Syntax Analysis

Uday Khedker

(www.cse.iitb.ac.in/˜uday)

Department of Computer Science and Engineering,
Indian Institute of Technology, Bombay

February 2024

Outline

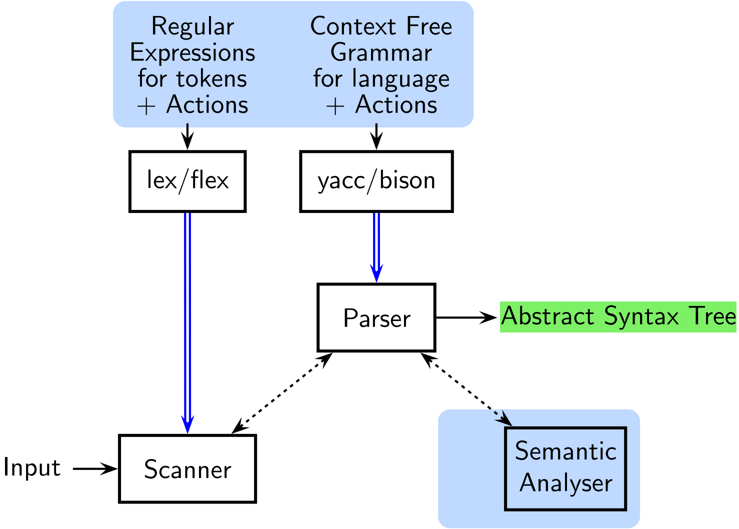
- Grammars, derivations, and parse trees
- Introduction to bottom-up parsing
- Shift reduce parsing
- SLR(1) parsing
- Conceptual issues in LR parsing
- CLR(1) parsing
- LALR(1) parsing

Grammars, derivations, and parse trees

Introduction to Parsing

parsing-slides-sanyal-part1.pdf

A Compiler Front End



Syntax Analysis aka Parsing

- Determines a structure in the input by discovering relationships between tokens representing the input
- This structure is represented by a syntax tree (aka parse tree)
- If a parse tree can be constructed, the input is *syntactically* valid i.e., it is *well-formed* as defined by the language

  It may not be *semantically* valid
- A description of syntax should be
  - unambiguous, correct, complete, and
  - convenient for use by the designers and implementers of a language

  A Context-free grammar (aka grammar) meets these requirements

Derivation

- Transformation of a sequence of grammar symbols
- Obtained by replacing non-terminals by the RHS of a production

Derivation

- Transformation of a sequence of grammar symbols
- Obtained by replacing non-terminals by the RHS of a production
- Consider the following grammar of expressions

$$E \rightarrow E + T \mid T$$
$$T \rightarrow T * F \mid F$$
$$F \rightarrow \text{id}$$

Derivation

- Transformation of a sequence of grammar symbols
- Obtained by replacing non-terminals by the RHS of a production
- Consider the following grammar of expressions

$$\begin{aligned} &E \rightarrow E + T \mid T \\ &T \rightarrow T * F \mid F \\ &F \rightarrow \text{id} \end{aligned}$$

- A possible derivation is

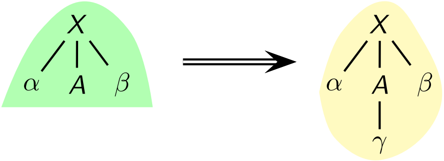
$$\begin{aligned} E &\Rightarrow E + T \\ &\Rightarrow T + T \\ &\Rightarrow F + T \\ &\Rightarrow \text{id} + T \\ &\Rightarrow \text{id} + T * F \\ &\Rightarrow \text{id} + F * F \\ &\Rightarrow \text{id} + \text{id} * F \\ &\Rightarrow \text{id} + \text{id} * \text{id} \end{aligned}$$

Notational Conventions

Symbol type	Convention
single terminal	letters `a`, `b`, `c`, `operators` delimiters, keywords
single nonterminal	letters A, B, C and names such as *declaration* , *list* and S is the start symbol
single grammar symbol (symbol from $\{N \cup T\}$)	X, Y, Z
string of terminals	letters `x` , `y` , `z`
string of grammar symbols	α, β, γ
null string	ϵ

Formalizing a Derivation

- Let $A \rightarrow \gamma$ denote a production and $\alpha A \beta$ denote a string of grammar symbols
- Replacing A in $\alpha A \beta$ by γ gives $\alpha \gamma \beta$
    - We say that $\alpha A \beta$ *derives* $\alpha \gamma \beta$ in one step
    - We write it as $\alpha A \beta \Rightarrow \alpha \gamma \beta$
    - It represents the expansion of a subtree during parsing



- Formally $\alpha_1 \Rightarrow \alpha_2$ is a relation $(N \cup T)^* \times (N \cup T)^*$
- A multi-step derivation is a composition of multiple single step derivations
    - $\alpha_1 \overset{*}{\Rightarrow} \alpha_2$ means α_1 derives α_2 in zero or more steps
    - $\alpha_1 \overset{+}{\Rightarrow} \alpha_2$ means α_1 derives α_2 in one or more steps

The Language Generated by a Grammar

- $L(G) = \{w \mid S \overset{+}{\Rightarrow} w, w \in T^*\}$, where
  - S is the start non-terminal of grammar G, and
  - T is the set of terminal symbols of G
- The strings in $L(G)$ are called the sentences of G
- A string $S \overset{*}{\Rightarrow} \alpha$ is called a sentential form of G
- Every sentence of G is also a sentential form of G
- Grammars G_1 and G_2 are equivalent if $L(G_1) = L(G_2)$

Sentential Forms and Sentences

$$G_1 \; \boxed{\begin{array}{l} E \rightarrow E+T \mid T \\ T \rightarrow T*F \mid F \\ F \rightarrow \text{id} \end{array}} \qquad G_2 \; \boxed{\begin{array}{l} E \rightarrow E+E \\ E \rightarrow E*E \\ E \rightarrow \text{id} \end{array}}$$

- $L(G_1) = L(G_2)$
- $\{\text{id} + \text{id} * \text{id}, \text{id} * \text{id} + \text{id}\} \subset L(G_1)$ (and hence, also of $L(G_2)$)
- $E + T$, $F + E$, $\text{id} + T * F$ are sentential forms of G_1 but not of G_2
- $E + E$, $E * E$, $\text{id} + E * E$ are sentential forms of G_2 but not of G_1

Sentential forms depend on the grammars whereas the sentences depend on the languages generated by grammars

Leftmost and Rightmost Derivations

- A derivation $\alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \Rightarrow \alpha_k$ is a
  - leftmost derivation, denoted $\alpha_1 \overset{lm}{\Rightarrow} \alpha_2 \overset{lm}{\Rightarrow} \ldots \overset{lm}{\Rightarrow} \alpha_k$, if

    every α_{i+1} is obtained from α_i by replacing the leftmost non-terminal occurring in α_i by the RHS of some production of the non-terminal
  - rightmost derivation, denoted $\alpha_1 \overset{rm}{\Rightarrow} \alpha_2 \overset{rm}{\Rightarrow} \ldots \overset{rm}{\Rightarrow} \alpha_k$, if

    every α_{i+1} is obtained from α_i by replacing the rightmost non-terminal occurring in α_i by the RHS of some production of the non-terminal
- A sentential form α is called
  - a left sentential form, if it occurs in a leftmost derivation
  - a right sentential form, if it occurs in a rightmost derivation

Note that α could be both a right and a left sentential form

Leftmost and Rightmost Derivations

Grammar
$$\boxed{\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

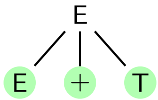
Leftmost Derivation	Rightmost Derivation
E	

Leftmost and Rightmost Derivations

Grammar
$$\boxed{\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

Leftmost Derivation	Rightmost Derivation
$E \overset{lm}{\Rightarrow} E + T$	



Leftmost and Rightmost Derivations

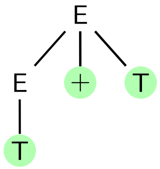
Grammar

$$\boxed{\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: id + id ∗ id

Leftmost Derivation	Rightmost Derivation

$$\begin{aligned} E &\overset{lm}{\Rightarrow} E + T \\ &\overset{lm}{\Rightarrow} T + T \end{aligned}$$



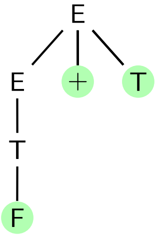
Leftmost and Rightmost Derivations

Grammar
$$\boxed{\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

Leftmost Derivation	Rightmost Derivation

$$\begin{aligned} E &\overset{lm}{\Rightarrow} E + T \\ &\overset{lm}{\Rightarrow} T + T \\ &\overset{lm}{\Rightarrow} F + T \end{aligned}$$



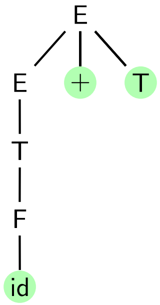
Leftmost and Rightmost Derivations

Grammar
$$\boxed{\begin{array}{l} E \rightarrow E+T \mid T \\ T \rightarrow T*F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

Leftmost Derivation

$$\begin{aligned} E &\overset{lm}{\Rightarrow} E+T \\ &\overset{lm}{\Rightarrow} T+T \\ &\overset{lm}{\Rightarrow} F+T \\ &\overset{lm}{\Rightarrow} \text{id}+T \end{aligned}$$



Rightmost Derivation

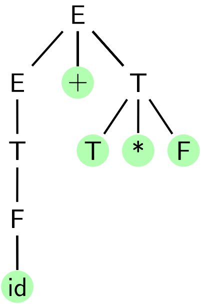
Leftmost and Rightmost Derivations

Grammar
$$\boxed{\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

Leftmost Derivation	Rightmost Derivation

$$\begin{aligned} E &\overset{lm}{\Rightarrow} E + T \\ &\overset{lm}{\Rightarrow} T + T \\ &\overset{lm}{\Rightarrow} F + T \\ &\overset{lm}{\Rightarrow} \text{id} + T \\ &\overset{lm}{\Rightarrow} \text{id} + T * F \end{aligned}$$



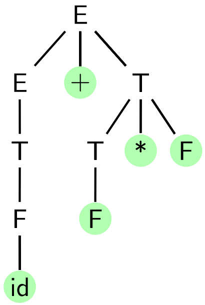
Leftmost and Rightmost Derivations

Grammar
$$\boxed{\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

Leftmost Derivation	Rightmost Derivation

$$\begin{aligned} E &\overset{lm}{\Rightarrow} E + T \\ &\overset{lm}{\Rightarrow} T + T \\ &\overset{lm}{\Rightarrow} F + T \\ &\overset{lm}{\Rightarrow} \text{id} + T \\ &\overset{lm}{\Rightarrow} \text{id} + T * F \\ &\overset{lm}{\Rightarrow} \text{id} + F * F \end{aligned}$$



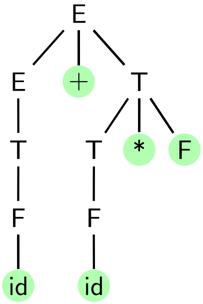
Leftmost and Rightmost Derivations

Grammar $\boxed{\begin{array}{l} E \rightarrow E+T \mid T \\ T \rightarrow T*F \mid F \\ F \rightarrow \text{id} \end{array}}$

Sentence: $\text{id} + \text{id} * \text{id}$

Leftmost Derivation	Rightmost Derivation

$$\begin{aligned} E &\overset{lm}{\Rightarrow} E+T \\ &\overset{lm}{\Rightarrow} T+T \\ &\overset{lm}{\Rightarrow} F+T \\ &\overset{lm}{\Rightarrow} \text{id}+T \\ &\overset{lm}{\Rightarrow} \text{id}+T*F \\ &\overset{lm}{\Rightarrow} \text{id}+F*F \\ &\overset{lm}{\Rightarrow} \text{id}+\text{id}*F \end{aligned}$$



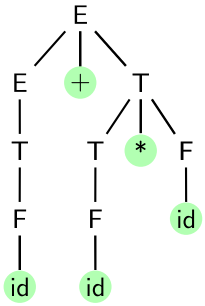
Leftmost and Rightmost Derivations

Grammar
$$\boxed{\begin{array}{l} E \rightarrow E+T \mid T \\ T \rightarrow T*F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

Leftmost Derivation	Rightmost Derivation

$$\begin{array}{ll} E & \overset{lm}{\Rightarrow} E + T \\ & \overset{lm}{\Rightarrow} T + T \\ & \overset{lm}{\Rightarrow} F + T \\ & \overset{lm}{\Rightarrow} \text{id} + T \\ & \overset{lm}{\Rightarrow} \text{id} + T * F \\ & \overset{lm}{\Rightarrow} \text{id} + F * F \\ & \overset{lm}{\Rightarrow} \text{id} + \text{id} * F \\ & \overset{lm}{\Rightarrow} \text{id} + \text{id} * \text{id} \end{array}$$



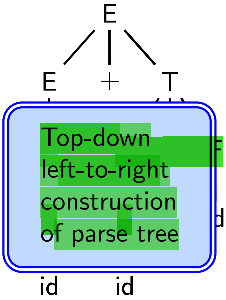
Leftmost and Rightmost Derivations

Grammar
$$\boxed{\begin{array}{l} E \rightarrow E+T \mid T \\ T \rightarrow T*F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

Leftmost Derivation	Rightmost Derivation

$$\begin{aligned} E &\overset{lm}{\Rightarrow} E+T \\ &\overset{lm}{\Rightarrow} T+T \\ &\overset{lm}{\Rightarrow} F+T \\ &\overset{lm}{\Rightarrow} \text{id}+T \\ &\overset{lm}{\Rightarrow} \text{id}+T*F \\ &\overset{lm}{\Rightarrow} \text{id}+F*F \\ &\overset{lm}{\Rightarrow} \text{id}+\text{id}*F \\ &\overset{lm}{\Rightarrow} \text{id}+\text{id}*\text{id} \end{aligned}$$



Top-down left-to-right construction of parse tree

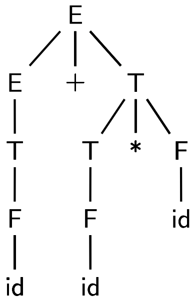
Leftmost and Rightmost Derivations

Grammar
$$\boxed{\begin{array}{l} E \rightarrow E+T \mid T \\ T \rightarrow T*F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

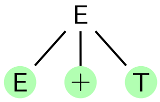
Leftmost Derivation

$$\begin{aligned} E &\overset{lm}{\Rightarrow} E+T \\ &\overset{lm}{\Rightarrow} T+T \\ &\overset{lm}{\Rightarrow} F+T \\ &\overset{lm}{\Rightarrow} \text{id}+T \\ &\overset{lm}{\Rightarrow} \text{id}+T*F \\ &\overset{lm}{\Rightarrow} \text{id}+F*F \\ &\overset{lm}{\Rightarrow} \text{id}+\text{id}*F \\ &\overset{lm}{\Rightarrow} \text{id}+\text{id}*\text{id} \end{aligned}$$



Rightmost Derivation

$$E \overset{rm}{\Rightarrow} E+T$$



Leftmost and Rightmost Derivations

Grammar

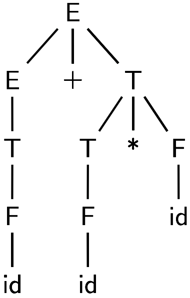
$$\boxed{\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

Leftmost Derivation	Rightmost Derivation

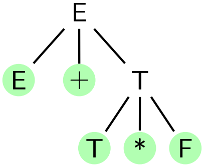
Leftmost Derivation

$$\begin{aligned} E &\overset{lm}{\Rightarrow} E + T \\ &\overset{lm}{\Rightarrow} T + T \\ &\overset{lm}{\Rightarrow} F + T \\ &\overset{lm}{\Rightarrow} \text{id} + T \\ &\overset{lm}{\Rightarrow} \text{id} + T * F \\ &\overset{lm}{\Rightarrow} \text{id} + F * F \\ &\overset{lm}{\Rightarrow} \text{id} + \text{id} * F \\ &\overset{lm}{\Rightarrow} \text{id} + \text{id} * \text{id} \end{aligned}$$



Rightmost Derivation

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + T \\ &\overset{rm}{\Rightarrow} E + T * F \end{aligned}$$



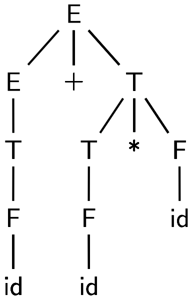
Leftmost and Rightmost Derivations

$$\text{Grammar}\ \boxed{\begin{array}{l} E \rightarrow E+T \mid T \\ T \rightarrow T*F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

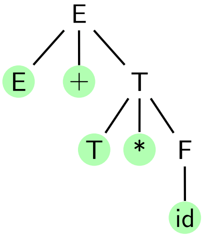
Leftmost Derivation

$$\begin{aligned} E &\overset{lm}{\Rightarrow} E+T \\ &\overset{lm}{\Rightarrow} T+T \\ &\overset{lm}{\Rightarrow} F+T \\ &\overset{lm}{\Rightarrow} \text{id}+T \\ &\overset{lm}{\Rightarrow} \text{id}+T*F \\ &\overset{lm}{\Rightarrow} \text{id}+F*F \\ &\overset{lm}{\Rightarrow} \text{id}+\text{id}*F \\ &\overset{lm}{\Rightarrow} \text{id}+\text{id}*\text{id} \end{aligned}$$



Rightmost Derivation

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E+T \\ &\overset{rm}{\Rightarrow} E+T*F \\ &\overset{rm}{\Rightarrow} E+T*\text{id} \end{aligned}$$



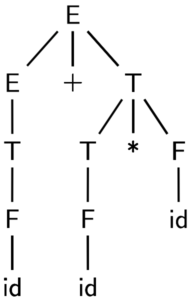
Leftmost and Rightmost Derivations

Grammar
$$\boxed{\begin{array}{l} E \rightarrow E+T \mid T \\ T \rightarrow T*F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

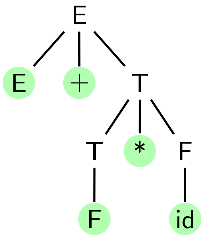
Leftmost Derivation

$$\begin{aligned} E &\overset{lm}{\Rightarrow} E+T \\ &\overset{lm}{\Rightarrow} T+T \\ &\overset{lm}{\Rightarrow} F+T \\ &\overset{lm}{\Rightarrow} \text{id}+T \\ &\overset{lm}{\Rightarrow} \text{id}+T*F \\ &\overset{lm}{\Rightarrow} \text{id}+F*F \\ &\overset{lm}{\Rightarrow} \text{id}+\text{id}*F \\ &\overset{lm}{\Rightarrow} \text{id}+\text{id}*\text{id} \end{aligned}$$



Rightmost Derivation

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E+T \\ &\overset{rm}{\Rightarrow} E+T*F \\ &\overset{rm}{\Rightarrow} E+T*\text{id} \\ &\overset{rm}{\Rightarrow} E+F*\text{id} \end{aligned}$$



Leftmost and Rightmost Derivations

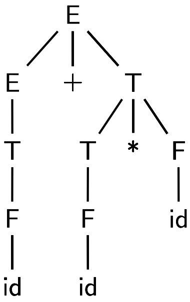
Grammar
$$\boxed{\begin{array}{l} E \rightarrow E+T \mid T \\ T \rightarrow T*F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

Leftmost Derivation	Rightmost Derivation

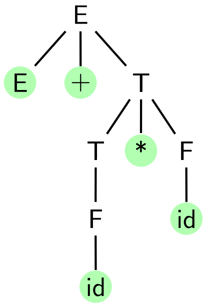
Leftmost Derivation

$$\begin{array}{ll} E & \overset{lm}{\Rightarrow} E+T \\ & \overset{lm}{\Rightarrow} T+T \\ & \overset{lm}{\Rightarrow} F+T \\ & \overset{lm}{\Rightarrow} \text{id}+T \\ & \overset{lm}{\Rightarrow} \text{id}+T*F \\ & \overset{lm}{\Rightarrow} \text{id}+F*F \\ & \overset{lm}{\Rightarrow} \text{id}+\text{id}*F \\ & \overset{lm}{\Rightarrow} \text{id}+\text{id}*\text{id} \end{array}$$



Rightmost Derivation

$$\begin{array}{ll} E & \overset{rm}{\Rightarrow} E+T \\ & \overset{rm}{\Rightarrow} E+T*F \\ & \overset{rm}{\Rightarrow} E+T*\text{id} \\ & \overset{rm}{\Rightarrow} E+F*\text{id} \\ & \overset{rm}{\Rightarrow} E+\text{id}*\text{id} \end{array}$$



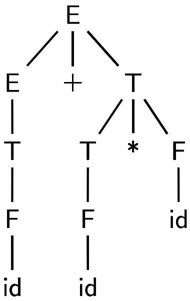
Leftmost and Rightmost Derivations

Grammar
$$\boxed{\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

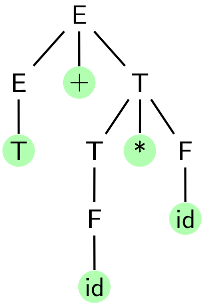
Leftmost Derivation

$$\begin{aligned} E &\overset{lm}{\Rightarrow} E + T \\ &\overset{lm}{\Rightarrow} T + T \\ &\overset{lm}{\Rightarrow} F + T \\ &\overset{lm}{\Rightarrow} \text{id} + T \\ &\overset{lm}{\Rightarrow} \text{id} + T * F \\ &\overset{lm}{\Rightarrow} \text{id} + F * F \\ &\overset{lm}{\Rightarrow} \text{id} + \text{id} * F \\ &\overset{lm}{\Rightarrow} \text{id} + \text{id} * \text{id} \end{aligned}$$



Rightmost Derivation

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + T \\ &\overset{rm}{\Rightarrow} E + T * F \\ &\overset{rm}{\Rightarrow} E + T * \text{id} \\ &\overset{rm}{\Rightarrow} E + F * \text{id} \\ &\overset{rm}{\Rightarrow} E + \text{id} * \text{id} \\ &\overset{rm}{\Rightarrow} T + \text{id} * \text{id} \end{aligned}$$



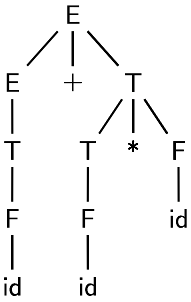
Leftmost and Rightmost Derivations

Grammar
$$\boxed{\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

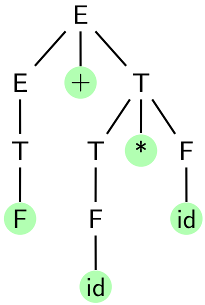
Leftmost Derivation

$$\begin{aligned} E &\overset{lm}{\Rightarrow} E + T \\ &\overset{lm}{\Rightarrow} T + T \\ &\overset{lm}{\Rightarrow} F + T \\ &\overset{lm}{\Rightarrow} \text{id} + T \\ &\overset{lm}{\Rightarrow} \text{id} + T * F \\ &\overset{lm}{\Rightarrow} \text{id} + F * F \\ &\overset{lm}{\Rightarrow} \text{id} + \text{id} * F \\ &\overset{lm}{\Rightarrow} \text{id} + \text{id} * \text{id} \end{aligned}$$



Rightmost Derivation

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + T \\ &\overset{rm}{\Rightarrow} E + T * F \\ &\overset{rm}{\Rightarrow} E + T * \text{id} \\ &\overset{rm}{\Rightarrow} E + F * \text{id} \\ &\overset{rm}{\Rightarrow} E + \text{id} * \text{id} \\ &\overset{rm}{\Rightarrow} T + \text{id} * \text{id} \\ &\overset{rm}{\Rightarrow} F + \text{id} * \text{id} \end{aligned}$$



Leftmost and Rightmost Derivations

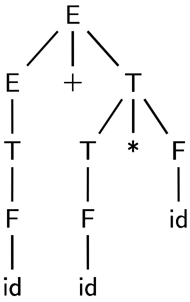
Grammar

$$\boxed{\begin{array}{l} E \rightarrow E+T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

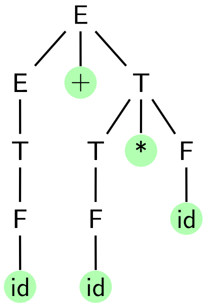
Leftmost Derivation

$$\begin{array}{ll} E & \overset{lm}{\Rightarrow} E + T \\ & \overset{lm}{\Rightarrow} T + T \\ & \overset{lm}{\Rightarrow} F + T \\ & \overset{lm}{\Rightarrow} \text{id} + T \\ & \overset{lm}{\Rightarrow} \text{id} + T * F \\ & \overset{lm}{\Rightarrow} \text{id} + F * F \\ & \overset{lm}{\Rightarrow} \text{id} + \text{id} * F \\ & \overset{lm}{\Rightarrow} \text{id} + \text{id} * \text{id} \end{array}$$



Rightmost Derivation

$$\begin{array}{ll} E & \overset{rm}{\Rightarrow} E + T \\ & \overset{rm}{\Rightarrow} E + T * F \\ & \overset{rm}{\Rightarrow} E + T * \text{id} \\ & \overset{rm}{\Rightarrow} E + F * \text{id} \\ & \overset{rm}{\Rightarrow} E + \text{id} * \text{id} \\ & \overset{rm}{\Rightarrow} T + \text{id} * \text{id} \\ & \overset{rm}{\Rightarrow} F + \text{id} * \text{id} \\ & \overset{rm}{\Rightarrow} \text{id} + \text{id} * \text{id} \end{array}$$



IIT Bombay
cs302: Implementation of Programming Languages

Topic:

Syntax Analysis

Section:

Grammars, Derivations, and Parse Trees

Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in Parsing

CLR(1) Parsing

LALR(1) Parsing

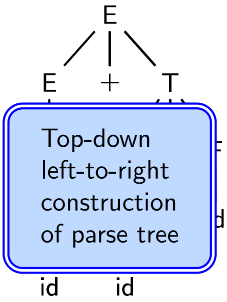
Leftmost and Rightmost Derivations

Grammar
$$\boxed{\begin{array}{l} E \rightarrow E+T \mid T \\ T \rightarrow T*F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

Leftmost Derivation

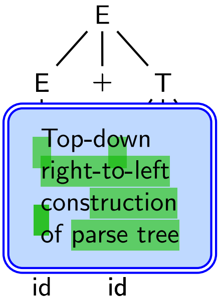
$$\begin{array}{ll} E & \overset{lm}{\Rightarrow} E+T \\ & \overset{lm}{\Rightarrow} T+T \\ & \overset{lm}{\Rightarrow} F+T \\ & \overset{lm}{\Rightarrow} \text{id}+T \\ & \overset{lm}{\Rightarrow} \text{id}+T*F \\ & \overset{lm}{\Rightarrow} \text{id}+F*F \\ & \overset{lm}{\Rightarrow} \text{id}+\text{id}*F \\ & \overset{lm}{\Rightarrow} \text{id}+\text{id}*\text{id} \end{array}$$



Top-down left-to-right construction of parse tree

Rightmost Derivation

$$\begin{array}{ll} E & \overset{rm}{\Rightarrow} E+T \\ & \overset{rm}{\Rightarrow} E+T*F \\ & \overset{rm}{\Rightarrow} E+T*\text{id} \\ & \overset{rm}{\Rightarrow} E+F*\text{id} \\ & \overset{rm}{\Rightarrow} E+\text{id}*\text{id} \\ & \overset{rm}{\Rightarrow} T+\text{id}*\text{id} \\ & \overset{rm}{\Rightarrow} F+\text{id}*\text{id} \\ & \overset{rm}{\Rightarrow} \text{id}+\text{id}*\text{id} \end{array}$$



Top-down right-to-left construction of parse tree

Derivations and Sentences

G_1
$$\boxed{\begin{array}{l} E \rightarrow E+T \mid T \\ T \rightarrow T*F \mid F \\ F \rightarrow \text{id} \end{array}}$$

G_2
$$\boxed{\begin{array}{l} E \rightarrow E+E \\ E \rightarrow E*E \\ E \rightarrow \text{id} \end{array}}$$

- Although $L(G_1) = L(G_2)$,
  - G_1 has a unique leftmost(rightmost) derivation for every sentence
  - G_2 admits multiple leftmost(rightmost) derivations for some sentences
- For sentence $\text{id} + \text{id} * \text{id}$, G_2 admits the following two leftmost derivations
  - $E \overset{lm}{\Rightarrow} E+E \overset{lm}{\Rightarrow} \text{id}+E \overset{lm}{\Rightarrow} \text{id}+E*E \overset{lm}{\Rightarrow} \text{id}+\text{id}*E \overset{lm}{\Rightarrow} \text{id}+\text{id}*\text{id}$
    This derivation represents the grouping $\text{id} + (\text{id} * \text{id})$
  - $E \overset{lm}{\Rightarrow} E*E \overset{lm}{\Rightarrow} E+E*E \overset{lm}{\Rightarrow} \text{id}+E*E \overset{lm}{\Rightarrow} \text{id}+\text{id}*E \overset{lm}{\Rightarrow} \text{id}+\text{id}*\text{id}$
    This derivation represents the grouping $(\text{id} + \text{id}) * \text{id}$

Ambiguous Grammars

- A grammar G is ambiguous, if $L(G)$ contains a sentence for which there are
  - multiple parse tress, or equivalently
  - multiple leftmost derivations, or equivalently
  - multiple rightmost derivations
- Between the two expressions grammars, G_1 is unambiguous, G_2 is ambiguous

$$G_1 \boxed{\begin{array}{l} E \to E + T \mid T \\ T \to T * F \mid F \\ F \to \text{id} \end{array}} \qquad G_2 \boxed{\begin{array}{l} E \to E + E \\ E \to E * E \\ E \to \text{id} \end{array}}$$

Ambiguity in Expressions Grammar

Grammar	Input
$E \rightarrow E + E$	$\text{id} + \text{id} * \text{id}$
$E \rightarrow E * E$	
$E \rightarrow \text{id}$	

Ambiguity in Expressions Grammar

Grammar	Input
$E \rightarrow E + E$ $E \rightarrow E * E$ $E \rightarrow \text{id}$	$\text{id} + \text{id} * \text{id}$

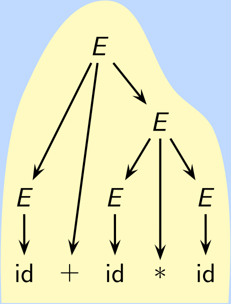
$$
\begin{aligned}
E &\overset{lm}{\Rightarrow} E + E \\
&\overset{lm}{\Rightarrow} \text{id} + E \\
&\overset{lm}{\Rightarrow} \text{id} + E * E \\
&\overset{lm}{\Rightarrow} \text{id} + \text{id} * E \\
&\overset{lm}{\Rightarrow} \text{id} + \text{id} * \text{id}
\end{aligned}
$$

Ambiguity in Expressions Grammar

Grammar	Input
$E \rightarrow E + E$	$\text{id} + \text{id} * \text{id}$
$E \rightarrow E * E$	
$E \rightarrow \text{id}$	

$\text{id} + (\text{id} * \text{id})$

$$
\begin{aligned}
E &\overset{lm}{\Rightarrow} E + E \\
&\overset{lm}{\Rightarrow} \text{id} + E \\
&\overset{lm}{\Rightarrow} \text{id} + E * E \\
&\overset{lm}{\Rightarrow} \text{id} + \text{id} * E \\
&\overset{lm}{\Rightarrow} \text{id} + \text{id} * \text{id}
\end{aligned}
$$

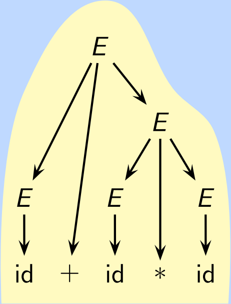


Ambiguity in Expressions Grammar

Grammar	Input
$E \rightarrow E + E$	$\text{id} + \text{id} * \text{id}$
$E \rightarrow E * E$	
$E \rightarrow \text{id}$	

$\text{id} + (\text{id} * \text{id})$

$$\begin{aligned} E &\overset{lm}{\Rightarrow} E + E \\ &\overset{lm}{\Rightarrow} \text{id} + E \\ &\overset{lm}{\Rightarrow} \text{id} + E * E \\ &\overset{lm}{\Rightarrow} \text{id} + \text{id} * E \\ &\overset{lm}{\Rightarrow} \text{id} + \text{id} * \text{id} \end{aligned}$$



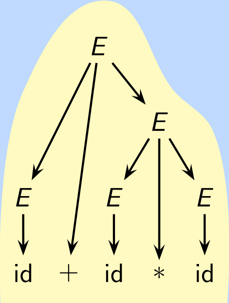
$$\begin{aligned} E &\overset{lm}{\Rightarrow} E * E \\ &\overset{lm}{\Rightarrow} E + E * E \\ &\overset{lm}{\Rightarrow} \text{id} + E * E \\ &\overset{lm}{\Rightarrow} \text{id} + \text{id} * E \\ &\overset{lm}{\Rightarrow} \text{id} + \text{id} * \text{id} \end{aligned}$$

Ambiguity in Expressions Grammar

Grammar	Input
$E \rightarrow E+E$ $E \rightarrow E*E$ $E \rightarrow \text{id}$	$\text{id} + \text{id} * \text{id}$

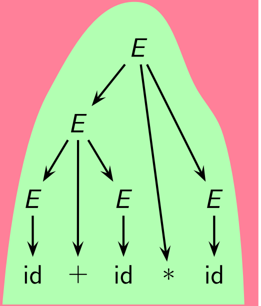
$\text{id} + (\text{id} * \text{id})$

$$
\begin{aligned}
E &\overset{lm}{\Rightarrow} E + E \\
&\overset{lm}{\Rightarrow} \text{id} + E \\
&\overset{lm}{\Rightarrow} \text{id} + E * E \\
&\overset{lm}{\Rightarrow} \text{id} + \text{id} * E \\
&\overset{lm}{\Rightarrow} \text{id} + \text{id} * \text{id}
\end{aligned}
$$



$(\text{id} + \text{id}) * \text{id}$

$$
\begin{aligned}
E &\overset{lm}{\Rightarrow} E * E \\
&\overset{lm}{\Rightarrow} E + E * E \\
&\overset{lm}{\Rightarrow} \text{id} + E * E \\
&\overset{lm}{\Rightarrow} \text{id} + \text{id} * E \\
&\overset{lm}{\Rightarrow} \text{id} + \text{id} * \text{id}
\end{aligned}
$$



Disambiguating Expressions Grammar

- Option 1: Choose the right derivation during parsing

  Specify the following in the yacc script

  - Give higher precedence to $*$ than $+$
  - Make both $+$ and $*$ as left-associative

- Option 2: Rewrite the grammar to use the same rules as above

$$\boxed{\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}}$$

  - Since "$*$" is buried inside T, rule $E \rightarrow E + T$ gives higher precedence to "$*$"
  - Since rule $E \rightarrow E + T$ is left-recursive, it makes "$+$", left-associative
  - Since rule $T \rightarrow T * F$ is left-recursive, it makes "$*$", left-associative

Left recursion: When a non-terminal appears as the leftmost symbol in one of its own production rules. For example, in E -> E + T, the non-terminal E appears at the leftmost position on the right side.
Right recursion: When a non-terminal appears as the rightmost symbol in one of its own production rules. For example, a rule like E -> T + E would be right-recursive.

Ambiguity in IF-ELSE Grammar

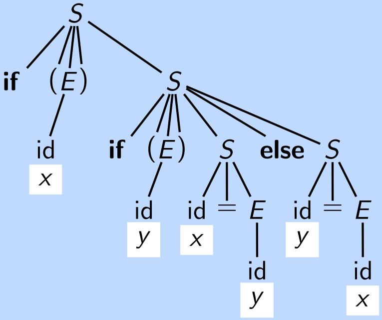
$S \rightarrow \textbf{if}\ (E)\ S\ \textbf{else}\ \ S$

$S \rightarrow \textbf{if}\ (E)\ S$

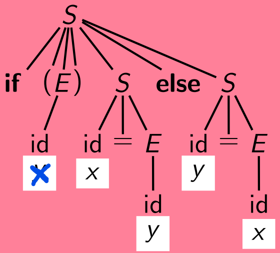
$S \rightarrow \text{id} = E$

E->id

Consider Sentence $\textbf{if}\ (x)\ \textbf{if}\ (y)\ x = y\ \textbf{else}\ \ y = x$



$\textbf{if}\ (x)\ \boxed{\textbf{if}\ (y)\ x = y\ \textbf{else}\ y = x}$



S->if(E)S
S->id=E

$\textbf{if}\ (x)\ \boxed{\textbf{if}\ (y)\ x = y}\ \textbf{else}\ y = x$

Disambiguating IF-ELSE Grammar

- Common rule followed by programming languages

  Every **else** must belong to the closest unmatched **if**
- Option 1: Give higher precedence to **else** than ")"

  Question: What associativities should we use?
- Option 2: Rewrite the grammar by defining matchedIF and unmatchedIF statements

$$\boxed{\begin{array}{rcl} S & \rightarrow & \text{matchedIF} \mid \text{unmatchedIF} \mid \text{id} = E \\ \text{matchedIF} & \rightarrow & \textbf{if}\ (E)\ \text{matchedIF}\ \textbf{else}\ \text{matchedIF} \\ \text{unmatchedIF} & \rightarrow & \textbf{if}\ (E)\ S \\ \text{unmatchedIF} & \rightarrow & \textbf{if}\ (E)\ \text{matchedIF}\ \textbf{else}\ \text{unmatchedIF} \end{array}}$$

Intuition: When **if** and **else** are derived from the same production, the parse tree between them should not have an unmatched **if**

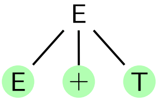
Rightmost Derivation for Bottom-Up Parsing

Grammar

$$\boxed{\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

Rightmost Derivation	Rightmost Derivation in Reverse
$E \overset{rm}{\Rightarrow} E + T$	

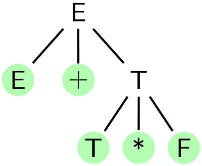


Rightmost Derivation for Bottom-Up Parsing

Grammar
$$\boxed{\begin{array}{l} E \rightarrow E+T \mid T \\ T \rightarrow T*F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

Rightmost Derivation	Rightmost Derivation in Reverse
$E \overset{rm}{\Rightarrow} E + T$ $\overset{rm}{\Rightarrow} E + T * F$	



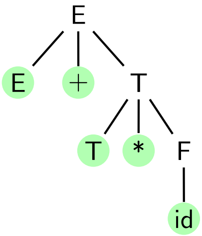
Rightmost Derivation for Bottom-Up Parsing

Grammar
$$\boxed{\begin{array}{l} E \rightarrow E+T \mid T \\ T \rightarrow T*F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

Rightmost Derivation	Rightmost Derivation in Reverse

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E+T \\ &\overset{rm}{\Rightarrow} E+T*F \\ &\overset{rm}{\Rightarrow} E+T*\text{id} \end{aligned}$$



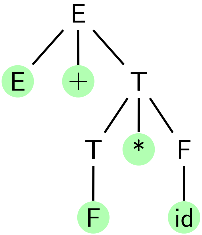
Rightmost Derivation for Bottom-Up Parsing

Grammar
$$\boxed{\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

Rightmost Derivation	Rightmost Derivation in Reverse

$$\begin{array}{ll} E & \overset{rm}{\Rightarrow} E + T \\ & \overset{rm}{\Rightarrow} E + T * F \\ & \overset{rm}{\Rightarrow} E + T * \text{id} \\ & \overset{rm}{\Rightarrow} E + F * \text{id} \end{array}$$



IIT Bombay
cs302: Implementation of Programming Languages

Topic:
Syntax Analysis
Section:
Grammars, Derivations, and Parse Trees
Shift Reduce Parsing
SLR(1) Parsing
Conceptual Issues in Parsing
CLR(1) Parsing
LALR(1) Parsing

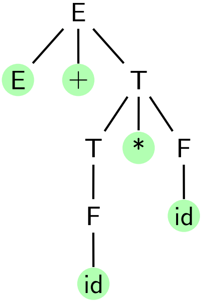
Rightmost Derivation for Bottom-Up Parsing

Grammar
$$\boxed{\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

Rightmost Derivation	Rightmost Derivation in Reverse

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + T \\ &\overset{rm}{\Rightarrow} E + T * F \\ &\overset{rm}{\Rightarrow} E + T * \text{id} \\ &\overset{rm}{\Rightarrow} E + F * \text{id} \\ &\overset{rm}{\Rightarrow} E + \text{id} * \text{id} \end{aligned}$$



Rightmost Derivation for Bottom-Up Parsing

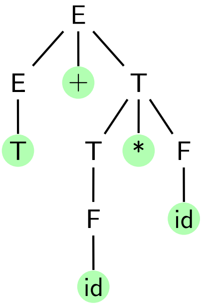
Grammar

$$\boxed{\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

Rightmost Derivation | Rightmost Derivation in Reverse

$$\begin{array}{ll} E & \overset{rm}{\Rightarrow} E + T \\ & \overset{rm}{\Rightarrow} E + T * F \\ & \overset{rm}{\Rightarrow} E + T * \text{id} \\ & \overset{rm}{\Rightarrow} E + F * \text{id} \\ & \overset{rm}{\Rightarrow} E + \text{id} * \text{id} \\ & \overset{rm}{\Rightarrow} T + \text{id} * \text{id} \end{array}$$



Rightmost Derivation for Bottom-Up Parsing

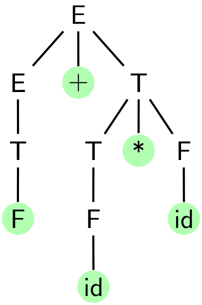
Grammar

$$\boxed{\begin{array}{l} E \rightarrow E+T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

Rightmost Derivation	Rightmost Derivation in Reverse

$$\begin{array}{ll} E & \overset{rm}{\Rightarrow} E+T \\ & \overset{rm}{\Rightarrow} E+T*F \\ & \overset{rm}{\Rightarrow} E+T*\text{id} \\ & \overset{rm}{\Rightarrow} E+F*\text{id} \\ & \overset{rm}{\Rightarrow} E+\text{id}*\text{id} \\ & \overset{rm}{\Rightarrow} T+\text{id}*\text{id} \\ & \overset{rm}{\Rightarrow} F+\text{id}*\text{id} \end{array}$$



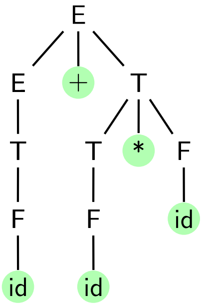
Rightmost Derivation for Bottom-Up Parsing

Grammar
$$\boxed{\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

Rightmost Derivation	Rightmost Derivation in Reverse

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + T \\ &\overset{rm}{\Rightarrow} E + T * F \\ &\overset{rm}{\Rightarrow} E + T * \text{id} \\ &\overset{rm}{\Rightarrow} E + F * \text{id} \\ &\overset{rm}{\Rightarrow} E + \text{id} * \text{id} \\ &\overset{rm}{\Rightarrow} T + \text{id} * \text{id} \\ &\overset{rm}{\Rightarrow} F + \text{id} * \text{id} \\ &\overset{rm}{\Rightarrow} \text{id} + \text{id} * \text{id} \end{aligned}$$



Rightmost Derivation for Bottom-Up Parsing

Grammar

$$\boxed{\begin{array}{l} E \rightarrow E+T \mid T \\ T \rightarrow T*F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: id + id ∗ id

Rightmost Derivation	Rightmost Derivation in Reverse

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E+T \\ &\overset{rm}{\Rightarrow} E+T*F \\ &\overset{rm}{\Rightarrow} E+T*\text{id} \\ &\overset{rm}{\Rightarrow} E+F*\text{id} \\ &\overset{rm}{\Rightarrow} E+\text{id}*\text{id} \\ &\overset{rm}{\Rightarrow} T+\text{id}*\text{id} \\ &\overset{rm}{\Rightarrow} F+\text{id}*\text{id} \\ &\overset{rm}{\Rightarrow} \text{id}+\text{id}*\text{id} \end{aligned}$$

Top-down right-to-left construction of parse tree (need reading the entire input before any action)

Rightmost Derivation for Bottom-Up Parsing

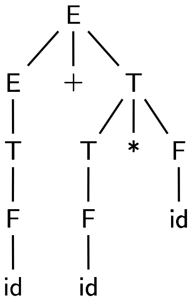
Grammar

$$\boxed{\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

Rightmost Derivation	Rightmost Derivation in Reverse

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + T \\ &\overset{rm}{\Rightarrow} E + T * F \\ &\overset{rm}{\Rightarrow} E + T * \text{id} \\ &\overset{rm}{\Rightarrow} E + F * \text{id} \\ &\overset{rm}{\Rightarrow} E + \text{id} * \text{id} \\ &\overset{rm}{\Rightarrow} T + \text{id} * \text{id} \\ &\overset{rm}{\Rightarrow} F + \text{id} * \text{id} \\ &\overset{rm}{\Rightarrow} \text{id} + \text{id} * \text{id} \end{aligned}$$



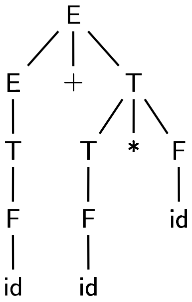
Rightmost Derivation for Bottom-Up Parsing

Grammar
$$\boxed{\begin{array}{l} E \rightarrow E+T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

Rightmost Derivation	Rightmost Derivation in Reverse

$$\begin{array}{l} E \overset{rm}{\Rightarrow} E + T \\ \quad \overset{rm}{\Rightarrow} E + T * F \\ \quad \overset{rm}{\Rightarrow} E + T * \text{id} \\ \quad \overset{rm}{\Rightarrow} E + F * \text{id} \\ \quad \overset{rm}{\Rightarrow} E + \text{id} * \text{id} \\ \quad \overset{rm}{\Rightarrow} T + \text{id} * \text{id} \\ \quad \overset{rm}{\Rightarrow} F + \text{id} * \text{id} \\ \quad \overset{rm}{\Rightarrow} \text{id} + \text{id} * \text{id} \end{array}$$



$\text{id} + \text{id} * \text{id}$

id + id * id

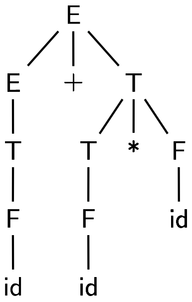
Rightmost Derivation for Bottom-Up Parsing

Grammar
$$\boxed{\begin{array}{l} E \rightarrow E+T \mid T \\ T \rightarrow T*F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

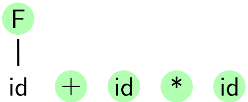
Rightmost Derivation

$$\begin{array}{rl} E & \overset{rm}{\Rightarrow} E+T \\ & \overset{rm}{\Rightarrow} E+T*F \\ & \overset{rm}{\Rightarrow} E+T*\text{id} \\ & \overset{rm}{\Rightarrow} E+F*\text{id} \\ & \overset{rm}{\Rightarrow} E+\text{id}*\text{id} \\ & \overset{rm}{\Rightarrow} T+\text{id}*\text{id} \\ & \overset{rm}{\Rightarrow} F+\text{id}*\text{id} \\ & \overset{rm}{\Rightarrow} \text{id}+\text{id}*\text{id} \end{array}$$



Rightmost Derivation in Reverse

$$\begin{array}{l} \text{id} + \text{id} * \text{id} \\ \overset{rm}{\Rightarrow} F + \text{id} * \text{id} \end{array}$$



Rightmost Derivation for Bottom-Up Parsing

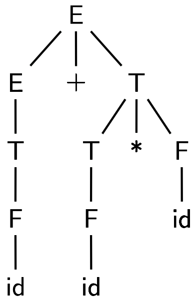
Grammar

$$\boxed{\begin{array}{l} E \rightarrow E+T \mid T \\ T \rightarrow T*F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

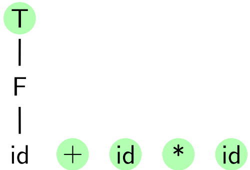
Rightmost Derivation

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E+T \\ &\overset{rm}{\Rightarrow} E+T*F \\ &\overset{rm}{\Rightarrow} E+T*\text{id} \\ &\overset{rm}{\Rightarrow} E+F*\text{id} \\ &\overset{rm}{\Rightarrow} E+\text{id}*\text{id} \\ &\overset{rm}{\Rightarrow} T+\text{id}*\text{id} \\ &\overset{rm}{\Rightarrow} F+\text{id}*\text{id} \\ &\overset{rm}{\Rightarrow} \text{id}+\text{id}*\text{id} \end{aligned}$$



Rightmost Derivation in Reverse

$$\begin{aligned} &\text{id} + \text{id} * \text{id} \\ &\overset{rm}{\Rightarrow} F + \text{id} * \text{id} \\ &\overset{rm}{\Rightarrow} T + \text{id} * \text{id} \end{aligned}$$



Rightmost Derivation for Bottom-Up Parsing

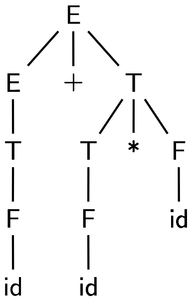
Grammar

$$\boxed{\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

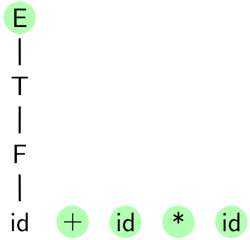
Rightmost Derivation

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + T \\ &\overset{rm}{\Rightarrow} E + T * F \\ &\overset{rm}{\Rightarrow} E + T * \text{id} \\ &\overset{rm}{\Rightarrow} E + F * \text{id} \\ &\overset{rm}{\Rightarrow} E + \text{id} * \text{id} \\ &\overset{rm}{\Rightarrow} T + \text{id} * \text{id} \\ &\overset{rm}{\Rightarrow} F + \text{id} * \text{id} \\ &\overset{rm}{\Rightarrow} \text{id} + \text{id} * \text{id} \end{aligned}$$



Rightmost Derivation in Reverse

$$\begin{aligned} &\text{id} + \text{id} * \text{id} \\ &\overset{rm}{\Rightarrow} F + \text{id} * \text{id} \\ &\overset{rm}{\Rightarrow} T + \text{id} * \text{id} \\ &\overset{rm}{\Rightarrow} E + \text{id} * \text{id} \end{aligned}$$



Rightmost Derivation for Bottom-Up Parsing

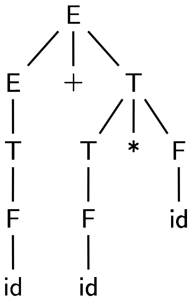
Grammar

$$\boxed{\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

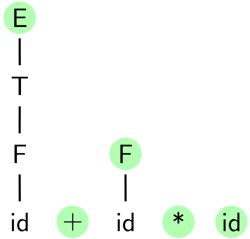
Rightmost Derivation

$$\begin{array}{l} E \overset{rm}{\Rightarrow} E + T \\ \quad \overset{rm}{\Rightarrow} E + T * F \\ \quad \overset{rm}{\Rightarrow} E + T * \text{id} \\ \quad \overset{rm}{\Rightarrow} E + F * \text{id} \\ \quad \overset{rm}{\Rightarrow} E + \text{id} * \text{id} \\ \quad \overset{rm}{\Rightarrow} T + \text{id} * \text{id} \\ \quad \overset{rm}{\Rightarrow} F + \text{id} * \text{id} \\ \quad \overset{rm}{\Rightarrow} \text{id} + \text{id} * \text{id} \end{array}$$



Rightmost Derivation in Reverse

$$\begin{array}{l} \text{id} + \text{id} * \text{id} \\ \overset{rm}{\Rightarrow} F + \text{id} * \text{id} \\ \overset{rm}{\Rightarrow} T + \text{id} * \text{id} \\ \overset{rm}{\Rightarrow} E + \text{id} * \text{id} \\ \overset{rm}{\Rightarrow} E + F * \text{id} \end{array}$$



Rightmost Derivation for Bottom-Up Parsing

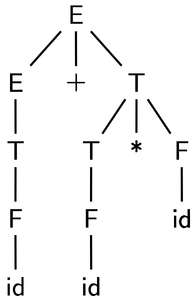
Grammar

$$\boxed{\begin{array}{l} E \rightarrow E+T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

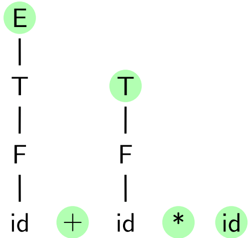
Rightmost Derivation

$$\begin{array}{ll} E & \overset{rm}{\Rightarrow} E+T \\ & \overset{rm}{\Rightarrow} E+T*F \\ & \overset{rm}{\Rightarrow} E+T*\text{id} \\ & \overset{rm}{\Rightarrow} E+F*\text{id} \\ & \overset{rm}{\Rightarrow} E+\text{id}*\text{id} \\ & \overset{rm}{\Rightarrow} T+\text{id}*\text{id} \\ & \overset{rm}{\Rightarrow} F+\text{id}*\text{id} \\ & \overset{rm}{\Rightarrow} \text{id}+\text{id}*\text{id} \end{array}$$



Rightmost Derivation in Reverse

$$\begin{array}{l} \text{id} + \text{id} * \text{id} \\ \overset{rm}{\Rightarrow} F+\text{id}*\text{id} \\ \overset{rm}{\Rightarrow} T+\text{id}*\text{id} \\ \overset{rm}{\Rightarrow} E+\text{id}*\text{id} \\ \overset{rm}{\Rightarrow} E+F*\text{id} \\ \overset{rm}{\Rightarrow} E+T*\text{id} \end{array}$$



Rightmost Derivation for Bottom-Up Parsing

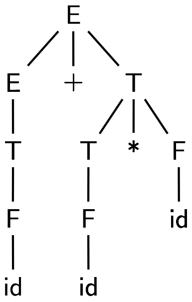
Grammar

$$\boxed{\begin{array}{l} E \rightarrow E+T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

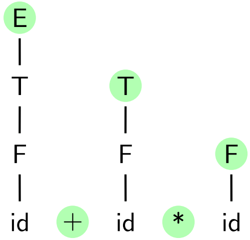
Rightmost Derivation

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E+T \\ &\overset{rm}{\Rightarrow} E+T*F \\ &\overset{rm}{\Rightarrow} E+T*\text{id} \\ &\overset{rm}{\Rightarrow} E+F*\text{id} \\ &\overset{rm}{\Rightarrow} E+\text{id}*\text{id} \\ &\overset{rm}{\Rightarrow} T+\text{id}*\text{id} \\ &\overset{rm}{\Rightarrow} F+\text{id}*\text{id} \\ &\overset{rm}{\Rightarrow} \text{id}+\text{id}*\text{id} \end{aligned}$$



Rightmost Derivation in Reverse

$$\begin{aligned} &\text{id} + \text{id} * \text{id} \\ \overset{rm}{\Rightarrow}\ & F + \text{id} * \text{id} \\ \overset{rm}{\Rightarrow}\ & T + \text{id} * \text{id} \\ \overset{rm}{\Rightarrow}\ & E + \text{id} * \text{id} \\ \overset{rm}{\Rightarrow}\ & E + F * \text{id} \\ \overset{rm}{\Rightarrow}\ & E + T * \text{id} \\ \overset{rm}{\Rightarrow}\ & E + T * F \end{aligned}$$



Rightmost Derivation for Bottom-Up Parsing

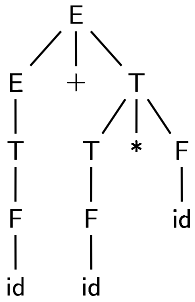
Grammar

$$\boxed{\begin{array}{l} E \rightarrow E+T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

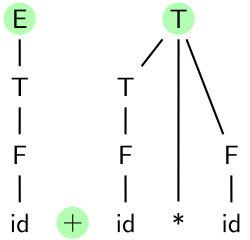
Rightmost Derivation

$$\begin{array}{ll} E & \overset{rm}{\Rightarrow} E+T \\ & \overset{rm}{\Rightarrow} E+T*F \\ & \overset{rm}{\Rightarrow} E+T*\text{id} \\ & \overset{rm}{\Rightarrow} E+F*\text{id} \\ & \overset{rm}{\Rightarrow} E+\text{id}*\text{id} \\ & \overset{rm}{\Rightarrow} T+\text{id}*\text{id} \\ & \overset{rm}{\Rightarrow} F+\text{id}*\text{id} \\ & \overset{rm}{\Rightarrow} \text{id}+\text{id}*\text{id} \end{array}$$



Rightmost Derivation in Reverse

$$\begin{array}{l} \text{id}+\text{id}*\text{id} \\ \overset{rm}{\Rightarrow} F+\text{id}*\text{id} \\ \overset{rm}{\Rightarrow} T+\text{id}*\text{id} \\ \overset{rm}{\Rightarrow} E+\text{id}*\text{id} \\ \overset{rm}{\Rightarrow} E+F*\text{id} \\ \overset{rm}{\Rightarrow} E+T*\text{id} \\ \overset{rm}{\Rightarrow} E+T*F \\ \overset{rm}{\Rightarrow} E+T \end{array}$$



Rightmost Derivation for Bottom-Up Parsing

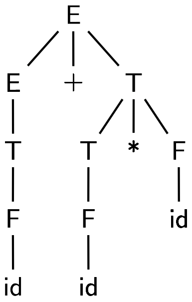
Grammar

$$\boxed{\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

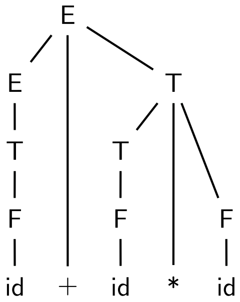
Rightmost Derivation

$$\begin{array}{ll} E & \overset{rm}{\Rightarrow} E + T \\ & \overset{rm}{\Rightarrow} E + T * F \\ & \overset{rm}{\Rightarrow} E + T * \text{id} \\ & \overset{rm}{\Rightarrow} E + F * \text{id} \\ & \overset{rm}{\Rightarrow} E + \text{id} * \text{id} \\ & \overset{rm}{\Rightarrow} T + \text{id} * \text{id} \\ & \overset{rm}{\Rightarrow} F + \text{id} * \text{id} \\ & \overset{rm}{\Rightarrow} \text{id} + \text{id} * \text{id} \end{array}$$



Rightmost Derivation in Reverse

$$\begin{array}{l} \text{id} + \text{id} * \text{id} \\ \overset{rm}{\Rightarrow} F + \text{id} * \text{id} \\ \overset{rm}{\Rightarrow} T + \text{id} * \text{id} \\ \overset{rm}{\Rightarrow} E + \text{id} * \text{id} \\ \overset{rm}{\Rightarrow} E + F * \text{id} \\ \overset{rm}{\Rightarrow} E + T * \text{id} \\ \overset{rm}{\Rightarrow} E + T * F \\ \overset{rm}{\Rightarrow} E + T \\ \overset{rm}{\Rightarrow} E \end{array}$$



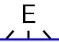
Rightmost Derivation for Bottom-Up Parsing

Grammar
$$\boxed{\begin{array}{l} E \rightarrow E+T \mid T \\ T \rightarrow T*F \mid F \\ F \rightarrow \text{id} \end{array}}$$

Sentence: $\text{id} + \text{id} * \text{id}$

Rightmost Derivation

$$\begin{array}{rl} E & \overset{rm}{\Rightarrow} E+T \\ & \overset{rm}{\Rightarrow} E+T*F \\ & \overset{rm}{\Rightarrow} E+T*\text{id} \\ & \overset{rm}{\Rightarrow} E+F*\text{id} \\ & \overset{rm}{\Rightarrow} E+\text{id}*\text{id} \\ & \overset{rm}{\Rightarrow} T+\text{id}*\text{id} \\ & \overset{rm}{\Rightarrow} F+\text{id}*\text{id} \\ & \overset{rm}{\Rightarrow} \text{id}+\text{id}*\text{id} \end{array}$$

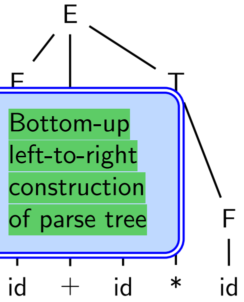


Top-down right-to-left construction of parse tree (need reading the entire input before any action)

Rightmost Derivation in Reverse

note, input string was read one by one

$$\begin{array}{rl} & \text{id}+\text{id}*\text{id} \\ \overset{rm}{\Rightarrow} & F+\text{id}*\text{id} \\ \overset{rm}{\Rightarrow} & T+\text{id}*\text{id} \\ \overset{rm}{\Rightarrow} & E+\text{id}*\text{id} \\ \overset{rm}{\Rightarrow} & E+F*\text{id} \\ \overset{rm}{\Rightarrow} & E+T*\text{id} \\ \overset{rm}{\Rightarrow} & E+T*F \\ \overset{rm}{\Rightarrow} & E+T \\ \overset{rm}{\Rightarrow} & E \end{array}$$



Bottom-up left-to-right construction of parse tree

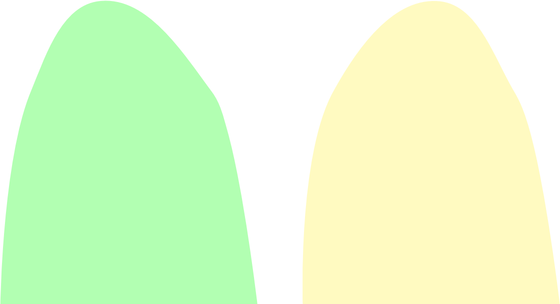
Shift Reduce Parsing

An Overview of Shift Reduce Parsing

Grammar	Input
$E \rightarrow E + E$ $E \rightarrow E * E$ $E \rightarrow \text{id}$	$\text{id} + \text{id} * \text{id}$

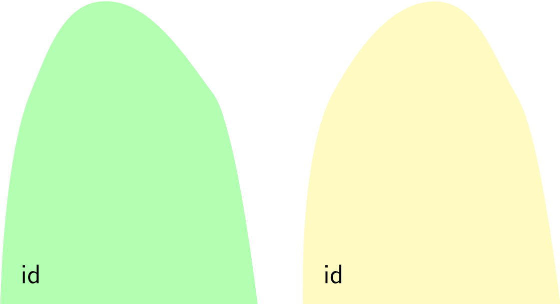
An Overview of Shift Reduce Parsing

Grammar	Input
$E \rightarrow E + E$	id + id $*$ id
$E \rightarrow E * E$	
$E \rightarrow$ id	



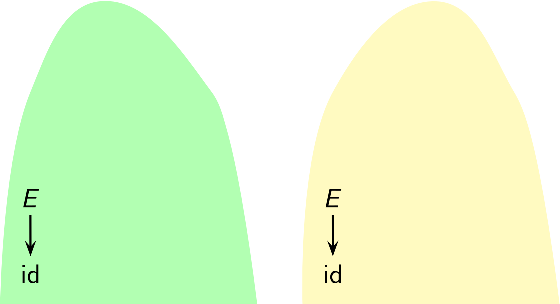
An Overview of Shift Reduce Parsing

Grammar	Input
$E \rightarrow E + E$ $E \rightarrow E * E$ $E \rightarrow \text{id}$	$\text{id} + \text{id} * \text{id}$



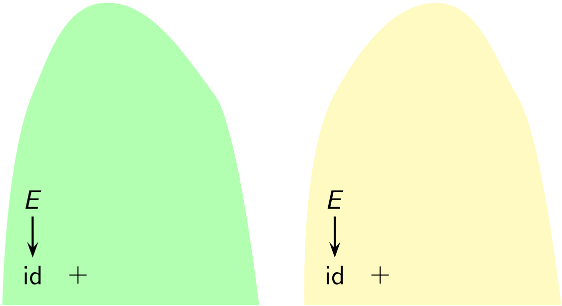
An Overview of Shift Reduce Parsing

Grammar	Input
$E \rightarrow E + E$	$\text{id} + \text{id} * \text{id}$
$E \rightarrow E * E$	
$E \rightarrow \text{id}$	



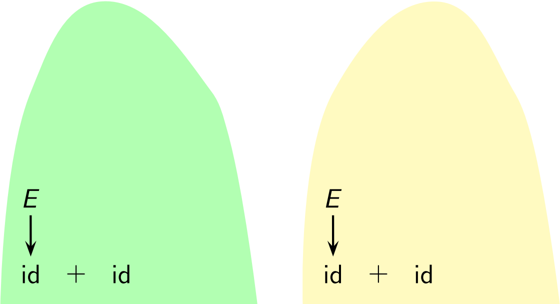
An Overview of Shift Reduce Parsing

Grammar	Input
$E \rightarrow E + E$ $E \rightarrow E * E$ $E \rightarrow \mathrm{id}$	$\mathrm{id} + \mathrm{id} * \mathrm{id}$



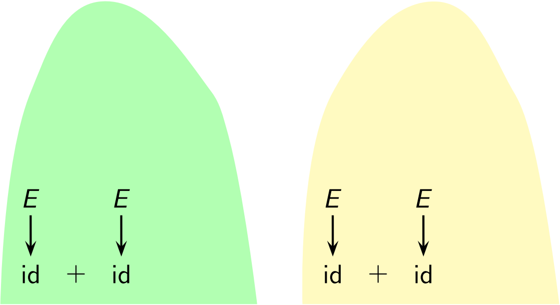
An Overview of Shift Reduce Parsing

Grammar	Input
$E \rightarrow E + E$	$\text{id} + \text{id} * \text{id}$
$E \rightarrow E * E$	
$E \rightarrow \text{id}$	



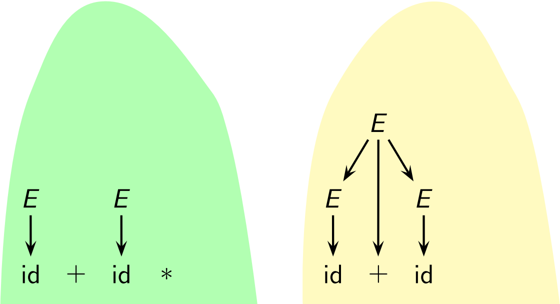
An Overview of Shift Reduce Parsing

Grammar	Input
$E \rightarrow E + E$	$\text{id} + \text{id} * \text{id}$
$E \rightarrow E * E$	
$E \rightarrow \text{id}$	



