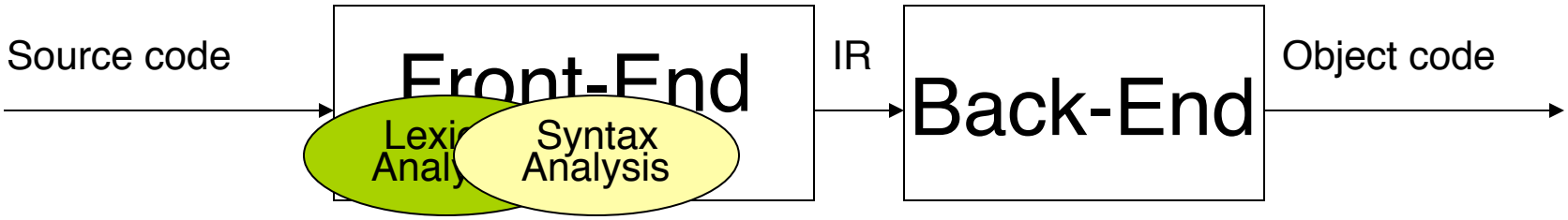
Compiler Design

Lecture 6:
Syntax Analysis
Bottom-Up Parsing (part IV)

Dr. Momtazi
momtazi@aut.ac.ir

based on the slides of the course book

Parsing (Syntax Analysis)



- Syntax Analysis:
  - Derivation and parse trees ✓
  - Top-down parsing ✓
  - **Bottom-up parsing**

Outline

- Introduction
- Shift-reduce Parsing
- **LR Parsing**
  - LR(0)
  - LR(1)
  - **SLR**
  - LALR
- Ambiguity
- Error-handling

SLR(1)

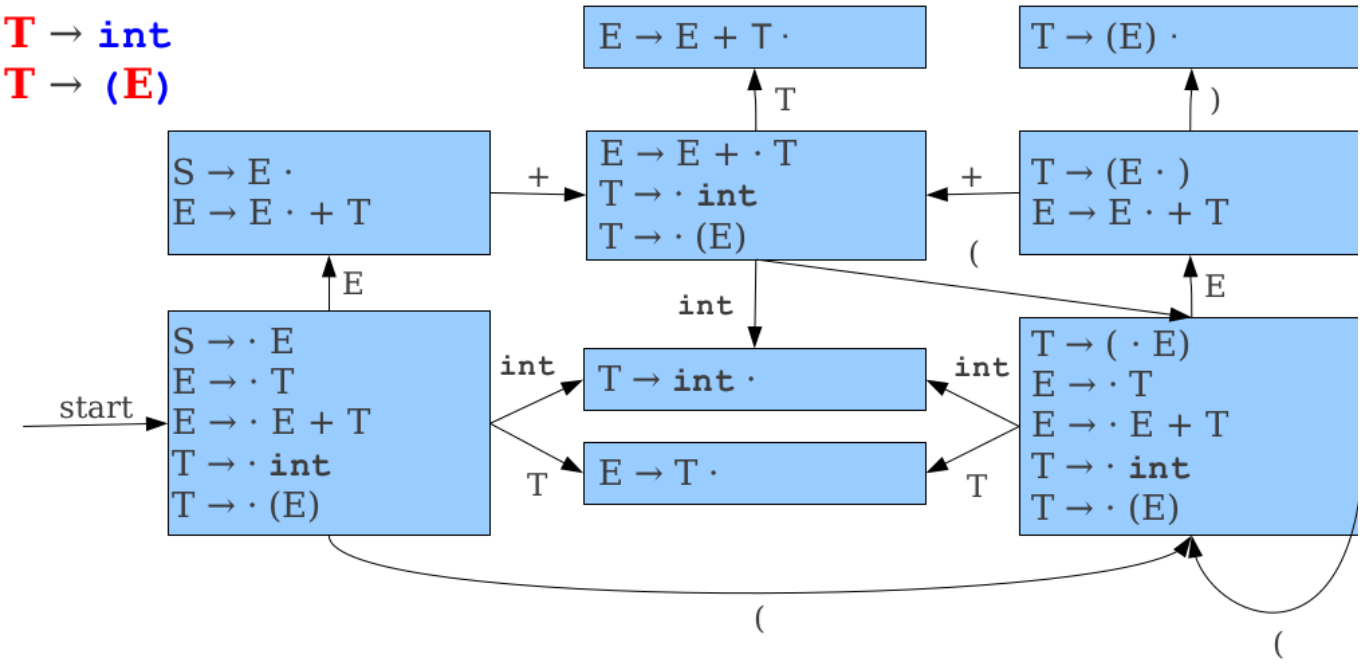
- Simple LR(1)
- Minor modification to LR(0) automaton that uses lookahead to avoid shift/reduce conflicts.
- Idea: Only reduce $A \rightarrow \omega$ if the next token t is in FOLLOW(A).
- Automaton identical to LR(0) automaton; only change is when we choose to reduce.

SLR(1) Parsing

$S \rightarrow E$
$E \rightarrow T$
$E \rightarrow E + T$
$T \rightarrow \texttt{int}$
$T \rightarrow (E)$

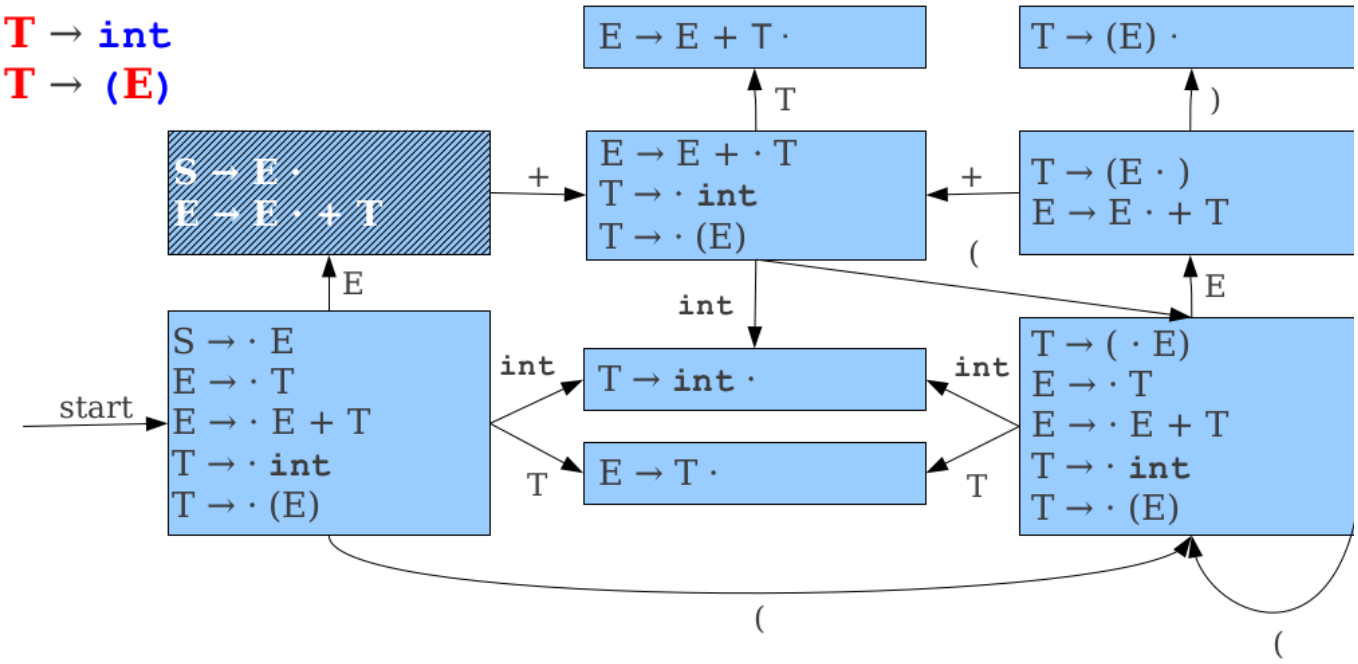
SLR(1) Parsing

S → E
E → T
E → E + T
T → int
T → (E)

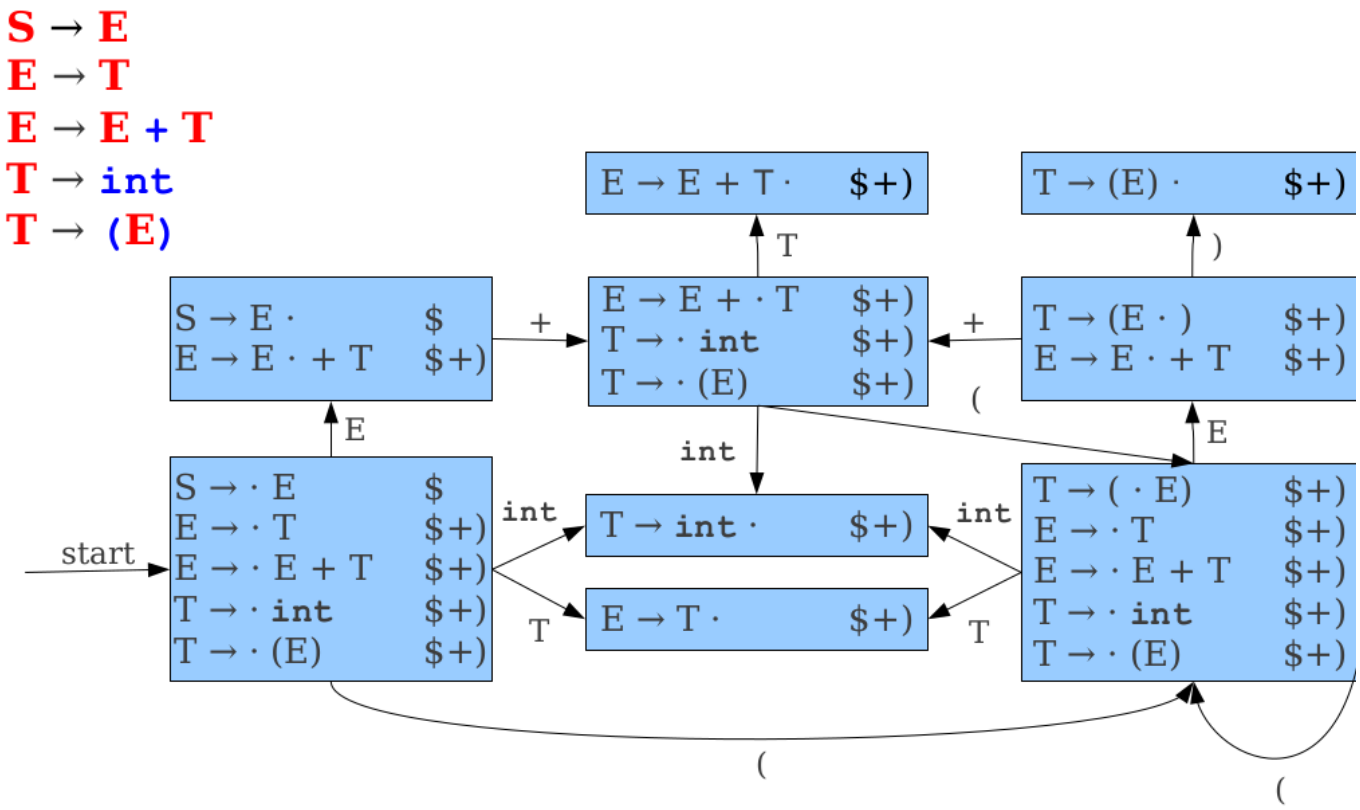


SLR(1) Parsing

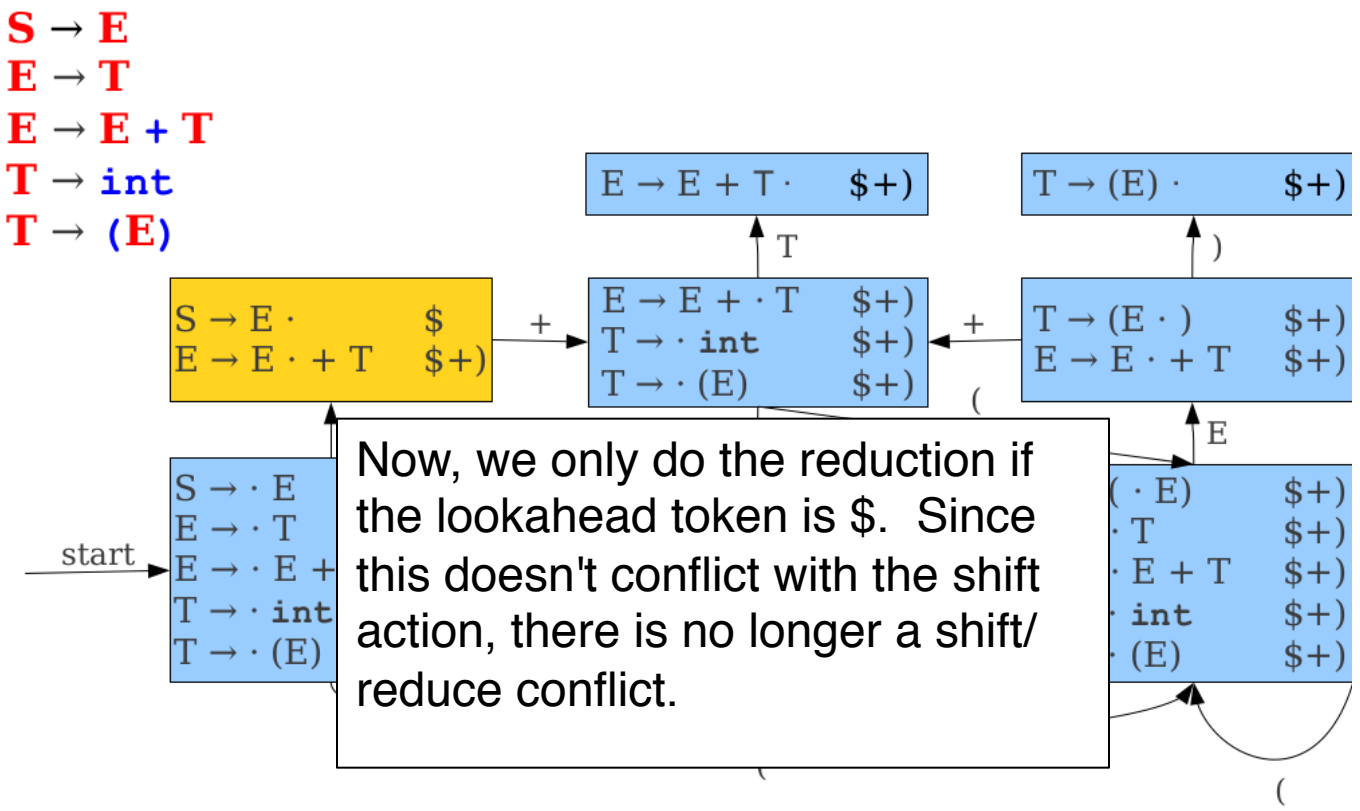
S → E
E → T
E → E + T
T → int
T → (E)



