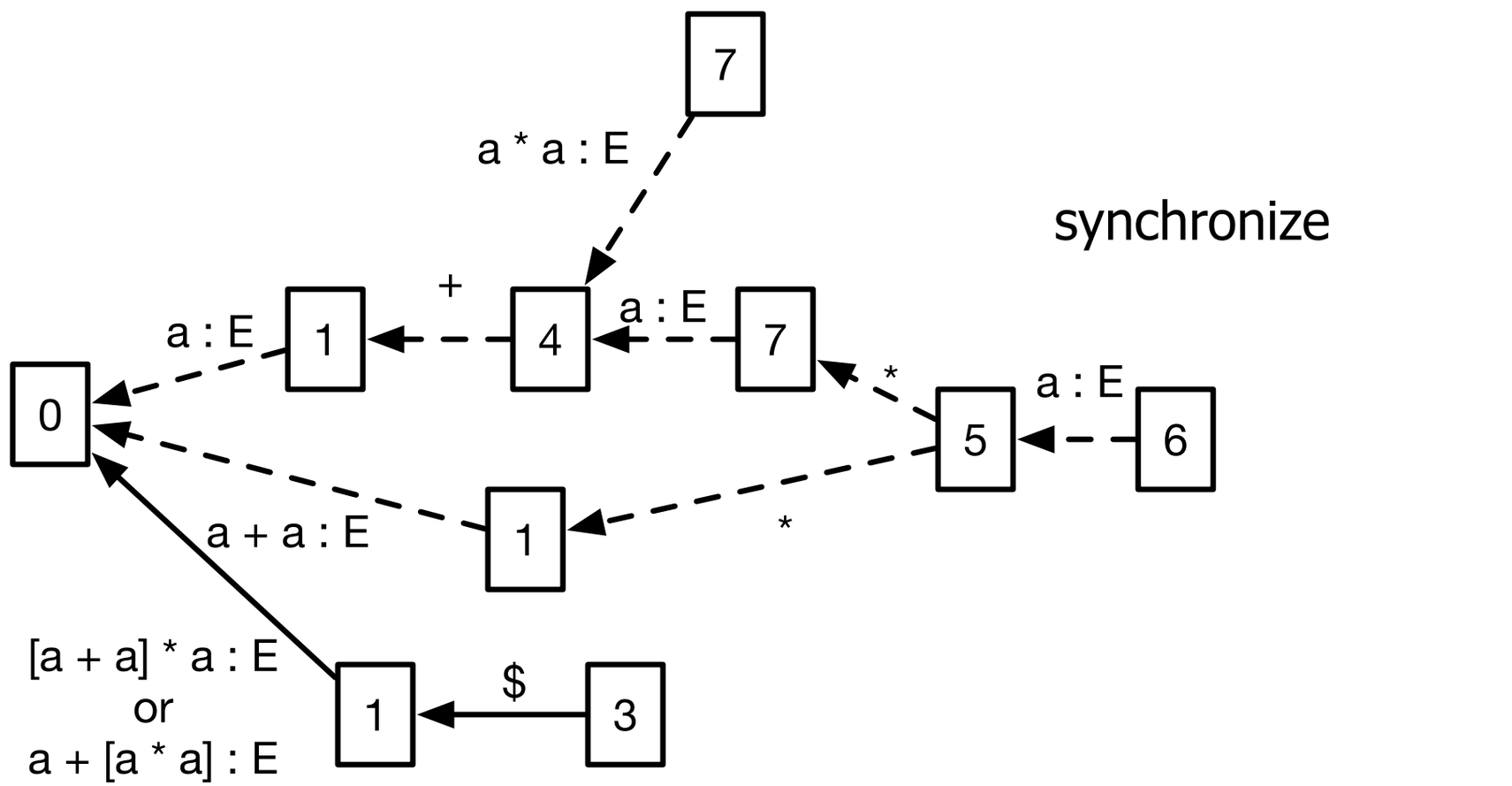


Generalized LR

Parsing

(0) S → E $
(1) E → E + E
(2) E → E * E
(3) E → a

State	Action				Goto	
	a	+	*	$	S	E
0	s2					1
1		s4	s5	s3		
2		r3	r3	r3		
3				acc		
4	s2					7
5	s2					6
6		s4/r2	s5/r2	r2		
7		s4/r1	s5/r1	r1		



accept with trees on the link to initial state

V

Scannerless Generalized-LR Parsing

Scannerless Generalized LR

- Integrates scanning + parsing into a single GLR algorithm.
- Normalization separates lexical and context-free symbols.
- Crucial for avoiding conflicts when composing languages.
- Introduces lexical ambiguities. Solution:
  - Follow Restrictions (implement longest match).
  - Reject Rules (implement reserved keywords).