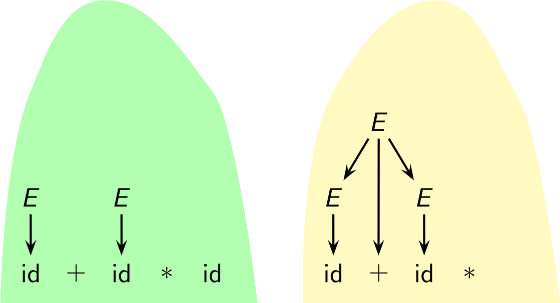
An Overview of Shift Reduce Parsing

Grammar	Input
$E \to E + E$ $E \to E * E$ $E \to \text{id}$	$\text{id} + \text{id} * \text{id}$



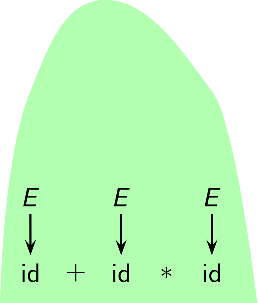
An Overview of Shift Reduce Parsing

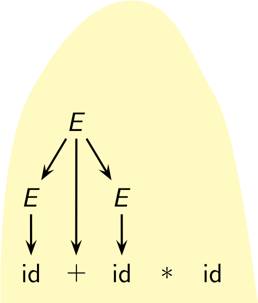
Grammar	Input
$E \rightarrow E + E$	$\text{id} + \text{id} * \text{id}$
$E \rightarrow E * E$	
$E \rightarrow \text{id}$	



An Overview of Shift Reduce Parsing

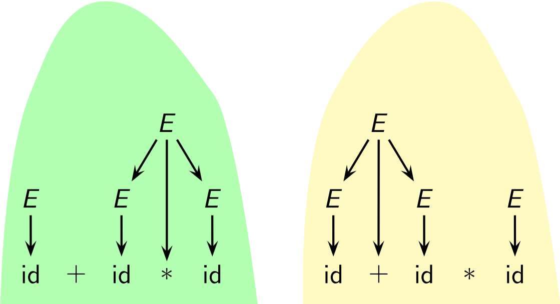
Grammar	Input
$E \rightarrow E + E$	$\text{id} + \text{id} * \text{id}$
$E \rightarrow E * E$	
$E \rightarrow \text{id}$	





An Overview of Shift Reduce Parsing

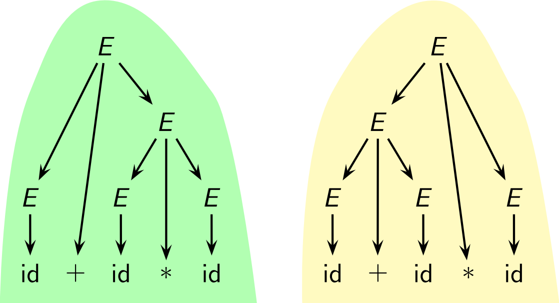
Grammar	Input
$E \rightarrow E + E$ $E \rightarrow E * E$ $E \rightarrow \text{id}$	$\text{id} + \text{id} * \text{id}$



An Overview of Shift Reduce Parsing

Grammar	Input
$E \rightarrow E + E$ $E \rightarrow E * E$ $E \rightarrow \text{id}$	$\text{id} + \text{id} * \text{id}$

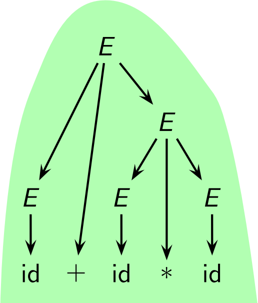
bottom up in ambiguous ways



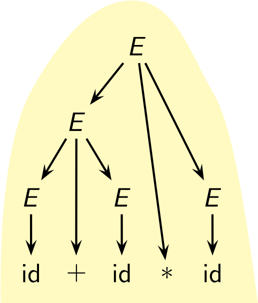
An Overview of Shift Reduce Parsing

Grammar	Input
$E \rightarrow E + E$	$\text{id} + \text{id} * \text{id}$
$E \rightarrow E * E$	
$E \rightarrow \text{id}$	

$\text{id} + (\text{id} * \text{id})$



$(\text{id} + \text{id}) * \text{id}$

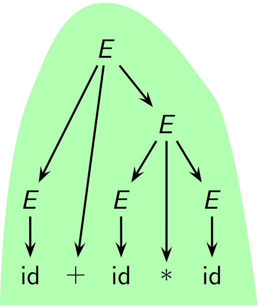


An Overview of Shift Reduce Parsing

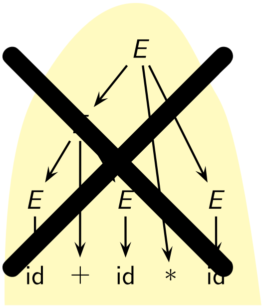
Grammar	Input
$E \rightarrow E + E$	$\text{id} + \text{id} * \text{id}$
$E \rightarrow E * E$	
$E \rightarrow \text{id}$	

lower priority vale pehle apply hote h rule me , then followed by higher priority operaratos, like here, + is used first and * is used later for correct grammar

$\text{id} + (\text{id} * \text{id})$



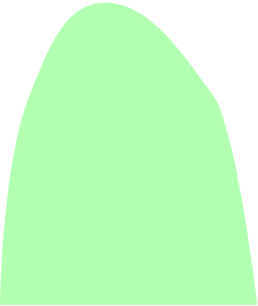
$(\text{id} + \text{id}) * \text{id}$

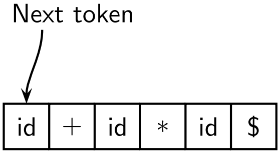


An Overview of Shift Reduce Parsing

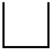
Grammar	Input
$E \rightarrow E + E$	$\text{id} + \text{id} * \text{id}$
$E \rightarrow E * E$	
$E \rightarrow \text{id}$	

$\text{id} + (\text{id} * \text{id})$





Next Action: Shift



Parsing Stack

An Overview of Shift Reduce Parsing

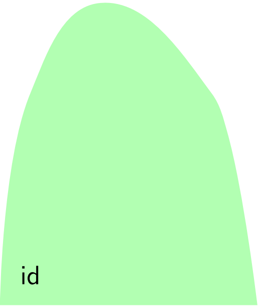
Grammar

$E \rightarrow E + E$
$E \rightarrow E * E$
$E \rightarrow \text{id}$

Input

$\text{id} + \text{id} * \text{id}$

$\text{id} + (\text{id} * \text{id})$



Next token

$+$	id	$*$	id	\$

Next Action: Reduce by $E \rightarrow \text{id}$

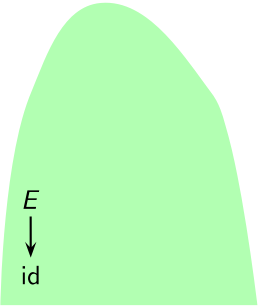
id

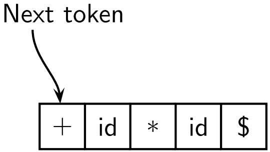
Parsing Stack

An Overview of Shift Reduce Parsing

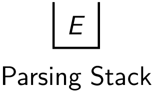
Grammar	Input
$E \rightarrow E + E$	$\text{id} + \text{id} * \text{id}$
$E \rightarrow E * E$	
$E \rightarrow \text{id}$	

$\text{id} + (\text{id} * \text{id})$



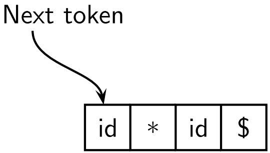


Next Action: Shift

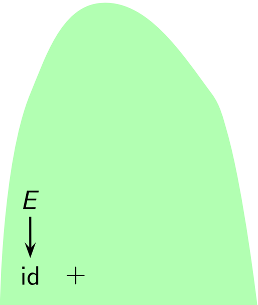


An Overview of Shift Reduce Parsing

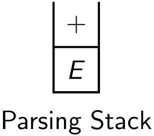
Grammar	Input
$E \rightarrow E + E$	$\text{id} + \text{id} * \text{id}$
$E \rightarrow E * E$	
$E \rightarrow \text{id}$	



$\text{id} + (\text{id} * \text{id})$



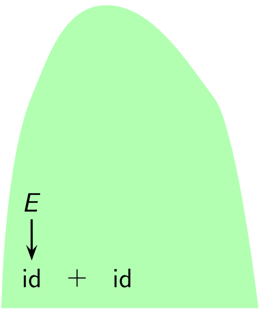
Next Action: Shift

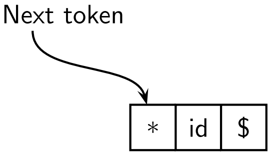


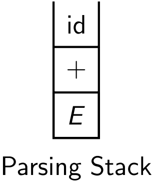
An Overview of Shift Reduce Parsing

Grammar	Input
$E \rightarrow E + E$ $E \rightarrow E * E$ $E \rightarrow \text{id}$	$\text{id} + \text{id} * \text{id}$

$\text{id} + (\text{id} * \text{id})$



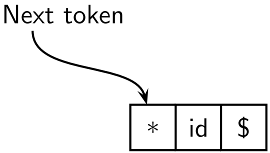




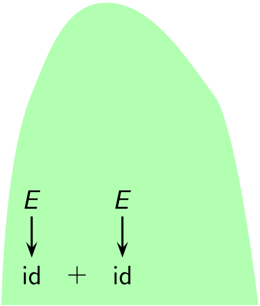
Next Action: Reduce by $E \rightarrow \text{id}$

An Overview of Shift Reduce Parsing

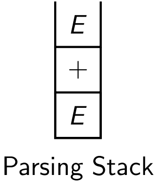
Grammar	Input
$E \rightarrow E + E$ $E \rightarrow E * E$ $E \rightarrow \text{id}$	$\text{id} + \text{id} * \text{id}$



$\text{id} + (\text{id} * \text{id})$



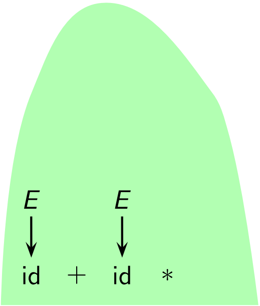
Next Action: Shift

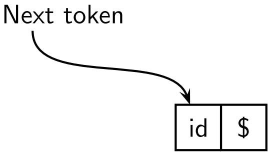


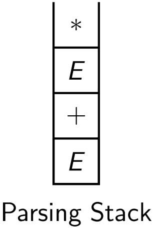
An Overview of Shift Reduce Parsing

Grammar	Input
$E \rightarrow E + E$	id $+$ id $*$ id
$E \rightarrow E * E$	
$E \rightarrow$ id	

id $+$ (id $*$ id)



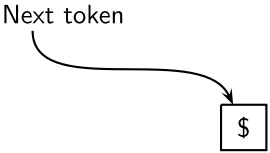




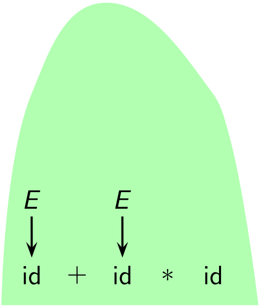
Next Action: Shift

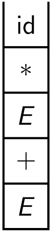
An Overview of Shift Reduce Parsing

Grammar	Input
$E \rightarrow E + E$	$\text{id} + \text{id} * \text{id}$
$E \rightarrow E * E$	
$E \rightarrow \text{id}$	



$\text{id} + (\text{id} * \text{id})$





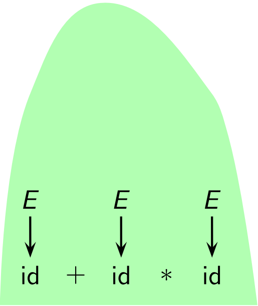
Parsing Stack

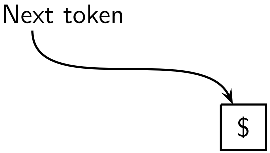
Next Action: Reduce by $E \rightarrow \text{id}$

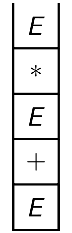
An Overview of Shift Reduce Parsing

Grammar	Input
$E \to E + E$	$\text{id} + \text{id} * \text{id}$
$E \to E * E$	
$E \to \text{id}$	

$\text{id} + (\text{id} * \text{id})$




Next token



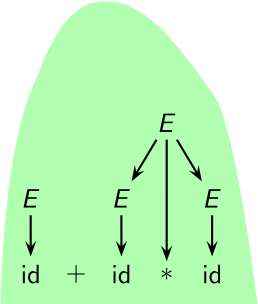
Parsing Stack

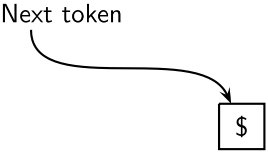
Next Action: Reduce by $E \to E * E$

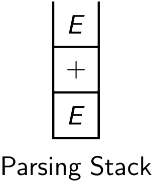
An Overview of Shift Reduce Parsing

Grammar	Input
$E \rightarrow E+E$	$\text{id} + \text{id} * \text{id}$
$E \rightarrow E * E$	
$E \rightarrow \text{id}$	

$\text{id} + (\text{id} * \text{id})$





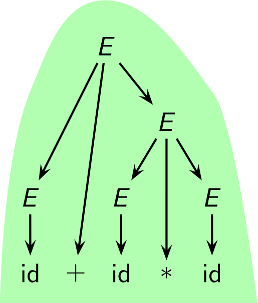


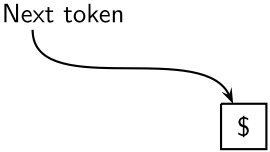
Next Action: Reduce by $E \rightarrow E + E$

An Overview of Shift Reduce Parsing

Grammar	Input
$E \rightarrow E + E$	$\text{id} + \text{id} * \text{id}$
$E \rightarrow E * E$	
$E \rightarrow \text{id}$	

$\text{id} + (\text{id} * \text{id})$





Next Action: Accept

start state 'E' pe vapas aa gya

Parsing Stack

An Overview of Shift Reduce Parsing

Grammar	Input
1. $E \rightarrow E + E$ 2. $E \rightarrow E * E$ 3. $E \rightarrow \text{id}$	$\text{id} + \text{id} * \text{id}$

An Overview of Shift Reduce Parsing

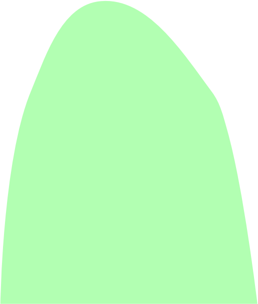
Grammar

1. $E \rightarrow E + E$
2. $E \rightarrow E * E$
3. $E \rightarrow \text{id}$

Input

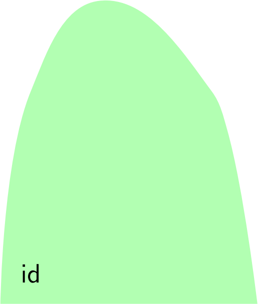
$\text{id} + \text{id} * \text{id}$

Step	Stack →	Input	Action
1	\$	$\text{id} + \text{id} * \text{id}\$$	shift



An Overview of Shift Reduce Parsing

Grammar	Input
1. $E \rightarrow E+E$ 2. $E \rightarrow E*E$ 3. $E \rightarrow \text{id}$	$\text{id} + \text{id} * \text{id}$



Step	Stack →	Input	Action
1	\$	$\text{id} + \text{id} * \text{id}\$$	shift
2	\$id	$+ \text{id} * \text{id}\$$	reduce by 3

An Overview of Shift Reduce Parsing

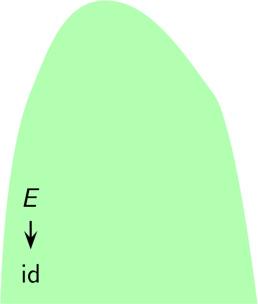
Grammar

1. $E \rightarrow E + E$
2. $E \rightarrow E * E$
3. $E \rightarrow \text{id}$

Input

$\text{id} + \text{id} * \text{id}$

Step	Stack →	Input	Action
1	\$	$\text{id} + \text{id} * \text{id}\$$	shift
2	\$id	$+ \text{id} * \text{id}\$$	reduce by 3
3	$\$E$	$+ \text{id} * \text{id}\$$	shift



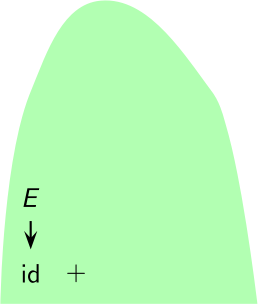
An Overview of Shift Reduce Parsing

Grammar

1. $E \rightarrow E + E$
2. $E \rightarrow E * E$
3. $E \rightarrow \text{id}$

Input

$\text{id} + \text{id} * \text{id}$



Step	Stack →	Input	Action
1	\$	id + id ∗ id\$	shift
2	\$id	+ id ∗ id\$	reduce by 3
3	\$$E$	+ id ∗ id\$	shift
4	\$$E$ +	id ∗ id\$	shift

An Overview of Shift Reduce Parsing

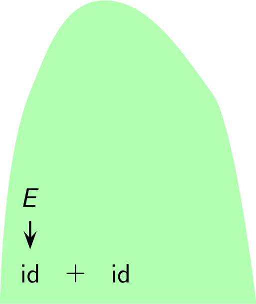
Grammar

1. $E \rightarrow E + E$
2. $E \rightarrow E * E$
3. $E \rightarrow \text{id}$

Input

$\text{id} + \text{id} * \text{id}$

Step	Stack →	Input	Action
1	$	$\text{id} + \text{id} * \text{id}\$$	shift
2	$\$\text{id}$	$+ \text{id} * \text{id}\$$	reduce by 3
3	$\$E$	$+ \text{id} * \text{id}\$$	shift
4	$\$E +$	$\text{id} * \text{id}\$$	shift
5	$\$E + \text{id}$	$* \text{id}\$$	reduce by 3



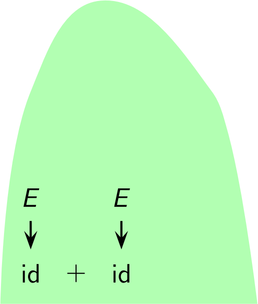
An Overview of Shift Reduce Parsing

Grammar

1. $E \rightarrow E + E$
2. $E \rightarrow E * E$
3. $E \rightarrow \text{id}$

Input

$\text{id} + \text{id} * \text{id}$



Step	Stack →	Input	Action
1	$\$$	$\text{id} + \text{id} * \text{id}\$$	shift
2	$\$\text{id}$	$+ \text{id} * \text{id}\$$	reduce by 3
3	$\$E$	$+ \text{id} * \text{id}\$$	shift
4	$\$E +$	$\text{id} * \text{id}\$$	shift
5	$\$E + \text{id}$	$* \text{id}\$$	reduce by 3
6	$\$E + E$	$* \text{id}\$$	shift

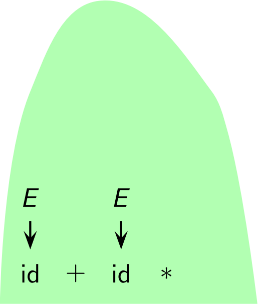
An Overview of Shift Reduce Parsing

Grammar

1. $E \rightarrow E + E$
2. $E \rightarrow E * E$
3. $E \rightarrow \text{id}$

Input

$\text{id} + \text{id} * \text{id}$



Step	Stack →	Input	Action
1	\$	$\text{id} + \text{id} * \text{id}\$$	shift
2	\$id	$+ \text{id} * \text{id}\$$	reduce by 3
3	$\$E$	$+ \text{id} * \text{id}\$$	shift
4	$\$E +$	$\text{id} * \text{id}\$$	shift
5	$\$E + \text{id}$	$* \text{id}\$$	reduce by 3
6	$\$E + E$	$* \text{id}\$$	shift
7	$\$E + E *$	$\text{id}\$$	shift

An Overview of Shift Reduce Parsing

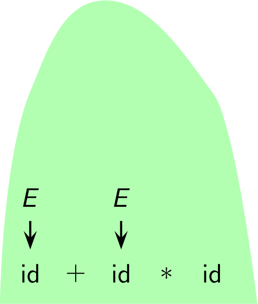
Grammar

1. $E \rightarrow E + E$
2. $E \rightarrow E * E$
3. $E \rightarrow \text{id}$

Input

$\text{id} + \text{id} * \text{id}$

Step	Stack →	Input	Action
1	$	id + id ∗ id$	shift
2	$id	+ id ∗ id$	reduce by 3
3	$E	+ id ∗ id$	shift
4	$E +	id ∗ id$	shift
5	$E + id	∗ id$	reduce by 3
6	$E + E	∗ id$	shift
7	$E + E ∗	id$	shift
8	$E + E ∗ id	$	reduce by 3



E E

id + id ∗ id

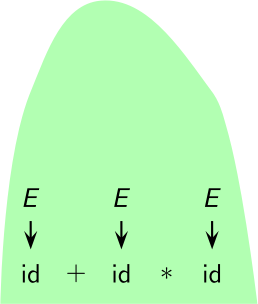
An Overview of Shift Reduce Parsing

Grammar

1. $E \rightarrow E + E$
2. $E \rightarrow E * E$
3. $E \rightarrow \text{id}$

Input

$\text{id} + \text{id} * \text{id}$



Step	Stack →	Input	Action
1	$	$\text{id} + \text{id} * \text{id}\$$	shift
2	$\$\text{id}$	$+ \text{id} * \text{id}\$$	reduce by 3
3	$\$E$	$+ \text{id} * \text{id}\$$	shift
4	$\$E +$	$\text{id} * \text{id}\$$	shift
5	$\$E + \text{id}$	$* \text{id}\$$	reduce by 3
6	$\$E + E$	$* \text{id}\$$	shift
7	$\$E + E *$	$\text{id}\$$	shift
8	$\$E + E * \text{id}$	$	reduce by 3
9	$\$E + E * E$	$	reduce by 2

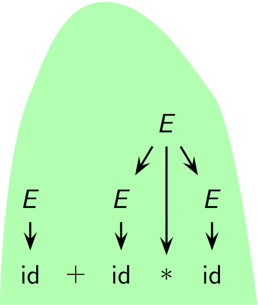
An Overview of Shift Reduce Parsing

Grammar

1. $E \rightarrow E + E$
2. $E \rightarrow E * E$
3. $E \rightarrow \text{id}$

Input

$\text{id} + \text{id} * \text{id}$



Step	Stack →	Input	Action
1	\$	$\text{id} + \text{id} * \text{id}\$$	shift
2	\$id	$+ \text{id} * \text{id}\$$	reduce by 3
3	$\$E$	$+ \text{id} * \text{id}\$$	shift
4	$\$E +$	$\text{id} * \text{id}\$$	shift
5	$\$E + \text{id}$	$* \text{id}\$$	reduce by 3
6	$\$E + E$	$* \text{id}\$$	shift
7	$\$E + E *$	$\text{id}\$$	shift
8	$\$E + E * \text{id}$	\$	reduce by 3
9	$\$E + E * E$	\$	reduce by 2
10	$\$E + E$	\$	reduce by 1

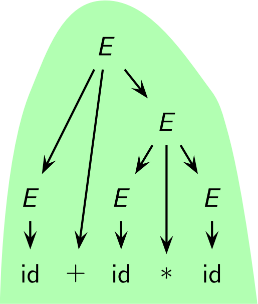
An Overview of Shift Reduce Parsing

Grammar

1. $E \to E + E$
2. $E \to E * E$
3. $E \to \text{id}$

Input

$\text{id} + \text{id} * \text{id}$



$ = end of input pe
'E' , start state aa
gya to accepted

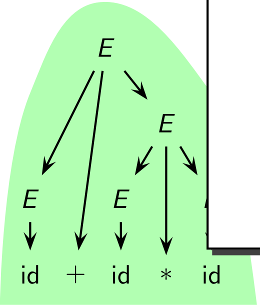
Step	Stack →	Input	Action
1	$\$$	$\text{id} + \text{id} * \text{id}\$$	shift
2	$\$\text{id}$	$+ \text{id} * \text{id}\$$	reduce by 3
3	$\$E$	$+ \text{id} * \text{id}\$$	shift
4	$\$E +$	$\text{id} * \text{id}\$$	shift
5	$\$E + \text{id}$	$* \text{id}\$$	reduce by 3
6	$\$E + E$	$* \text{id}\$$	shift
7	$\$E + E *$	$\text{id}\$$	shift
8	$\$E + E * \text{id}$	$\$$	reduce by 3
9	$\$E + E * E$	$\$$	reduce by 2
10	$\$E + E$	$\$$	reduce by 1
11	$\$E$	$\$$	accept

An Overview of Shift Reduce Parsing

Grammar

1. $E \rightarrow E + E$
2. $E \rightarrow E * E$
3. $E \rightarrow \text{id}$

Input



Observations

- A shift corresponds to creating a leaf node in the parse tree whereas a reduce corresponds to creating an internal node
- In every step i, concatenation of the stack and the remaining input gives a right sentential form (rsf_i)
- For every step i, $\text{rsf}_{i+1} \overset{rm}{\Rightarrow} \text{rsf}_i$
- In every step, the partial parse tree constructed until then, consists of a forest of trees
- In every step, the stack holds the root nodes of the trees contained in the forest

  A reduce action may amount to joining some of these trees

10	$E + E	$	reduce by 1
11	$E	$	accept

Shift Reduce Parsing

- In every step i, concatenation of the stack and the remaining input gives a right sentential form (rsf_i)
- For every step i, $\text{rsf}_{i+1} \overset{rm}{\Rightarrow} \text{rsf}_i$
- How do we go from rsf_i to rsf_{i+1}?
    - $S \overset{*rm}{\Rightarrow} \alpha\ A\ w \overset{rm}{\Rightarrow} \alpha\ \beta\ w$
    - A bottom-up parser reduces β occurring in $\alpha\beta w$ to A using the production $A \rightarrow \beta$
    - The rule $A \rightarrow \beta$ and the occurrence of β is the handle in $\alpha\beta w$

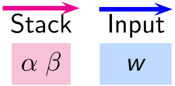
Shift Reduce Parsing

- Bottom up parsing is essentially the process of detecting handles and reducing them
- Different bottom-up parsers differ in the way they detect handles

Why do Handles Form the Basis of Bottom Up Parsing?

- Only terminal symbols can appear to the right of a handle in a rightmost sentential form

  Why?
  - $S \overset{*rm}{\Rightarrow} \alpha\ A\ w \overset{rm}{\Rightarrow} \alpha\ \beta\ w$
  - Since we are using a rightmost derivation, there cannot be a non-terminal to the right of A.
- The beauty of bottom up parsing lies in dividing a right sentential form $\alpha\beta w$ into two parts



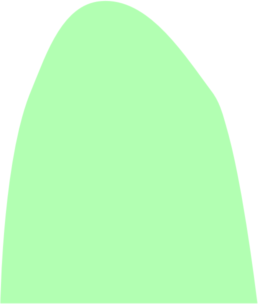
such that the handle *always* appears on the top of the stack

Why do Handles Form the Basis of Bottom Up Parsing?

Grammar	Input
1. $E \rightarrow E + E$ 2. $E \rightarrow E * E$ 3. $E \rightarrow \text{id}$	$\text{id} + \text{id} * \text{id}$

Why do Handles Form the Basis of Bottom Up Parsing?

Grammar	Input
1. $E \rightarrow E + E$ 2. $E \rightarrow E * E$ 3. $E \rightarrow \text{id}$	$\text{id} + \text{id} * \text{id}$



Step	Stack →	Input	Action
1	\$	$\text{id} + \text{id} * \text{id}\$$	shift

Why do Handles Form the Basis of Bottom Up Parsing?

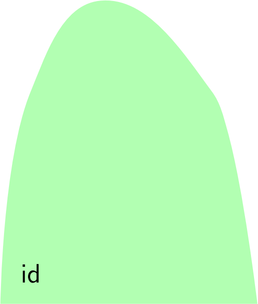
Grammar

1. $E \rightarrow E + E$
2. $E \rightarrow E * E$
3. $E \rightarrow \text{id}$

Input

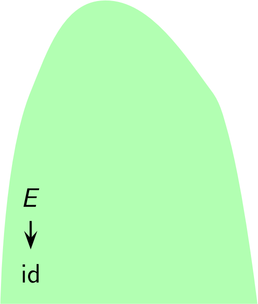
$\text{id} + \text{id} * \text{id}$

Step	Stack →	Input	Action
1	\$	$\text{id} + \text{id} * \text{id}\$$	shift
2	\$ id	$+ \text{id} * \text{id}\$$	reduce by 3



Why do Handles Form the Basis of Bottom Up Parsing?

Grammar	Input
1. $E \rightarrow E+E$ 2. $E \rightarrow E*E$ 3. $E \rightarrow \text{id}$	$\text{id} + \text{id} * \text{id}$



Step	Stack →	Input	Action
1	\$	$\text{id} + \text{id} * \text{id}\$$	shift
2	\$ id	$+ \text{id} * \text{id}\$$	reduce by 3
3	$\$E$	$+ \text{id} * \text{id}\$$	shift

Why do Handles Form the Basis of Bottom Up Parsing?

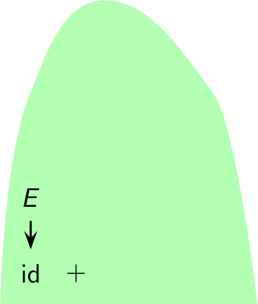
Grammar

1. $E \rightarrow E + E$
2. $E \rightarrow E * E$
3. $E \rightarrow \text{id}$

Input

$\text{id} + \text{id} * \text{id}$

Step	Stack →	Input	Action
1	\$	id + id ∗ id\$	shift
2	\$ id	+ id ∗ id\$	reduce by 3
3	\$$E$	+ id ∗ id\$	shift
4	\$$E$ +	id ∗ id\$	shift



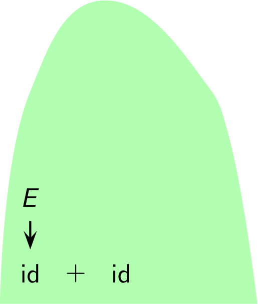
Why do Handles Form the Basis of Bottom Up Parsing?

Grammar

1. $E \rightarrow E + E$
2. $E \rightarrow E * E$
3. $E \rightarrow \text{id}$

Input

$\text{id} + \text{id} * \text{id}$



Step	Stack →	Input	Action
1	\$	$\text{id} + \text{id} * \text{id}\$$	shift
2	\$ id	$+ \text{id} * \text{id}\$$	reduce by 3
3	$\$E$	$+ \text{id} * \text{id}\$$	shift
4	$\$E +$	$\text{id} * \text{id}\$$	shift
5	$\$E +$ id	$* \text{id}\$$	reduce by 3

Why do Handles Form the Basis of Bottom Up Parsing?

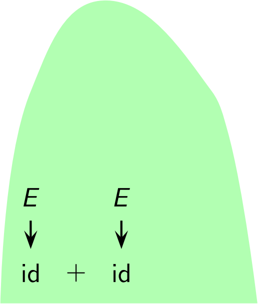
Grammar

1. $E \rightarrow E + E$
2. $E \rightarrow E * E$
3. $E \rightarrow \text{id}$

Input

$\text{id} + \text{id} * \text{id}$

Step	Stack →	Input	Action
1	$\$$	$\text{id} + \text{id} * \text{id}\$$	shift
2	$\$\ \text{id}$	$+ \text{id} * \text{id}\$$	reduce by 3
3	$\$E$	$+ \text{id} * \text{id}\$$	shift
4	$\$E +$	$\text{id} * \text{id}\$$	shift
5	$\$E + \text{id}$	$* \text{id}\$$	reduce by 3
6	$\$E + E$	$* \text{id}\$$	shift



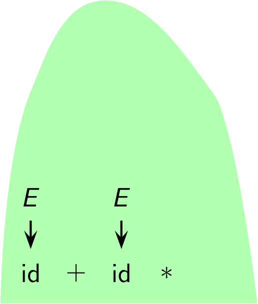
Why do Handles Form the Basis of Bottom Up Parsing?

Grammar

1. $E \rightarrow E + E$
2. $E \rightarrow E * E$
3. $E \rightarrow \text{id}$

Input

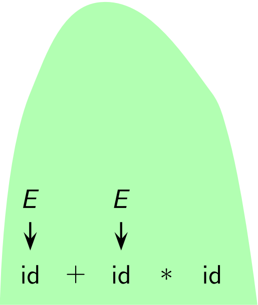
$\text{id} + \text{id} * \text{id}$



Step	Stack →	Input	Action
1	$\$$	$\text{id} + \text{id} * \text{id}\$$	shift
2	$\$\ \text{id}$	$+ \text{id} * \text{id}\$$	reduce by 3
3	$\$E$	$+ \text{id} * \text{id}\$$	shift
4	$\$E +$	$\text{id} * \text{id}\$$	shift
5	$\$E + \text{id}$	$* \text{id}\$$	reduce by 3
6	$\$E + E$	$* \text{id}\$$	shift
7	$\$E + E *$	$\text{id}\$$	shift

Why do Handles Form the Basis of Bottom Up Parsing?

Grammar	Input
1. $E \rightarrow E + E$ 2. $E \rightarrow E * E$ 3. $E \rightarrow \text{id}$	$\text{id} + \text{id} * \text{id}$



Step	Stack →	Input	Action
1	$\$$	$\text{id} + \text{id} * \text{id}\$$	shift
2	$\$\ \text{id}$	$+ \text{id} * \text{id}\$$	reduce by 3
3	$\$E$	$+ \text{id} * \text{id}\$$	shift
4	$\$E +$	$\text{id} * \text{id}\$$	shift
5	$\$E + \text{id}$	$* \text{id}\$$	reduce by 3
6	$\$E + E$	$* \text{id}\$$	shift
7	$\$E + E *$	$\text{id}\$$	shift
8	$\$E + E * \text{id}$	$\$$	reduce by 3

IIT Bombay
cs302: Implementation of Programming Languages

Topic:

Syntax Analysis

Section:

Grammars, Derivations, and Parse Trees

Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in Parsing

CLR(1) Parsing

LALR(1) Parsing

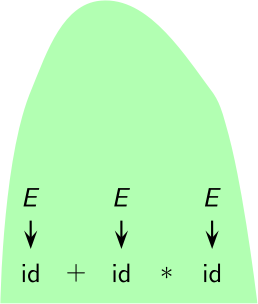
Why do Handles Form the Basis of Bottom Up Parsing?

Grammar

1. $E \rightarrow E + E$
2. $E \rightarrow E * E$
3. $E \rightarrow \text{id}$

Input

$\text{id} + \text{id} * \text{id}$



Step	Stack →	Input	Action
1	$	$\text{id} + \text{id} * \text{id}\$$	shift
2	$\$\ \text{id}$	$+ \text{id} * \text{id}\$$	reduce by 3
3	$\$E$	$+ \text{id} * \text{id}\$$	shift
4	$\$E +$	$\text{id} * \text{id}\$$	shift
5	$\$E + \text{id}$	$* \text{id}\$$	reduce by 3
6	$\$E + E$	$* \text{id}\$$	shift
7	$\$E + E *$	$\text{id}\$$	shift
8	$\$E + E * \text{id}$	$	reduce by 3
9	$\$E + E * E$	$	reduce by 2

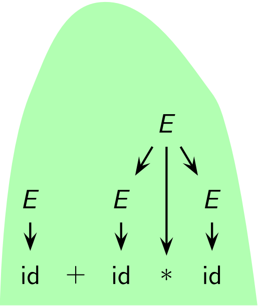
Why do Handles Form the Basis of Bottom Up Parsing?

Grammar

1. $E \rightarrow E+E$
2. $E \rightarrow E * E$
3. $E \rightarrow \text{id}$

Input

$\text{id} + \text{id} * \text{id}$

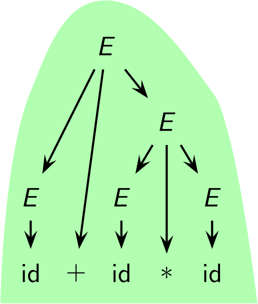


Step	Stack →	Input	Action
1	\$	$\text{id} + \text{id} * \text{id}\$$	shift
2	\$ id	$+ \text{id} * \text{id}\$$	reduce by 3
3	$\$E$	$+ \text{id} * \text{id}\$$	shift
4	$\$E +$	$\text{id} * \text{id}\$$	shift
5	$\$E + \text{id}$	$* \text{id}\$$	reduce by 3
6	$\$E + E$	$* \text{id}\$$	shift
7	$\$E + E *$	$\text{id}\$$	shift
8	$\$E + E * \text{id}$	\$	reduce by 3
9	$\$E + E * E$	\$	reduce by 2
10	$\$E + E$	\$	reduce by 1

Why do Handles Form the Basis of Bottom Up Parsing?

Grammar	Input
1. $E \rightarrow E + E$ 2. $E \rightarrow E * E$ 3. $E \rightarrow \text{id}$	$\text{id} + \text{id} * \text{id}$

alpha, beta, w all CAN BE EMPTY too in righ sentential form



Step	Stack $\rightarrow$	Input	Action
1	$\$$	$\text{id} + \text{id} * \text{id}\$$	shift
2	$\$ \, \text{id}$	$+ \text{id} * \text{id}\$$	reduce by 3
3	$\$E$	$+ \text{id} * \text{id}\$$	shift
4	$\$E +$	$\text{id} * \text{id}\$$	shift
5	$\$E + \text{id}$	$* \text{id}\$$	reduce by 3
6	$\$E + E$	$* \text{id}\$$	shift
7	$\$E + E *$	$\text{id}\$$	shift
8	$\$E + E * \text{id}$	$\$$	reduce by 3
9	$\$E + E * E$	$\$$	reduce by 2
10	$\$ E + E$	$\$$	reduce by 1
11	$\$E$	$\$$	accept

Shift Reduce Parsing

parsing-slides-sanyal-part2.pdf

SLR: Simple LR (Left-to-Right, Rightmost Derivation)
SLR(1): Simple LR with 1-token lookahead

SLR(1) Parsing

bottom-up shift reduce parser, resolving shift-reduce and reduce-reduce conflicts, etc.

SLR(1) Parsing Example

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

SLR(1) Parsing Example

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

Parsing Table

	id	$+$	$*$	\$	E
0	$s2$				$c1$
1		$s3$	$s4$	acc	
2		$r3$	$r3$	$r3$	
3	$s2$				$c5$
4	$s2$				$c6$
5		$\cancel{s3}/r1$	$s4/\cancel{r1}$	$r1$	
6		$\cancel{s3}/r2$	$\cancel{s4}/r2$	$r2$	

SLR(1) Parsing Example

1 $E \to E + E$
2 $E \to E * E$
3 $E \to \text{id}$

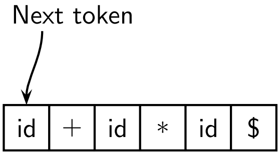
Shift reduce conflicts resolved using precedence and associativity

Parsing Table

	id	+	*	$	E
0	s2				c1
1		s3	s4	acc	
2		r3	r3	r3	
3	s2				c5
4	s2				c6
5		~~s3~~/r1	s4/~~r1~~	r1	
6		~~s3~~/r2	~~s4~~/r2	r2	

SLR(1) Parsing Example

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$



Next token

id	+	id	*	id	$

Parsing Table

	id	+	*	$	E
0	🟩 *s*2				*c*1
1		*s*3	*s*4	*acc*	
2		*r*3	*r*3	*r*3	
3	*s*2				*c*5
4	*s*2				*c*6
5		~~*s*3~~/*r*1	*s*4/~~*r*1~~	*r*1	
6		~~*s*3~~/*r*2	~~*s*4~~/*r*2	*r*2	

Next Action: Shift 2

0

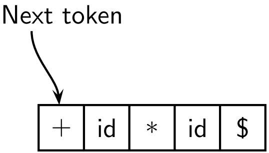
Parsing Stack

SLR(1) Parsing Example

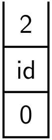
1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

Parsing Table

	id	+	*	$	E
0	s2				c1
1		s3	s4	acc	
2		r3	r3	r3	
3	s2				c5
4	s2				c6
5		~~s3~~/r1	s4/~~r1~~	r1	
6		~~s3~~/r2	~~s4~~/r2	r2	



Next Action: Reduce by Rule 3



Parsing Stack

SLR(1) Parsing Example

IIT Bombay
cs302: Implementation of Programming Languages

Topic:

Syntax Analysis

Section:

Grammars, Derivations, and Parse Trees

Shift Reduce Parsing

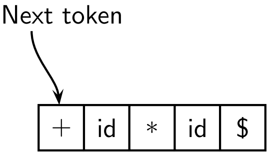
SLR(1) Parsing

Conceptual Issues in Parsing

CLR(1) Parsing

LALR(1) Parsing

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

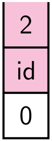

Next token

+	id	*	id	$

Parsing Table

	id	+	*	$	E
0	s2				c1
1		s3	s4	acc	
2		r3	r3	r3	
3	s2				c5
4	s2				c6
5		~~s3~~/r1	s4/~~r1~~	r1	
6		~~s3~~/r2	~~s4~~/r2	r2	

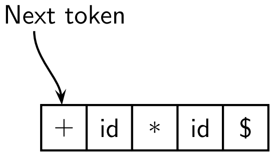
Next Action: Reduce by Rule 3


2
id
0

Parsing Stack

SLR(1) Parsing Example

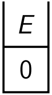
1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$



Parsing Table

	id	+	*	$	E
0	s2				c1
1		s3	s4	acc	
2		r3	r3	r3	
3	s2				c5
4	s2				c6
5		$\cancel{s3}/r1$	$s4/\cancel{r1}$	r1	
6		$\cancel{s3}/r2$	$\cancel{s4}/r2$	r2	

Next Action: Cover by 1



Parsing Stack

IIT Bombay
cs302: Implementation of Programming Languages

Topic:

Syntax Analysis

Section:

Grammars, Derivations, and Parse Trees

Shift Reduce Parsing

SLR(1) Parsing

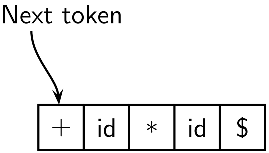
Conceptual Issues in Parsing

CLR(1) Parsing

LALR(1) Parsing

SLR(1) Parsing Example

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

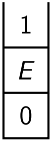


+	id	*	id	$

Parsing Table

	id	+	*	$	E
0	s2				c1
1		s3	s4	acc	
2		r3	r3	r3	
3	s2				c5
4	s2				c6
5		~~s3~~/r1	s4/~~r1~~	r1	
6		~~s3~~/r2	~~s4~~/r2	r2	

Next Action: Shift 3



Parsing Stack

SLR(1) Parsing Example

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

Next token

id	$*$	id	\$

Parsing Table

	id	$+$	$*$	\$	E
0	$s2$				$c1$
1		$s3$	$s4$	acc	
2		$r3$	$r3$	$r3$	
3	$s2$				$c5$
4	$s2$				$c6$
5		$\cancel{s3}/r1$	$s4/\cancel{r1}$	$r1$	
6		$\cancel{s3}/r2$	$\cancel{s4}/r2$	$r2$	

3
$+$
1
E
0

Parsing Stack

Next Action: Shift 2

IIT Bombay
cs302: Implementation of Programming Languages

Topic:

Syntax Analysis

Section:

Grammars, Derivations, and Parse Trees

Shift Reduce Parsing

SLR(1) Parsing

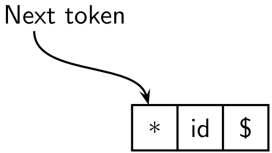
Conceptual Issues in Parsing

CLR(1) Parsing

LALR(1) Parsing

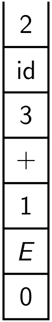
SLR(1) Parsing Example

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$



Parsing Table

	id	+	*	$	E
0	s2				c1
1		s3	s4	acc	
2		r3	r3	r3	
3	s2				c5
4	s2				c6
5		~~s3~~/r1	s4/~~r1~~	r1	
6		~~s3~~/r2	~~s4~~/r2	r2	



Parsing Stack

Next Action: Reduce by Rule 3

SLR(1) Parsing Example

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

Next token

*	id	$

Parsing Table

	id	+	*	$	E
0	$s2$				$c1$
1		$s3$	$s4$	acc	
2		$r3$	$r3$	$r3$	
3	$s2$				$c5$
4	$s2$				$c6$
5		$\cancel{s3}/r1$	$s4/\cancel{r1}$	$r1$	
6		$\cancel{s3}/r2$	$\cancel{s4}/r2$	$r2$	

2
id
3
+
1
E
0

Parsing Stack

Next Action: Reduce by Rule 3

SLR(1) Parsing Example

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

Next token

*	id	$

Parsing Table

	id	+	*	$	E
0	s2				c1
1		s3	s4	acc	
2		r3	r3	r3	
3	s2				c5
4	s2				c6
5		~~s3~~/r1	s4/~~r1~~	r1	
6		~~s3~~/r2	~~s4~~/r2	r2	

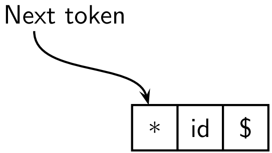
E
3
+
1
E
0

Parsing Stack

Next Action: Cover by 5

SLR(1) Parsing Example

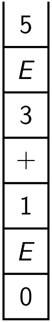
1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$



*	id	$

Parsing Table

	id	+	*	$	E
0	s2				c1
1		s3	s4	acc	
2		r3	r3	r3	
3	s2				c5
4	s2				c6
5		~~s3~~/r1	s4/~~r1~~	r1	
6		~~s3~~/r2	~~s4~~/r2	r2	



5
E
3
+
1
E
0

Parsing Stack

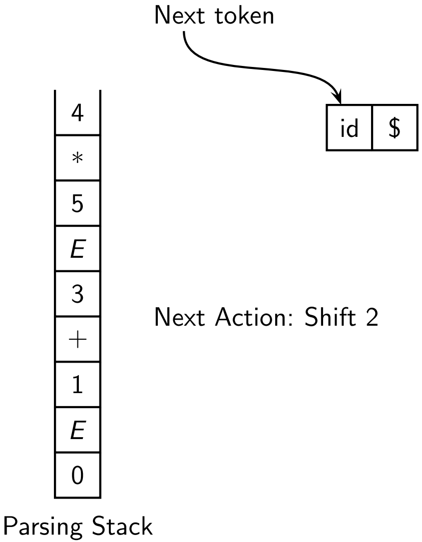
Next Action: Shift 4

SLR(1) Parsing Example

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

Parsing Table

	id	$+$	$*$	\$	E
0	$s2$				$c1$
1		$s3$	$s4$	acc	
2		$r3$	$r3$	$r3$	
3	$s2$				$c5$
4	$s2$				$c6$
5		~~$s3$~~/$r1$	$s4$/~~$r1$~~	$r1$	
6		~~$s3$~~/$r2$	~~$s4$~~/$r2$	$r2$	



Next token

id | \$

4
$*$
5
E
3
$+$
1
E
0

Parsing Stack

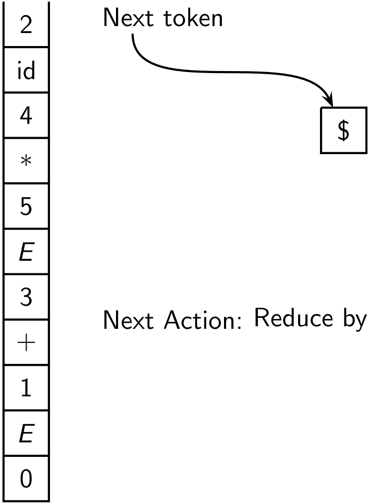
Next Action: Shift 2

SLR(1) Parsing Example

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

Parsing Table

	id	+	*	$	E
0	s2				c1
1		s3	s4	acc	
2		r3	r3	r3	
3	s2				c5
4	s2				c6
5		~~s3~~/r1	s4/~~r1~~	r1	
6		~~s3~~/r2	~~s4~~/r2	r2	



Parsing Stack

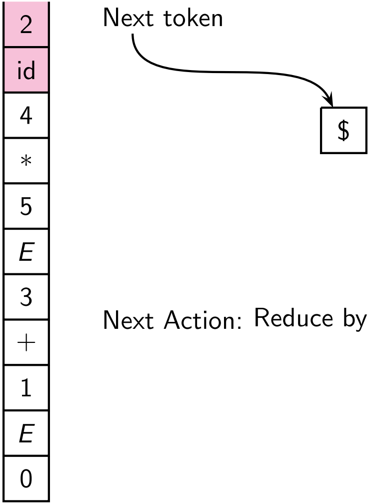
Next Action: Reduce by Rule 3

SLR(1) Parsing Example

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

Parsing Table

	id	+	*	$	E
0	s2				c1
1		s3	s4	acc	
2		r3	r3	r3	
3	s2				c5
4	s2				c6
5		~~s3~~/r1	s4/~~r1~~	r1	
6		~~s3~~/r2	~~s4~~/r2	r2	


2
id
4
*
5
E
3
+
1
E
0

Parsing Stack

Next token

$

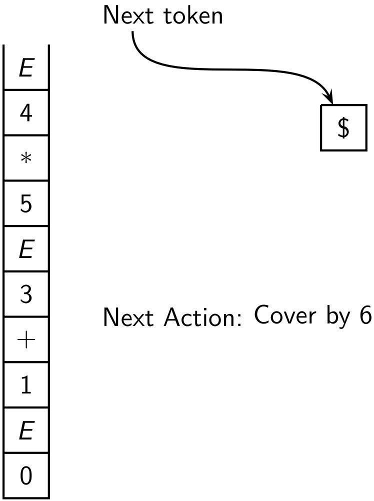
Next Action: Reduce by Rule 3

SLR(1) Parsing Example

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

Parsing Table

	id	+	*	$	E
0	s2				c1
1		s3	s4	acc	
2		r3	r3	r3	
3	s2				c5
4	s2				c6
5		~~s3~~/r1	s4/~~r1~~	r1	
6		~~s3~~/r2	~~s4~~/r2	r2	



Next token

E
4
*
5
E
3
+
1
E
0

$

Next Action: Cover by 6

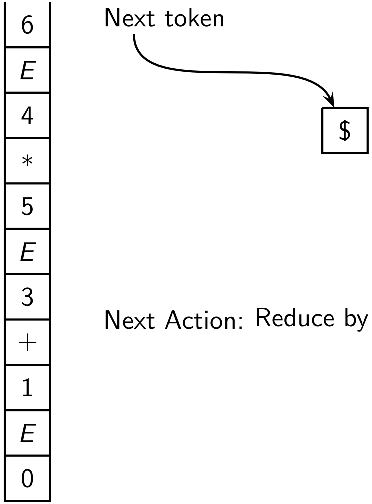
Parsing Stack

SLR(1) Parsing Example

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

Parsing Table

	id	+	*	$	E
0	s2				c1
1		s3	s4	acc	
2		r3	r3	r3	
3	s2				c5
4	s2				c6
5		~~s3~~/r1	s4/~~r1~~	r1	
6		~~s3~~/r2	~~s4~~/r2	r2	



Parsing Stack

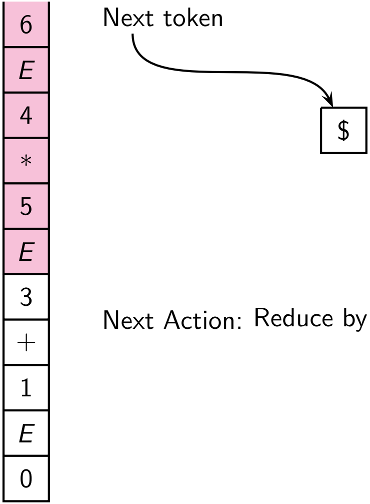
Next Action: Reduce by 2

SLR(1) Parsing Example

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

Parsing Table

	id	+	*	\$	E
0	s2				c1
1		s3	s4	acc	
2		r3	r3	r3	
3	s2				c5
4	s2				c6
5		~~s3~~/r1	s4/~~r1~~	r1	
6		~~s3~~/r2	~~s4~~/r2	r2	


6
E
4
*
5
E
3
+
1
E
0

Next token

\$

Parsing Stack

Next Action: Reduce by 2

SLR(1) Parsing Example

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

Next token

$

Parsing Table

	id	$+$	$*$	\$	E
0	$s2$				$c1$
1		$s3$	$s4$	acc	
2		$r3$	$r3$	$r3$	
3	$s2$				$c5$
4	$s2$				$c6$
5		~~$s3$~~$/r1$	$s4/$~~$r1$~~	$r1$	
6		~~$s3$~~$/r2$	~~$s4$~~$/r2$	$r2$	

E
3
$+$
1
E
0

Parsing Stack

Next Action: Cover by 5

SLR(1) Parsing Example

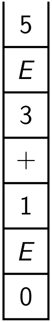
1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

Next token

$

Parsing Table

	id	+	*	$	E
0	s2				c1
1		s3	s4	acc	
2		r3	r3	r3	
3	s2				c5
4	s2				c6
5		~~s3~~/r1	s4/~~r1~~	r1	
6		~~s3~~/r2	~~s4~~/r2	r2	



Parsing Stack

Next Action: Reduce by Rule 1

SLR(1) Parsing Example

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

Next token

$

Parsing Table

	id	+	*	$	E
0	s2				c1
1		s3	s4	acc	
2		r3	r3	r3	
3	s2				c5
4	s2				c6
5		~~s3~~/r1	s4/~~r1~~	r1	
6		~~s3~~/r2	~~s4~~/r2	r2	

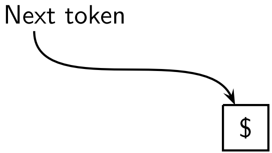
5
E
3
+
1
E
0

Parsing Stack

Next Action: Reduce by Rule 1

SLR(1) Parsing Example

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$



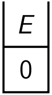
Next token

$

Parsing Table

	id	+	*	$	E
0	s2				🟩 c1
1		s3	s4	acc	
2		r3	r3	r3	
3	s2				c5
4	s2				c6
5		~~s3~~/r1	s4/~~r1~~	r1	
6		~~s3~~/r2	~~s4~~/r2	r2	

Next Action: Cover by 1



E

0

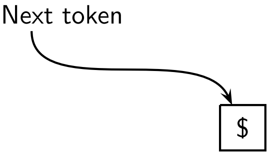
Parsing Stack

SLR(1) Parsing Example

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

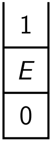
Parsing Table

	id	$+$	$*$	\$	E
0	$s2$				$c1$
1		$s3$	$s4$	acc	
2		$r3$	$r3$	$r3$	
3	$s2$				$c5$
4	$s2$				$c6$
5		$\cancel{s3}/r1$	$s4/\cancel{r1}$	$r1$	
6		$\cancel{s3}/r2$	$\cancel{s4}/r2$	$r2$	


Next token

\$

Next Action: Accept



Parsing Stack

SLR(1) Parsing Example

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

SLR(1) Parsing Example

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

Parsing Table

	id	+	*	$	E
0	s2				c1
1		s3	s4	acc	
2		r3	r3	r3	
3	s2				c5
4	s2				c6
5		~~s3~~/r1	s4/~~r1~~	r1	
6		~~s3~~/r2	~~s4~~/r2	r2	

SLR(1) Parsing Example

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

Shift reduce conflicts resolved using precedence and associativity

Parsing Table

	id	+	*	\$	E
0	s2				c1
1		s3	s4	acc	
2		r3	r3	r3	
3	s2				c5
4	s2				c6
5		~~s3~~/r1	s4/~~r1~~	r1	
6		~~s3~~/r2	~~s4~~/r2	r2	

SLR(1) Parsing Example

Combining the reduce and the following cover operation into a single step for convenience

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

Parsing Table

	id	$+$	$*$	\$	E
0	$s2$				$c1$
1		$s3$	$s4$	acc	
2		$r3$	$r3$	$r3$	
3	$s2$				$c5$
4	$s2$				$c6$
5		$\cancel{s3}/r1$	$s4/\cancel{r1}$	$r1$	
6		$\cancel{s3}/r2$	$\cancel{s4}/r2$	$r2$	

Step	Stack →	Input	Action
1	\$0	id + id ∗ id\$	$s2$
2	\$0 id 2	+ id ∗ id\$	$r3$ and $c1$
3	\$0 E 1	+ id ∗ id\$	$s3$
4	\$0 E 1 + 3	id ∗ id\$	$s2$
5	\$0 E 1 + 3 id 2	∗ id\$	$r3$ and $c5$
6	\$0 E 1 + 3 E 5	∗ id\$	$s4$
7	\$0 E 1 + 3 E 5 ∗ 4	id\$	$s2$
8	\$0 E 1 + 3 E 5 ∗ 4 id 2	\$	$r3$ and $c6$
9	\$0 E 1 + 3 E 5 ∗ 4 E6	\$	$r2$ and $c5$
10	\$0 E 1 + 3 E 5	\$	$r1$ and $c1$
11	\$0 E 1	\$	*accept*

handle.State x --reduced from--> LHS rule and cover by state

SLR(1) Parsing Example

1 $E \to E + E$
2 $E \to E * E$
3 $E \to \text{id}$

Parsing Table

	id	$+$	$*$	\$	E
0	$s2$				$c1$
1		$s3$	$s4$	acc	
2		$r3$	$r3$	$r3$	
3	$s2$				$c5$
4	$s2$				$c6$
5		$\cancel{s3}/r1$	$s4/\cancel{r1}$	$r1$	
6		$\cancel{s3}/r2$	$\cancel{s4}/r2$	$r2$	

Step	Stack $\to$	Input	Action
1	$\$0$	$\text{id} + \text{id} * \text{id}\$$	$s2$
2	$\$0\ \text{id}\ 2$	$+ \text{id} * \text{id}\$$	$r3$ and $c1$
3	$\$0\ E\ 1$	$+ \text{id} * \text{id}\$$	$s3$
4	$\$0\ E\ 1 + 3$	$\text{id} * \text{id}\$$	$s2$
5	$\$0\ E\ 1 + 3\ \text{id}\ 2$	$* \text{id}\$$	$r3$ and $c5$
6	$\$0\ E\ 1 + 3\ E\ 5$	$* \text{id}\$$	$s4$
7	$\$0\ E\ 1 + 3\ E\ 5 * 4$	$\text{id}\$$	$s2$
8	$\$0\ E\ 1 + 3\ E\ 5 * 4\ \text{id}\ 2$	\$	$r3$ and $c6$
9	$\$0\ E\ 1 + 3\ E\ 5 * 4\ E6$	\$	$r2$ and $c5$
10	$\$0\ E\ 1 + 3\ E\ 5$	\$	$r1$ and $c1$
11	$\$0\ E\ 1$	\$	*accept*

Shift Reduce Parsing: From Intuitions to Formal Algorithms

We undertake this journey in six steps using the ambiguous grammar of expressions. It illustrates how yacc allows disambiguating a grammar without rewriting it

$$\begin{array}{lcl} E & \rightarrow & E + E \\ E & \rightarrow & E * E \\ E & \rightarrow & \text{id} \end{array}$$

1. We assume that both $+$ and $*$ are left associative and $*$ takes precedence over $+$
   We see the influence of these choices on derivations by considering four inputs
   $\text{id} + \text{id} + \text{id}$, $\text{id} * \text{id} * \text{id}$, $\text{id} + \text{id} * \text{id}$, and $\text{id} * \text{id} + \text{id}$.
2. We see the meaning of a shift reduce parser tracing the rightmost derivation in reverse
   We see the meaning of handle pruning in tracing the rightmost derivation
3. We define the notions of viable prefixes for discovering handles
4. We define valid items to recognize viable prefixes
5. We define FOLLOW sets to define a criterion of handle pruning
6. We see the algorithm that constructs valid items

Shift Reduce Parsing: From Intuitions to Formal Algorithms

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

Shift Reduce Parsing: From Intuitions to Formal Algorithms

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

Step	Stack →	Input	Action
1	\$	$\text{id} + \text{id} * \text{id}\$$	shift
2	\$id	$+ \text{id} * \text{id}\$$	reduce by 3
3	$\$E$	$+ \text{id} * \text{id}\$$	shift
4	$\$E +$	$\text{id} * \text{id}\$$	shift
5	$\$E + \text{id}$	$* \text{id}\$$	reduce by 3
6	$\$E + E$	$* \text{id}\$$	shift
7	$\$E + E *$	$\text{id}\$$	shift
8	$\$E + E * \text{id}$	\$	reduce by 3
9	$\$E + E * E$	\$	reduce by 2
10	$\$E + E$	\$	reduce by 1
11	$\$E$	\$	accept

Shift Reduce Parsing: From Intuitions to Formal Algorithms

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

Shift reduce conflicts resolved using precedence and associativity

	id	+	$*$	\$	E
0	$s2$				$c1$
1		$s3$	$s4$	acc	
2		$r3$	$r3$	$r3$	
3	$s2$				$c5$
4	$s2$				$c6$
5		$\cancel{s3}/r1$	$s4/\cancel{r1}$	$r1$	
6		$\cancel{s3}/r2$	$\cancel{s4}/r2$	$r2$	

Step	Stack →	Input	Action
1	\$	id + id $*$ id\$	shift
2	\$id	+ id $*$ id\$	reduce by 3
3	\$$E$	+ id $*$ id\$	shift
4	\$$E$ +	id $*$ id\$	shift
5	\$$E$ + id	$*$ id\$	reduce by 3
6	\$$E + E$	$*$ id\$	shift
7	\$$E + E *$	id\$	shift
8	\$$E + E *$ id	\$	reduce by 3
9	\$$E + E * E$	\$	reduce by 2
10	\$$E + E$	\$	reduce by 1
11	\$$E$	\$	accept

Step	Stack →	Input	Action
1	\$0	id + id $*$ id\$	$s2$
2	\$0 id 2	+ id $*$ id\$	$r3$ and $c1$
3	\$0 E 1	+ id $*$ id\$	$s3$
4	\$0 E 1 + 3	id $*$ id\$	$s2$
5	\$0 E 1 + 3 id 2	$*$ id\$	$r3$ and $c5$
6	\$0 E 1 + 3 E 5	$*$ id\$	$s4$
7	\$0 E 1 + 3 E 5 $*$ 4	id\$	$s2$
8	\$0 E 1 + 3 E 5 $*$ 4 id 2	\$	$r3$ and $c6$
9	\$0 E 1 + 3 E 5 $*$ 4 E6	\$	$r2$ and $c5$
10	\$0 E 1 + 3 E 5	\$	$r1$ and $c1$
11	\$0 E 1	\$	$accept$

Shift Reduce Parsing: From Intuitions to Formal Algorithms

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow$ id

Combining the
reduce and the following cover
operation into a single step
for convenience

	id	$+$	$*$	\$	E
	$s2$				$c1$
		$s3$	$s4$	acc	
		$r3$	$r3$	$r3$	
					$c5$
	2				$c6$
5		$\cancel{s3}/r1$	$s4/\cancel{r1}$	$r1$	
6		$\cancel{s3}/r2$	$\cancel{s4}/r2$	$r2$	

Step	Stack →	Input	Action
1	\$	id $+$ id $*$ id\$	shift
2	\$id	$+$ id $*$ id\$	reduce by 3
3	\$$E$	$+$ id $*$ id\$	shift
4	\$$E +$	id $*$ id\$	shift
5	\$$E +$ id	$*$ id\$	reduce by 3
6	\$$E + E$	$*$ id\$	shift
7	\$$E + E *$	id\$	shift
8	\$$E + E *$ id	\$	reduce by 3
9	\$$E + E * E$	\$	reduce by 2
10	\$$E + E$	\$	reduce by 1
11	\$$E$	\$	accept

Step	Stack →	Input	Action
1	\$0	id $+$ id $*$ id\$	$s2$
2	\$0 id 2	$+$ id $*$ id\$	$r3$ and $c1$
3	\$0 E 1	$+$ id $*$ id\$	$s3$
4	\$0 E 1 $+$ 3	id $*$ id\$	$s2$
5	\$0 E 1 $+$ 3 id 2	$*$ id\$	$r3$ and $c5$
6	\$0 E 1 $+$ 3 E 5	$*$ id\$	$s4$
7	\$0 E 1 $+$ 3 E 5 $*$ 4	id\$	$s2$
8	\$0 E 1 $+$ 3 E 5 $*$ 4 id 2	\$	$r3$ and $c6$
9	\$0 E 1 $+$ 3 E 5 $*$ 4 E 6	\$	$r2$ and $c5$
10	\$0 E 1 $+$ 3 E 5	\$	$r1$ and $c1$
11	\$0 E 1	\$	$accept$

Shift Reduce Parsing: From Intuitions to Formal Algorithms

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

How do we make this journey?

	id	+	∗	$	E
0	$s2$				$c1$
1		$s3$	$s4$	acc	
2		$r3$	$r3$	$r3$	
3	$s2$				$c5$
4	$s2$				$c6$
5		$\cancel{s3}/r1$	$s4/\cancel{r1}$	$r1$	
6		$\cancel{s3}/r2$	$\cancel{s4}/r2$	$r2$	

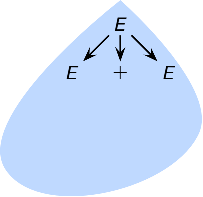
Step	Stack →	Input	Action
1	$	id + id ∗ id$	shift
2	$id	+ id ∗ id$	reduce by 3
3	$$E$	+ id ∗ id$	shift
4	$$E +$	id ∗ id$	shift
5	$$E +$ id	∗ id$	reduce by 3
6	$$E + E$	∗ id$	shift
7	$$E + E *$	id$	shift
8	$$E + E *$ id	$	reduce by 3
9	$$E + E * E$	$	reduce by 2
10	$$E + E$	$	reduce by 1
11	$$E$	$	accept

Step	Stack →	Input	Action
1	$0	id + id ∗ id$	$s2$
2	$0 id 2	+ id ∗ id$	$r3$ and $c1$
3	$0 E 1	+ id ∗ id$	$s3$
4	$0 E 1 + 3	id ∗ id$	$s2$
5	$0 E 1 + 3 id 2	∗ id$	$r3$ and $c5$
6	$0 E 1 + 3 E 5	∗ id$	$s4$
7	$0 E 1 + 3 E 5 ∗ 4	id$	$s2$
8	$0 E 1 + 3 E 5 ∗ 4 id 2	$	$r3$ and $c6$
9	$0 E 1 + 3 E 5 ∗ 4 E6	$	$r2$ and $c5$
10	$0 E 1 + 3 E 5	$	$r1$ and $c1$
11	$0 E 1	$	$accept$