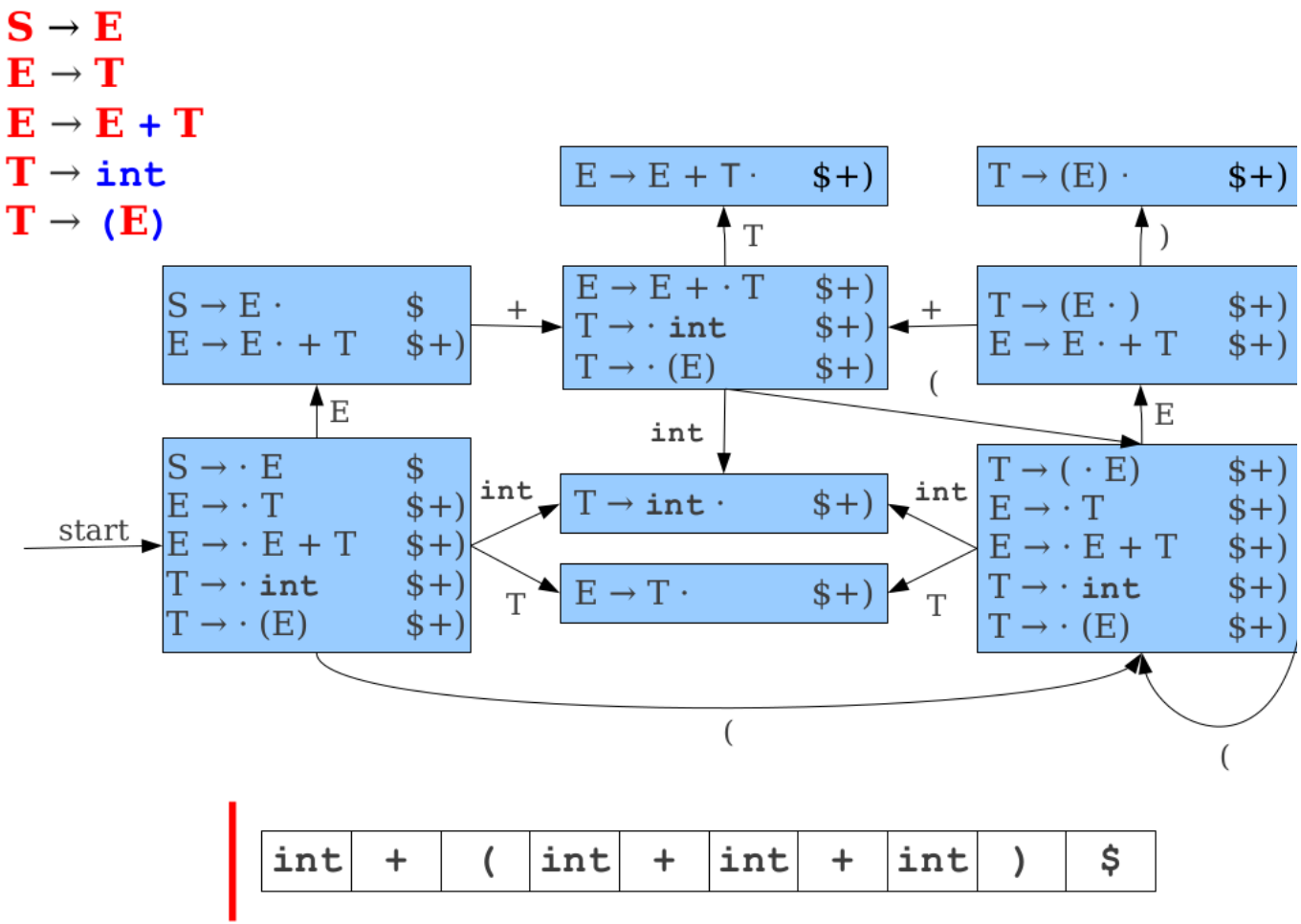
SLR(1) Parsing



SLR(1) Parsing

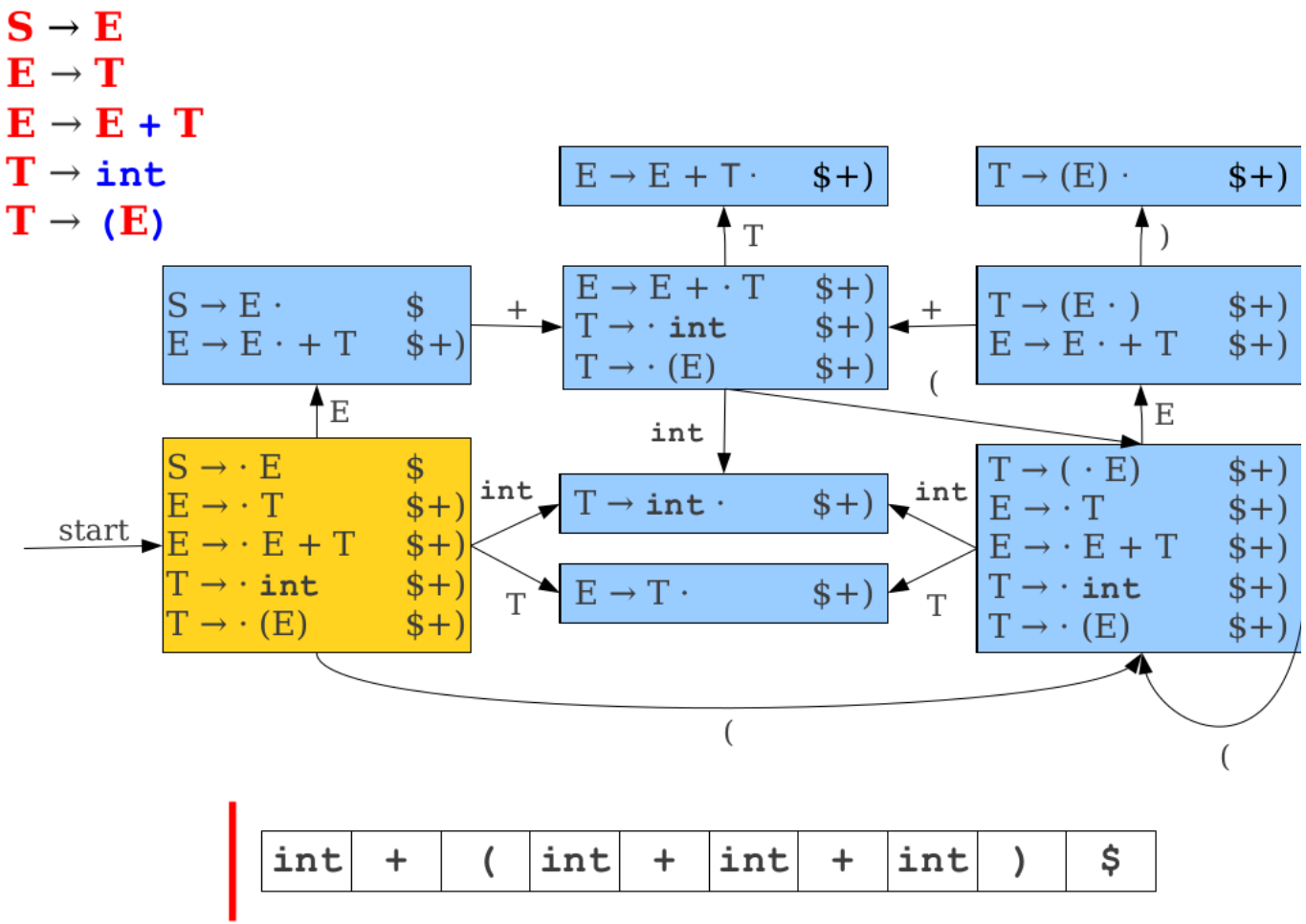


SLR(1) Parsing



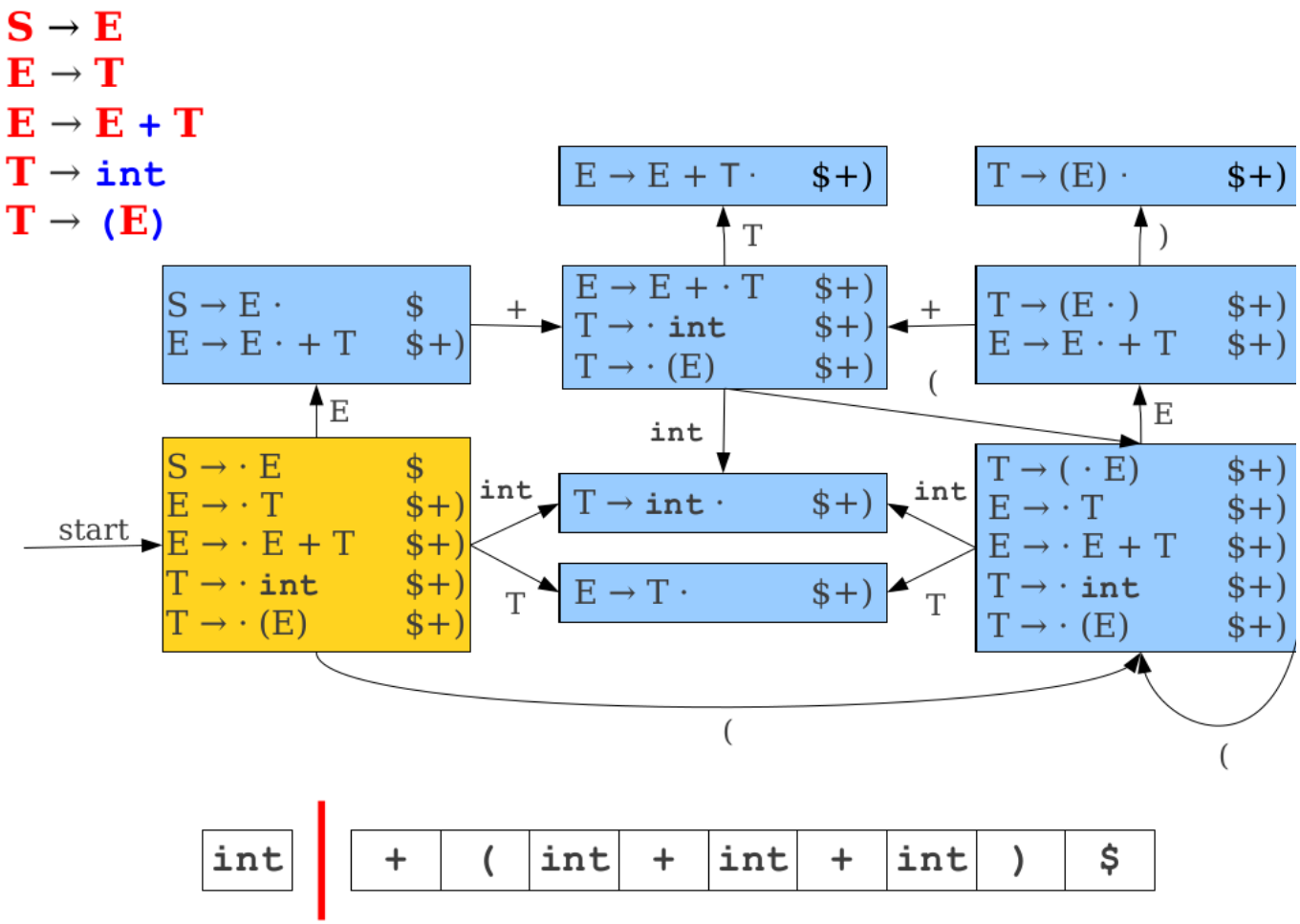
int	+	(	int	+	int	+	int	)	$

SLR(1) Parsing

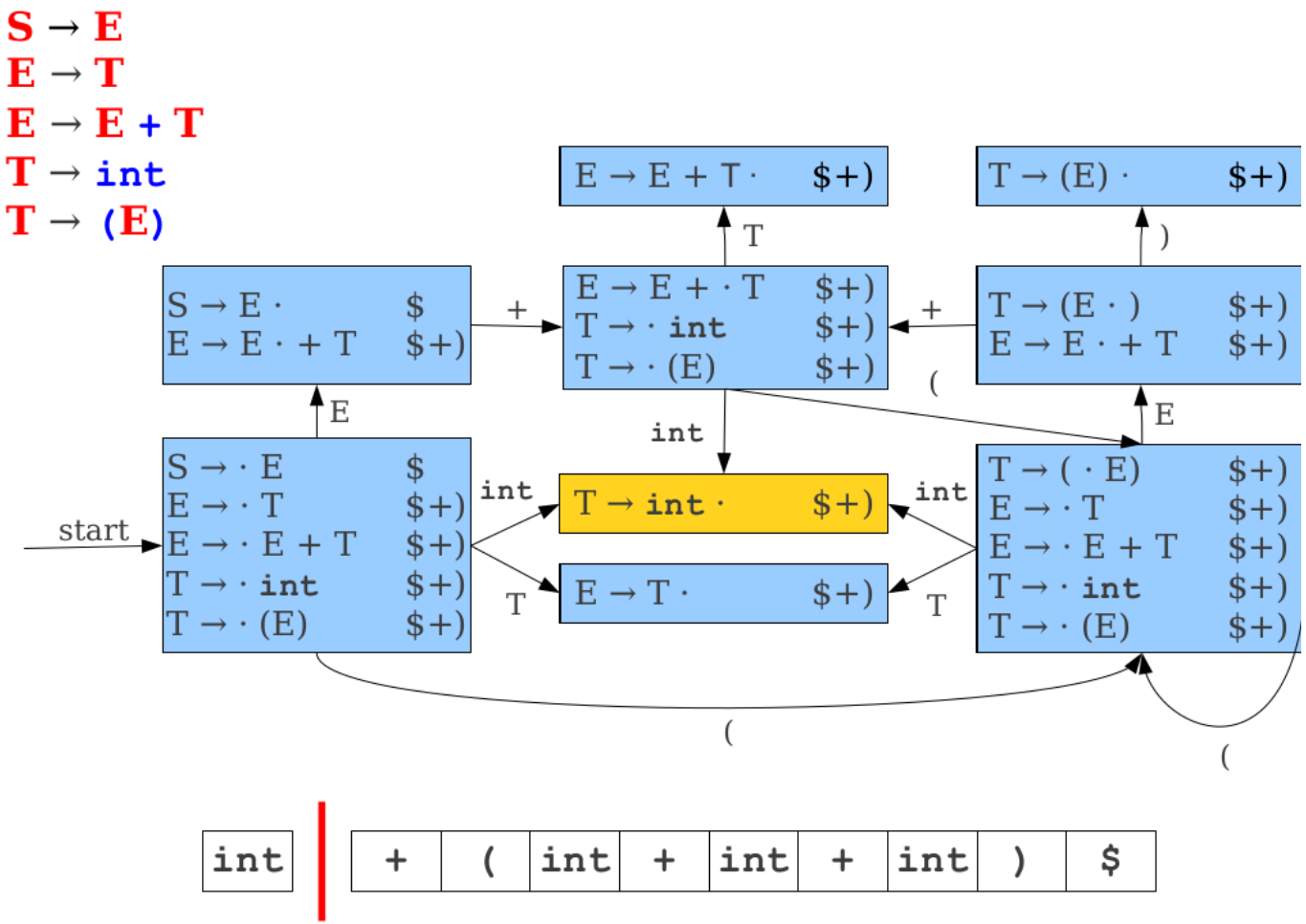


int	+	(	int	+	int	+	int	)	$

