

Syntax Analysis

Uday Khedker

(www.cse.iitb.ac.in/~uday)

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



February 2024



Outline

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

- 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



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



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

Grammars, derivations, and parse trees



Introduction to Parsing

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

`parsing-slides-sanyal-part1.pdf`



IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

Shift Reduce Parsing

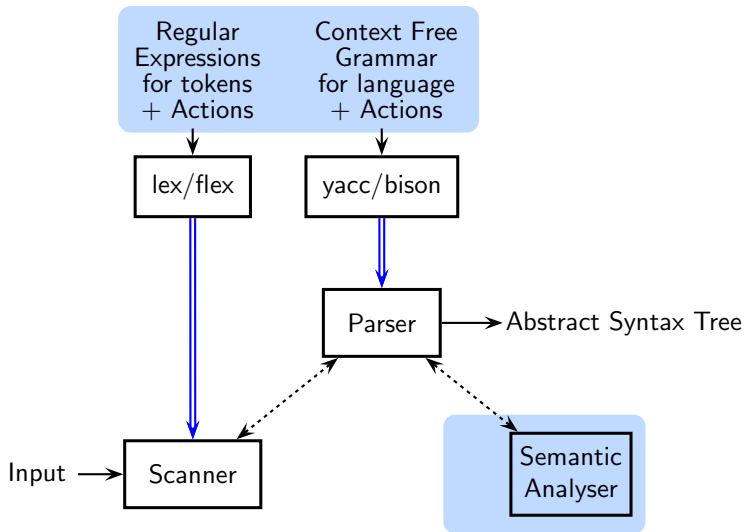
SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing

A Compiler Front End





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

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



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

Derivation

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



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

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

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T * F \mid F$$

$$F \rightarrow \text{id}$$



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

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

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T * F \mid F$$

$$F \rightarrow \text{id}$$

- A possible derivation is

$$E \Rightarrow E + T$$

$$\Rightarrow T + T$$

$$\Rightarrow F + T$$

$$\Rightarrow \text{id} + T$$

$$\Rightarrow \text{id} + T * F$$

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

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

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



Notational Conventions

Symbol type	Convention
single terminal	letters a, b, c , operators delimiters, keywords
single nonterminal	letters A, B, C and names such as <i>declaration</i> , <i>list</i> 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

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

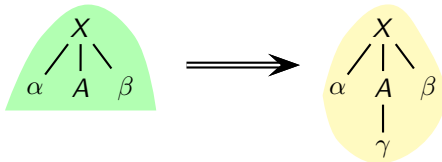
CLR(1) Parsing

LALR(1) Parsing



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
 - $\alpha_1 \xRightarrow{*} \alpha_2$ means α_1 derives α_2 in zero or more steps
 - $\alpha_1 \xRightarrow{+} \alpha_2$ means α_1 derives α_2 in one or more steps



The Language Generated by a Grammar

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

- $L(G) = \{w \mid S \xRightarrow{+} w, w \in T^*\}$, 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 $S \xRightarrow{*} \alpha$ 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 $L(G_1) = L(G_2)$



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

Sentential Forms and Sentences

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

$$G_2 \begin{cases} E \rightarrow E + E \\ E \rightarrow E * E \\ E \rightarrow \text{id} \end{cases}$$

- $L(G_1) = L(G_2)$
- $\{\text{id} + \text{id} * \text{id}, \text{id} * \text{id} + \text{id}\} \subset L(G_1)$ (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

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

- A derivation $\alpha_1 \Rightarrow \alpha_2 \Rightarrow \dots \Rightarrow \alpha_k$ is a
 - leftmost derivation, denoted $\alpha_1 \xRightarrow{lm} \alpha_2 \xRightarrow{lm} \dots \xRightarrow{lm} \alpha_k$, 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 $\alpha_1 \xRightarrow{rm} \alpha_2 \xRightarrow{rm} \dots \xRightarrow{rm} \alpha_k$, 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

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

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

Leftmost Derivation

Rightmost Derivation

E

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



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

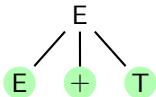
$$\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}$$

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

Leftmost Derivation

Rightmost Derivation

$$E \xRightarrow{lm} E + T$$





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

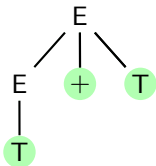
Grammar

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

Sentence: id + id * id

Leftmost Derivation

$$\begin{array}{l} E \xRightarrow{lm} E + T \\ \xRightarrow{lm} T + T \end{array}$$



Rightmost Derivation



Leftmost and Rightmost Derivations

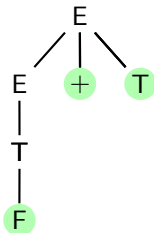
Grammar

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

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

Leftmost Derivation

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



Rightmost Derivation



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

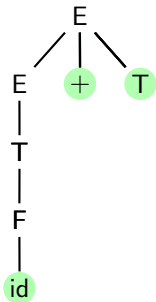
Grammar

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

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

Leftmost Derivation

$$\begin{array}{l} E \xRightarrow{lm} E + T \\ \xRightarrow{lm} T + T \\ \xRightarrow{lm} F + T \\ \xRightarrow{lm} \text{id} + T \end{array}$$



Rightmost Derivation



Leftmost and Rightmost Derivations

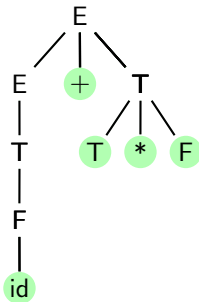
Grammar

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

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

Leftmost Derivation

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



Rightmost Derivation



Leftmost and Rightmost Derivations

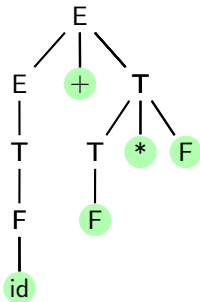
Grammar

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

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

Leftmost Derivation

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



Rightmost Derivation



Leftmost and Rightmost Derivations

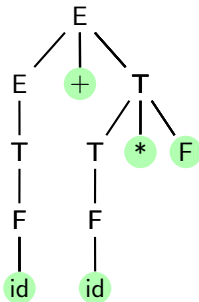
Grammar

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

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

Leftmost Derivation

$$\begin{aligned} E &\xRightarrow{lm} E + T \\ &\xRightarrow{lm} T + T \\ &\xRightarrow{lm} F + T \\ &\xRightarrow{lm} \text{id} + T \\ &\xRightarrow{lm} \text{id} + T * F \\ &\xRightarrow{lm} \text{id} + \textcolor{blue}{F} * F \\ &\xRightarrow{lm} \text{id} + \text{id} * F \end{aligned}$$



Rightmost Derivation



Leftmost and Rightmost Derivations

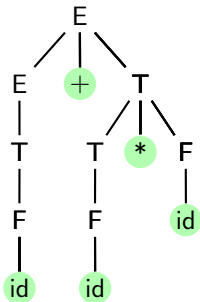
Grammar

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

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

Leftmost Derivation

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



Rightmost Derivation



Leftmost and Rightmost Derivations

Grammar

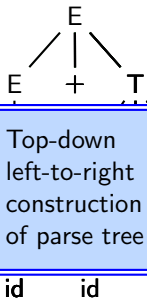
$$\begin{aligned} E &\rightarrow E + T \mid T \\ T &\rightarrow T * F \mid F \\ F &\rightarrow \text{id} \end{aligned}$$

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

Leftmost Derivation

Rightmost Derivation

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





Leftmost and Rightmost Derivations

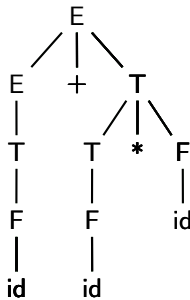
Grammar

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

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

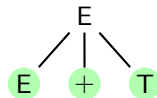
Leftmost Derivation

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



Rightmost Derivation

$$E \xRightarrow{rm} E + T$$





Leftmost and Rightmost Derivations

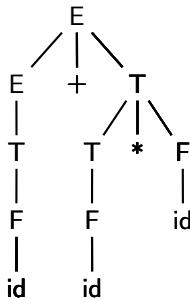
Grammar

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

Sentence: id + id * id

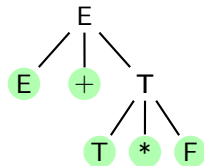
Leftmost Derivation

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



Rightmost Derivation

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





Leftmost and Rightmost Derivations

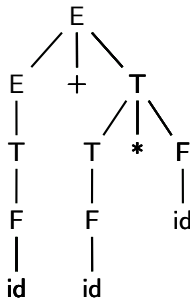
Grammar

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

Sentence: id + id * id

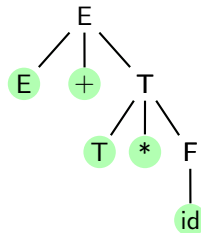
Leftmost Derivation

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



Rightmost Derivation

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





Leftmost and Rightmost Derivations

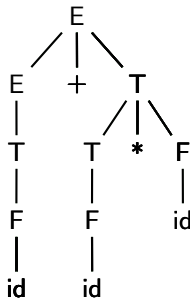
Grammar

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

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

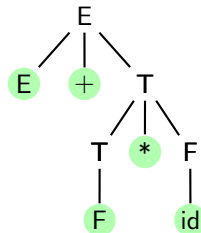
Leftmost Derivation

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



Rightmost Derivation

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





Leftmost and Rightmost Derivations

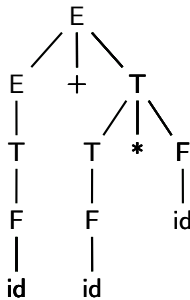
Grammar

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

Sentence: id + id * id

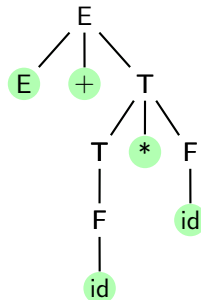
Leftmost Derivation

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



Rightmost Derivation

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





Leftmost and Rightmost Derivations

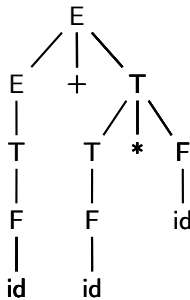
Grammar

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

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

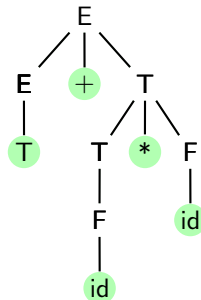
Leftmost Derivation

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



Rightmost Derivation

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





Leftmost and Rightmost Derivations

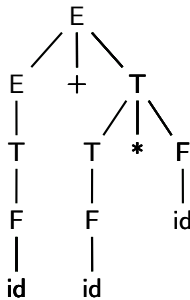
Grammar

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

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

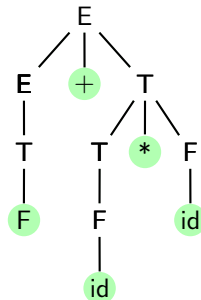
Leftmost Derivation

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



Rightmost Derivation

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





Leftmost and Rightmost Derivations

Grammar

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

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

Leftmost Derivation

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

Rightmost Derivation

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



Leftmost and Rightmost Derivations

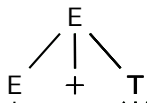
Grammar

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

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

Leftmost Derivation

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

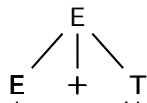


Top-down
left-to-right
construction
of parse tree

id id

Rightmost Derivation

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



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

id id

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



Derivations and Sentences

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

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

- 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
 - $E \xRightarrow{lm} E + E \xRightarrow{lm} \text{id} + E \xRightarrow{lm} \text{id} + E * E \xRightarrow{lm} \text{id} + \text{id} * E \xRightarrow{lm} \text{id} + \text{id} * \text{id}$
This derivation represents the grouping $\text{id} + (\text{id} * \text{id})$
 - $E \xRightarrow{lm} E * E \xRightarrow{lm} E + E * E \xRightarrow{lm} \text{id} + E * E \xRightarrow{lm} \text{id} + \text{id} * E \xRightarrow{lm} \text{id} + \text{id} * \text{id}$
This derivation represents the grouping $(\text{id} + \text{id}) * \text{id}$



Ambiguous Grammars

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

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

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

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



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

Ambiguity in Expressions Grammar

Grammar

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

Input

id + id * id



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

Ambiguity in Expressions Grammar

Grammar

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

Input

id + id * id

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



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

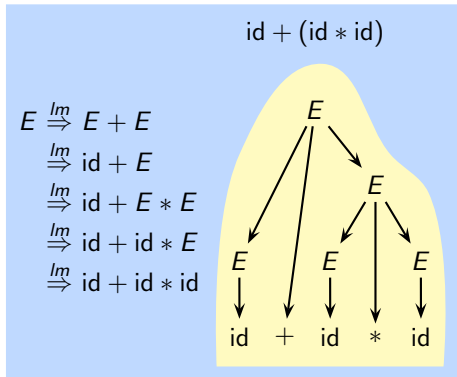
Ambiguity in Expressions Grammar

Grammar

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

Input

id + id * id





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

Ambiguity in Expressions Grammar

Grammar

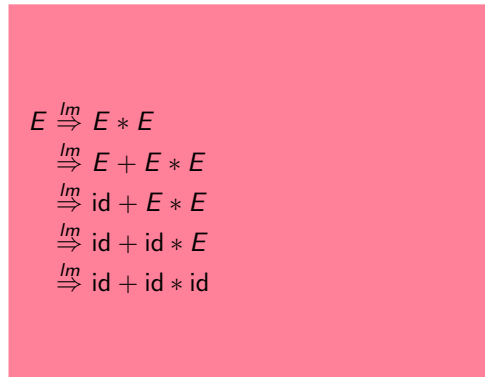
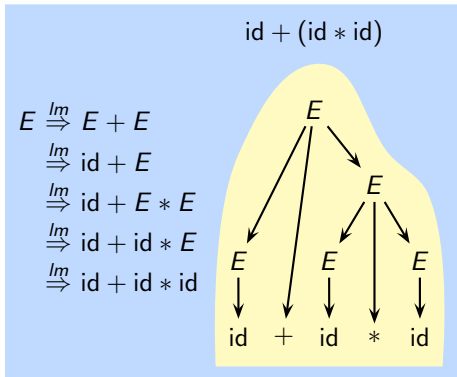
$$E \rightarrow E + E$$

$$E \rightarrow E * E$$

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

Input

id + id * id





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

Ambiguity in Expressions Grammar

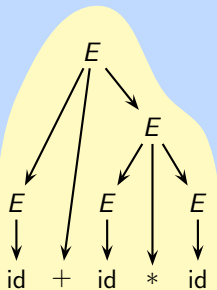
Grammar

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

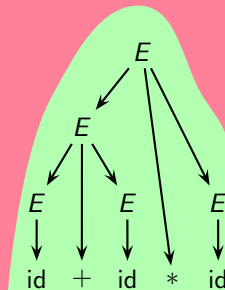
Input

id + id * id

id + (id * id)

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


(id + id) * id

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




Disambiguating Expressions Grammar

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

- 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

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

- 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

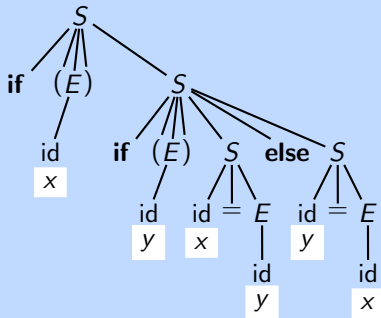


Ambiguity in IF-ELSE Grammar

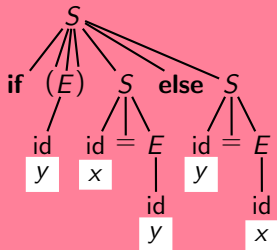
$S \rightarrow \text{if } (E) S \text{ else } S$
 $S \rightarrow \text{if } (E) S$
 $S \rightarrow \text{id} = E$

Consider Sentence

if (x) if (y) x = y else y = x



if (x) if (y) x = y else y = x



if (x) if (y) x = y else y = x

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



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

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

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

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

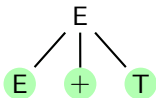
$E \rightarrow E + T \mid T$
$T \rightarrow T * F \mid F$
$F \rightarrow \text{id}$

Sentence: id + id * id

Rightmost Derivation

Rightmost Derivation in Reverse

$E \xRightarrow{rm} E + T$



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

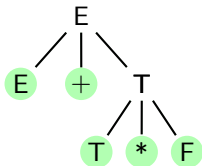
$$\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}$$