Step 1: Precedence and Associativity Rule Out Undesirable Derivations

Input id + id + id

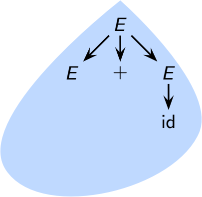
$E \overset{rm}{\Rightarrow} E + E$



Step 1: Precedence and Associativity Rule Out Undesirable Derivations

Input id + id + id

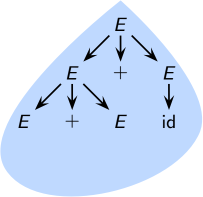
$$E \overset{rm}{\Rightarrow} E + E$$
$$\overset{rm}{\Rightarrow} E + \text{id}$$



Step 1: Precedence and Associativity Rule Out Undesirable Derivations

Input id + id + id

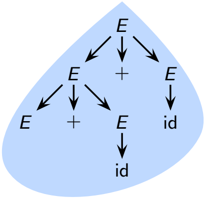
$$
\begin{aligned}
E &\overset{rm}{\Rightarrow} E + E \\
&\overset{rm}{\Rightarrow} E + \text{id} \\
&\overset{rm}{\Rightarrow} E + E + \text{id}
\end{aligned}
$$



Step 1: Precedence and Associativity Rule Out Undesirable Derivations

Input id + id + id

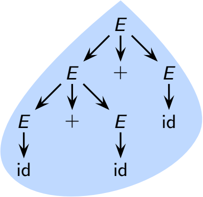
$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + \text{id} \\ &\overset{rm}{\Rightarrow} E + E + \text{id} \\ &\overset{rm}{\Rightarrow} E + \text{id} + \text{id} \end{aligned}$$



Step 1: Precedence and Associativity Rule Out Undesirable Derivations

Input id + id + id

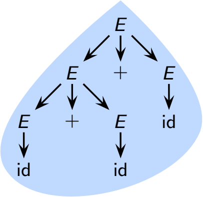
$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + \text{id} \\ &\overset{rm}{\Rightarrow} E + E + \text{id} \\ &\overset{rm}{\Rightarrow} E + \text{id} + \text{id} \\ &\overset{rm}{\Rightarrow} \text{id} + \text{id} + \text{id} \end{aligned}$$



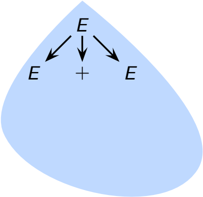
Step 1: Precedence and Associativity Rule Out Undesirable Derivations

Input id + id + id

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + \text{id} \\ &\overset{rm}{\Rightarrow} E + E + \text{id} \\ &\overset{rm}{\Rightarrow} E + \text{id} + \text{id} \\ &\overset{rm}{\Rightarrow} \text{id} + \text{id} + \text{id} \end{aligned}$$



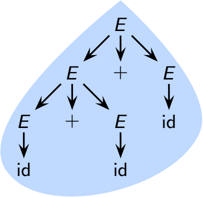
$$E \overset{rm}{\Rightarrow} E + E$$



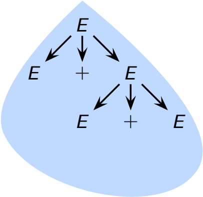
Step 1: Precedence and Associativity Rule Out Undesirable Derivations

Input id + id + id

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + \text{id} \\ &\overset{rm}{\Rightarrow} E + E + \text{id} \\ &\overset{rm}{\Rightarrow} E + \text{id} + \text{id} \\ &\overset{rm}{\Rightarrow} \text{id} + \text{id} + \text{id} \end{aligned}$$



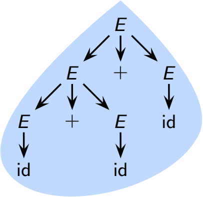
$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + E + E \end{aligned}$$



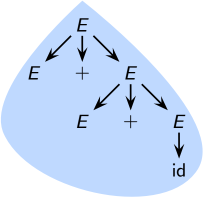
Step 1: Precedence and Associativity Rule Out Undesirable Derivations

Input id + id + id

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + \text{id} \\ &\overset{rm}{\Rightarrow} E + E + \text{id} \\ &\overset{rm}{\Rightarrow} E + \text{id} + \text{id} \\ &\overset{rm}{\Rightarrow} \text{id} + \text{id} + \text{id} \end{aligned}$$



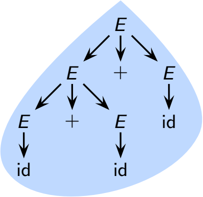
$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + E + E \\ &\overset{rm}{\Rightarrow} E + E + \text{id} \end{aligned}$$



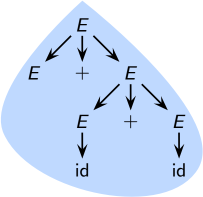
Step 1: Precedence and Associativity Rule Out Undesirable Derivations

Input id + id + id

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + \text{id} \\ &\overset{rm}{\Rightarrow} E + E + \text{id} \\ &\overset{rm}{\Rightarrow} E + \text{id} + \text{id} \\ &\overset{rm}{\Rightarrow} \text{id} + \text{id} + \text{id} \end{aligned}$$



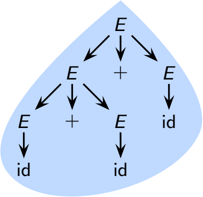
$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + E + E \\ &\overset{rm}{\Rightarrow} E + E + \text{id} \\ &\overset{rm}{\Rightarrow} E + \text{id} + \text{id} \end{aligned}$$



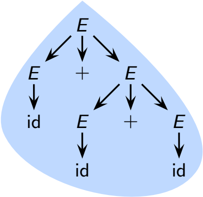
Step 1: Precedence and Associativity Rule Out Undesirable Derivations

Input id + id + id

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + \text{id} \\ &\overset{rm}{\Rightarrow} E + E + \text{id} \\ &\overset{rm}{\Rightarrow} E + \text{id} + \text{id} \\ &\overset{rm}{\Rightarrow} \text{id} + \text{id} + \text{id} \end{aligned}$$



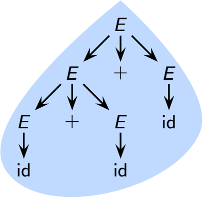
$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + E + E \\ &\overset{rm}{\Rightarrow} E + E + \text{id} \\ &\overset{rm}{\Rightarrow} E + \text{id} + \text{id} \\ &\overset{rm}{\Rightarrow} \text{id} + \text{id} + \text{id} \end{aligned}$$



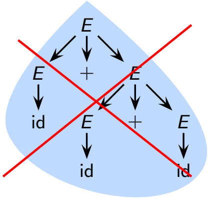
Step 1: Precedence and Associativity Rule Out Undesirable Derivations

Input id + id + id

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + \text{id} \\ &\overset{rm}{\Rightarrow} E + E + \text{id} \\ &\overset{rm}{\Rightarrow} E + \text{id} + \text{id} \\ &\overset{rm}{\Rightarrow} \text{id} + \text{id} + \text{id} \end{aligned}$$



$$\xcancel{\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + E + E \\ &\overset{rm}{\Rightarrow} E + E + \text{id} \\ &\overset{rm}{\Rightarrow} E + \text{id} + \text{id} \\ &\overset{rm}{\Rightarrow} \text{id} + \text{id} + \text{id} \end{aligned}}$$

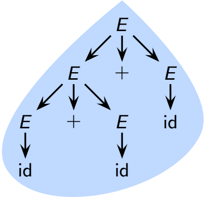


+ is left associative

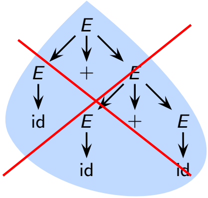
Step 1: Precedence and Associativity Rule Out Undesirable Derivations

Input id + id + id

$$
\begin{aligned}
E &\overset{rm}{\Rightarrow} E+E \\
&\overset{rm}{\Rightarrow} E+\text{id} \\
&\overset{rm}{\Rightarrow} E+E+\text{id} \\
&\overset{rm}{\Rightarrow} E+\text{id}+\text{id} \\
&\overset{rm}{\Rightarrow} \text{id}+\text{id}+\text{id}
\end{aligned}
$$



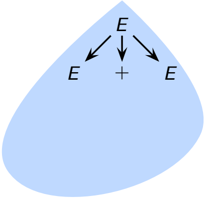
$$
\xcancel{
\begin{aligned}
E &\overset{rm}{\Rightarrow} E+E \\
&\overset{rm}{\Rightarrow} E+E+E \\
&\overset{rm}{\Rightarrow} E+E+\text{id} \\
&\overset{rm}{\Rightarrow} E+\text{id}+\text{id} \\
&\overset{rm}{\Rightarrow} \text{id}+\text{id}+\text{id}
\end{aligned}
}
$$



+ is left associative

Input id $*$ id + id

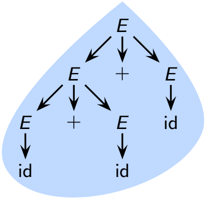
$$E \overset{rm}{\Rightarrow} E+E$$



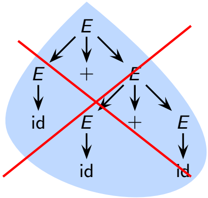
Step 1: Precedence and Associativity Rule Out Undesirable Derivations

$\boxed{\text{Input } \mathrm{id} + \mathrm{id} + \mathrm{id}}$

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + \mathrm{id} \\ &\overset{rm}{\Rightarrow} E + E + \mathrm{id} \\ &\overset{rm}{\Rightarrow} E + \mathrm{id} + \mathrm{id} \\ &\overset{rm}{\Rightarrow} \mathrm{id} + \mathrm{id} + \mathrm{id} \end{aligned}$$



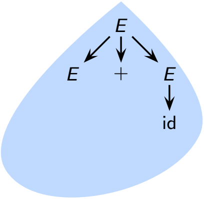
$$\xcancel{\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + E + E \\ &\overset{rm}{\Rightarrow} E + E + \mathrm{id} \\ &\overset{rm}{\Rightarrow} E + \mathrm{id} + \mathrm{id} \\ &\overset{rm}{\Rightarrow} \mathrm{id} + \mathrm{id} + \mathrm{id} \end{aligned}}$$



$\boxed{+ \text{ is left associative}}$

$\boxed{\text{Input } \mathrm{id} * \mathrm{id} + \mathrm{id}}$

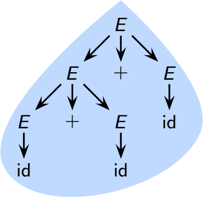
$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + \mathrm{id} \end{aligned}$$



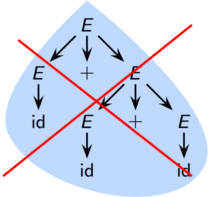
Step 1: Precedence and Associativity Rule Out Undesirable Derivations

Input id + id + id

$$
\begin{aligned}
E &\overset{rm}{\Rightarrow} E + E\\
&\overset{rm}{\Rightarrow} E + \text{id}\\
&\overset{rm}{\Rightarrow} E + E + \text{id}\\
&\overset{rm}{\Rightarrow} E + \text{id} + \text{id}\\
&\overset{rm}{\Rightarrow} \text{id} + \text{id} + \text{id}
\end{aligned}
$$



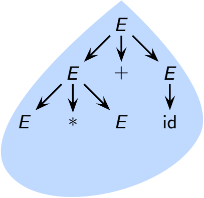
~~$E \overset{rm}{\Rightarrow} E + E$~~
~~$\overset{rm}{\Rightarrow} E + E + E$~~
~~$\overset{rm}{\Rightarrow} E + E + \text{id}$~~
~~$\overset{rm}{\Rightarrow} E + \text{id} + \text{id}$~~
~~$\overset{rm}{\Rightarrow} \text{id} + \text{id} + \text{id}$~~



+ is left associative

Input id $*$ id + id

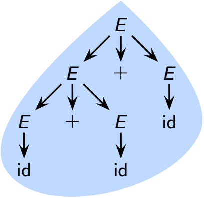
$$
\begin{aligned}
E &\overset{rm}{\Rightarrow} E + E\\
&\overset{rm}{\Rightarrow} E + \text{id}\\
&\overset{rm}{\Rightarrow} E * E + \text{id}
\end{aligned}
$$



Step 1: Precedence and Associativity Rule Out Undesirable Derivations

Input id + id + id

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + \text{id} \\ &\overset{rm}{\Rightarrow} E + E + \text{id} \\ &\overset{rm}{\Rightarrow} E + \text{id} + \text{id} \\ &\overset{rm}{\Rightarrow} \text{id} + \text{id} + \text{id} \end{aligned}$$



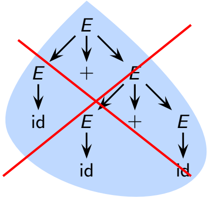
~~$E \overset{rm}{\Rightarrow} E + E$~~

~~$\overset{rm}{\Rightarrow} E + E + E$~~

~~$\overset{rm}{\Rightarrow} E + E + \text{id}$~~

~~$\overset{rm}{\Rightarrow} E + \text{id} + \text{id}$~~

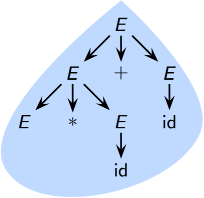
~~$\overset{rm}{\Rightarrow} \text{id} + \text{id} + \text{id}$~~



+ is left associative

Input id ∗ id + id

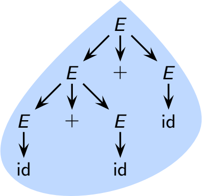
$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + \text{id} \\ &\overset{rm}{\Rightarrow} E * E + \text{id} \\ &\overset{rm}{\Rightarrow} E * \text{id} + \text{id} \end{aligned}$$



Step 1: Precedence and Associativity Rule Out Undesirable Derivations

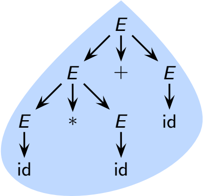
Input id + id + id

$$\begin{array}{lll} E & \overset{rm}{\Rightarrow} & E + E \\ & \overset{rm}{\Rightarrow} & E + \text{id} \\ & \overset{rm}{\Rightarrow} & E + E + \text{id} \\ & \overset{rm}{\Rightarrow} & E + \text{id} + \text{id} \\ & \overset{rm}{\Rightarrow} & \text{id} + \text{id} + \text{id} \end{array}$$

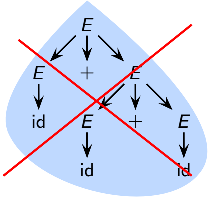


Input id ∗ id + id

$$\begin{array}{lll} E & \overset{rm}{\Rightarrow} & E + E \\ & \overset{rm}{\Rightarrow} & E + \text{id} \\ & \overset{rm}{\Rightarrow} & E * E + \text{id} \\ & \overset{rm}{\Rightarrow} & E * \text{id} + \text{id} \\ & \overset{rm}{\Rightarrow} & \text{id} * \text{id} + \text{id} \end{array}$$



$$\xcancel{\begin{array}{lll} E & \overset{rm}{\Rightarrow} & E + E \\ & \overset{rm}{\Rightarrow} & E + E + E \\ & \overset{rm}{\Rightarrow} & E + E + \text{id} \\ & \overset{rm}{\Rightarrow} & E + \text{id} + \text{id} \\ & \overset{rm}{\Rightarrow} & \text{id} + \text{id} + \text{id} \end{array}}$$

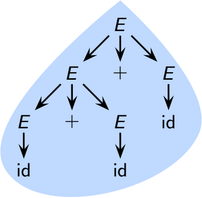


+ is left associative

Step 1: Precedence and Associativity Rule Out Undesirable Derivations

Input id + id + id

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + \text{id} \\ &\overset{rm}{\Rightarrow} E + E + \text{id} \\ &\overset{rm}{\Rightarrow} E + \text{id} + \text{id} \\ &\overset{rm}{\Rightarrow} \text{id} + \text{id} + \text{id} \end{aligned}$$



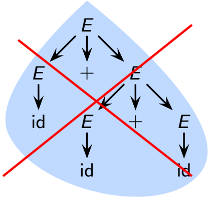
~~$E \overset{rm}{\Rightarrow} E + E$~~

~~$\overset{rm}{\Rightarrow} E + E + E$~~

~~$\overset{rm}{\Rightarrow} E + E + \text{id}$~~

~~$\overset{rm}{\Rightarrow} E + \text{id} + \text{id}$~~

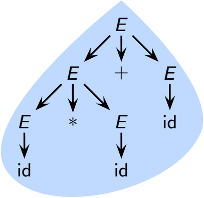
~~$\overset{rm}{\Rightarrow} \text{id} + \text{id} + \text{id}$~~



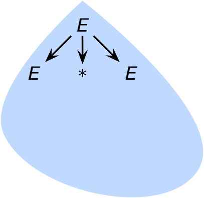
+ is left associative

Input id ∗ id + id

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + \text{id} \\ &\overset{rm}{\Rightarrow} E * E + \text{id} \\ &\overset{rm}{\Rightarrow} E * \text{id} + \text{id} \\ &\overset{rm}{\Rightarrow} \text{id} * \text{id} + \text{id} \end{aligned}$$



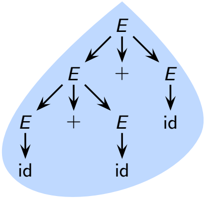
$$E \overset{rm}{\Rightarrow} E * E$$



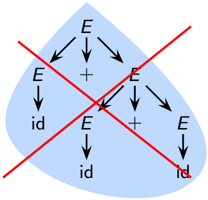
Step 1: Precedence and Associativity Rule Out Undesirable Derivations

Input id + id + id

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E+E \\ &\overset{rm}{\Rightarrow} E+\text{id} \\ &\overset{rm}{\Rightarrow} E+E+\text{id} \\ &\overset{rm}{\Rightarrow} E+\text{id}+\text{id} \\ &\overset{rm}{\Rightarrow} \text{id}+\text{id}+\text{id} \end{aligned}$$



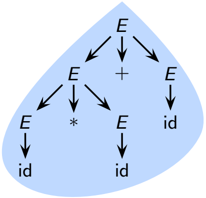
$$\begin{aligned} E &\overset{rm}{\Rightarrow} E+E \\ &\overset{rm}{\Rightarrow} E+E+E \\ &\overset{rm}{\Rightarrow} E+E+\text{id} \\ &\overset{rm}{\Rightarrow} E+\text{id}+\text{id} \\ &\overset{rm}{\Rightarrow} \text{id}+\text{id}+\text{id} \end{aligned}$$



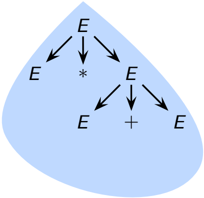
+ is left associative

Input id ∗ id + id

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E+E \\ &\overset{rm}{\Rightarrow} E+\text{id} \\ &\overset{rm}{\Rightarrow} E*E+\text{id} \\ &\overset{rm}{\Rightarrow} E*\text{id}+\text{id} \\ &\overset{rm}{\Rightarrow} \text{id}*\text{id}+\text{id} \end{aligned}$$



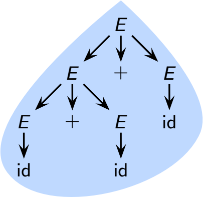
$$\begin{aligned} E &\overset{rm}{\Rightarrow} E*E \\ &\overset{rm}{\Rightarrow} E*E+E \end{aligned}$$



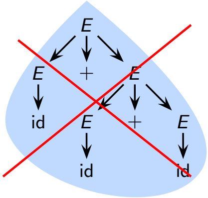
Step 1: Precedence and Associativity Rule Out Undesirable Derivations

Input id + id + id

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + \text{id} \\ &\overset{rm}{\Rightarrow} E + E + \text{id} \\ &\overset{rm}{\Rightarrow} E + \text{id} + \text{id} \\ &\overset{rm}{\Rightarrow} \text{id} + \text{id} + \text{id} \end{aligned}$$



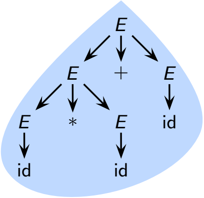
$$\xcancel{\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + E + E \\ &\overset{rm}{\Rightarrow} E + E + \text{id} \\ &\overset{rm}{\Rightarrow} E + \text{id} + \text{id} \\ &\overset{rm}{\Rightarrow} \text{id} + \text{id} + \text{id} \end{aligned}}$$



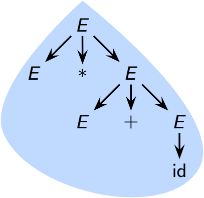
+ is left associative

Input id ∗ id + id

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + \text{id} \\ &\overset{rm}{\Rightarrow} E * E + \text{id} \\ &\overset{rm}{\Rightarrow} E * \text{id} + \text{id} \\ &\overset{rm}{\Rightarrow} \text{id} * \text{id} + \text{id} \end{aligned}$$



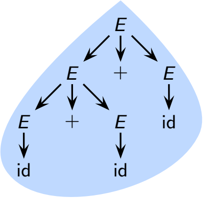
$$\begin{aligned} E &\overset{rm}{\Rightarrow} E * E \\ &\overset{rm}{\Rightarrow} E * E + E \\ &\overset{rm}{\Rightarrow} E * E + \text{id} \end{aligned}$$



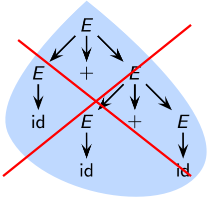
Step 1: Precedence and Associativity Rule Out Undesirable Derivations

Input id + id + id

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + \text{id} \\ &\overset{rm}{\Rightarrow} E + E + \text{id} \\ &\overset{rm}{\Rightarrow} E + \text{id} + \text{id} \\ &\overset{rm}{\Rightarrow} \text{id} + \text{id} + \text{id} \end{aligned}$$



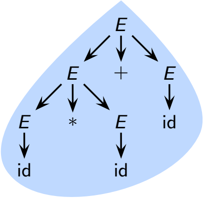
$$\xcancel{\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + E + E \\ &\overset{rm}{\Rightarrow} E + E + \text{id} \\ &\overset{rm}{\Rightarrow} E + \text{id} + \text{id} \\ &\overset{rm}{\Rightarrow} \text{id} + \text{id} + \text{id} \end{aligned}}$$



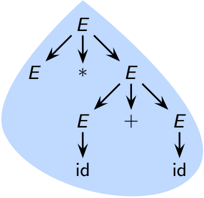
+ is left associative

Input id ∗ id + id

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + \text{id} \\ &\overset{rm}{\Rightarrow} E * E + \text{id} \\ &\overset{rm}{\Rightarrow} E * \text{id} + \text{id} \\ &\overset{rm}{\Rightarrow} \text{id} * \text{id} + \text{id} \end{aligned}$$



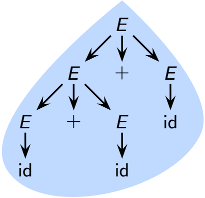
$$\begin{aligned} E &\overset{rm}{\Rightarrow} E * E \\ &\overset{rm}{\Rightarrow} E * E + E \\ &\overset{rm}{\Rightarrow} E * E + \text{id} \\ &\overset{rm}{\Rightarrow} E * \text{id} + \text{id} \end{aligned}$$



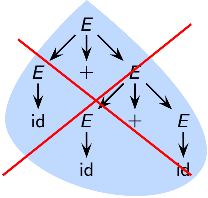
Step 1: Precedence and Associativity Rule Out Undesirable Derivations

$\boxed{\text{Input id} + \text{id} + \text{id}}$

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + \text{id} \\ &\overset{rm}{\Rightarrow} E + E + \text{id} \\ &\overset{rm}{\Rightarrow} E + \text{id} + \text{id} \\ &\overset{rm}{\Rightarrow} \text{id} + \text{id} + \text{id} \end{aligned}$$



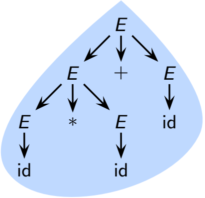
~~$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + E + E \\ &\overset{rm}{\Rightarrow} E + E + \text{id} \\ &\overset{rm}{\Rightarrow} E + \text{id} + \text{id} \\ &\overset{rm}{\Rightarrow} \text{id} + \text{id} + \text{id} \end{aligned}$$~~



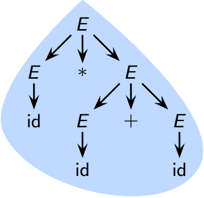
$\boxed{+ \text{ is left associative}}$

$\boxed{\text{Input id} * \text{id} + \text{id}}$

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + \text{id} \\ &\overset{rm}{\Rightarrow} E * E + \text{id} \\ &\overset{rm}{\Rightarrow} E * \text{id} + \text{id} \\ &\overset{rm}{\Rightarrow} \text{id} * \text{id} + \text{id} \end{aligned}$$



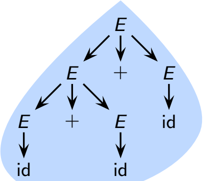
$$\begin{aligned} E &\overset{rm}{\Rightarrow} E * E \\ &\overset{rm}{\Rightarrow} E * E + E \\ &\overset{rm}{\Rightarrow} E * E + \text{id} \\ &\overset{rm}{\Rightarrow} E * \text{id} + \text{id} \\ &\overset{rm}{\Rightarrow} \text{id} * \text{id} + \text{id} \end{aligned}$$



Step 1: Precedence and Associativity Rule Out Undesirable Derivations

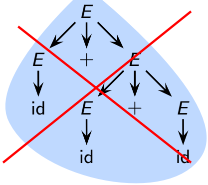
Input id + id + id

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + \text{id} \\ &\overset{rm}{\Rightarrow} E + E + \text{id} \\ &\overset{rm}{\Rightarrow} E + \text{id} + \text{id} \\ &\overset{rm}{\Rightarrow} \text{id} + \text{id} + \text{id} \end{aligned}$$



Left Associativity of + (Left Side): The non-crossed derivation shows that id + id + id is parsed as (id + id) + id, ensuring left associativity. The incorrect right-associative parse (id + (id + id)) is crossed out.

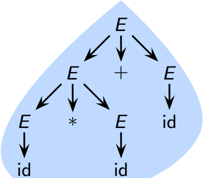
$$\xcancel{\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + E + E \\ &\overset{rm}{\Rightarrow} E + E + \text{id} \\ &\overset{rm}{\Rightarrow} E + \text{id} + \text{id} \\ &\overset{rm}{\Rightarrow} \text{id} + \text{id} + \text{id} \end{aligned}}$$



$\boxed{+ \text{ is left associative}}$

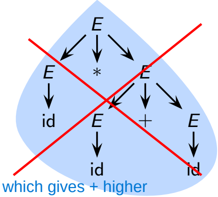
Input id ∗ id + id

$$\begin{aligned} E &\overset{rm}{\Rightarrow} E + E \\ &\overset{rm}{\Rightarrow} E + \text{id} \\ &\overset{rm}{\Rightarrow} E * E + \text{id} \\ &\overset{rm}{\Rightarrow} E * \text{id} + \text{id} \\ &\overset{rm}{\Rightarrow} \text{id} * \text{id} + \text{id} \end{aligned}$$



Higher Precedence of * Over + (Right Side): The correct parse ensures id * id + id is interpreted as (id * id) + id, following operator precedence.

$$\xcancel{\begin{aligned} E &\overset{rm}{\Rightarrow} E * E \\ &\overset{rm}{\Rightarrow} E * E + E \\ &\overset{rm}{\Rightarrow} E * E + \text{id} \\ &\overset{rm}{\Rightarrow} E * \text{id} + \text{id} \\ &\overset{rm}{\Rightarrow} \text{id} * \text{id} + \text{id} \end{aligned}}$$



The incorrect parse (id) * (id + id), which gives + higher precedence, is crossed out.

$\boxed{* \text{ has a higher precedence than } +}$

Step 1: Precedence and Associativity Rule Out Undesirable Derivations

Input id + id + id

Input id ∗ id + id

$E \overset{rm}{\Rightarrow} E + E$

$E \overset{rm}{\Rightarrow} E + E$

The moral of the story

- Right sentential forms containing the strings $E + E + E$, $E * E * E$, and $E * E + E$ are ruled out by our choice of precedence and associativity
- The grouping that we want is $(E + E) + E$, $(E * E) * E$, and $(E * E) + E$ so the non-terminals in the parenthesis should be derived first
- However, the parenthesized term does not occur in the rightmost position and hence it cannot be derived first in a rightmost derivation
- The string $E + E * E$ can appear in a rightmost derivation because the grouping is $E + (E * E)$ and the parenthesized term occurs in the rightmost position

+ is left associative

∗ has a higher precedence than +

Step 2: Shift Reduce Actions, Rightmost Derivations, and Handles

$$E \overset{rm}{\Rightarrow} E + E \overset{rm}{\Rightarrow} E + E * E \overset{rm}{\Rightarrow} E + E * \mathrm{id} \overset{rm}{\Rightarrow} E + \mathrm{id} * \mathrm{id} \overset{rm}{\Rightarrow} \mathrm{id} + \mathrm{id} * \mathrm{id}$$

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \mathrm{id}$

Step	Stack $\rightarrow$	Input	Action
1	$\$$	$\mathrm{id} + \mathrm{id} * \mathrm{id}\$$	shift
2	$\$\mathrm{id}$	$+ \mathrm{id} * \mathrm{id}\$$	reduce by 3
3	$\$E$	$+ \mathrm{id} * \mathrm{id}\$$	shift
4	$\$E +$	$\mathrm{id} * \mathrm{id}\$$	shift
5	$\$E + \mathrm{id}$	$* \mathrm{id}\$$	reduce by 3
6	$\$E + E$	$* \mathrm{id}\$$	shift
7	$\$E + E *$	$\mathrm{id}\$$	shift
8	$\$E + E * \mathrm{id}$	$\$$	reduce by 3
9	$\$E + E * E$	$\$$	reduce by 2
10	$\$E + E$	$\$$	reduce by 1
11	$\$E$	$\$$	accept

Step 2: Shift Reduce Actions, Rightmost Derivations, and Handles

$$E \overset{rm}{\Rightarrow} E + E \overset{rm}{\Rightarrow} E + E * E \overset{rm}{\Rightarrow} E + E * \text{id} \overset{rm}{\Rightarrow} E + \text{id} * \text{id} \overset{rm}{\Rightarrow} \boxed{\text{id} + \text{id} * \text{id}}$$

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

Step	Stack →	Input	Action
1	\$	id + id ∗ id\$	shift
2	\$id	+ id ∗ id\$	reduce by 3
3	\$$E$	+ id ∗ id\$	shift
4	\$$E$ +	id ∗ id\$	shift
5	\$$E$ + id	∗ id\$	reduce by 3
6	\$$E$ + E	∗ id\$	shift
7	\$$E$ + E ∗	id\$	shift
8	\$$E$ + E ∗ id	\$	reduce by 3
9	\$$E$ + E ∗ E	\$	reduce by 2
10	\$$E$ + E	\$	reduce by 1
11	\$$E$	\$	accept

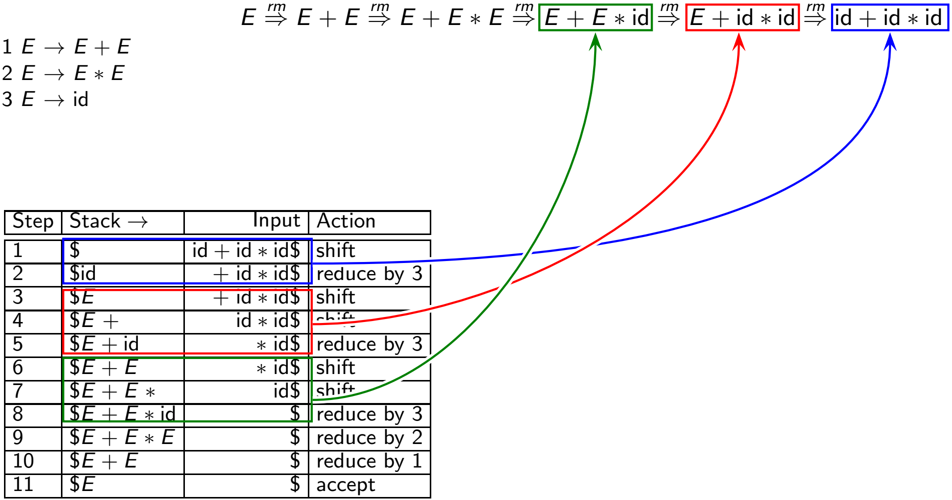
Step 2: Shift Reduce Actions, Rightmost Derivations, and Handles

$$E \overset{rm}{\Rightarrow} E+E \overset{rm}{\Rightarrow} E+E*E \overset{rm}{\Rightarrow} E+E*\text{id} \overset{rm}{\Rightarrow} \boxed{E+\text{id}*\text{id}} \overset{rm}{\Rightarrow} \boxed{\text{id}+\text{id}*\text{id}}$$

1 $E \rightarrow E+E$
2 $E \rightarrow E*E$
3 $E \rightarrow \text{id}$

Step	Stack →	Input	Action
1	\$	id + id ∗ id\$	shift
2	\$id	+ id ∗ id\$	reduce by 3
3	\$$E$	+ id ∗ id\$	shift
4	\$$E$ +	id ∗ id\$	shift
5	\$$E$ + id	∗ id\$	reduce by 3
6	\$$E+E$	∗ id\$	shift
7	\$$E+E$ ∗	id\$	shift
8	\$$E+E$ ∗ id	\$	reduce by 3
9	\$$E+E*E$	\$	reduce by 2
10	\$$E+E$	\$	reduce by 1
11	\$$E$	\$	accept

Step 2: Shift Reduce Actions, Rightmost Derivations, and Handles



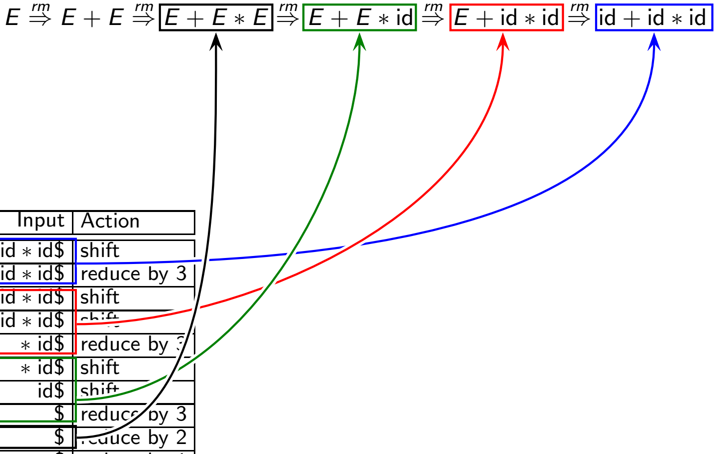
$$E \overset{rm}{\Rightarrow} E+E \overset{rm}{\Rightarrow} E+E*E \overset{rm}{\Rightarrow} \boxed{E+E*\text{id}} \overset{rm}{\Rightarrow} \boxed{E+\text{id}*\text{id}} \overset{rm}{\Rightarrow} \boxed{\text{id}+\text{id}*\text{id}}$$

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

Step	Stack →	Input	Action
1	$\$$	$\text{id} + \text{id} * \text{id}\$$	shift
2	$\$\text{id}$	$+ \text{id} * \text{id}\$$	reduce by 3
3	$\$E$	$+ \text{id} * \text{id}\$$	shift
4	$\$E +$	$\text{id} * \text{id}\$$	shift
5	$\$E + \text{id}$	$* \text{id}\$$	reduce by 3
6	$\$E + E$	$* \text{id}\$$	shift
7	$\$E + E *$	$\text{id}\$$	shift
8	$\$E + E * \text{id}$	$\$$	reduce by 3
9	$\$E + E * E$	$\$$	reduce by 2
10	$\$E + E$	$\$$	reduce by 1
11	$\$E$	$\$$	accept

Step 2: Shift Reduce Actions, Rightmost Derivations, and Handles

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

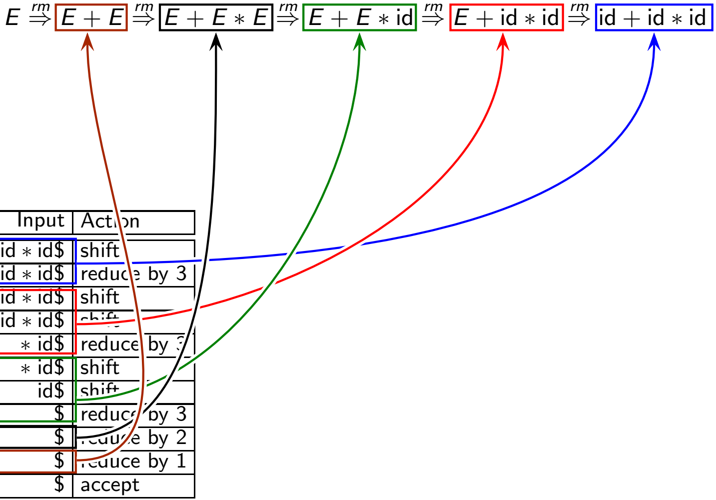


$$E \overset{rm}{\Rightarrow} E + E \overset{rm}{\Rightarrow} \boxed{E + E * E} \overset{rm}{\Rightarrow} \boxed{E + E * \text{id}} \overset{rm}{\Rightarrow} \boxed{E + \text{id} * \text{id}} \overset{rm}{\Rightarrow} \boxed{\text{id} + \text{id} * \text{id}}$$

Step	Stack →	Input	Action
1	\$	id + id ∗ id\$	shift
2	\$id	+ id ∗ id\$	reduce by 3
3	\$$E$	+ id ∗ id\$	shift
4	\$$E$ +	id ∗ id\$	shift
5	\$$E$ + id	∗ id\$	reduce by 3
6	\$$E$ + E	∗ id\$	shift
7	\$$E$ + E ∗	id\$	shift
8	\$$E$ + E ∗ id	\$	reduce by 3
9	\$$E$ + E ∗ E	\$	reduce by 2
10	\$$E$ + E	\$	reduce by 1
11	\$$E$	\$	accept

Step 2: Shift Reduce Actions, Rightmost Derivations, and Handles

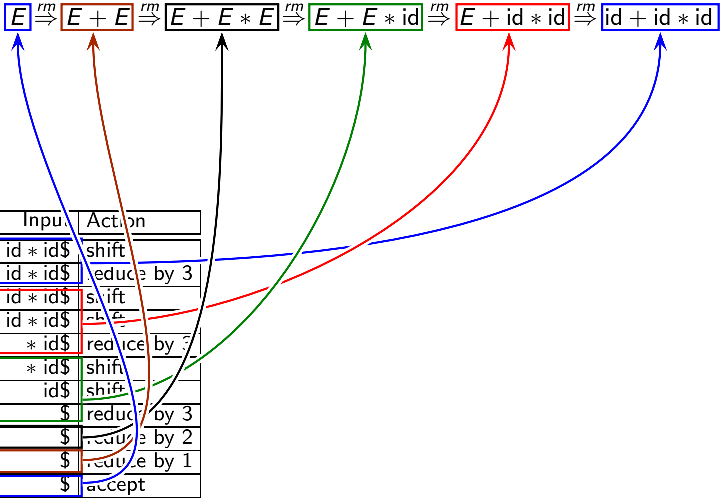
1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$



$$E \overset{rm}{\Rightarrow} \boxed{E + E} \overset{rm}{\Rightarrow} \boxed{E + E * E} \overset{rm}{\Rightarrow} \boxed{E + E * \text{id}} \overset{rm}{\Rightarrow} \boxed{E + \text{id} * \text{id}} \overset{rm}{\Rightarrow} \boxed{\text{id} + \text{id} * \text{id}}$$

Step	Stack →	Input	Action
1	$	id + id ∗ id$	shift
2	$id	+ id ∗ id$	reduce by 3
3	$E	+ id ∗ id$	shift
4	$E +	id ∗ id$	shift
5	$E + id	∗ id$	reduce by 3
6	$E + E	∗ id$	shift
7	$E + E ∗	id$	shift
8	$E + E ∗ id	$	reduce by 3
9	$E + E ∗ E	$	reduce by 2
10	$E + E	$	reduce by 1
11	$E	$	accept

Step 2: Shift Reduce Actions, Rightmost Derivations, and Handles

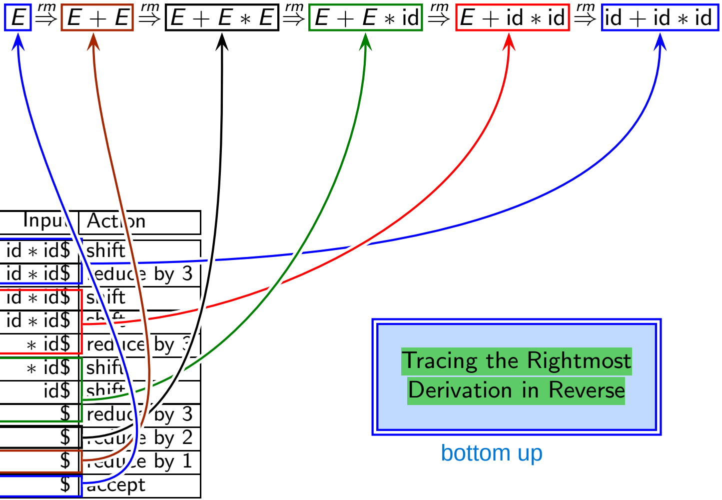


$$\boxed{E} \overset{rm}{\Rightarrow} \boxed{E+E} \overset{rm}{\Rightarrow} \boxed{E+E*E} \overset{rm}{\Rightarrow} \boxed{E+E*\text{id}} \overset{rm}{\Rightarrow} \boxed{E+\text{id}*\text{id}} \overset{rm}{\Rightarrow} \boxed{\text{id}+\text{id}*\text{id}}$$

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

Step	Stack →	Input	Action
1	\$	id + id ∗ id\$	shift
2	\$id	+ id ∗ id\$	reduce by 3
3	\$$E$	+ id ∗ id\$	shift
4	\$$E$ +	id ∗ id\$	shift
5	\$$E$ + id	∗ id\$	reduce by 3
6	\$$E$ + E	∗ id\$	shift
7	\$$E$ + E ∗	id\$	shift
8	\$$E$ + E ∗ id	\$	reduce by 3
9	\$$E$ + E ∗ E	\$	reduce by 2
10	\$$E$ + E	\$	reduce by 1
11	\$$E$	\$	accept

Step 2: Shift Reduce Actions, Rightmost Derivations, and Handles



$$\boxed{E} \overset{rm}{\Rightarrow} \boxed{E+E} \overset{rm}{\Rightarrow} \boxed{E+E*E} \overset{rm}{\Rightarrow} \boxed{E+E*\text{id}} \overset{rm}{\Rightarrow} \boxed{E+\text{id}*\text{id}} \overset{rm}{\Rightarrow} \boxed{\text{id}+\text{id}*\text{id}}$$

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

Step	Stack →	Input	Action
1	\$	id + id ∗ id\$	shift
2	\$id	+ id ∗ id\$	reduce by 3
3	\$$E$	+ id ∗ id\$	shift
4	\$$E$ +	id ∗ id\$	shift
5	\$$E$ + id	∗ id\$	reduce by 3
6	\$$E + E$	∗ id\$	shift
7	\$$E + E$ ∗	id\$	shift
8	\$$E + E$ ∗ id	\$	reduce by 3
9	\$$E + E * E$	\$	reduce by 2
10	\$$E + E$	\$	reduce by 1
11	\$$E$	\$	accept

Tracing the Rightmost Derivation in Reverse

bottom up

Step 2: Shift Reduce Actions, Rightmost Derivations, and Handles

$$E \overset{rm}{\Rightarrow} E+E \overset{rm}{\Rightarrow} E+E*E \overset{rm}{\Rightarrow} E+E*\text{id} \overset{rm}{\Rightarrow} E+\text{id}*\text{id} \overset{rm}{\Rightarrow} \text{id}+\text{id}*\text{id}$$

1 $E \rightarrow E+E$
2 $E \rightarrow E*E$
3 $E \rightarrow \text{id}$

Step	Stack →	Input	Action
1	\$	$\text{id}+\text{id}*\text{id}\$$	shift
2	\$id	$+\text{id}*\text{id}\$$	reduce by 3
3	$\$E$	$+\text{id}*\text{id}\$$	shift
4	$\$E+$	$\text{id}*\text{id}\$$	shift
5	$\$E+\text{id}$	$*\text{id}\$$	reduce by 3
6	$\$E+E$	$*\text{id}\$$	shift
7	$\$E+E*$	$\text{id}\$$	shift
8	$\$E+E*\text{id}$	\$	reduce by 3
9	$\$E+E*E$	\$	reduce by 2
10	$\$E+E$	\$	reduce by 1
11	$\$E$	\$	accept

Rightmost derivations are traced in reverse by identifying handles in right sentential forms (beginning with the sentence) and pruning them for constructing the previous right sentential form

Step 2: Shift Reduce Actions, Rightmost Derivations, and Handles

$$E \overset{rm}{\Rightarrow} E + E \overset{rm}{\Rightarrow} E + E * E \overset{rm}{\Rightarrow} E + E * \text{id} \overset{rm}{\Rightarrow} \boxed{E} + \text{id} * \text{id} \overset{rm}{\Rightarrow} \boxed{\boxed{\text{id}} + \text{id} * \text{id}}$$

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
$\boxed{3 \ E \rightarrow \text{id}}$

Step	Stack $\rightarrow$	Input	Action
1	\$	id + id ∗ id\$	shift
2	\$id	+ id ∗ id\$	reduce by 3
3	$\$E$	+ id ∗ id\$	shift
4	$\$E +$	id ∗ id\$	shift
5	$\$E +$ id	∗ id\$	reduce by 3
6	$\$E + E$	∗ id\$	shift
7	$\$E + E *$	id\$	shift
8	$\$E + E *$ id	\$	reduce by 3
9	$\$E + E * E$	\$	reduce by 2
10	$\$E + E$	\$	reduce by 1
11	$\$E$	\$	accept

Handle

Right Sentential Form

Step 2: Shift Reduce Actions, Rightmost Derivations, and Handles

$$E \overset{rm}{\Rightarrow} E + E \overset{rm}{\Rightarrow} E + E * E \overset{rm}{\Rightarrow} E + \boxed{E} * \text{id} \overset{rm}{\Rightarrow} \boxed{E + \boxed{\text{id}} * \text{id}} \overset{rm}{\Rightarrow} \text{id} + \text{id} * \text{id}$$

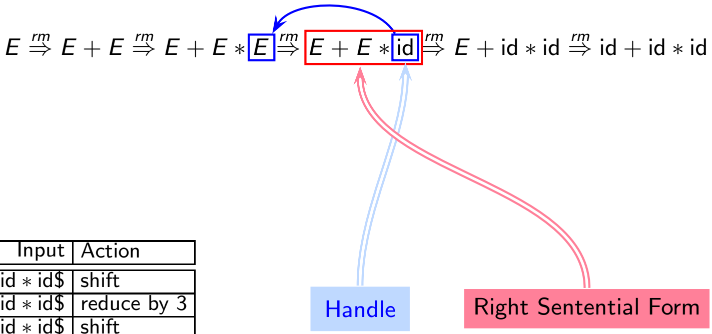
Handle

Right Sentential Form

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
$\boxed{3\ E \rightarrow \text{id}}$

Step	Stack →	Input	Action
1	\$	id + id ∗ id\$	shift
2	\$id	+ id ∗ id\$	reduce by 3
3	\$$E$	+ id ∗ id\$	shift
4	\$$E$ +	id ∗ id\$	shift
5	\$$E$ + id	∗ id\$	reduce by 3
6	\$$E$ + E	∗ id\$	shift
7	\$$E$ + E ∗	id\$	shift
8	\$$E$ + E ∗ id	\$	reduce by 3
9	\$$E$ + E ∗ E	\$	reduce by 2
10	\$$E$ + E	\$	reduce by 1
11	\$$E$	\$	accept

Step 2: Shift Reduce Actions, Rightmost Derivations, and Handles



$$E \overset{rm}{\Rightarrow} E + E \overset{rm}{\Rightarrow} E + E * \boxed{E} \overset{rm}{\Rightarrow} \boxed{E + E * \boxed{\text{id}}} \overset{rm}{\Rightarrow} E + \text{id} * \text{id} \overset{rm}{\Rightarrow} \text{id} + \text{id} * \text{id}$$

Handle

Right Sentential Form

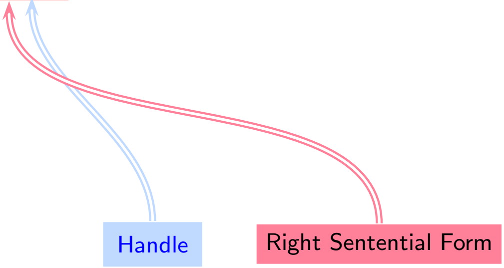
1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
$\boxed{3\ E \rightarrow \text{id}}$

Step	Stack →	Input	Action
1	\$	id + id ∗ id\$	shift
2	\$id	+ id ∗ id\$	reduce by 3
3	\$$E$	+ id ∗ id\$	shift
4	\$$E$ +	id ∗ id\$	shift
5	\$$E$ + id	∗ id\$	reduce by 3
6	\$$E + E$	∗ id\$	shift
7	\$$E + E$ ∗	id\$	shift
8	\$$E + E$ ∗ id	\$	reduce by 3
9	\$$E + E * E$	\$	reduce by 2
10	\$$E + E$	\$	reduce by 1
11	\$$E$	\$	accept

Step 2: Shift Reduce Actions, Rightmost Derivations, and Handles

$$E \overset{rm}{\Rightarrow} E + \boxed{E} \overset{rm}{\Rightarrow} \boxed{E + \boxed{E * E}} \overset{rm}{\Rightarrow} E + E * \text{id} \overset{rm}{\Rightarrow} E + \text{id} * \text{id} \overset{rm}{\Rightarrow} \text{id} + \text{id} * \text{id}$$

1 $E \rightarrow E + E$
$\boxed{2\ E \rightarrow E * E}$
3 $E \rightarrow \text{id}$



Handle

Right Sentential Form

Step	Stack →	Input	Action
1	\$	id + id ∗ id\$	shift
2	\$id	+ id ∗ id\$	reduce by 3
3	\$$E$	+ id ∗ id\$	shift
4	\$$E$ +	id ∗ id\$	shift
5	\$$E$ + id	∗ id\$	reduce by 3
6	\$$E$ + E	∗ id\$	shift
7	\$$E$ + E ∗	id\$	shift
8	\$$E$ + E ∗ id	\$	reduce by 3
9	\$$E$ + E ∗ E	\$	reduce by 2
10	\$$E$ + E	\$	reduce by 1
11	\$$E$	\$	accept

Step 2: Shift Reduce Actions, Rightmost Derivations, and Handles

$$\boxed{E} \overset{rm}{\Rightarrow} \boxed{\boxed{E+E}} \overset{rm}{\Rightarrow} E+E*E \overset{rm}{\Rightarrow} E+E*\text{id} \overset{rm}{\Rightarrow} E+\text{id}*\text{id} \overset{rm}{\Rightarrow} \text{id}+\text{id}*\text{id}$$

$\boxed{1\ E \rightarrow E+E}$
$2\ E \rightarrow E*E$
$3\ E \rightarrow \text{id}$

Handle

Right Sentential Form

Step	Stack →	Input	Action
1	\$	id + id ∗ id\$	shift
2	\$id	+ id ∗ id\$	reduce by 3
3	\$$E$	+ id ∗ id\$	shift
4	\$$E$ +	id ∗ id\$	shift
5	\$$E$ + id	∗ id\$	reduce by 3
6	\$$E$ + E	∗ id\$	shift
7	\$$E$ + E ∗	id\$	shift
8	\$$E$ + E ∗ id	\$	reduce by 3
9	\$$E$ + E ∗ E	\$	reduce by 2
10	\$$E$ + E	\$	reduce by 1
11	\$$E$	\$	accept

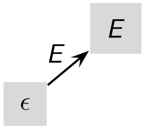
Step 3: Identifying Handles in Right Sentential Forms

- Our goal is to discover a prefix of right sentential form that ends with a handle
- Viable Prefix. A prefix of a right sentential form that does not extend beyond the handle
    - It is either a string with no handle, or first kind
    - a string that ends with the handle second kind
- By suffixing appropriate symbols to a viable prefix of the first kind, we can create a viable prefix of the second kind
- By suffixing terminal symbols to the viable prefix of the second kind, we can create a right sentential form
- The set of viable prefixes forms a regular language, thus they can be recognized by a DFA
- The handles in a viable prefix can be identified using a stack
- We keep pushing viable prefixes on the stack until the handle appears on the top of the stack

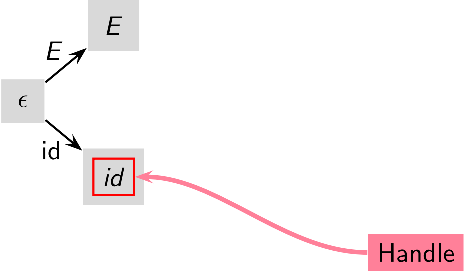
Step 3: Viable Prefixes for Our Grammar (After Incorporating Precedences and Associativities)

ϵ

Step 3: Viable Prefixes for Our Grammar (After Incorporating Precedences and Associativities)

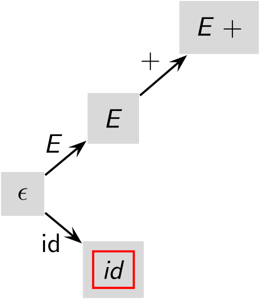


Step 3: Viable Prefixes for Our Grammar (After Incorporating Precedences and Associativities)

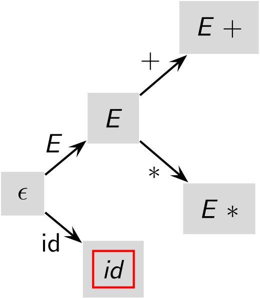


Viable prefix id must be reduced to E and no grammar symbol can be suffixed to it (because there is no rule with a symbol after id)

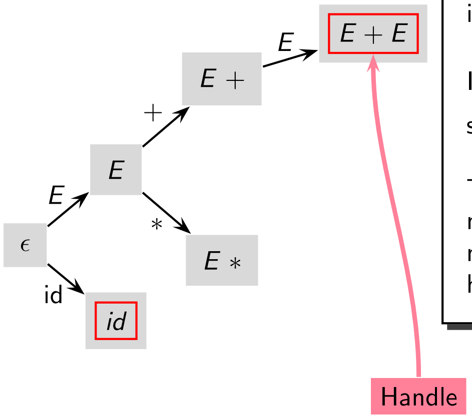
Step 3: Viable Prefixes for Our Grammar (After Incorporating Precedences and Associativities)



Step 3: Viable Prefixes for Our Grammar (After Incorporating Precedences and Associativities)



Step 3: Viable Prefixes for Our Grammar (After Incorporating Precedences and Associativities)

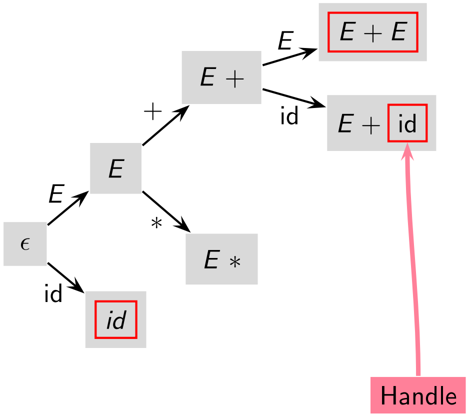


Viable prefix $E + E$ must be reduced to E if it is not followed by a "*"

If $E + E$ is followed by a "*", "*" should be shifted and $E + E$ should not be reduced

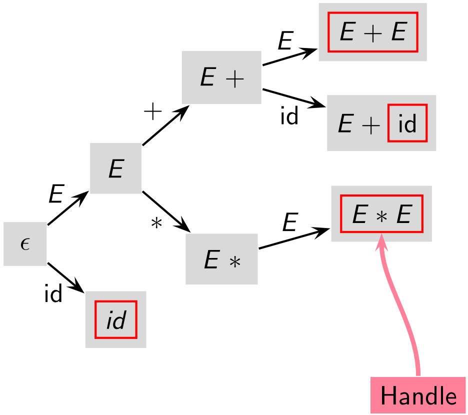
The occurrence of a potential handle does not mean it should be reduced, the next terminal symbol decides whether it is an actual handle (and if so, it should be reduced)

Step 3: Viable Prefixes for Our Grammar (After Incorporating Precedences and Associativities)



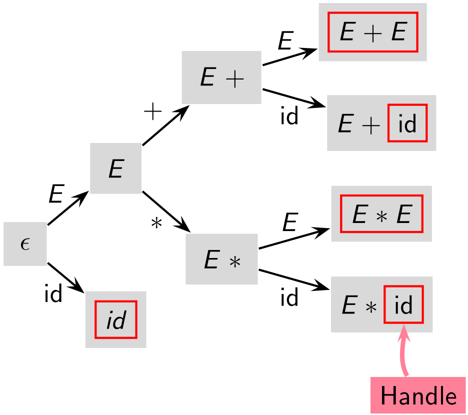
Viable prefix $E+\text{id}$ must be reduced to $E+E$ and no grammar symbol can be suffixed to it

Step 3: Viable Prefixes for Our Grammar (After Incorporating Precedences and Associativities)



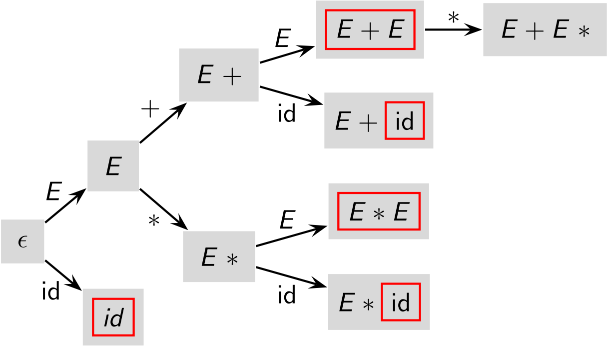
Viable prefix $E * E$ must be reduced to E and no grammar symbol can be suffixed to it

Step 3: Viable Prefixes for Our Grammar (After Incorporating Precedences and Associativities)

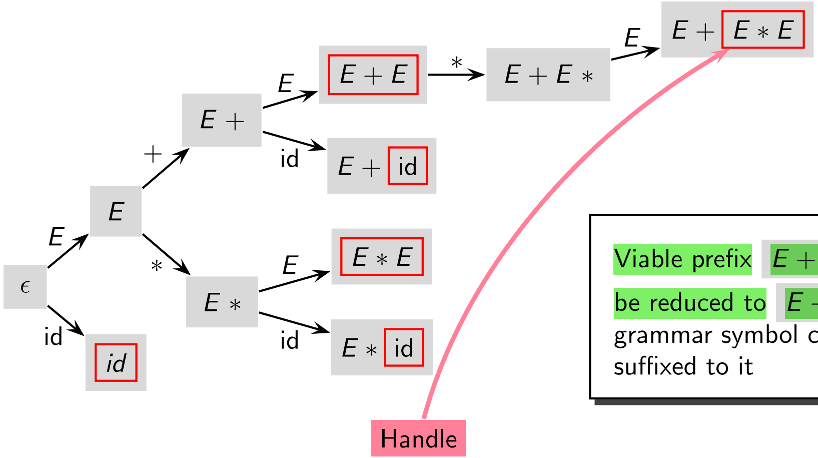


Viable prefix $E *$ id must be reduced to $E * E$ and no grammar symbol can be suffixed to it

Step 3: Viable Prefixes for Our Grammar (After Incorporating Precedences and Associativities)

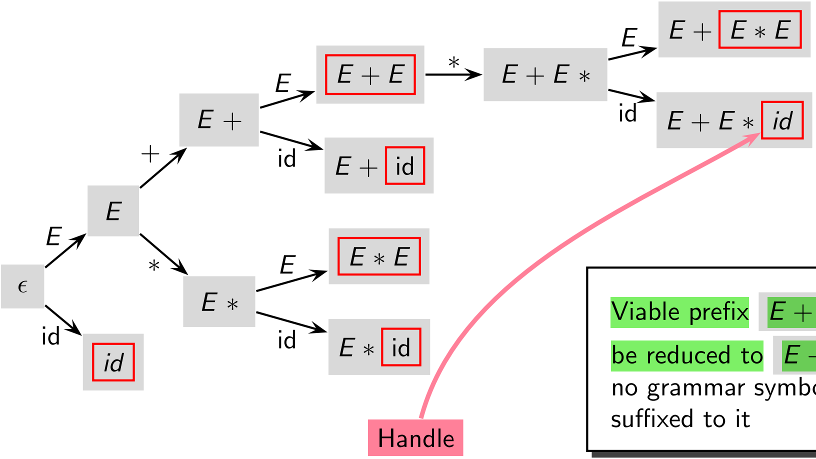


Step 3: Viable Prefixes for Our Grammar (After Incorporating Precedences and Associativities)



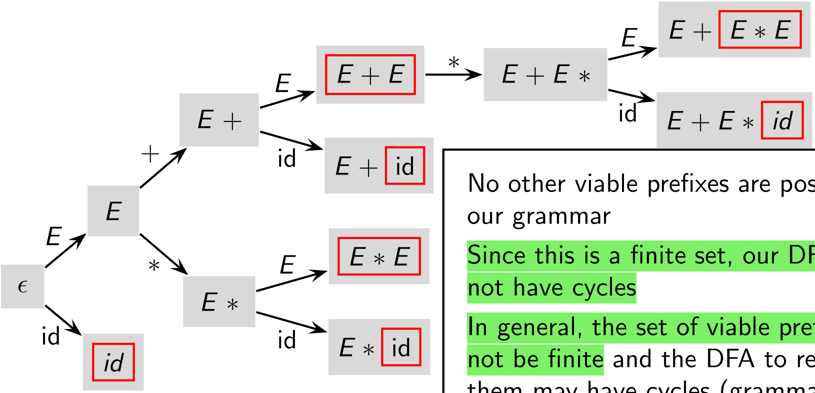
Viable prefix $E+E*E$ must be reduced to $E+E$ and no grammar symbol can be suffixed to it

Step 3: Viable Prefixes for Our Grammar (After Incorporating Precedences and Associativities)



Viable prefix $E + E * \text{id}$ must be reduced to $E + E * E$ and no grammar symbol can be suffixed to it

Step 3: Viable Prefixes for Our Grammar (After Incorporating Precedences and Associativities)



No other viable prefixes are possible for our grammar

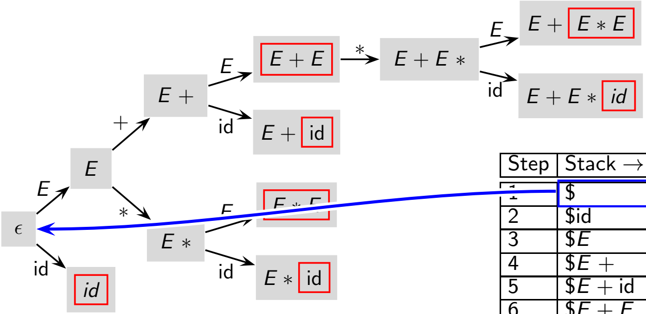
Since this is a finite set, our DFA would not have cycles

In general, the set of viable prefixes need not be finite and the DFA to recognize them may have cycles (grammar $L \rightarrow \text{id}, L \mid \text{id}$ has infinitely many viable prefixes and its DFA has cycles)

Step 3: Viable Prefixes for Our Example

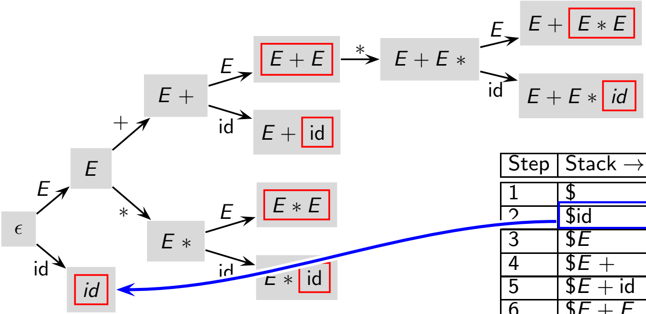
Step	Stack $\rightarrow$	Input	Action
1	\$	id + id ∗ id\$	shift
2	\$id	+ id ∗ id\$	reduce by 3
3	$\$E$	+ id ∗ id\$	shift
4	$\$E +$	id ∗ id\$	shift
5	$\$E +$ id	∗ id\$	reduce by 3
6	$\$E + E$	∗ id\$	shift
7	$\$E + E *$	id\$	shift
8	$\$E + E *$ id	\$	reduce by 3
9	$\$E + E * E$	\$	reduce by 2
10	$\$E + E$	\$	reduce by 1
11	$\$E$	\$	accept

Step 3: Viable Prefixes for Our Example



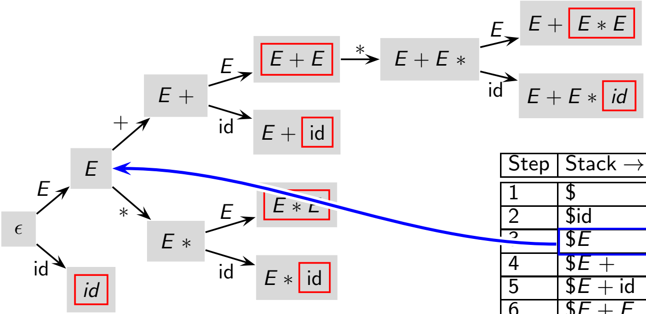
Step	Stack →	Input	Action
1	\$	id + id ∗ id\$	shift
2	\$id	+ id ∗ id\$	reduce by 3
3	\$$E$	+ id ∗ id\$	shift
4	\$$E$ +	id ∗ id\$	shift
5	\$$E$ + id	∗ id\$	reduce by 3
6	\$$E$ + E	∗ id\$	shift
7	\$$E$ + E ∗	id\$	shift
8	\$$E$ + E ∗ id	\$	reduce by 3
9	\$$E$ + E ∗ E	\$	reduce by 2
10	\$$E$ + E	\$	reduce by 1
11	\$$E$	\$	accept