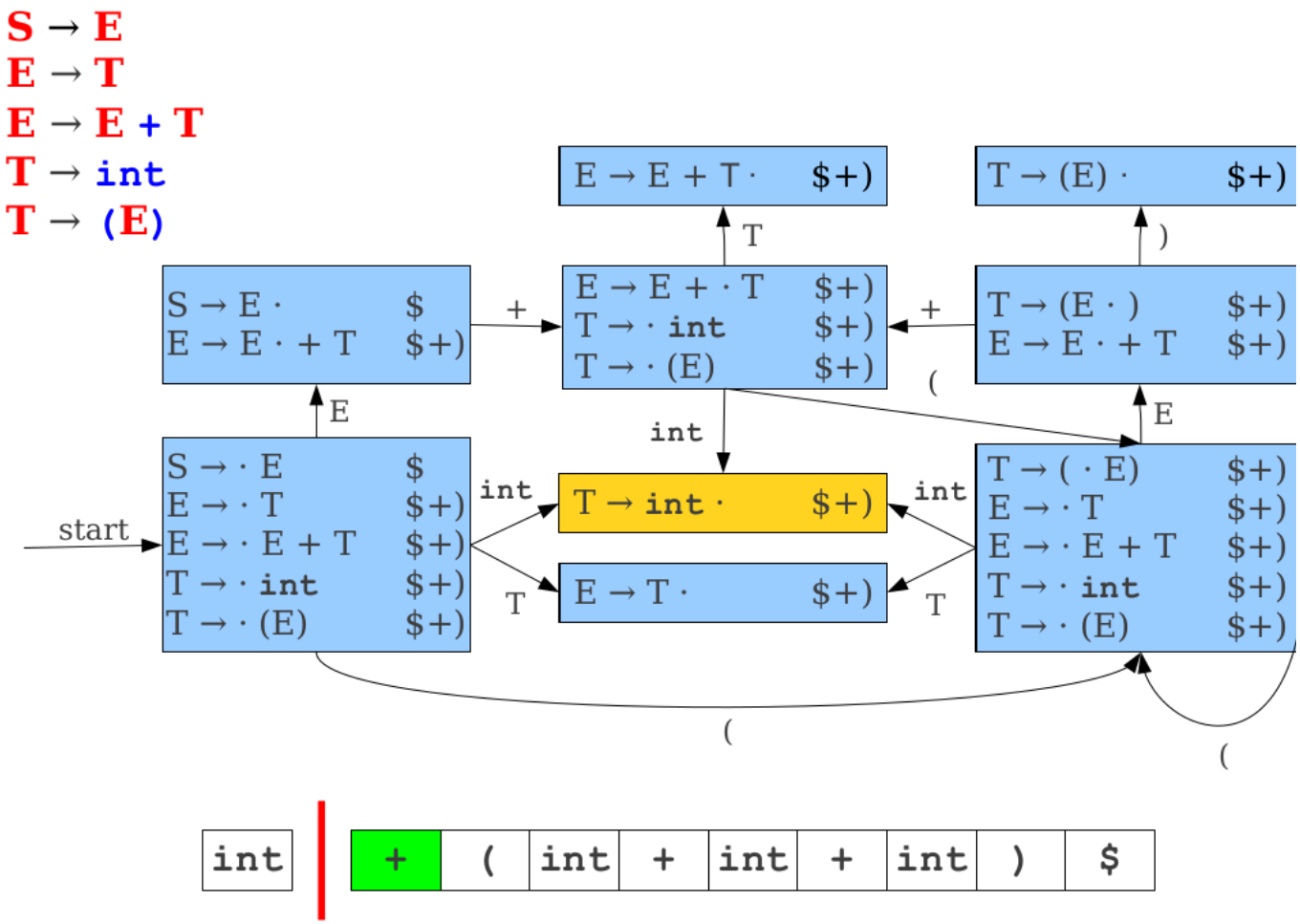
SLR(1) Parsing



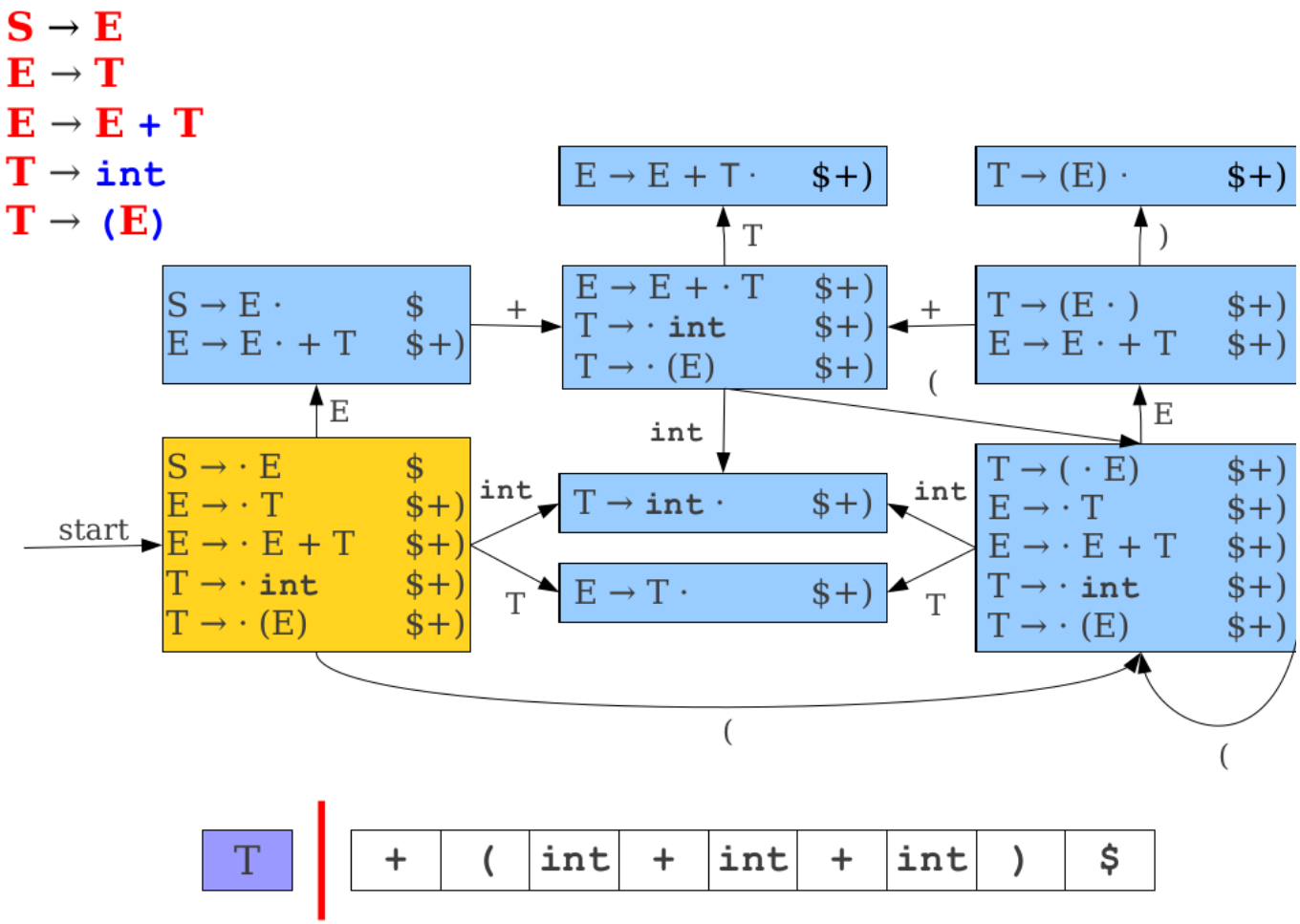
SLR(1) Parsing



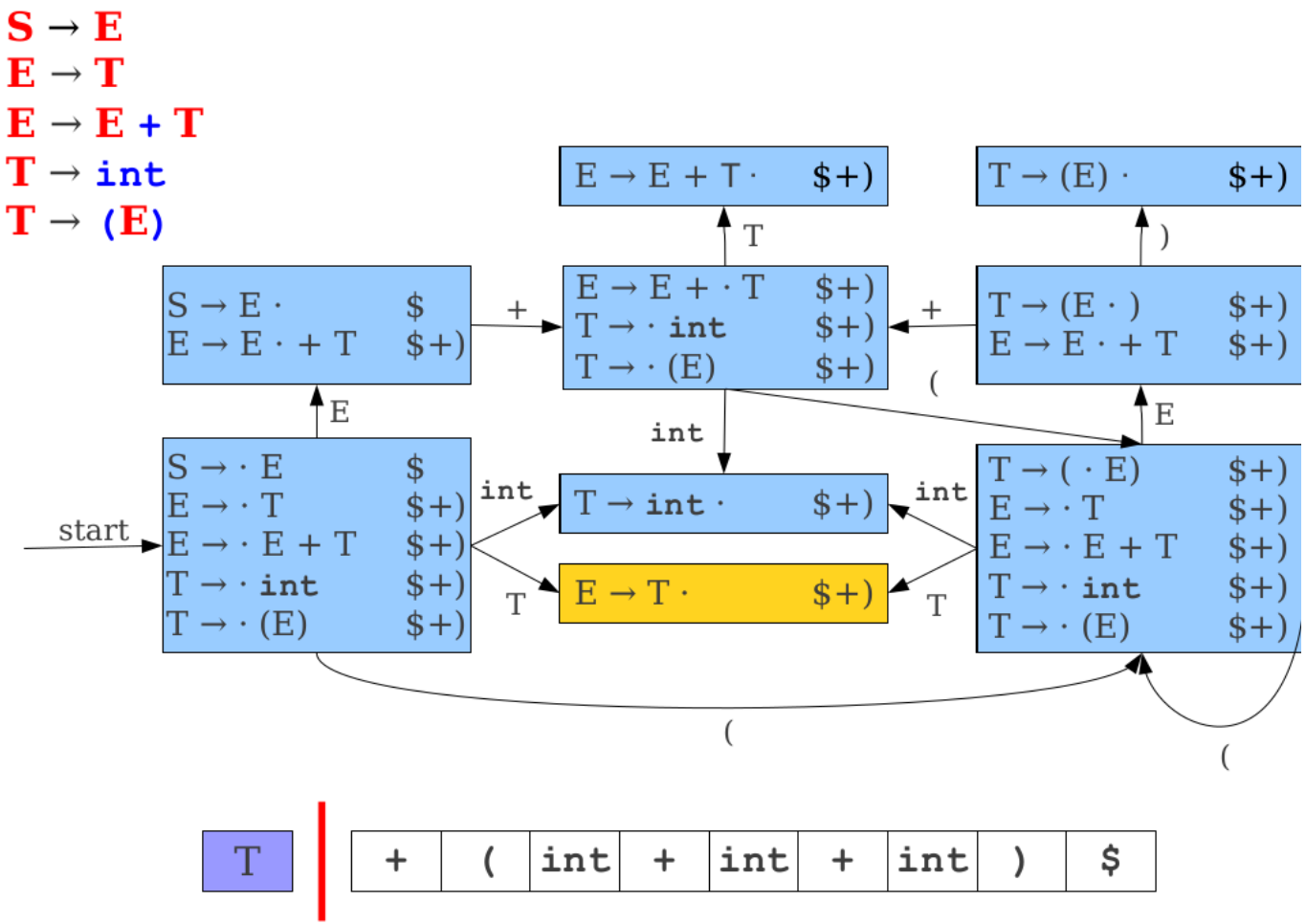
SLR(1) Parsing



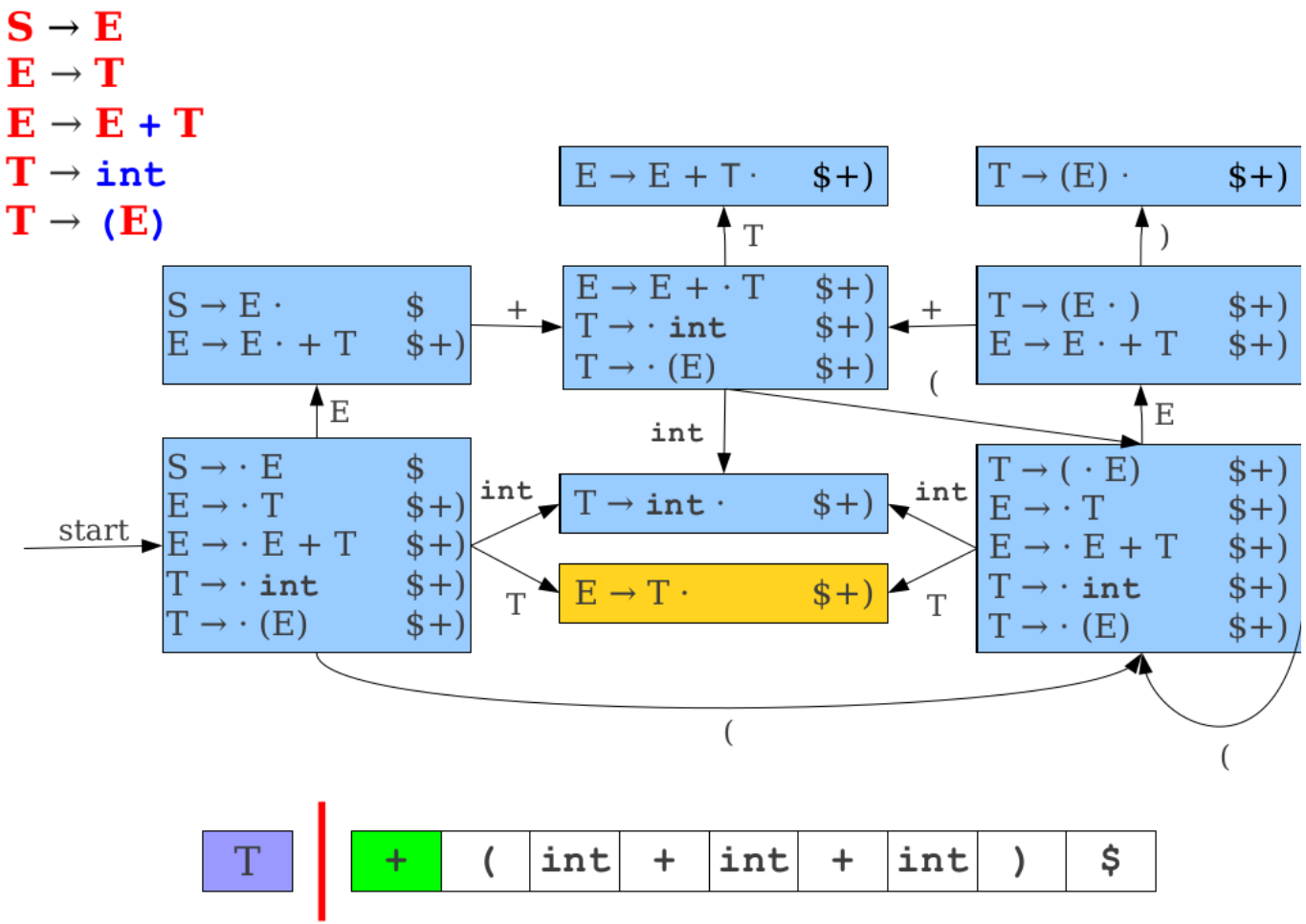
SLR(1) Parsing



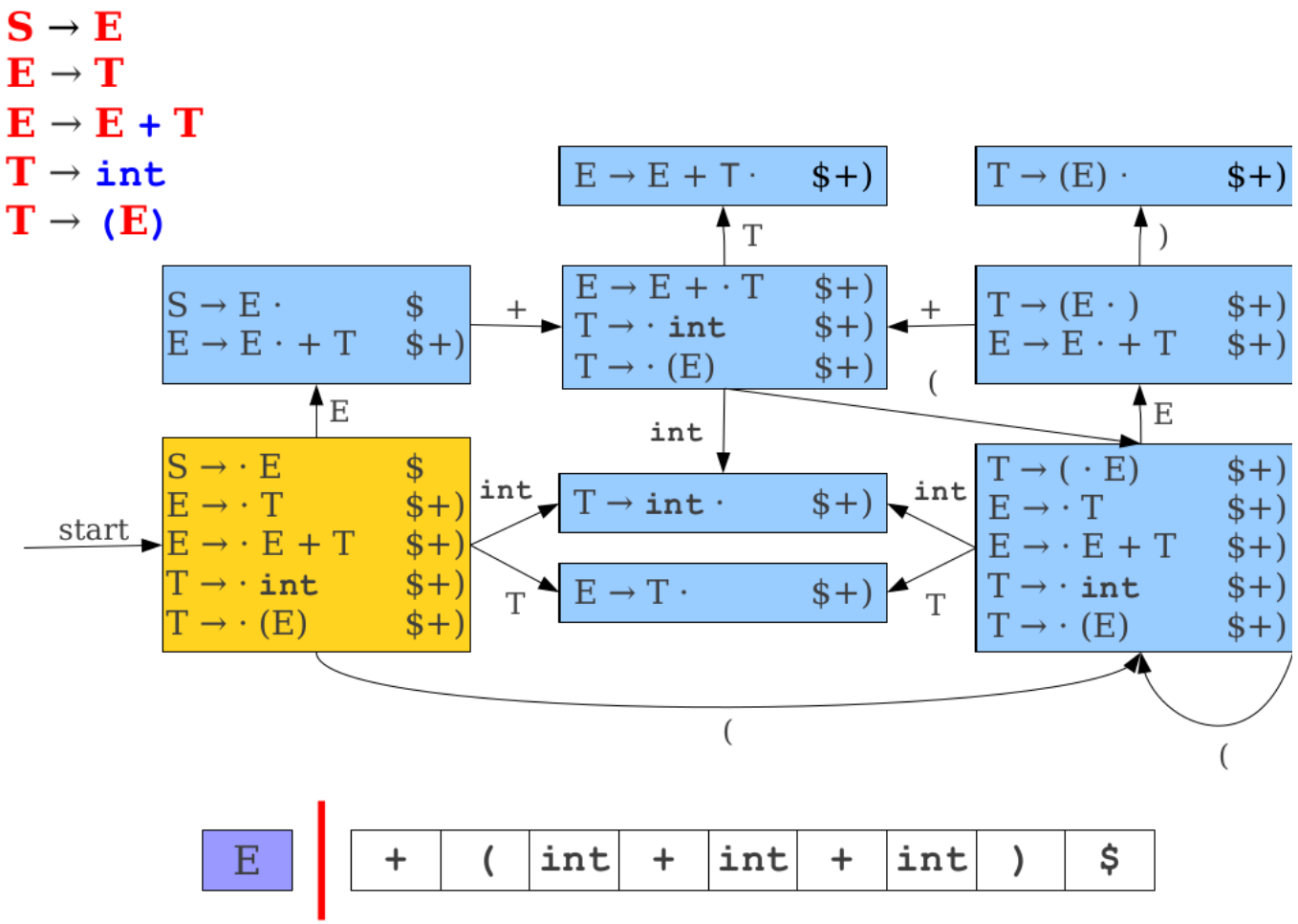
SLR(1) Parsing



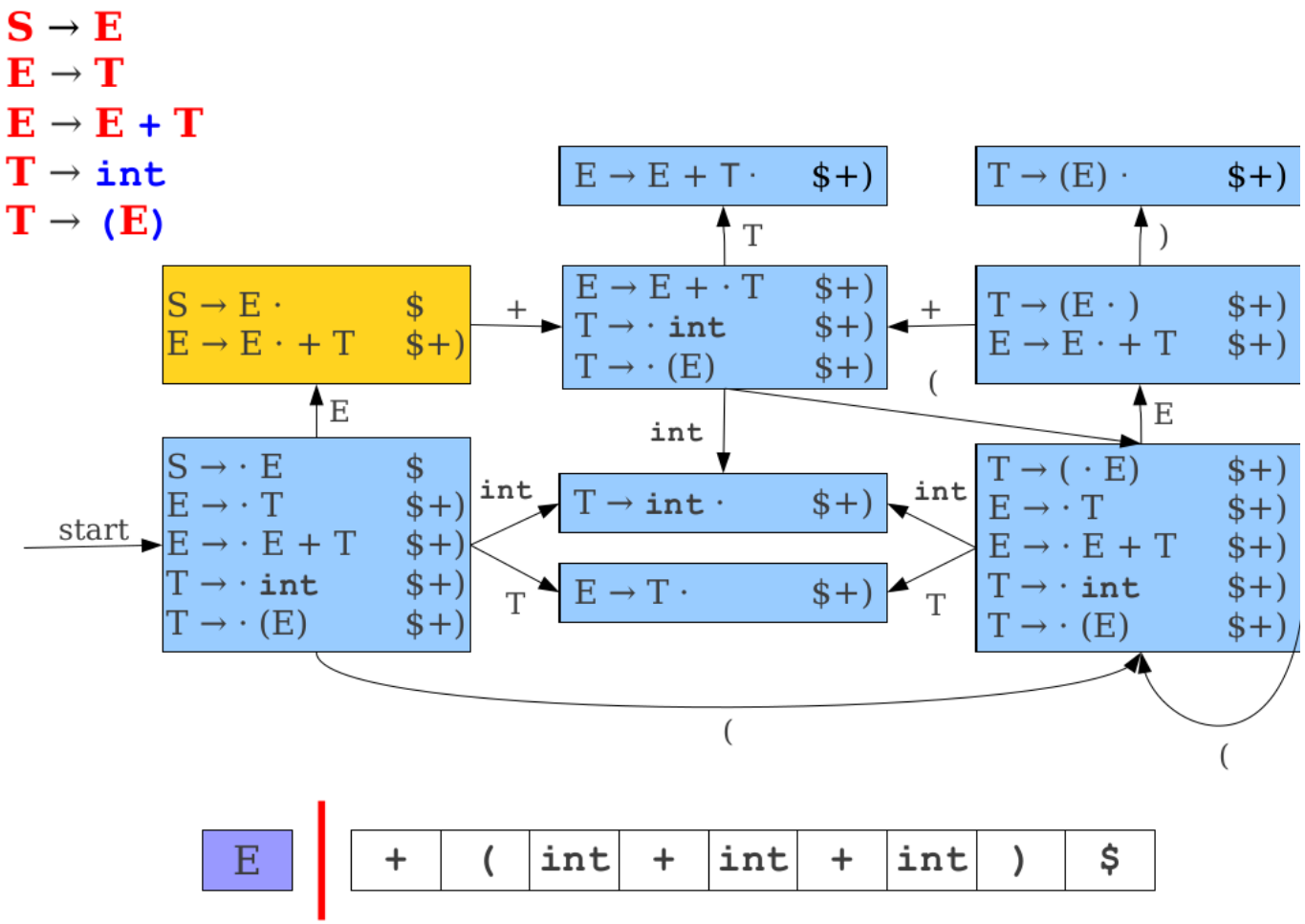