Scannerless Generalized LR

Normalization

```
context-free syntax

  Exp.Add    = <<Exp> + <Exp>> {left}
  Exp.Inc    = <<Exp>++>
  Exp.ID     = ID

lexical syntax

  ID         = [a-zA-Z] IDRest*
  IDRest     = [a-zA-Z0-9]
  LAYOUT     = [\ \t\n\r]
  ID         = "let" {reject}
```

Scannerless Generalized LR

Normalization

Separate lexical from context-free symbols and separate symbols in context-free syntax by optional layout.

```
context-free syntax

  Exp.Add    = <<Exp> + <Exp>> {left}
  Exp.Inc    = <<Exp>++>
  Exp.ID     = ID

lexical syntax

  ID         = [a-zA-Z] IDRest*
  IDRest     = [a-zA-Z0-9]
  LAYOUT     = [\ \t\n\r]
  ID         = "let" {reject}
```

```
Exp-CF.Add = Exp-CF LAYOUT?-CF "+" LAYOUT?-CF Exp-CF {left}
Exp-CF.Inc = Exp-CF LAYOUT?-CF "++"
Exp-CF.ID = ID-CF
ID-LEX = [a-zA-Z] IDRest*-LEX
IDRest*-LEX = [a-zA-Z0-9]
ID-LEX = "let" {reject}
LAYOUT-CF = LAYOUT-LEX
```

Scannerless Generalized LR

Normalization

Create injections from lexical to context-free symbols

```
context-free syntax

  Exp.Add    = <<Exp> + <Exp>> {left}
  Exp.Inc    = <<Exp>++>
  Exp.ID     = ID

lexical syntax

  ID         = [a-zA-Z] IDRest*
  IDRest     = [a-zA-Z0-9]
  LAYOUT     = [\ \t\n\r]
  ID         = "let" {reject}
```

```
LAYOUT-CF = LAYOUT-LEX
IDRest-CF = IDRest-LEX
IDRest*-CF = IDRest*-LEX
ID-CF = ID-LEX
```

Scannerless Generalized LR

Normalization

Normalize literals symbols and character classes.

```
context-free syntax

  Exp.Add    = <<Exp> + <Exp>> {left}
  Exp.Inc    = <<Exp>++>
  Exp.ID     = ID

lexical syntax

  ID         = [a-zA-Z] IDRest*
  IDRest     = [a-zA-Z0-9]
  LAYOUT     = [\ \t\n\r]
  ID         = "let" {reject}
```

```
"+" = [\43]
"++" = [\43] [\43]
"let" = [\108] [\101] [\116]
ID-LEX       = [\65-\90\97-\122] IDRest*-LEX
IDRest*-LEX = [\48-\57\65-\90\97-\122]
LAYOUT-LEX  = [\9-\10\13\32]
```

Scannerless Generalized LR

Normalization

Normalize regular expressions.

```
context-free syntax

  Exp.Add    = <<Exp> + <Exp>> {left}
  Exp.Inc    = <<Exp>++>
  Exp.ID     = ID

lexical syntax

  ID         = [a-zA-Z] IDRest*
  IDRest     = [a-zA-Z0-9]
  LAYOUT     = [\ \t\n\r]
  ID         = "let" {reject}
```

```
LAYOUT?-CF = LAYOUT-CF
LAYOUT?-CF =
IDRest+-LEX = IDRest-LEX
IDRest+-LEX = IDRest+-LEX IDRest-LEX
IDRest*-LEX =
IDRest*-LEX = IDRest+-LEX
IDRest+-CF = IDRest+-LEX
```

Scannerless Generalized LR

Normalization

Define extra rules for the start symbols.

```
context-free start-symbols
  Exp

<START> = LAYOUT?-CF Exp-CF LAYOUT?-CF
<Start> = <START> [\256]
```

Scannerless Generalized LR

Parse Table Generation

- Generation is based on SLR(1) item-sets.
- Uses character classes instead of tokens.
- Follow sets and goto actions are calculated for productions instead of non-terminals.

Scannerless Generalized LR

Lexical Disambiguation

- A reject rule is of the form:

```
ID = "let" {reject}
```

- When SGLR does a reduction with a reject rule, it marks the link as rejected.

- Further action on a stack is forbidden whenever all links to it are rejected.

- Follow restrictions are implemented as filters on the follow sets of productions.

Generalized LR

Context-free Disambiguation

- Priority rules: applied at parse table generation.

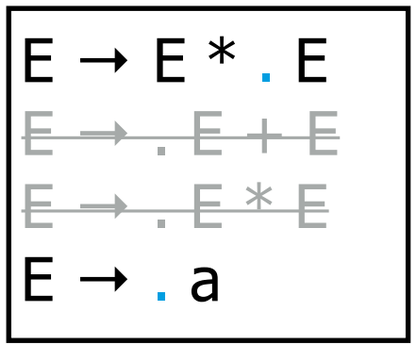
E → E * . E
~~E → . E + E~~
E → . E * E
E → . a

multiplication has higher priority than addition!

Generalized LR

Context-free Disambiguation

- Priority rules: applied at parse table generation.

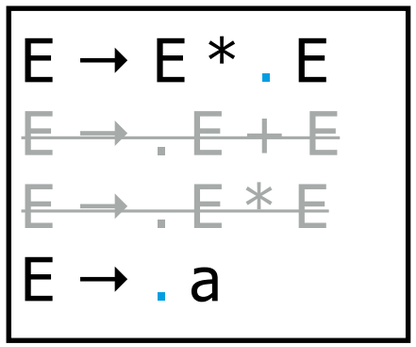


multiplication is left associative!