Step 3: Viable Prefixes for Our Example



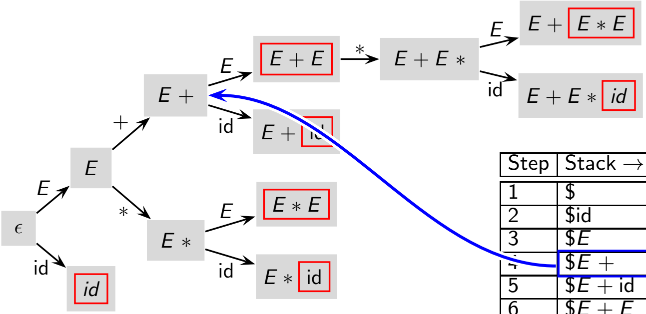
Step	Stack →	Input	Action
1	\$	id + id ∗ id\$	shift
2	\$id	+ id ∗ id\$	reduce by 3
3	\$$E$	+ id ∗ id\$	shift
4	\$$E$ +	id ∗ id\$	shift
5	\$$E$ + id	∗ id\$	reduce by 3
6	\$$E + E$	∗ id\$	shift
7	\$$E + E$ ∗	id\$	shift
8	\$$E + E$ ∗ id	\$	reduce by 3
9	\$$E + E * E$	\$	reduce by 2
10	\$$E + E$	\$	reduce by 1
11	\$$E$	\$	accept

Step 3: Viable Prefixes for Our Example



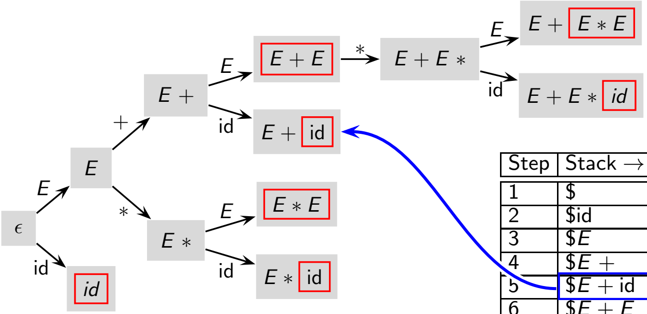
Step	Stack →	Input	Action
1	$	id + id ∗ id$	shift
2	$id	+ id ∗ id$	reduce by 3
3	$E	+ id ∗ id$	shift
4	$E +	id ∗ id$	shift
5	$E + id	∗ id$	reduce by 3
6	$E + E	∗ id$	shift
7	$E + E ∗	id$	shift
8	$E + E ∗ id	$	reduce by 3
9	$E + E ∗ E	$	reduce by 2
10	$E + E	$	reduce by 1
11	$E	$	accept

Step 3: Viable Prefixes for Our Example



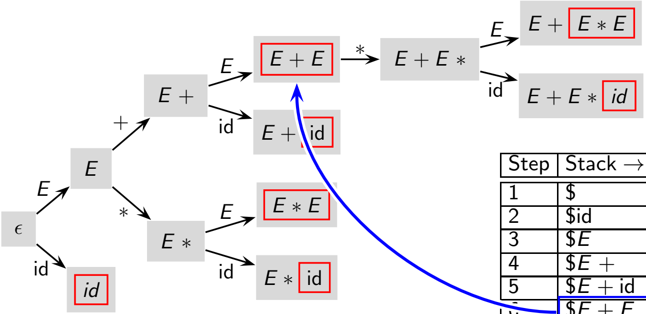
Step	Stack →	Input	Action
1	$	id + id ∗ id$	shift
2	$id	+ id ∗ id$	reduce by 3
3	$$E$	+ id ∗ id$	shift
4	$$E +$	id ∗ id$	shift
5	$$E +$ id	∗ id$	reduce by 3
6	$$E + E$	∗ id$	shift
7	$$E + E *$	id$	shift
8	$$E + E *$ id	$	reduce by 3
9	$$E + E * E$	$	reduce by 2
10	$$E + E$	$	reduce by 1
11	$$E$	$	accept

Step 3: Viable Prefixes for Our Example



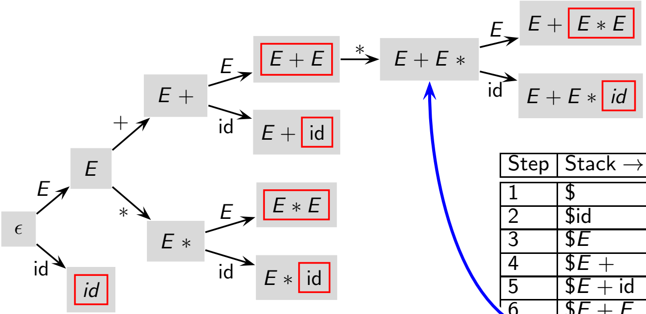
Step	Stack →	Input	Action
1	$\$$	$\text{id} + \text{id} * \text{id}\$$	shift
2	$\$\text{id}$	$+ \text{id} * \text{id}\$$	reduce by 3
3	$\$E$	$+ \text{id} * \text{id}\$$	shift
4	$\$E\,+$	$\text{id} * \text{id}\$$	shift
5	$\$E + \text{id}$	$* \text{id}\$$	reduce by 3
6	$\$E + E$	$* \text{id}\$$	shift
7	$\$E + E\,*$	$\text{id}\$$	shift
8	$\$E + E * \text{id}$	$\$$	reduce by 3
9	$\$E + E * E$	$\$$	reduce by 2
10	$\$E + E$	$\$$	reduce by 1
11	$\$E$	$\$$	accept

Step 3: Viable Prefixes for Our Example



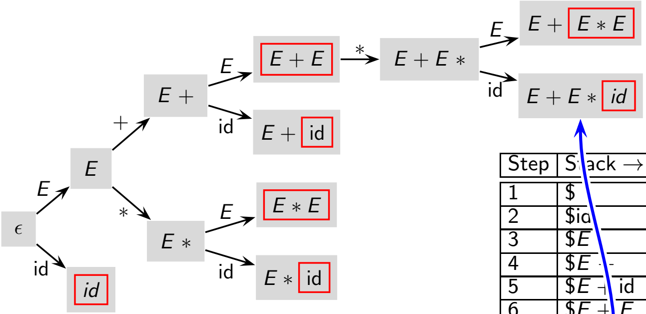
Step	Stack $\rightarrow$	Input	Action
1	\$	id + id * id\$	shift
2	\$id	+ id * id\$	reduce by 3
3	$\$E$	+ id * id\$	shift
4	$\$E +$	id * id\$	shift
5	$\$E +$ id	* id\$	reduce by 3
6	$\$E + E$	* id\$	shift
7	$\$E + E *$	id\$	shift
8	$\$E + E *$ id	\$	reduce by 3
9	$\$E + E * E$	\$	reduce by 2
10	$\$E + E$	\$	reduce by 1
11	$\$E$	\$	accept

Step 3: Viable Prefixes for Our Example



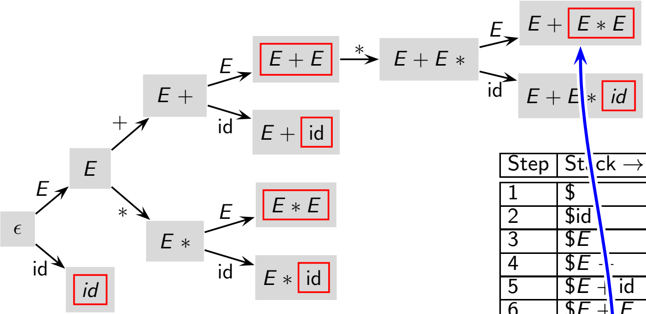
Step	Stack →	Input	Action
1	$\$$	id + id ∗ id$\$$	shift
2	$\$$id	+ id ∗ id$\$$	reduce by 3
3	$\$E$	+ id ∗ id$\$$	shift
4	$\$E+$	id ∗ id$\$$	shift
5	$\$E+$ id	∗ id$\$$	reduce by 3
6	$\$E+E$	∗ id$\$$	shift
7	$\$E+E*$	id$\$$	shift
8	$\$E+E*$ id	$\$$	reduce by 3
9	$\$E+E*E$	$\$$	reduce by 2
10	$\$E+E$	$\$$	reduce by 1
11	$\$E$	$\$$	accept

Step 3: Viable Prefixes for Our Example



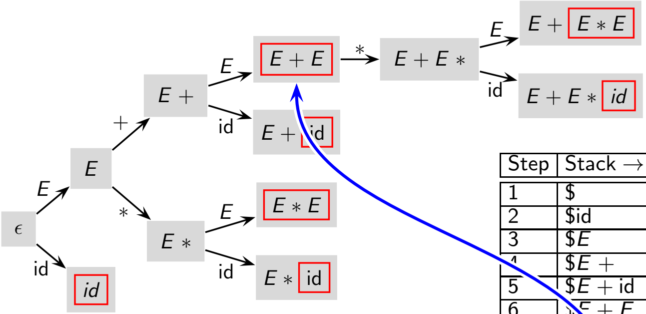
Step	Stack →	Input	Action
1	$	id + id ∗ id$	shift
2	$id	+ id ∗ id$	reduce by 3
3	$E	+ id ∗ id$	shift
4	$E +	id ∗ id$	shift
5	$E + id	∗ id$	reduce by 3
6	$E + E	∗ id$	shift
7	$E + E ∗	id$	shift
8	$E + E ∗ id	$	reduce by 3
9	$E + E ∗ E	$	reduce by 2
10	$E + E	$	reduce by 1
11	$E	$	accept

Step 3: Viable Prefixes for Our Example



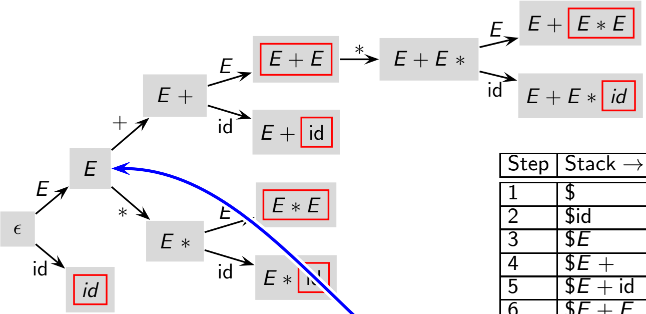
Step	Stack →	Input	Action
1	$	id + id ∗ id$	shift
2	$id	+ id ∗ id$	reduce by 3
3	$E	+ id ∗ id$	shift
4	$E +	id ∗ id$	shift
5	$E + id	∗ id$	reduce by 3
6	$E + E	∗ id$	shift
7	$E + E ∗	id$	shift
8	$E + E ∗ id	$	reduce by 3
9	$E + E ∗ E	$	reduce by 2
10	$E + E	$	reduce by 1
11	$E	$	accept

Step 3: Viable Prefixes for Our Example



Step	Stack →	Input	Action
1	\$	id + id * id\$	shift
2	\$id	+ id * id\$	reduce by 3
3	\$$E$	+ id * id\$	shift
4	\$$E$ +	id * id\$	shift
5	\$$E$ + id	* id\$	reduce by 3
6	\$$E + E$	* id\$	shift
7	\$$E + E *$	id\$	shift
8	\$$E + E *$ id	\$	reduce by 3
9	\$$E + E * E$	\$	reduce by 2
10	\$$E + E$	\$	reduce by 1
11	\$$E$	\$	accept

Step 3: Viable Prefixes for Our Example



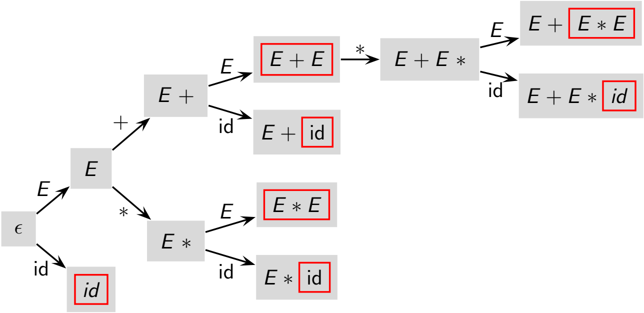
Step	Stack →	Input	Action
1	\$	id + id ∗ id\$	shift
2	\$id	+ id ∗ id\$	reduce by 3
3	\$$E$	+ id ∗ id\$	shift
4	\$$E$ +	id ∗ id\$	shift
5	\$$E$ + id	∗ id\$	reduce by 3
6	\$$E$ + E	∗ id\$	shift
7	\$$E$ + E ∗	id\$	shift
8	\$$E$ + E ∗ id	\$	reduce by 3
9	\$$E$ + E ∗ E	\$	reduce by 2
10	\$$E$ + E	\$	reduce by 1
11	\$$E$	\$	accept

Step 4: Valid Items for Viable Prefixes

- An item is a grammar production with a dot (•) in it somewhere in the RHS
- The dot separates what has been seen from what may be seen in the input
- We identify a set of items for a viable prefix to form a state of the parser

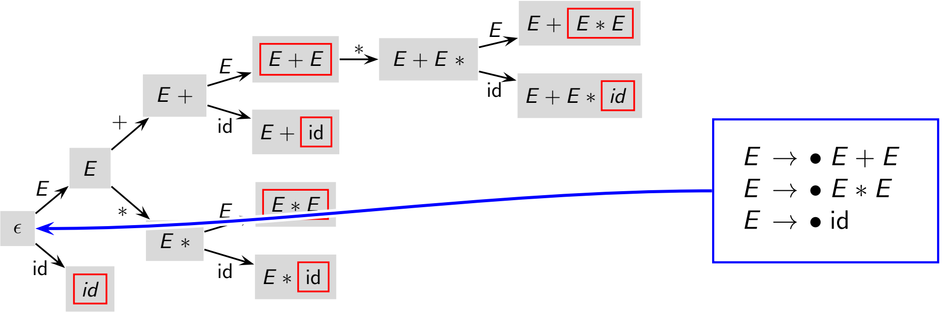
Step 4: Valid Items for Viable Prefixes

- An item is a grammar production with a dot (•) in it somewhere in the RHS
- The dot separates what has been seen from what may be seen in the input
- We identify a set of items for a viable prefix to form a state of the parser



Step 4: Valid Items for Viable Prefixes

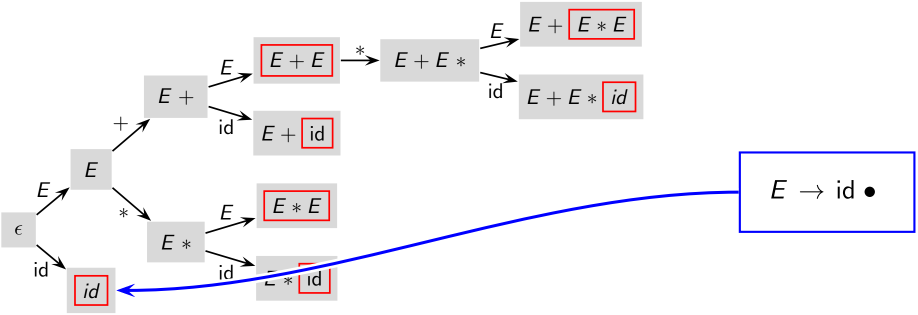
- An item is a grammar production with a dot (•) in it somewhere in the RHS
- The dot separates what has been seen from what may be seen in the input
- We identify a set of items for a viable prefix to form a state of the parser



$$E \rightarrow \bullet\, E + E$$
$$E \rightarrow \bullet\, E * E$$
$$E \rightarrow \bullet\, \text{id}$$

Step 4: Valid Items for Viable Prefixes

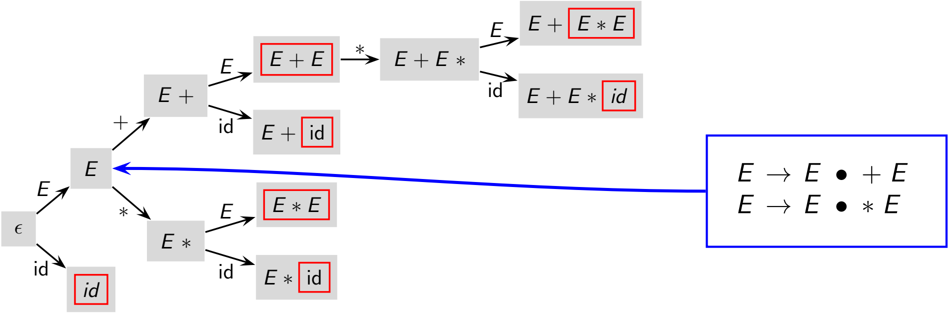
- An item is a grammar production with a dot (•) in it somewhere in the RHS
- The dot separates what has been seen from what may be seen in the input
- We identify a set of items for a viable prefix to form a state of the parser



]

Step 4: Valid Items for Viable Prefixes

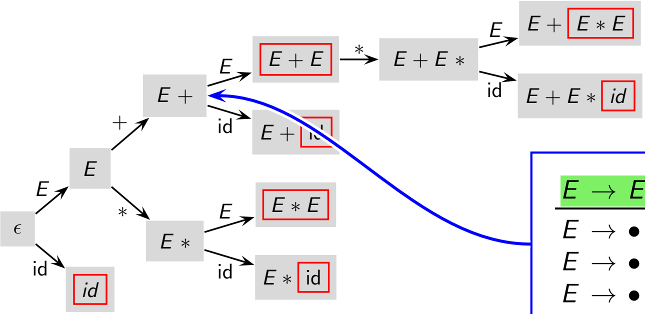
- An item is a grammar production with a dot (•) in it somewhere in the RHS
- The dot separates what has been seen from what may be seen in the input
- We identify a set of items for a viable prefix to form a state of the parser



$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

Step 4: Valid Items for Viable Prefixes

- An item is a grammar production with a dot (•) in it somewhere in the RHS
- The dot separates what has been seen from what may be seen in the input
- We identify a set of items for a viable prefix to form a state of the parser



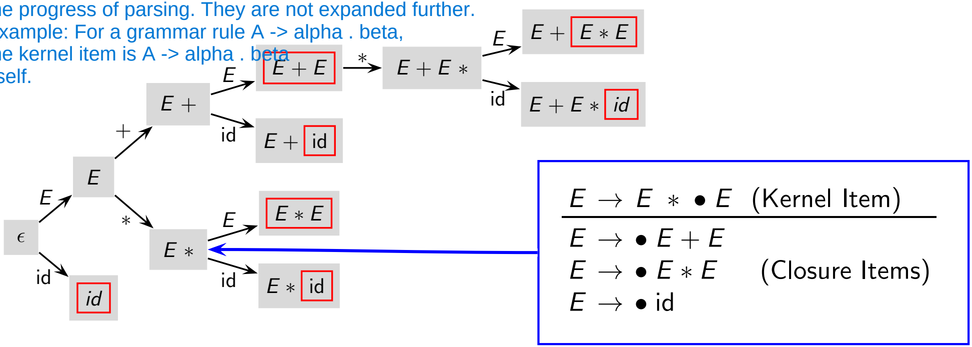
$E \rightarrow E + \bullet E$ (Kernel Item)

$E \rightarrow \bullet E + E$
$E \rightarrow \bullet E * E$ (Closure Items)
$E \rightarrow \bullet \text{id}$

Step 4: Valid Items for Viable Prefixes

- An item is a grammar production with a dot (•) in it somewhere in the RHS
- The dot separates what has been seen from what may be seen in the input
- We identify a set of items for a viable prefix to form a state of the parser

Kernel Item:
Kernel items are the core items in a parser state that are directly derived from the grammar rules and represent the progress of parsing. They are not expanded further.
Example: For a grammar rule A -> alpha . beta,
the kernel item is A -> alpha . beta
itself.



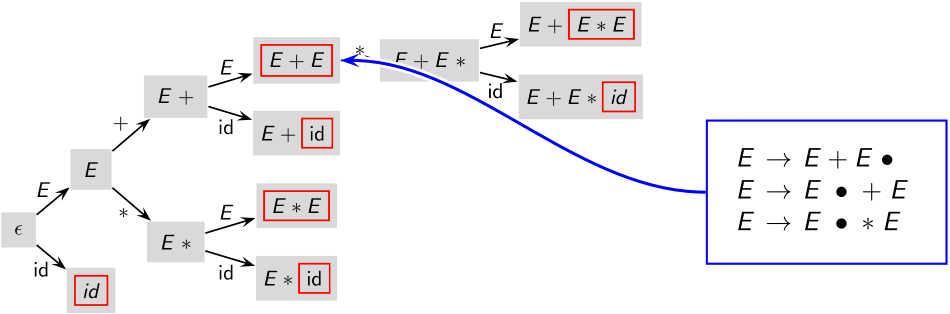
$$E \rightarrow E \ast \bullet E \quad \text{(Kernel Item)}$$

$$E \rightarrow \bullet E + E$$
$$E \rightarrow \bullet E \ast E \qquad \text{(Closure Items)}$$
$$E \rightarrow \bullet\, \text{id}$$

Closure Item:
Closure items are additional items added to a parser state by expanding non-terminals after the dot (•) in kernel items. They represent all possible productions that can be derived from the current state.
Example: If the kernel item is A -> alpha . B(beta) and B -> gamma is a production, the closure includes B -> •.gamma.

Step 4: Valid Items for Viable Prefixes

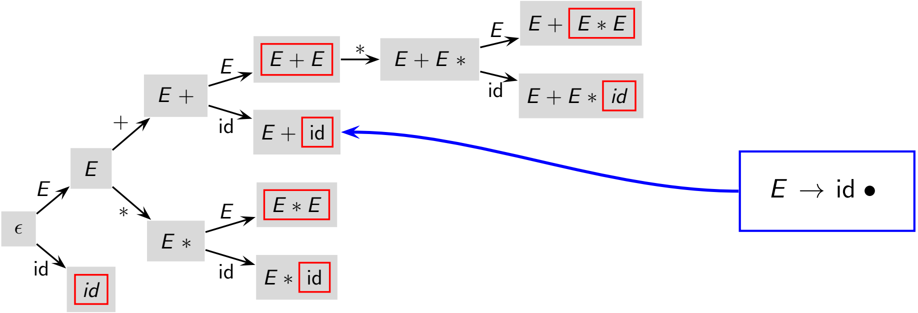
- An item is a grammar production with a dot (•) in it somewhere in the RHS
- The dot separates what has been seen from what may be seen in the input
- We identify a set of items for a viable prefix to form a state of the parser



$$E \rightarrow E + E \bullet$$
$$E \rightarrow E \bullet + E$$
$$E \rightarrow E \bullet * E$$

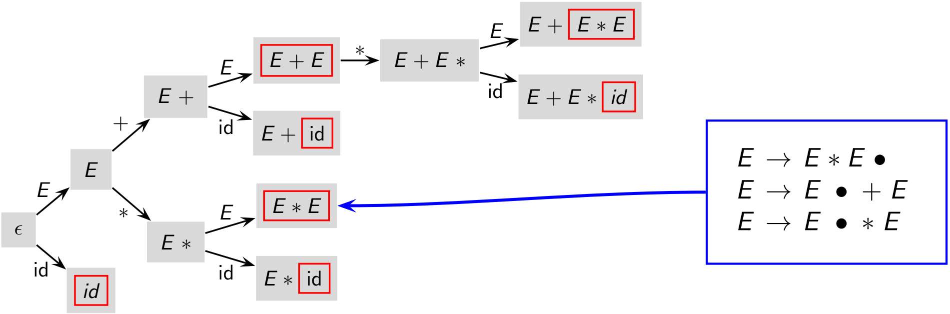
Step 4: Valid Items for Viable Prefixes

- An item is a grammar production with a dot ($\bullet$) in it somewhere in the RHS
- The dot separates what has been seen from what may be seen in the input
- We identify a set of items for a viable prefix to form a state of the parser



Step 4: Valid Items for Viable Prefixes

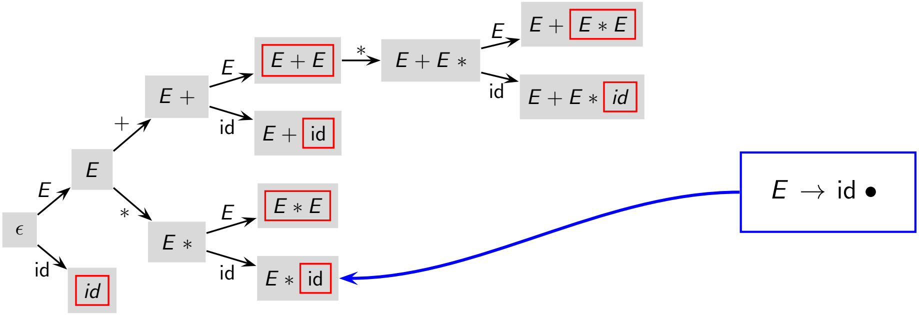
- An item is a grammar production with a dot (•) in it somewhere in the RHS
- The dot separates what has been seen from what may be seen in the input
- We identify a set of items for a viable prefix to form a state of the parser



$$E \rightarrow E * E \bullet$$
$$E \rightarrow E \bullet + E$$
$$E \rightarrow E \bullet * E$$

Step 4: Valid Items for Viable Prefixes

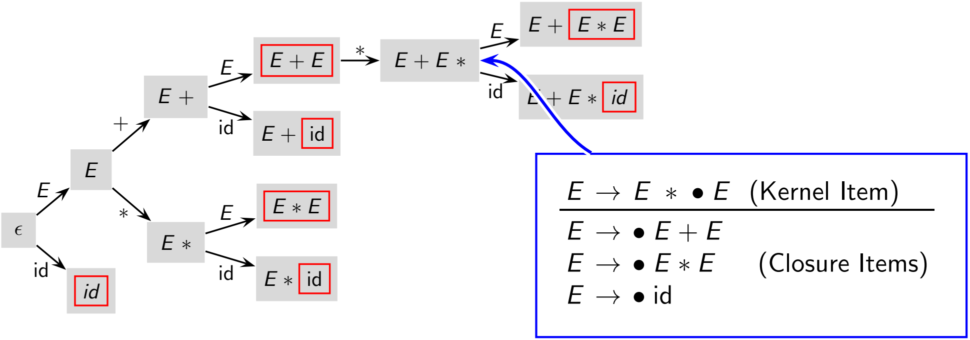
- An item is a grammar production with a dot (•) in it somewhere in the RHS
- The dot separates what has been seen from what may be seen in the input
- We identify a set of items for a viable prefix to form a state of the parser



$E \rightarrow \text{id} \bullet$

Step 4: Valid Items for Viable Prefixes

- An item is a grammar production with a dot (•) in it somewhere in the RHS
- The dot separates what has been seen from what may be seen in the input
- We identify a set of items for a viable prefix to form a state of the parser

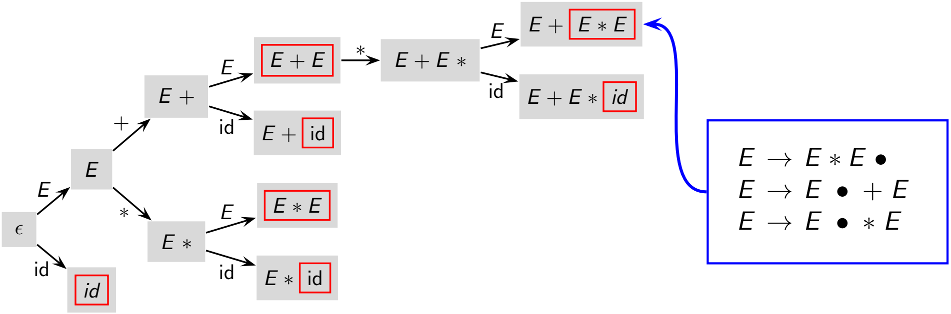


$E \rightarrow E * \bullet E$ (Kernel Item)

$E \rightarrow \bullet E + E$
$E \rightarrow \bullet E * E$ (Closure Items)
$E \rightarrow \bullet \text{id}$

Step 4: Valid Items for Viable Prefixes

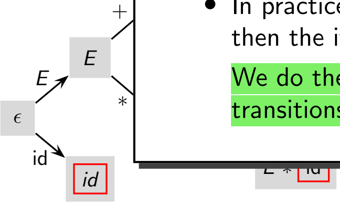
- An item is a grammar production with a dot (•) in it somewhere in the RHS
- The dot separates what has been seen from what may be seen in the input
- We identify a set of items for a viable prefix to form a state of the parser



$$E \rightarrow E * E \bullet$$
$$E \rightarrow E \bullet + E$$
$$E \rightarrow E \bullet * E$$

Step 4: Valid Items for Viable Prefixes

- An item is a grammar production with a dot (•) in it somewhere in the RHS
- The dot separates what has been seen from what may be seen in the input
- We iden



- An item set may not describe a viable prefix on its own (Prefixes of a viable prefix may be described by other item sets)
- Item sets for different viable prefixes may be same
- In practice, we do not construct the viable prefixes and then the item sets for them

  We do the opposite: we construct the item sets and the transitions between them give us the viable prefixes

Step 5: The Last Piece of Jigsaw Puzzle: Computing FOLLOW Sets

Consider $\beta A w \overset{rm}{\Rightarrow} \beta \alpha w$ and $A \rightarrow \alpha$

When do we reduce occurrence of α in $\gamma = \beta\alpha$ using $A \rightarrow \alpha$ using LR(k) items?
(i.e., when do we decide that α and $A \rightarrow \alpha$ form a handle in γ?)

Read the input from Left to right

Trace the Rightmost derivation in Reverse

The number of lookahead symbols in the items

Step 5: The Last Piece of Jigsaw Puzzle: Computing FOLLOW Sets

Consider $\beta A w \overset{rm}{\Rightarrow} \beta \alpha w$ and $A \rightarrow \alpha$
When do we reduce occurrence of α in $\gamma = \beta\alpha$ using $A \rightarrow \alpha$ using LR(k) items?
(i.e., when do we decide that α and $A \rightarrow \alpha$ form a handle in γ?)

Read the input from Left to right

Trace the Rightmost derivation in Reverse

The number of lookahead symbols in the items

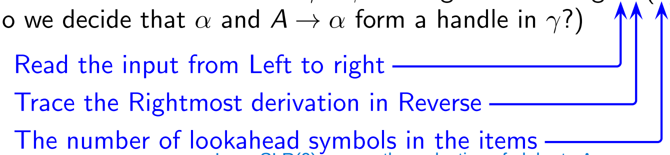
- As soon as we find α in γ

- When we find α in γ and the next input token can follow A in some sentential form

- When we find α in γ and the next input token follows A in γ

Step 5: The Last Piece of Jigsaw Puzzle: Computing FOLLOW Sets

Consider $\beta A w \overset{rm}{\Rightarrow} \beta \alpha w$ and $A \rightarrow \alpha$
When do we reduce occurrence of α in $\gamma = \beta\alpha$ using $A \rightarrow \alpha$ using LR(k) items?
(i.e., when do we decide that α and $A \rightarrow \alpha$ form a handle in γ?)



Read the input from Left to right

Trace the Rightmost derivation in Reverse

The number of lookahead symbols in the items

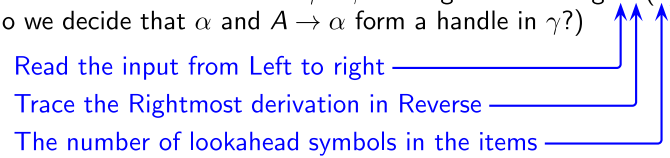
In an SLR(0) parser, the reduction of alpha to A occurs as soon as alpha is found in the input string gamma, without using lookahead, relying only on the current state and grammar rules; for example, in the grammar rule E -> id, if the input contains id + id, the parser reduces the first id to E immediately.

- As soon as we find α in γ
  LR(0) items and no lookahead in the input SLR(0) Parser
- When we find α in γ and the next input token can follow A in some sentential form
- When we find α in γ and the next input token follows A in γ

Step 5: The Last Piece of Jigsaw Puzzle: Computing FOLLOW Sets

Consider $\beta A w \overset{rm}{\Rightarrow} \beta \alpha w$ and $A \rightarrow \alpha$

When do we reduce occurrence of α in $\gamma = \beta\alpha$ using $A \rightarrow \alpha$ using LR(k) items?
(i.e., when do we decide that α and $A \rightarrow \alpha$ form a handle in γ?)



Read the input from Left to right

Trace the Rightmost derivation in Reverse

The number of lookahead symbols in the items

- As soon as we find α in γ

  LR(0) items and no lookahead in the input — SLR(0) Parser

- When we find α in γ and the next input token can follow A in some sentential form

  The SLR(1) parser extends the SLR(0) parser by using one lookahead symbol, reducing alpha to A if alpha is found in gamma and the next input token can follow A in some sentential form;

  LR(0) items and 1 lookahead in the input — SLR(1) Parser

  for example, in the grammar rule E -> id, if the input is id + id, the parser reduces the first id to E only if the next token (+) can follow E in expressions like E + E.

- When we find α in γ and the next input token follows A in γ

Step 5: The Last Piece of Jigsaw Puzzle: Computing FOLLOW Sets

Consider $\beta A w \overset{rm}{\Rightarrow} \beta \alpha w$ and $A \rightarrow \alpha$
When do we reduce occurrence of α in $\gamma = \beta\alpha$ using $A \rightarrow \alpha$ using LR(k) items?
(i.e., when do we decide that α and $A \rightarrow \alpha$ form a handle in γ?)

Read the input from Left to right

Trace the Rightmost derivation in Reverse

The number of lookahead symbols in the items

- As soon as we find α in γ

  LR(0) items and no lookahead in the input — SLR(0) Parser
- When we find α in γ and the next input token can follow A in some sentential form

  LR(0) items and 1 lookahead in the input — SLR(1) Parser
- When we find α in γ and the next input token follows A in γ

  LR(1) items and 1 lookahead in the input — CLR(1) Parser

The CLR(1) parser is more precise than the SLR(1) parser, using one lookahead symbol and considering the specific context in which A appears, reducing alpha to A if alpha is found in gamma and the next input token specifically follows A in the current context;
for example, in the grammar rule E -> id and input id ')', the CLR(1) parser reduces id to E only if) specifically follows E in the current context, ensuring context-sensitive reductions.

Step 5: FIRST and FOLLOW Sets

- FIRST(β) contains the terminals that may begin a string derivable from β

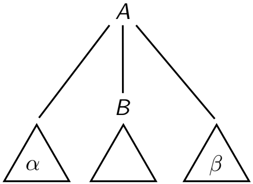
  If β derives ϵ, then $\epsilon \in$ FIRST(β)

  It is computed as the least fixed point solution of the following constraints

  For $A \rightarrow X_1X_2\ldots X_k$, FIRST$(A) \supseteq X_i, 1 \leq i \leq k,$ provided $\forall j < i, \epsilon \in$ FIRST(X_j)

- FOLLOW(A) contains the terminals that follow A in some sentential form

  It is computed as the least fixed point solution of the following constraints



For production $A \rightarrow \alpha B \beta$

- If A is the start non-terminal

  FOLLOW$(A) \supseteq \{\$\}$
- FOLLOW$(B) \supseteq$ FIRST$(\beta) - \{\epsilon\}$
- If β is ϵ or $\epsilon \in$ FIRST(β)

  FOLLOW$(B) \supseteq$ FOLLOW(A)

Step 5: FIRST and FOLLOW Sets

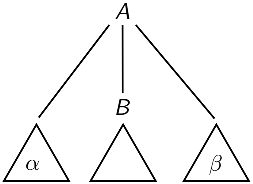
- FIRST(β) contains the terminals that may begin a string derivable from β

  If β derives ϵ, then $\epsilon \in$ FIRST(β)

  It is computed as the least fixed point solution of the following constraints

  For $A \rightarrow X_1 X_2 \ldots X_k$, FIRST$(A) \supseteq X_i, 1 \leq i \leq k,$ provided $\forall j < i, \epsilon \in$ FIRST(X_j)
- FOLLOW(A) contains the terminals that follow A in some sentential form

  It is computed as the least fixed point solution of the following constraints



For production $A \rightarrow \alpha B \beta$

- If A is the start non-terminal

  FOLLOW$(A) \supseteq \{\$\}$
- FOLLOW$(B) \supseteq$ FIRST$(\beta) - \{\epsilon\}$
- If β is ϵ or $\epsilon \in$ FIRST(β)

  FOLLOW$(B) \supseteq$ FOLLOW(A)

For our grammar

$E \rightarrow E + E$
$E \rightarrow E * E$
$E \rightarrow$ id

FIRST$(E) = \{$id$\}$
FOLLOW$(E) = \{\$, +, *\}$

Step 6: Computing LR(0) Item Sets for Expressions Grammar

An item does not contain any lookahead symbol

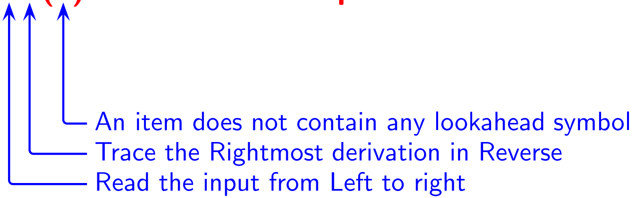
Trace the Rightmost derivation in Reverse

Read the input from Left to right

Step 6: LR(0) Items Sets

parsing-slides-sanyal-part3.pdf

Step 6: Computing LR(0) Item Sets for Expressions Grammar



An item does not contain any lookahead symbol

Trace the Rightmost derivation in Reverse

Read the input from Left to right

Step 6: Computing LR(0) Item Sets for Expressions Grammar

0 $E' \rightarrow E$
1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

- Augment the grammar by adding a synthetic start symbol
- Construct the start state by putting a dot at the start of the start symbol and taking a closure (add every rule for every non-terminal that has a dot before it in some rule)
- Identify transitions on every symbol that has a dot before it to construct new states
- For every state so identified, take a closure and identify transitions on every symbol that has a dot before it

Step 6: Computing LR(0) Item Sets for Expressions Grammar

$0\ E' \rightarrow E$
$1\ E \rightarrow E + E$
$2\ E \rightarrow E * E$
$3\ E \rightarrow \text{id}$

I_0

Kernel items

$E' \rightarrow \bullet\, E$

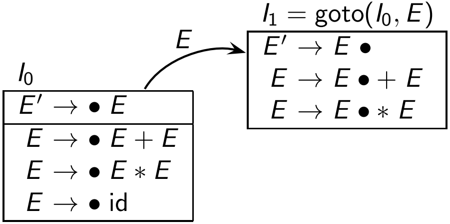
$E \rightarrow \bullet\, E + E$
$E \rightarrow \bullet\, E * E$
$E \rightarrow \bullet\, \text{id}$

Closure items

- Augment the grammar by adding a synthetic start symbol
- Construct the start state by putting a dot at the start of the start symbol and taking a closure (add every rule for every non-terminal that has a dot before it in some rule)
- Identify transitions on every symbol that has a dot before it to construct new states
- For every state so identified, take a closure and identify transitions on every symbol that has a dot before it

Step 6: Computing LR(0) Item Sets for Expressions Grammar

$0\ E' \rightarrow E$
$1\ E \rightarrow E + E$
$2\ E \rightarrow E * E$
$3\ E \rightarrow \text{id}$



I_0

$E' \rightarrow \bullet\, E$
$E \rightarrow \bullet\, E + E$
$E \rightarrow \bullet\, E * E$
$E \rightarrow \bullet\, \text{id}$

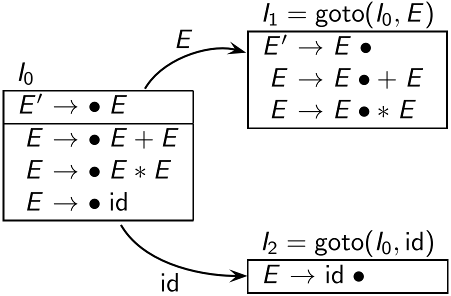
E

$I_1 = \text{goto}(I_0, E)$

$E' \rightarrow E\, \bullet$
$E \rightarrow E\, \bullet + E$
$E \rightarrow E\, \bullet * E$

Step 6: Computing LR(0) Item Sets for Expressions Grammar

$0\ E' \rightarrow E$
$1\ E \rightarrow E + E$
$2\ E \rightarrow E * E$
$3\ E \rightarrow \text{id}$



I_0
$E' \rightarrow \bullet\ E$
$E \rightarrow \bullet\ E + E$
$E \rightarrow \bullet\ E * E$
$E \rightarrow \bullet\ \text{id}$

E

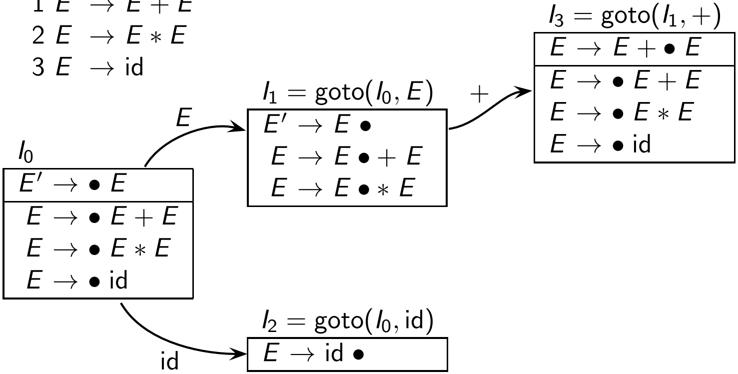
$I_1 = \text{goto}(I_0, E)$
$E' \rightarrow E\ \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

id

$I_2 = \text{goto}(I_0, \text{id})$
$E \rightarrow \text{id}\ \bullet$

Step 6: Computing LR(0) Item Sets for Expressions Grammar

$$\begin{array}{lll} 0 & E' & \rightarrow E \\ 1 & E & \rightarrow E + E \\ 2 & E & \rightarrow E * E \\ 3 & E & \rightarrow \text{id} \end{array}$$



I_0

$E' \rightarrow \bullet E$

$E \rightarrow \bullet E + E$

$E \rightarrow \bullet E * E$

$E \rightarrow \bullet \text{id}$

$I_1 = \text{goto}(I_0, E)$

$E' \rightarrow E \bullet$

$E \rightarrow E \bullet + E$

$E \rightarrow E \bullet * E$

$I_2 = \text{goto}(I_0, \text{id})$

$E \rightarrow \text{id} \bullet$

$I_3 = \text{goto}(I_1, +)$

$E \rightarrow E + \bullet E$

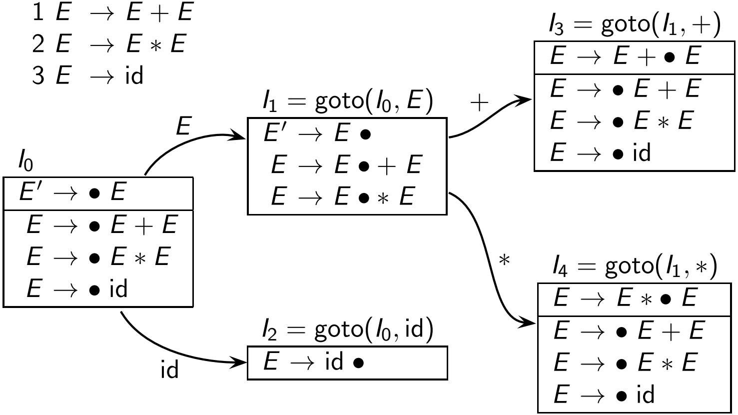
$E \rightarrow \bullet E + E$

$E \rightarrow \bullet E * E$

$E \rightarrow \bullet \text{id}$

Step 6: Computing LR(0) Item Sets for Expressions Grammar

0 $E' \rightarrow E$
1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$



I_0
$E' \rightarrow \bullet\, E$
$E \rightarrow \bullet\, E + E$
$E \rightarrow \bullet\, E * E$
$E \rightarrow \bullet\, \text{id}$

$I_1 = \text{goto}(I_0, E)$
$E' \rightarrow E\, \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

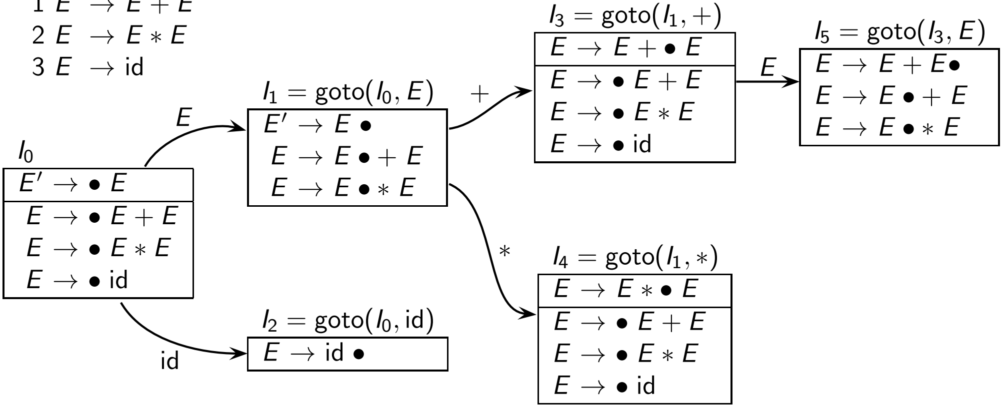
$I_2 = \text{goto}(I_0, \text{id})$
$E \rightarrow \text{id}\, \bullet$

$I_3 = \text{goto}(I_1, +)$
$E \rightarrow E + \bullet\, E$
$E \rightarrow \bullet\, E + E$
$E \rightarrow \bullet\, E * E$
$E \rightarrow \bullet\, \text{id}$

$I_4 = \text{goto}(I_1, *)$
$E \rightarrow E * \bullet\, E$
$E \rightarrow \bullet\, E + E$
$E \rightarrow \bullet\, E * E$
$E \rightarrow \bullet\, \text{id}$

Step 6: Computing LR(0) Item Sets for Expressions Grammar

0 $E' \to E$
1 $E \to E + E$
2 $E \to E * E$
3 $E \to \text{id}$



I_0
$E' \to \bullet E$
$E \to \bullet E + E$
$E \to \bullet E * E$
$E \to \bullet \text{id}$

$I_1 = \text{goto}(I_0, E)$
$E' \to E \bullet$
$E \to E \bullet + E$
$E \to E \bullet * E$

$I_2 = \text{goto}(I_0, \text{id})$
$E \to \text{id} \bullet$

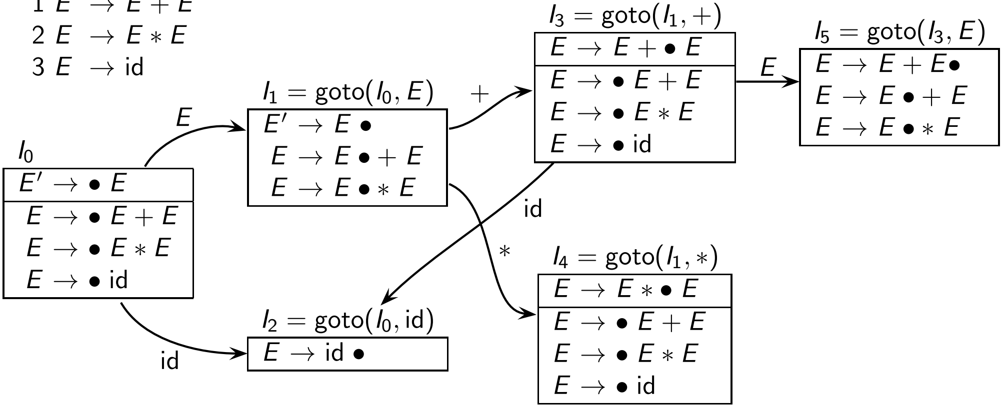
$I_3 = \text{goto}(I_1, +)$
$E \to E + \bullet E$
$E \to \bullet E + E$
$E \to \bullet E * E$
$E \to \bullet \text{id}$

$I_4 = \text{goto}(I_1, *)$
$E \to E * \bullet E$
$E \to \bullet E + E$
$E \to \bullet E * E$
$E \to \bullet \text{id}$

$I_5 = \text{goto}(I_3, E)$
$E \to E + E \bullet$
$E \to E \bullet + E$
$E \to E \bullet * E$

Step 6: Computing LR(0) Item Sets for Expressions Grammar

0 $E' \to E$
1 $E \to E + E$
2 $E \to E * E$
3 $E \to \text{id}$



I_0
$E' \to \bullet E$
$E \to \bullet E + E$
$E \to \bullet E * E$
$E \to \bullet \text{id}$

$I_1 = \text{goto}(I_0, E)$
$E' \to E \bullet$
$E \to E \bullet + E$
$E \to E \bullet * E$

$I_2 = \text{goto}(I_0, \text{id})$
$E \to \text{id} \bullet$

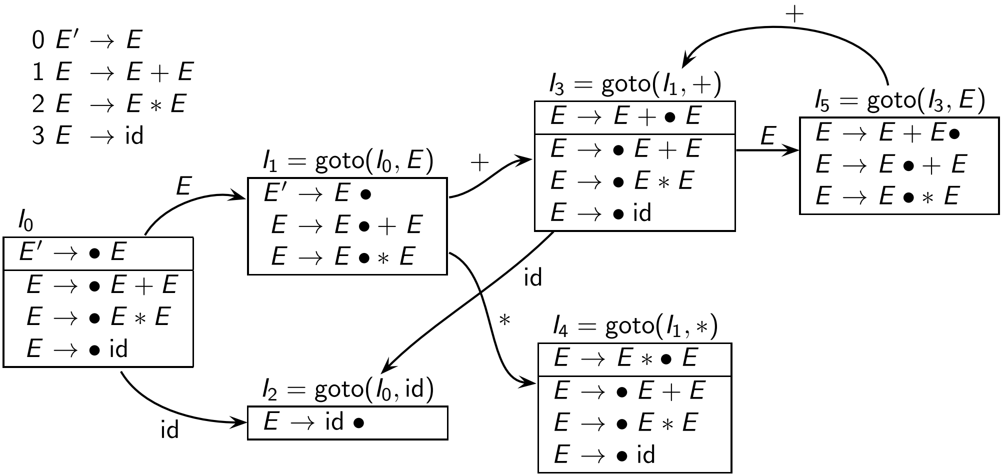
$I_3 = \text{goto}(I_1, +)$
$E \to E + \bullet E$
$E \to \bullet E + E$
$E \to \bullet E * E$
$E \to \bullet \text{id}$

$I_4 = \text{goto}(I_1, *)$
$E \to E * \bullet E$
$E \to \bullet E + E$
$E \to \bullet E * E$
$E \to \bullet \text{id}$

$I_5 = \text{goto}(I_3, E)$
$E \to E + E \bullet$
$E \to E \bullet + E$
$E \to E \bullet * E$

Step 6: Computing LR(0) Item Sets for Expressions Grammar

$0\ E' \rightarrow E$
$1\ E \rightarrow E + E$
$2\ E \rightarrow E * E$
$3\ E \rightarrow \text{id}$



I_0
$E' \rightarrow \bullet E$
$E \rightarrow \bullet E + E$
$E \rightarrow \bullet E * E$
$E \rightarrow \bullet \text{id}$

$I_1 = \text{goto}(I_0, E)$
$E' \rightarrow E \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

$I_2 = \text{goto}(I_0, \text{id})$
$E \rightarrow \text{id} \bullet$

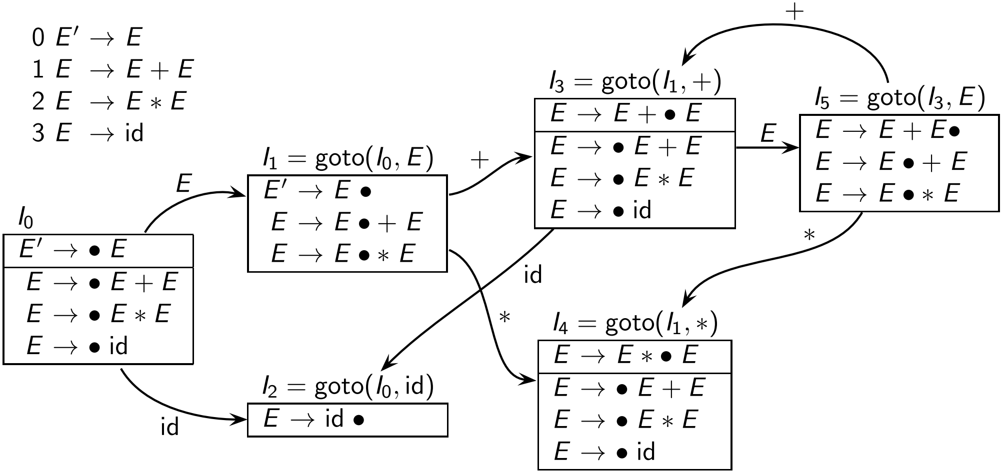
$I_3 = \text{goto}(I_1, +)$
$E \rightarrow E + \bullet E$
$E \rightarrow \bullet E + E$
$E \rightarrow \bullet E * E$
$E \rightarrow \bullet \text{id}$

$I_4 = \text{goto}(I_1, *)$
$E \rightarrow E * \bullet E$
$E \rightarrow \bullet E + E$
$E \rightarrow \bullet E * E$
$E \rightarrow \bullet \text{id}$

$I_5 = \text{goto}(I_3, E)$
$E \rightarrow E + E \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

Step 6: Computing LR(0) Item Sets for Expressions Grammar

0 $E' \rightarrow E$
1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$



I_0
$E' \rightarrow \bullet\, E$
$E \rightarrow \bullet\, E + E$
$E \rightarrow \bullet\, E * E$
$E \rightarrow \bullet\, \text{id}$

$I_1 = \text{goto}(I_0, E)$
$E' \rightarrow E\, \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

$I_2 = \text{goto}(I_0, \text{id})$
$E \rightarrow \text{id}\, \bullet$

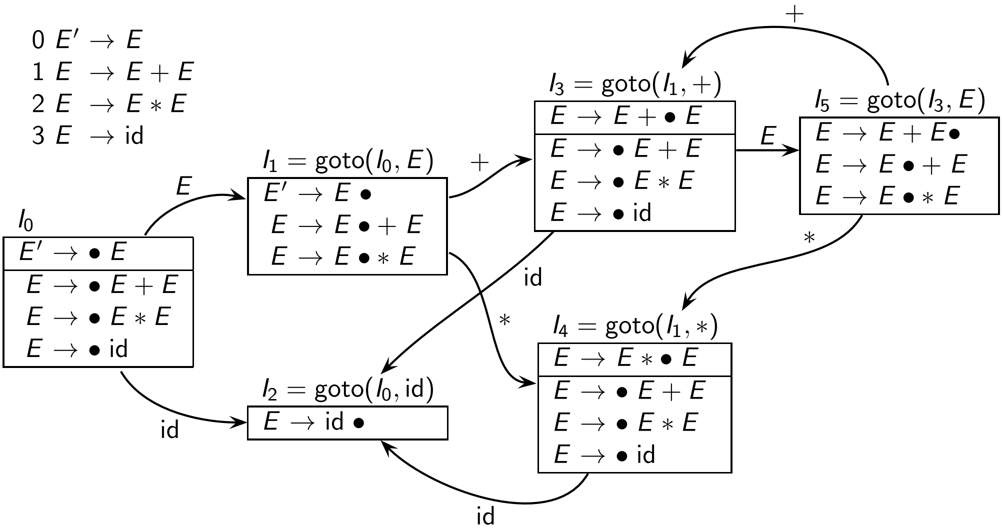
$I_3 = \text{goto}(I_1, +)$
$E \rightarrow E + \bullet\, E$
$E \rightarrow \bullet\, E + E$
$E \rightarrow \bullet\, E * E$
$E \rightarrow \bullet\, \text{id}$

$I_4 = \text{goto}(I_1, *)$
$E \rightarrow E * \bullet\, E$
$E \rightarrow \bullet\, E + E$
$E \rightarrow \bullet\, E * E$
$E \rightarrow \bullet\, \text{id}$

$I_5 = \text{goto}(I_3, E)$
$E \rightarrow E + E \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

Step 6: Computing LR(0) Item Sets for Expressions Grammar

$0 \quad E' \rightarrow E$
$1 \quad E \rightarrow E + E$
$2 \quad E \rightarrow E * E$
$3 \quad E \rightarrow \text{id}$



I_0
$E' \rightarrow \bullet E$
$E \rightarrow \bullet E + E$
$E \rightarrow \bullet E * E$
$E \rightarrow \bullet \text{id}$

$I_1 = \text{goto}(I_0, E)$
$E' \rightarrow E \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

$I_2 = \text{goto}(I_0, \text{id})$
$E \rightarrow \text{id} \bullet$

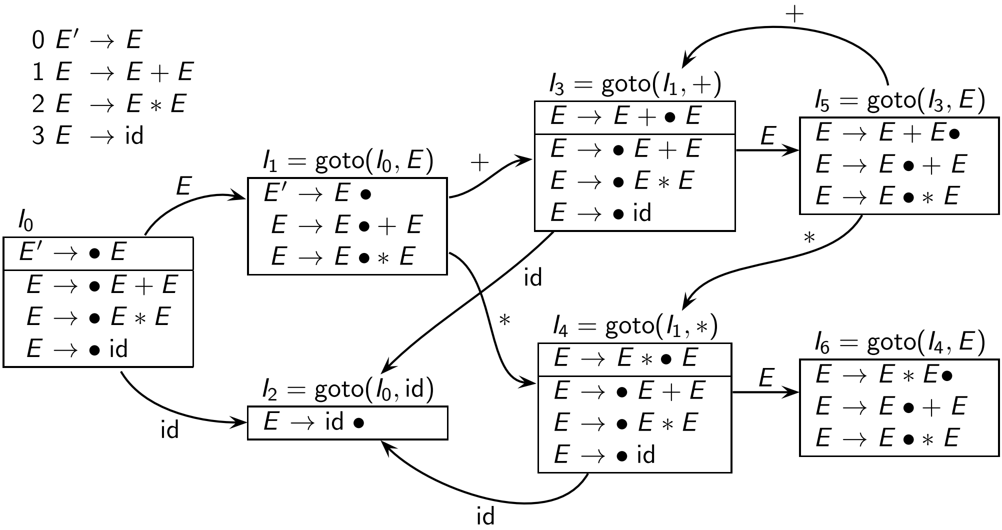
$I_3 = \text{goto}(I_1, +)$
$E \rightarrow E + \bullet E$
$E \rightarrow \bullet E + E$
$E \rightarrow \bullet E * E$
$E \rightarrow \bullet \text{id}$

$I_4 = \text{goto}(I_1, *)$
$E \rightarrow E * \bullet E$
$E \rightarrow \bullet E + E$
$E \rightarrow \bullet E * E$
$E \rightarrow \bullet \text{id}$

$I_5 = \text{goto}(I_3, E)$
$E \rightarrow E + E \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

Step 6: Computing LR(0) Item Sets for Expressions Grammar

0 $E' \rightarrow E$
1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$



I_0
$E' \rightarrow \bullet\, E$
$E \rightarrow \bullet\, E + E$
$E \rightarrow \bullet\, E * E$
$E \rightarrow \bullet\, \text{id}$

$I_1 = \text{goto}(I_0, E)$
$E' \rightarrow E\, \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

$I_2 = \text{goto}(I_0, \text{id})$
$E \rightarrow \text{id}\, \bullet$

$I_3 = \text{goto}(I_1, +)$
$E \rightarrow E + \bullet\, E$
$E \rightarrow \bullet\, E + E$
$E \rightarrow \bullet\, E * E$
$E \rightarrow \bullet\, \text{id}$

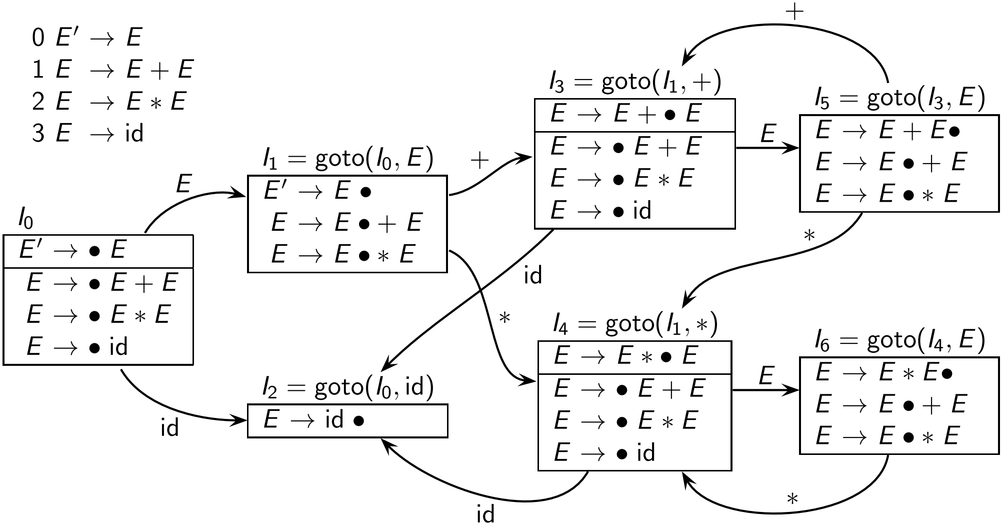
$I_4 = \text{goto}(I_1, *)$
$E \rightarrow E * \bullet\, E$
$E \rightarrow \bullet\, E + E$
$E \rightarrow \bullet\, E * E$
$E \rightarrow \bullet\, \text{id}$

$I_5 = \text{goto}(I_3, E)$
$E \rightarrow E + E \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

$I_6 = \text{goto}(I_4, E)$
$E \rightarrow E * E \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

Step 6: Computing LR(0) Item Sets for Expressions Grammar

$0\ E' \rightarrow E$
$1\ E \rightarrow E + E$
$2\ E \rightarrow E * E$
$3\ E \rightarrow \text{id}$



I_0
$E' \rightarrow \bullet E$
$E \rightarrow \bullet E + E$
$E \rightarrow \bullet E * E$
$E \rightarrow \bullet\, \text{id}$

$I_1 = \text{goto}(I_0, E)$
$E' \rightarrow E \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

$I_2 = \text{goto}(I_0, \text{id})$
$E \rightarrow \text{id} \bullet$

$I_3 = \text{goto}(I_1, +)$
$E \rightarrow E + \bullet E$
$E \rightarrow \bullet E + E$
$E \rightarrow \bullet E * E$
$E \rightarrow \bullet\, \text{id}$

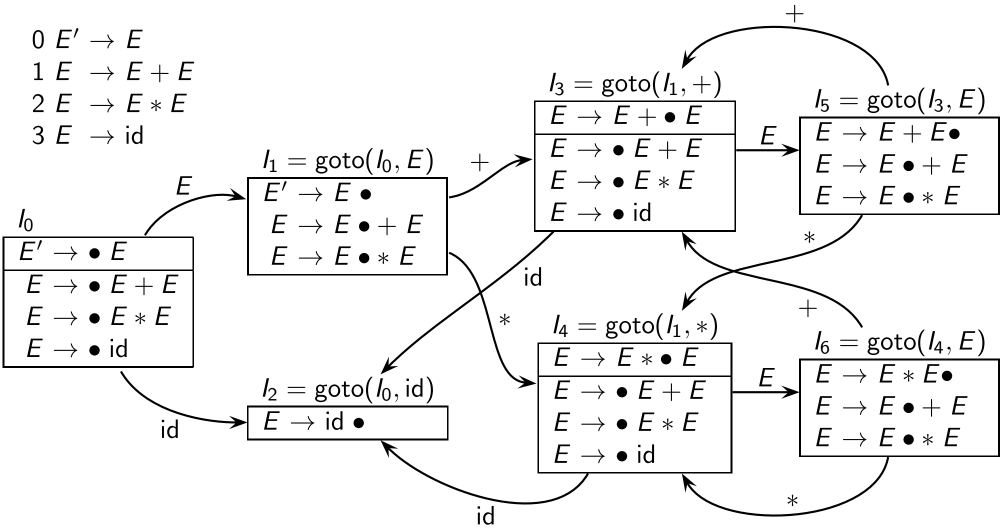
$I_4 = \text{goto}(I_1, *)$
$E \rightarrow E * \bullet E$
$E \rightarrow \bullet E + E$
$E \rightarrow \bullet E * E$
$E \rightarrow \bullet\, \text{id}$

$I_5 = \text{goto}(I_3, E)$
$E \rightarrow E + E \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

$I_6 = \text{goto}(I_4, E)$
$E \rightarrow E * E \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

Step 6: Computing LR(0) Item Sets for Expressions Grammar

0 $E' \to E$
1 $E \to E + E$
2 $E \to E * E$
3 $E \to \text{id}$



I_0
$E' \to \bullet\, E$
$E \to \bullet\, E + E$
$E \to \bullet\, E * E$
$E \to \bullet\, \text{id}$

$I_1 = \text{goto}(I_0, E)$
$E' \to E\, \bullet$
$E \to E \bullet + E$
$E \to E \bullet * E$

$I_2 = \text{goto}(I_0, \text{id})$
$E \to \text{id}\, \bullet$

$I_3 = \text{goto}(I_1, +)$
$E \to E + \bullet\, E$
$E \to \bullet\, E + E$
$E \to \bullet\, E * E$
$E \to \bullet\, \text{id}$

$I_4 = \text{goto}(I_1, *)$
$E \to E * \bullet\, E$
$E \to \bullet\, E + E$
$E \to \bullet\, E * E$
$E \to \bullet\, \text{id}$

$I_5 = \text{goto}(I_3, E)$
$E \to E + E \bullet$
$E \to E \bullet + E$
$E \to E \bullet * E$

$I_6 = \text{goto}(I_4, E)$
$E \to E * E \bullet$
$E \to E \bullet + E$
$E \to E \bullet * E$

IIT Bombay
cs302: Implementation of Programming Languages

Topic:

Syntax Analysis

Section:

Grammars, Derivations, and Parse Trees

Shift Reduce Parsing

SLR(1) Parsing

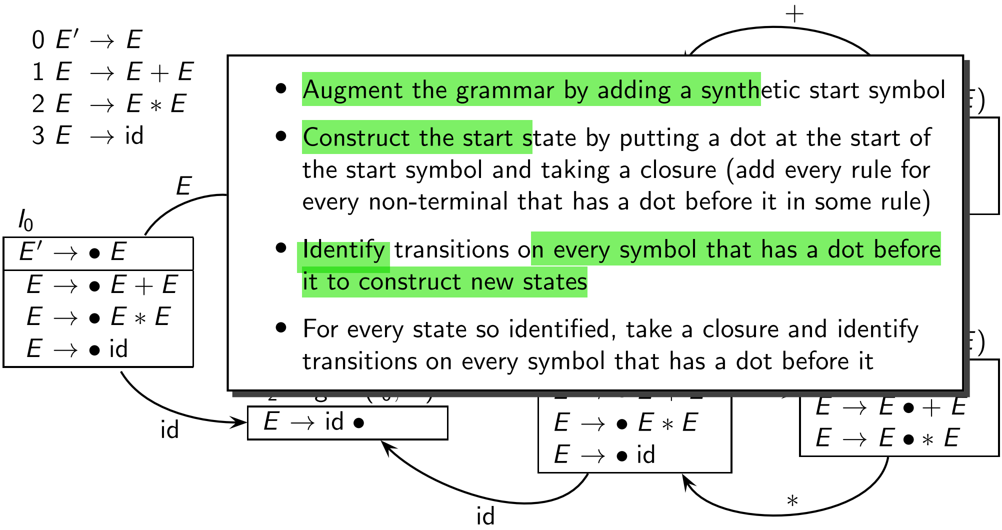
Conceptual Issues in Parsing

CLR(1) Parsing

LALR(1) Parsing

Step 6: Computing LR(0) Item Sets for Expressions Grammar

0 $E' \rightarrow E$
1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

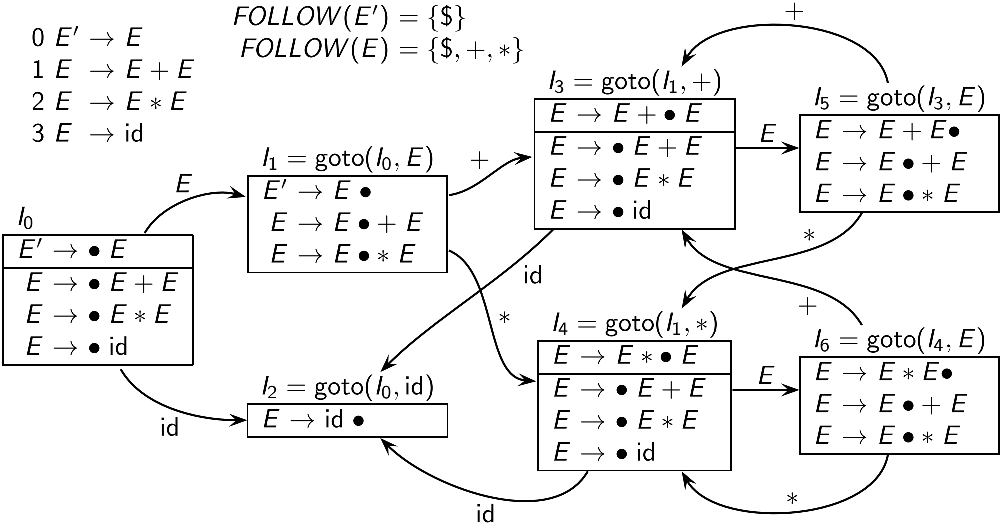


- Augment the grammar by adding a synthetic start symbol
- Construct the start state by putting a dot at the start of the start symbol and taking a closure (add every rule for every non-terminal that has a dot before it in some rule)
- Identify transitions on every symbol that has a dot before it to construct new states
- For every state so identified, take a closure and identify transitions on every symbol that has a dot before it

Step 6: Computing LR(0) Item Sets for Expressions Grammar

0 $E' \rightarrow E$
1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

$FOLLOW(E') = \{\$\}$
$FOLLOW(E) = \{\$, +, *\}$



I_0
$E' \rightarrow \bullet\, E$
$E \rightarrow \bullet\, E + E$
$E \rightarrow \bullet\, E * E$
$E \rightarrow \bullet\, \text{id}$

$I_1 = \text{goto}(I_0, E)$
$E' \rightarrow E \,\bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

$I_2 = \text{goto}(I_0, \text{id})$
$E \rightarrow \text{id} \,\bullet$

$I_3 = \text{goto}(I_1, +)$
$E \rightarrow E + \bullet\, E$
$E \rightarrow \bullet\, E + E$
$E \rightarrow \bullet\, E * E$
$E \rightarrow \bullet\, \text{id}$

$I_4 = \text{goto}(I_1, *)$
$E \rightarrow E * \bullet\, E$
$E \rightarrow \bullet\, E + E$
$E \rightarrow \bullet\, E * E$
$E \rightarrow \bullet\, \text{id}$

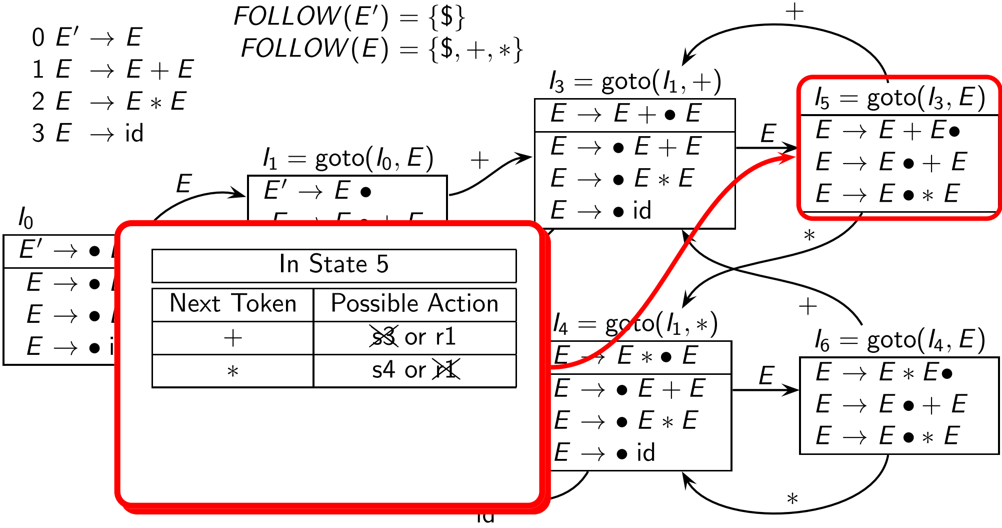
$I_5 = \text{goto}(I_3, E)$
$E \rightarrow E + E \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

$I_6 = \text{goto}(I_4, E)$
$E \rightarrow E * E \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

Step 6: Computing LR(0) Item Sets for Expressions Grammar

0 $E' \to E$
1 $E \to E + E$
2 $E \to E * E$
3 $E \to \text{id}$

$FOLLOW(E') = \{\$\}$
$FOLLOW(E) = \{\$, +, *\}$



I_0
$E' \to \bullet$
$E \to \bullet$
$E \to \bullet$
$E \to \bullet$ i

$I_1 = \text{goto}(I_0, E)$
$E' \to E \bullet$

$I_3 = \text{goto}(I_1, +)$
$E \to E + \bullet E$
$E \to \bullet E + E$
$E \to \bullet E * E$
$E \to \bullet \text{id}$

$I_5 = \text{goto}(I_3, E)$
$E \to E + E \bullet$
$E \to E \bullet + E$
$E \to E \bullet * E$

$I_4 = \text{goto}(I_1, *)$
$E \to E * \bullet E$
$E \to \bullet E + E$
$E \to \bullet E * E$
$E \to \bullet \text{id}$

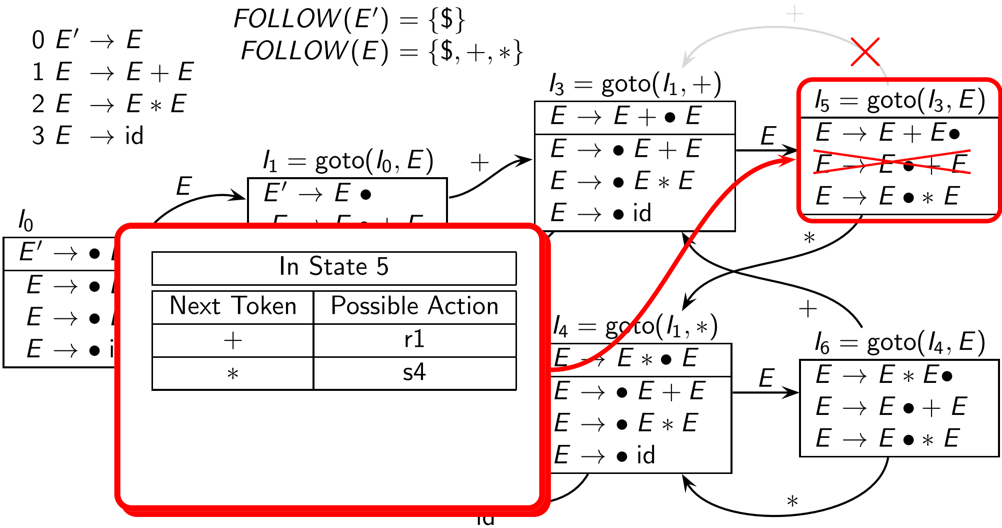
$I_6 = \text{goto}(I_4, E)$
$E \to E * E \bullet$
$E \to E \bullet + E$
$E \to E \bullet * E$

In State 5	
Next Token	Possible Action
$+$	~~s3~~ or r1
$*$	s4 or ~~r1~~

Step 6: Computing LR(0) Item Sets for Expressions Grammar

0 $E' \rightarrow E$
1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

$FOLLOW(E') = \{\$\}$
$FOLLOW(E) = \{\$, +, *\}$



In State 5	
Next Token	Possible Action
+	r1
*	s4

Step 6: Computing LR(0) Item Sets for Expressions Grammar

$FOLLOW(E') = \{\$\}$

$FOLLOW(E) = \{\$, +, *\}$

0 $E' \to E$

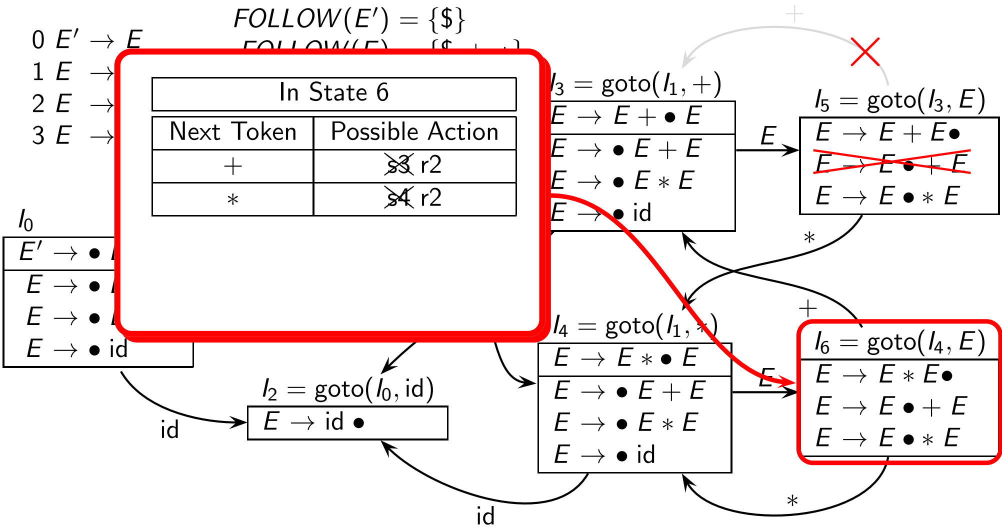
1 $E \to$

2 $E \to$

3 $E \to$

In State 6

Next Token	Possible Action
$+$	~~s3~~ r2
$*$	~~s4~~ r2

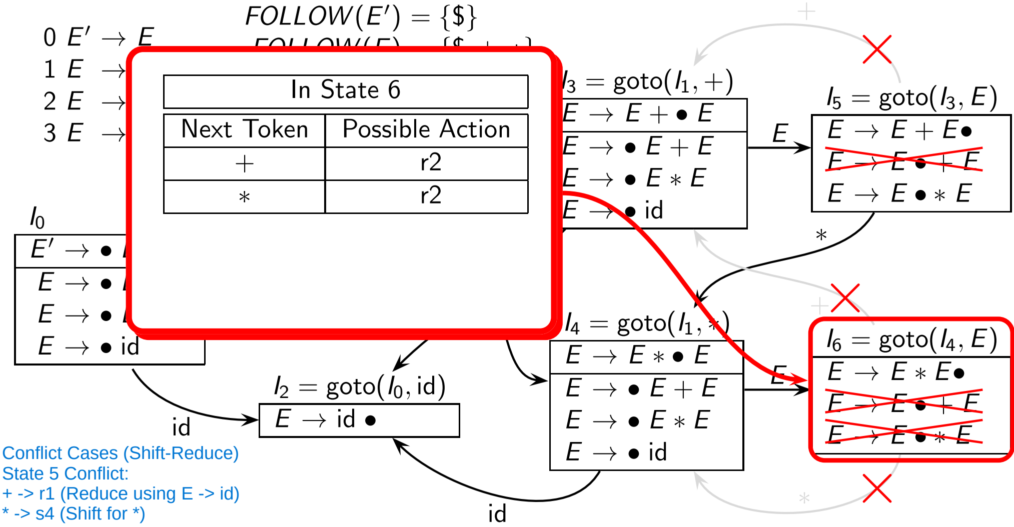


Step 6: Computing LR(0) Item Sets for Expressions Grammar

$FOLLOW(E') = \{\$\}$

0 $E' \rightarrow E$
1 $E \rightarrow$
2 $E \rightarrow$
3 $E \rightarrow$

In State 6	
Next Token	Possible Action
$+$	r2
$*$	r2

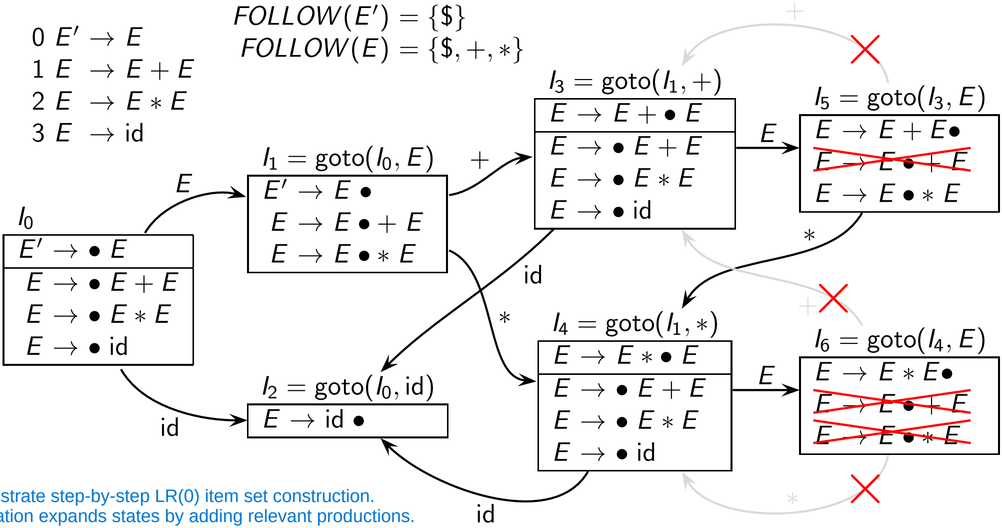


Conflict Cases (Shift-Reduce)
State 5 Conflict:
+ -> r1 (Reduce using E -> id)
* -> s4 (Shift for *)
State 6 Conflict:
Both + and * result in r2 (Reduce using E -> E * E)

Step 6: Computing LR(0) Item Sets for Expressions Grammar

$0 \; E' \rightarrow E$
$1 \; E \rightarrow E + E$
$2 \; E \rightarrow E * E$
$3 \; E \rightarrow \text{id}$

$FOLLOW(E') = \{\$\}$
$FOLLOW(E) = \{\$, +, *\}$



I_0
$E' \rightarrow \bullet\, E$
$E \rightarrow \bullet\, E + E$
$E \rightarrow \bullet\, E * E$
$E \rightarrow \bullet\, \text{id}$

$I_1 = \text{goto}(I_0, E)$
$E' \rightarrow E \,\bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

$I_2 = \text{goto}(I_0, \text{id})$
$E \rightarrow \text{id} \,\bullet$

$I_3 = \text{goto}(I_1, +)$
$E \rightarrow E + \bullet\, E$
$E \rightarrow \bullet\, E + E$
$E \rightarrow \bullet\, E * E$
$E \rightarrow \bullet\, \text{id}$

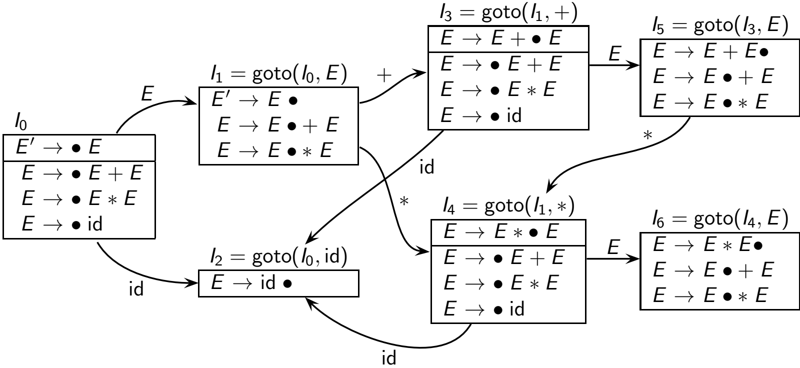
$I_4 = \text{goto}(I_1, *)$
$E \rightarrow E * \bullet\, E$
$E \rightarrow \bullet\, E + E$
$E \rightarrow \bullet\, E * E$
$E \rightarrow \bullet\, \text{id}$

$I_5 = \text{goto}(I_3, E)$
$E \rightarrow E + E \bullet$
$\cancel{E \rightarrow E \bullet + E}$
$E \rightarrow E \bullet * E$

$I_6 = \text{goto}(I_4, E)$
$E \rightarrow E * E \bullet$
$\cancel{E \rightarrow E \bullet + E}$
$\cancel{E \rightarrow E \bullet * E}$

The slides illustrate step-by-step LR(0) item set construction.
Closure operation expands states by adding relevant productions.
Goto transitions help form the LR(0) parsing table.
Shift-Reduce conflicts appear in states 5 and 6, requiring precedence resolution.

The DFA of Item Sets Accepts Viable Prefixes



I_0

$E' \rightarrow \bullet\ E$
$E \rightarrow \bullet\ E + E$
$E \rightarrow \bullet\ E * E$
$E \rightarrow \bullet\ \text{id}$

$I_1 = \text{goto}(I_0, E)$

$E' \rightarrow E\ \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

$I_2 = \text{goto}(I_0, \text{id})$

$E \rightarrow \text{id}\ \bullet$

$I_3 = \text{goto}(I_1, +)$

$E \rightarrow E + \bullet\ E$
$E \rightarrow \bullet\ E + E$
$E \rightarrow \bullet\ E * E$
$E \rightarrow \bullet\ \text{id}$

$I_4 = \text{goto}(I_1, *)$

$E \rightarrow E * \bullet\ E$
$E \rightarrow \bullet\ E + E$
$E \rightarrow \bullet\ E * E$
$E \rightarrow \bullet\ \text{id}$

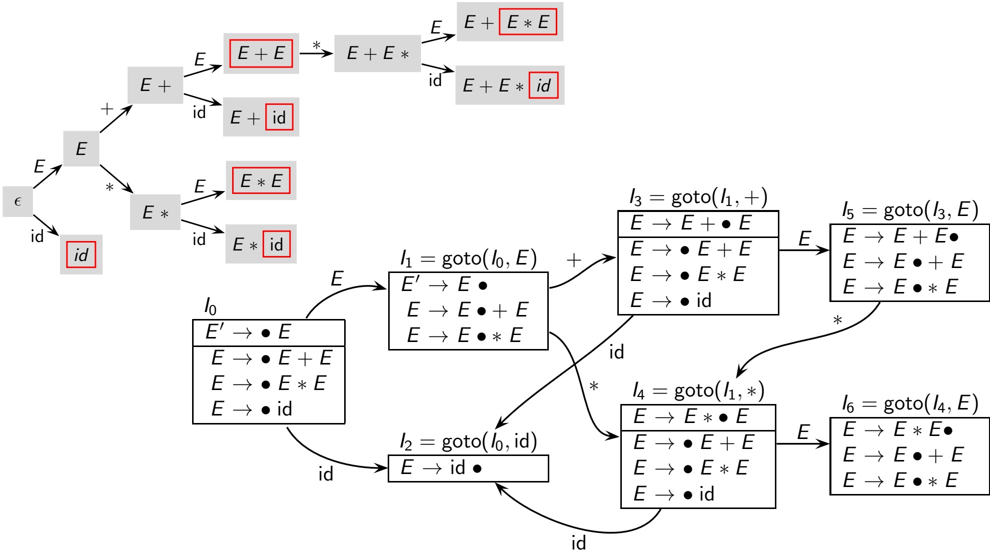
$I_5 = \text{goto}(I_3, E)$

$E \rightarrow E + E\bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

$I_6 = \text{goto}(I_4, E)$

$E \rightarrow E * E\bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

The DFA of Item Sets Accepts Viable Prefixes



ϵ $\xrightarrow{E}$ E

ϵ $\xrightarrow{\text{id}}$ $\boxed{id}$

E $\xrightarrow{+}$ $E+$

E $\xrightarrow{*}$ $E*$

$E+$ $\xrightarrow{E}$ $\boxed{E+E}$

$E+$ $\xrightarrow{\text{id}}$ $E+\boxed{\text{id}}$

$\boxed{E+E}$ $\xrightarrow{*}$ $E+E*$

$E+E*$ $\xrightarrow{E}$ $E+\boxed{E*E}$

$E+E*$ $\xrightarrow{\text{id}}$ $E+E*\boxed{id}$

$E*$ $\xrightarrow{E}$ $\boxed{E*E}$

$E*$ $\xrightarrow{\text{id}}$ $E*\boxed{\text{id}}$

I_0
$E' \rightarrow \bullet E$
$E \rightarrow \bullet E+E$
$E \rightarrow \bullet E*E$
$E \rightarrow \bullet \text{id}$

$I_1 = \text{goto}(I_0, E)$
$E' \rightarrow E\bullet$
$E \rightarrow E\bullet + E$
$E \rightarrow E\bullet * E$

$I_2 = \text{goto}(I_0, \text{id})$
$E \rightarrow \text{id}\,\bullet$

$I_3 = \text{goto}(I_1, +)$
$E \rightarrow E+\bullet E$
$E \rightarrow \bullet E+E$
$E \rightarrow \bullet E*E$
$E \rightarrow \bullet \text{id}$

$I_4 = \text{goto}(I_1, *)$
$E \rightarrow E*\bullet E$
$E \rightarrow \bullet E+E$
$E \rightarrow \bullet E*E$
$E \rightarrow \bullet \text{id}$

$I_5 = \text{goto}(I_3, E)$
$E \rightarrow E+E\bullet$
$E \rightarrow E\bullet + E$
$E \rightarrow E\bullet * E$

$I_6 = \text{goto}(I_4, E)$
$E \rightarrow E*E\bullet$
$E \rightarrow E\bullet + E$
$E \rightarrow E\bullet * E$

The DFA ensures that no invalid prefix enters the stack, helping the parser process inputs correctly.

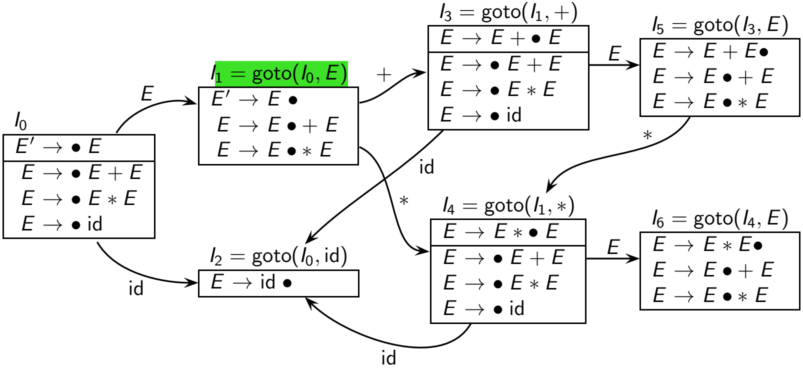
Putting it All Together: Constructing the Parsing Table

empty program is ACCEPTED

	id	$+$	$*$	\$	E
0	$s2$				$c1$
1		$s3$	$s4$	acc	
2		$r3$	$r3$	$r3$	
3	$s2$				$c5$
4	$s2$				$c6$
5		$r1$	$s4$	$r1$	
6		$r2$	$r2$	$r2$	

0 $E' \rightarrow E$
1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

$FOLLOW(E') = \{\$\}$
$FOLLOW(E) = \{\$, +, *\}$



I_0
$E' \rightarrow \bullet E$
$E \rightarrow \bullet E + E$
$E \rightarrow \bullet E * E$
$E \rightarrow \bullet \text{id}$

$I_1 = \text{goto}(I_0, E)$
$E' \rightarrow E \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

$I_2 = \text{goto}(I_0, \text{id})$
$E \rightarrow \text{id} \bullet$

$I_3 = \text{goto}(I_1, +)$
$E \rightarrow E + \bullet E$
$E \rightarrow \bullet E + E$
$E \rightarrow \bullet E * E$
$E \rightarrow \bullet \text{id}$

$I_4 = \text{goto}(I_1, *)$
$E \rightarrow E * \bullet E$
$E \rightarrow \bullet E + E$
$E \rightarrow \bullet E * E$
$E \rightarrow \bullet \text{id}$

$I_5 = \text{goto}(I_3, E)$
$E \rightarrow E + E \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

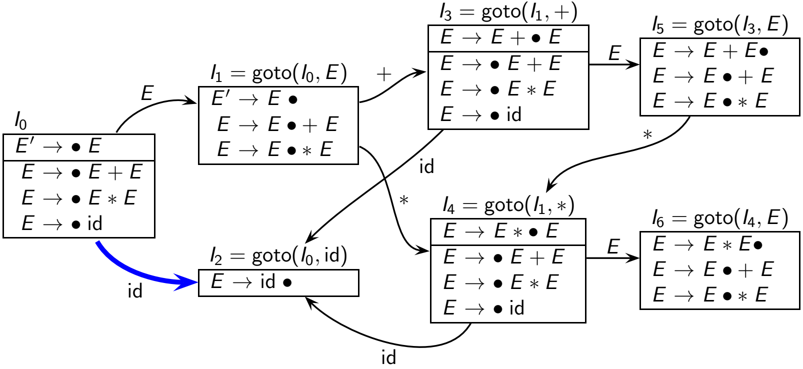
$I_6 = \text{goto}(I_4, E)$
$E \rightarrow E * E \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

Putting it All Together: Constructing the Parsing Table

0 $E' \to E$
1 $E \to E + E$
2 $E \to E * E$
3 $E \to \text{id}$

$FOLLOW(E') = \{\$\}$
$FOLLOW(E) = \{\$, +, *\}$

	id	+	*	\$	E
0	$s2$				$c1$
1		$s3$	$s4$	acc	
2		$r3$	$r3$	$r3$	
3	$s2$				$c5$
4	$s2$				$c6$
5		$r1$	$s4$	$r1$	
6		$r2$	$r2$	$r2$	

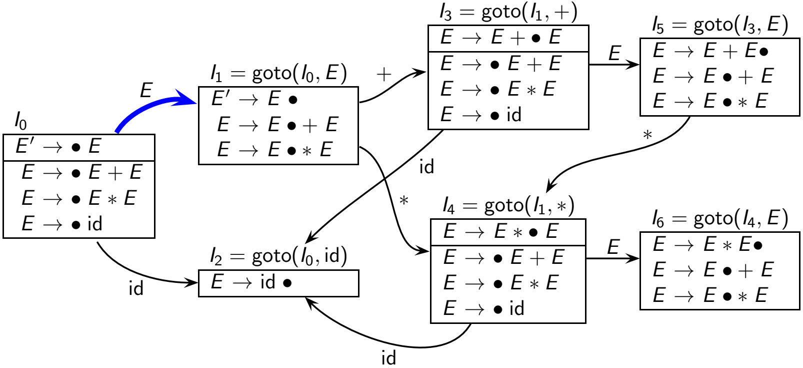


Putting it All Together: Constructing the Parsing Table

$0\ E' \rightarrow E$
$1\ E \rightarrow E + E$
$2\ E \rightarrow E * E$
$3\ E \rightarrow \text{id}$

$FOLLOW(E') = \{\$\}$
$FOLLOW(E) = \{\$, +, *\}$

	id	+	*	$	E
0	s2				c1
1		s3	s4	acc	
2		r3	r3	r3	
3	s2				c5
4	s2				c6
5		r1	s4	r1	
6		r2	r2	r2	



I_0
$E' \rightarrow \bullet E$
$E \rightarrow \bullet E + E$
$E \rightarrow \bullet E * E$
$E \rightarrow \bullet \text{id}$

$I_1 = \text{goto}(I_0, E)$
$E' \rightarrow E \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

$I_2 = \text{goto}(I_0, \text{id})$
$E \rightarrow \text{id} \bullet$

$I_3 = \text{goto}(I_1, +)$
$E \rightarrow E + \bullet E$
$E \rightarrow \bullet E + E$
$E \rightarrow \bullet E * E$
$E \rightarrow \bullet \text{id}$

$I_4 = \text{goto}(I_1, *)$
$E \rightarrow E * \bullet E$
$E \rightarrow \bullet E + E$
$E \rightarrow \bullet E * E$
$E \rightarrow \bullet \text{id}$

$I_5 = \text{goto}(I_3, E)$
$E \rightarrow E + E \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

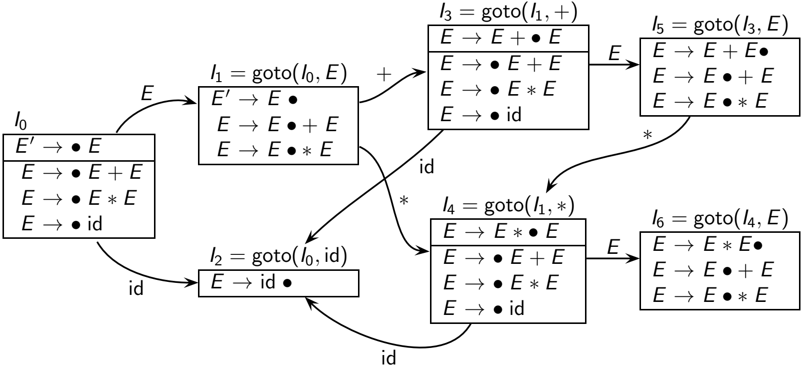
$I_6 = \text{goto}(I_4, E)$
$E \rightarrow E * E \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

Putting it All Together: Constructing the Parsing Table

$\boxed{0 \; E' \to E}$
$1 \; E \to E + E$
$2 \; E \to E * E$
$3 \; E \to \text{id}$

$FOLLOW(E') = \{\$\}$
$FOLLOW(E) = \{\$, +, *\}$

	id	+	*	$	E
0	s2				c1
1		s3	s4	$\boxed{acc}$	
2		r3	r3	r3	
3	s2				c5
4	s2				c6
5		r1	s4	r1	
6		r2	r2	r2	



I_0
$E' \to \bullet E$
$E \to \bullet E + E$
$E \to \bullet E * E$
$E \to \bullet \text{id}$

$I_1 = \text{goto}(I_0, E)$
$E' \to E \bullet$
$E \to E \bullet + E$
$E \to E \bullet * E$

$I_2 = \text{goto}(I_0, \text{id})$
$E \to \text{id} \bullet$

$I_3 = \text{goto}(I_1, +)$
$E \to E + \bullet E$
$E \to \bullet E + E$
$E \to \bullet E * E$
$E \to \bullet \text{id}$

$I_4 = \text{goto}(I_1, *)$
$E \to E * \bullet E$
$E \to \bullet E + E$
$E \to \bullet E * E$
$E \to \bullet \text{id}$

$I_5 = \text{goto}(I_3, E)$
$E \to E + E \bullet$
$E \to E \bullet + E$
$E \to E \bullet * E$

$I_6 = \text{goto}(I_4, E)$
$E \to E * E \bullet$
$E \to E \bullet + E$
$E \to E \bullet * E$

IIT Bombay
cs302: Implementation of Programming Languages

Topic:

Syntax Analysis

Section:

Grammars, Derivations, and Parse Trees

Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in Parsing

CLR(1) Parsing

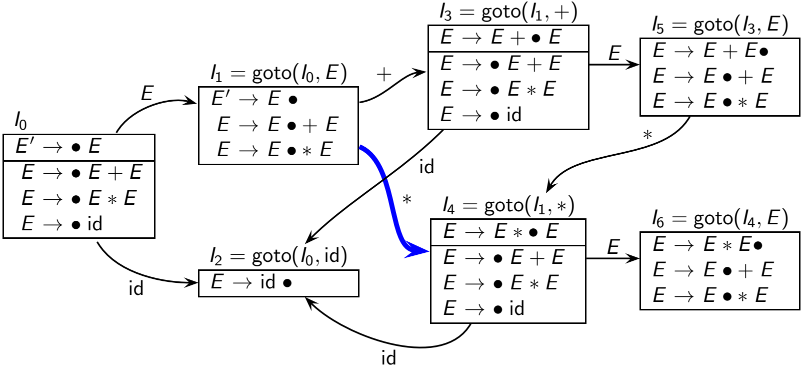
LALR(1) Parsing

Putting it All Together: Constructing the Parsing Table

0 $E' \rightarrow E$
1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

$$FOLLOW(E') = \{\$\}$$
$$FOLLOW(E) = \{\$, +, *\}$$

	id	$+$	$*$	\$	E
0	$s2$				$c1$
1		$s3$	$\boxed{s4}$	acc	
2		$r3$	$r3$	$r3$	
3	$s2$				$c5$
4	$s2$				$c6$
5		$r1$	$s4$	$r1$	
6		$r2$	$r2$	$r2$	



I_0
$E' \rightarrow \bullet E$
$E \rightarrow \bullet E + E$
$E \rightarrow \bullet E * E$
$E \rightarrow \bullet \text{id}$

$I_1 = \text{goto}(I_0, E)$
$E' \rightarrow E \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

$I_2 = \text{goto}(I_0, \text{id})$
$E \rightarrow \text{id} \bullet$

$I_3 = \text{goto}(I_1, +)$
$E \rightarrow E + \bullet E$
$E \rightarrow \bullet E + E$
$E \rightarrow \bullet E * E$
$E \rightarrow \bullet \text{id}$

$I_4 = \text{goto}(I_1, *)$
$E \rightarrow E * \bullet E$
$E \rightarrow \bullet E + E$
$E \rightarrow \bullet E * E$
$E \rightarrow \bullet \text{id}$

$I_5 = \text{goto}(I_3, E)$
$E \rightarrow E + E \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

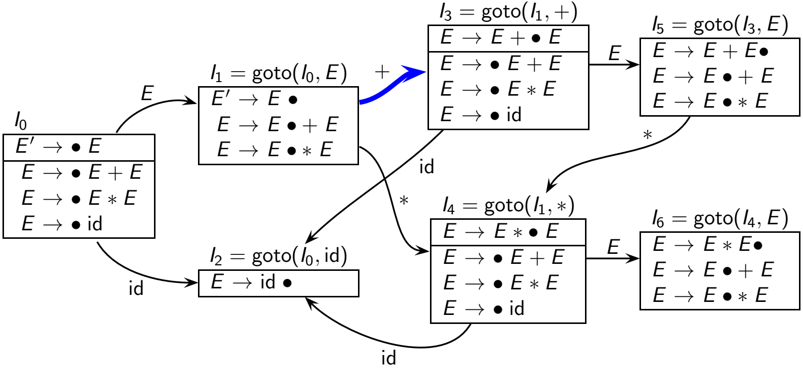
$I_6 = \text{goto}(I_4, E)$
$E \rightarrow E * E \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

Putting it All Together: Constructing the Parsing Table

	id	$+$	$*$	\$	E
0	$s2$				$c1$
1		$s3$	$s4$	acc	
2		$r3$	$r3$	$r3$	
3	$s2$				$c5$
4	$s2$				$c6$
5		$r1$	$s4$	$r1$	
6		$r2$	$r2$	$r2$	

0 $E' \to E$
1 $E \to E + E$
2 $E \to E * E$
3 $E \to \text{id}$

$$FOLLOW(E') = \{\$\}$$
$$FOLLOW(E) = \{\$, +, *\}$$

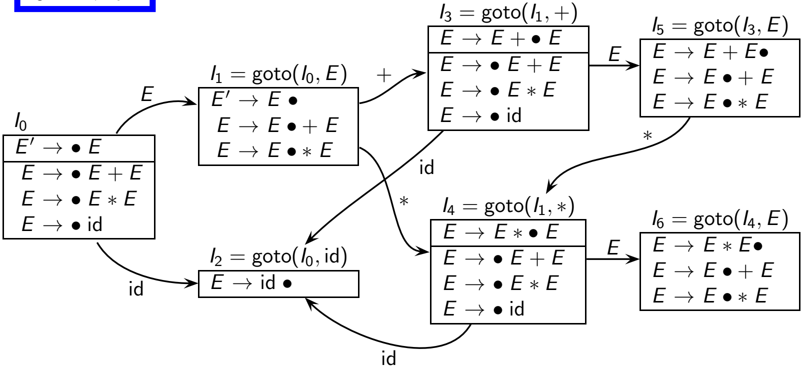


Putting it All Together: Constructing the Parsing Table

0 $E' \to E$
1 $E \to E + E$
2 $E \to E * E$
3 $E \to \text{id}$

$FOLLOW(E') = \{\$\}$
$FOLLOW(E) = \{\$, +, *\}$

	id	+	*	$	E
0	s2				c1
1		s3	s4	acc	
2		r3	r3	r3	
3	s2				c5
4	s2				c6
5		r1	s4	r1	
6		r2	r2	r2	

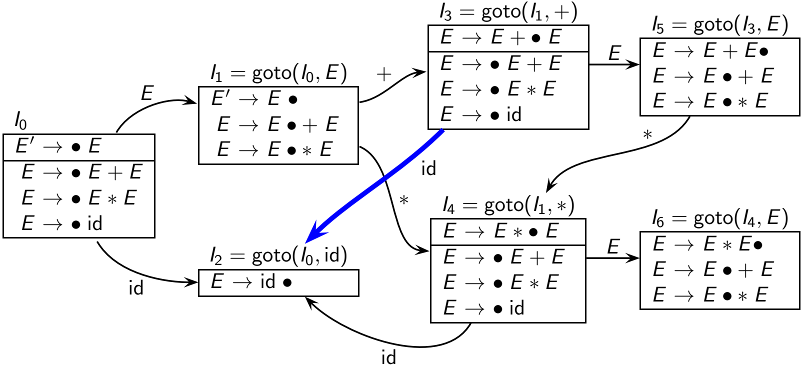


Putting it All Together: Constructing the Parsing Table

	id	$+$	$*$	\$	E
0	$s2$				$c1$
1		$s3$	$s4$	acc	
2		$r3$	$r3$	$r3$	
3	$s2$				$c5$
4	$s2$				$c6$
5		$r1$	$s4$	$r1$	
6		$r2$	$r2$	$r2$	

$0\ E' \rightarrow E$
$1\ E \rightarrow E + E$
$2\ E \rightarrow E * E$
$3\ E \rightarrow \text{id}$

$FOLLOW(E') = \{\$\}$
$FOLLOW(E) = \{\$, +, *\}$



I_0
$E' \rightarrow \bullet E$
$E \rightarrow \bullet E + E$
$E \rightarrow \bullet E * E$
$E \rightarrow \bullet \text{id}$

$I_1 = \text{goto}(I_0, E)$
$E' \rightarrow E \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

$I_2 = \text{goto}(I_0, \text{id})$
$E \rightarrow \text{id} \bullet$

$I_3 = \text{goto}(I_1, +)$
$E \rightarrow E + \bullet E$
$E \rightarrow \bullet E + E$
$E \rightarrow \bullet E * E$
$E \rightarrow \bullet \text{id}$

$I_4 = \text{goto}(I_1, *)$
$E \rightarrow E * \bullet E$
$E \rightarrow \bullet E + E$
$E \rightarrow \bullet E * E$
$E \rightarrow \bullet \text{id}$

$I_5 = \text{goto}(I_3, E)$
$E \rightarrow E + E \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

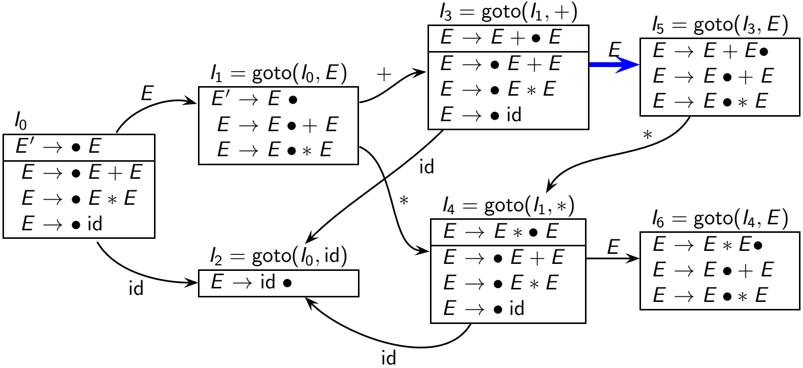
$I_6 = \text{goto}(I_4, E)$
$E \rightarrow E * E \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

Putting it All Together: Constructing the Parsing Table

	id	$+$	$*$	\$	E
0	$s2$				$c1$
1		$s3$	$s4$	acc	
2		$r3$	$r3$	$r3$	
3	$s2$				$\boxed{c5}$
4	$s2$				$c6$
5		$r1$	$s4$	$r1$	
6		$r2$	$r2$	$r2$	

$0\ E' \rightarrow E$
$1\ E \rightarrow E + E$
$2\ E \rightarrow E * E$
$3\ E \rightarrow \text{id}$

$FOLLOW(E') = \{\$\}$
$FOLLOW(E) = \{\$, +, *\}$



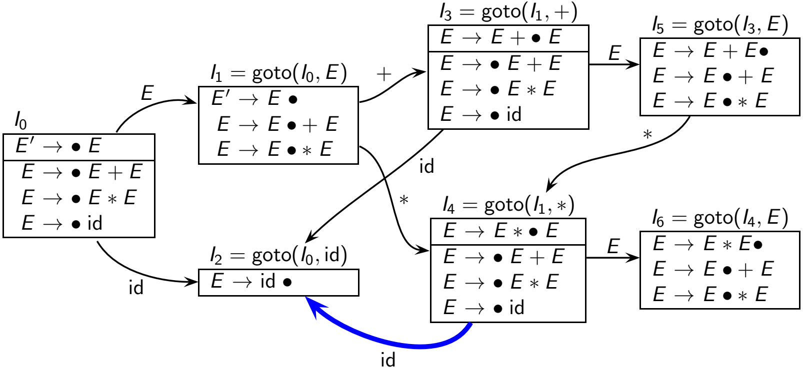
Putting it All Together: Constructing the Parsing Table

$$
\begin{array}{lll}
0 & E' & \rightarrow E \\
1 & E & \rightarrow E + E \\
2 & E & \rightarrow E * E \\
3 & E & \rightarrow \text{id}
\end{array}
$$

$FOLLOW(E') = \{\$\}$

$FOLLOW(E) = \{\$, +, *\}$

	id	+	*	\$	E
0	*s2*				*c1*
1		*s3*	*s4*	*acc*	
2		*r3*	*r3*	*r3*	
3	*s2*				*c5*
4	*s2*				*c6*
5		*r1*	*s4*	*r1*	
6		*r2*	*r2*	*r2*	



I_0
$E' \rightarrow \bullet E$
$E \rightarrow \bullet E + E$
$E \rightarrow \bullet E * E$
$E \rightarrow \bullet \text{id}$

$I_1 = \text{goto}(I_0, E)$
$E' \rightarrow E \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

$I_2 = \text{goto}(I_0, \text{id})$
$E \rightarrow \text{id} \bullet$

$I_3 = \text{goto}(I_1, +)$
$E \rightarrow E + \bullet E$
$E \rightarrow \bullet E + E$
$E \rightarrow \bullet E * E$
$E \rightarrow \bullet \text{id}$

$I_4 = \text{goto}(I_1, *)$
$E \rightarrow E * \bullet E$
$E \rightarrow \bullet E + E$
$E \rightarrow \bullet E * E$
$E \rightarrow \bullet \text{id}$

$I_5 = \text{goto}(I_3, E)$
$E \rightarrow E + E \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

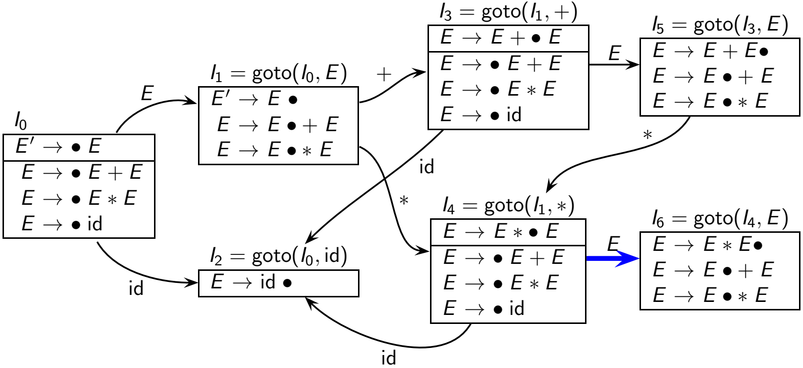
$I_6 = \text{goto}(I_4, E)$
$E \rightarrow E * E \bullet$
$E \rightarrow E \bullet + E$
$E \rightarrow E \bullet * E$

Putting it All Together: Constructing the Parsing Table

0 $E' \to E$
1 $E \to E + E$
2 $E \to E * E$
3 $E \to \text{id}$

$FOLLOW(E') = \{\$\}$
$FOLLOW(E) = \{\$, +, *\}$

	id	+	*	\$	E
0	s2				c1
1		s3	s4	acc	
2		r3	r3	r3	
3	s2				c5
4	s2				c6
5		r1	s4	r1	
6		r2	r2	r2	

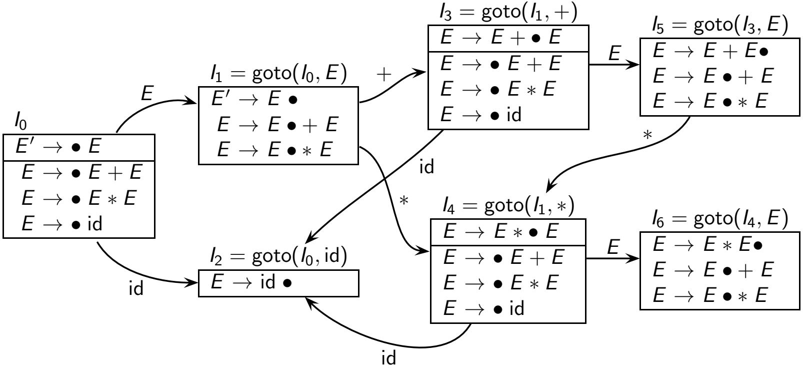


Putting it All Together: Constructing the Parsing Table

$0\ E' \rightarrow E$
$\boxed{1\ E \rightarrow E + E}$
$2\ E \rightarrow E * E$
$3\ E \rightarrow \text{id}$

$FOLLOW(E') = \{\$\}$
$FOLLOW(E) = \{\$, +, *\}$

	id	+	*	$	E
0	s2				c1
1		s3	s4	acc	
2		r3	r3	r3	
3	s2				c5
4	s2				c6
5		$\boxed{r1}$	s4	$\boxed{r1}$	
6		r2	r2	r2	

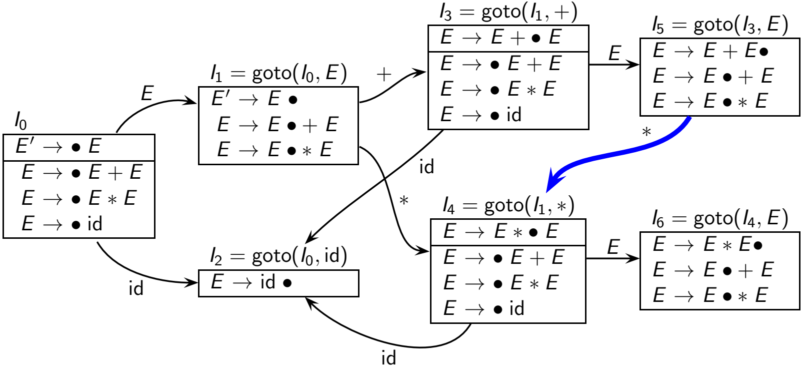


Putting it All Together: Constructing the Parsing Table

$0\ E' \rightarrow E$
$1\ E \rightarrow E + E$
$2\ E \rightarrow E * E$
$3\ E \rightarrow \text{id}$

$FOLLOW(E') = \{\$\}$
$FOLLOW(E) = \{\$, +, *\}$

	id	+	*	$	E
0	*s*2				*c*1
1		*s*3	*s*4	*acc*	
2		*r*3	*r*3	*r*3	
3	*s*2				*c*5
4	*s*2				*c*6
5		*r*1	$\boxed{s4}$	*r*1	
6		*r*2	*r*2	*r*2	

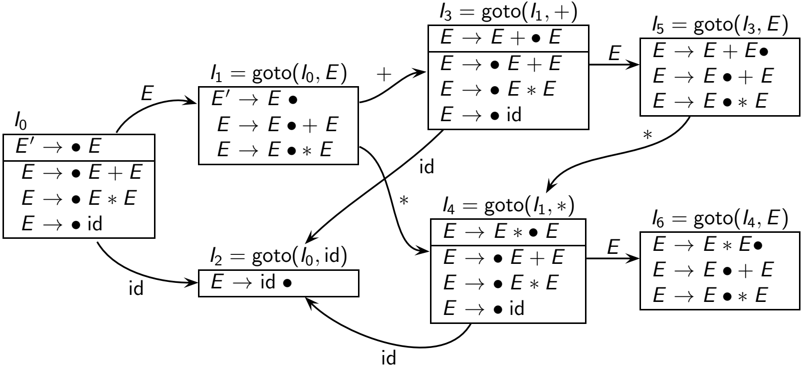


Putting it All Together: Constructing the Parsing Table

0 $E' \rightarrow E$
1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

$FOLLOW(E') = \{\$\}$
$FOLLOW(E) = \{\$, +, *\}$

	id	+	*	\$	E
0	$s2$				$c1$
1		$s3$	$s4$	acc	
2		$r3$	$r3$	$r3$	
3	$s2$				$c5$
4	$s2$				$c6$
5		$r1$	$s4$	$r1$	
6		$r2$	$r2$	$r2$	



Destination Reached: From Intuitions to Formal Algorithms in Shift Reduce Parsing

1 $E \rightarrow E + E$
2 $E \rightarrow E * E$
3 $E \rightarrow \text{id}$

	id	+	*	\$	E
0	s2				c1
1		s3	s4	acc	
2		r3	r3	r3	
3	s2				c5
4	s2				c6
5		~~s3~~/r1	s4/~~r1~~	r1	
6		~~s3~~/r2	~~s4~~/r2	r2	

Step	Stack →	Input	Action
1	\$	id + id ∗ id\$	shift
2	\$id	+ id ∗ id\$	reduce by 3
3	\$E	+ id ∗ id\$	shift
4	\$E +	id ∗ id\$	shift
5	\$E + id	∗ id\$	reduce by 3
6	\$E + E	∗ id\$	shift
7	\$E + E ∗	id\$	shift
8	\$E + E ∗ id	\$	reduce by 3
9	\$E + E ∗ E	\$	reduce by 2
10	\$E + E	\$	reduce by 1
11	\$E	\$	accept

Step	Stack →	Input	Action
1	\$0	id + id ∗ id\$	s2
2	\$0 id 2	+ id ∗ id\$	r3 and c1
3	\$0 E 1	+ id ∗ id\$	s3
4	\$0 E 1 + 3	id ∗ id\$	s2
5	\$0 E 1 + 3 id 2	∗ id\$	r3 and c5
6	\$0 E 1 + 3 E 5	∗ id\$	s4
7	\$0 E 1 + 3 E 5 ∗ 4	id\$	s2
8	\$0 E 1 + 3 E 5 ∗ 4 id 2	\$	r3 and c6
9	\$0 E 1 + 3 E 5 ∗ 4 E6	\$	r2 and c5
10	\$0 E 1 + 3 E 5	\$	r1 and c1
11	\$0 E 1	\$	accept

Explaining Conflicts in Yacc

- Using bison -d option to generate .output file
- Using bison -g option to generate dot file of LR(0) automaton
- lex-yacc-intro-programs/yacc-conflict-demo/README
- yacc-actions-demo/simcalc-using-lex-yacc-c++
  To show the need of %prec UMINUS

Conceptual Issues in Parsing

Conceptual Issues in Parsing

parsing-slides-sanyal-part4.pdf

CLR(1) Parsing

Limitation of SLR(1) Parsing

- We illustrate the limitations of SLR(1) parsing by using the pointer assignment grammar given below

$$\begin{array}{rcl} S & \rightarrow & L = R \mid R \\ L & \rightarrow & *R \mid \text{id} \\ R & \rightarrow & L \end{array}$$

- We compute the FOLLOW sets and sets of LR(0) items to demonstrate the problem
- We explain the cause of the problem
- This explanation leads us to a more precise method of CLR(1) parsing (Canonical LR(1) parsing that uses the LR(1) items)

Computing the FOLLOW Sets for Pointer Assignment Grammar

$$S' \rightarrow S$$

$$S \rightarrow L = R \mid R$$

$$L \rightarrow *R \mid \text{id}$$
$$R \rightarrow L$$

Computing the FOLLOW Sets for Pointer Assignment Grammar

$$
\begin{array}{rcl}
S' & \rightarrow & S \qquad\qquad \Rightarrow \quad \text{FOLLOW}(S') \supseteq \{\$\} \\
 & & \qquad\qquad\qquad\quad \text{FOLLOW}(S) \supseteq \text{FOLLOW}(S') \\
S & \rightarrow & L = R \mid R \\
L & \rightarrow & *R \mid \text{id} \\
R & \rightarrow & L
\end{array}
$$

Computing the FOLLOW Sets for Pointer Assignment Grammar

$$\begin{array}{rcl} S' & \rightarrow & S \\ S & \rightarrow & L = R \mid R \\ L & \rightarrow & *R \mid \text{id} \\ R & \rightarrow & L \end{array}$$

$$\begin{array}{lcl} \Rightarrow \text{FOLLOW}(S') & \supseteq & \{\$\} \\ \text{FOLLOW}(S) & \supseteq & \text{FOLLOW}(S') \\ \Rightarrow \text{FOLLOW}(L) & \supseteq & \{=\} \\ \text{FOLLOW}(R) & \supseteq & \text{FOLLOW}(S) \end{array}$$

Computing the FOLLOW Sets for Pointer Assignment Grammar

$$\begin{array}{rcll} S' & \rightarrow & S & \Rightarrow \text{FOLLOW}(S') \supseteq \{\$\} \\ & & & \text{FOLLOW}(S) \supseteq \text{FOLLOW}(S') \\ S & \rightarrow & L = R \mid R & \Rightarrow \text{FOLLOW}(L) \supseteq \{=\} \\ & & & \text{FOLLOW}(R) \supseteq \text{FOLLOW}(S) \\ L & \rightarrow & *R \mid \text{id} & \Rightarrow \text{FOLLOW}(R) \supseteq \text{FOLLOW}(L) \\ R & \rightarrow & L & \end{array}$$

Computing the FOLLOW Sets for Pointer Assignment Grammar

$$\begin{array}{lcllcl}
S' & \rightarrow & S & \Rightarrow & \text{FOLLOW}(S') & \supseteq & \{\$\} \\
 & & & & \text{FOLLOW}(S) & \supseteq & \text{FOLLOW}(S') \\
S & \rightarrow & L = R \mid R & \Rightarrow & \text{FOLLOW}(L) & \supseteq & \{=\} \\
 & & & & \text{FOLLOW}(R) & \supseteq & \text{FOLLOW}(S) \\
L & \rightarrow & *R \mid \text{id} & \Rightarrow & \text{FOLLOW}(R) & \supseteq & \text{FOLLOW}(L) \\
R & \rightarrow & L & \Rightarrow & \text{FOLLOW}(L) & \supseteq & \text{FOLLOW}(R)
\end{array}$$

Computing the FOLLOW Sets for Pointer Assignment Grammar

$$\begin{array}{rcllcl}
S' & \rightarrow & S & \Rightarrow \text{FOLLOW}(S') & \supseteq & \{\$\} \\
 & & & \text{FOLLOW}(S) & \supseteq & \text{FOLLOW}(S') \\
S & \rightarrow & L = R \mid R & \Rightarrow \text{FOLLOW}(L) & \supseteq & \{=\} \\
 & & & \text{FOLLOW}(R) & \supseteq & \text{FOLLOW}(S) \\
L & \rightarrow & *R \mid \text{id} & \Rightarrow \text{FOLLOW}(R) & \supseteq & \text{FOLLOW}(L) \\
R & \rightarrow & L & \Rightarrow \text{FOLLOW}(L) & \supseteq & \text{FOLLOW}(R)
\end{array}$$

	FOLLOW
S'	$\{\$\}$
S	$\{\$\}$
R	$\{=, \$\}$
L	$\{=, \$\}$

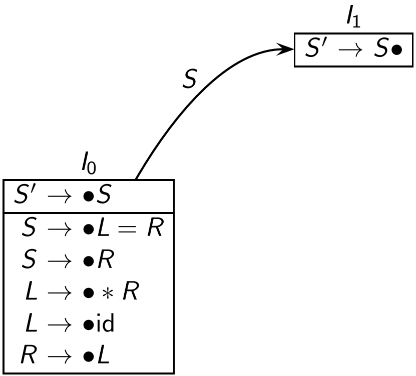
LR(0) Item Sets for Pointer Assignment Grammar

I_0

$S' \rightarrow \bullet S$

$S \rightarrow \bullet L = R$
$S \rightarrow \bullet R$
$L \rightarrow \bullet * R$
$L \rightarrow \bullet \mathrm{id}$
$R \rightarrow \bullet L$

LR(0) Item Sets for Pointer Assignment Grammar



I_0

$S' \rightarrow \bullet S$

$S \rightarrow \bullet L = R$

$S \rightarrow \bullet R$

$L \rightarrow \bullet * R$

$L \rightarrow \bullet \text{id}$

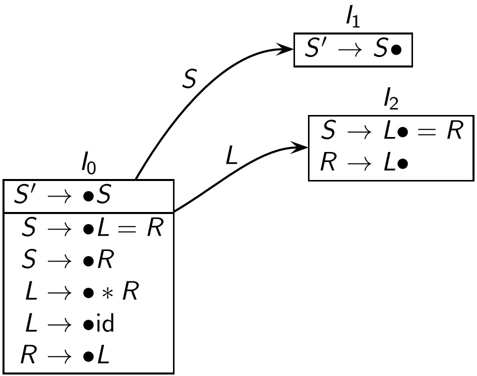
$R \rightarrow \bullet L$

S

I_1

$S' \rightarrow S \bullet$

LR(0) Item Sets for Pointer Assignment Grammar



I_0

$S' \rightarrow \bullet S$

$S \rightarrow \bullet L = R$
$S \rightarrow \bullet R$
$L \rightarrow \bullet * R$
$L \rightarrow \bullet \text{id}$
$R \rightarrow \bullet L$

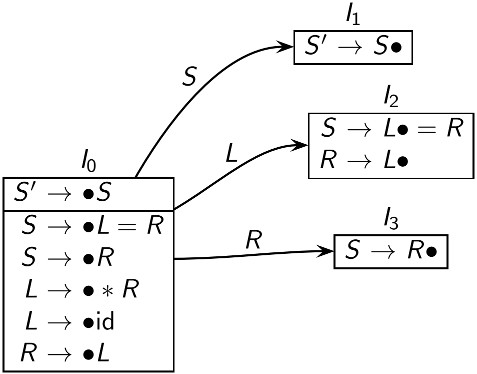
I_1

$S' \rightarrow S \bullet$

I_2

$S \rightarrow L \bullet = R$
$R \rightarrow L \bullet$

LR(0) Item Sets for Pointer Assignment Grammar



I_0

$S' \rightarrow \bullet S$

$S \rightarrow \bullet L = R$

$S \rightarrow \bullet R$

$L \rightarrow \bullet * R$

$L \rightarrow \bullet \text{id}$

$R \rightarrow \bullet L$

I_1

$S' \rightarrow S \bullet$

I_2

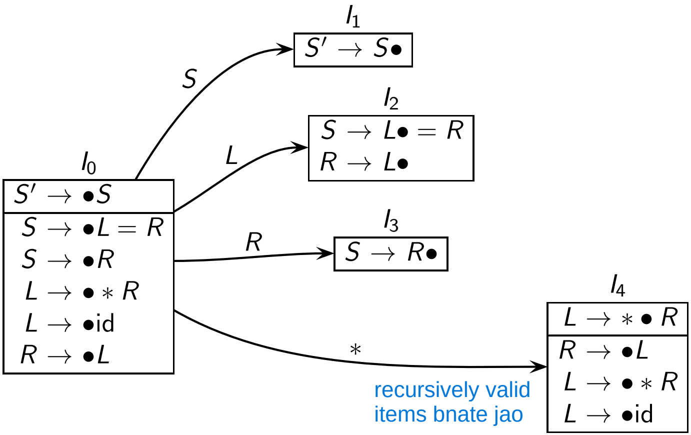
$S \rightarrow L \bullet = R$

$R \rightarrow L \bullet$

I_3

$S \rightarrow R \bullet$

LR(0) Item Sets for Pointer Assignment Grammar



I_0

$S' \rightarrow \bullet S$

$S \rightarrow \bullet L = R$

$S \rightarrow \bullet R$

$L \rightarrow \bullet * R$

$L \rightarrow \bullet \text{id}$

$R \rightarrow \bullet L$

I_1

$S' \rightarrow S \bullet$

I_2

$S \rightarrow L \bullet = R$

$R \rightarrow L \bullet$

I_3

$S \rightarrow R \bullet$

I_4

$L \rightarrow * \bullet R$

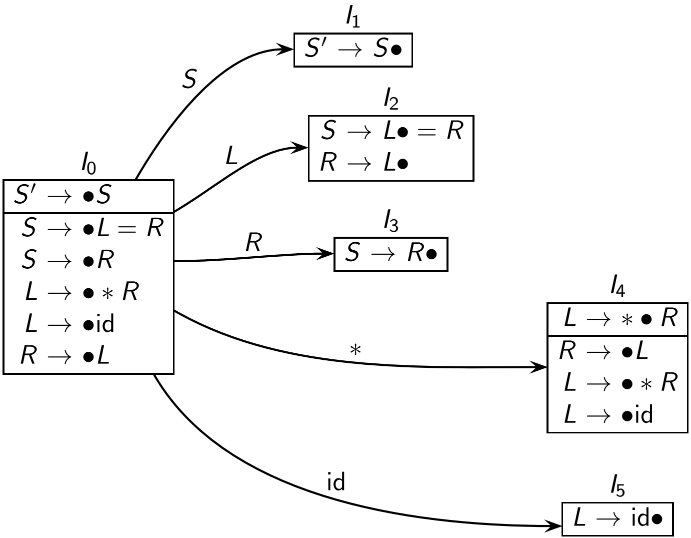
$R \rightarrow \bullet L$

$L \rightarrow \bullet * R$

$L \rightarrow \bullet \text{id}$

recursively valid
items bnate jao

LR(0) Item Sets for Pointer Assignment Grammar



I_0

$S' \rightarrow \bullet S$

$S \rightarrow \bullet L = R$
$S \rightarrow \bullet R$
$L \rightarrow \bullet * R$
$L \rightarrow \bullet \text{id}$
$R \rightarrow \bullet L$

I_1

$S' \rightarrow S\bullet$

I_2

$S \rightarrow L\bullet = R$
$R \rightarrow L\bullet$

I_3

$S \rightarrow R\bullet$

I_4

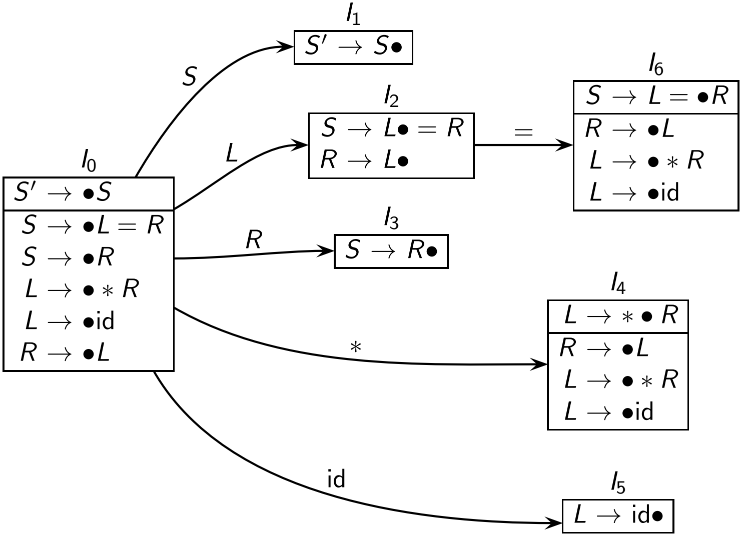
$L \rightarrow * \bullet R$

$R \rightarrow \bullet L$
$L \rightarrow \bullet * R$
$L \rightarrow \bullet \text{id}$