SLR(1) Parsing



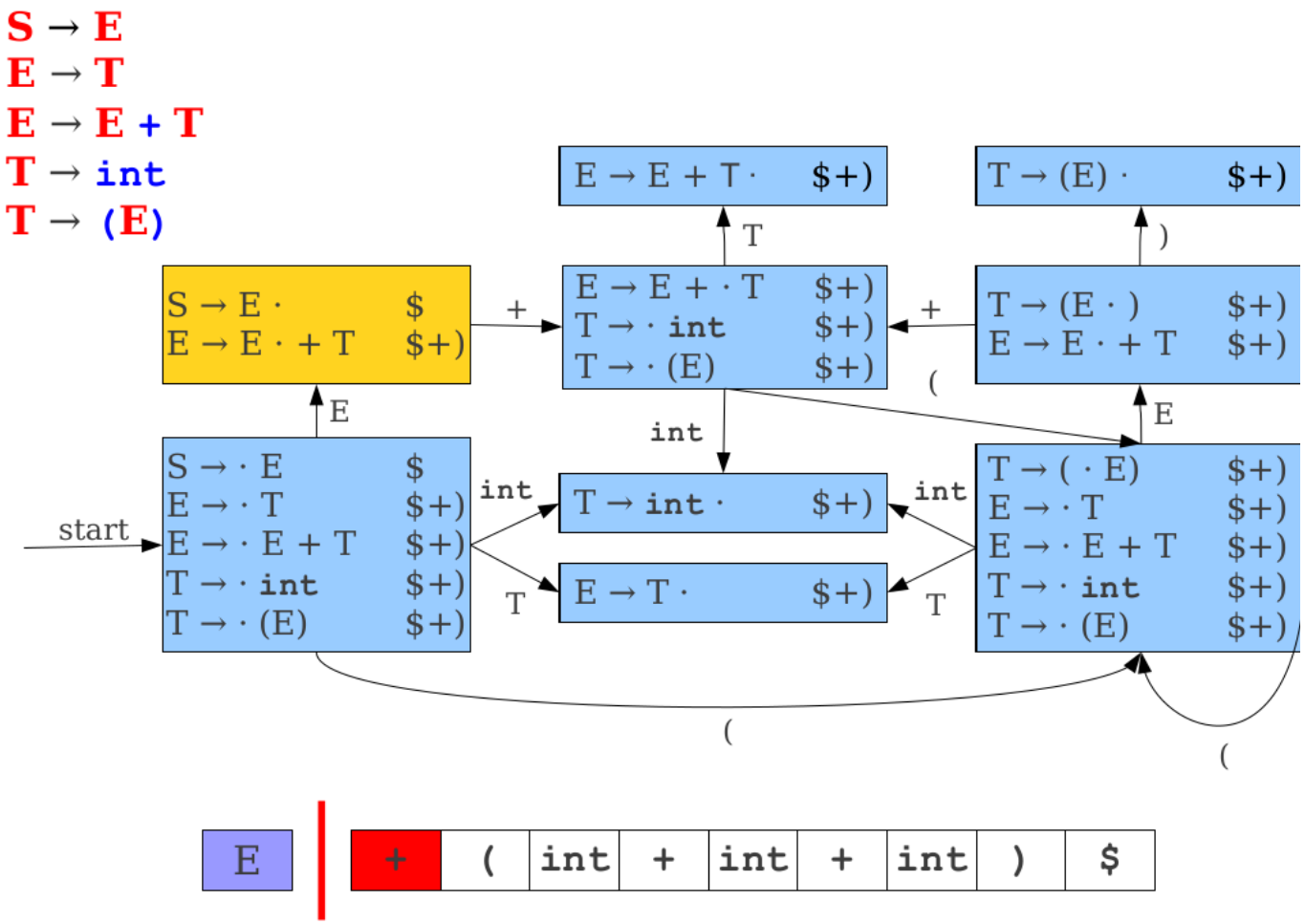
SLR(1) Parsing



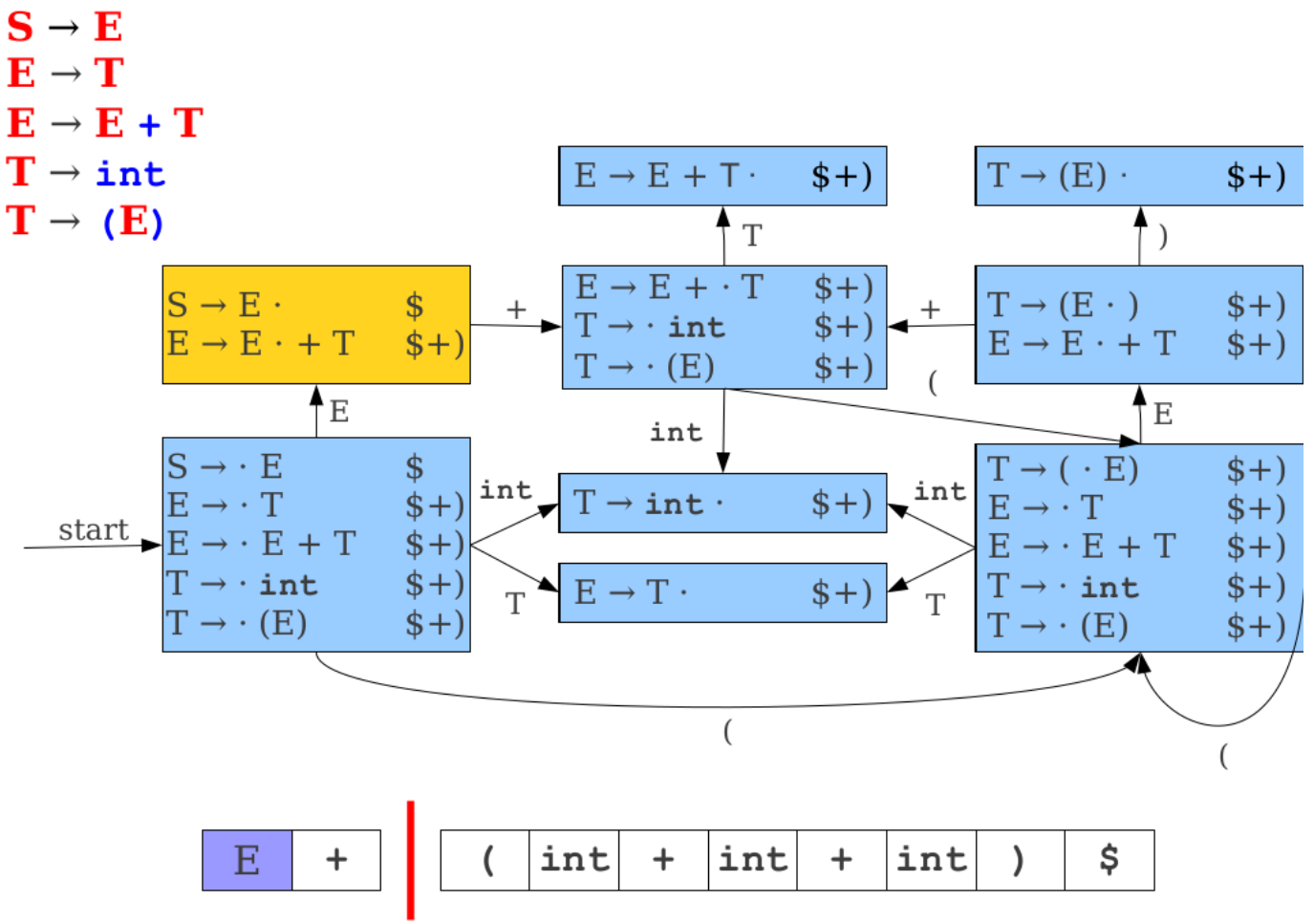
SLR(1) Parsing



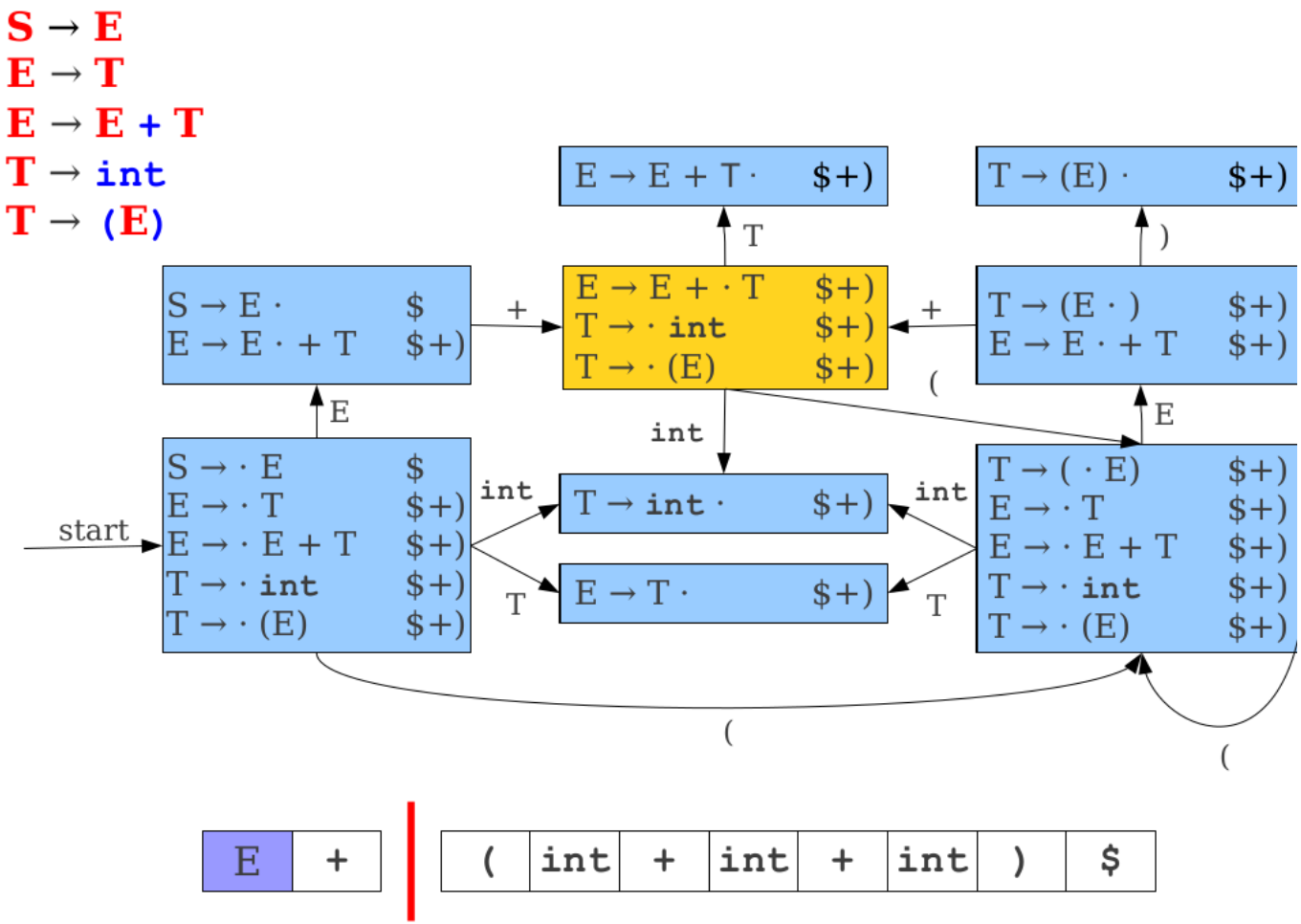
SLR(1) Parsing



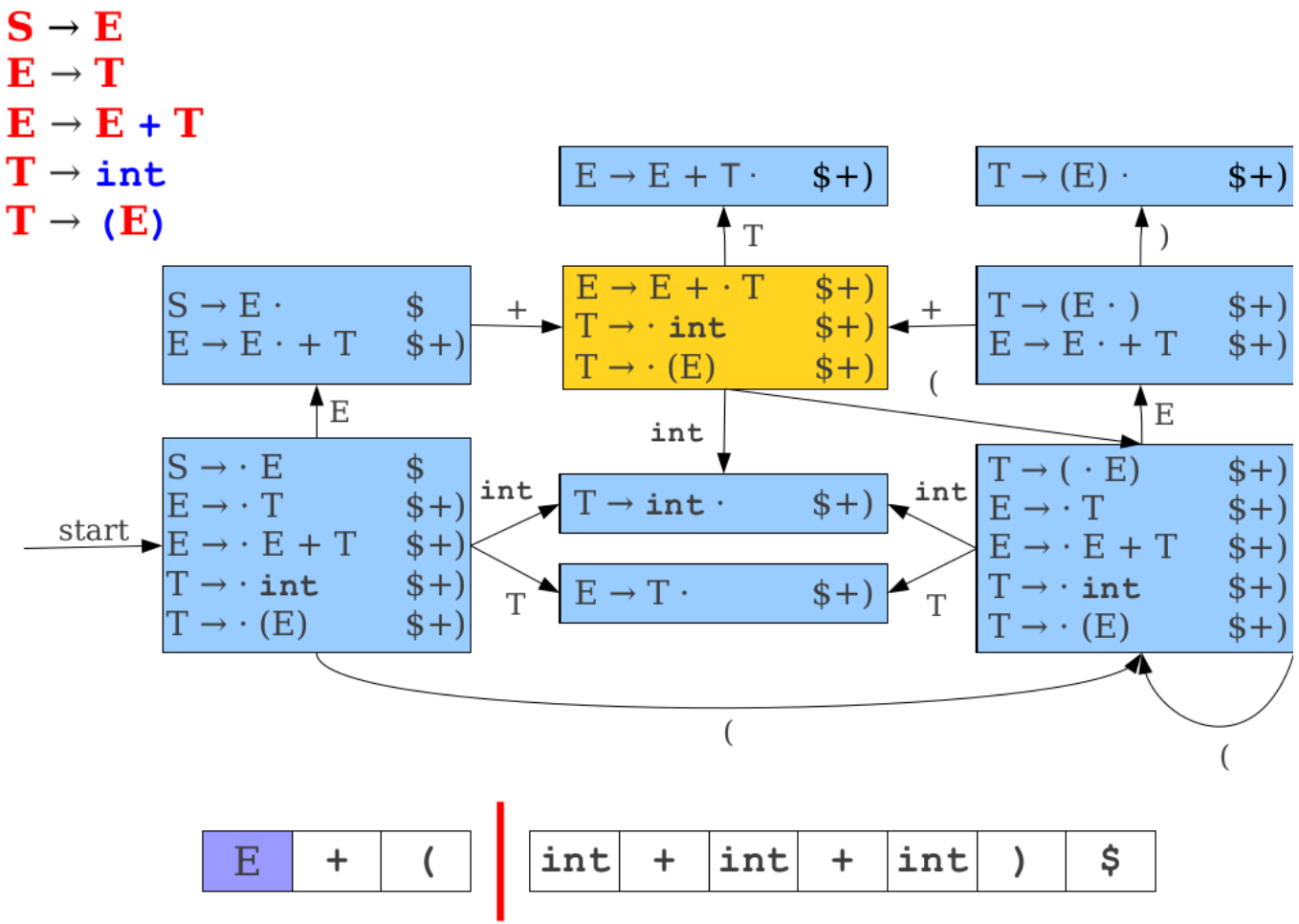
SLR(1) Parsing



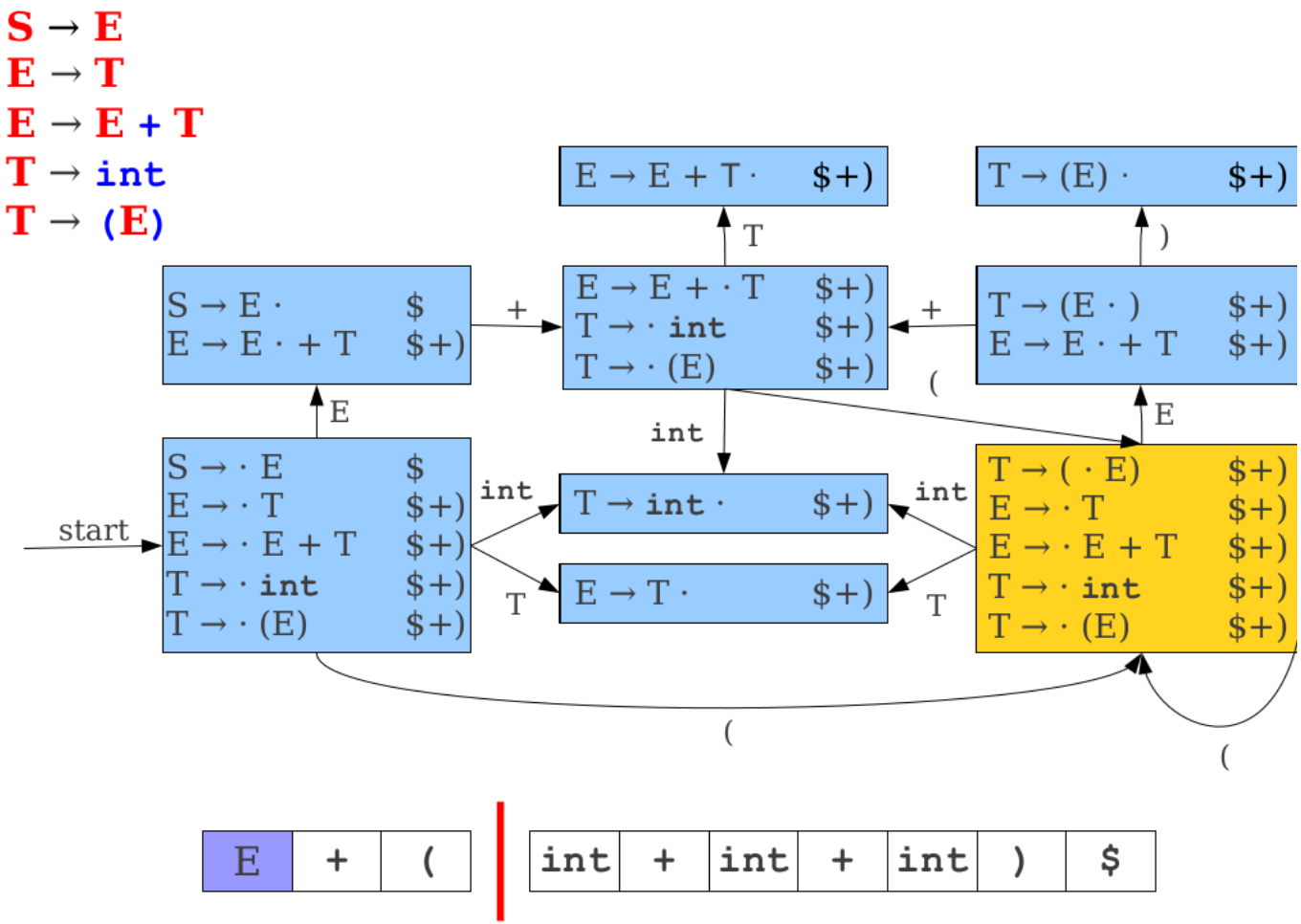
SLR(1) Parsing



SLR(1) Parsing



SLR(1) Parsing



Analysis of SLR(1)

- Exploits lookahead in a small space.
  - Small automaton – same number of states as in as LR(0).
  - Works on many more grammars than LR(0)

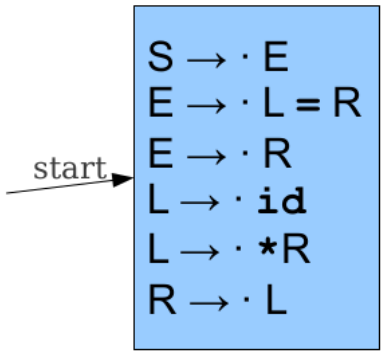
- Too weak for most grammars: lose context from not having extra states.