Generalized LR

Context-free Disambiguation

- Priority rules: applied at parse table generation.



multiplication is left associative!

- Disambiguation filters: applied after parsing (prefer and avoid).

VI

Summary

Summary
lessons learned

Summary
lessons learned

How can we generate LR parse tables?

- items, closure, goto

Summary
lessons learned

How can we generate LR parse tables?

- items, closure, goto

How can we improve LR(0) parse table generation?

- SLR: consider FOLLOW sets to avoid shift-reduce conflicts
- LR(1): consider look-ahead in states
- LALR(1): unify LR(1) states to reduce state space

Summary
lessons learned

How can we generate LR parse tables?

- items, closure, goto

How can we improve LR(0) parse table generation?

- SLR: consider FOLLOW sets to avoid shift-reduce conflicts
- LR(1): consider look-ahead in states
- LALR(1): unify LR(1) states to reduce state space

How can we handle conflicts in the parse table?

- generalized parsing - supports all class of context free grammars
- scannerless generalized LR - allow for proper language composition.

Literature

learn more

Literature

learn more

LR parsing

Andrew W. Appel, Jens Palsberg: Modern Compiler Implementation in Java, 2nd edition. 2002

Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, Monica S. Lam: Compilers: Principles, Techniques, and Tools, 2nd edition. 2006

Literature

learn more

LR parsing

Andrew W. Appel, Jens Palsberg: Modern Compiler Implementation in Java, 2nd edition. 2002

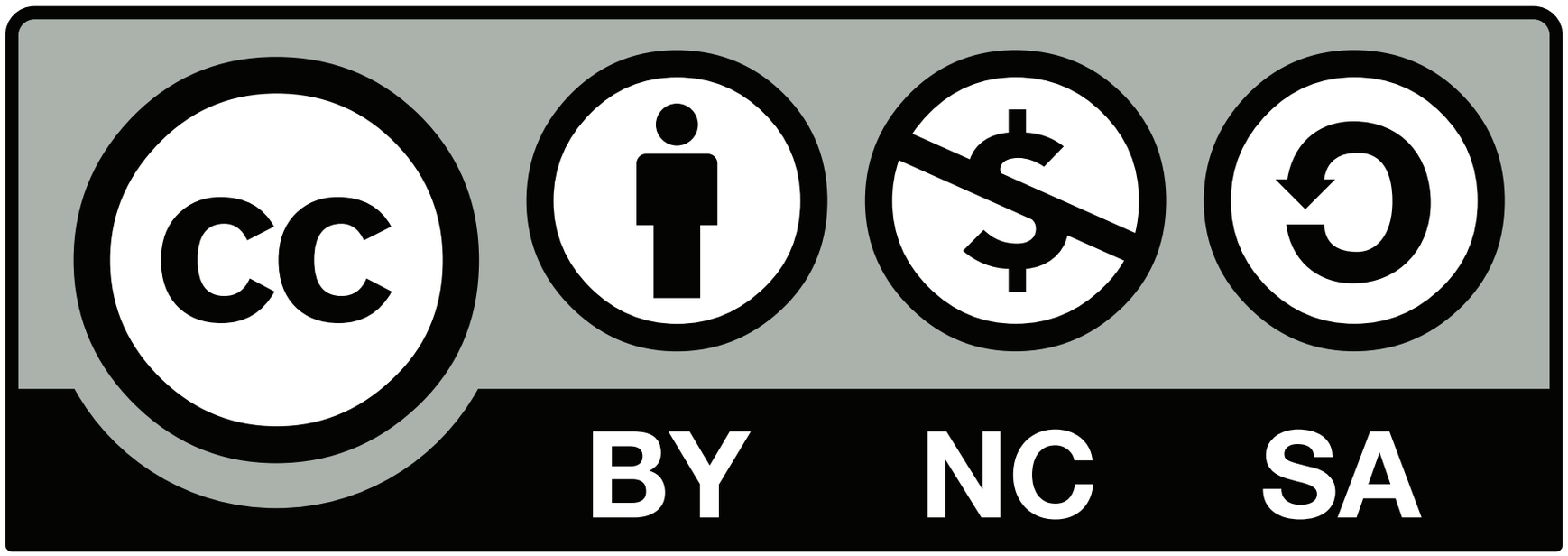
Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, Monica S. Lam: Compilers: Principles, Techniques, and Tools, 2nd edition. 2006

Generalised LR parsing

Eelco Visser: Syntax Definition for Language Prototyping. PhD thesis 1997

M.G.J. van den Brand, J. Scheerder, J.J. Vinju, and E. Visser: Disambiguation Filters for Scannerless Generalized LR Parsers. CC 2002

copyrights


BY
NC
SA

Pictures

copyrights

Slide 1:

Book Scanner by Ben Woosley, some rights reserved

Slide 19:

Ostsee by Mario Thiel, some rights reserved