I_5

$L \rightarrow \text{id}\bullet$

LR(0) Item Sets for Pointer Assignment Grammar



I_0
$S' \rightarrow \bullet S$
$S \rightarrow \bullet L = R$
$S \rightarrow \bullet R$
$L \rightarrow \bullet * R$
$L \rightarrow \bullet \text{id}$
$R \rightarrow \bullet L$

I_1
$S' \rightarrow S \bullet$

I_2
$S \rightarrow L \bullet = R$
$R \rightarrow L \bullet$

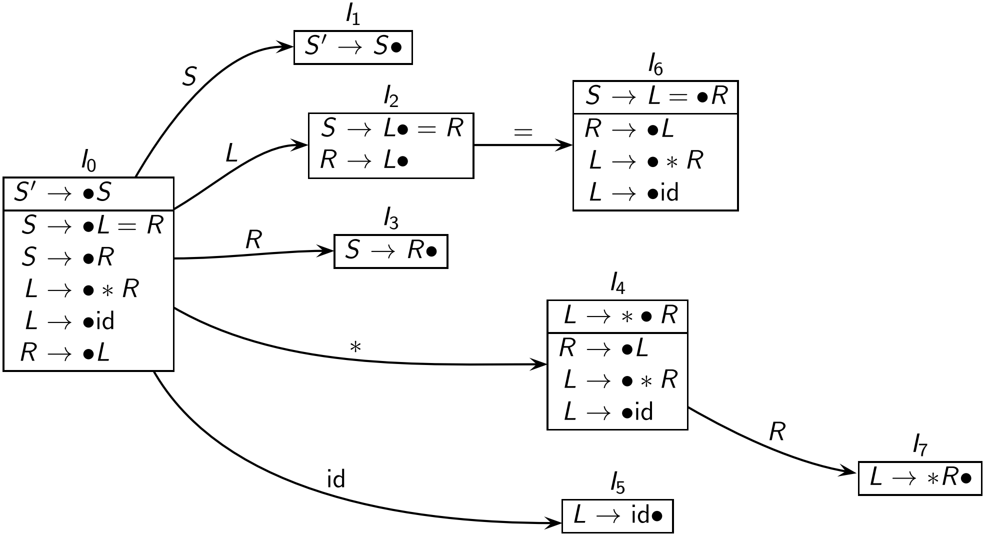
I_3
$S \rightarrow R \bullet$

I_4
$L \rightarrow * \bullet R$
$R \rightarrow \bullet L$
$L \rightarrow \bullet * R$
$L \rightarrow \bullet \text{id}$

I_5
$L \rightarrow \text{id} \bullet$

I_6
$S \rightarrow L = \bullet R$
$R \rightarrow \bullet L$
$L \rightarrow \bullet * R$
$L \rightarrow \bullet \text{id}$

LR(0) Item Sets for Pointer Assignment Grammar



I_0

$S' \rightarrow \bullet S$

$S \rightarrow \bullet L = R$

$S \rightarrow \bullet R$

$L \rightarrow \bullet * R$

$L \rightarrow \bullet \text{id}$

$R \rightarrow \bullet L$

I_1

$S' \rightarrow S \bullet$

I_2

$S \rightarrow L \bullet = R$

$R \rightarrow L \bullet$

I_3

$S \rightarrow R \bullet$

I_4

$L \rightarrow * \bullet R$

$R \rightarrow \bullet L$

$L \rightarrow \bullet * R$

$L \rightarrow \bullet \text{id}$

I_5

$L \rightarrow \text{id} \bullet$

I_6

$S \rightarrow L = \bullet R$

$R \rightarrow \bullet L$

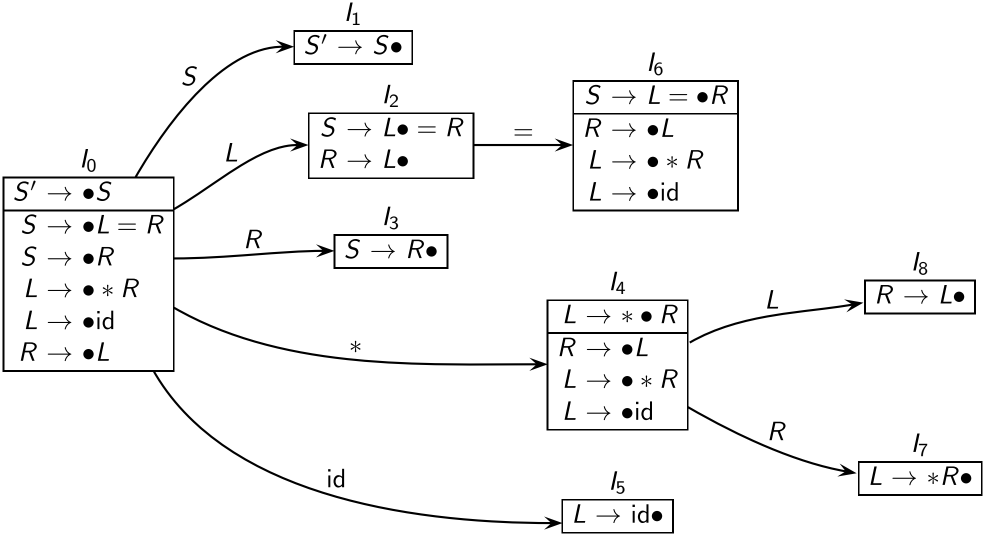
$L \rightarrow \bullet * R$

$L \rightarrow \bullet \text{id}$

I_7

$L \rightarrow * R \bullet$

LR(0) Item Sets for Pointer Assignment Grammar



I_0
$S' \to \bullet S$
$S \to \bullet L = R$
$S \to \bullet R$
$L \to \bullet * R$
$L \to \bullet \text{id}$
$R \to \bullet L$

I_1
$S' \to S \bullet$

I_2
$S \to L \bullet = R$
$R \to L \bullet$

I_3
$S \to R \bullet$

I_4
$L \to * \bullet R$
$R \to \bullet L$
$L \to \bullet * R$
$L \to \bullet \text{id}$

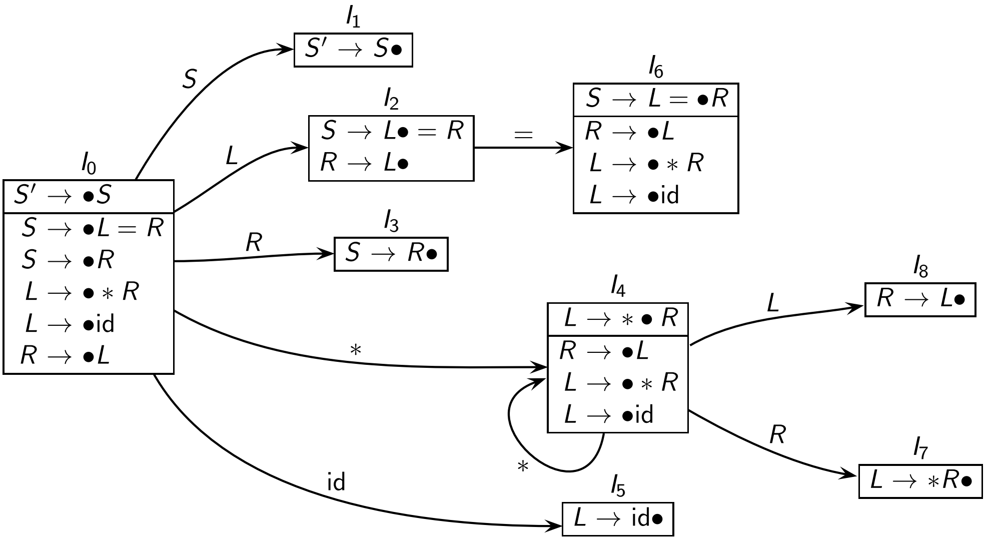
I_5
$L \to \text{id} \bullet$

I_6
$S \to L = \bullet R$
$R \to \bullet L$
$L \to \bullet * R$
$L \to \bullet \text{id}$

I_7
$L \to * R \bullet$

I_8
$R \to L \bullet$

LR(0) Item Sets for Pointer Assignment Grammar



I_0
$S' \to \bullet S$
$S \to \bullet L = R$
$S \to \bullet R$
$L \to \bullet * R$
$L \to \bullet \text{id}$
$R \to \bullet L$

I_1
$S' \to S \bullet$

I_2
$S \to L \bullet = R$
$R \to L \bullet$

I_3
$S \to R \bullet$

I_4
$L \to * \bullet R$
$R \to \bullet L$
$L \to \bullet * R$
$L \to \bullet \text{id}$

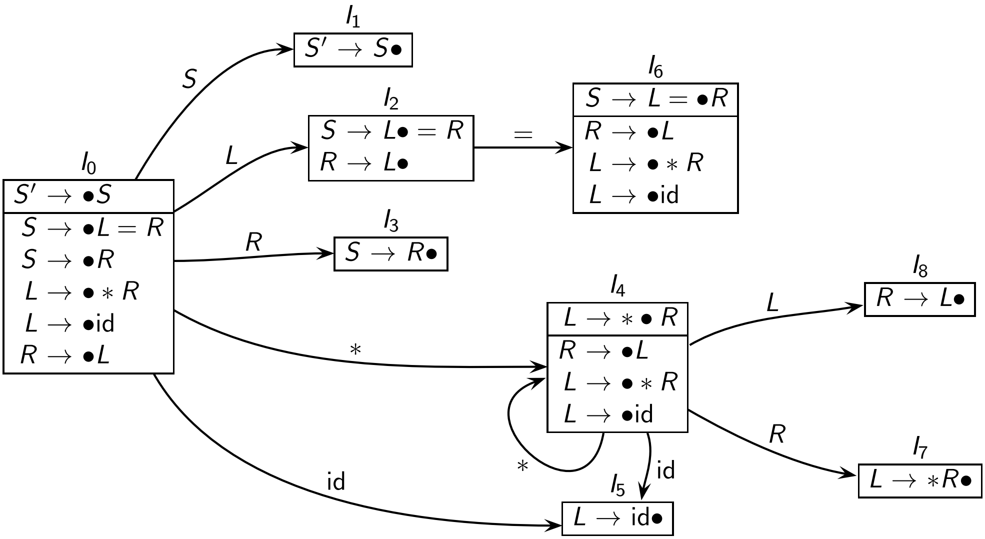
I_5
$L \to \text{id} \bullet$

I_6
$S \to L = \bullet R$
$R \to \bullet L$
$L \to \bullet * R$
$L \to \bullet \text{id}$

I_7
$L \to * R \bullet$

I_8
$R \to L \bullet$

LR(0) Item Sets for Pointer Assignment Grammar



I_0

$S' \to \bullet S$

$S \to \bullet L = R$
$S \to \bullet R$
$L \to \bullet * R$
$L \to \bullet \text{id}$
$R \to \bullet L$

I_1

$S' \to S \bullet$

I_2

$S \to L \bullet = R$
$R \to L \bullet$

I_3

$S \to R \bullet$

I_4

$L \to * \bullet R$

$R \to \bullet L$
$L \to \bullet * R$
$L \to \bullet \text{id}$

I_5

$L \to \text{id} \bullet$

I_6

$S \to L = \bullet R$

$R \to \bullet L$
$L \to \bullet * R$
$L \to \bullet \text{id}$

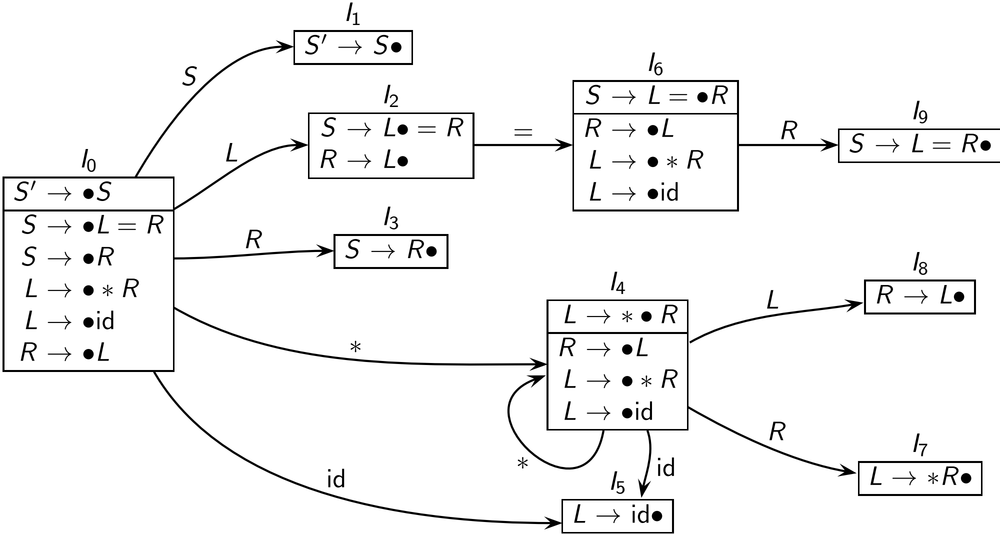
I_7

$L \to * R \bullet$

I_8

$R \to L \bullet$

LR(0) Item Sets for Pointer Assignment Grammar



I_0
$S' \to \bullet S$
$S \to \bullet L = R$
$S \to \bullet R$
$L \to \bullet * R$
$L \to \bullet \text{id}$
$R \to \bullet L$

I_1
$S' \to S\bullet$

I_2
$S \to L\bullet = R$
$R \to L\bullet$

I_3
$S \to R\bullet$

I_4
$L \to * \bullet R$
$R \to \bullet L$
$L \to \bullet * R$
$L \to \bullet \text{id}$

I_5
$L \to \text{id}\bullet$

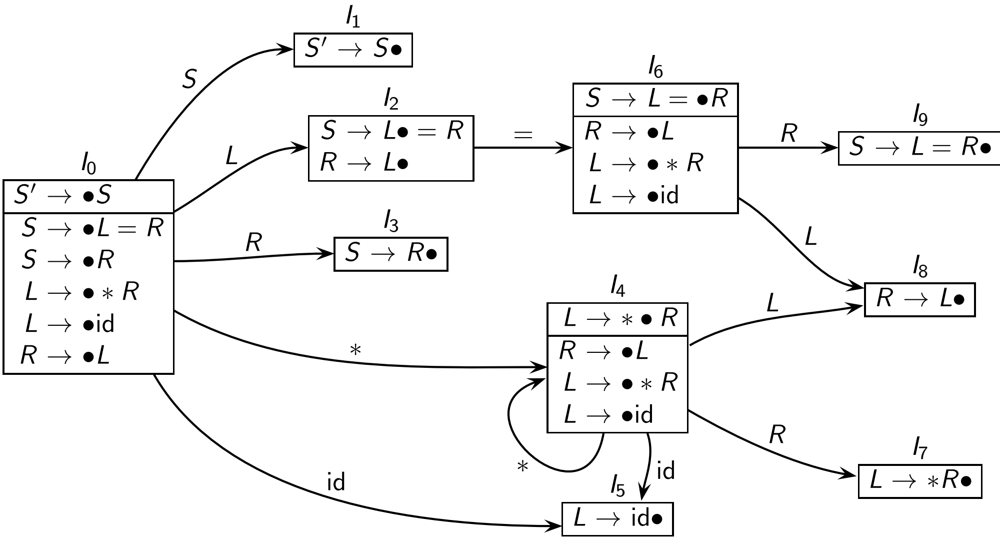
I_6
$S \to L = \bullet R$
$R \to \bullet L$
$L \to \bullet * R$
$L \to \bullet \text{id}$

I_7
$L \to *R\bullet$

I_8
$R \to L\bullet$

I_9
$S \to L = R\bullet$

LR(0) Item Sets for Pointer Assignment Grammar



I_0
$S' \to \bullet S$
$S \to \bullet L = R$
$S \to \bullet R$
$L \to \bullet * R$
$L \to \bullet \text{id}$
$R \to \bullet L$

I_1
$S' \to S\bullet$

I_2
$S \to L\bullet = R$
$R \to L\bullet$

I_3
$S \to R\bullet$

I_4
$L \to * \bullet R$
$R \to \bullet L$
$L \to \bullet * R$
$L \to \bullet \text{id}$

I_5
$L \to \text{id}\bullet$

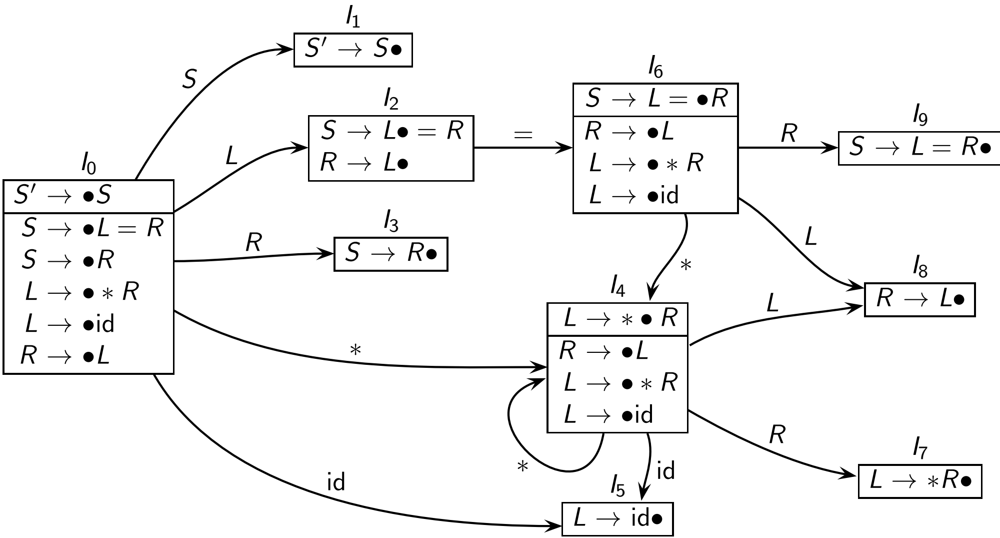
I_6
$S \to L = \bullet R$
$R \to \bullet L$
$L \to \bullet * R$
$L \to \bullet \text{id}$

I_7
$L \to *R\bullet$

I_8
$R \to L\bullet$

I_9
$S \to L = R\bullet$

LR(0) Item Sets for Pointer Assignment Grammar



I_0
$S' \rightarrow \bullet S$
$S \rightarrow \bullet L = R$
$S \rightarrow \bullet R$
$L \rightarrow \bullet * R$
$L \rightarrow \bullet \mathrm{id}$
$R \rightarrow \bullet L$

I_1
$S' \rightarrow S \bullet$

I_2
$S \rightarrow L \bullet = R$
$R \rightarrow L \bullet$

I_3
$S \rightarrow R \bullet$

I_4
$L \rightarrow * \bullet R$
$R \rightarrow \bullet L$
$L \rightarrow \bullet * R$
$L \rightarrow \bullet \mathrm{id}$

I_5
$L \rightarrow \mathrm{id} \bullet$

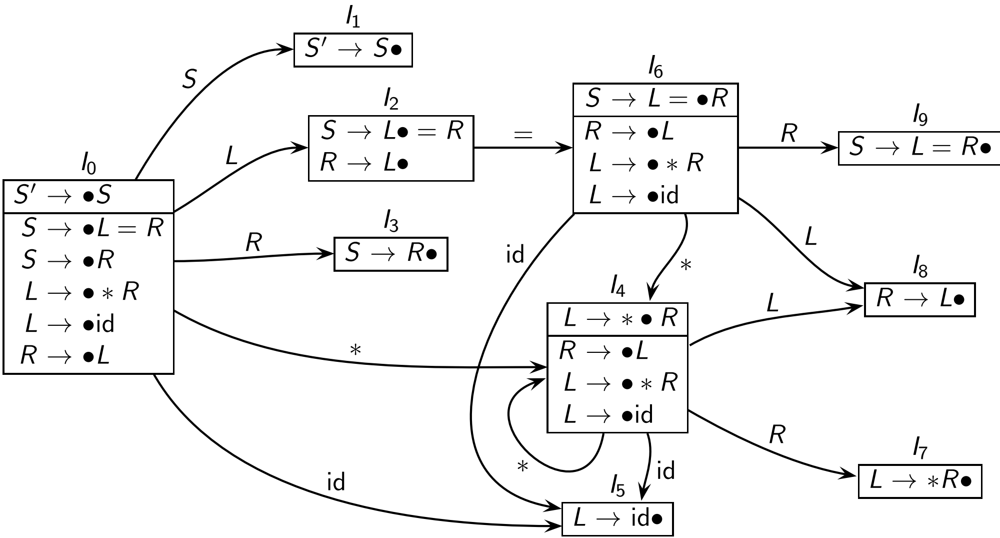
I_6
$S \rightarrow L = \bullet R$
$R \rightarrow \bullet L$
$L \rightarrow \bullet * R$
$L \rightarrow \bullet \mathrm{id}$

I_7
$L \rightarrow * R \bullet$

I_8
$R \rightarrow L \bullet$

I_9
$S \rightarrow L = R \bullet$

LR(0) Item Sets for Pointer Assignment Grammar



I_0
$S' \rightarrow \bullet S$
$S \rightarrow \bullet L = R$
$S \rightarrow \bullet R$
$L \rightarrow \bullet * R$
$L \rightarrow \bullet \text{id}$
$R \rightarrow \bullet L$

I_1
$S' \rightarrow S \bullet$

I_2
$S \rightarrow L \bullet = R$
$R \rightarrow L \bullet$

I_3
$S \rightarrow R \bullet$

I_4
$L \rightarrow * \bullet R$
$R \rightarrow \bullet L$
$L \rightarrow \bullet * R$
$L \rightarrow \bullet \text{id}$

I_5
$L \rightarrow \text{id} \bullet$

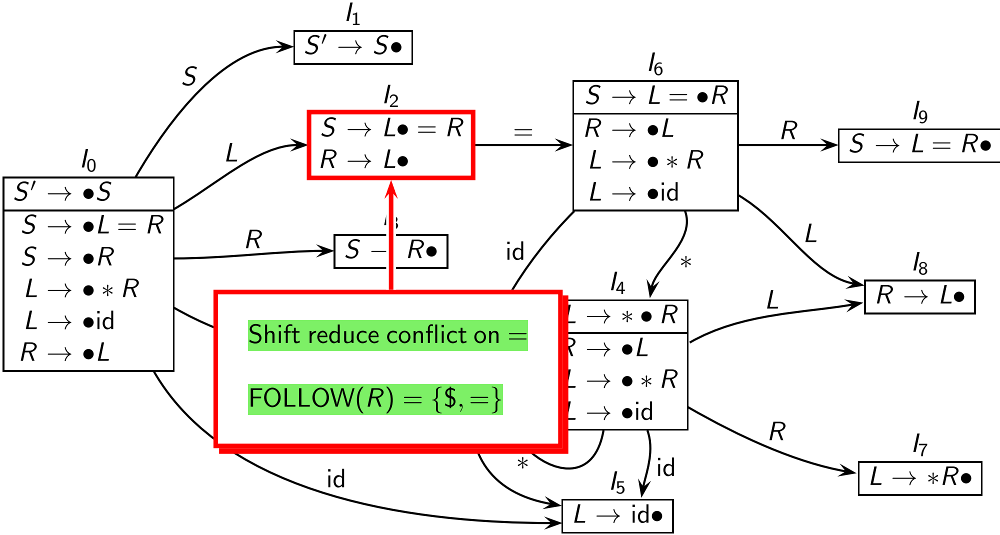
I_6
$S \rightarrow L = \bullet R$
$R \rightarrow \bullet L$
$L \rightarrow \bullet * R$
$L \rightarrow \bullet \text{id}$

I_7
$L \rightarrow * R \bullet$

I_8
$R \rightarrow L \bullet$

I_9
$S \rightarrow L = R \bullet$

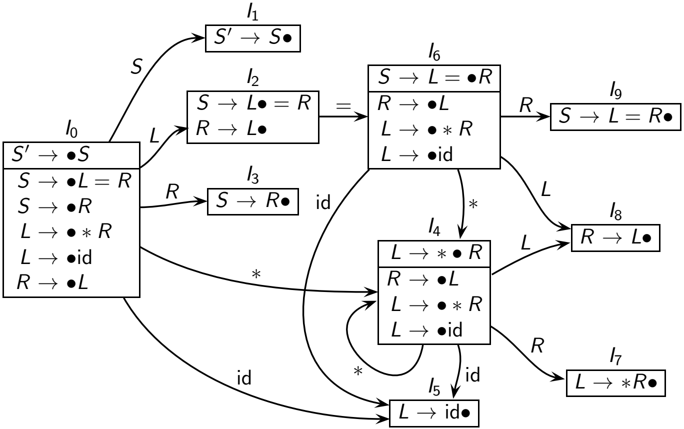
LR(0) Item Sets for Pointer Assignment Grammar



Shift reduce conflict on $=$

$\text{FOLLOW}(R) = \{\$, =\}$

Limitation of SLR(1) Parsing



I_0
$S' \rightarrow \bullet S$
$S \rightarrow \bullet L = R$
$S \rightarrow \bullet R$
$L \rightarrow \bullet * R$
$L \rightarrow \bullet \text{id}$
$R \rightarrow \bullet L$

I_1
$S' \rightarrow S \bullet$

I_2
$S \rightarrow L \bullet = R$
$R \rightarrow L \bullet$

I_3
$S \rightarrow R \bullet$

I_4
$L \rightarrow * \bullet R$
$R \rightarrow \bullet L$
$L \rightarrow \bullet * R$
$L \rightarrow \bullet \text{id}$

I_5
$L \rightarrow \text{id} \bullet$

I_6
$S \rightarrow L = \bullet R$
$R \rightarrow \bullet L$
$L \rightarrow \bullet * R$
$L \rightarrow \bullet \text{id}$

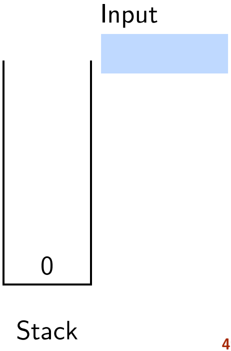
I_7
$L \rightarrow * R \bullet$

I_8
$R \rightarrow L \bullet$

I_9
$S \rightarrow L = R \bullet$

1	S	$\rightarrow$	$L = R$
2	S	$\rightarrow$	R
3	L	$\rightarrow$	$*R$
4	L	$\rightarrow$	id
5	R	$\rightarrow$	L

	FOLLOW
S'	$\{\$\}$
S	$\{\$\}$
R	$\{=, \$\}$
L	$\{=, \$\}$

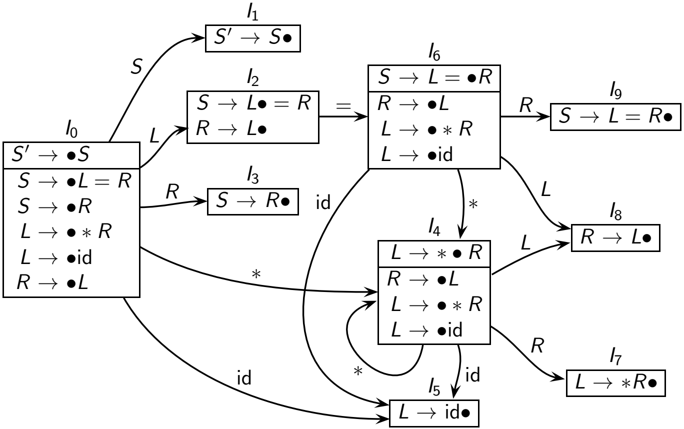


Input

0

Stack

Limitation of SLR(1) Parsing



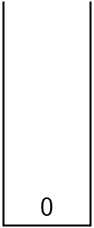
1	S	$\to$	$L = R$
2	S	$\to$	R
3	L	$\to$	$*R$
4	L	$\to$	id
5	R	$\to$	L

	FOLLOW
S'	$\{\$\}$
S	$\{\$\}$
R	$\{=, \$\}$
L	$\{=, \$\}$

Shift 5

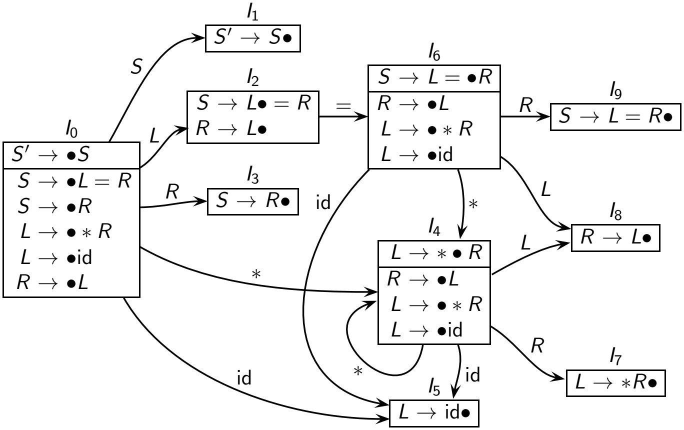
Input

id = id$



Stack

Limitation of SLR(1) Parsing



1	S	$\rightarrow$	$L = R$
2	S	$\rightarrow$	R
3	L	$\rightarrow$	$*R$
4	L	$\rightarrow$	id
5	R	$\rightarrow$	L

	FOLLOW
S'	$\{\$\}$
S	$\{\$\}$
R	$\{=, \$\}$
L	$\{=, \$\}$

Reduce by 4

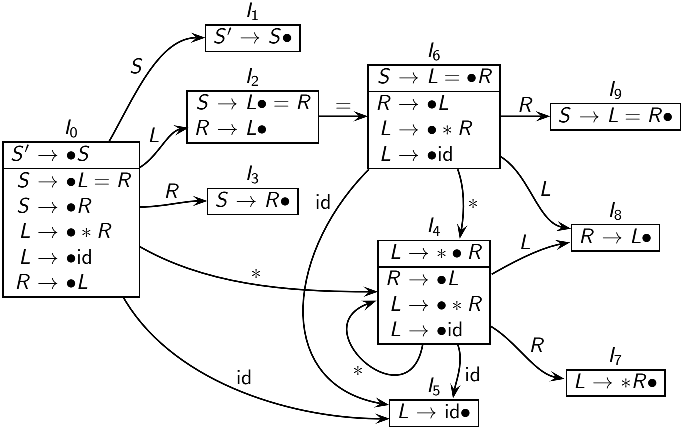
Input

$= \text{id}\$$

5
id
0

Stack

Limitation of SLR(1) Parsing



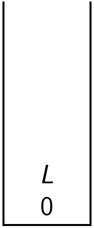
1	S	$\to$	$L = R$
2	S	$\to$	R
3	L	$\to$	$*R$
4	L	$\to$	id
5	R	$\to$	L

	FOLLOW
S'	$\{\$\}$
S	$\{\$\}$
R	$\{=, \$\}$
L	$\{=, \$\}$

Cover by 2

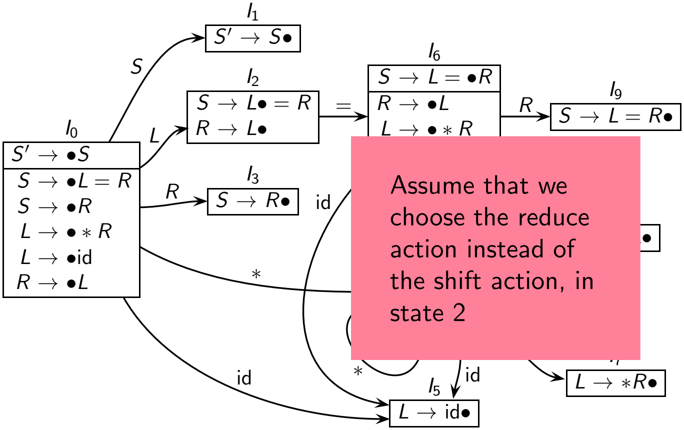
Input

$= \text{id}\$$



Stack

Limitation of SLR(1) Parsing



Assume that we choose the reduce action instead of the shift action, in state 2

Reduce by 5

Input

$= \text{id}\$$

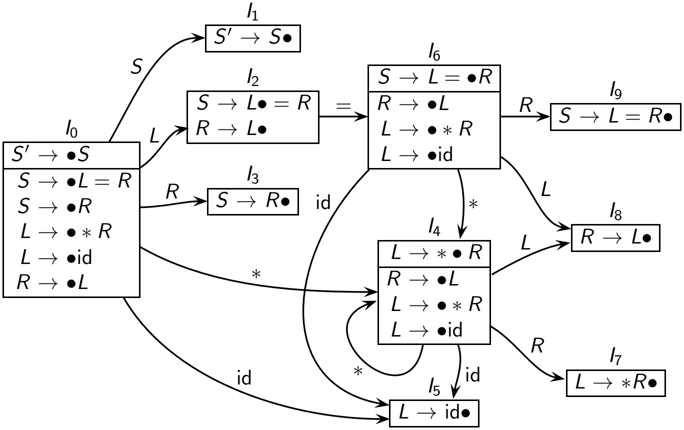
2
L
0

Stack

1	S	$\to$	$L = R$
2	S	$\to$	R
3	L	$\to$	$*R$
4	L	$\to$	id
5	R	$\to$	L

	FOLLOW
S'	$\{\$\}$
S	$\{\$\}$
R	$\{=, \$\}$
L	$\{=, \$\}$

Limitation of SLR(1) Parsing



I_0: $S' \to \bullet S$, $S \to \bullet L = R$, $S \to \bullet R$, $L \to \bullet * R$, $L \to \bullet \text{id}$, $R \to \bullet L$

I_1: $S' \to S\bullet$

I_2: $S \to L\bullet = R$, $R \to L\bullet$

I_3: $S \to R\bullet$

I_4: $L \to * \bullet R$, $R \to \bullet L$, $L \to \bullet * R$, $L \to \bullet \text{id}$

I_5: $L \to \text{id}\bullet$

I_6: $S \to L = \bullet R$, $R \to \bullet L$, $L \to \bullet * R$, $L \to \bullet \text{id}$

I_7: $L \to *R\bullet$

I_8: $R \to L\bullet$

I_9: $S \to L = R\bullet$

Cover by 3

Input

$= \text{id}\$$

R

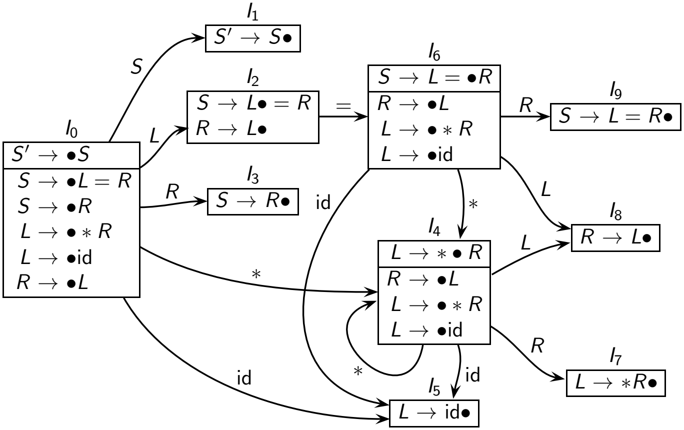
0

Stack

1	S	$\to$	$L = R$
2	S	$\to$	R
3	L	$\to$	$*R$
4	L	$\to$	id
5	R	$\to$	L

	FOLLOW
S'	$\{\$\}$
S	$\{\$\}$
R	$\{=, \$\}$
L	$\{=, \$\}$

Limitation of SLR(1) Parsing



Error

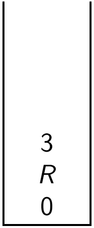
No action on $=$

1	S	$\to$	$L = R$
2	S	$\to$	R
3	L	$\to$	$*R$
4	L	$\to$	id
5	R	$\to$	L

	FOLLOW
S'	$\{\$\}$
S	$\{\$\}$
R	$\{=, \$\}$
L	$\{=, \$\}$

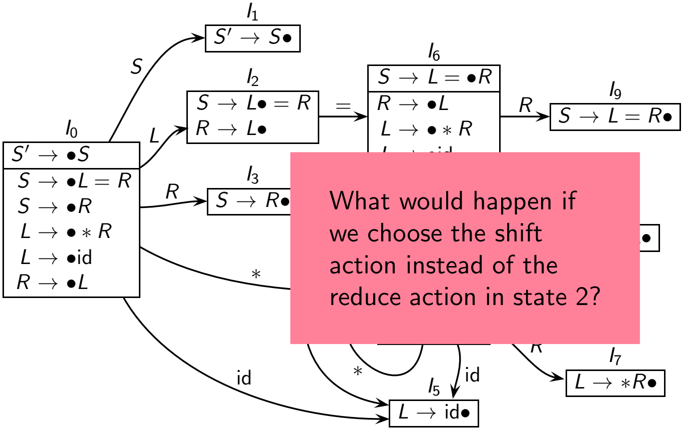
Input

$= \text{id}\$$



Stack

Limitation of SLR(1) Parsing



What would happen if we choose the shift action instead of the reduce action in state 2?

1 $S \to L = R$
2 $S \to R$
3 $L \to *R$
4 $L \to \text{id}$
5 $R \to L$

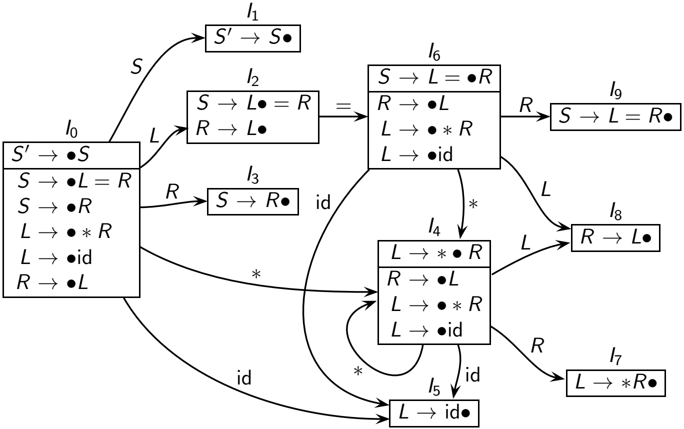
	FOLLOW
S'	$\{\$\}$
S	$\{\$\}$
R	$\{=, \$\}$
L	$\{=, \$\}$

Input
id = id\$

0

Stack

Limitation of SLR(1) Parsing



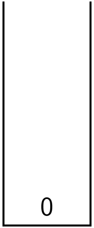
Shift 5

1 $S \to L = R$
2 $S \to R$
3 $L \to *R$
4 $L \to \text{id}$
5 $R \to L$

	FOLLOW
S'	$\{\$\}$
S	$\{\$\}$
R	$\{=, \$\}$
L	$\{=, \$\}$

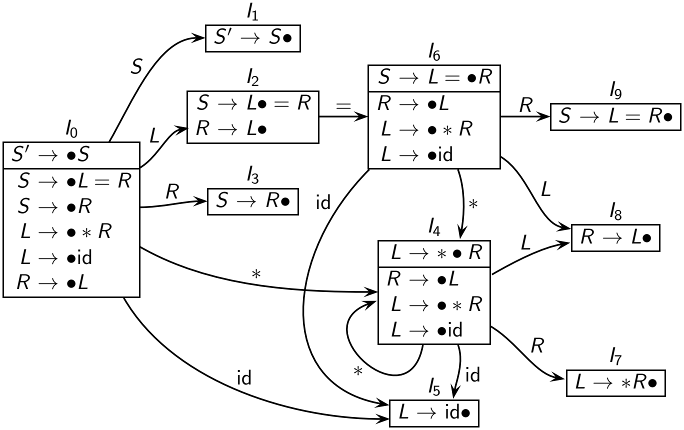
Input

$\text{id} = \text{id}\$$



Stack

Limitation of SLR(1) Parsing



1	S	$\to$	$L = R$
2	S	$\to$	R
3	L	$\to$	$*R$
4	L	$\to$	id
5	R	$\to$	L

	FOLLOW
S'	$\{\$\}$
S	$\{\$\}$
R	$\{=, \$\}$
L	$\{=, \$\}$

Reduce by 4

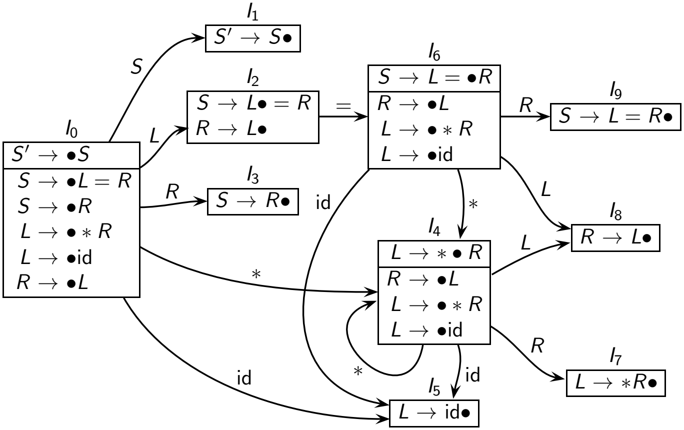
Input

$= \text{id}\$$

5
id
0

Stack

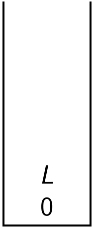
Limitation of SLR(1) Parsing



Cover by 2

Input

$= \text{id}\$$

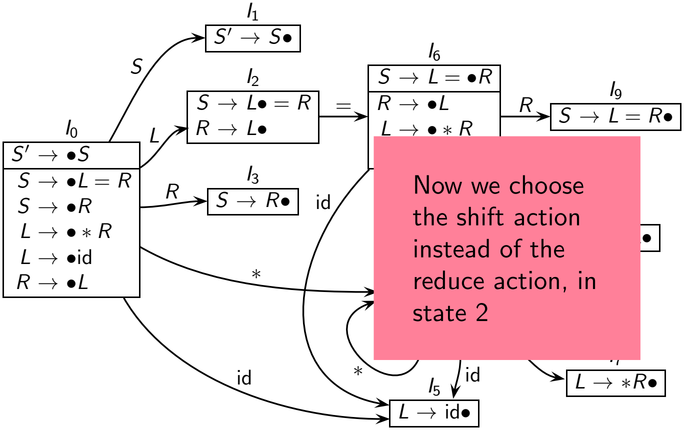


Stack

1 $S \to L = R$
2 $S \to R$
3 $L \to *R$
4 $L \to \text{id}$
5 $R \to L$

	FOLLOW
S'	$\{\$\}$
S	$\{\$\}$
R	$\{=, \$\}$
L	$\{=, \$\}$

Limitation of SLR(1) Parsing



Now we choose the shift action instead of the reduce action, in state 2

shift 6

Input

$= \text{id}\$$

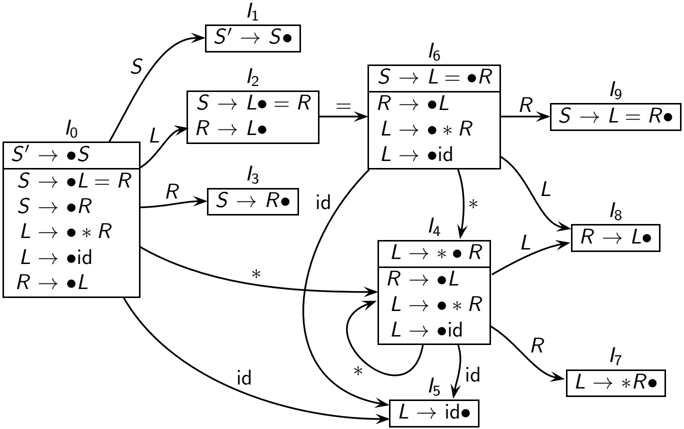
2
L
0

Stack

1	S	$\to$	$L = R$
2	S	$\to$	R
3	L	$\to$	$*R$
4	L	$\to$	id
5	R	$\to$	L

	FOLLOW
S'	$\{\$\}$
S	$\{\$\}$
R	$\{=, \$\}$
L	$\{=, \$\}$

Limitation of SLR(1) Parsing



Shift 5

Input

id\$

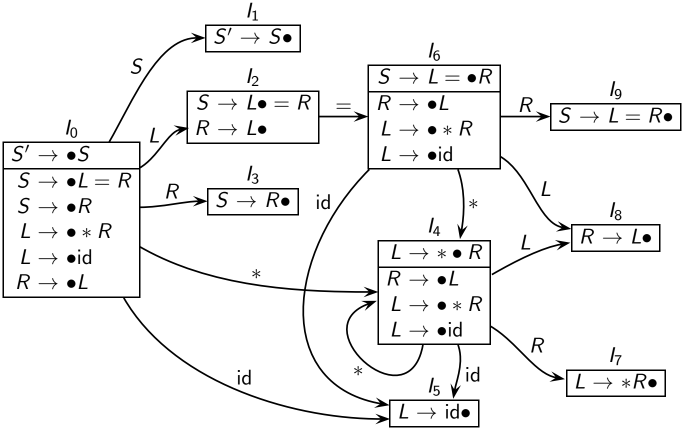
6
=
5
L
0

Stack

1 $S \rightarrow L = R$
2 $S \rightarrow R$
3 $L \rightarrow * R$
4 $L \rightarrow \text{id}$
5 $R \rightarrow L$

	FOLLOW
S'	$\{\$\}$
S	$\{\$\}$
R	$\{=, \$\}$
L	$\{=, \$\}$

Limitation of SLR(1) Parsing



Reduce by 4

1	S	$\to$	$L = R$
2	S	$\to$	R
3	L	$\to$	$*R$
4	L	$\to$	id
5	R	$\to$	L

	FOLLOW
S'	$\{\$\}$
S	$\{\$\}$
R	$\{=, \$\}$
L	$\{=, \$\}$

Input

$

5
id
6
=
5
L
0

Stack

IIT Bombay
cs302: Implementation of Programming Languages

Topic:

Syntax Analysis

Section:

Grammars, Derivations, and Parse Trees

Shift Reduce Parsing

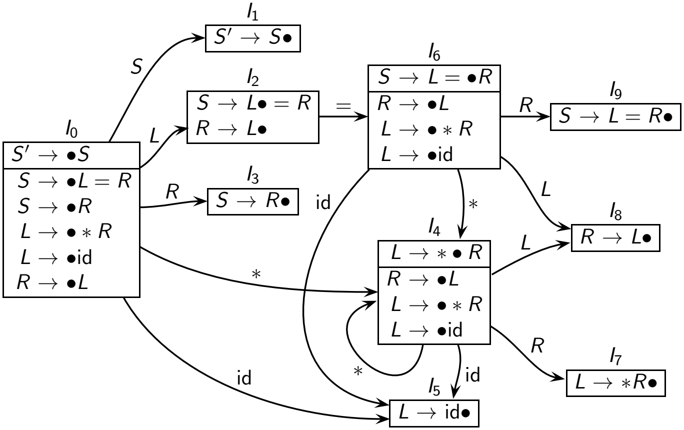
SLR(1) Parsing

Conceptual Issues in Parsing

CLR(1) Parsing

LALR(1) Parsing

Limitation of SLR(1) Parsing



I_0: $S' \to \bullet S$; $S \to \bullet L = R$; $S \to \bullet R$; $L \to \bullet * R$; $L \to \bullet \text{id}$; $R \to \bullet L$

I_1: $S' \to S \bullet$

I_2: $S \to L \bullet = R$; $R \to L \bullet$

I_3: $S \to R \bullet$

I_4: $L \to * \bullet R$; $R \to \bullet L$; $L \to \bullet * R$; $L \to \bullet \text{id}$

I_5: $L \to \text{id} \bullet$

I_6: $S \to L = \bullet R$; $R \to \bullet L$; $L \to \bullet * R$; $L \to \bullet \text{id}$

I_7: $L \to * R \bullet$

I_8: $R \to L \bullet$

I_9: $S \to L = R \bullet$

1	S	$\to$	$L = R$
2	S	$\to$	R
3	L	$\to$	$*R$
4	L	$\to$	id
5	R	$\to$	L

	FOLLOW
S'	$\{\$\}$
S	$\{\$\}$
R	$\{=, \$\}$
L	$\{=, \$\}$

Cover by 8

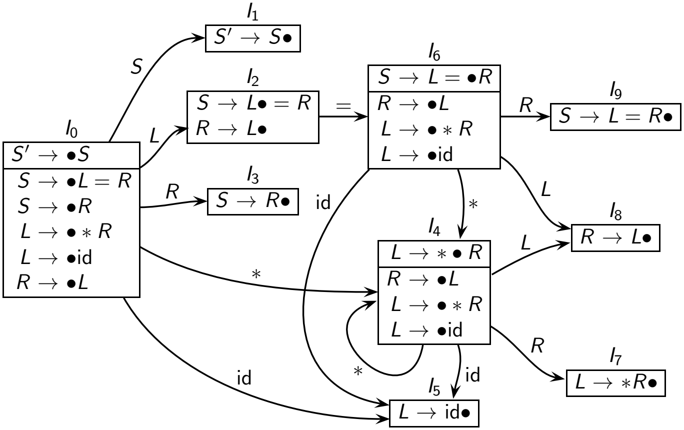
Input

$

L
6
$=$
5
L
0

Stack

Limitation of SLR(1) Parsing



I_0: $S' \to \bullet S$, $S \to \bullet L = R$, $S \to \bullet R$, $L \to \bullet * R$, $L \to \bullet \text{id}$, $R \to \bullet L$

I_1: $S' \to S\bullet$

I_2: $S \to L\bullet = R$, $R \to L\bullet$

I_3: $S \to R\bullet$

I_4: $L \to * \bullet R$, $R \to \bullet L$, $L \to \bullet * R$, $L \to \bullet \text{id}$

I_5: $L \to \text{id}\bullet$

I_6: $S \to L = \bullet R$, $R \to \bullet L$, $L \to \bullet * R$, $L \to \bullet \text{id}$

I_7: $L \to *R\bullet$

I_8: $R \to L\bullet$

I_9: $S \to L = R\bullet$

1	S	$\to$	$L = R$
2	S	$\to$	R
3	L	$\to$	$*R$
4	L	$\to$	id
5	R	$\to$	L

	FOLLOW
S'	$\{\$\}$
S	$\{\$\}$
R	$\{=, \$\}$
L	$\{=, \$\}$

Reduce by 5

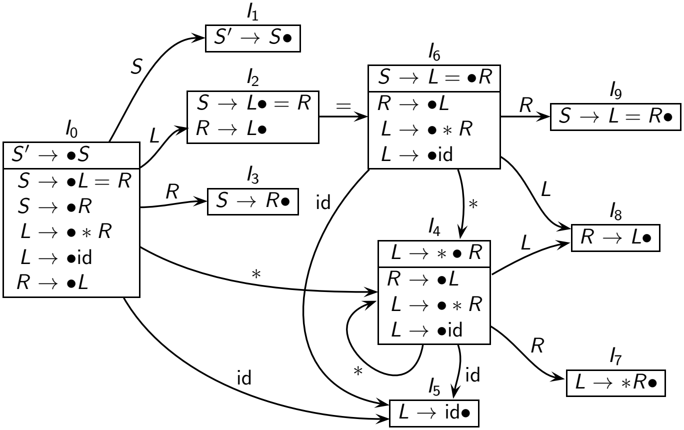
Input

\$

8
L
6
$=$
5
L
0

Stack

Limitation of SLR(1) Parsing



1	S	$\to$	$L = R$
2	S	$\to$	R
3	L	$\to$	$*R$
4	L	$\to$	id
5	R	$\to$	L

	FOLLOW
S'	$\{\$\}$
S	$\{\$\}$
R	$\{=, \$\}$
L	$\{=, \$\}$

Cover by 9

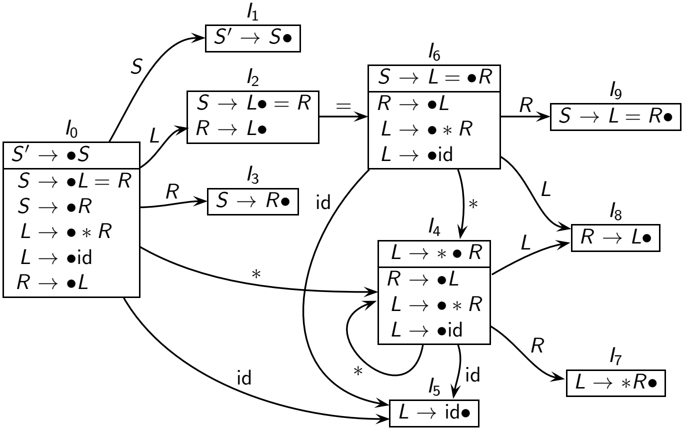
Input

$

R
6
$=$
5
L
0

Stack

Limitation of SLR(1) Parsing



1 $S \to L = R$
2 $S \to R$
3 $L \to *R$
4 $L \to \text{id}$
5 $R \to L$

	FOLLOW
S'	$\{\$\}$
S	$\{\$\}$
R	$\{=, \$\}$
L	$\{=, \$\}$

Reduce by 1

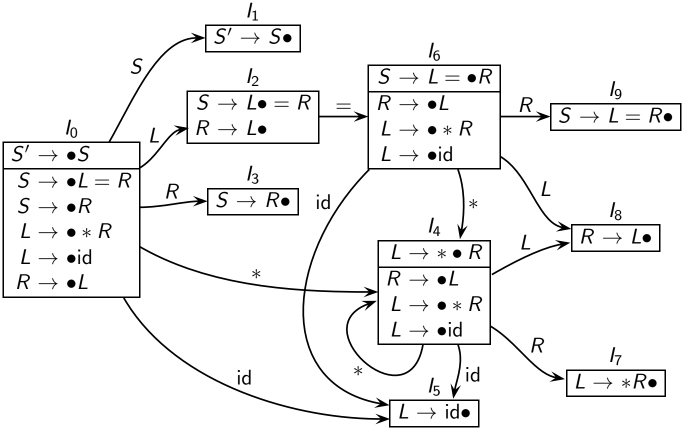
Input

$

9
R
6
$=$
5
L
0

Stack

Limitation of SLR(1) Parsing



I_0: $S' \rightarrow \bullet S$; $S \rightarrow \bullet L = R$; $S \rightarrow \bullet R$; $L \rightarrow \bullet * R$; $L \rightarrow \bullet \text{id}$; $R \rightarrow \bullet L$

I_1: $S' \rightarrow S \bullet$

I_2: $S \rightarrow L \bullet = R$; $R \rightarrow L \bullet$

I_3: $S \rightarrow R \bullet$

I_4: $L \rightarrow * \bullet R$; $R \rightarrow \bullet L$; $L \rightarrow \bullet * R$; $L \rightarrow \bullet \text{id}$

I_5: $L \rightarrow \text{id} \bullet$

I_6: $S \rightarrow L = \bullet R$; $R \rightarrow \bullet L$; $L \rightarrow \bullet * R$; $L \rightarrow \bullet \text{id}$

I_7: $L \rightarrow * R \bullet$

I_8: $R \rightarrow L \bullet$

I_9: $S \rightarrow L = R \bullet$

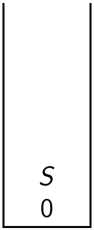
1	S	$\rightarrow$	$L = R$	
2	S	$\rightarrow$	R	
3	L	$\rightarrow$	$*R$	
4	L	$\rightarrow$	id	
5	R	$\rightarrow$	L	

	FOLLOW
S'	$\{\$\}$
S	$\{\$\}$
R	$\{=, \$\}$
L	$\{=, \$\}$

Cover by 1

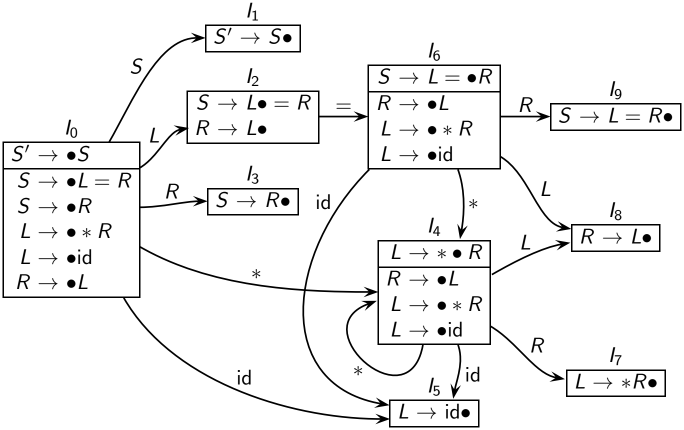
Input

$



Stack

Limitation of SLR(1) Parsing



1	S	$\to$	$L = R$
2	S	$\to$	R
3	L	$\to$	$*R$
4	L	$\to$	id
5	R	$\to$	L

	FOLLOW
S'	$\{\$\}$
S	$\{\$\}$
R	$\{=, \$\}$
L	$\{=, \$\}$

Accept

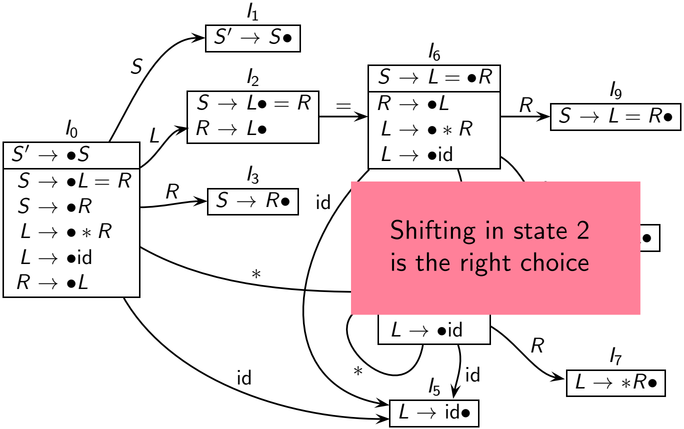
Input

$

1
S
0

Stack

Limitation of SLR(1) Parsing



Shifting in state 2
is the right choice

Accept

Input

$

1
S
0

Stack

1 $S \to L = R$
2 $S \to R$
3 $L \to *R$
4 $L \to \text{id}$
5 $R \to L$

	FOLLOW
S'	$\{\$\}$
S	$\{\$\}$
R	$\{=, \$\}$
L	$\{=, \$\}$

Limitation of SLR(1) Parsing: Use of FOLLOW Information

- Let FOLLOW$(A) = \{b, c\}$. Then b may follow A in some right sentential forms whereas in some other right sentential form, c may follow A

  A symbol in follow set need not follow A in *every* right sentential form

Limitation of SLR(1) Parsing: Use of FOLLOW Information

- Let FOLLOW(A) $= \{b, c\}$. Then b may follow A in some right sentential forms whereas in some other right sentential form, c may follow A

  A symbol in follow set need not follow A in *every* right sentential form
- We should declare handle $A \rightarrow \alpha$ in a viable prefix γ only if the follow symbols actually follows A in the right sentential form containing γ

Limitation of SLR(1) Parsing: Use of FOLLOW Information

- Let $\text{FOLLOW}(A) = \{b, c\}$. Then b may follow A in some right sentential forms whereas in some other right sentential form, c may follow A

  A symbol in follow set need not follow A in *every* right sentential form
- We should declare handle $A \rightarrow \alpha$ in a viable prefix γ only if the follow symbols actually follows A in the right sentential form containing γ
- In our grammar, there is no right sentential form with a prefix '$R =$'
  - Since we need '$=$' in our right sentential form, consider $S \overset{rm}{\Rightarrow} L = R$
  - L can derive either id or $*R$ but not R

$$\begin{array}{lcl} S & \rightarrow & L = R \mid R \\ L & \rightarrow & *R \mid \text{id} \\ R & \rightarrow & L \end{array}$$

'$=$' is in $\text{FOLLOW}(R)$ only for the right sentential forms that begin with a '$*$'

LR(1) Item Sets

Two changes from LR(0) construction

- Items are of the form $A \to \alpha \bullet \beta, a$ consisting of
  - the *core* $A \to \alpha \bullet \beta$ and
  - the *lookahead* a

  If S is the start symbol, then I_0 contains $S' \to \bullet S, \$$

- Closure of an item $A \to \alpha \bullet B\beta, a$ contains the items of the form $B \to \bullet\gamma, \text{FIRST}(\beta a)$

Transition of an item $A \to \alpha \bullet B\beta, a$ on B gives an item $A \to \alpha B \bullet \beta, a$

The lookahead does not change during a transition

LR(1) Item Sets

Two changes from LR(0) construction

- Items are of the form $A \rightarrow \alpha \bullet \beta, a$ consisting of
  - the *core* $A \rightarrow \alpha \bullet \beta$ and
  - th

If S i

- Closu

of th

The goal is to compute different subsets of FOLLOW(A) for $A \rightarrow \alpha$ in different right sentential forms

Since the construction of sets of items creates a DFA to recognize all viable prefixes, the subsets of FOLLOW can be computed for the productions in sets of items

Transition of an item $A \rightarrow \alpha \bullet B\beta, a$ on B gives an item $A \rightarrow \alpha B \bullet \beta, a$

The lookahead does not change during a transition

LR(1) Item Sets

Two changes from LR(0) construction

- Items are of the form $A \rightarrow \alpha \bullet \beta, a$ consisting of
  - the *core* $A \rightarrow \alpha \bullet \beta$ and
  - the *lookahead* a

  If S is the start symbol, then I_0 contains $S' \rightarrow \bullet S, \$$
- Closure of an item $A \rightarrow \alpha \bullet B\beta, a$ contains the items of the form $B \rightarrow \bullet\gamma, \text{FIRST}(\beta a)$

Transition of an item $A \rightarrow \alpha \bullet B\beta, a$ on B gives an item $A \rightarrow \alpha B \bullet \beta, a$

The lookahead does not change during a transition

$S' \rightarrow \bullet S, \quad \$$

I_0

$$\begin{aligned} S &\rightarrow L = R \mid R \\ L &\rightarrow *R \mid \text{id} \\ R &\rightarrow L \end{aligned}$$

LR(1) Item Sets

Two changes from LR(0) construction

- Items are of the form $A \rightarrow \alpha \bullet \beta, a$ consisting of
  - the *core* $A \rightarrow \alpha \bullet \beta$ and
  - the *lookahead* a

  If S is the start symbol, then I_0 contains $S' \rightarrow \bullet S, \$$
- Closure of an item $A \rightarrow \alpha \bullet B\beta, a$ contains the items of the form $B \rightarrow \bullet\gamma, \text{FIRST}(\beta a)$

Transition of an item $A \rightarrow \alpha \bullet B\beta, a$ on B gives an item $A \rightarrow \alpha B \bullet \beta, a$

The lookahead does not change during a transition

$$\begin{array}{|l|}\hline S' \rightarrow \bullet S, \quad \$ \\ \hline S \rightarrow \bullet L = R, \ \$ \\ S \rightarrow \bullet R, \quad \$ \\ \\ \hline \end{array} \quad I_0$$

$$\begin{array}{lcl} S & \rightarrow & L = R \mid R \\ L & \rightarrow & *R \mid \text{id} \\ R & \rightarrow & L \end{array}$$

LR(1) Item Sets

Two changes from LR(0) construction

- Items are of the form $A \rightarrow \alpha \bullet \beta, a$ consisting of
    - the *core* $A \rightarrow \alpha \bullet \beta$ and
    - the *lookahead* a

  If S is the start symbol, then I_0 contains $S' \rightarrow \bullet S, \$$
- Closure of an item $A \rightarrow \alpha \bullet B\beta, a$ contains the items of the form $B \rightarrow \bullet\gamma, \text{FIRST}(\beta a)$

Transition of an item $A \rightarrow \alpha \bullet B\beta, a$ on B gives an item $A \rightarrow \alpha B \bullet \beta, a$

The lookahead does not change during a transition

$$\begin{array}{|lll|}\hline S' & \rightarrow \bullet S, & \$ \\ \hline S & \rightarrow \bullet L = R, & \$ \\ S & \rightarrow \bullet R, & \$ \\ L & \rightarrow \bullet * R, & = \\ L & \rightarrow \bullet \text{id}, & = \\ & & \\ \hline \end{array} \quad I_0$$

$$\begin{array}{lll} S & \rightarrow & L = R \mid R \\ L & \rightarrow & *R \mid \text{id} \\ R & \rightarrow & L \end{array}$$

LR(1) Item Sets

Two changes from LR(0) construction

- Items are of the form $A \rightarrow \alpha \bullet \beta, a$ consisting of
  - the *core* $A \rightarrow \alpha \bullet \beta$ and
  - the *lookahead* a

  If S is the start symbol, then I_0 contains $S' \rightarrow \bullet S, \$$
- Closure of an item $A \rightarrow \alpha \bullet B\beta, a$ contains the items of the form $B \rightarrow \bullet\gamma, \text{FIRST}(\beta a)$

Transition of an item $A \rightarrow \alpha \bullet B\beta, a$ on B gives an item $A \rightarrow \alpha B \bullet \beta, a$

The lookahead does not change during a transition

$$\begin{array}{|lll|}
\hline
S' \rightarrow \bullet S, & & \$ \\
\hline
S \rightarrow \bullet L = R, & & \$ \\
S \rightarrow \bullet R, & & \$ \\
L \rightarrow \bullet * R, & & = \\
L \rightarrow \bullet \text{id}, & & = \\
R \rightarrow \bullet L, & & \$ \\
L \rightarrow \bullet * R, & & \$ \\
\hline
\end{array} \quad I_0$$

$$\begin{array}{lcl}
S & \rightarrow & L = R \mid R \\
L & \rightarrow & *R \mid \text{id} \\
R & \rightarrow & L
\end{array}$$

LR(1) Item Sets

Two changes from LR(0) construction

- Items are of the form $A \rightarrow \alpha \bullet \beta, a$ consisting of
    - the *core* $A \rightarrow \alpha \bullet \beta$ and
    - the *lookahead* a

  If S is the start symbol, then I_0 contains $S' \rightarrow \bullet S, \$$
- Closure of an item $A \rightarrow \alpha \bullet B\beta, a$ contains the items of the form $B \rightarrow \bullet\gamma, \text{FIRST}(\beta a)$

Transition of an item $A \rightarrow \alpha \bullet B\beta, a$ on B gives an item $A \rightarrow \alpha B \bullet \beta, a$