The Limits of SLR(1)

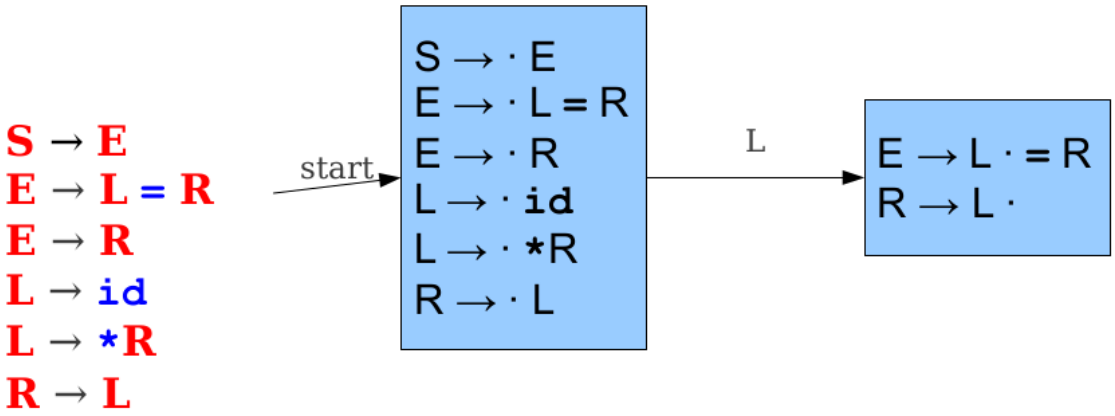
$S \rightarrow E$
$E \rightarrow L = R$
$E \rightarrow R$
$L \rightarrow id$
$L \rightarrow *R$
$R \rightarrow L$

The Limits of SLR(1)

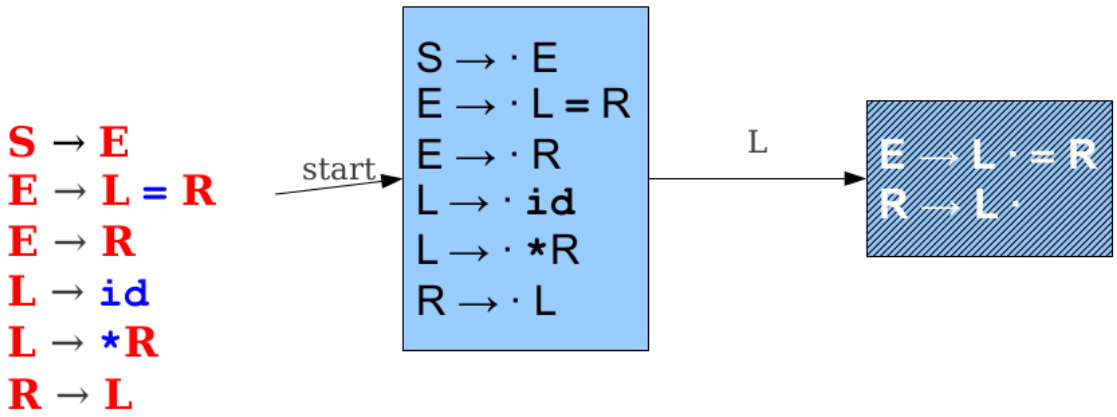
$S \rightarrow E$
$E \rightarrow L = R$
$E \rightarrow R$
$L \rightarrow \texttt{id}$
$L \rightarrow \texttt{*}R$
$R \rightarrow L$



The Limits of SLR(1)

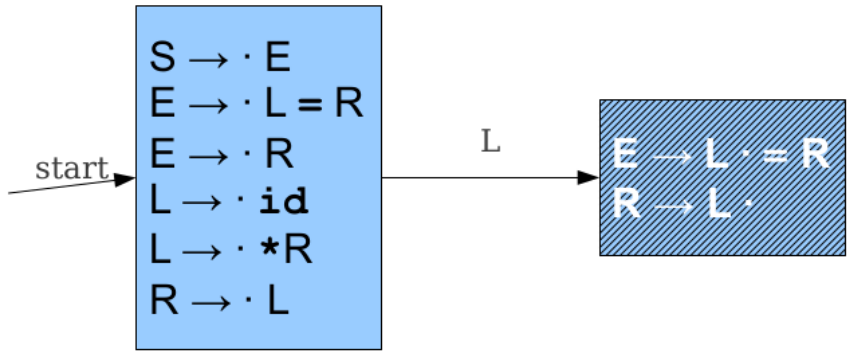


The Limits of SLR(1)



The Limits of SLR(1)

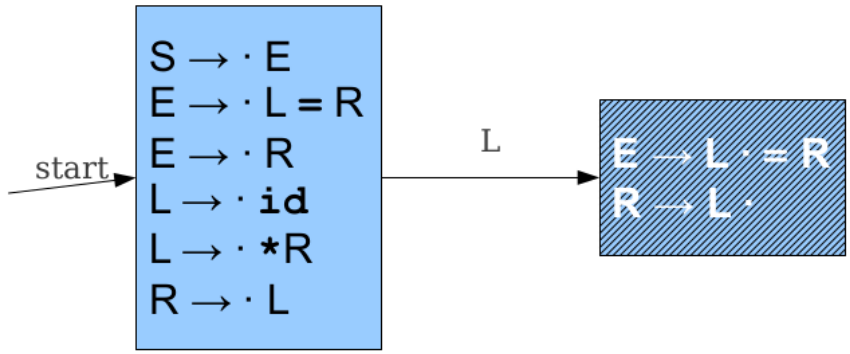
S → E
E → L = R
E → R
L → id
L → *R
R → L



E → L · = R tells us to shift on seeing **=**
R → L · tells us to reduce on FOLLOW(**R**).

The Limits of SLR(1)

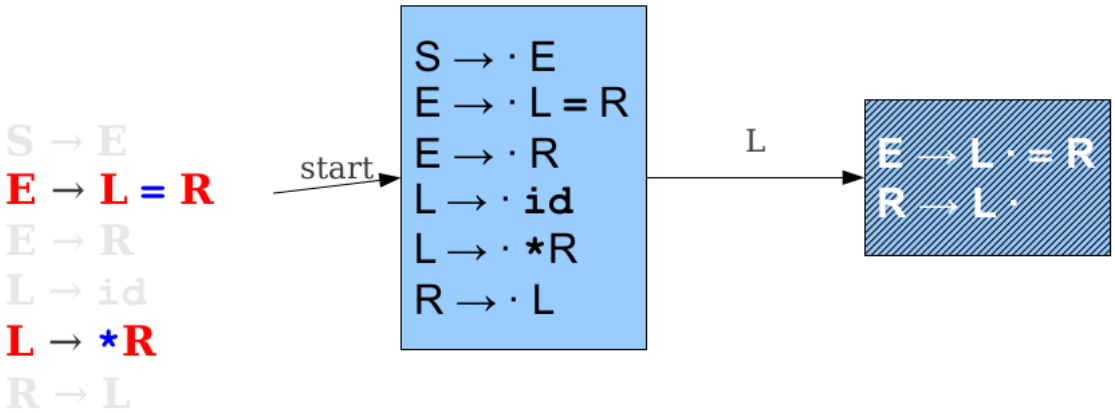
S → E
E → L = R
E → R
L → id
L → *R
R → L



E → L · = R tells us to shift on seeing **=**

R → L · tells us to reduce on FOLLOW(**R**).

The Limits of SLR(1)



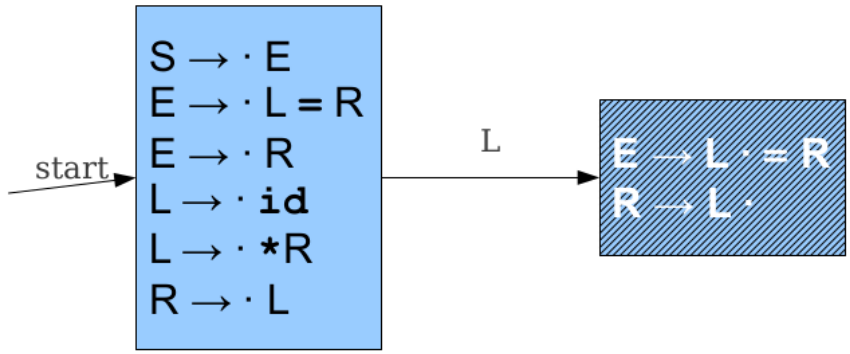
E → **L** · **=** **R** tells us to shift on seeing **=**

R → **L** · tells us to reduce on FOLLOW(**R**).

= ∈ FOLLOW(**R**).

The Limits of SLR(1)

S → E
E → L = R
E → R
L → id
L → *R
R → L



E → L · = R tells us to shift on seeing **=**
R → L · tells us to reduce on FOLLOW(**R**).

= ∈ FOLLOW(**R**).

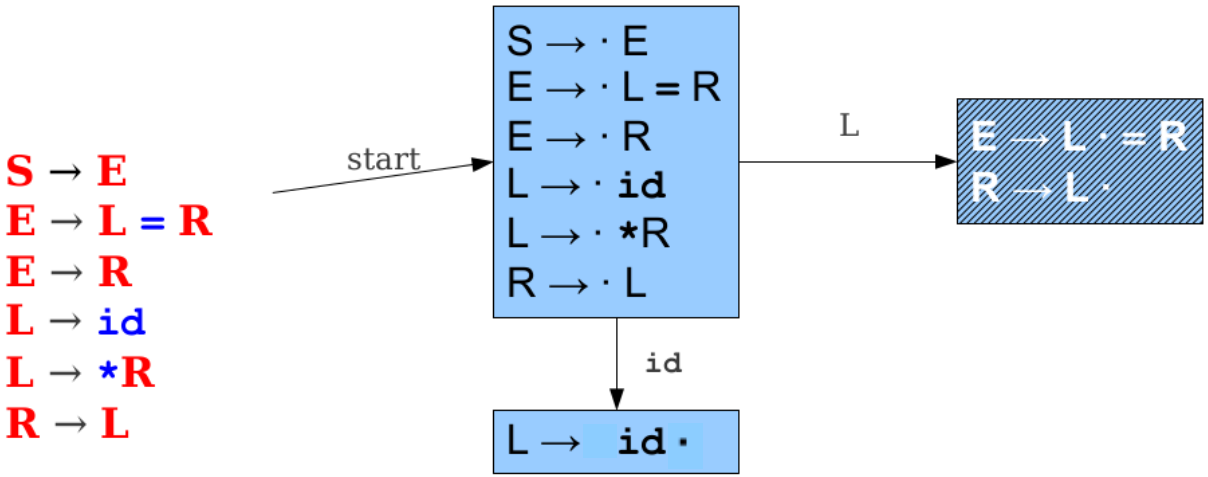
We have a conflict!

Why is SLR(1) Weak?

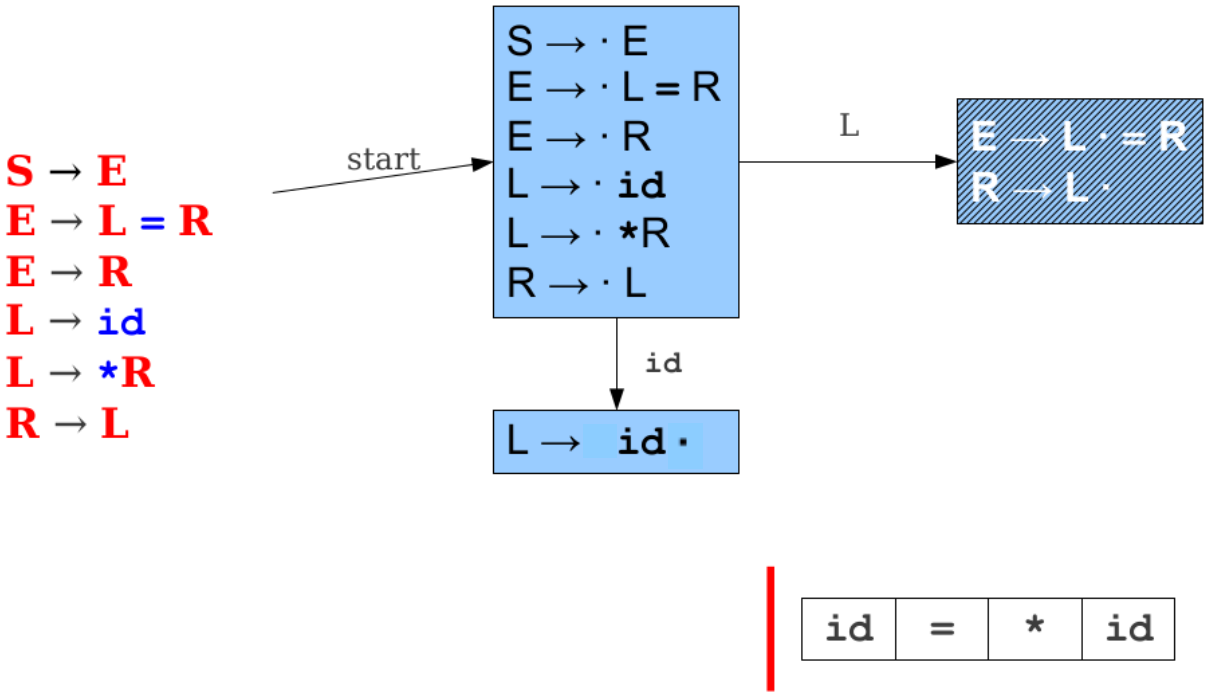
- With LR(1), incredible contextual information.
  - Lookaheads at each state only possible after applying the productions that could get us there.