Sentence: id + id * id

Rightmost Derivation

Rightmost Derivation in Reverse

$E \xRightarrow{rm} E + T$
 $\xRightarrow{rm} E + T * F$



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

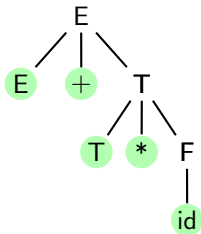
$E \rightarrow E + T \mid T$
$T \rightarrow T * F \mid F$
$F \rightarrow \text{id}$

Sentence: id + id * id

Rightmost Derivation

Rightmost Derivation in Reverse

$E \xRightarrow{rm} E + T$
 $\xRightarrow{rm} E + T * F$
 $\xRightarrow{rm} E + T * \text{id}$



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

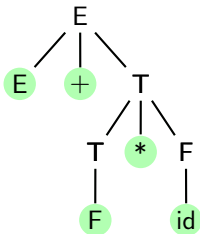
$E \rightarrow E + T \mid T$
$T \rightarrow T * F \mid F$
$F \rightarrow \text{id}$

Sentence: id + id * id

Rightmost Derivation

Rightmost Derivation in Reverse

$E \xRightarrow{rm} E + T$
 $\xRightarrow{rm} E + T * F$
 $\xRightarrow{rm} E + T * \text{id}$
 $\xRightarrow{rm} E + F * \text{id}$



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

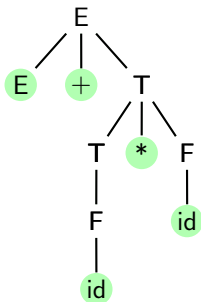
$$\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}$$

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

Rightmost Derivation

Rightmost Derivation in Reverse

$E \xRightarrow{rm} E + T$
 $\xRightarrow{rm} E + T * F$
 $\xRightarrow{rm} E + T * \text{id}$
 $\xRightarrow{rm} E + \textcolor{blue}{F} * \text{id}$
 $\xRightarrow{rm} E + \text{id} * \text{id}$





Rightmost Derivation for Bottom-Up Parsing

Grammar

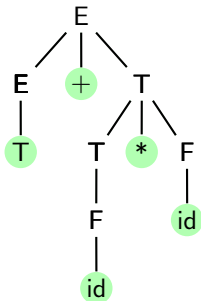
$$\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}$$

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

Rightmost Derivation

Rightmost Derivation in Reverse

$E \xRightarrow{rm} E + T$
 $\xRightarrow{rm} E + T * F$
 $\xRightarrow{rm} E + T * \text{id}$
 $\xRightarrow{rm} E + F * \text{id}$
 $\xRightarrow{rm} \textcolor{blue}{E} + \text{id} * \text{id}$
 $\xRightarrow{rm} T + \text{id} * \text{id}$





Rightmost Derivation for Bottom-Up Parsing

Grammar

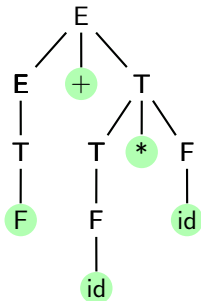
$$\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}$$

Sentence: id + id * id

Rightmost Derivation

Rightmost Derivation in Reverse

$E \xRightarrow{rm} E + T$
 $\xRightarrow{rm} E + T * F$
 $\xRightarrow{rm} E + T * \text{id}$
 $\xRightarrow{rm} E + F * \text{id}$
 $\xRightarrow{rm} E + \text{id} * \text{id}$
 $\xRightarrow{rm} \textcolor{blue}{T} + \text{id} * \text{id}$
 $\xRightarrow{rm} F + \text{id} * \text{id}$





Rightmost Derivation for Bottom-Up Parsing

Grammar

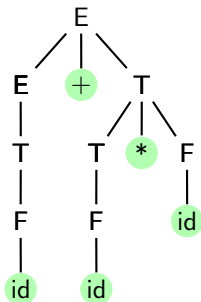
$$\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}$$

Sentence: id + id * id

Rightmost Derivation

Rightmost Derivation in Reverse

$E \xRightarrow{rm} E + T$
 $\xRightarrow{rm} E + T * F$
 $\xRightarrow{rm} E + T * \text{id}$
 $\xRightarrow{rm} E + F * \text{id}$
 $\xRightarrow{rm} E + \text{id} * \text{id}$
 $\xRightarrow{rm} T + \text{id} * \text{id}$
 $\xRightarrow{rm} \textcolor{blue}{F} + \text{id} * \text{id}$
 $\xRightarrow{rm} \text{id} + \text{id} * \text{id}$





Rightmost Derivation for Bottom-Up Parsing

Grammar

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

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

Rightmost Derivation

Rightmost Derivation in Reverse

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

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



Rightmost Derivation for Bottom-Up Parsing

Grammar

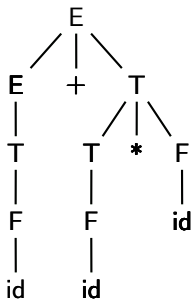
$E \rightarrow E + T \mid T$
$T \rightarrow T * F \mid F$
$F \rightarrow \text{id}$

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

Rightmost Derivation

Rightmost Derivation in Reverse

$E \xRightarrow{rm} E + T$
 $\xRightarrow{rm} E + T * F$
 $\xRightarrow{rm} E + T * \text{id}$
 $\xRightarrow{rm} E + F * \text{id}$
 $\xRightarrow{rm} E + \text{id} * \text{id}$
 $\xRightarrow{rm} T + \text{id} * \text{id}$
 $\xRightarrow{rm} F + \text{id} * \text{id}$
 $\xRightarrow{rm} \text{id} + \text{id} * \text{id}$





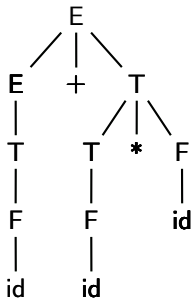
Rightmost Derivation for Bottom-Up Parsing

Grammar

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

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

Rightmost Derivation

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


Rightmost Derivation in Reverse

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

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



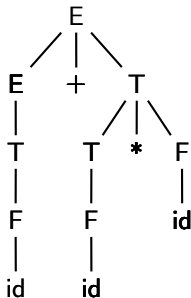
Rightmost Derivation for Bottom-Up Parsing

Grammar

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

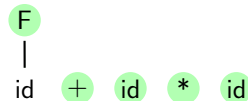
Sentence: id + id * id

Rightmost Derivation

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


Rightmost Derivation in Reverse

id + id * id
 \xRightarrow{rm} **F** + id * id





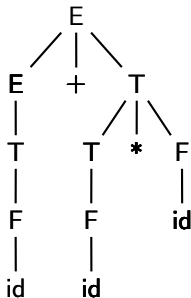
Rightmost Derivation for Bottom-Up Parsing

Grammar

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

Sentence: id + id * id

Rightmost Derivation

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


Rightmost Derivation in Reverse

$$\begin{array}{l} \text{id} + \text{id} * \text{id} \\ \xRightarrow{rm} F + \text{id} * \text{id} \\ \xRightarrow{rm} T + \text{id} * \text{id} \end{array}$$



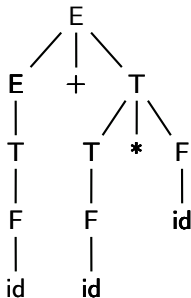

Rightmost Derivation for Bottom-Up Parsing

Grammar

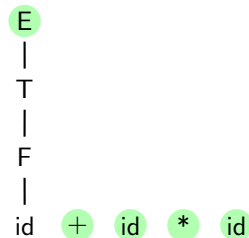
$$\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}$$

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

Rightmost Derivation

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


Rightmost Derivation in Reverse

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




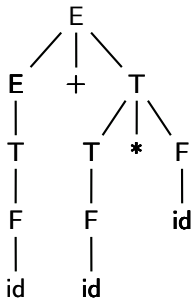
Rightmost Derivation for Bottom-Up Parsing

Grammar

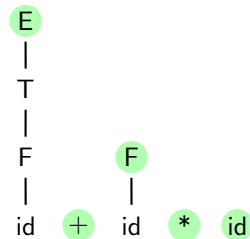
$$\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}$$

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

Rightmost Derivation

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


Rightmost Derivation in Reverse

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




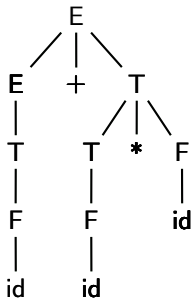
Rightmost Derivation for Bottom-Up Parsing

Grammar

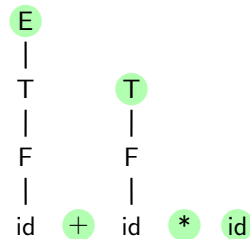
$$\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}$$

Sentence: id + id * id

Rightmost Derivation

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


Rightmost Derivation in Reverse

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




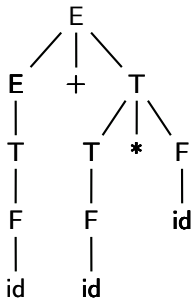
Rightmost Derivation for Bottom-Up Parsing

Grammar

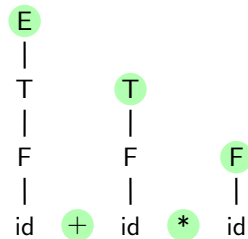
$$\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow \text{id} \end{array}$$

Sentence: id + id * id

Rightmost Derivation

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


Rightmost Derivation in Reverse

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




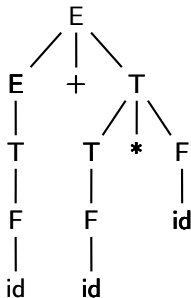
Rightmost Derivation for Bottom-Up Parsing

Grammar

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

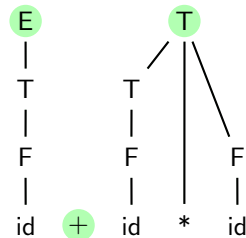
Sentence: id + id * id

Rightmost Derivation

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


Rightmost Derivation in Reverse

id + id * id

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




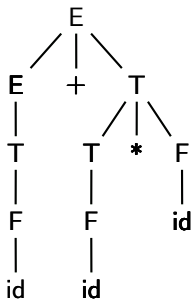
Rightmost Derivation for Bottom-Up Parsing

Grammar

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

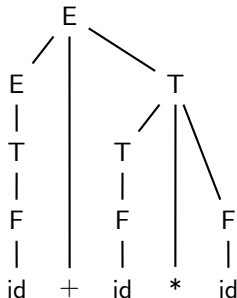
Sentence: $\text{id} + \text{id} * \text{id}$

Rightmost Derivation

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


Rightmost Derivation in Reverse

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

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




Rightmost Derivation for Bottom-Up Parsing

Grammar

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

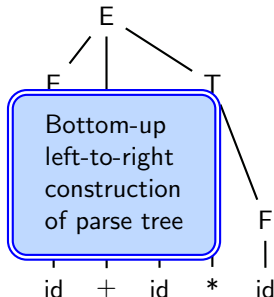
Sentence: id + id * id

Rightmost Derivation

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

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

Rightmost Derivation in Reverse

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




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



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

Shift Reduce Parsing



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

An Overview of Shift Reduce Parsing

Grammar

$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow \text{id}$

Input

id + id * id



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

An Overview of Shift Reduce Parsing

Grammar

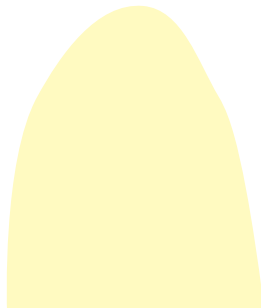
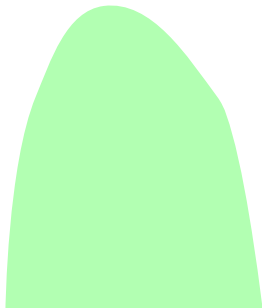
$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow \text{id}$

Input

id + id * id





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

An Overview of Shift Reduce Parsing

Grammar

$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow \text{id}$

Input

id + id * id

id

id



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

An Overview of Shift Reduce Parsing

Grammar

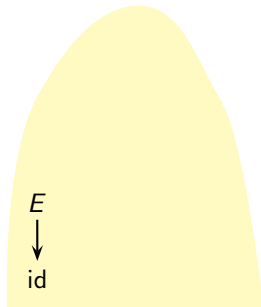
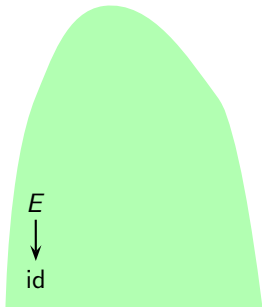
$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow \text{id}$

Input

id + id * id





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

An Overview of Shift Reduce Parsing

Grammar

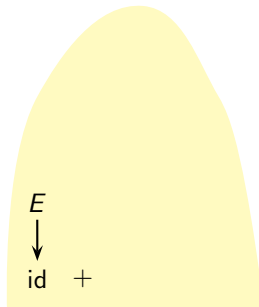
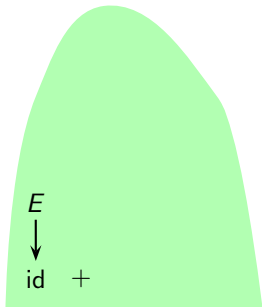
$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow \text{id}$

Input

id + id * id





An Overview of Shift Reduce Parsing

Grammar

$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow \text{id}$

Input

id + id * id

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

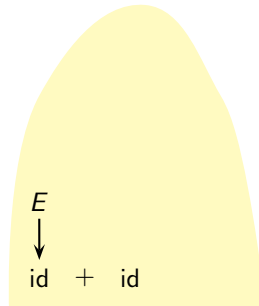
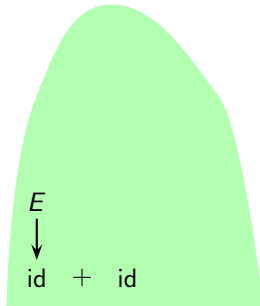
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





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

An Overview of Shift Reduce Parsing

Grammar

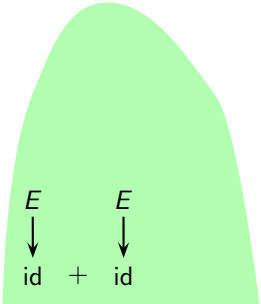
$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow \text{id}$

Input

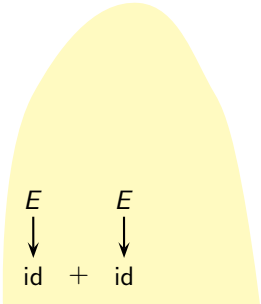
id + id * id



E
↓
id

+

E
↓
id



E
↓
id

+

E
↓
id



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

An Overview of Shift Reduce Parsing

Grammar

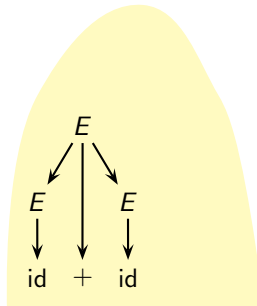
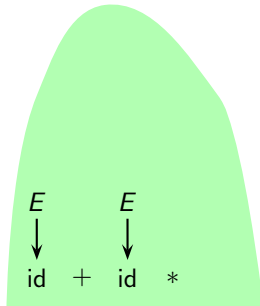
$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow id$

Input

id + id * id





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

An Overview of Shift Reduce Parsing

Grammar

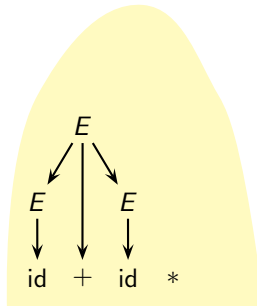
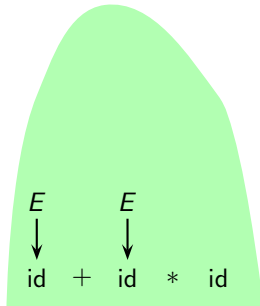
$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow id$

Input

id + id * id





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

An Overview of Shift Reduce Parsing

Grammar

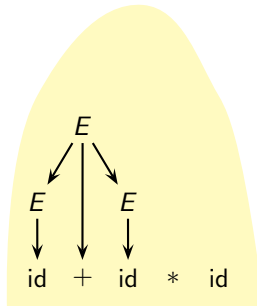
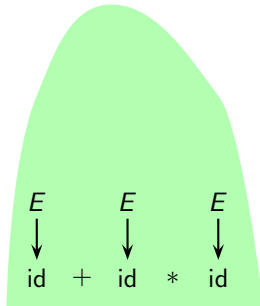
$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow id$