The lookahead does not change during a transition

$S' \rightarrow \bullet S,$	$\$$	I_0
$S \rightarrow \bullet L = R,$	$\$$	
$S \rightarrow \bullet R,$	$\$$	
$L \rightarrow \bullet * R,$	$=$	
$L \rightarrow \bullet \text{id},$	$=$	
$R \rightarrow \bullet L,$	$\$$	
$L \rightarrow \bullet * R,$	$\$$	
$L \rightarrow \bullet \text{id},$	$\$$	

$$\begin{array}{lcl} S & \rightarrow & L = R \mid R \\ L & \rightarrow & *R \mid \text{id} \\ R & \rightarrow & L \end{array}$$

LR(1) Item Sets

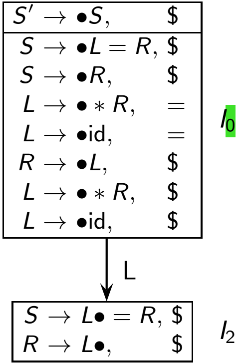
Two changes from LR(0) construction

- Items are of the form $A \rightarrow \alpha \bullet \beta, a$ consisting of
  - the *core* $A \rightarrow \alpha \bullet \beta$ and
  - the *lookahead* a

  If S is the start symbol, then I_0 contains $S' \rightarrow \bullet S, \$$
- Closure of an item $A \rightarrow \alpha \bullet B\beta, a$ contains the items of the form $B \rightarrow \bullet\gamma, \text{FIRST}(\beta a)$

Transition of an item $A \rightarrow \alpha \bullet B\beta, a$ on B gives an item $A \rightarrow \alpha B \bullet \beta, a$

The lookahead does not change during a transition



$S' \rightarrow \bullet S, \quad \$$

$S \rightarrow \bullet L = R, \$$

$S \rightarrow \bullet R, \quad \$$

$L \rightarrow \bullet * R, \quad =$

$L \rightarrow \bullet \text{id}, \quad =$

$R \rightarrow \bullet L, \quad \$$

$L \rightarrow \bullet * R, \quad \$$

$L \rightarrow \bullet \text{id}, \quad \$$

$S \rightarrow L \bullet = R, \$$

$R \rightarrow L \bullet, \quad \$$

$$
\begin{array}{lcl}
S & \rightarrow & L = R \mid R \\
L & \rightarrow & *R \mid \text{id} \\
R & \rightarrow & L
\end{array}
$$

LR(1) Item Sets

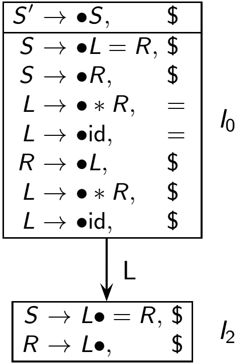
Two changes from LR(0) construction

- Items are of the form $A \rightarrow \alpha \bullet \beta, a$ consisting of
    - the *core* $A \rightarrow \alpha \bullet \beta$ and
    - the *lookahead* a

  If S is the start symbol, then I_0 contains $S' \rightarrow \bullet S, \$$
- Closure of an item $A \rightarrow \alpha \bullet B\beta, a$ contains the items of the form $B \rightarrow \bullet\gamma, \text{FIRST}(\beta a)$

Transition of an item $A \rightarrow \alpha \bullet B\beta, a$ on B gives an item $A \rightarrow \alpha B \bullet \beta, a$

The lookahead does not change during a transition



$S' \rightarrow \bullet S, \quad \$$
$S \rightarrow \bullet L = R, \quad \$$
$S \rightarrow \bullet R, \quad \$$
$L \rightarrow \bullet * R, \quad =$
$L \rightarrow \bullet \text{id}, \quad =$
$R \rightarrow \bullet L, \quad \$$
$L \rightarrow \bullet * R, \quad \$$
$L \rightarrow \bullet \text{id}, \quad \$$

I_0

L

$S \rightarrow L \bullet = R, \quad \$$
$R \rightarrow L \bullet, \quad \$$

I_2

Reduction by $R \rightarrow L\bullet$ only on \$ and not on $=$

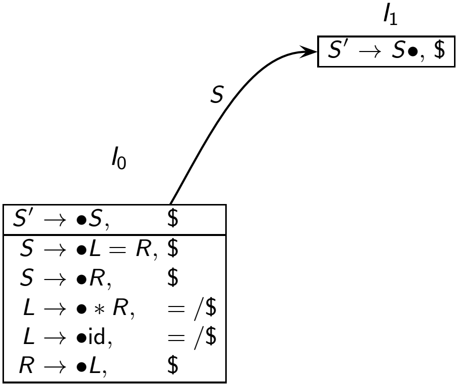
No shift reduce conflict

LR(1) Sets of Items for Pointer Assignment Grammar

I_0

$S' \rightarrow \bullet S,$	$\$$
$S \rightarrow \bullet L = R,$	$\$$
$S \rightarrow \bullet R,$	$\$$
$L \rightarrow \bullet * R,$	$= /\$$
$L \rightarrow \bullet \text{id},$	$= /\$$
$R \rightarrow \bullet L,$	$\$$

LR(1) Sets of Items for Pointer Assignment Grammar



I_0

$S' \rightarrow \bullet S, \quad \$$

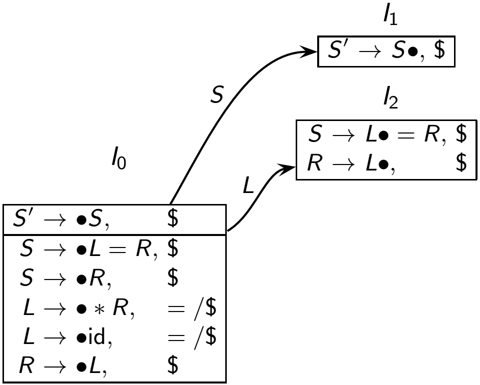
$S \rightarrow \bullet L = R, \quad \$$
$S \rightarrow \bullet R, \quad \$$
$L \rightarrow \bullet * R, \quad = / \$$
$L \rightarrow \bullet \text{id}, \quad = / \$$
$R \rightarrow \bullet L, \quad \$$

$\xrightarrow{S}$

I_1

$S' \rightarrow S \bullet, \$$

LR(1) Sets of Items for Pointer Assignment Grammar



I_0

$S' \to \bullet S,\quad \$$

$S \to \bullet L = R,\ \$$

$S \to \bullet R,\quad \$$

$L \to \bullet * R,\quad = /\$$

$L \to \bullet \text{id},\quad = /\$$

$R \to \bullet L,\quad \$$

I_1

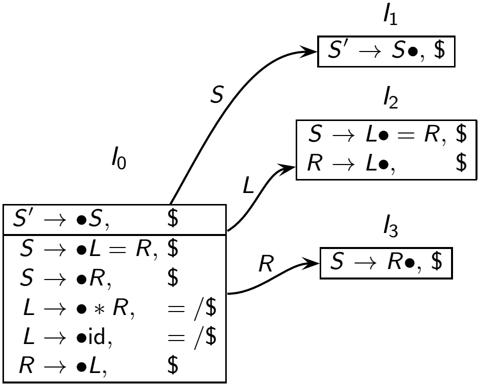
$S' \to S\bullet,\ \$$

I_2

$S \to L\bullet = R,\ \$$

$R \to L\bullet,\quad \$$

LR(1) Sets of Items for Pointer Assignment Grammar



I_0

$S' \rightarrow \bullet S, \quad \$$

$S \rightarrow \bullet L = R, \ \$$

$S \rightarrow \bullet R, \quad \$$

$L \rightarrow \bullet * R, \quad =/\$$

$L \rightarrow \bullet \text{id}, \quad =/\$$

$R \rightarrow \bullet L, \quad \$$

I_1

$S' \rightarrow S \bullet, \ \$$

I_2

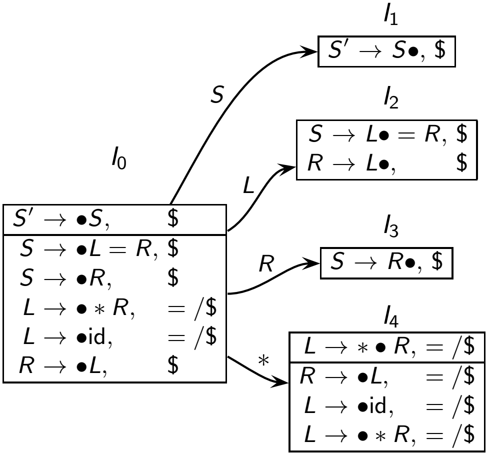
$S \rightarrow L \bullet = R, \ \$$

$R \rightarrow L \bullet, \quad \$$

I_3

$S \rightarrow R \bullet, \ \$$

LR(1) Sets of Items for Pointer Assignment Grammar



I_0

$S' \rightarrow \bullet S, \quad \$$

$S \rightarrow \bullet L = R, \quad \$$

$S \rightarrow \bullet R, \quad \$$

$L \rightarrow \bullet * R, \quad = /\$$

$L \rightarrow \bullet \text{id}, \quad = /\$$

$R \rightarrow \bullet L, \quad \$$

I_1

$S' \rightarrow S\bullet, \ \$$

I_2

$S \rightarrow L\bullet = R, \ \$$

$R \rightarrow L\bullet, \quad \$$

I_3

$S \rightarrow R\bullet, \ \$$

I_4

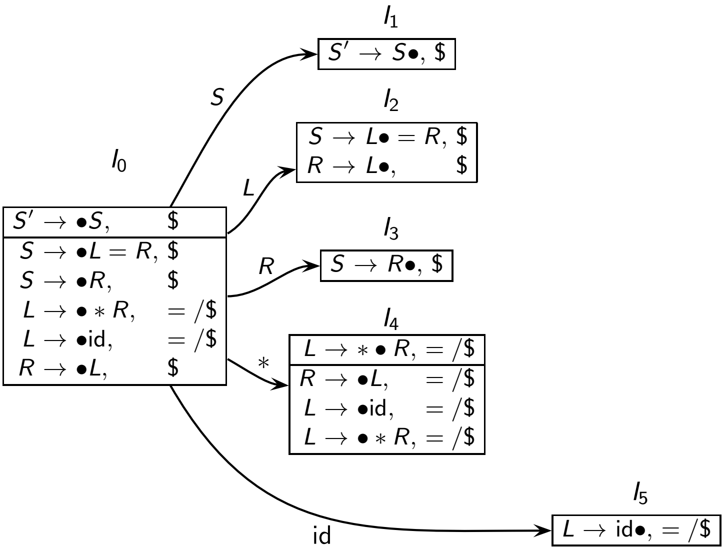
$L \rightarrow * \bullet R, \ = /\$$

$R \rightarrow \bullet L, \quad = /\$$

$L \rightarrow \bullet \text{id}, \quad = /\$$

$L \rightarrow \bullet * R, \ = /\$$

LR(1) Sets of Items for Pointer Assignment Grammar



I_0

$S' \rightarrow \bullet S, \quad \$$

$S \rightarrow \bullet L = R, \quad \$$

$S \rightarrow \bullet R, \quad \$$

$L \rightarrow \bullet * R, \quad = /\$$

$L \rightarrow \bullet \text{id}, \quad = /\$$

$R \rightarrow \bullet L, \quad \$$

I_1

$S' \rightarrow S\bullet, \ \$$

I_2

$S \rightarrow L\bullet = R, \ \$$

$R \rightarrow L\bullet, \quad \$$

I_3

$S \rightarrow R\bullet, \ \$$

I_4

$L \rightarrow * \bullet R, \ = /\$$

$R \rightarrow \bullet L, \quad = /\$$

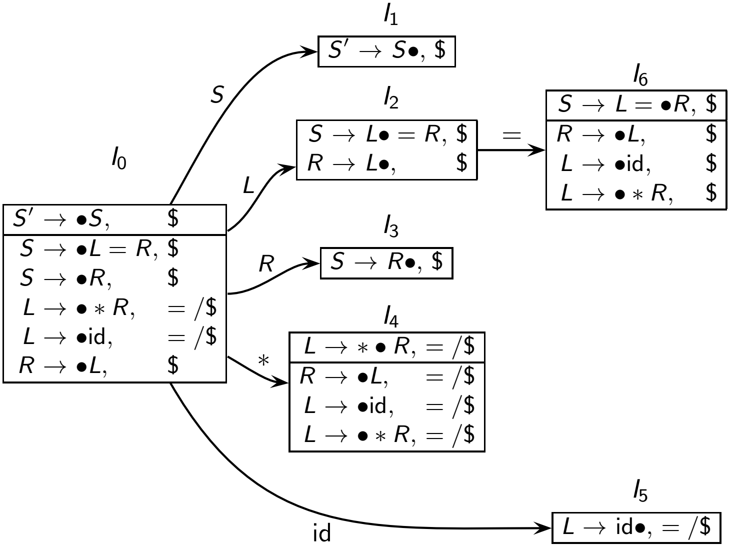
$L \rightarrow \bullet \text{id}, \quad = /\$$

$L \rightarrow \bullet * R, \ = /\$$

I_5

$L \rightarrow \text{id}\bullet, \ = /\$$

LR(1) Sets of Items for Pointer Assignment Grammar



I_0

$S' \to \bullet S, \quad \$$

$S \to \bullet L = R, \quad \$$

$S \to \bullet R, \quad \$$

$L \to \bullet * R, \quad = /\$$

$L \to \bullet \text{id}, \quad = /\$$

$R \to \bullet L, \quad \$$

I_1

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

I_2

$S \to L \bullet = R, \ \$$

$R \to L \bullet, \quad \$$

I_3

$S \to R \bullet, \ \$$

I_4

$L \to * \bullet R, \ = /\$$

$R \to \bullet L, \quad = /\$$

$L \to \bullet \text{id}, \quad = /\$$

$L \to \bullet * R, \ = /\$$

I_5

$L \to \text{id} \bullet, \ = /\$$

I_6

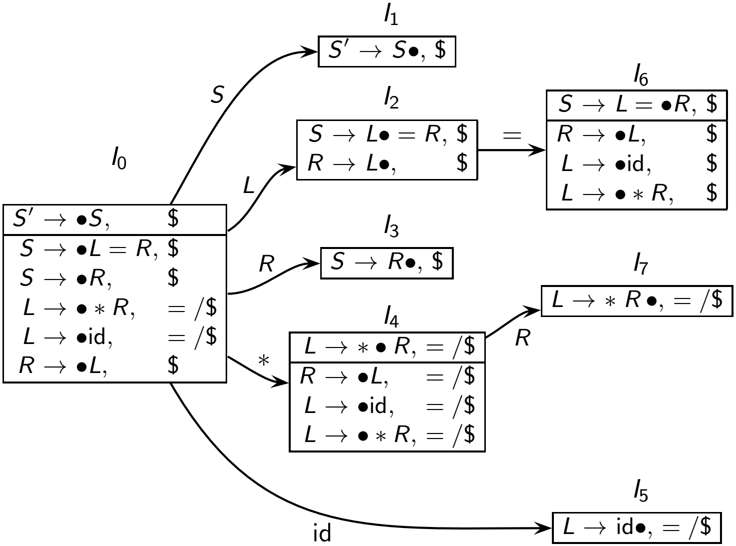
$S \to L = \bullet R, \ \$$

$R \to \bullet L, \quad \$$

$L \to \bullet \text{id}, \quad \$$

$L \to \bullet * R, \quad \$$

LR(1) Sets of Items for Pointer Assignment Grammar



I_0

$S' \rightarrow \bullet S, \quad \$$
$S \rightarrow \bullet L = R, \quad \$$
$S \rightarrow \bullet R, \quad \$$
$L \rightarrow \bullet * R, \quad =/\$$
$L \rightarrow \bullet \text{id}, \quad =/\$$
$R \rightarrow \bullet L, \quad \$$

I_1

$S' \rightarrow S\bullet, \ \$$

I_2

$S \rightarrow L\bullet = R, \ \$$
$R \rightarrow L\bullet, \quad \$$

I_3

$S \rightarrow R\bullet, \ \$$

I_4

$L \rightarrow * \bullet R, \ =/\$$
$R \rightarrow \bullet L, \quad =/\$$
$L \rightarrow \bullet \text{id}, \quad =/\$$
$L \rightarrow \bullet * R, \ =/\$$

I_5

$L \rightarrow \text{id}\bullet, \ =/\$$

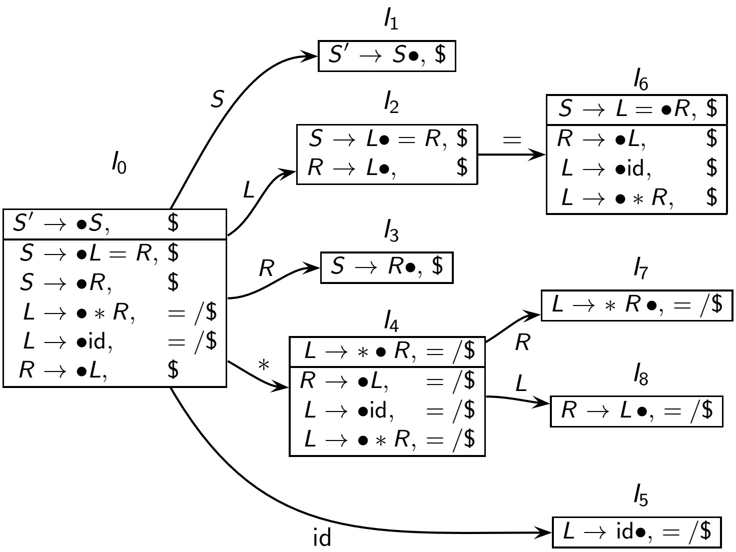
I_6

$S \rightarrow L = \bullet R, \ \$$
$R \rightarrow \bullet L, \quad \$$
$L \rightarrow \bullet \text{id}, \quad \$$
$L \rightarrow \bullet * R, \quad \$$

I_7

$L \rightarrow * R \bullet, \ =/\$$

LR(1) Sets of Items for Pointer Assignment Grammar



I_0

$S' \to \bullet S, \quad \$$

$S \to \bullet L = R, \quad \$$

$S \to \bullet R, \quad \$$

$L \to \bullet * R, \quad = / \$$

$L \to \bullet \text{id}, \quad = / \$$

$R \to \bullet L, \quad \$$

I_1

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

I_2

$S \to L \bullet = R, \ \$$

$R \to L \bullet, \quad \$$

I_3

$S \to R \bullet, \ \$$

I_4

$L \to * \bullet R, \ = / \$$

$R \to \bullet L, \quad = / \$$

$L \to \bullet \text{id}, \quad = / \$$

$L \to \bullet * R, \ = / \$$

I_5

$L \to \text{id} \bullet, \ = / \$$

I_6

$S \to L = \bullet R, \ \$$

$R \to \bullet L, \quad \$$

$L \to \bullet \text{id}, \quad \$$

$L \to \bullet * R, \quad \$$

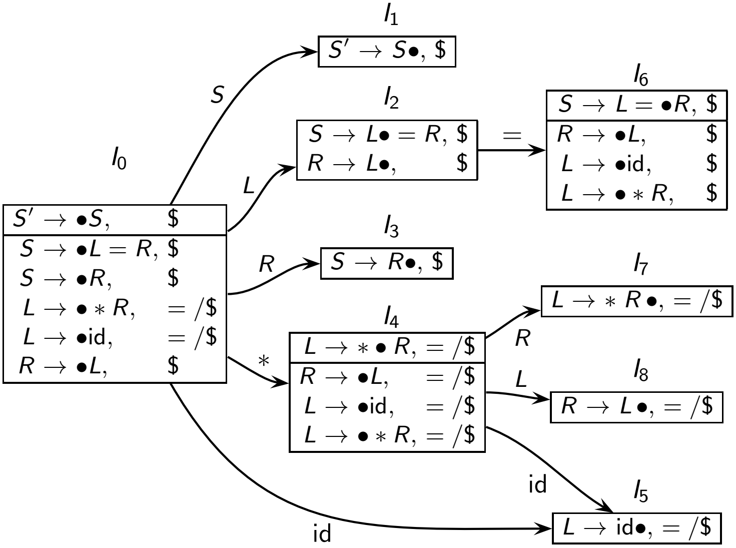
I_7

$L \to * R \bullet, \ = / \$$

I_8

$R \to L \bullet, \ = / \$$

LR(1) Sets of Items for Pointer Assignment Grammar



I_0

$S' \rightarrow \bullet S,$	$\$$
$S \rightarrow \bullet L = R,$	$\$$
$S \rightarrow \bullet R,$	$\$$
$L \rightarrow \bullet * R,$	$= /\$$
$L \rightarrow \bullet \text{id},$	$= /\$$
$R \rightarrow \bullet L,$	$\$$

I_1

$S' \rightarrow S\bullet,\ \$$

I_2

$S \rightarrow L\bullet = R,$	$\$$
$R \rightarrow L\bullet,$	$\$$

I_3

$S \rightarrow R\bullet,\ \$$

I_4

$L \rightarrow * \bullet R,$	$= /\$$
$R \rightarrow \bullet L,$	$= /\$$
$L \rightarrow \bullet \text{id},$	$= /\$$
$L \rightarrow \bullet * R,$	$= /\$$

I_5

$L \rightarrow \text{id}\bullet,\ = /\$$

I_6

$S \rightarrow L = \bullet R,$	$\$$
$R \rightarrow \bullet L,$	$\$$
$L \rightarrow \bullet \text{id},$	$\$$
$L \rightarrow \bullet * R,$	$\$$

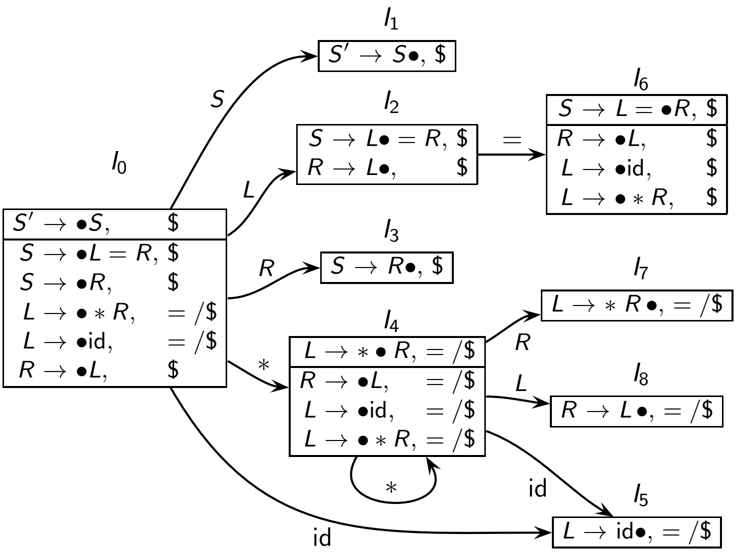
I_7

$L \rightarrow * R\bullet,\ = /\$$

I_8

$R \rightarrow L\bullet,\ = /\$$

LR(1) Sets of Items for Pointer Assignment Grammar



I_0

$S' \rightarrow \bullet S, \quad \$$
$S \rightarrow \bullet L = R, \quad \$$
$S \rightarrow \bullet R, \quad \$$
$L \rightarrow \bullet * R, \quad =/\$$
$L \rightarrow \bullet \text{id}, \quad =/\$$
$R \rightarrow \bullet L, \quad \$$

I_1

$S' \rightarrow S \bullet, \ \$$

I_2

$S \rightarrow L \bullet = R, \ \$$
$R \rightarrow L \bullet, \quad \$$

I_3

$S \rightarrow R \bullet, \ \$$

I_4

$L \rightarrow * \bullet R, \ =/\$$
$R \rightarrow \bullet L, \quad =/\$$
$L \rightarrow \bullet \text{id}, \quad =/\$$
$L \rightarrow \bullet * R, \ =/\$$

I_5

$L \rightarrow \text{id} \bullet, \ =/\$$

I_6

$S \rightarrow L = \bullet R, \ \$$
$R \rightarrow \bullet L, \quad \$$
$L \rightarrow \bullet \text{id}, \quad \$$
$L \rightarrow \bullet * R, \quad \$$

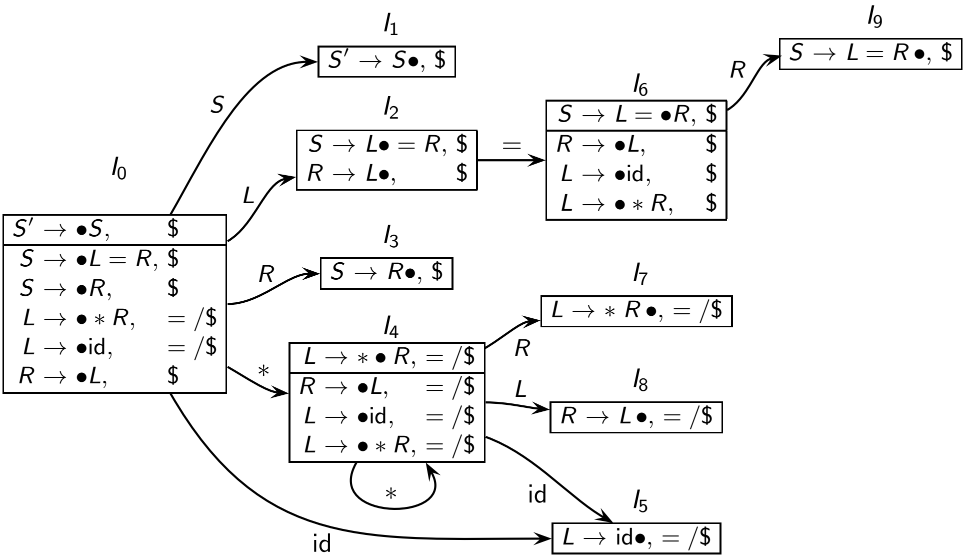
I_7

$L \rightarrow * R \bullet, \ =/\$$

I_8

$R \rightarrow L \bullet, \ =/\$$

LR(1) Sets of Items for Pointer Assignment Grammar



I_0

$S' \rightarrow \bullet S,\ \$$
$S \rightarrow \bullet L = R,\ \$$
$S \rightarrow \bullet R,\ \$$
$L \rightarrow \bullet * R,\ =/\$$
$L \rightarrow \bullet \text{id},\ =/\$$
$R \rightarrow \bullet L,\ \$$

I_1

$S' \rightarrow S\bullet,\ \$$

I_2

$S \rightarrow L\bullet = R,\ \$$
$R \rightarrow L\bullet,\ \$$

I_3

$S \rightarrow R\bullet,\ \$$

I_4

$L \rightarrow * \bullet R,\ =/\$$
$R \rightarrow \bullet L,\ =/\$$
$L \rightarrow \bullet \text{id},\ =/\$$
$L \rightarrow \bullet * R,\ =/\$$

I_5

$L \rightarrow \text{id}\bullet,\ =/\$$

I_6

$S \rightarrow L = \bullet R,\ \$$
$R \rightarrow \bullet L,\ \$$
$L \rightarrow \bullet \text{id},\ \$$
$L \rightarrow \bullet * R,\ \$$

I_7

$L \rightarrow * R\bullet,\ =/\$$

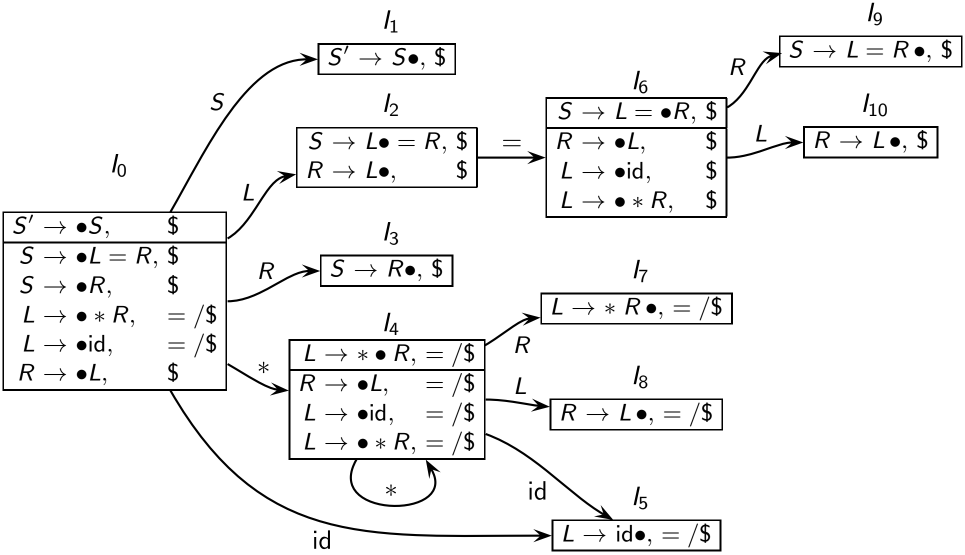
I_8

$R \rightarrow L\bullet,\ =/\$$

I_9

$S \rightarrow L = R\bullet,\ \$$

LR(1) Sets of Items for Pointer Assignment Grammar



I_0
$S' \to \bullet S, \quad \$$
$S \to \bullet L = R, \quad \$$
$S \to \bullet R, \quad \$$
$L \to \bullet * R, \quad =/\$$
$L \to \bullet \text{id}, \quad =/\$$
$R \to \bullet L, \quad \$$

I_1
$S' \to S \bullet, \ \$$

I_2
$S \to L \bullet = R, \ \$$
$R \to L \bullet, \quad \$$

I_3
$S \to R \bullet, \ \$$

I_4
$L \to * \bullet R, \ =/\$$
$R \to \bullet L, \quad =/\$$
$L \to \bullet \text{id}, \quad =/\$$
$L \to \bullet * R, \ =/\$$

I_5
$L \to \text{id} \bullet, \ =/\$$

I_6
$S \to L = \bullet R, \ \$$
$R \to \bullet L, \quad \$$
$L \to \bullet \text{id}, \quad \$$
$L \to \bullet * R, \quad \$$

I_7
$L \to * R \bullet, \ =/\$$

I_8
$R \to L \bullet, \ =/\$$

I_9
$S \to L = R \bullet, \ \$$

I_{10}
$R \to L \bullet, \ \$$

IIT Bombay
cs302: Implementation of Programming Languages

Topic:

Syntax Analysis

Section:

Grammars, Derivations, and Parse Trees

Shift Reduce Parsing

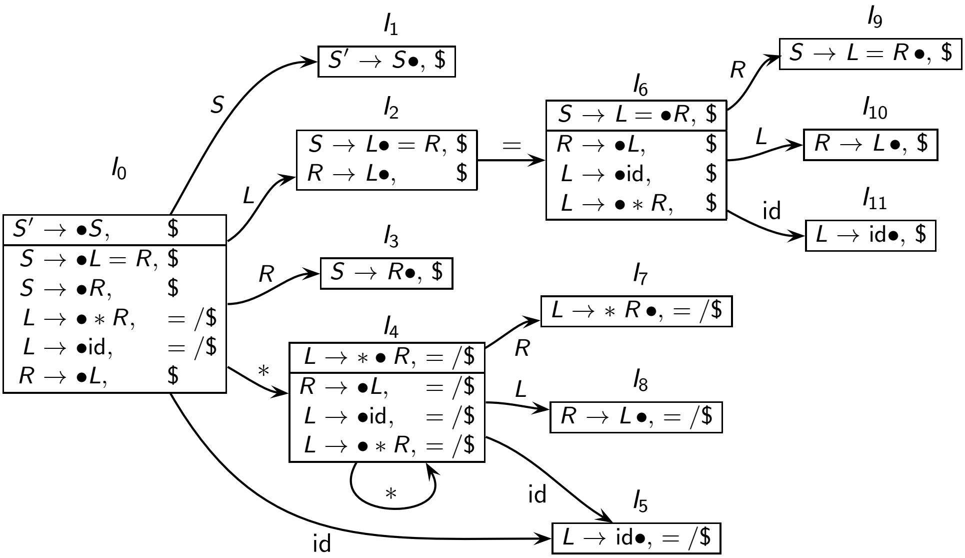
SLR(1) Parsing

Conceptual Issues in Parsing

CLR(1) Parsing

LALR(1) Parsing

LR(1) Sets of Items for Pointer Assignment Grammar



I_0
$S' \rightarrow \bullet S, \quad \$$
$S \rightarrow \bullet L = R, \quad \$$
$S \rightarrow \bullet R, \quad \$$
$L \rightarrow \bullet * R, \quad =/\$$
$L \rightarrow \bullet \mathrm{id}, \quad =/\$$
$R \rightarrow \bullet L, \quad \$$

I_1
$S' \rightarrow S\bullet, \ \$$

I_2
$S \rightarrow L\bullet = R, \ \$$
$R \rightarrow L\bullet, \quad \$$

I_3
$S \rightarrow R\bullet, \ \$$

I_4
$L \rightarrow * \bullet R, \ =/\$$
$R \rightarrow \bullet L, \quad =/\$$
$L \rightarrow \bullet \mathrm{id}, \quad =/\$$
$L \rightarrow \bullet * R, \ =/\$$

I_5
$L \rightarrow \mathrm{id}\bullet, \ =/\$$

I_6
$S \rightarrow L = \bullet R, \ \$$
$R \rightarrow \bullet L, \quad \$$
$L \rightarrow \bullet \mathrm{id}, \quad \$$
$L \rightarrow \bullet * R, \quad \$$

I_7
$L \rightarrow * R\bullet, \ =/\$$

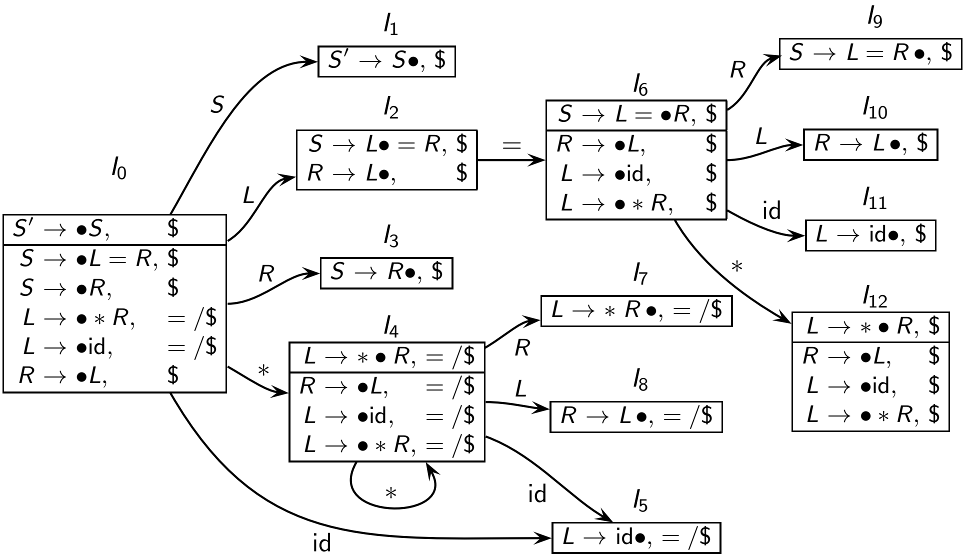
I_8
$R \rightarrow L\bullet, \ =/\$$

I_9
$S \rightarrow L = R\bullet, \ \$$

I_{10}
$R \rightarrow L\bullet, \ \$$

I_{11}
$L \rightarrow \mathrm{id}\bullet, \ \$$

LR(1) Sets of Items for Pointer Assignment Grammar



I_0
$S' \rightarrow \bullet S, \quad \$$
$S \rightarrow \bullet L = R, \ \$$
$S \rightarrow \bullet R, \quad \$$
$L \rightarrow \bullet * R, \quad =/\$$
$L \rightarrow \bullet \text{id}, \quad =/\$$
$R \rightarrow \bullet L, \quad \$$

I_1
$S' \rightarrow S \bullet, \ \$$

I_2
$S \rightarrow L \bullet = R, \ \$$
$R \rightarrow L \bullet, \quad \$$

I_3
$S \rightarrow R \bullet, \ \$$

I_4
$L \rightarrow * \bullet R, \ =/\$$
$R \rightarrow \bullet L, \quad =/\$$
$L \rightarrow \bullet \text{id}, \quad =/\$$
$L \rightarrow \bullet * R, \ =/\$$

I_5
$L \rightarrow \text{id} \bullet, \ =/\$$

I_6
$S \rightarrow L = \bullet R, \ \$$
$R \rightarrow \bullet L, \quad \$$
$L \rightarrow \bullet \text{id}, \quad \$$
$L \rightarrow \bullet * R, \quad \$$

I_7
$L \rightarrow * R \bullet, \ =/\$$

I_8
$R \rightarrow L \bullet, \ =/\$$

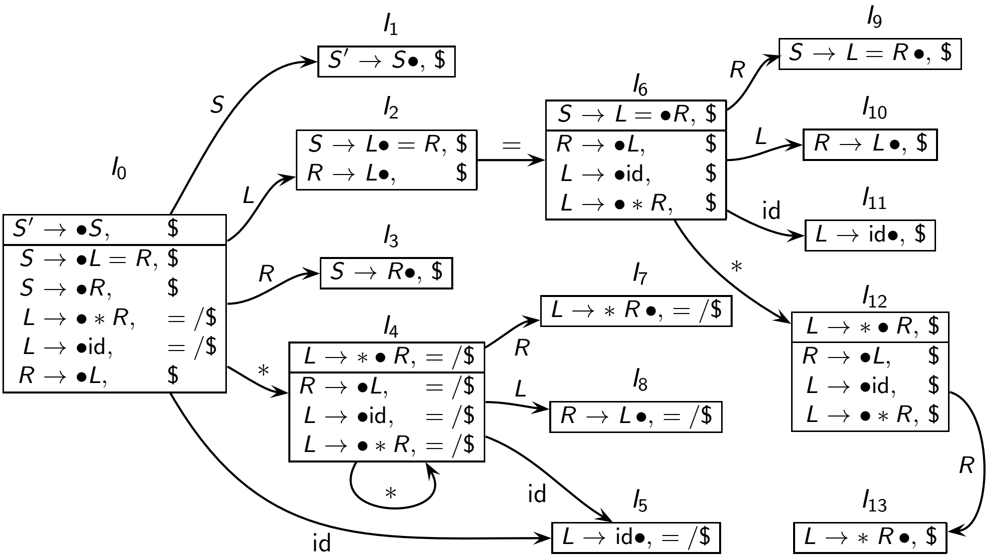
I_9
$S \rightarrow L = R \bullet, \ \$$

I_{10}
$R \rightarrow L \bullet, \ \$$

I_{11}
$L \rightarrow \text{id} \bullet, \ \$$

I_{12}
$L \rightarrow * \bullet R, \ \$$
$R \rightarrow \bullet L, \quad \$$
$L \rightarrow \bullet \text{id}, \quad \$$
$L \rightarrow \bullet * R, \ \$$

LR(1) Sets of Items for Pointer Assignment Grammar



I_0

$$\begin{array}{lll} S' \rightarrow \bullet S, & \$ \\ \hline S \rightarrow \bullet L = R, & \$ \\ S \rightarrow \bullet R, & \$ \\ L \rightarrow \bullet * R, & =/\$ \\ L \rightarrow \bullet \text{id}, & =/\$ \\ R \rightarrow \bullet L, & \$ \end{array}$$

I_1

$S' \rightarrow S\bullet, \$$

I_2

$$\begin{array}{ll} S \rightarrow L\bullet = R, & \$ \\ R \rightarrow L\bullet, & \$ \end{array}$$

I_3

$S \rightarrow R\bullet, \$$

I_4

$$\begin{array}{ll} L \rightarrow * \bullet R, & =/\$ \\ \hline R \rightarrow \bullet L, & =/\$ \\ L \rightarrow \bullet \text{id}, & =/\$ \\ L \rightarrow \bullet * R, & =/\$ \end{array}$$

I_5

$L \rightarrow \text{id}\bullet, =/\$$

I_6

$$\begin{array}{ll} S \rightarrow L = \bullet R, & \$ \\ \hline R \rightarrow \bullet L, & \$ \\ L \rightarrow \bullet \text{id}, & \$ \\ L \rightarrow \bullet * R, & \$ \end{array}$$

I_7

$L \rightarrow * R \bullet, =/\$$

I_8

$R \rightarrow L \bullet, =/\$$

I_9

$S \rightarrow L = R \bullet, \$$

I_{10}

$R \rightarrow L \bullet, \$$

I_{11}

$L \rightarrow \text{id}\bullet, \$$

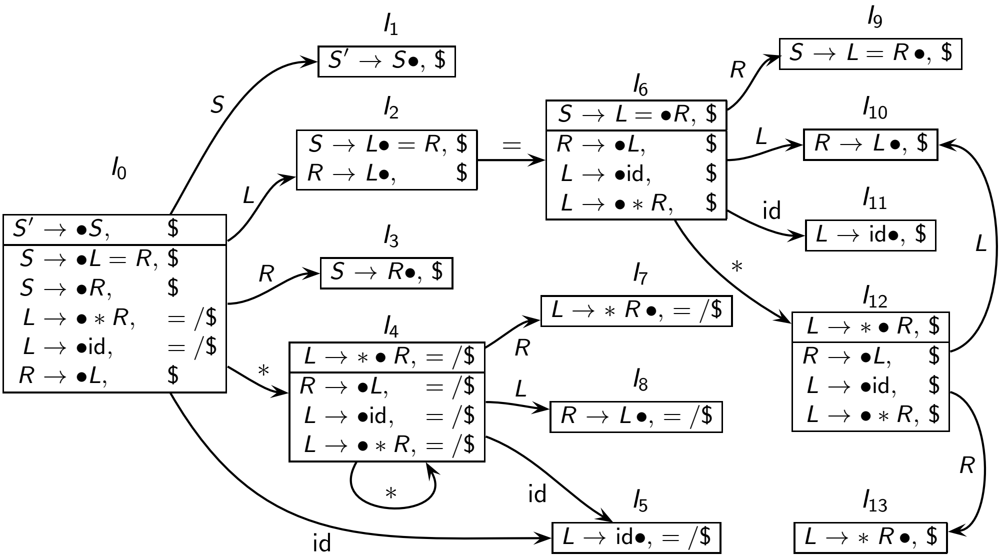
I_{12}

$$\begin{array}{ll} L \rightarrow * \bullet R, & \$ \\ \hline R \rightarrow \bullet L, & \$ \\ L \rightarrow \bullet \text{id}, & \$ \\ L \rightarrow \bullet * R, & \$ \end{array}$$

I_{13}

$L \rightarrow * R \bullet, \$$

LR(1) Sets of Items for Pointer Assignment Grammar



I_0

$S' \to \bullet S, \quad \$$

$S \to \bullet L = R, \quad \$$

$S \to \bullet R, \quad \$$

$L \to \bullet * R, \quad =/\$$

$L \to \bullet \text{id}, \quad =/\$$

$R \to \bullet L, \quad \$$

I_1

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

I_2

$S \to L \bullet = R, \ \$$

$R \to L \bullet, \quad \$$

I_3

$S \to R \bullet, \ \$$

I_4

$L \to * \bullet R, \ =/\$$

$R \to \bullet L, \quad =/\$$

$L \to \bullet \text{id}, \quad =/\$$

$L \to \bullet * R, \ =/\$$

I_5

$L \to \text{id} \bullet, \ =/\$$

I_6

$S \to L = \bullet R, \ \$$

$R \to \bullet L, \quad \$$

$L \to \bullet \text{id}, \quad \$$

$L \to \bullet * R, \quad \$$

I_7

$L \to * R \bullet, \ =/\$$

I_8

$R \to L \bullet, \ =/\$$

I_9

$S \to L = R \bullet, \ \$$

I_{10}

$R \to L \bullet, \ \$$

I_{11}

$L \to \text{id} \bullet, \ \$$

I_{12}

$L \to * \bullet R, \ \$$

$R \to \bullet L, \quad \$$

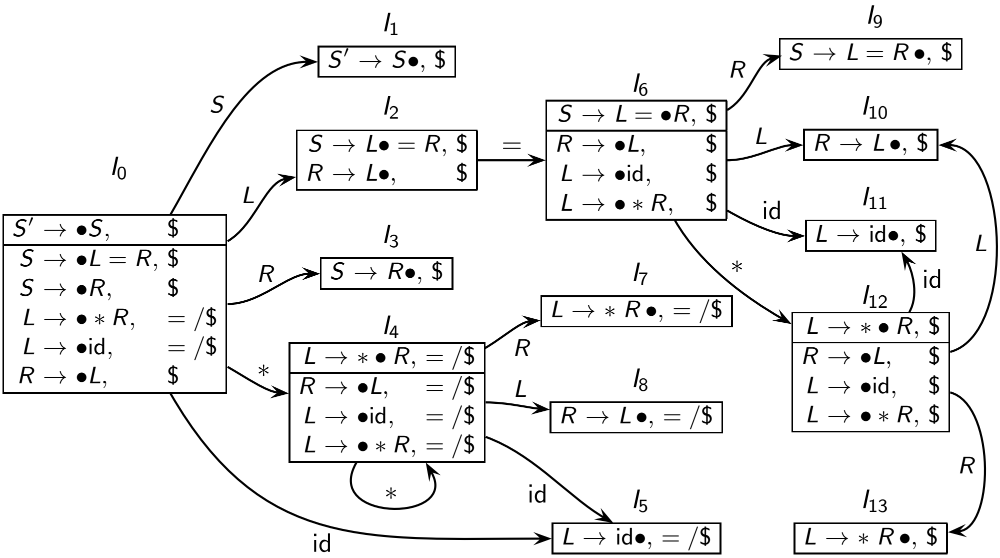
$L \to \bullet \text{id}, \quad \$$

$L \to \bullet * R, \ \$$

I_{13}

$L \to * R \bullet, \ \$$

LR(1) Sets of Items for Pointer Assignment Grammar



I_0
$S' \to \bullet S, \quad \$$
$S \to \bullet L = R, \quad \$$
$S \to \bullet R, \quad \$$
$L \to \bullet * R, \quad =/\$$
$L \to \bullet \text{id}, \quad =/\$$
$R \to \bullet L, \quad \$$

I_1
$S' \to S\bullet, \ \$$

I_2
$S \to L\bullet = R, \ \$$
$R \to L\bullet, \quad \$$

I_3
$S \to R\bullet, \ \$$

I_4
$L \to * \bullet R, \ =/\$$
$R \to \bullet L, \quad =/\$$
$L \to \bullet \text{id}, \quad =/\$$
$L \to \bullet * R, \ =/\$$

I_5
$L \to \text{id}\bullet, \ =/\$$

I_6
$S \to L = \bullet R, \ \$$
$R \to \bullet L, \quad \$$
$L \to \bullet \text{id}, \quad \$$
$L \to \bullet * R, \quad \$$

I_7
$L \to * R\bullet, \ =/\$$

I_8
$R \to L\bullet, \ =/\$$

I_9
$S \to L = R\bullet, \ \$$

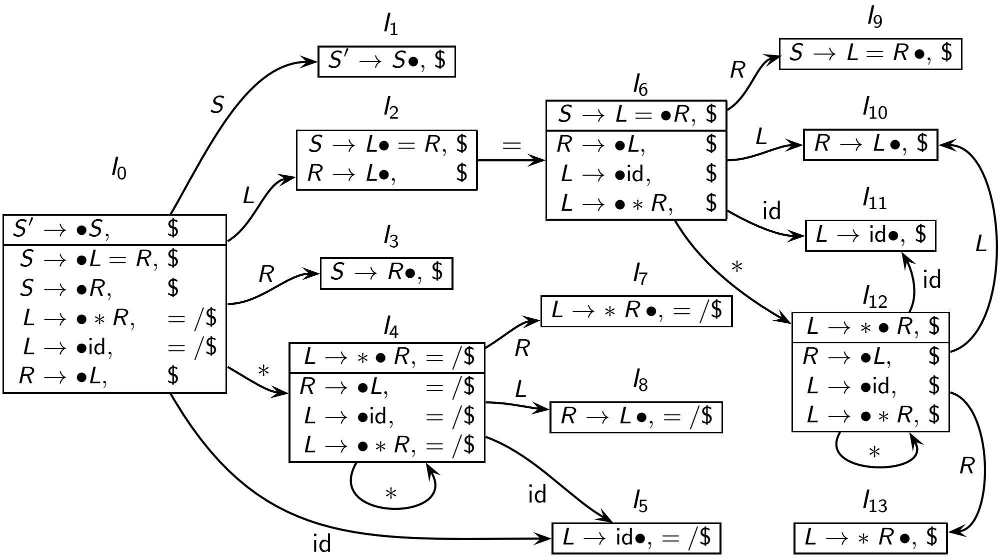
I_{10}
$R \to L\bullet, \ \$$

I_{11}
$L \to \text{id}\bullet, \ \$$

I_{12}
$L \to * \bullet R, \ \$$
$R \to \bullet L, \quad \$$
$L \to \bullet \text{id}, \quad \$$
$L \to \bullet * R, \ \$$

I_{13}
$L \to * R\bullet, \ \$$

LR(1) Sets of Items for Pointer Assignment Grammar



I_0
$S' \rightarrow \bullet S, \quad \$$
$S \rightarrow \bullet L = R, \quad \$$
$S \rightarrow \bullet R, \quad \$$
$L \rightarrow \bullet * R, \quad =/\$$
$L \rightarrow \bullet \text{id}, \quad =/\$$
$R \rightarrow \bullet L, \quad \$$

I_1
$S' \rightarrow S \bullet, \ \$$

I_2
$S \rightarrow L \bullet = R, \quad \$$
$R \rightarrow L \bullet, \quad \$$

I_3
$S \rightarrow R \bullet, \ \$$

I_4
$L \rightarrow * \bullet R, \ =/\$$
$R \rightarrow \bullet L, \quad =/\$$
$L \rightarrow \bullet \text{id}, \quad =/\$$
$L \rightarrow \bullet * R, \ =/\$$

I_5
$L \rightarrow \text{id} \bullet, \ =/\$$

I_6
$S \rightarrow L = \bullet R, \ \$$
$R \rightarrow \bullet L, \quad \$$
$L \rightarrow \bullet \text{id}, \quad \$$
$L \rightarrow \bullet * R, \quad \$$

I_7
$L \rightarrow * R \bullet, \ =/\$$

I_8
$R \rightarrow L \bullet, \ =/\$$

I_9
$S \rightarrow L = R \bullet, \ \$$

I_{10}
$R \rightarrow L \bullet, \ \$$

I_{11}
$L \rightarrow \text{id} \bullet, \ \$$

I_{12}
$L \rightarrow * \bullet R, \ \$$
$R \rightarrow \bullet L, \quad \$$
$L \rightarrow \bullet \text{id}, \quad \$$
$L \rightarrow \bullet * R, \quad \$$

I_{13}
$L \rightarrow * R \bullet, \ \$$

IIT Bombay
cs302: Implementation of Programming Languages

Topic:

Syntax Analysis

Section:

Grammars, Derivations, and Parse Trees

Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in Parsing

CLR(1) Parsing

LALR(1) Parsing

LR(1) (aka CLR(1)) Parsing Table for Pointer Assignment Grammar

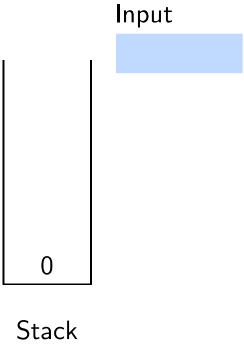
0 $S' \rightarrow S$
1 $S \rightarrow L = R$
2 $S \rightarrow R$
3 $L \rightarrow * R$
4 $L \rightarrow \text{id}$
5 $R \rightarrow L$

State	Action				Goto		
	`id`	`*`	`=`	`$`	S	L	R
0	$s5$	$s4$			$c1$	$c2$	$c3$
1				acc			
2			$s6$	$r5$			
3				$r2$			
4	$s5$	$s4$				$c8$	$c7$
5			$r4$	$r4$			
6	$s11$	$s12$				$c10$	$c9$
7			$r3$	$r3$			
8			$r5$	$r5$			
9				$r1$			
10				$r5$			
11				$r4$			
12	$s11$	$s12$				c10	c13
13				$r3$			

LR(1) (aka CLR(1)) Parsing for the Pointer Assignment Grammar

State	Action				Goto		
	`id`	$*$	`=`	$	S	L	R
0	$s5$	$s4$			$c1$	$c2$	$c3$
1				acc			
2			$s6$	$r5$			
3				$r2$			
4	$s5$	$s4$				$c8$	$c7$
5			$r4$	$r4$			
6	$s11$	$s12$				$c10$	$c9$
7			$r3$	$r3$			
8			$r5$	$r5$			
9				$r1$			
10				$r5$			
11				$r4$			
12	$s11$	$s12$				c10	c13
13				$r3$			

0 $S' \rightarrow S$
1 $S \rightarrow L = R$
2 $S \rightarrow R$
3 $L \rightarrow * R$
4 $L \rightarrow \text{id}$
5 $R \rightarrow L$



LR(1) (aka CLR(1)) Parsing for the Pointer Assignment Grammar

State	Action				Goto		
	id	*	=	$	S	L	R
0	s5	s4			c1	c2	c3
1				acc			
2			s6	r5			
3				r2			
4	s5	s4				c8	c7
5			r4	r4			
6	s11	s12				c10	c9
7			r3	r3			
8			r5	r5			
9				r1			
10				r5			
11				r4			
12	s11	s12				c10	c13
13				r3			

0 $S' \rightarrow S$
1 $S \rightarrow L = R$
2 $S \rightarrow R$
3 $L \rightarrow * R$
4 $L \rightarrow \text{id}$
5 $R \rightarrow L$

Input

id = id$

Shift 5

0

Stack

LR(1) (aka CLR(1)) Parsing for the Pointer Assignment Grammar

State	Action				Goto		
	id	*	=	$	S	L	R
0	s5	s4			c1	c2	c3
1				acc			
2			s6	r5			
3				r2			
4	s5	s4				c8	c7
5			r4	r4			
6	s11	s12				c10	c9
7			r3	r3			
8			r5	r5			
9				r1			
10				r5			
11				r4			
12	s11	s12				c10	c13
13				r3			

0 $S' \rightarrow S$
1 $S \rightarrow L = R$
2 $S \rightarrow R$
3 $L \rightarrow * R$
4 $L \rightarrow \text{id}$
5 $R \rightarrow L$

Input

$= \text{id}\$$

Reduce by 4

5
id
0

Stack

IIT Bombay

cs302: Implementation of Programming Languages

Topic:

Syntax Analysis

Section:

Grammars, Derivations, and Parse Trees

Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in Parsing

CLR(1) Parsing

LALR(1) Parsing

LR(1) (aka CLR(1)) Parsing for the Pointer Assignment Grammar

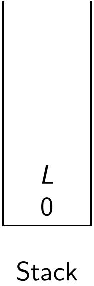
State	Action				Goto		
	id	*	=	$	S	L	R
0	$s5$	$s4$			$c1$	$c2$	$c3$
1				acc			
2			$s6$	$r5$			
3				$r2$			
4	$s5$	$s4$				$c8$	$c7$
5			$r4$	$r4$			
6	$s11$	$s12$				$c10$	$c9$
7			$r3$	$r3$			
8			$r5$	$r5$			
9				$r1$			
10				$r5$			
11				$r4$			
12	$s11$	$s12$				c10	c13
13				$r3$			

0 $S' \rightarrow S$
1 $S \rightarrow L = R$
2 $S \rightarrow R$
3 $L \rightarrow * R$
4 $L \rightarrow \text{id}$
5 $R \rightarrow L$

Input

$= \text{id}\$$

Cover by 2



LR(1) (aka CLR(1)) Parsing for the Pointer Assignment Grammar

State	Action				Goto		
	`id`	`*`	`=`	`$`	S	L	R
0	$s5$	$s4$			$c1$	$c2$	$c3$
1				acc			
2			$s6$	$r5$			
3				$r2$			
4	$s5$	$s4$				$c8$	$c7$
5			$r4$	$r4$			
6	$s11$	$s12$				$c10$	$c9$
7			$r3$	$r3$			
8			$r5$	$r5$			
9				$r1$			
10				$r5$			
11				$r4$			
12	$s11$	$s12$				c10	c13
13				$r3$			

0 $S' \to S$
1 $S \to L = R$
2 $S \to R$
3 $L \to * R$
4 $L \to \text{id}$
5 $R \to L$

Input

$= \text{id}\$$

Shift 6

2
L
0

Stack

LR(1) (aka CLR(1)) Parsing for the Pointer Assignment Grammar

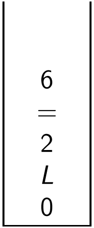
State	Action				Goto		
	`id`	$*$	`=`	\$	S	L	R
0	$s5$	$s4$			$c1$	$c2$	$c3$
1				acc			
2			$s6$	$r5$			
3				$r2$			
4	$s5$	$s4$				$c8$	$c7$
5			$r4$	$r4$			
6	$s11$	$s12$				$c10$	$c9$
7			$r3$	$r3$			
8			$r5$	$r5$			
9				$r1$			
10				$r5$			
11				$r4$			
12	$s11$	$s12$				c10	c13
13				$r3$			

0 $S' \rightarrow S$
1 $S \rightarrow L = R$
2 $S \rightarrow R$
3 $L \rightarrow * R$
4 $L \rightarrow \text{id}$
5 $R \rightarrow L$

Input

id\$

Shift 11


6
=
2
L
0

Stack

LR(1) (aka CLR(1)) Parsing for the Pointer Assignment Grammar

State	Action				Goto		
	`id`	$*$	=	\$	S	L	R
0	$s5$	$s4$			$c1$	$c2$	$c3$
1				acc			
2			$s6$	$r5$			
3				$r2$			
4	$s5$	$s4$				$c8$	$c7$
5			$r4$	$r4$			
6	$s11$	$s12$				$c10$	$c9$
7			$r3$	$r3$			
8			$r5$	$r5$			
9				$r1$			
10				$r5$			
11				$r4$			
12	$s11$	$s12$				c10	c13
13				$r3$			

0 $S' \rightarrow S$
1 $S \rightarrow L = R$
2 $S \rightarrow R$
3 $L \rightarrow * R$
4 $L \rightarrow \text{id}$
5 $R \rightarrow L$

Input

\$

11
id
6
=
2
L
0

Stack

Reduce by 4

LR(1) (aka CLR(1)) Parsing for the Pointer Assignment Grammar

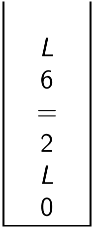
State	Action				Goto		
	`id`	$*$	`=`	\$	S	L	R
0	$s5$	$s4$			$c1$	$c2$	$c3$
1				acc			
2			$s6$	$r5$			
3				$r2$			
4	$s5$	$s4$				$c8$	$c7$
5			$r4$	$r4$			
6	$s11$	$s12$				$c10$	$c9$
7			$r3$	$r3$			
8			$r5$	$r5$			
9				$r1$			
10				$r5$			
11				$r4$			
12	$s11$	$s12$				c10	c13
13				$r3$			

0 $S' \rightarrow S$
1 $S \rightarrow L = R$
2 $S \rightarrow R$
3 $L \rightarrow * R$
4 $L \rightarrow \text{id}$
5 $R \rightarrow L$

Input

\$

Cover by 10


L
6
$=$
2
L
0

Stack

LR(1) (aka CLR(1)) Parsing for the Pointer Assignment Grammar

State	Action				Goto		
	`id`	$*$	`=`	\$	S	L	R
0	$s5$	$s4$			$c1$	$c2$	$c3$
1				acc			
2			$s6$	$r5$			
3				$r2$			
4	$s5$	$s4$				$c8$	$c7$
5			$r4$	$r4$			
6	$s11$	$s12$				$c10$	$c9$
7			$r3$	$r3$			
8			$r5$	$r5$			
9				$r1$			
10				$r5$			
11				$r4$			
12	$s11$	$s12$				c10	c13
13				$r3$			

0 $S' \rightarrow S$
1 $S \rightarrow L = R$
2 $S \rightarrow R$
3 $L \rightarrow * R$
4 $L \rightarrow \text{id}$
5 $R \rightarrow L$

Input

\$

10
L
6
$=$
2
L
0

Stack

Reduce by 5

LR(1) (aka CLR(1)) Parsing for the Pointer Assignment Grammar

State	Action				Goto		
	`id`	$*$	=	\$	S	L	R
0	$s5$	$s4$			$c1$	$c2$	$c3$
1				acc			
2			$s6$	$r5$			
3				$r2$			
4	$s5$	$s4$				$c8$	$c7$
5			$r4$	$r4$			
6	$s11$	$s12$				$c10$	$c9$
7			$r3$	$r3$			
8			$r5$	$r5$			
9				$r1$			
10				$r5$			
11				$r4$			
12	$s11$	$s12$				c10	c13
13				$r3$			

0 $S' \rightarrow S$
1 $S \rightarrow L = R$
2 $S \rightarrow R$
3 $L \rightarrow * R$
4 $L \rightarrow \text{id}$
5 $R \rightarrow L$

Input

\$

Cover by 9

R
6
=
2
L
0

Stack

LR(1) (aka CLR(1)) Parsing for the Pointer Assignment Grammar

State	Action				Goto		
	`id`	*	=	$	S	L	R
0	$s5$	$s4$			$c1$	$c2$	$c3$
1				acc			
2			$s6$	$r5$			
3				$r2$			
4	$s5$	$s4$				$c8$	$c7$
5			$r4$	$r4$			
6	$s11$	$s12$				$c10$	$c9$
7			$r3$	$r3$			
8			$r5$	$r5$			
9				$r1$			
10				$r5$			
11				$r4$			
12	$s11$	$s12$				c10	c13
13				$r3$			

0 $S' \rightarrow S$
1 $S \rightarrow L = R$
2 $S \rightarrow R$
3 $L \rightarrow * R$
4 $L \rightarrow \text{id}$
5 $R \rightarrow L$

Input

$

Reduce by 1

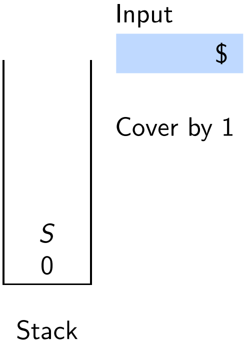
9
R
6
$=$
2
L
0

Stack

LR(1) (aka CLR(1)) Parsing for the Pointer Assignment Grammar

State	Action				Goto		
	`id`	`*`	`=`	`$`	S	L	R
0	$s5$	$s4$			$c1$	$c2$	$c3$
1				acc			
2			$s6$	$r5$			
3				$r2$			
4	$s5$	$s4$				$c8$	$c7$
5			$r4$	$r4$			
6	$s11$	$s12$				$c10$	$c9$
7			$r3$	$r3$			
8			$r5$	$r5$			
9				$r1$			
10				$r5$			
11				$r4$			
12	$s11$	$s12$				c10	c13
13				$r3$			

0 $S' \rightarrow S$
1 $S \rightarrow L = R$
2 $S \rightarrow R$
3 $L \rightarrow * R$
4 $L \rightarrow \text{id}$
5 $R \rightarrow L$


Input

$

Cover by 1

S
0

Stack

LR(1) (aka CLR(1)) Parsing for the Pointer Assignment Grammar

State	Action				Goto		
	id	*	=	$	S	L	R
0	$s5$	$s4$			$c1$	$c2$	$c3$
1				acc			
2			$s6$	$r5$			
3				$r2$			
4	$s5$	$s4$				$c8$	$c7$
5			$r4$	$r4$			
6	$s11$	$s12$				$c10$	$c9$
7			$r3$	$r3$			
8			$r5$	$r5$			
9				$r1$			
10				$r5$			
11				$r4$			
12	$s11$	$s12$				c10	c13
13				$r3$			

0 $S' \rightarrow S$
1 $S \rightarrow L = R$
2 $S \rightarrow R$
3 $L \rightarrow * R$
4 $L \rightarrow \text{id}$
5 $R \rightarrow L$

Input

$

Accept

1
S
0

Stack

Another Example of LR(1) (aka CLR(1)) Parsing

$$
\begin{aligned}
A &\rightarrow aBe\\
A &\rightarrow aCd\\
A &\rightarrow bBd\\
A &\rightarrow bCe\\
B &\rightarrow f\\
C &\rightarrow f
\end{aligned}
$$

Another Example of LR(1) (aka CLR(1)) Parsing

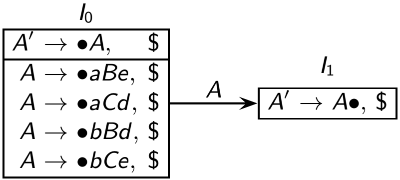
$A \rightarrow aBe$
$A \rightarrow aCd$
$A \rightarrow bBd$
$A \rightarrow bCe$
$B \rightarrow f$
$C \rightarrow f$

I_0

$A' \rightarrow \bullet A,$	$\$$
$A \rightarrow \bullet aBe,$	$\$$
$A \rightarrow \bullet aCd,$	$\$$
$A \rightarrow \bullet bBd,$	$\$$
$A \rightarrow \bullet bCe,$	$\$$

Another Example of LR(1) (aka CLR(1)) Parsing

$A \rightarrow aBe$
$A \rightarrow aCd$
$A \rightarrow bBd$
$A \rightarrow bCe$
$B \rightarrow f$
$C \rightarrow f$



I_0

$A' \rightarrow \bullet A, \quad \$$

$A \rightarrow \bullet aBe, \ \$$
$A \rightarrow \bullet aCd, \ \$$
$A \rightarrow \bullet bBd, \ \$$
$A \rightarrow \bullet bCe, \ \$$

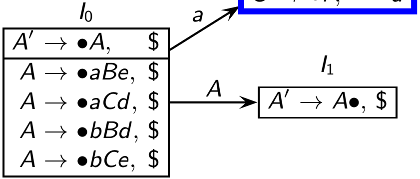
A

I_1

$A' \rightarrow A\bullet, \ \$$

Another Example of LR(1) (aka CLR(1)) Parsing

$$A \to aBe$$
$$A \to aCd$$
$$A \to bBd$$
$$A \to bCe$$
$$B \to f$$
$$C \to f$$



I_0

$A' \to \bullet A, \quad \$$

$A \to \bullet aBe, \quad \$$
$A \to \bullet aCd, \quad \$$
$A \to \bullet bBd, \quad \$$
$A \to \bullet bCe, \quad \$$

a

I_2

$A \to a \bullet Be, \quad \$$
$A \to a \bullet Cd, \quad \$$

$B \to \bullet f, \quad e$
$C \to \bullet f, \quad d$

A

I_1

$A' \to A \bullet, \quad \$$

Closure of $P \to \alpha \bullet Q\beta, p$ contains items of the form $Q \to \bullet\gamma$, FIRST(βp)