- With SLR(1), minimal context.
  - FOLLOW(A) means “what could follow A somewhere in the grammar?,” even if in a particular state A couldn't possibly have that symbol after it.

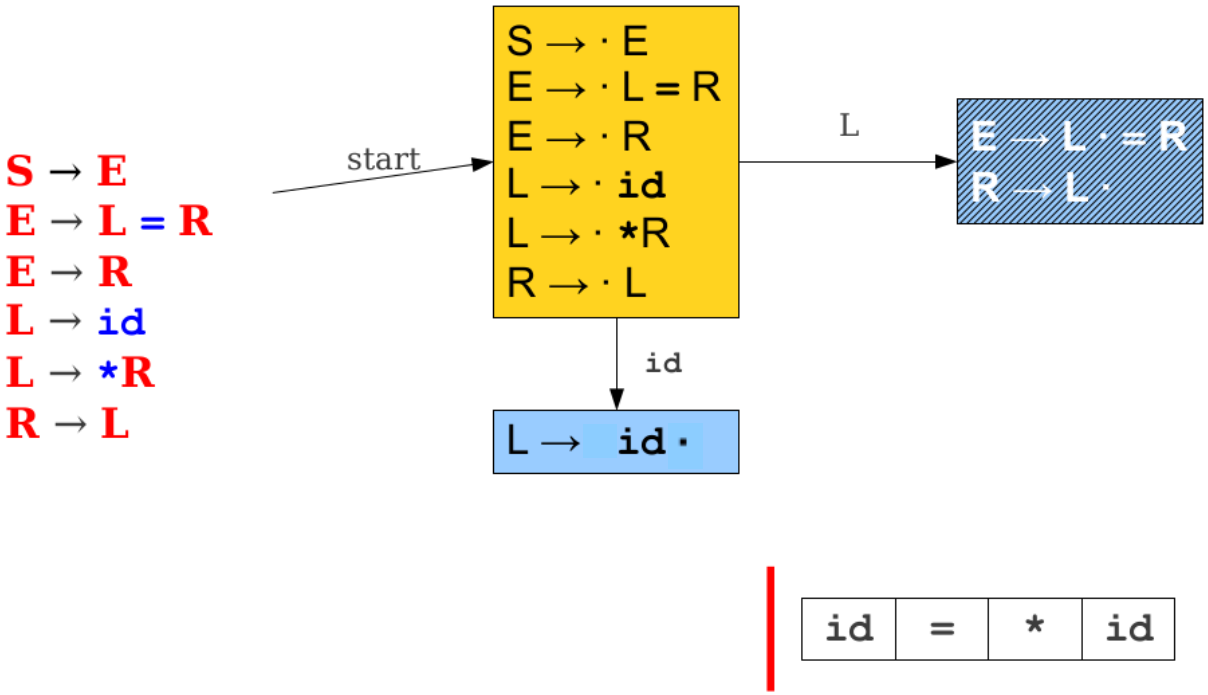
A Lack of Context



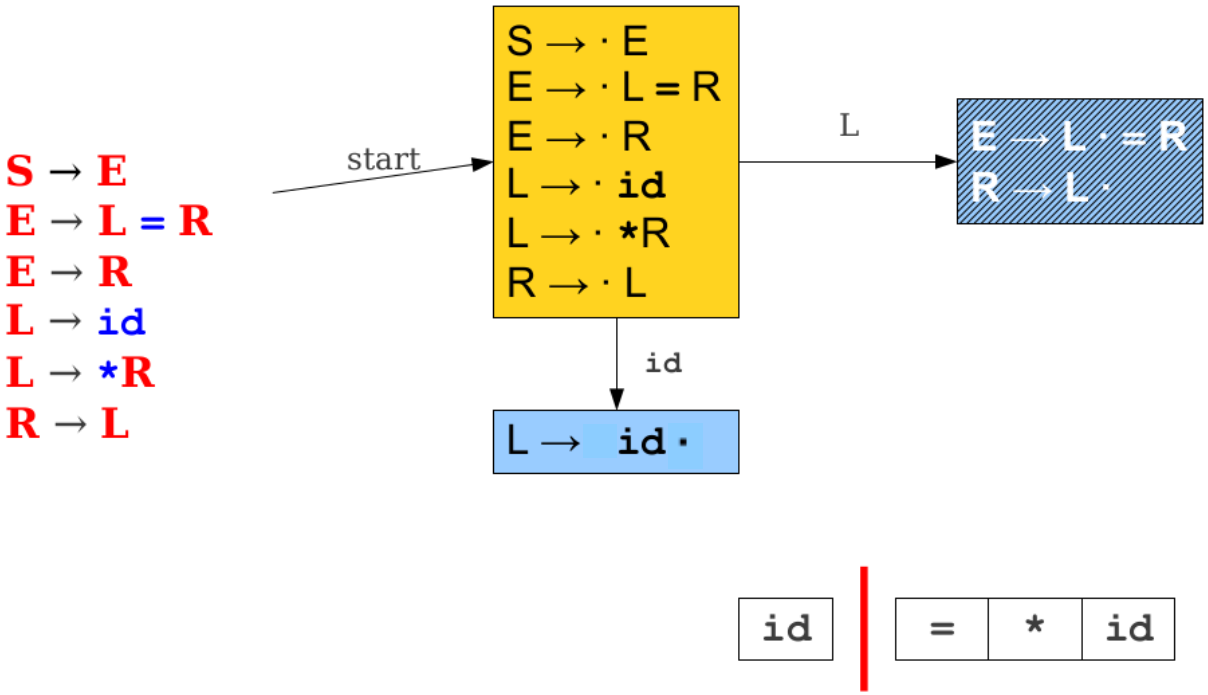
A Lack of Context



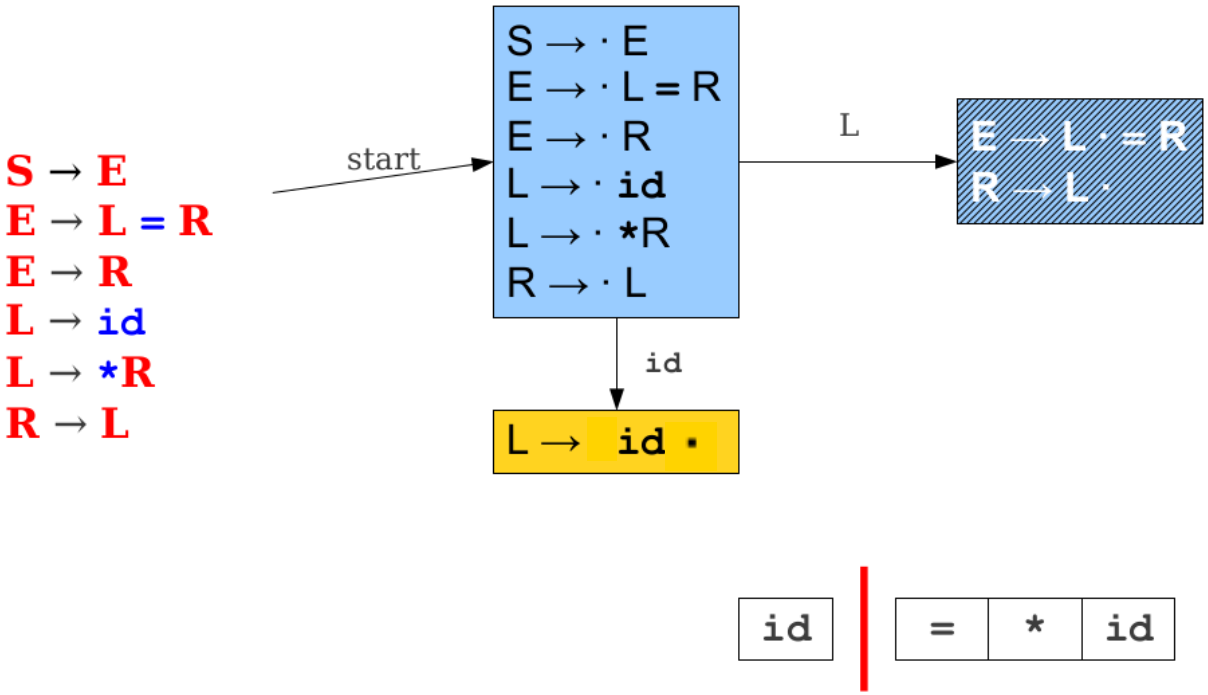
A Lack of Context



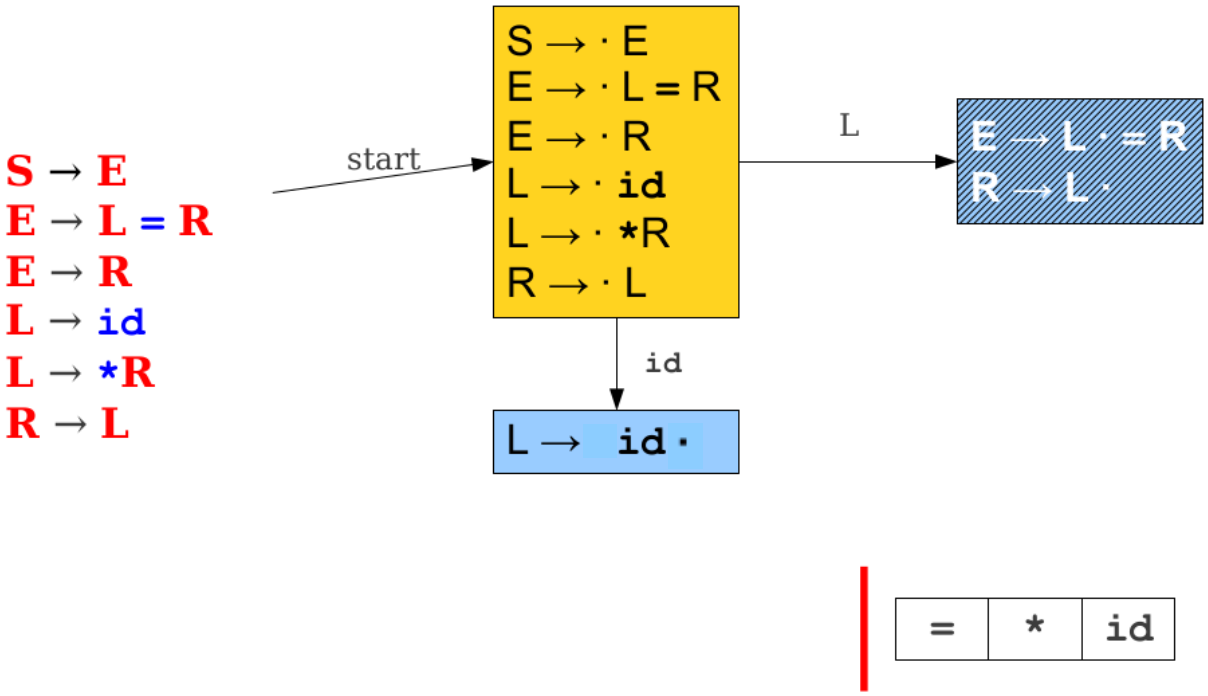
A Lack of Context



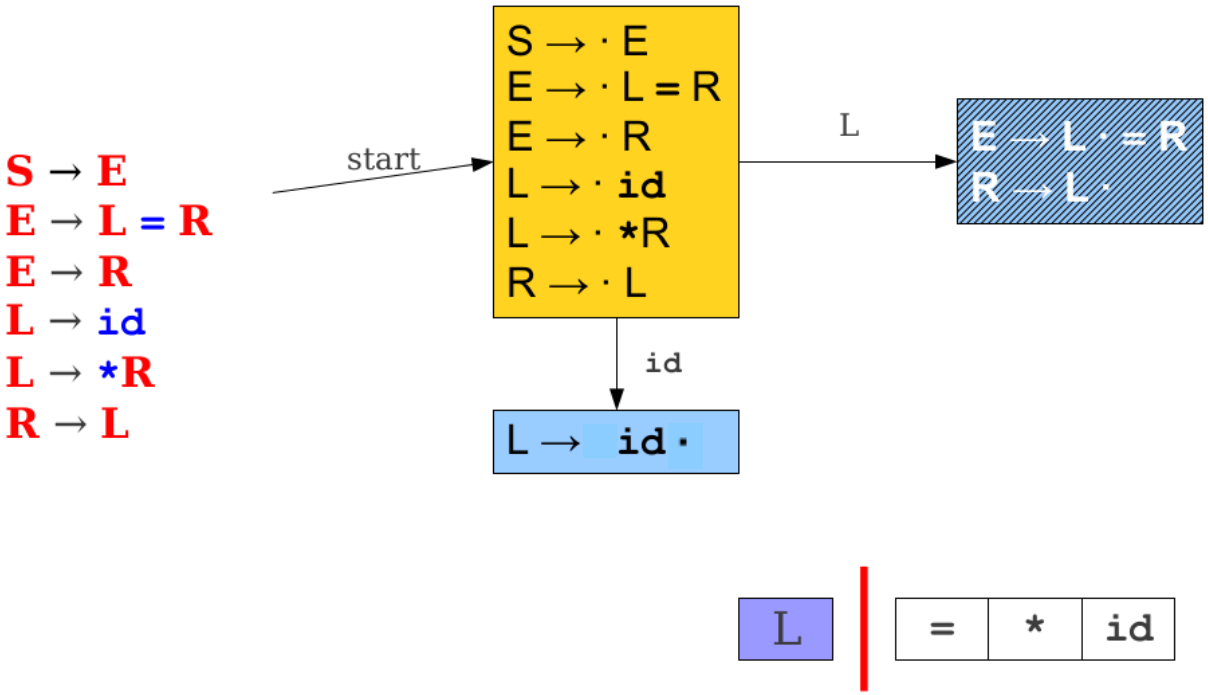
A Lack of Context



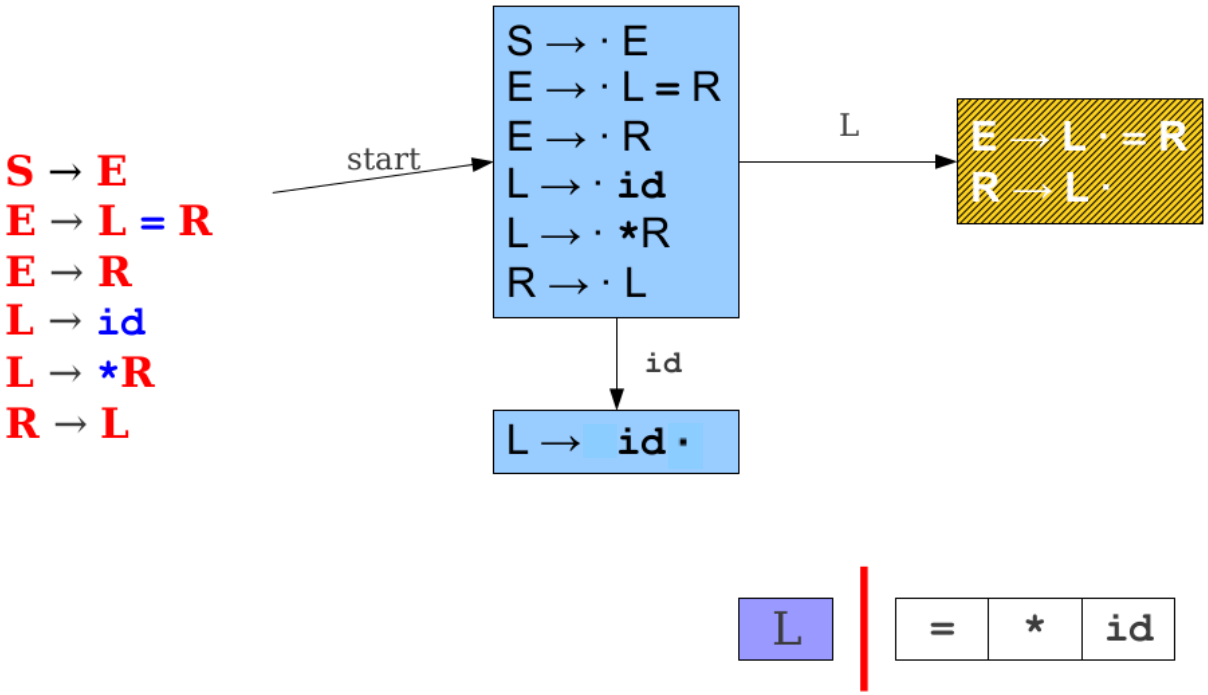
A Lack of Context



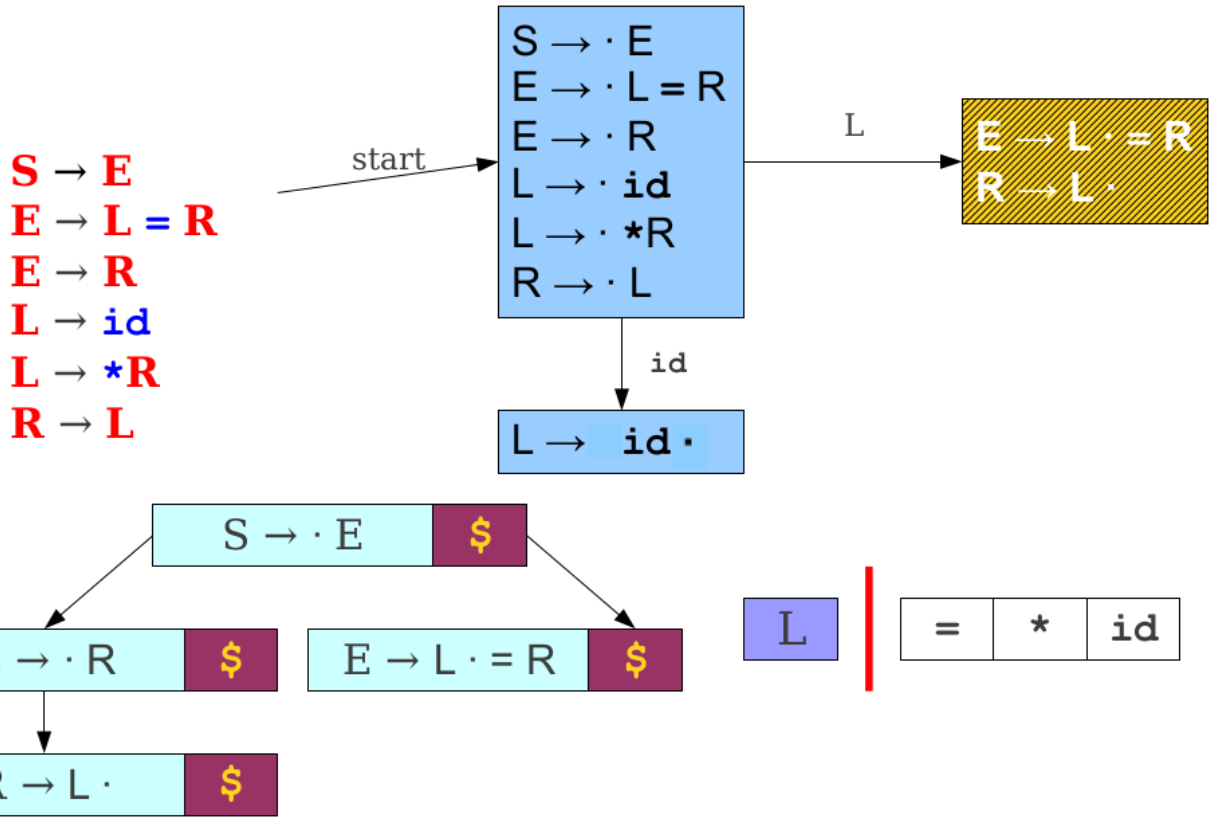
A Lack of Context



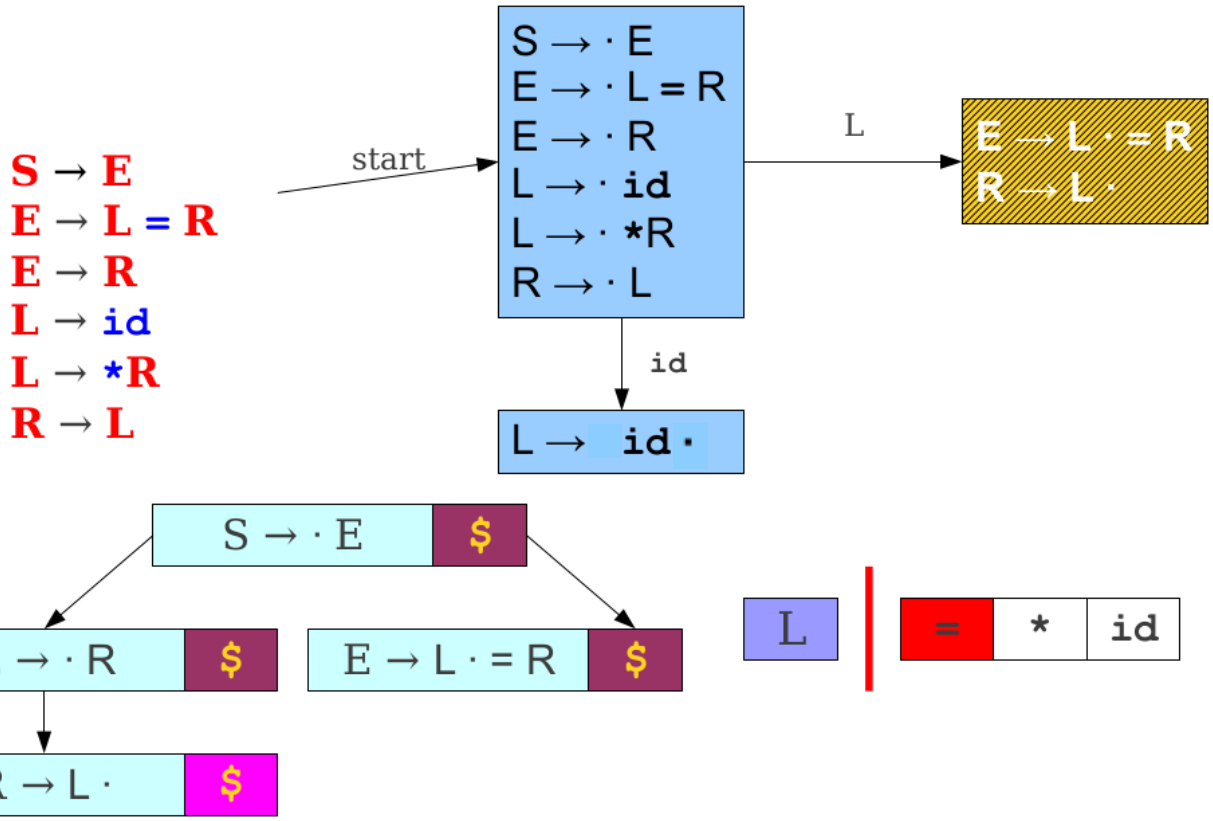
A Lack of Context



A Lack of Context



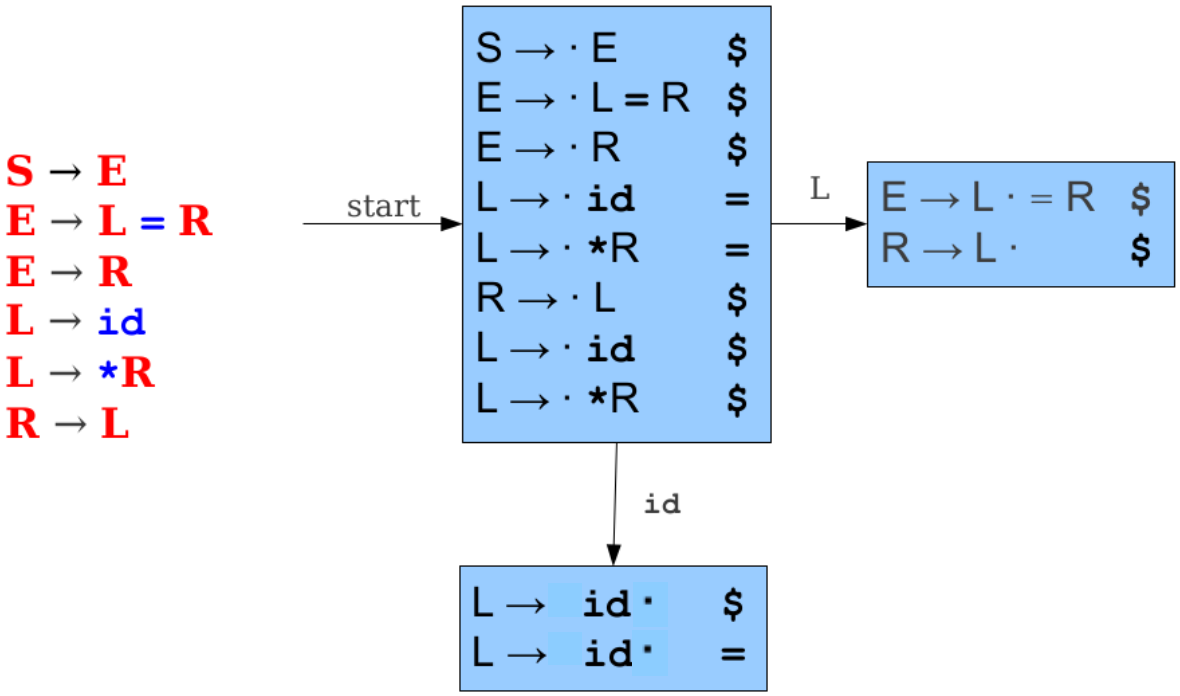
A Lack of Context



For Reference: LR(1) States

$S \rightarrow E$
$E \rightarrow L = R$
$E \rightarrow R$
$L \rightarrow \texttt{id}$
$L \rightarrow *R$
$R \rightarrow L$