Input

id + id * id





An Overview of Shift Reduce Parsing

Grammar

$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow \text{id}$

Input

id + id * id

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

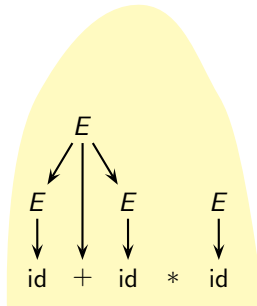
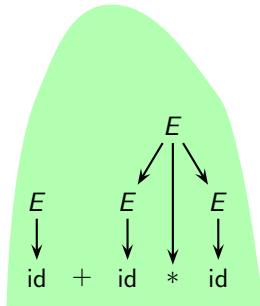
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





An Overview of Shift Reduce Parsing

Grammar

$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow id$

Input

id + id * id

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

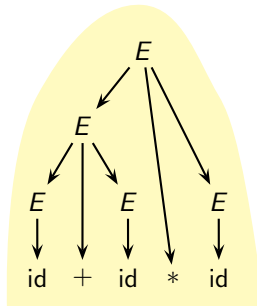
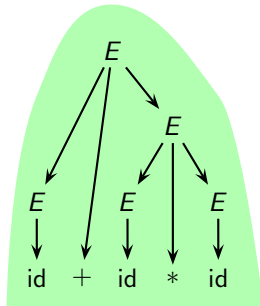
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





An Overview of Shift Reduce Parsing

Grammar

$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow id$

Input

id + id * id

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

Shift Reduce Parsing

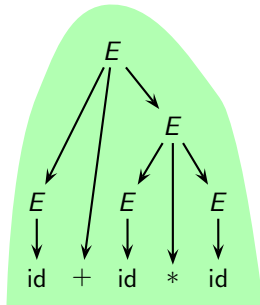
SLR(1) Parsing

Conceptual Issues in
Parsing

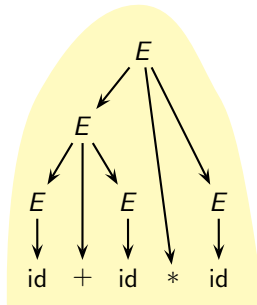
CLR(1) Parsing

LALR(1) Parsing

id + (id * id)



(id + id) * id





An Overview of Shift Reduce Parsing

Grammar

$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow \text{id}$

Input

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

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

Shift Reduce Parsing

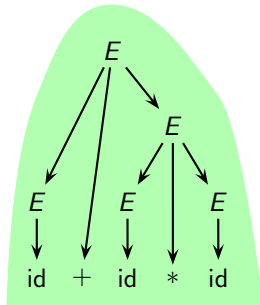
SLR(1) Parsing

Conceptual Issues in
Parsing

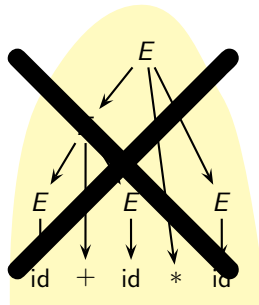
CLR(1) Parsing

LALR(1) Parsing

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



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





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

An Overview of Shift Reduce Parsing

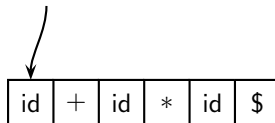
Grammar

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

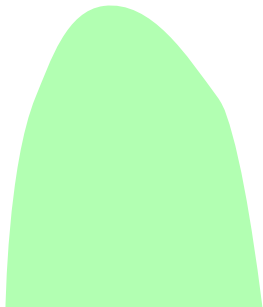
Input

id + id * id

Next token



id + (id * id)



Next Action: Shift



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

An Overview of Shift Reduce Parsing

Grammar

$E \rightarrow E + E$

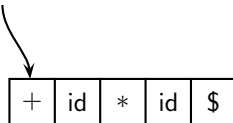
$E \rightarrow E * E$

$E \rightarrow id$

Input

id + id * id

Next token



id + (id * id)

Next Action: Reduce by $E \rightarrow id$

id

Parsing Stack

id



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

An Overview of Shift Reduce Parsing

Grammar

$E \rightarrow E + E$

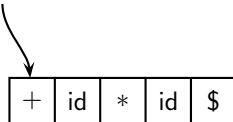
$E \rightarrow E * E$

$E \rightarrow id$

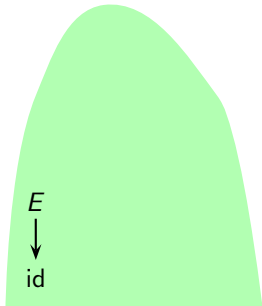
Input

id + id * id

Next token



id + (id * id)



Next Action: Shift

E

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

An Overview of Shift Reduce Parsing

Grammar

$E \rightarrow E + E$

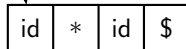
$E \rightarrow E * E$

$E \rightarrow \text{id}$

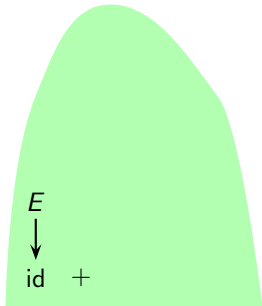
Input

id + id * id

Next token



id + (id * id)



Next Action: Shift



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

An Overview of Shift Reduce Parsing

Grammar

$E \rightarrow E + E$

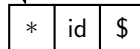
$E \rightarrow E * E$

$E \rightarrow id$

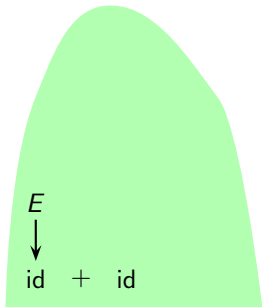
Input

id + id * id

Next token



id + (id * id)



Parsing Stack

Next Action: Reduce by $E \rightarrow id$



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

An Overview of Shift Reduce Parsing

Grammar

$E \rightarrow E + E$

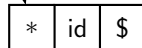
$E \rightarrow E * E$

$E \rightarrow \text{id}$

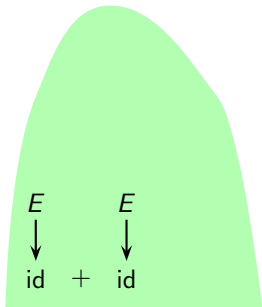
Input

id + id * id

Next token



id + (id * id)



Next Action: Shift



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

An Overview of Shift Reduce Parsing

Grammar

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

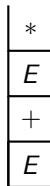
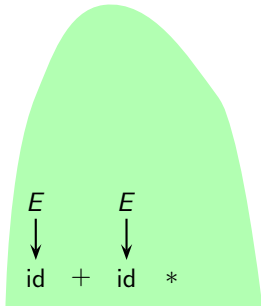
Input

id + id * id

Next token



id + (id * id)



Next Action: Shift

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

An Overview of Shift Reduce Parsing

Grammar

$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow id$

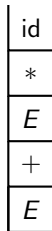
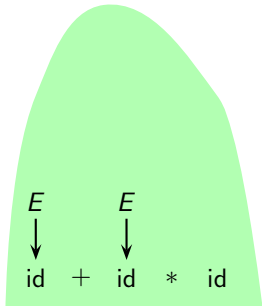
Input

id + id * id

Next token



id + (id * id)



Parsing Stack

Next Action: Reduce by $E \rightarrow id$



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

An Overview of Shift Reduce Parsing

Grammar

$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow \text{id}$

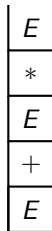
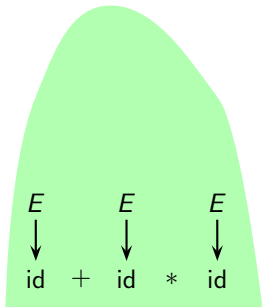
Input

id + id * id

Next token



id + (id * id)



Parsing Stack

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



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

An Overview of Shift Reduce Parsing

Grammar

$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow \text{id}$

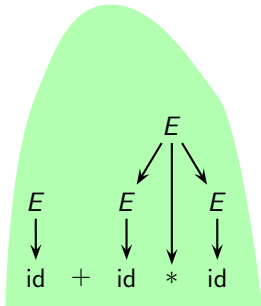
Input

id + id * id

Next token



id + (id * id)



Parsing Stack

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



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

An Overview of Shift Reduce Parsing

Grammar

$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow id$

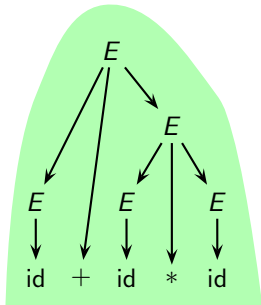
Input

id + id * id

Next token



id + (id * id)



Next Action: Accept



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

An Overview of Shift Reduce Parsing

Grammar

1. $E \rightarrow E + E$

2. $E \rightarrow E * E$

3. $E \rightarrow \text{id}$

Input

id + id * id



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

An Overview of Shift Reduce Parsing

Grammar

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

Input

id + id * id

Step	Stack \rightarrow	Input	Action
1	\$	id + id * id\$	shift



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

An Overview of Shift Reduce Parsing

Grammar

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

Input

id + id * id

Step	Stack \rightarrow	Input	Action
1	\$	id + id * id\$	shift
2	\$id	+ id * id\$	reduce by 3

id



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

An Overview of Shift Reduce Parsing

Grammar

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

Input

id + id * id

Step	Stack \rightarrow	Input	Action
1	\$	id + id * id\$	shift
2	\$id	+ id * id\$	reduce by 3
3	\$E	+ id * id\$	shift

E
 \downarrow
id



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

An Overview of Shift Reduce Parsing

Grammar

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

Input

id + id * id

E
↓
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



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

An Overview of Shift Reduce Parsing

Grammar

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

Input

id + id * id

E
↓
id + 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



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

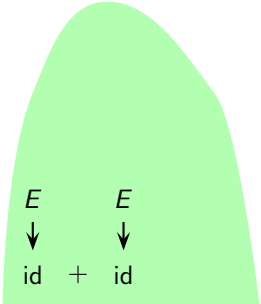
An Overview of Shift Reduce Parsing

Grammar

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

Input

id + id * id



E E
↓ ↓
id + 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



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

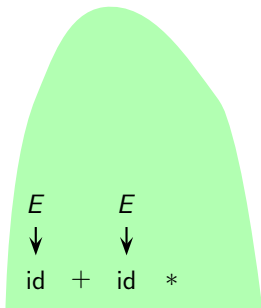
An Overview of Shift Reduce Parsing

Grammar

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

Input

id + id * 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



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

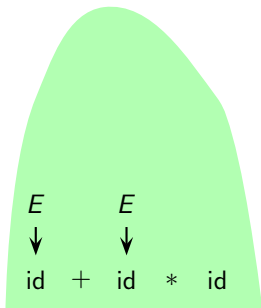
An Overview of Shift Reduce Parsing

Grammar

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

Input

id + id * 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



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

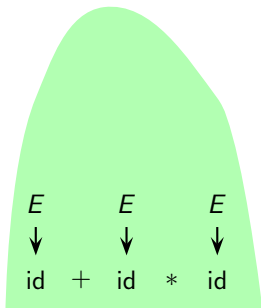
An Overview of Shift Reduce Parsing

Grammar

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

Input

id + id * 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



An Overview of Shift Reduce Parsing

Grammar

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

Input

id + id * id

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

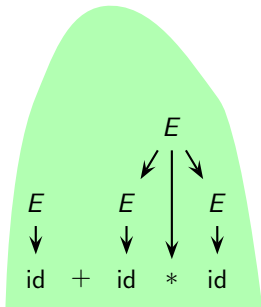
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing



Step	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



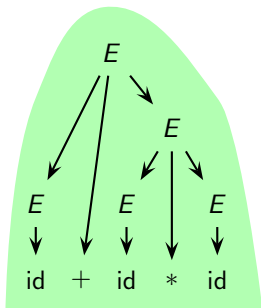
An Overview of Shift Reduce Parsing

Grammar

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

Input

id + id * 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



An Overview of Shift Reduce Parsing

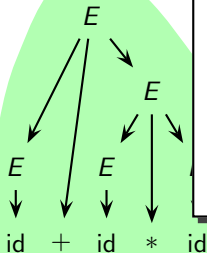
Grammar

Input

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

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 , $rsf_{i+1} \xRightarrow{rm} rsf_i$
 - 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

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

- In every step i , concatenation of the stack and the remaining input gives a right sentential form (rsf_i)
- For every step i , $\text{rsf}_{i+1} \xRightarrow{rm} \text{rsf}_i$
- How do we go from rsf_i to rsf_{i+1} ?
 - $S \xRightarrow{*rm} \alpha A w \xRightarrow{rm} \alpha \beta w$
 - 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

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

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



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

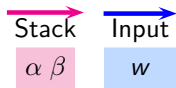
LALR(1) Parsing

Why do Handles Form the Basis of Bottom Up Parsing?

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

Why?

- $S \xRightarrow{*rm} \alpha A w \xRightarrow{rm} \alpha \beta w$
- 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?



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

Grammar

$$1. E \rightarrow E + E$$

$$2. E \rightarrow E * E$$

$$3. E \rightarrow id$$

Input

id + id * id



Why do Handles Form the Basis of Bottom Up Parsing?

Grammar

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

Input

id + id * id

Step	Stack \rightarrow	Input	Action
1	\$	id + id * id\$	shift

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. $E \rightarrow id$

Input

id + id * id

Step	Stack \rightarrow	Input	Action
1	\$	id + id * id\$	shift
2	\$ id	+ id * id\$	reduce by 3

id

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. $E \rightarrow id$

Input

id + id * id

Step	Stack \rightarrow	Input	Action
1	\$	id + id * id\$	shift
2	\$ id	+ id * id\$	reduce by 3
3	\$ E	+ id * id\$	shift

E



id

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. $E \rightarrow id$

Input

id + id * 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

E
↓
id +

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. $E \rightarrow id$

Input

id + id * 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

E



id + id

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. $E \rightarrow id$

Input

id + id * 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

E E
 \downarrow \downarrow
 id + id

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. $E \rightarrow id$

Input

id + id * id

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

E E
 \downarrow \downarrow
 id + 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



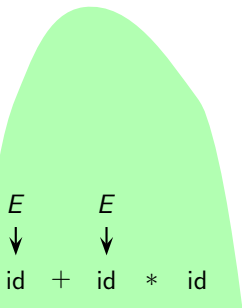
Why do Handles Form the Basis of Bottom Up Parsing?

Grammar

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

Input

id + id * 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



Why do Handles Form the Basis of Bottom Up Parsing?

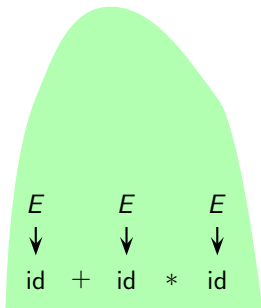
Grammar

Input

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

id + id * 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



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?

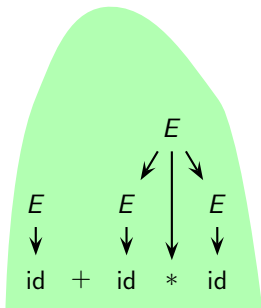
Grammar

Input

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

id + id * 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



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?