In our example

- For $Q = B$, β is e and p is \$

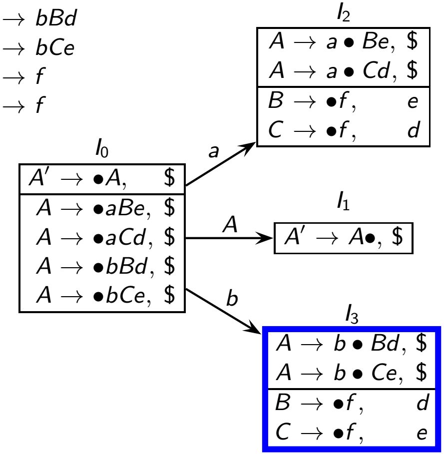
  If we expect to see a string derivable from B in this state, the string must be followed by FIRST(βp) = FIRST($e\$$) = e

- For $Q = C$, β is d and p is \$

  If we expect to see a string derivable from C in this state, the string must be followed by FIRST(βp) = FIRST($d\$$) = d

Another Example of LR(1) (aka CLR(1)) Parsing

$A \rightarrow aBe$
$A \rightarrow aCd$
$A \rightarrow bBd$
$A \rightarrow bCe$
$B \rightarrow f$
$C \rightarrow f$



I_0
$A' \rightarrow \bullet A, \quad \$$
$A \rightarrow \bullet aBe, \quad \$$
$A \rightarrow \bullet aCd, \quad \$$
$A \rightarrow \bullet bBd, \quad \$$
$A \rightarrow \bullet bCe, \quad \$$

I_2
$A \rightarrow a \bullet Be, \quad \$$
$A \rightarrow a \bullet Cd, \quad \$$
$B \rightarrow \bullet f, \quad e$
$C \rightarrow \bullet f, \quad d$

I_1
$A' \rightarrow A\bullet, \quad \$$

I_3
$A \rightarrow b \bullet Bd, \quad \$$
$A \rightarrow b \bullet Ce, \quad \$$
$B \rightarrow \bullet f, \quad d$
$C \rightarrow \bullet f, \quad e$

Closure of $P \rightarrow \alpha \bullet Q\beta, p$ contains items of the form $Q \rightarrow \bullet\, \gamma,\ \text{FIRST}(\beta p)$

In our example

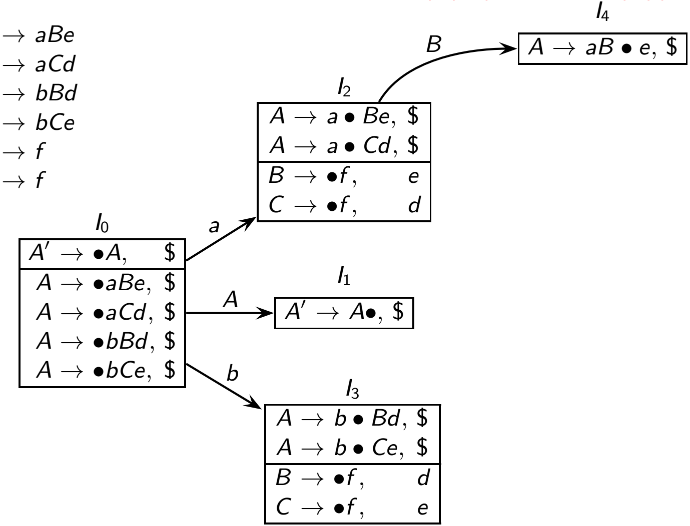
- For $Q = B$, β is d and p is \$

  If we expect to see a string derivable from B in this state, the string must be followed by $\text{FIRST}(\beta p) = \text{FIRST}(d\$) = d$
- For $Q = C$, β is e and p is \$

  If we expect to see a string derivable from C in this state, the string must be followed by $\text{FIRST}(\beta p) = \text{FIRST}(e\$) = e$

Another Example of LR(1) (aka CLR(1)) Parsing

$A \to aBe$
$A \to aCd$
$A \to bBd$
$A \to bCe$
$B \to f$
$C \to f$



I_0
$A' \to \bullet A,\ \$$
$A \to \bullet aBe,\ \$$
$A \to \bullet aCd,\ \$$
$A \to \bullet bBd,\ \$$
$A \to \bullet bCe,\ \$$

I_1
$A' \to A\bullet,\ \$$

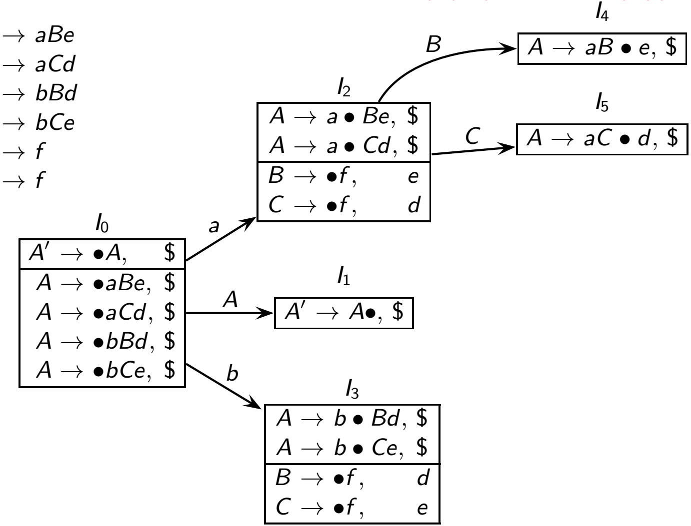
I_2
$A \to a \bullet Be,\ \$$
$A \to a \bullet Cd,\ \$$
$B \to \bullet f,\ e$
$C \to \bullet f,\ d$

I_3
$A \to b \bullet Bd,\ \$$
$A \to b \bullet Ce,\ \$$
$B \to \bullet f,\ d$
$C \to \bullet f,\ e$

I_4
$A \to aB \bullet e,\ \$$

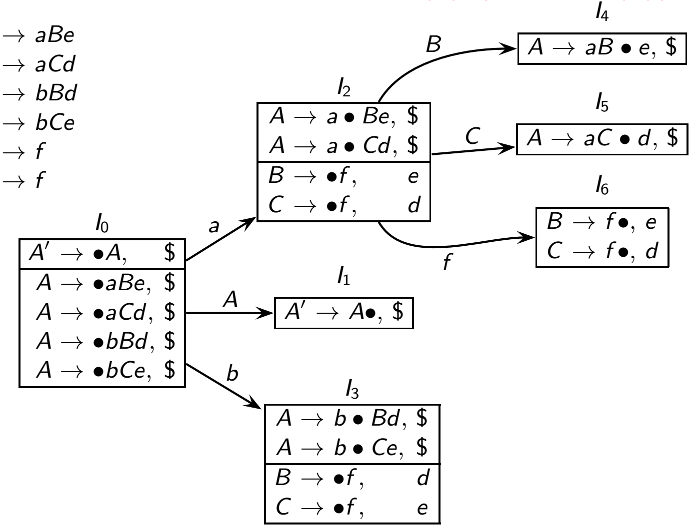
Another Example of LR(1) (aka CLR(1)) Parsing

$A \to aBe$
$A \to aCd$
$A \to bBd$
$A \to bCe$
$B \to f$
$C \to f$



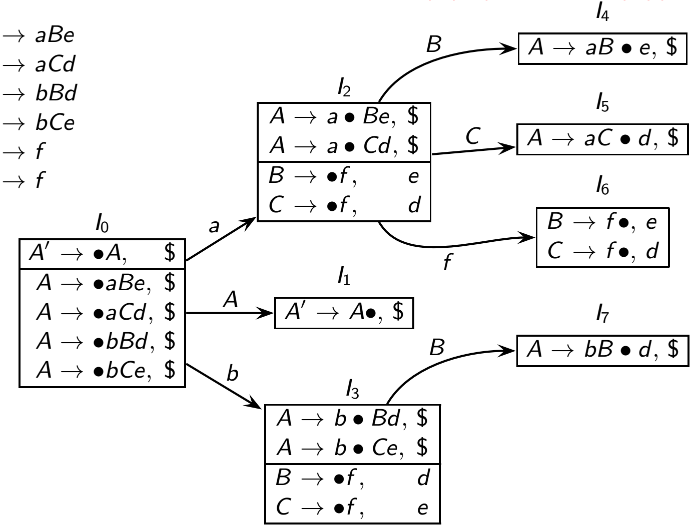
Another Example of LR(1) (aka CLR(1)) Parsing

$A \rightarrow aBe$
$A \rightarrow aCd$
$A \rightarrow bBd$
$A \rightarrow bCe$
$B \rightarrow f$
$C \rightarrow f$



Another Example of LR(1) (aka CLR(1)) Parsing

$A \rightarrow aBe$
$A \rightarrow aCd$
$A \rightarrow bBd$
$A \rightarrow bCe$
$B \rightarrow f$
$C \rightarrow f$



I_0
$A' \rightarrow \bullet A, \quad \$$
$A \rightarrow \bullet aBe, \quad \$$
$A \rightarrow \bullet aCd, \quad \$$
$A \rightarrow \bullet bBd, \quad \$$
$A \rightarrow \bullet bCe, \quad \$$

I_1
$A' \rightarrow A\bullet, \ \$$

I_2
$A \rightarrow a \bullet Be, \ \$$
$A \rightarrow a \bullet Cd, \ \$$
$B \rightarrow \bullet f, \quad e$
$C \rightarrow \bullet f, \quad d$

I_3
$A \rightarrow b \bullet Bd, \ \$$
$A \rightarrow b \bullet Ce, \ \$$
$B \rightarrow \bullet f, \quad d$
$C \rightarrow \bullet f, \quad e$

I_4
$A \rightarrow aB \bullet e, \ \$$

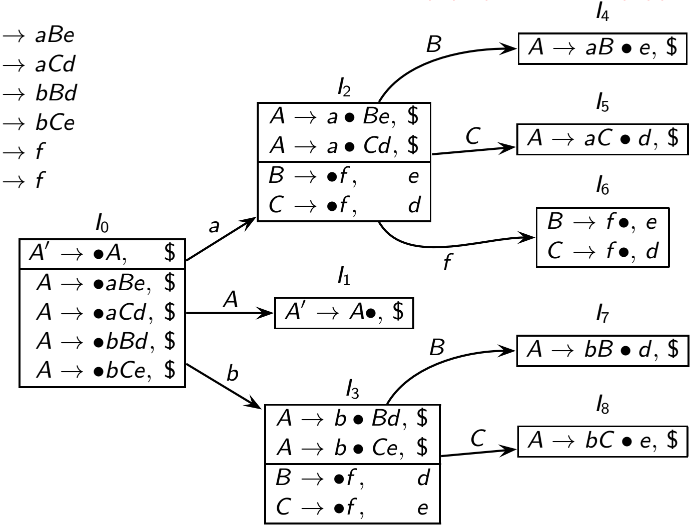
I_5
$A \rightarrow aC \bullet d, \ \$$

I_6
$B \rightarrow f\bullet, \ e$
$C \rightarrow f\bullet, \ d$

I_7
$A \rightarrow bB \bullet d, \ \$$

Another Example of LR(1) (aka CLR(1)) Parsing

$A \rightarrow aBe$
$A \rightarrow aCd$
$A \rightarrow bBd$
$A \rightarrow bCe$
$B \rightarrow f$
$C \rightarrow f$



I_0
$A' \rightarrow \bullet A, \quad \$$
$A \rightarrow \bullet aBe, \ \$$
$A \rightarrow \bullet aCd, \ \$$
$A \rightarrow \bullet bBd, \ \$$
$A \rightarrow \bullet bCe, \ \$$

a

A

b

I_1
$A' \rightarrow A\bullet, \ \$$

I_2
$A \rightarrow a \bullet Be, \ \$$
$A \rightarrow a \bullet Cd, \ \$$
$B \rightarrow \bullet f, \quad e$
$C \rightarrow \bullet f, \quad d$

B

C

f

I_3
$A \rightarrow b \bullet Bd, \ \$$
$A \rightarrow b \bullet Ce, \ \$$
$B \rightarrow \bullet f, \quad d$
$C \rightarrow \bullet f, \quad e$

B

C

I_4
$A \rightarrow aB \bullet e, \ \$$

I_5
$A \rightarrow aC \bullet d, \ \$$

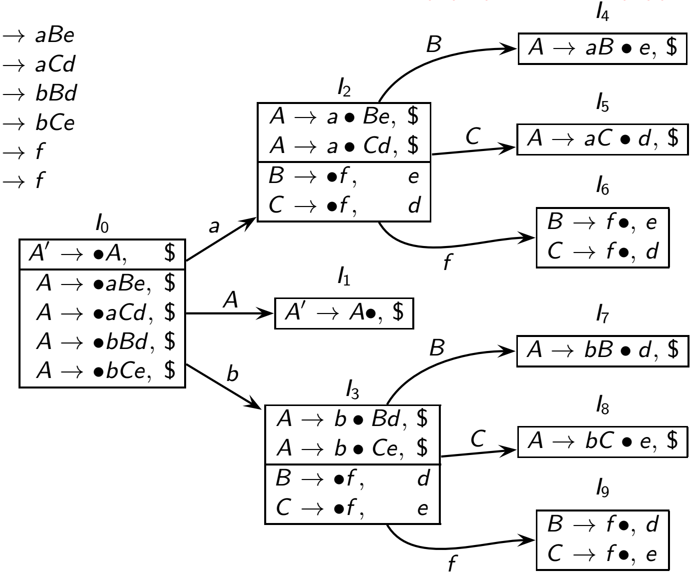
I_6
$B \rightarrow f\bullet, \ e$
$C \rightarrow f\bullet, \ d$

I_7
$A \rightarrow bB \bullet d, \ \$$

I_8
$A \rightarrow bC \bullet e, \ \$$

Another Example of LR(1) (aka CLR(1)) Parsing

$A \rightarrow aBe$
$A \rightarrow aCd$
$A \rightarrow bBd$
$A \rightarrow bCe$
$B \rightarrow f$
$C \rightarrow f$



I_0
$A' \rightarrow \bullet A, \ \$$
$A \rightarrow \bullet aBe, \ \$$
$A \rightarrow \bullet aCd, \ \$$
$A \rightarrow \bullet bBd, \ \$$
$A \rightarrow \bullet bCe, \ \$$

I_1
$A' \rightarrow A \bullet, \ \$$

I_2
$A \rightarrow a \bullet Be, \ \$$
$A \rightarrow a \bullet Cd, \ \$$
$B \rightarrow \bullet f, \ e$
$C \rightarrow \bullet f, \ d$

I_3
$A \rightarrow b \bullet Bd, \ \$$
$A \rightarrow b \bullet Ce, \ \$$
$B \rightarrow \bullet f, \ d$
$C \rightarrow \bullet f, \ e$

I_4
$A \rightarrow aB \bullet e, \ \$$

I_5
$A \rightarrow aC \bullet d, \ \$$

I_6
$B \rightarrow f \bullet, \ e$
$C \rightarrow f \bullet, \ d$

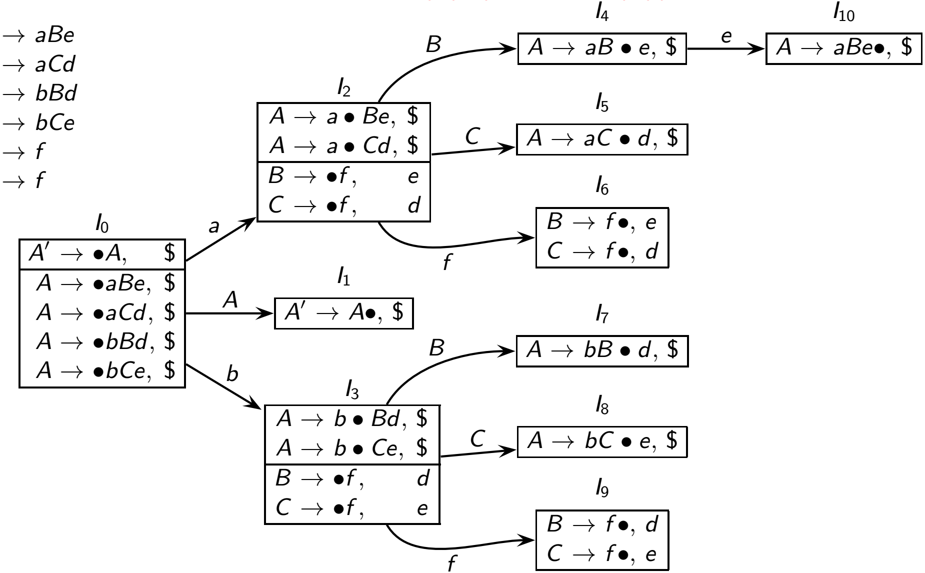
I_7
$A \rightarrow bB \bullet d, \ \$$

I_8
$A \rightarrow bC \bullet e, \ \$$

I_9
$B \rightarrow f \bullet, \ d$
$C \rightarrow f \bullet, \ e$

Another Example of LR(1) (aka CLR(1)) Parsing

$A \rightarrow aBe$
$A \rightarrow aCd$
$A \rightarrow bBd$
$A \rightarrow bCe$
$B \rightarrow f$
$C \rightarrow f$



I_0
$A' \rightarrow \bullet A,\ \$$
$A \rightarrow \bullet aBe,\ \$$
$A \rightarrow \bullet aCd,\ \$$
$A \rightarrow \bullet bBd,\ \$$
$A \rightarrow \bullet bCe,\ \$$

I_1
$A' \rightarrow A\bullet,\ \$$

I_2
$A \rightarrow a \bullet Be,\ \$$
$A \rightarrow a \bullet Cd,\ \$$
$B \rightarrow \bullet f,\ e$
$C \rightarrow \bullet f,\ d$

I_3
$A \rightarrow b \bullet Bd,\ \$$
$A \rightarrow b \bullet Ce,\ \$$
$B \rightarrow \bullet f,\ d$
$C \rightarrow \bullet f,\ e$

I_4
$A \rightarrow aB \bullet e,\ \$$

I_5
$A \rightarrow aC \bullet d,\ \$$

I_6
$B \rightarrow f\bullet,\ e$
$C \rightarrow f\bullet,\ d$

I_7
$A \rightarrow bB \bullet d,\ \$$

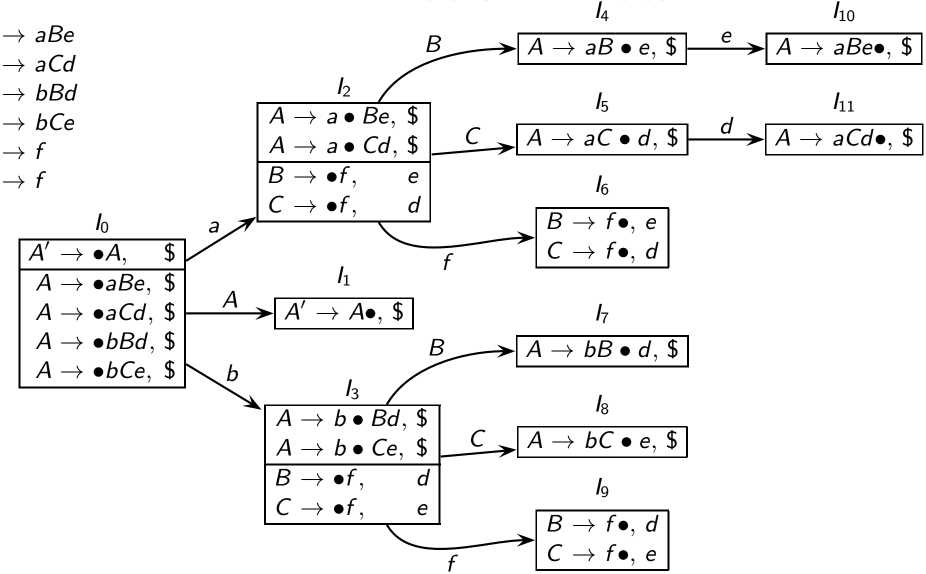
I_8
$A \rightarrow bC \bullet e,\ \$$

I_9
$B \rightarrow f\bullet,\ d$
$C \rightarrow f\bullet,\ e$

I_{10}
$A \rightarrow aBe\bullet,\ \$$

Another Example of LR(1) (aka CLR(1)) Parsing

$A \rightarrow aBe$
$A \rightarrow aCd$
$A \rightarrow bBd$
$A \rightarrow bCe$
$B \rightarrow f$
$C \rightarrow f$



I_0
$A' \rightarrow \bullet A, \quad \$$
$A \rightarrow \bullet aBe, \ \$$
$A \rightarrow \bullet aCd, \ \$$
$A \rightarrow \bullet bBd, \ \$$
$A \rightarrow \bullet bCe, \ \$$

I_1
$A' \rightarrow A\bullet, \ \$$

I_2
$A \rightarrow a \bullet Be, \ \$$
$A \rightarrow a \bullet Cd, \ \$$
$B \rightarrow \bullet f, \quad e$
$C \rightarrow \bullet f, \quad d$

I_3
$A \rightarrow b \bullet Bd, \ \$$
$A \rightarrow b \bullet Ce, \ \$$
$B \rightarrow \bullet f, \quad d$
$C \rightarrow \bullet f, \quad e$

I_4
$A \rightarrow aB \bullet e, \ \$$

I_5
$A \rightarrow aC \bullet d, \ \$$

I_6
$B \rightarrow f\bullet, \ e$
$C \rightarrow f\bullet, \ d$

I_7
$A \rightarrow bB \bullet d, \ \$$

I_8
$A \rightarrow bC \bullet e, \ \$$

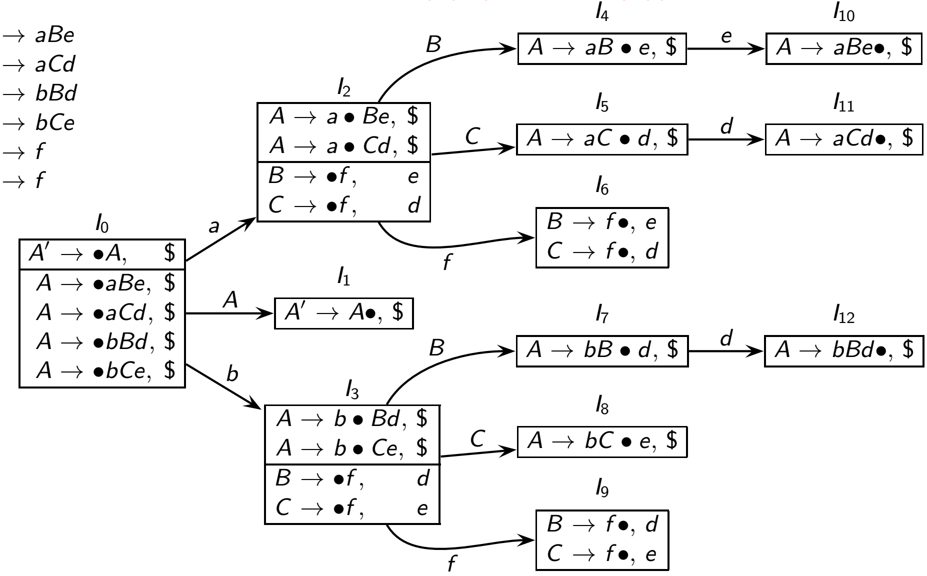
I_9
$B \rightarrow f\bullet, \ d$
$C \rightarrow f\bullet, \ e$

I_{10}
$A \rightarrow aBe\bullet, \ \$$

I_{11}
$A \rightarrow aCd\bullet, \ \$$

Another Example of LR(1) (aka CLR(1)) Parsing

$A \rightarrow aBe$
$A \rightarrow aCd$
$A \rightarrow bBd$
$A \rightarrow bCe$
$B \rightarrow f$
$C \rightarrow f$



I_0: $A' \rightarrow \bullet A,\ \$$ | $A \rightarrow \bullet aBe,\ \$$ | $A \rightarrow \bullet aCd,\ \$$ | $A \rightarrow \bullet bBd,\ \$$ | $A \rightarrow \bullet bCe,\ \$$

I_1: $A' \rightarrow A\bullet,\ \$$

I_2: $A \rightarrow a \bullet Be,\ \$$ | $A \rightarrow a \bullet Cd,\ \$$ | $B \rightarrow \bullet f,\ e$ | $C \rightarrow \bullet f,\ d$

I_3: $A \rightarrow b \bullet Bd,\ \$$ | $A \rightarrow b \bullet Ce,\ \$$ | $B \rightarrow \bullet f,\ d$ | $C \rightarrow \bullet f,\ e$

I_4: $A \rightarrow aB \bullet e,\ \$$

I_5: $A \rightarrow aC \bullet d,\ \$$

I_6: $B \rightarrow f\bullet,\ e$ | $C \rightarrow f\bullet,\ d$

I_7: $A \rightarrow bB \bullet d,\ \$$

I_8: $A \rightarrow bC \bullet e,\ \$$

I_9: $B \rightarrow f\bullet,\ d$ | $C \rightarrow f\bullet,\ e$

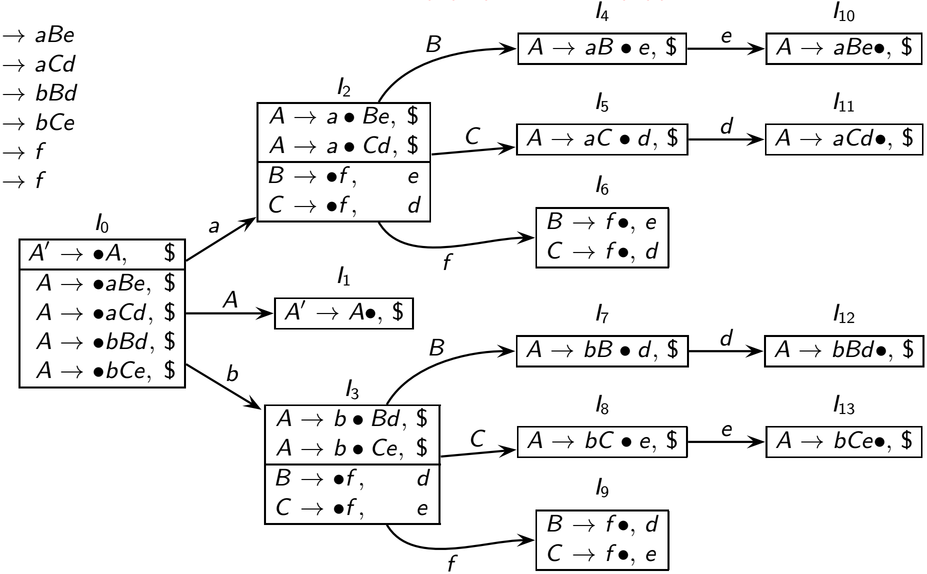
I_{10}: $A \rightarrow aBe\bullet,\ \$$

I_{11}: $A \rightarrow aCd\bullet,\ \$$

I_{12}: $A \rightarrow bBd\bullet,\ \$$

Another Example of LR(1) (aka CLR(1)) Parsing

$A \rightarrow aBe$
$A \rightarrow aCd$
$A \rightarrow bBd$
$A \rightarrow bCe$
$B \rightarrow f$
$C \rightarrow f$



I_0:
$A' \rightarrow \bullet A, \quad \$$
$A \rightarrow \bullet aBe, \quad \$$
$A \rightarrow \bullet aCd, \quad \$$
$A \rightarrow \bullet bBd, \quad \$$
$A \rightarrow \bullet bCe, \quad \$$

I_1: $A' \rightarrow A\bullet, \ \$$

I_2:
$A \rightarrow a \bullet Be, \ \$$
$A \rightarrow a \bullet Cd, \ \$$
$B \rightarrow \bullet f, \quad e$
$C \rightarrow \bullet f, \quad d$

I_3:
$A \rightarrow b \bullet Bd, \ \$$
$A \rightarrow b \bullet Ce, \ \$$
$B \rightarrow \bullet f, \quad d$
$C \rightarrow \bullet f, \quad e$

I_4: $A \rightarrow aB \bullet e, \ \$$

I_5: $A \rightarrow aC \bullet d, \ \$$

I_6:
$B \rightarrow f\bullet, \ e$
$C \rightarrow f\bullet, \ d$

I_7: $A \rightarrow bB \bullet d, \ \$$

I_8: $A \rightarrow bC \bullet e, \ \$$

I_9:
$B \rightarrow f\bullet, \ d$
$C \rightarrow f\bullet, \ e$

I_{10}: $A \rightarrow aBe\bullet, \ \$$

I_{11}: $A \rightarrow aCd\bullet, \ \$$

I_{12}: $A \rightarrow bBd\bullet, \ \$$

I_{13}: $A \rightarrow bCe\bullet, \ \$$

LALR(1) Parsing

LALR(1) Parsing

- Merge item sets with identical cores (may have different lookaheads)

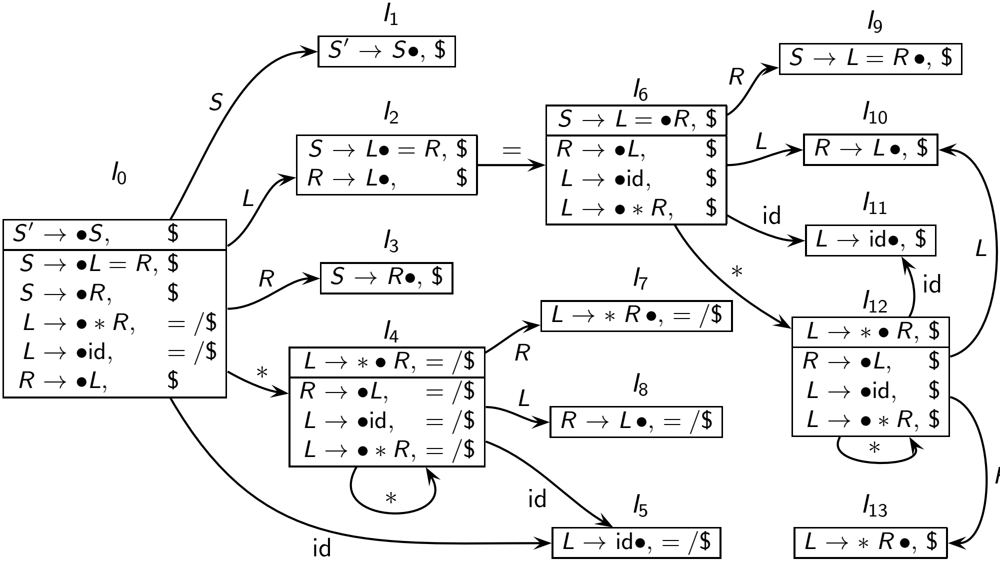
  States I_i : $A \rightarrow \alpha \bullet \beta, a$ and I_j : $A \rightarrow \alpha \bullet \beta, b$

  can be merged to create a new state I_{ij} : $A \rightarrow \alpha \bullet \beta, a/b$

- In practice, we do not construct LR(1) items to construct LALR(1) parser

  We construct LR(0) items and use a look-ahead propagation algorithm

LALR(1) Parsing for Pointer Assignment Grammar



I_0
$S' \to \bullet S,\ \$$
$S \to \bullet L = R,\ \$$
$S \to \bullet R,\ \$$
$L \to \bullet * R,\ =/\$$
$L \to \bullet \text{id},\ =/\$$
$R \to \bullet L,\ \$$

I_1
$S' \to S \bullet,\ \$$

I_2
$S \to L \bullet = R,\ \$$
$R \to L \bullet,\ \$$

I_3
$S \to R \bullet,\ \$$

I_4
$L \to * \bullet R,\ =/\$$
$R \to \bullet L,\ =/\$$
$L \to \bullet \text{id},\ =/\$$
$L \to \bullet * R,\ =/\$$

I_5
$L \to \text{id} \bullet,\ =/\$$

I_6
$S \to L = \bullet R,\ \$$
$R \to \bullet L,\ \$$
$L \to \bullet \text{id},\ \$$
$L \to \bullet * R,\ \$$

I_7
$L \to * R \bullet,\ =/\$$

I_8
$R \to L \bullet,\ =/\$$

I_9
$S \to L = R \bullet,\ \$$

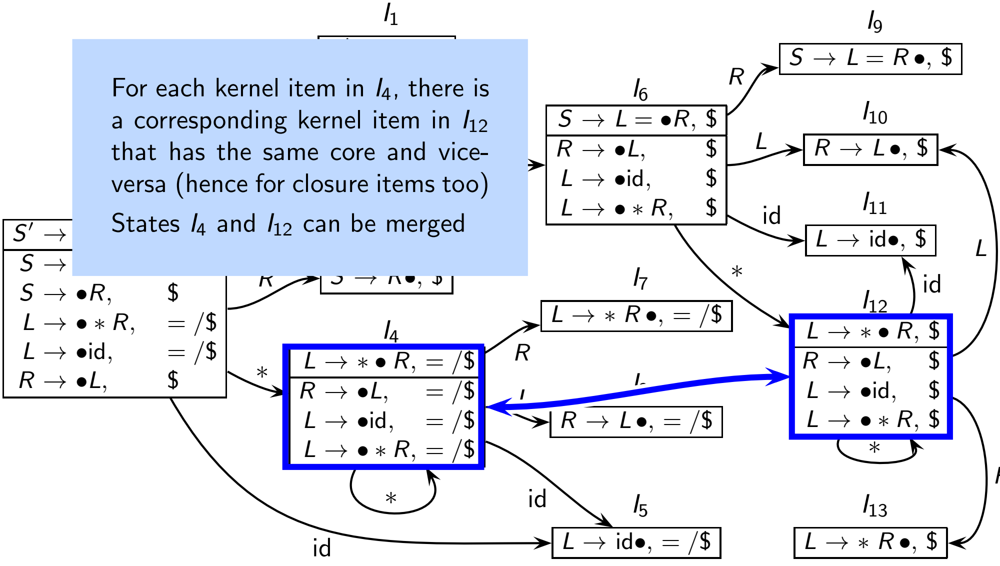
I_{10}
$R \to L \bullet,\ \$$

I_{11}
$L \to \text{id} \bullet,\ \$$

I_{12}
$L \to * \bullet R,\ \$$
$R \to \bullet L,\ \$$
$L \to \bullet \text{id},\ \$$
$L \to \bullet * R,\ \$$

I_{13}
$L \to * R \bullet,\ \$$

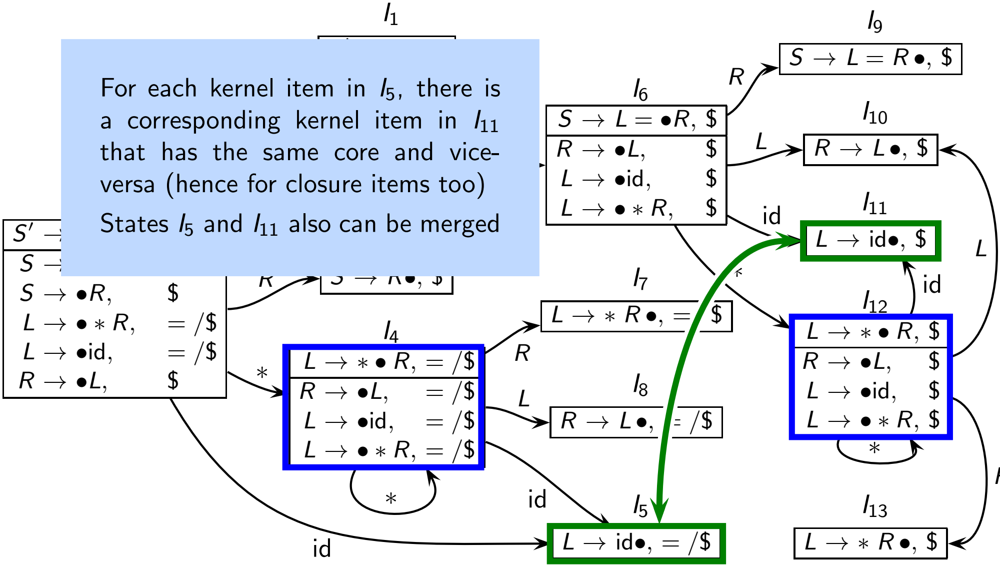
LALR(1) Parsing for Pointer Assignment Grammar



For each kernel item in I_4, there is a corresponding kernel item in I_{12} that has the same core and vice-versa (hence for closure items too)

States I_4 and I_{12} can be merged

LALR(1) Parsing for Pointer Assignment Grammar



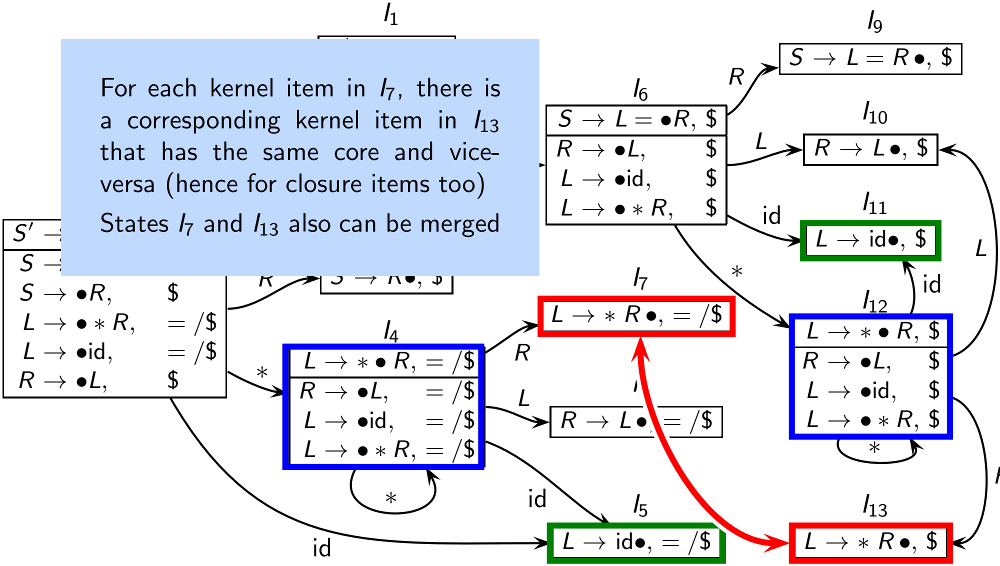
For each kernel item in I_5, there is a corresponding kernel item in I_{11} that has the same core and vice-versa (hence for closure items too)

States I_5 and I_{11} also can be merged

LALR(1) Parsing for Pointer Assignment Grammar

For each kernel item in I_7, there is a corresponding kernel item in I_{13} that has the same core and vice-versa (hence for closure items too)

States I_7 and I_{13} also can be merged



I_1

$S' \rightarrow$

$S \rightarrow$

$S \rightarrow \bullet R,\ \$$

$L \rightarrow \bullet * R,\ =/\$$

$L \rightarrow \bullet \text{id},\ =/\$$

$R \rightarrow \bullet L,\ \$$

$S \rightarrow R\bullet,\ \$$

I_4

$L \rightarrow * \bullet R,\ =/\$$

$R \rightarrow \bullet L,\ =/\$$

$L \rightarrow \bullet \text{id},\ =/\$$

$L \rightarrow \bullet * R,\ =/\$$

I_5

$L \rightarrow \text{id}\bullet,\ =/\$$

I_6

$S \rightarrow L = \bullet R,\ \$$

$R \rightarrow \bullet L,\ \$$

$L \rightarrow \bullet \text{id},\ \$$

$L \rightarrow \bullet * R,\ \$$

I_7

$L \rightarrow * R\bullet,\ =/\$$

$R \rightarrow L\bullet,\ =/\$$

I_9

$S \rightarrow L = R\bullet,\ \$$

I_{10}

$R \rightarrow L\bullet,\ \$$

I_{11}

$L \rightarrow \text{id}\bullet,\ \$$

I_{12}

$L \rightarrow * \bullet R,\ \$$

$R \rightarrow \bullet L,\ \$$

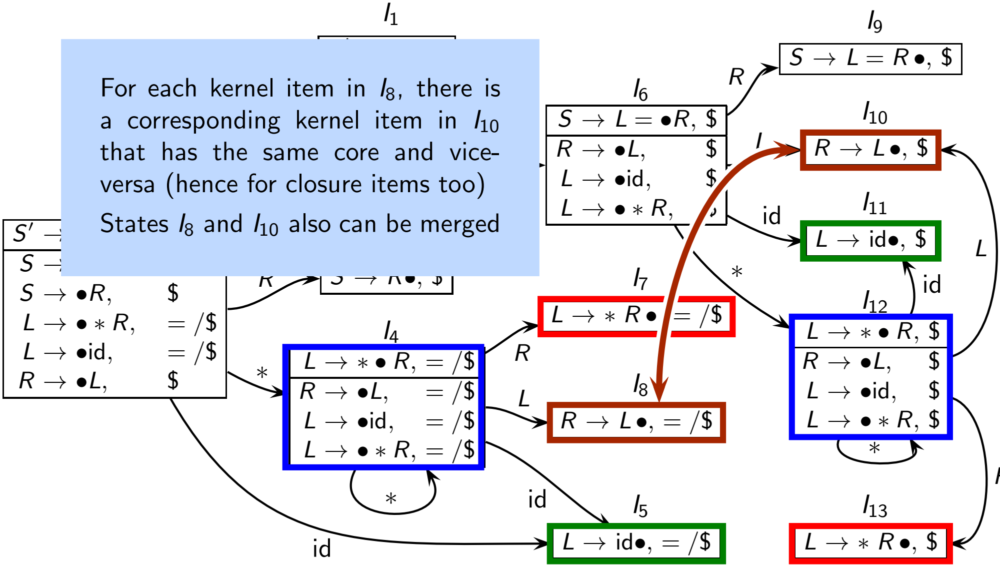
$L \rightarrow \bullet \text{id},\ \$$

$L \rightarrow \bullet * R,\ \$$

I_{13}

$L \rightarrow * R\bullet,\ \$$

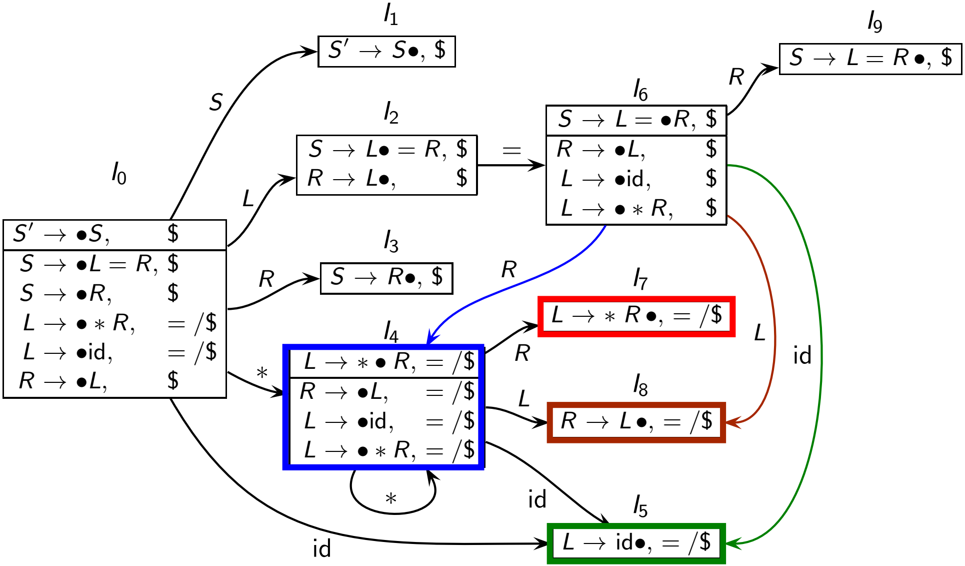
LALR(1) Parsing for Pointer Assignment Grammar



For each kernel item in I_8, there is a corresponding kernel item in I_{10} that has the same core and vice-versa (hence for closure items too)

States I_8 and I_{10} also can be merged

LALR(1) Parsing for Pointer Assignment Grammar



I_0

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

$S \to \bullet L = R,\ \$$

$S \to \bullet R,\ \$$

$L \to \bullet * R,\ =/\$$

$L \to \bullet \text{id},\ =/\$$

$R \to \bullet L,\ \$$

I_1

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

I_2

$S \to L \bullet = R,\ \$$

$R \to L \bullet,\ \$$

I_3

$S \to R \bullet,\ \$$

I_4

$L \to * \bullet R,\ =/\$$

$R \to \bullet L,\ =/\$$

$L \to \bullet \text{id},\ =/\$$

$L \to \bullet * R,\ =/\$$

I_5

$L \to \text{id} \bullet,\ =/\$$

I_6

$S \to L = \bullet R,\ \$$

$R \to \bullet L,\ \$$

$L \to \bullet \text{id},\ \$$

$L \to \bullet * R,\ \$$

I_7

$L \to * R \bullet,\ =/\$$

I_8

$R \to L \bullet,\ =/\$$

I_9

$S \to L = R \bullet,\ \$$

LALR(1) Parsing Table for Pointer Assignment Grammar

0 $S' \rightarrow S$
1 $S \rightarrow L = R$
2 $S \rightarrow R$
3 $L \rightarrow * R$
4 $L \rightarrow \text{id}$
5 $R \rightarrow L$

State	Action				Goto		
	`id`	*	=	$	S	L	R
0	$s5$	$s4$			$c1$	$c2$	$c3$
1				acc			
2			$s6$	$r5$			
3				$r2$			
4	$s5$	$s4$				$c8$	$c7$
5			$r4$	$r4$			
6	$s5$	$s4$				$c8$	$c9$
7			$r3$	$r3$			
8			$r5$	$r5$			
9				$r1$			

LALR(1) Vs CLR(1) Parsing

- Can merging of LR(1) states introduce shift-reduce conflict?
- Can merging of LR(1) states introduce reduce-reduce conflict?

Can Merging LR(1) Sets of Items Introduce Shift-Reduce Conflict?

- To merge states I_i and I_j, they should have identical cores but different lookaheads (if the lookaheads are same then the states will not be distinct)

Can Merging LR(1) Sets of Items Introduce Shift-Reduce Conflict?

- To merge states I_i and I_j, they should have identical cores but different lookaheads (if the lookaheads are same then the states will not be distinct)
- Let I_i : $\begin{array}{ll} A \rightarrow \alpha \bullet a\beta, & p \\ B \rightarrow \gamma\bullet, & q \end{array}$ and I_j : $\begin{array}{ll} A \rightarrow \alpha \bullet a\beta, & r \\ B \rightarrow \gamma\bullet, & s \end{array}$ where p, q, r, s are arbitrary terminals

  So that the merged state is I_{ij} : $\begin{array}{ll} A \rightarrow \alpha \bullet a\beta, & p/r \\ B \rightarrow \gamma\bullet, & q/s \end{array}$

Can Merging LR(1) Sets of Items Introduce Shift-Reduce Conflict?

- To merge states I_i and I_j, they should have identical cores but different lookaheads (if the lookaheads are same then the states will not be distinct)
- Let I_i : $\begin{array}{ll} A \rightarrow \alpha \bullet a\beta, & p \\ B \rightarrow \gamma\bullet, & q \end{array}$ and I_j : $\begin{array}{ll} A \rightarrow \alpha \bullet a\beta, & r \\ B \rightarrow \gamma\bullet, & s \end{array}$ where p, q, r, s are arbitrary terminals

  So that the merged state is I_{ij} : $\begin{array}{ll} A \rightarrow \alpha \bullet a\beta, & p/r \\ B \rightarrow \gamma\bullet, & q/s \end{array}$
- For a shift-reduce conflict in I_{ij}, either q or s must be a.

Can Merging LR(1) Sets of Items Introduce Shift-Reduce Conflict?

- To merge states I_i and I_j, they should have identical cores but different lookaheads (if the lookaheads are same then the states will not be distinct)
- Let I_i : $\begin{array}{ll} A \to \alpha \bullet a\beta, & p \\ B \to \gamma\bullet, & q \end{array}$ and I_j : $\begin{array}{ll} A \to \alpha \bullet a\beta, & r \\ B \to \gamma\bullet, & s \end{array}$ where p, q, r, s are arbitrary terminals

  So that the merged state is I_{ij} : $\begin{array}{ll} A \to \alpha \bullet a\beta, & p/r \\ B \to \gamma\bullet, & q/s \end{array}$
- For a shift-reduce conflict in I_{ij}, either q or s must be a.
  - If q is a, then I_i is $\begin{array}{ll} A \to \alpha \bullet a\beta, & p \\ B \to \gamma\bullet, & a \end{array}$ and thus I_i has a shift-reduce conflict

Can Merging LR(1) Sets of Items Introduce Shift-Reduce Conflict?

- To merge states I_i and I_j, they should have identical cores but different lookaheads (if the lookaheads are same then the states will not be distinct)
- Let I_i : $\begin{array}{ll} A \rightarrow \alpha \bullet a\beta, & p \\ B \rightarrow \gamma\bullet, & q \end{array}$ and I_j : $\begin{array}{ll} A \rightarrow \alpha \bullet a\beta, & r \\ B \rightarrow \gamma\bullet, & s \end{array}$ where p, q, r, s are arbitrary terminals

  So that the merged state is I_{ij} : $\begin{array}{ll} A \rightarrow \alpha \bullet a\beta, & p/r \\ B \rightarrow \gamma\bullet, & q/s \end{array}$
- For a shift-reduce conflict in I_{ij}, either q or s must be a.
  - If q is a, then I_i is $\begin{array}{ll} A \rightarrow \alpha \bullet a\beta, & p \\ B \rightarrow \gamma\bullet, & a \end{array}$ and thus I_i has a shift-reduce conflict
  - If s is a, then I_j is $\begin{array}{ll} A \rightarrow \alpha \bullet a\beta, & r \\ B \rightarrow \gamma\bullet, & a \end{array}$ and thus I_j has a shift-reduce conflict

Can Merging LR(1) Sets of Items Introduce Shift-Reduce Conflict?

- To merge states I_i and I_j, they should have identical cores but different lookaheads (if the lookaheads are same then the states will not be distinct)
- Let I_i : A B , s are arbitrary ter
  So that the
- For a shift-r
  - If q is $B \to \gamma \bullet, \quad a$ -reduce conflict
  - If s is a, then I_j is $\begin{array}{ll} A \to \alpha \bullet a\beta, & r \\ B \to \gamma \bullet, & a \end{array}$ and thus I_j has a shift-reduce conflict

A set I_{ij} of items in an LALR(1) parser can have a shift-reduce conflict *if and only if* a set I_i of LR(1) items merged to form I_{ij} has the same shift-reduce conflict

This is because a shift-reduce conflict depends both on a lookahead and a terminal in the core of an item

Can Merging LR(1) Sets of Items Introduce Reduce-Reduce Conflict?

- Let I_i : $\begin{array}{ll} A \rightarrow \alpha\bullet, & p \\ B \rightarrow \alpha\bullet, & q \end{array}$ and I_j : $\begin{array}{ll} A \rightarrow \alpha\bullet, & r \\ B \rightarrow \alpha\bullet, & s \end{array}$

  So that the merged state is I_{ij} : $\begin{array}{ll} A \rightarrow \alpha\bullet, & p/r \\ B \rightarrow \alpha\bullet, & q/s \end{array}$

Can Merging LR(1) Sets of Items Introduce Reduce-Reduce Conflict?

- Let I_i : $\begin{array}{ll} A \rightarrow \alpha\bullet, & p \\ B \rightarrow \alpha\bullet, & q \end{array}$ and I_j : $\begin{array}{ll} A \rightarrow \alpha\bullet, & r \\ B \rightarrow \alpha\bullet, & s \end{array}$

  So that the merged state is I_{ij} : $\begin{array}{ll} A \rightarrow \alpha\bullet, & p/r \\ B \rightarrow \alpha\bullet, & q/s \end{array}$
- For a reduce-reduce conflict in I_{ij} such that there is no reduce-reduce conflict in I_i or I_j,
  - $p = s$. This is possible without a reduce-reduce conflict in I_i and I_j
  - $r = q$. This is also possible without a reduce-reduce conflict in I_i and I_j

Can Merging LR(1) Sets of Items Introduce Reduce-Reduce Conflict?

- Let I_i : A
  B

  So that the

- For a reduc ce conflict in I_i or I_j,
  - $p = s$. and I_j
  - $r = q$. This is also possible without a reduce-reduce conflict in I_i and I_j

Merging LR(1) sets of items can introduce reduce-reduce conflicts *even* if the original sets do not have a reduce-reduce conflict

This is because a reduce-reduce conflict depends only on lookaheads and a complete item. The terminals in a core do not play any role

LALR(1) Vs LR(1) Parsing

- Merging of LR(1) states for LALR(1) parsing cannot introduce shift-reduce conflicts
- Merging of LR(1) states for LALR(1) parsing may introduce reduce-reduce conflicts

LALR(1) Vs LR(1) Parsing

- Merging of LR(1) states for LALR(1) parsing cannot introduce shift-reduce conflicts
- Merging of LR(1) states for LALR(1) parsing may introduce reduce-reduce conflicts
- Let $\mathbb{G}(P)$ be the set of grammars admitted by a parsing method P (i.e. conflict-free parsers can be created for these grammars using P)

  Then, $\mathbb{G}(LALR(1)) \subset \mathbb{G}(LR(1))$

LALR(1) Vs LR(1) Parsing

- Merging of LR(1) states for LALR(1) parsing cannot introduce shift-reduce conflicts
- Merging of LR(1) states for LALR(1) parsing may introduce reduce-reduce conflicts
- Let $\mathbb{G}(P)$ be the set of grammars admitted by a parsing method P (i.e. conflict-free parsers can be created for these grammars using P)
  Then, $\mathbb{G}(LALR(1)) \subset \mathbb{G}(LR(1))$
- Consider a grammar $G \in \mathbb{G}(LALR(1))$
  - Can an LALR(1) parser for G reject $w \in L(G)$ because of merging of states?
  - Can an LALR(1) parser for G accept $w' \notin L(G)$ because of merging of states?

LALR(1) Vs LR(1) Parsing

- Merging of LR(1) states for LALR(1) parsing cannot introduce shift-reduce conflicts
- Merging of LR(1) states for LALR(1) parsing may introduce reduce-reduce conflicts
- Let $\mathbb{G}(P)$ be the set of grammars admitted by a parsing method P (i.e. conflict-free parsers can be created for these grammars using P)

  Then, $\mathbb{G}(LALR(1)) \subset \mathbb{G}(LR(1))$
- Consider a grammar $G \in \mathbb{G}(LALR(1))$
  - Can an LALR(1) parser for G reject $w \in L(G)$ because of merging of states? No
  - Can an LALR(1) parser for G accept $w' \notin L(G)$ because of merging of states? No

If a parsing method admits a grammar G then the corresponding parser for G accepts all sentences in $L(G)$ and rejects all sentences not in $L(G)$

LALR(1) Vs LR(1) Parsing

- Merging of LR(1) states for LALR(1) parsing cannot introduce shift-reduce conflicts
- Merging of LR(1) states for LALR(1) parsing may introduce reduce-reduce conflicts

Deterministic parsers: LL(1), LR(0), SLR(1), LR(1), LALR(1), etc.

- Let $\mathbb{G}(P)$ be the set of grammars admitted by a parsing method P (i.e. conflict-free parsers can be created for these grammars using P)

  Then, $\mathbb{G}(LALR(1)) \subset \mathbb{G}(LR(1))$
- Consider a grammar $G \in \mathbb{G}(LALR(1))$
  - Can an LALR(1) parser for G reject $w \in L(G)$ because of merging of states? No
  - Can an LALR(1) parser for G accept $w' \notin L(G)$ because of merging of states? No

  For deterministic parsers (LL(1), LR(1), etc.), G must be conflict-free.

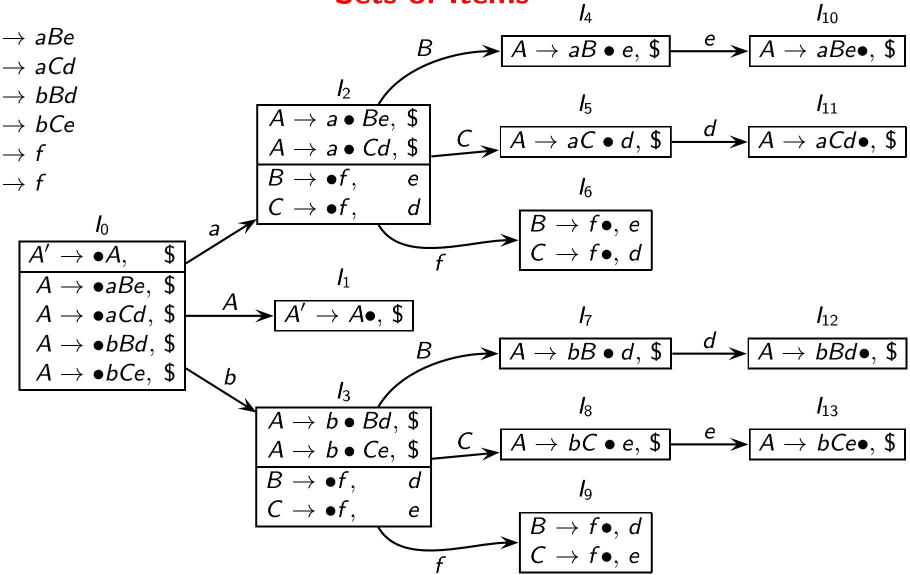
  If a parsing method admits a grammar G then the corresponding parser for G accepts all sentences in $L(G)$ and rejects all sentences not in $L(G)$
- Consider a grammar $G \notin \mathbb{G}(LALR(1))$

  An LALR(1) parser may still accept $L(G)$ because it may admit G' such that $L(G) = L(G')$

this principle applies to other parser types as well, though with some important distinctions

Example of Reduce-Reduce Conflict Caused by Merging LR(1) Sets of Items

$$A \to aBe$$
$$A \to aCd$$
$$A \to bBd$$
$$A \to bCe$$
$$B \to f$$
$$C \to f$$



I_0
$A' \to \bullet A, \; \$$
$A \to \bullet aBe, \; \$$
$A \to \bullet aCd, \; \$$
$A \to \bullet bBd, \; \$$
$A \to \bullet bCe, \; \$$

I_1
$A' \to A\bullet, \; \$$

I_2
$A \to a \bullet Be, \; \$$
$A \to a \bullet Cd, \; \$$
$B \to \bullet f, \; e$
$C \to \bullet f, \; d$

I_3
$A \to b \bullet Bd, \; \$$
$A \to b \bullet Ce, \; \$$
$B \to \bullet f, \; d$
$C \to \bullet f, \; e$

I_4
$A \to aB \bullet e, \; \$$

I_5
$A \to aC \bullet d, \; \$$

I_6
$B \to f\bullet, \; e$
$C \to f\bullet, \; d$

I_7
$A \to bB \bullet d, \; \$$

I_8
$A \to bC \bullet e, \; \$$

I_9
$B \to f\bullet, \; d$
$C \to f\bullet, \; e$

I_{10}
$A \to aBe\bullet, \; \$$

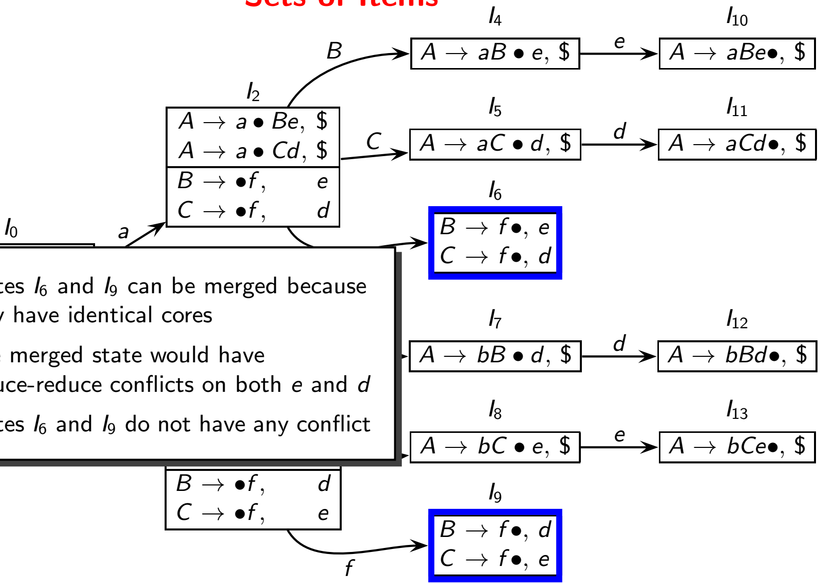
I_{11}
$A \to aCd\bullet, \; \$$

I_{12}
$A \to bBd\bullet, \; \$$

I_{13}
$A \to bCe\bullet, \; \$$

Example of Reduce-Reduce Conflict Caused by Merging LR(1) Sets of Items

$A \rightarrow aBe$
$A \rightarrow aCd$
$A \rightarrow bBd$
$A \rightarrow bCe$
$B \rightarrow f$
$C \rightarrow f$



I_0 $\xrightarrow{a}$ I_2: $A \rightarrow a \bullet Be,\ \$$; $A \rightarrow a \bullet Cd,\ \$$; $B \rightarrow \bullet f,\ e$; $C \rightarrow \bullet f,\ d$

$I_2 \xrightarrow{B} I_4$: $A \rightarrow aB \bullet e,\ \$$ $\xrightarrow{e}$ I_{10}: $A \rightarrow aBe \bullet,\ \$$

$I_2 \xrightarrow{C} I_5$: $A \rightarrow aC \bullet d,\ \$$ $\xrightarrow{d}$ I_{11}: $A \rightarrow aCd \bullet,\ \$$

$I_2 \rightarrow I_6$: $B \rightarrow f \bullet,\ e$; $C \rightarrow f \bullet,\ d$

I_7: $A \rightarrow bB \bullet d,\ \$$ $\xrightarrow{d}$ I_{12}: $A \rightarrow bBd \bullet,\ \$$

I_8: $A \rightarrow bC \bullet e,\ \$$ $\xrightarrow{e}$ I_{13}: $A \rightarrow bCe \bullet,\ \$$

$B \rightarrow \bullet f,\ d$; $C \rightarrow \bullet f,\ e$ $\xrightarrow{f}$ I_9: $B \rightarrow f \bullet,\ d$; $C \rightarrow f \bullet,\ e$

- States I_6 and I_9 can be merged because they have identical cores
- The merged state would have reduce-reduce conflicts on both e and d
- States I_6 and I_9 do not have any conflict

A Practical Example of Reduce-Reduce Conflict in LR(1) Parsing

$$\begin{array}{rcl} \text{program} & \rightarrow & \text{func\_decl var\_decl} \\ \text{program} & \rightarrow & \text{var\_decl func\_decl} \\ \text{var\_decl} & \rightarrow & \text{data\_type ID ;} \\ \text{data\_type} & \rightarrow & \text{INT} \\ \text{func\_decl} & \rightarrow & \text{return\_type ID ( )} \\ \text{return\_type} & \rightarrow & \text{INT} \\ \text{return\_type} & \rightarrow & \text{VOID} \end{array}$$

For the input “int f . . .”, when we see the token INT, the next token is ID

In this situation, the parser does not know if it should reduce INT to return_type or data_type

State I_0 contains the following items

$$\begin{array}{rcl} \text{data\_type} & \rightarrow & \bullet\ \text{INT, ID} \\ \text{return\_type} & \rightarrow & \bullet\ \text{INT, ID} \end{array}$$

The transition on INT gives the following set of items showing a reduce-reduce conflict on ID

$$\begin{array}{rcl} \text{data\_type} & \rightarrow & \text{INT}\ \bullet\text{, ID} \\ \text{return\_type} & \rightarrow & \text{INT}\ \bullet\text{, ID} \end{array}$$

A Practical Example of Reduce-Reduce Conflict in LR(1) Parsing

In this particular case, the conflict can be removed by replacing every occurrence of the non-terminals data_type and return_type by every RHS of the non-terminal

Original Grammar	Transformed Grammar
program → func_decl var_decl	program → func_decl var_decl
program → var_decl func_decl	program → var_decl func_decl
var_decl → data_type ID ;	var_decl → INT ID ;
data_type → INT	func_decl → INT ID ( )
func_decl → return_type ID ( )	func_decl → VOID ID ( )
return_type → INT	
return_type → VOID	

A Summary of Bottom Up Parsing Methods

Parsing Method	Items Used	Reduction by $A \rightarrow \alpha$	Remarks
SLR(0)	LR(0)	On any terminal	
SLR(1)	LR(0)	On the terminals in FOLLOW(A)	
LR(1), also known as Canonical LR(1) or CLR(1)	LR(1)	On lookahead a in the item "$A \rightarrow \alpha\bullet, a$"	
LALR(1)	LR(1)	On lookahead a in the item "$A \rightarrow \alpha\bullet, a$"	Conceptually, the sets of items are obtained by merging LR(1) item sets that differ only in the lookahead symbols Practically, lookaheads are propagated starting from \$ on LR(0) items

Comparison of Bottom-Up Methods and Corresponding Grammars

- A grammar G is accepted by a parsing method P if a conflict-free parser can be constructed for G using P
- An ambiguous grammar is not accepted by any parsing method
- A grammar is called SLR(0), SLR(1), LR(1), or LALR(1) if it is accepted respectively, by the SLR(0), SLR(1), LR(1), or LALR(1) parsing method
    - Every SLR(0) grammar is also SLR(1) grammar but not vice-versa
    - Every SLR(1) grammar is also LALR(1) grammar but not vice-versa
    - Every LALR(1) grammar is also LR(1) grammar but not vice-versa
- The expressions grammar ($E \rightarrow E + E \mid E * E \mid \text{id}$) is not accepted by any parsing method because it is ambiguous

  (without post-facto instrumentation of parsing tables using precedences and associativities)