For Reference: LR(1) States

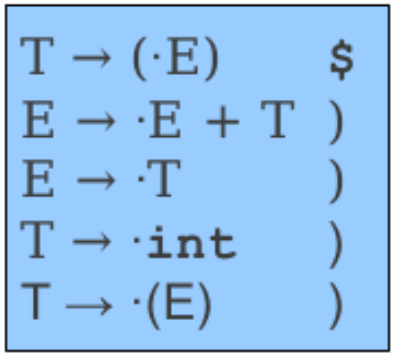


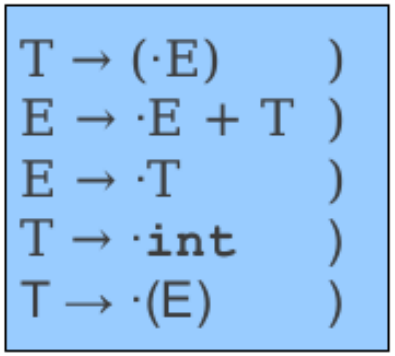
LR(1) and SLR(1)

- SLR(1) is weak because it has no contextual information.
- LR(1) is impractical because its contextual information makes the automaton too big.
- Can we retain the LR(1) automaton's contextual information without all its states?

Review of LR(1)

- Each state in an LR(1) automaton is a combination of an LR(0) state and lookahead information.
- Two LR(1) items have the same core if they are identical except for lookahead.





A Surprisingly Powerful Idea

- In an LR(1) automaton, we have multiple states with the same core but different lookahead.
- What if we merge all these states together?
- This is called LALR(1)
  - Lookahead(1) LR(0)

Outline

- Introduction
- Shift-reduce Parsing
- **LR Parsing**
  - LR(0)
  - LR(1)
  - SLR
  - **LALR**
- Ambiguity
- Error-handling

Question?