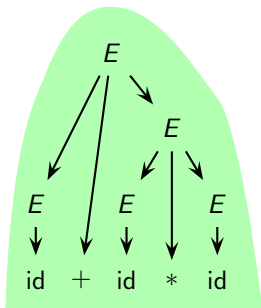
Grammar

Input

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

id + id * 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



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



Shift Reduce Parsing

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

[parsing-slides-sanyal-part2.pdf](#)



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



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



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

$$1 \ E \rightarrow E + E$$

$$2 \ E \rightarrow E * E$$

$$3 \ E \rightarrow \text{id}$$



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

$$1 \ E \rightarrow E + E$$

$$2 \ E \rightarrow E * E$$

$$3 \ E \rightarrow \text{id}$$

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	



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 $E \rightarrow id$

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 $E \rightarrow id$

Next token

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 2

0

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		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 $E \rightarrow id$

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 $E \rightarrow id$

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: Cover by 1

E
0

Parsing Stack



SLR(1) Parsing Example

1 $E \rightarrow E + E$

2 $E \rightarrow E * E$

3 $E \rightarrow id$

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: Shift 3

1
E
0

Parsing Stack



SLR(1) Parsing Example

1 $E \rightarrow E + E$

2 $E \rightarrow E * E$

3 $E \rightarrow id$

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	

3
+
1
E
0

Next Action: Shift 2

Parsing Stack



SLR(1) Parsing Example

1 $E \rightarrow E + E$

2 $E \rightarrow E * E$

3 $E \rightarrow id$

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	

2
id
3
+
1
E
0

Next Action: Reduce by Rule 3

Parsing Stack



SLR(1) Parsing Example

1 $E \rightarrow E + E$

2 $E \rightarrow E * E$

3 $E \rightarrow id$

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	

2
id
3
+
1
E
0

Next Action: Reduce by Rule 3

Parsing Stack



SLR(1) Parsing Example

1 $E \rightarrow E + E$

2 $E \rightarrow E * E$

3 $E \rightarrow id$

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

Next Action: Cover by 5

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 $E \rightarrow id$

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	

5
E
3
+
1
E
0

Next Action: Shift 4

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 $E \rightarrow id$

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	\$
----	----

Next Action: Shift 2

4
*
5
E
3
+
1
E
0

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 $E \rightarrow id$

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

Next token



Next Action: Reduce by Rule 3

Parsing Stack



SLR(1) Parsing Example

1 $E \rightarrow E + E$

2 $E \rightarrow E * E$

3 $E \rightarrow id$

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

Next token



Next Action: Reduce by Rule 3

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	

E
4
*
5
E
3
+
1
E
0

Parsing Stack

Next token



Next Action: Cover by 6



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

Parsing Stack

Next token



Next Action: Reduce by 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	

6
E
4
*
5
E
3
+
1
E
0

Parsing Stack

Next token



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

Next Action: Cover by 5

Parsing Stack



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

Next Action: Reduce by Rule 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 $E \rightarrow id$

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

Next Action: Reduce by Rule 1

Parsing Stack



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 $E \rightarrow \text{id}$

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: Accept

1
E
0

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

$$1 \ E \rightarrow E + E$$

$$2 \ E \rightarrow E * E$$

$$3 \ E \rightarrow \text{id}$$



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

$$1 \ E \rightarrow E + E$$

$$2 \ E \rightarrow E * E$$

$$3 \ E \rightarrow \text{id}$$

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	



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 $E \rightarrow id$

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 $E \rightarrow id$

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

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	

Step	Stack \rightarrow	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



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

$$1 \ E \rightarrow E + E$$

$$2 \ E \rightarrow E * E$$

$$3 \ E \rightarrow \text{id}$$

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	

Step	Stack \rightarrow	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



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

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

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

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

`id + id + id` , `id * id * id` , `id + id * id` , and `id * id + 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



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

$$1 \ E \rightarrow E + E$$

$$2 \ E \rightarrow E * E$$

$$3 \ E \rightarrow \text{id}$$



Shift Reduce Parsing: From Intuitions to Formal Algorithms

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

$$1 \ E \rightarrow E + E$$

$$2 \ E \rightarrow E * E$$

$$3 \ E \rightarrow id$$

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



Shift Reduce Parsing: From Intuitions to Formal Algorithms

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 $E \rightarrow id$

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		s3/r1	s4/r1	r1	
6		s3/r2	s4/r2	r2	

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	Stack \rightarrow	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

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 $E \rightarrow id$

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

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

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



Shift Reduce Parsing: From Intuitions to Formal Algorithms

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

$$1 \ E \rightarrow E + E$$

$$2 \ E \rightarrow E * E$$

$$3 \ E \rightarrow id$$

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		s3/r1	s4/r1	r1	
6		s3/r2	s4/r2	r2	

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



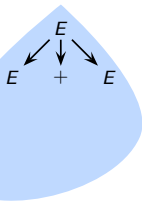
Step	Stack \rightarrow	Input	Action
1	\$0	id + id * id\$	s2
2	\$0 id 2	+ id * id\$	r3 and c1
3	\$0 E 1	+ id * id\$	s3
4	\$0 E 1 + 3	id * id\$	s2
5	\$0 E 1 + 3 id 2	* id\$	r3 and c5
6	\$0 E 1 + 3 E 5	* id\$	s4
7	\$0 E 1 + 3 E 5 * 4	id\$	s2
8	\$0 E 1 + 3 E 5 * 4 id 2	\$	r3 and c6
9	\$0 E 1 + 3 E 5 * 4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



Step 1: Precedence and Associativity Rule Out Undesirable Derivations

Input $id + id + id$

$$E \xRightarrow{rm} E + E$$



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



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

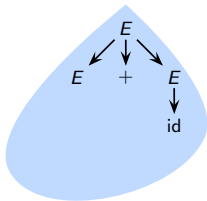
CLR(1) Parsing

LALR(1) Parsing

Step 1: Precedence and Associativity Rule Out Undesirable Derivations

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

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





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

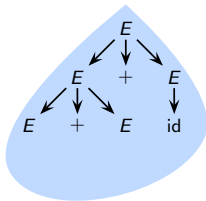
CLR(1) Parsing

LALR(1) Parsing

Step 1: Precedence and Associativity Rule Out Undesirable Derivations

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

$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + \text{id}$
 $\xRightarrow{rm} E + E + \text{id}$

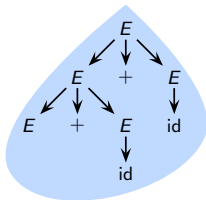




Step 1: Precedence and Associativity Rule Out Undesirable Derivations

Input $id + id + id$

$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + id$
 $\xRightarrow{rm} E + E + id$
 $\xRightarrow{rm} E + id + id$



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

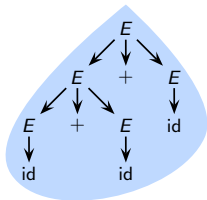
LALR(1) Parsing



Step 1: Precedence and Associativity Rule Out Undesirable Derivations

Input $id + id + id$

$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + id$
 $\xRightarrow{rm} E + E + id$
 $\xRightarrow{rm} E + id + id$
 $\xRightarrow{rm} id + id + id$



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

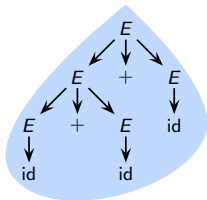
LALR(1) Parsing



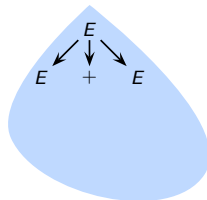
Step 1: Precedence and Associativity Rule Out Undesirable Derivations

Input $id + id + id$

$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + id$
 $\xRightarrow{rm} E + E + id$
 $\xRightarrow{rm} E + id + id$
 $\xRightarrow{rm} id + id + id$



$E \xRightarrow{rm} E + E$



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

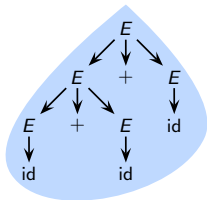
LALR(1) Parsing



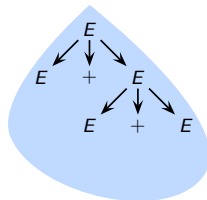
Step 1: Precedence and Associativity Rule Out Undesirable Derivations

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

$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + \text{id}$
 $\xRightarrow{rm} E + E + \text{id}$
 $\xRightarrow{rm} E + \text{id} + \text{id}$
 $\xRightarrow{rm} \text{id} + \text{id} + \text{id}$



$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + E + E$



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

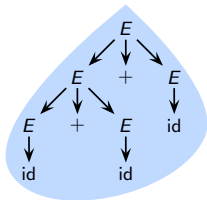
LALR(1) Parsing



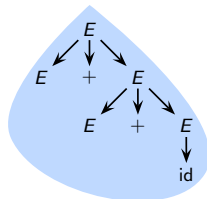
Step 1: Precedence and Associativity Rule Out Undesirable Derivations

Input $id + id + id$

$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + id$
 $\xRightarrow{rm} E + E + id$
 $\xRightarrow{rm} E + id + id$
 $\xRightarrow{rm} id + id + id$



$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + E + E$
 $\xRightarrow{rm} E + E + id$



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

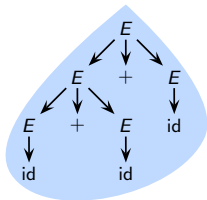
LALR(1) Parsing



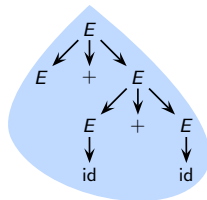
Step 1: Precedence and Associativity Rule Out Undesirable Derivations

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

$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + \text{id}$
 $\xRightarrow{rm} E + E + \text{id}$
 $\xRightarrow{rm} E + \text{id} + \text{id}$
 $\xRightarrow{rm} \text{id} + \text{id} + \text{id}$



$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + E + E$
 $\xRightarrow{rm} E + E + \text{id}$
 $\xRightarrow{rm} E + \text{id} + \text{id}$



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

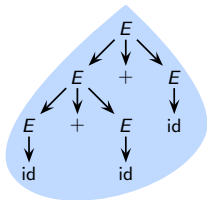
LALR(1) Parsing



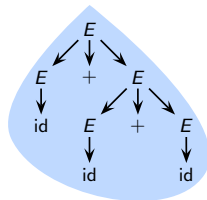
Step 1: Precedence and Associativity Rule Out Undesirable Derivations

Input $id + id + id$

$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + id$
 $\xRightarrow{rm} E + E + id$
 $\xRightarrow{rm} E + id + id$
 $\xRightarrow{rm} id + id + id$



$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + E + E$
 $\xRightarrow{rm} E + E + id$
 $\xRightarrow{rm} E + id + id$
 $\xRightarrow{rm} id + id + id$



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

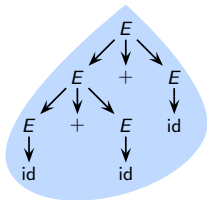
LALR(1) Parsing



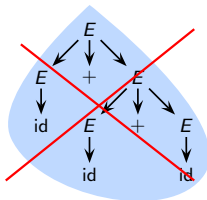
Step 1: Precedence and Associativity Rule Out Undesirable Derivations

Input id + id + id

$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + id$
 $\xRightarrow{rm} E + E + id$
 $\xRightarrow{rm} E + id + id$
 $\xRightarrow{rm} id + id + id$



~~$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + E + E$
 $\xRightarrow{rm} E + E + id$
 $\xRightarrow{rm} E + id + id$
 $\xRightarrow{rm} id + id + id$~~



+ is left associative

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

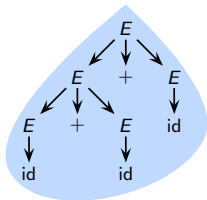
LALR(1) Parsing



Step 1: Precedence and Associativity Rule Out Undesirable Derivations

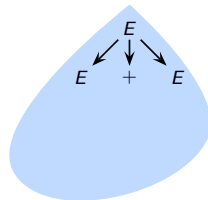
Input $id + id + id$

$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + id$
 $\xRightarrow{rm} E + E + id$
 $\xRightarrow{rm} E + id + id$
 $\xRightarrow{rm} id + id + id$

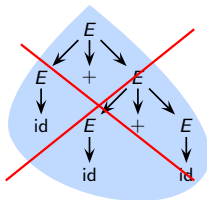


Input $id * id + id$

$E \xRightarrow{rm} E + E$



~~$E \xRightarrow{rm} E + E$~~
 ~~$\xRightarrow{rm} E + E + E$~~
 ~~$\xRightarrow{rm} E + E + id$~~
 ~~$\xRightarrow{rm} E + id + id$~~
 ~~$\xRightarrow{rm} id + id + id$~~



$+$ is left associative

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

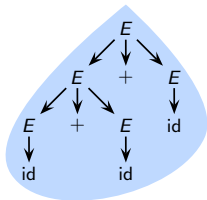
LALR(1) Parsing



Step 1: Precedence and Associativity Rule Out Undesirable Derivations

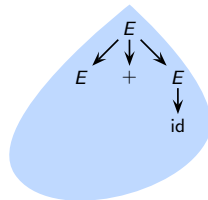
Input $id + id + id$

$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + id$
 $\xRightarrow{rm} E + E + id$
 $\xRightarrow{rm} E + id + id$
 $\xRightarrow{rm} id + id + id$

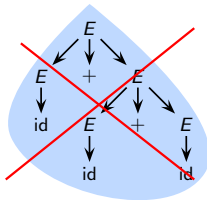


Input $id * id + id$

$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + id$



~~$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + E + E$
 $\xRightarrow{rm} E + E + id$
 $\xRightarrow{rm} E + id + id$
 $\xRightarrow{rm} id + id + id$~~



$+$ is left associative

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

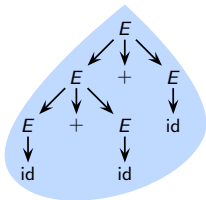
LALR(1) Parsing



Step 1: Precedence and Associativity Rule Out Undesirable Derivations

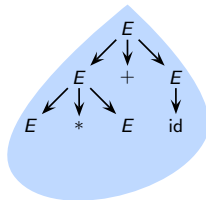
Input $\text{id} + \text{id} + \text{id}$

$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + \text{id}$
 $\xRightarrow{rm} E + E + \text{id}$
 $\xRightarrow{rm} E + \text{id} + \text{id}$
 $\xRightarrow{rm} \text{id} + \text{id} + \text{id}$

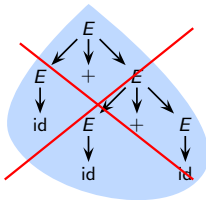


Input $\text{id} * \text{id} + \text{id}$

$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + \text{id}$
 $\xRightarrow{rm} E * E + \text{id}$



~~$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + E + E$
 $\xRightarrow{rm} E + E + \text{id}$
 $\xRightarrow{rm} E + \text{id} + \text{id}$
 $\xRightarrow{rm} \text{id} + \text{id} + \text{id}$~~



$+$ is left associative

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

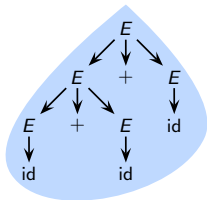
LALR(1) Parsing



Step 1: Precedence and Associativity Rule Out Undesirable Derivations

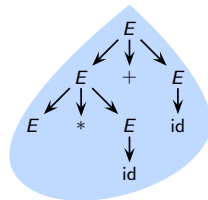
Input id + id + id

$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + id$
 $\xRightarrow{rm} E + E + id$
 $\xRightarrow{rm} E + id + id$
 $\xRightarrow{rm} id + id + id$

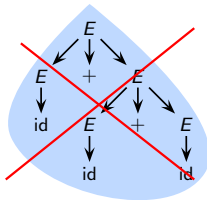


Input id * id + id

$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + id$
 $\xRightarrow{rm} E * E + id$
 $\xRightarrow{rm} E * id + id$



~~$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + E + E$
 $\xRightarrow{rm} E + E + id$
 $\xRightarrow{rm} E + id + id$
 $\xRightarrow{rm} id + id + id$~~



+ is left associative

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

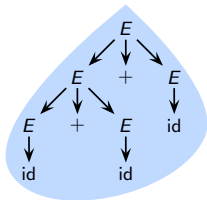
LALR(1) Parsing



Step 1: Precedence and Associativity Rule Out Undesirable Derivations

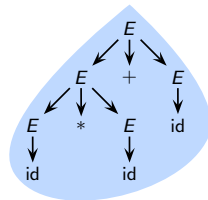
Input $id + id + id$

$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + id$
 $\xRightarrow{rm} E + E + id$
 $\xRightarrow{rm} E + id + id$
 $\xRightarrow{rm} id + id + id$

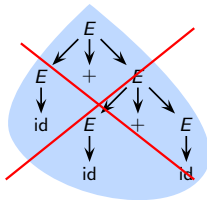


Input $id * id + id$

$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + id$
 $\xRightarrow{rm} E * E + id$
 $\xRightarrow{rm} E * id + id$
 $\xRightarrow{rm} id * id + id$



~~$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + E + E$
 $\xRightarrow{rm} E + E + id$
 $\xRightarrow{rm} E + id + id$
 $\xRightarrow{rm} id + id + id$~~



$+$ is left associative

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

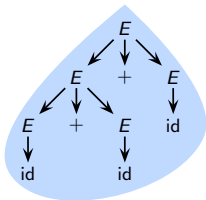
LALR(1) Parsing



Step 1: Precedence and Associativity Rule Out Undesirable Derivations

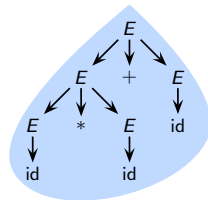
Input $id + id + id$

$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + id$
 $\xRightarrow{rm} E + E + id$
 $\xRightarrow{rm} E + id + id$
 $\xRightarrow{rm} id + id + id$

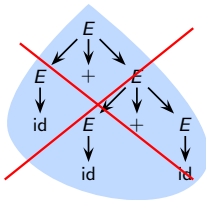


Input $id * id + id$

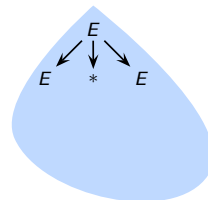
$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + id$
 $\xRightarrow{rm} E * E + id$
 $\xRightarrow{rm} E * id + id$
 $\xRightarrow{rm} id * id + id$



~~$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + E + E$
 $\xRightarrow{rm} E + E + id$
 $\xRightarrow{rm} E + id + id$
 $\xRightarrow{rm} id + id + id$~~



$E \xRightarrow{rm} E * E$



$+$ is left associative

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

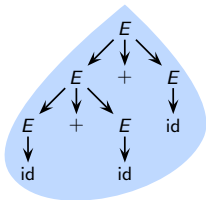
LALR(1) Parsing



Step 1: Precedence and Associativity Rule Out Undesirable Derivations

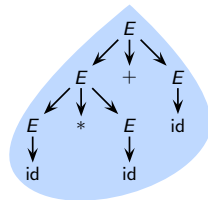
Input $id + id + id$

$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + id$
 $\xRightarrow{rm} E + E + id$
 $\xRightarrow{rm} E + id + id$
 $\xRightarrow{rm} id + id + id$

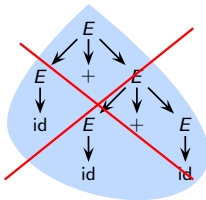


Input $id * id + id$

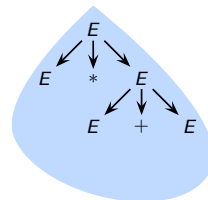
$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + id$
 $\xRightarrow{rm} E * E + id$
 $\xRightarrow{rm} E * id + id$
 $\xRightarrow{rm} id * id + id$



~~$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + E + E$
 $\xRightarrow{rm} E + E + id$
 $\xRightarrow{rm} E + id + id$
 $\xRightarrow{rm} id + id + id$~~



$E \xRightarrow{rm} E * E$
 $\xRightarrow{rm} E * E + E$



$+$ is left associative

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

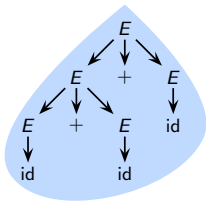
LALR(1) Parsing



Step 1: Precedence and Associativity Rule Out Undesirable Derivations

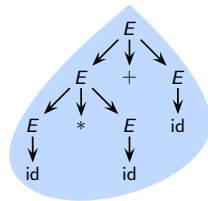
Input $id + id + id$

$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + id$
 $\xRightarrow{rm} E + E + id$
 $\xRightarrow{rm} E + id + id$
 $\xRightarrow{rm} id + id + id$

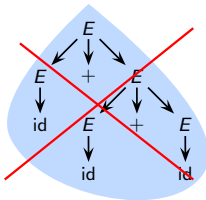


Input $id * id + id$

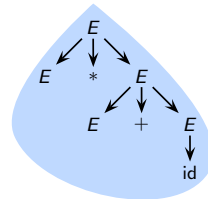
$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + id$
 $\xRightarrow{rm} E * E + id$
 $\xRightarrow{rm} E * id + id$
 $\xRightarrow{rm} id * id + id$



~~$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + E + E$
 $\xRightarrow{rm} E + E + id$
 $\xRightarrow{rm} E + id + id$
 $\xRightarrow{rm} id + id + id$~~



$E \xRightarrow{rm} E * E$
 $\xRightarrow{rm} E * E + E$
 $\xRightarrow{rm} E * E + id$



+ is left associative

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

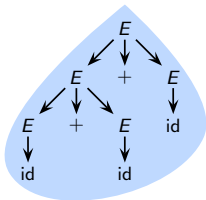
LALR(1) Parsing



Step 1: Precedence and Associativity Rule Out Undesirable Derivations

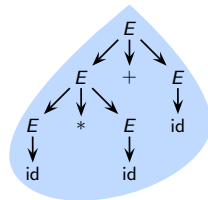
Input $\text{id} + \text{id} + \text{id}$

$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + \text{id}$
 $\xRightarrow{rm} E + E + \text{id}$
 $\xRightarrow{rm} E + \text{id} + \text{id}$
 $\xRightarrow{rm} \text{id} + \text{id} + \text{id}$

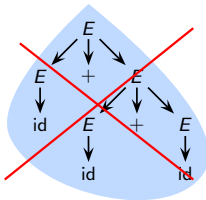


Input $\text{id} * \text{id} + \text{id}$

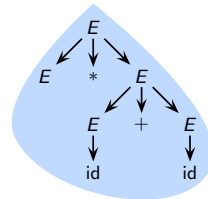
$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + \text{id}$
 $\xRightarrow{rm} E * E + \text{id}$
 $\xRightarrow{rm} E * \text{id} + \text{id}$
 $\xRightarrow{rm} \text{id} * \text{id} + \text{id}$



~~$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + E + E$
 $\xRightarrow{rm} E + E + \text{id}$
 $\xRightarrow{rm} E + \text{id} + \text{id}$
 $\xRightarrow{rm} \text{id} + \text{id} + \text{id}$~~



$E \xRightarrow{rm} E * E$
 $\xRightarrow{rm} E * E + E$
 $\xRightarrow{rm} E * E + \text{id}$
 $\xRightarrow{rm} E * \text{id} + \text{id}$



$+$ is left associative

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

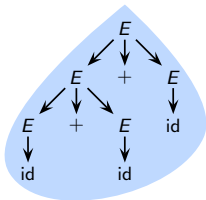
LALR(1) Parsing



Step 1: Precedence and Associativity Rule Out Undesirable Derivations

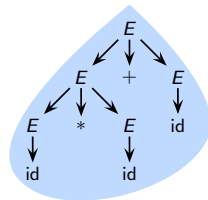
Input $id + id + id$

$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + id$
 $\xRightarrow{rm} E + E + id$
 $\xRightarrow{rm} E + id + id$
 $\xRightarrow{rm} id + id + id$

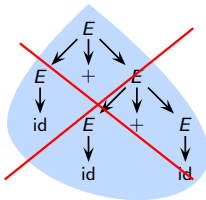


Input $id * id + id$

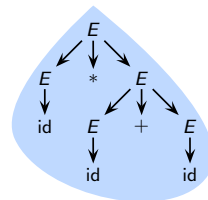
$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + id$
 $\xRightarrow{rm} E * E + id$
 $\xRightarrow{rm} E * id + id$
 $\xRightarrow{rm} id * id + id$



~~$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + E + E$
 $\xRightarrow{rm} E + E + id$
 $\xRightarrow{rm} E + id + id$
 $\xRightarrow{rm} id + id + id$~~



$E \xRightarrow{rm} E * E$
 $\xRightarrow{rm} E * E + E$
 $\xRightarrow{rm} E * E + id$
 $\xRightarrow{rm} E * id + id$
 $\xRightarrow{rm} id * id + id$



$+$ is left associative

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

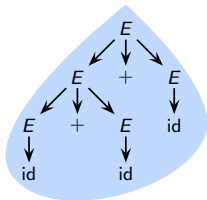
LALR(1) Parsing



Step 1: Precedence and Associativity Rule Out Undesirable Derivations

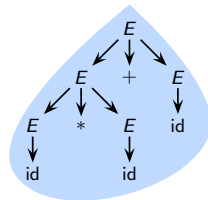
Input $id + id + id$

$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + id$
 $\xRightarrow{rm} E + E + id$
 $\xRightarrow{rm} E + id + id$
 $\xRightarrow{rm} id + id + id$

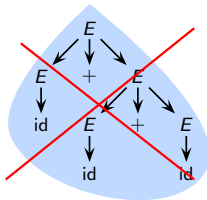


Input $id * id + id$

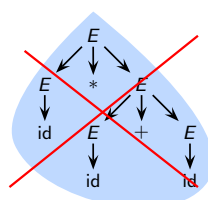
$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + id$
 $\xRightarrow{rm} E * E + id$
 $\xRightarrow{rm} E * id + id$
 $\xRightarrow{rm} id * id + id$



~~$E \xRightarrow{rm} E + E$
 $\xRightarrow{rm} E + E + E$
 $\xRightarrow{rm} E + E + id$
 $\xRightarrow{rm} E + id + id$
 $\xRightarrow{rm} id + id + id$~~



~~$E \xRightarrow{rm} E * E$
 $\xRightarrow{rm} E * E + E$
 $\xRightarrow{rm} E * E + id$
 $\xRightarrow{rm} E * id + id$
 $\xRightarrow{rm} id * id + id$~~



$+$ is left associative

$*$ has a higher precedence than $+$



Step 1: Precedence and Associativity Rule Out Undesirable Derivations

Input $id + id + id$

Input $id * id + id$

$E \xrightarrow{rm} E + E$

E

$E \xrightarrow{rm} E + E$

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 $+$

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 2: Shift Reduce Actions, Rightmost Derivations, and Handles

$$E \xRightarrow{rm} E + E \xRightarrow{rm} E + E * E \xRightarrow{rm} E + E * id \xRightarrow{rm} E + id * id \xRightarrow{rm} id + id * id$$

$$1 \ E \rightarrow E + E$$

$$2 \ E \rightarrow E * E$$

$$3 \ E \rightarrow id$$

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	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 2: Shift Reduce Actions, Rightmost Derivations, and Handles

$$E \xRightarrow{rm} E + E \xRightarrow{rm} E + E * E \xRightarrow{rm} E + E * id \xRightarrow{rm} E + id * id \xRightarrow{rm} id + id * id$$

$$1 \ E \rightarrow E + E$$

$$2 \ E \rightarrow E * E$$

$$3 \ E \rightarrow id$$

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	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 2: Shift Reduce Actions, Rightmost Derivations, and Handles

$$E \xRightarrow{rm} E + E \xRightarrow{rm} E + E * E \xRightarrow{rm} E + E * id \xRightarrow{rm} \boxed{E + id * id} \xRightarrow{rm} \boxed{id + id * id}$$

$$1 \ E \rightarrow E + E$$

$$2 \ E \rightarrow E * E$$

$$3 \ E \rightarrow id$$

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

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 2: Shift Reduce Actions, Rightmost Derivations, and Handles

$$E \xRightarrow{rm} E + E \xRightarrow{rm} E + E * E \xRightarrow{rm} \boxed{E + E * id} \xRightarrow{rm} \boxed{E + id * id} \xRightarrow{rm} \boxed{id + id * id}$$

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

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



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

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

$$E \xRightarrow{rm} E + E \xRightarrow{rm} \boxed{E + E * E} \xRightarrow{rm} \boxed{E + E * id} \xRightarrow{rm} \boxed{E + id * id} \xRightarrow{rm} \boxed{id + id * id}$$

$$1 \ E \rightarrow E + E$$

$$2 \ E \rightarrow E * E$$

$$3 \ E \rightarrow id$$

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 2
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

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 $E \rightarrow id$

$E \xRightarrow{rm} E + E \xRightarrow{rm} E + E * E \xRightarrow{rm} E + E * id \xRightarrow{rm} E + id * id \xRightarrow{rm} id + id * id$

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



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

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

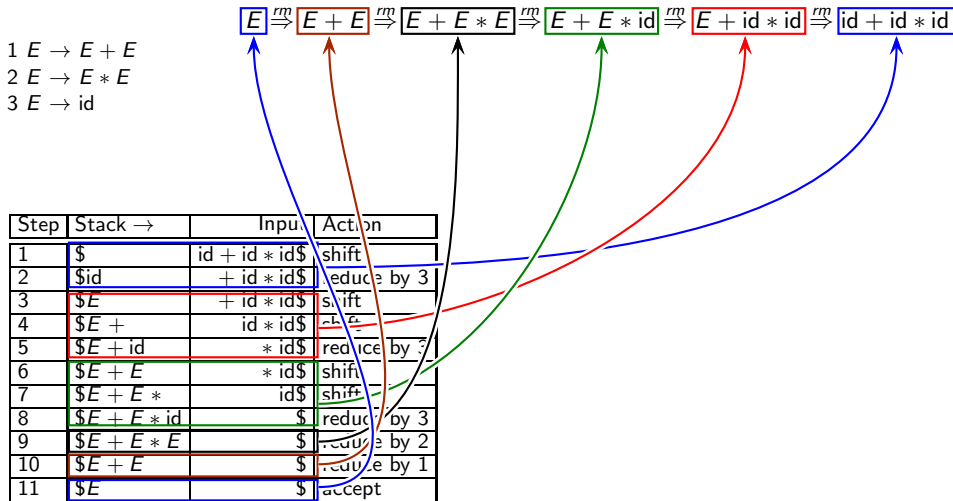
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





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

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 $E \rightarrow id$

$E \xRightarrow{rm} E + E \xRightarrow{rm} E + E * E \xRightarrow{rm} E + E * id \xRightarrow{rm} E + id * id \xRightarrow{rm} id + id * id$

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

Tracing the Rightmost
Derivation in Reverse



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

$$E \xRightarrow{rm} E + E \xRightarrow{rm} E + E * E \xRightarrow{rm} E + E * id \xRightarrow{rm} E + id * id \xRightarrow{rm} id + id * id$$

$$1 \ E \rightarrow E + E$$

$$2 \ E \rightarrow E * E$$

$$3 \ E \rightarrow id$$

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	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

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

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 $E \rightarrow id$

$E \xRightarrow{rm} E + E \xRightarrow{rm} E + E * E \xRightarrow{rm} E + E * id \xRightarrow{rm} \boxed{E} + id * id \xRightarrow{rm} \boxed{id} + id * id$

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

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 $E \rightarrow id$

$$E \xRightarrow{rm} E + E \xRightarrow{rm} E + E * E \xRightarrow{rm} E + \boxed{E} * id \xRightarrow{rm} \boxed{E + id} * id \xRightarrow{rm} id + id * id$$

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

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 $E \rightarrow id$

$E \xRightarrow{rm} E + E \xRightarrow{rm} E + E * E \xRightarrow{rm} E + E * id \xRightarrow{rm} E + id * id \xRightarrow{rm} id + id * id$

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

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 $E \rightarrow id$

$$E \xRightarrow{rm} E + E \xRightarrow{rm} E + E * E \xRightarrow{rm} E + E * id \xRightarrow{rm} E + id * id \xRightarrow{rm} id + id * id$$

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

1 $E \rightarrow E + E$

2 $E \rightarrow E * E$

3 $E \rightarrow id$

$E \xRightarrow{rm} E + E \xRightarrow{rm} E + E * E \xRightarrow{rm} E + E * id \xRightarrow{rm} E + id * id \xRightarrow{rm} id + id * id$

Handle

Right Sentential Form

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

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 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
 - a string that ends with the handle
- 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

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

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

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:
Syntax Analysis

Section:
Grammars,
Derivations, and Parse
Trees

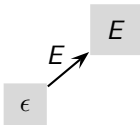
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





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

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:
Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

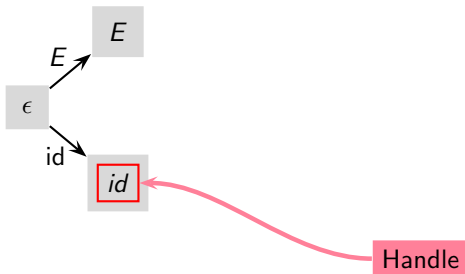
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing



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)

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:
Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

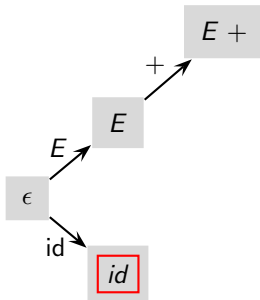
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





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

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:
Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

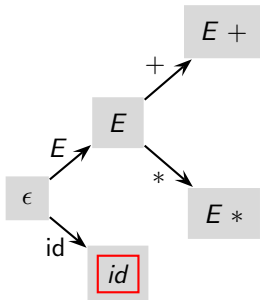
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





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

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:
Syntax Analysis

Section:
Grammars,
Derivations, and Parse
Trees

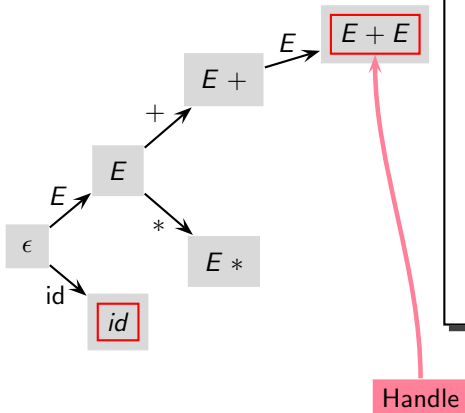
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing



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)

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:
Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

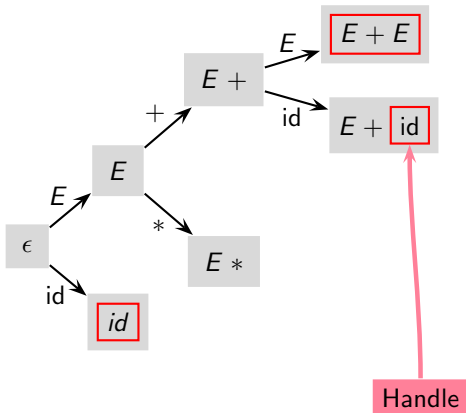
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing



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)

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:
Syntax Analysis

Section:
Grammars,
Derivations, and Parse
Trees

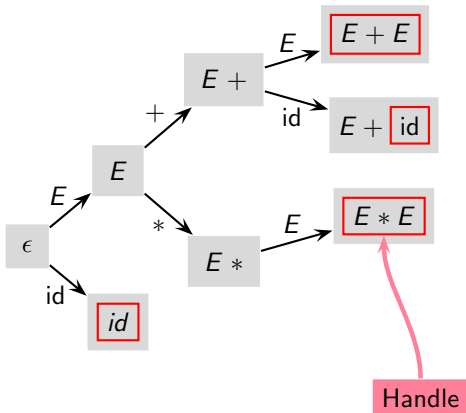
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing



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)

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:
Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

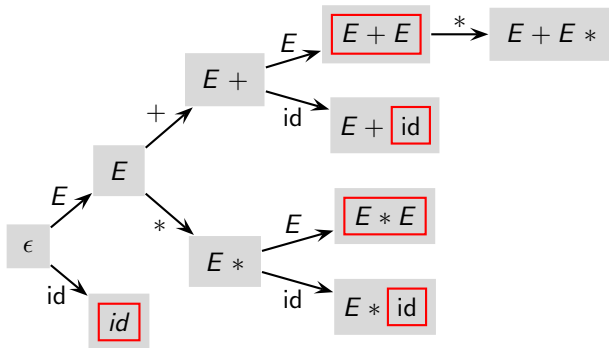
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





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

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:
Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

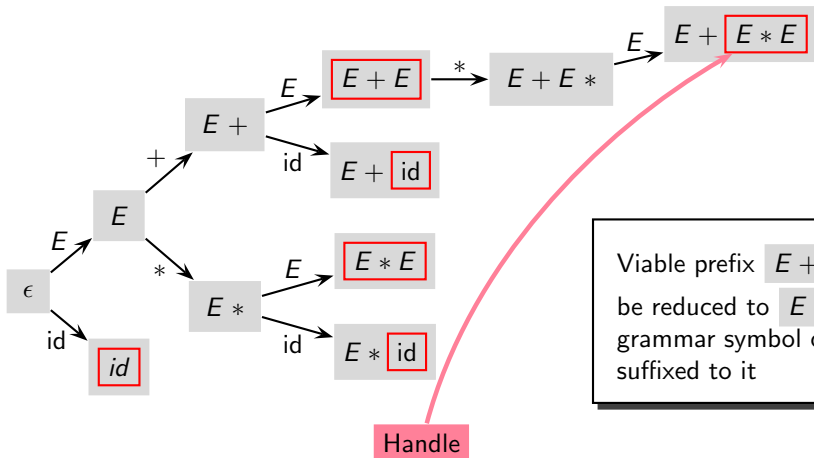
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing



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)

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

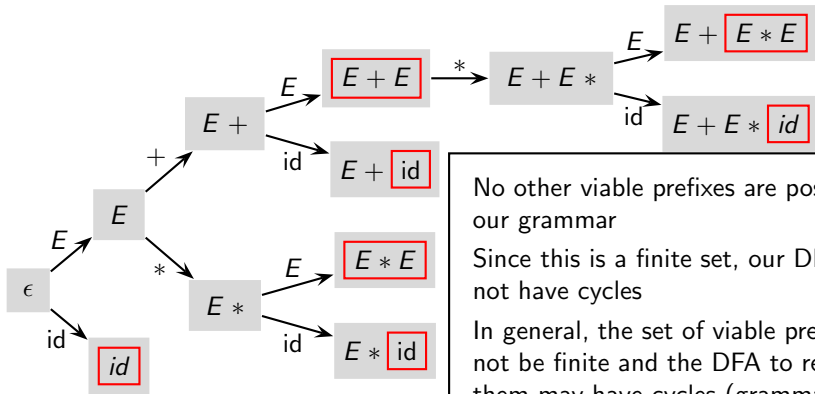
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing



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 $L \rightarrow id, L \mid id$ has infinitely many viable prefixes and its DFA has cycles)



Step 3: Viable Prefixes for Our 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

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

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

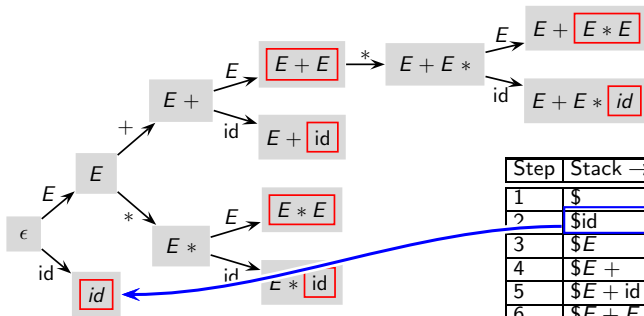
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing



Step	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

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

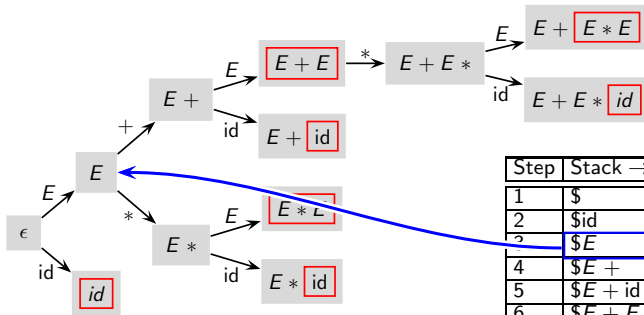
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing



Step	Stack \rightarrow	Input	Action
1	\$	id + id * id\$	shift
2	\$id	+ id * id\$	reduce by 3
3	<u>\$E</u>	+ 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

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

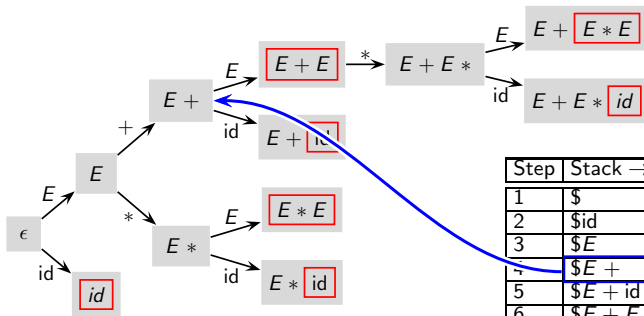
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing



Step	Stack \rightarrow	Input	Action
1	\$	id + id * id\$	shift
2	\$id	+ id * id\$	reduce by 3
3	\$E	+ id * id\$	shift
4	<u>\$E +</u>	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

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

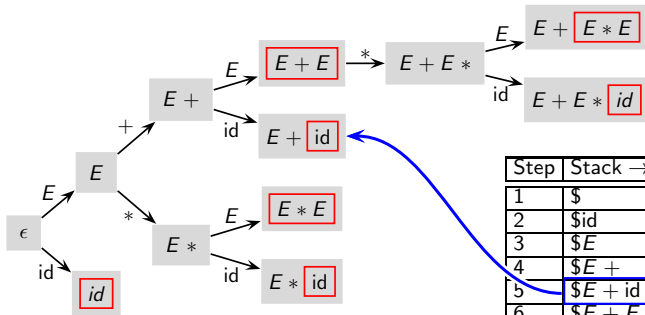
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing



Step	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

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

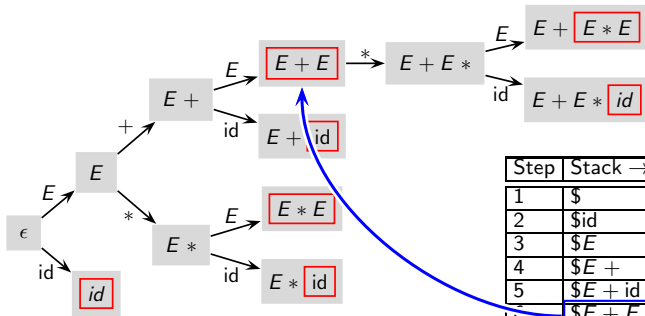
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing



Step	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

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

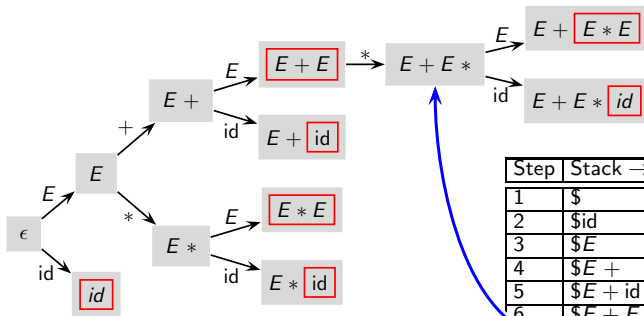
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing



Step	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

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

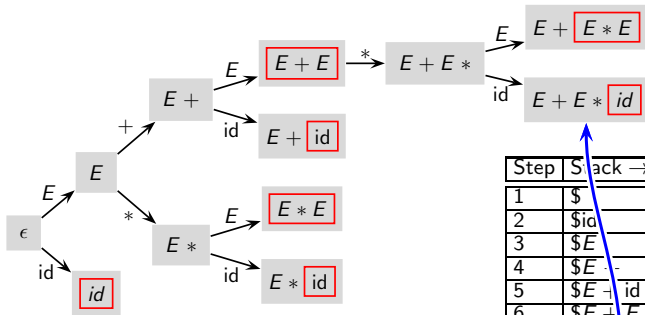
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing



Step	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

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

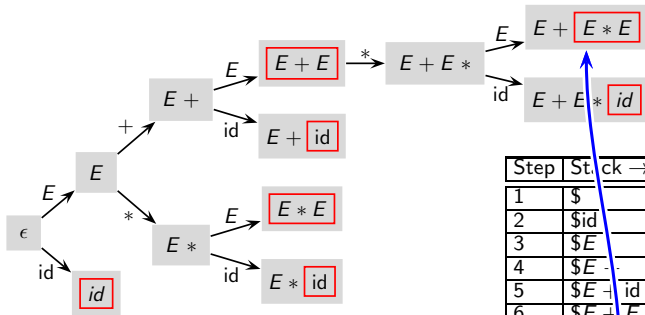
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing



Step	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



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 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 (\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

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

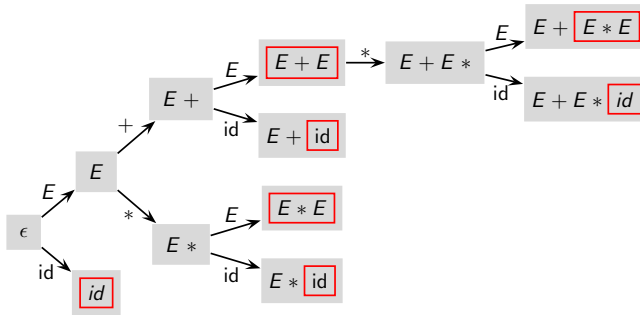
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





Step 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

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

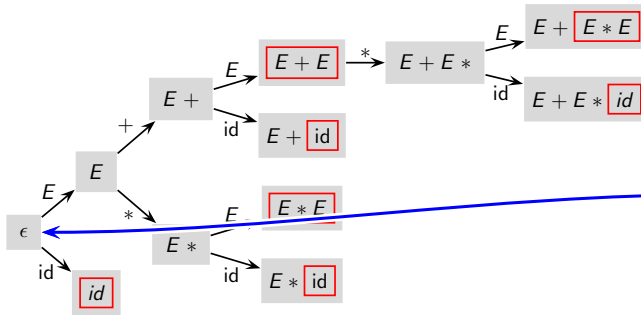
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing



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



IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

Shift Reduce Parsing

SLR(1) Parsing

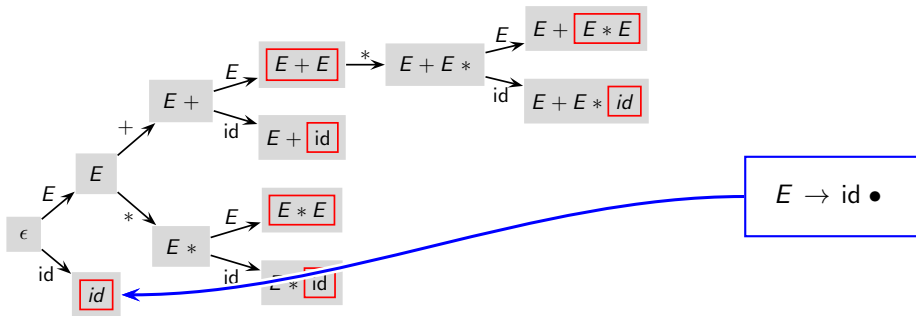
Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing

Step 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

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

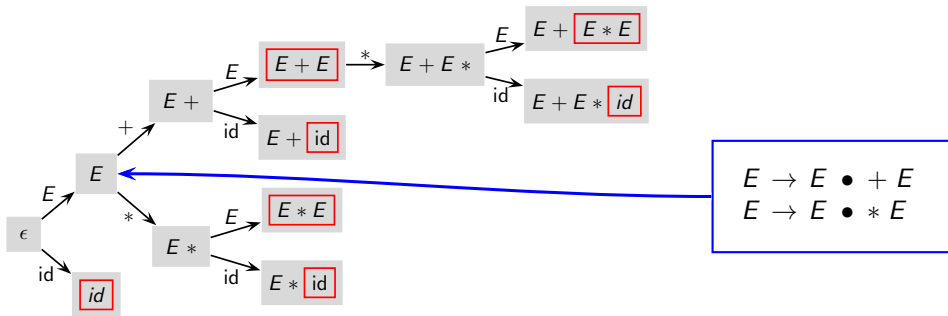
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





Step 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

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

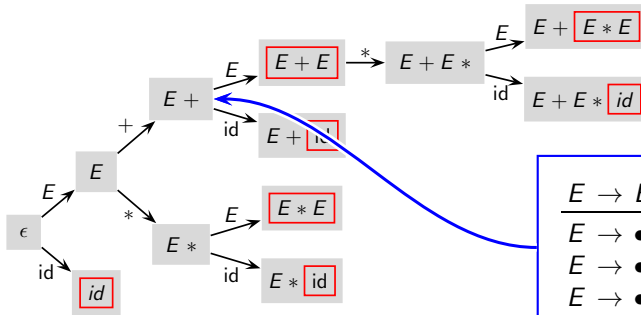
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing



$$\frac{E \rightarrow E + \bullet E \text{ (Kernel Item)}}{E \rightarrow \bullet E + E$$
$$E \rightarrow \bullet E * E \quad \text{(Closure Items)}$$
$$E \rightarrow \bullet id$$



IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

Shift Reduce Parsing

SLR(1) Parsing

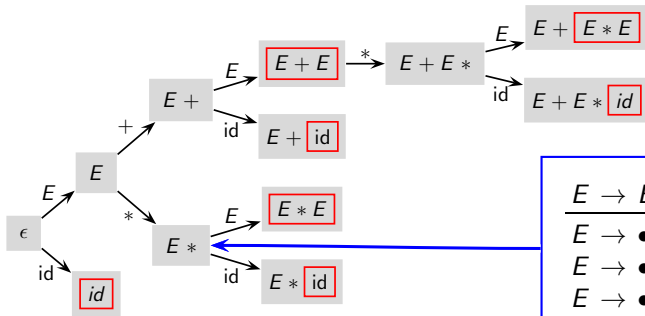
Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing

Step 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)	
<hr/>	
$E \rightarrow \bullet E + E$	(Closure Items)
$E \rightarrow \bullet E * E$	
$E \rightarrow \bullet id$	



IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

Shift Reduce Parsing

SLR(1) Parsing

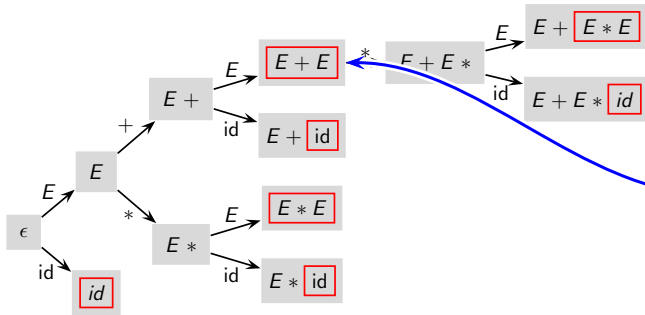
Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing

Step 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



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



IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars, Derivations, and Parse Trees

Shift Reduce Parsing

SLR(1) Parsing

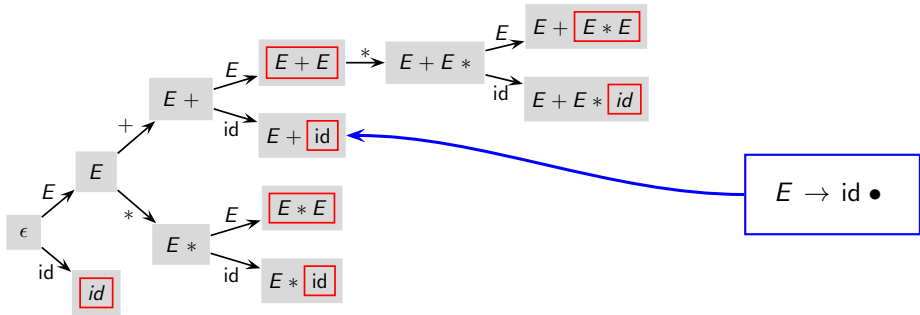
Conceptual Issues in Parsing

CLR(1) Parsing

LALR(1) Parsing

Step 4: Valid Items for Viable Prefixes

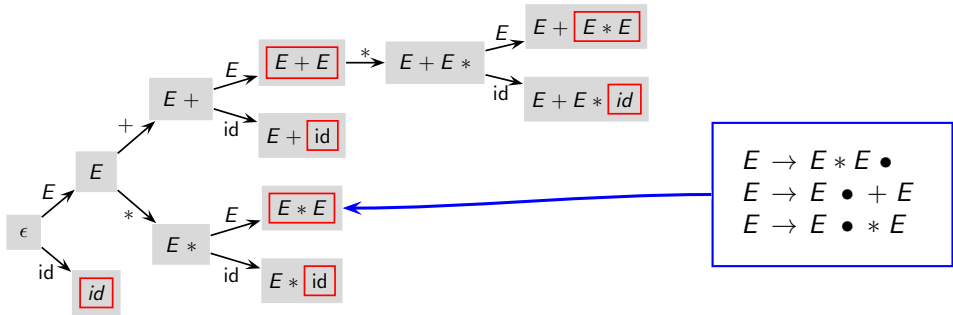
- An item is a grammar production with a dot (\bullet) in it somewhere in the RHS
- The dot separates what has been seen from what may be seen in the input
- We identify a set of items for a viable prefix to form a state of the parser





Step 4: Valid Items for Viable Prefixes

- An item is a grammar production with a dot (\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





IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

Shift Reduce Parsing

SLR(1) Parsing

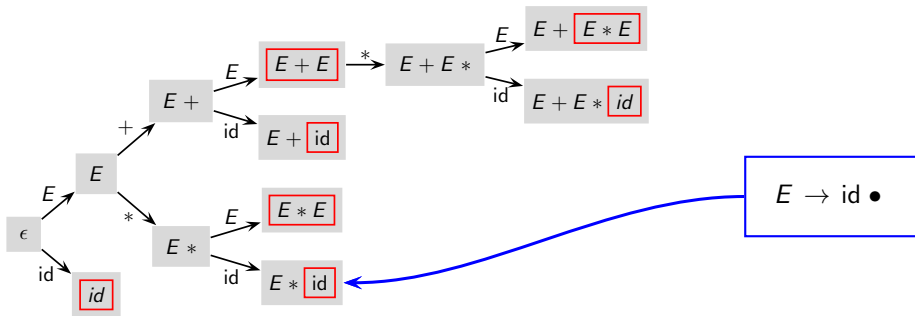
Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing

Step 4: Valid Items for Viable Prefixes

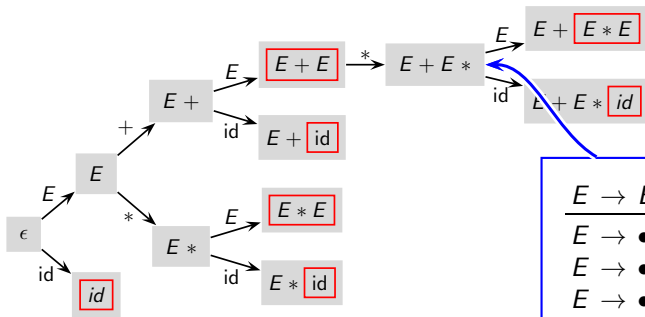
- An item is a grammar production with a dot (•) in it somewhere in the RHS
- The dot separates what has been seen from what may be seen in the input
- We identify a set of items for a viable prefix to form a state of the parser





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



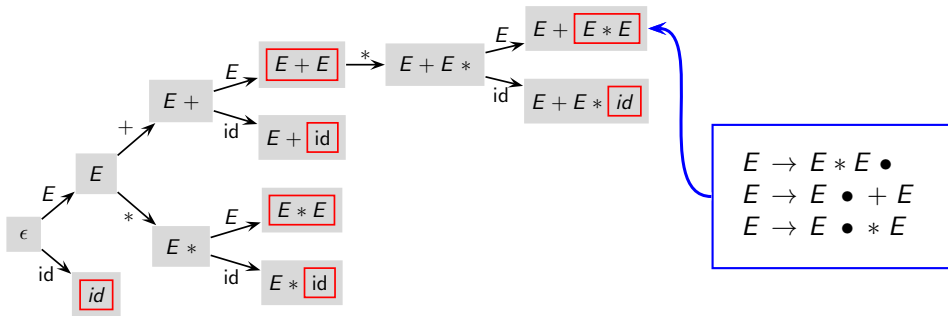
$E \rightarrow E * \bullet E$ (Kernel Item)

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



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



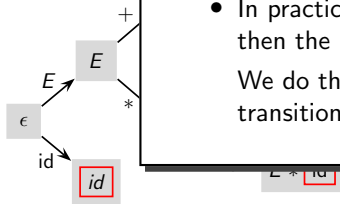


Step 4: Valid Items for Viable Prefixes

- 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

- 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



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



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

Consider $\beta A w \xRightarrow{rm} \beta \alpha w$ 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





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

Consider $\beta Aw \xRightarrow{rm} \beta \alpha w$ 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



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

Consider $\beta Aw \xRightarrow{rm} \beta \alpha w$ 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
- When we find α in γ and the next input token follows A in γ



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

Consider $\beta A w \xRightarrow{rm} \beta \alpha w$ 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



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

Consider $\beta A w \xRightarrow{rm} \beta \alpha w$ 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



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 5: FIRST and FOLLOW Sets

- $\text{FIRST}(\beta)$ contains the terminals that may begin a string derivable from β

If β derives ϵ , then $\epsilon \in \text{FIRST}(\beta)$

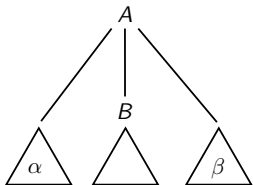
It is computed as the least fixed point solution of the following constraints

For $A \rightarrow X_1 X_2 \dots X_k$, $\text{FIRST}(A) \supseteq X_i$, $1 \leq i \leq k$, provided $\forall j < i, \epsilon \in \text{FIRST}(X_j)$

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



Step 5: FIRST and FOLLOW Sets

- $\text{FIRST}(\beta)$ contains the terminals that may begin a string derivable from β

If β derives ϵ , then $\epsilon \in \text{FIRST}(\beta)$

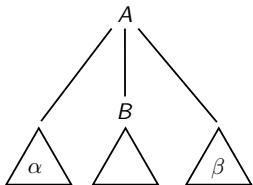
It is computed as the least fixed point solution of the following constraints

For $A \rightarrow X_1 X_2 \dots X_k$, $\text{FIRST}(A) \supseteq X_i, 1 \leq i \leq k$, provided $\forall j < i, \epsilon \in \text{FIRST}(X_j)$

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

For our grammar

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



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

- 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

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

parsing-slides-sanyal-part3.pdf



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

- An item does not contain any lookahead symbol
- Trace the Rightmost derivation in Reverse
- Read the input from Left to right



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



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

$$0 \ E' \rightarrow E$$

$$1 \ E \rightarrow E + E$$

$$2 \ E \rightarrow E * E$$

$$3 \ E \rightarrow \text{id}$$

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

Kernel 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

Closure items



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

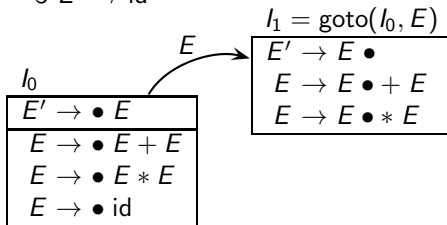
Step 6: Computing LR(0) Item Sets for Expressions Grammar

0 $E' \rightarrow E$

1 $E \rightarrow E + E$

2 $E \rightarrow E * E$

3 $E \rightarrow \text{id}$





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

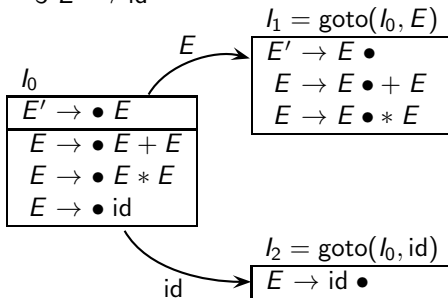
Step 6: Computing LR(0) Item Sets for Expressions Grammar

0 $E' \rightarrow E$

1 $E \rightarrow E + E$

2 $E \rightarrow E * E$

3 $E \rightarrow \text{id}$





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

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

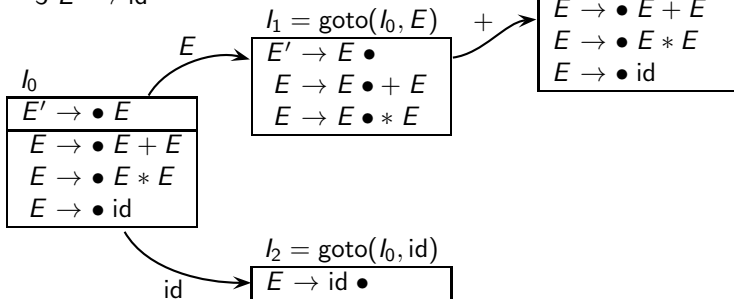
LALR(1) Parsing

0 $E' \rightarrow E$

1 $E \rightarrow E + E$

2 $E \rightarrow E * E$

3 $E \rightarrow id$





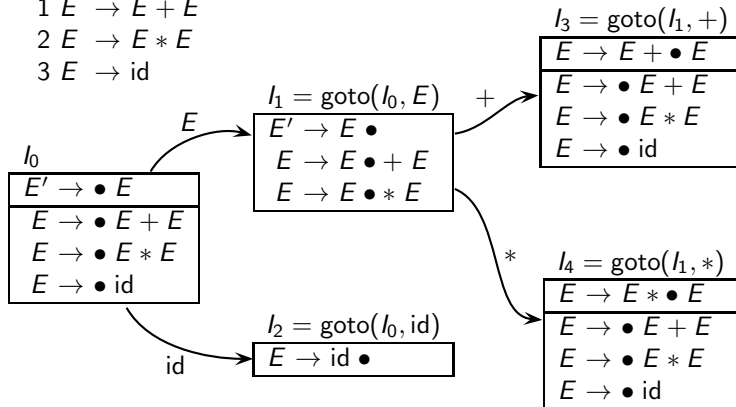
Step 6: Computing LR(0) Item Sets for Expressions Grammar

0 $E' \rightarrow E$

1 $E \rightarrow E + E$

2 $E \rightarrow E * E$

3 $E \rightarrow id$





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

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

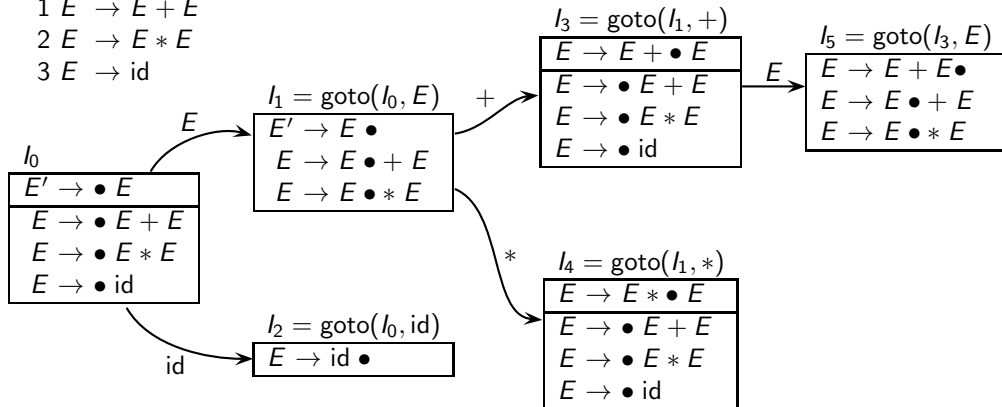
LALR(1) Parsing

0 $E' \rightarrow E$

1 $E \rightarrow E + E$

2 $E \rightarrow E * E$

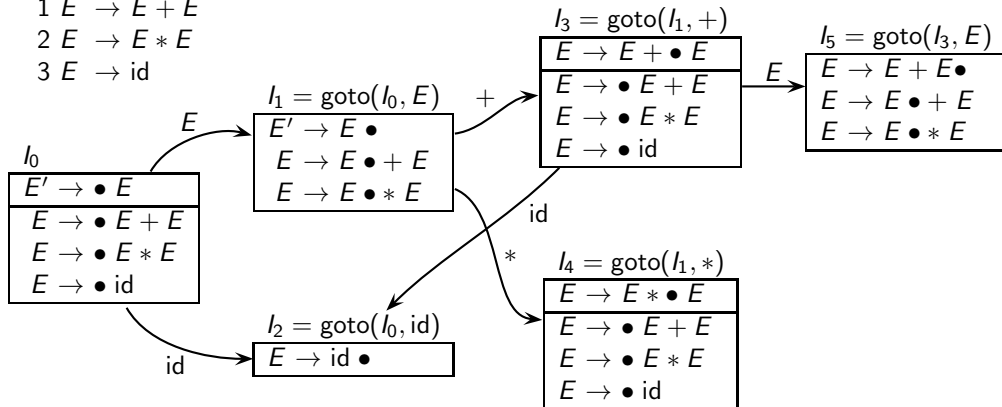
3 $E \rightarrow id$





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 $E \rightarrow id$





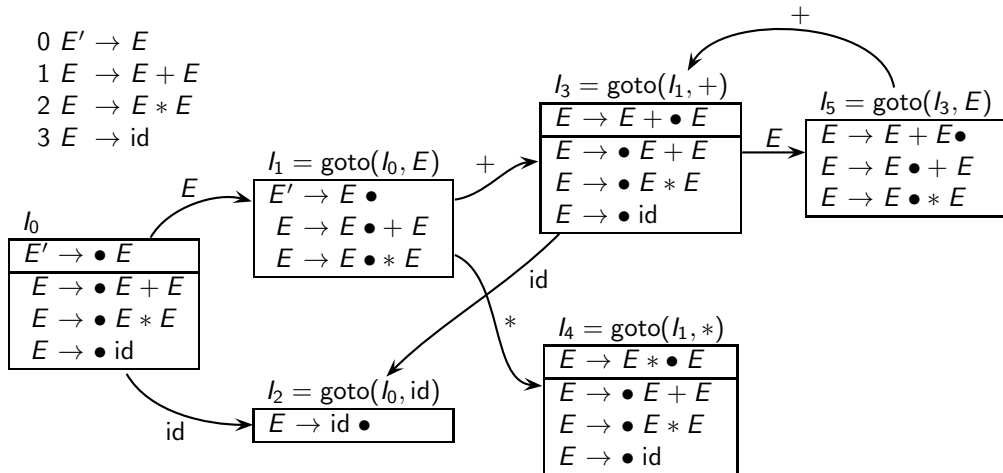
Step 6: Computing LR(0) Item Sets for Expressions Grammar

0 $E' \rightarrow E$

1 $E \rightarrow E + E$

2 $E \rightarrow E * E$

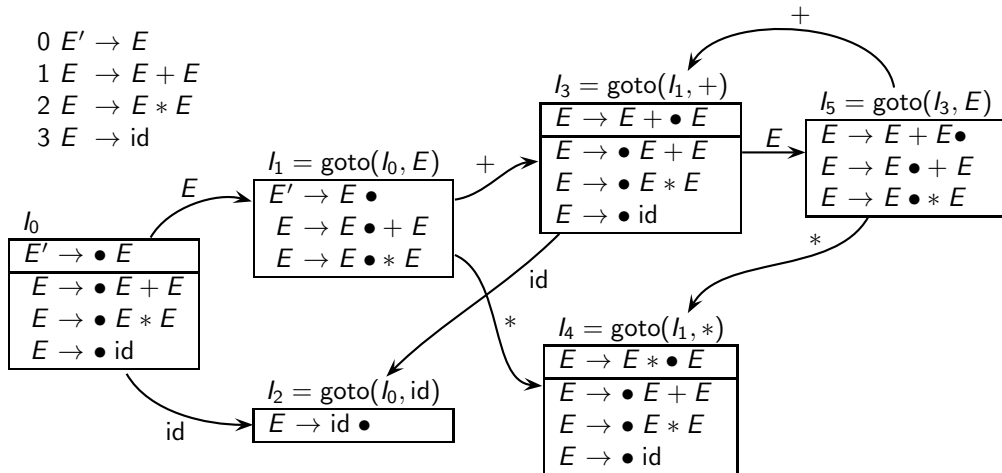
3 $E \rightarrow id$





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 $E \rightarrow id$





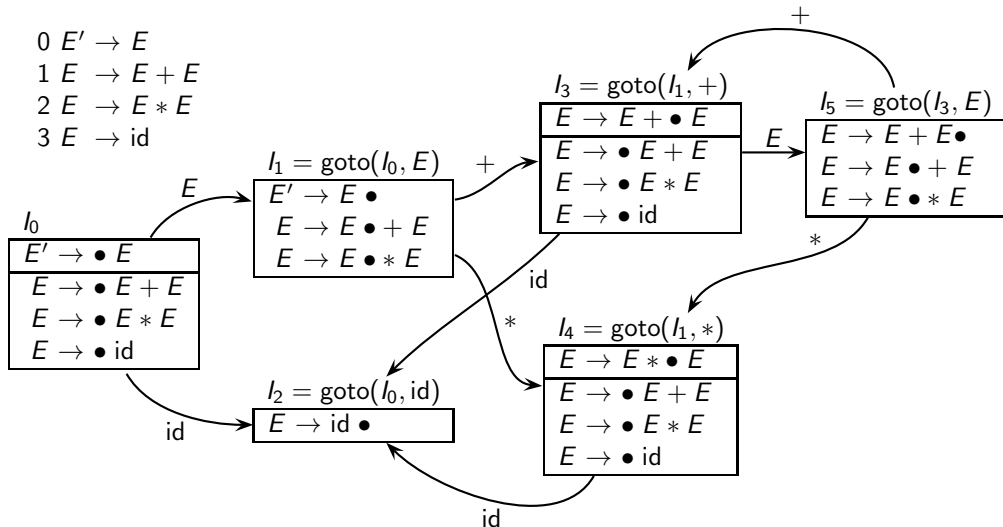
Step 6: Computing LR(0) Item Sets for Expressions Grammar

0 $E' \rightarrow E$

1 $E \rightarrow E + E$

2 $E \rightarrow E * E$

3 $E \rightarrow id$





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 $E \rightarrow id$

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:
Syntax Analysis

Section:
Grammars,
Derivations, and Parse
Trees

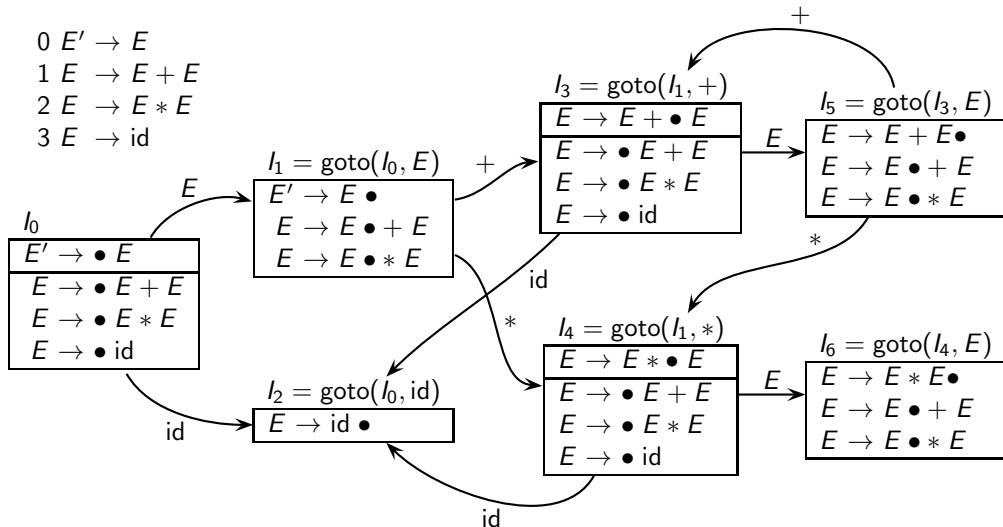
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





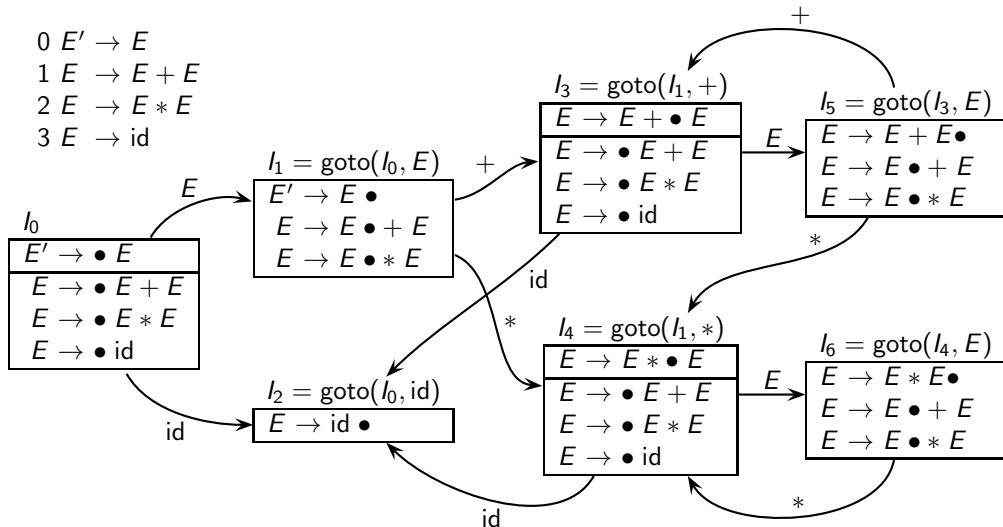
Step 6: Computing LR(0) Item Sets for Expressions Grammar

0 $E' \rightarrow E$

1 $E \rightarrow E + E$

2 $E \rightarrow E * E$

3 $E \rightarrow id$





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 $E \rightarrow id$

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

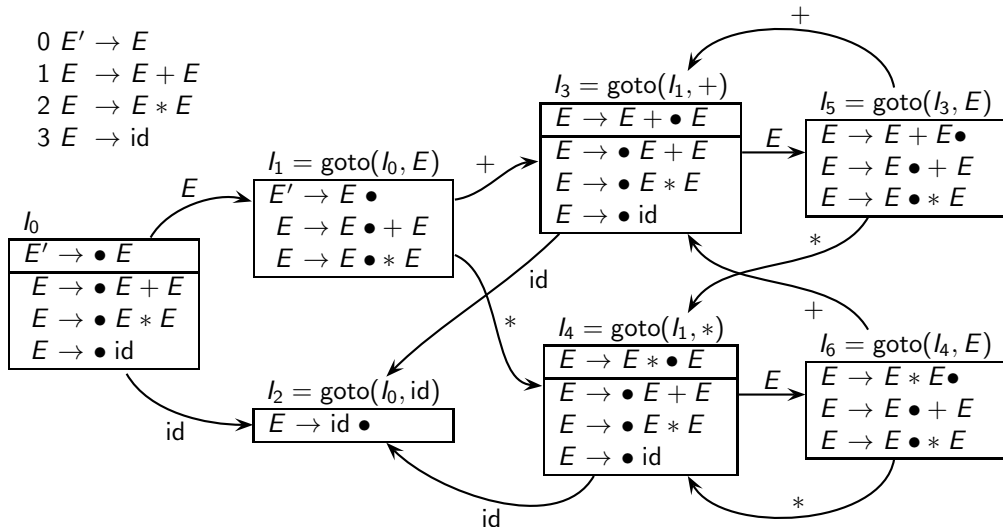
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing



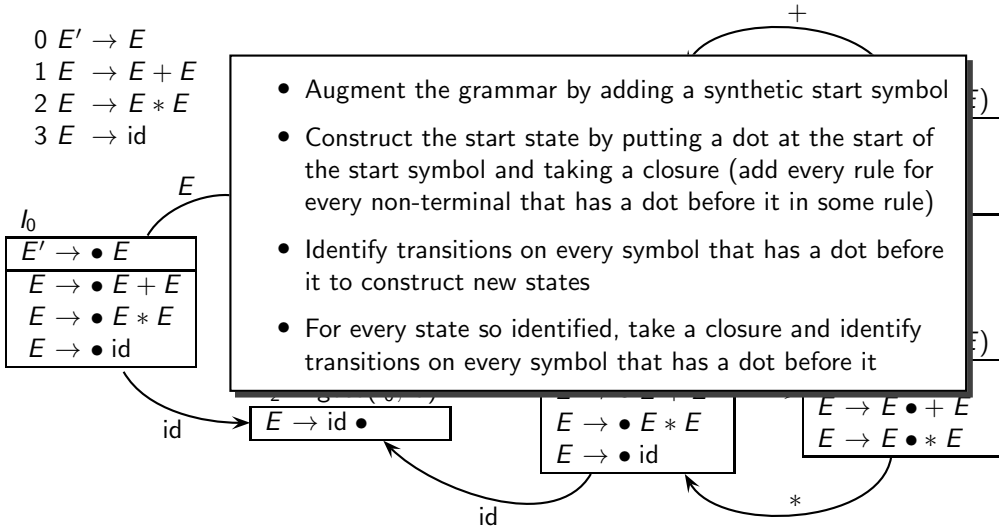


Step 6: Computing LR(0) Item Sets for Expressions Grammar

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

- 0 $E' \rightarrow E$
- 1 $E \rightarrow E + E$
- 2 $E \rightarrow E * E$
- 3 $E \rightarrow id$





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

0 $E' \rightarrow E$

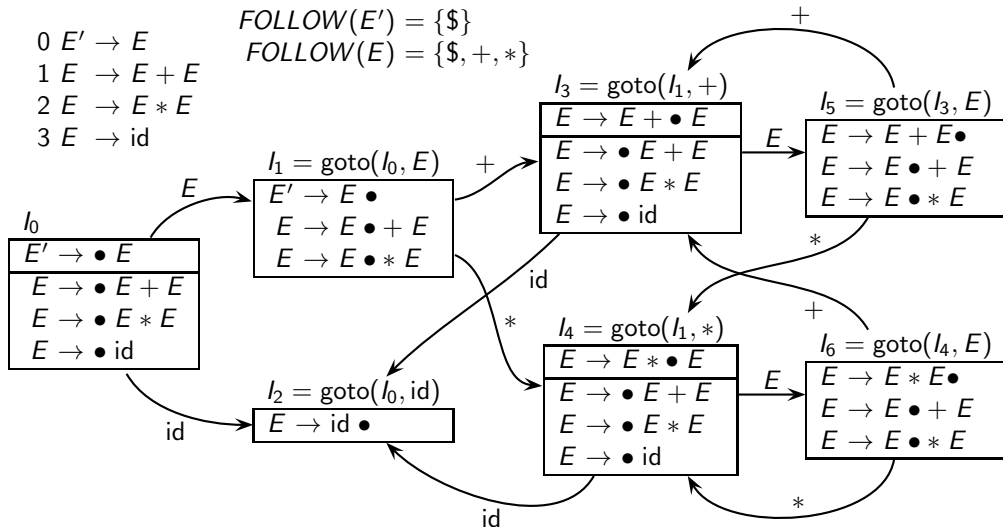
1 $E \rightarrow E + E$

2 $E \rightarrow E * E$

3 $E \rightarrow id$

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

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

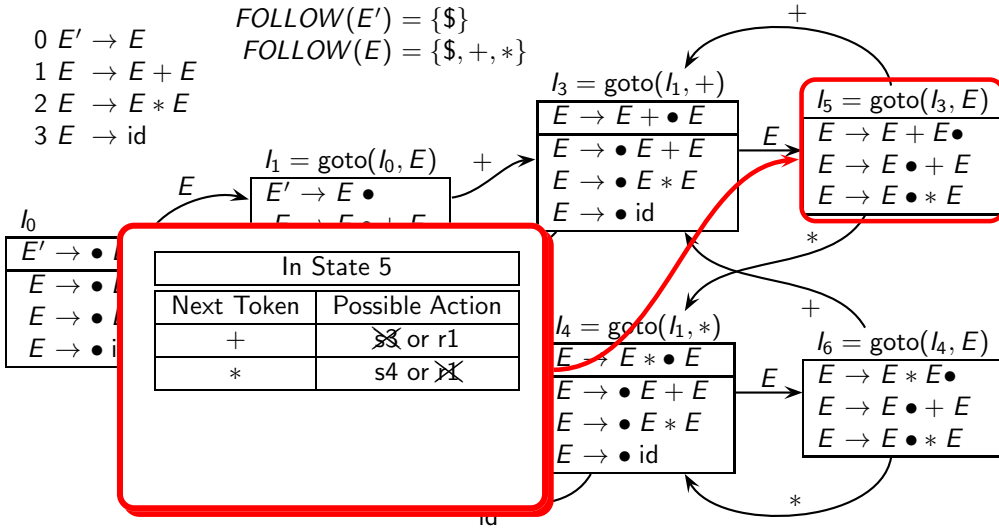




Step 6: Computing LR(0) Item Sets for Expressions Grammar

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

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:
Syntax Analysis
Section:
Grammars,
Derivations, and Parse
Trees

Shift Reduce Parsing

SLR(1) Parsing

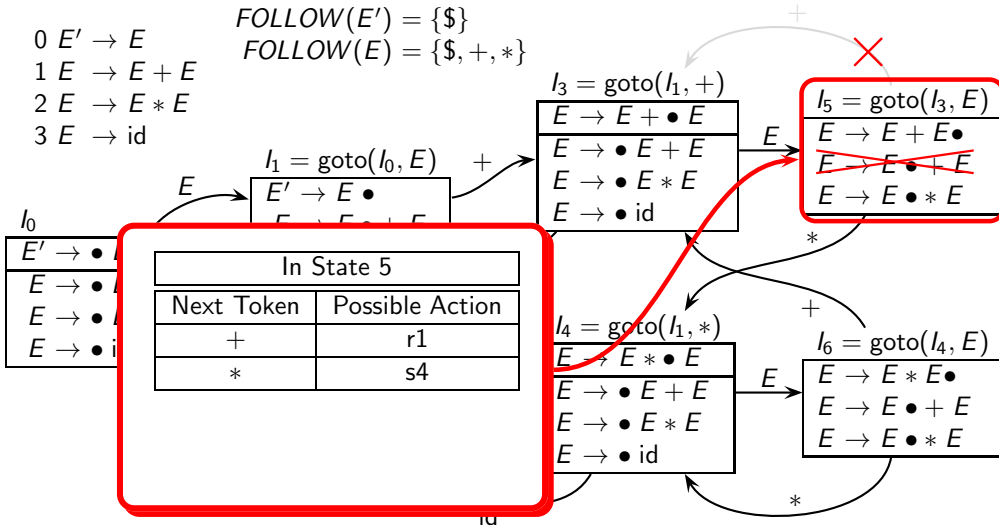
Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing

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

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





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

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:
Syntax Analysis

Section:
Grammars,
Derivations, and Parse
Trees

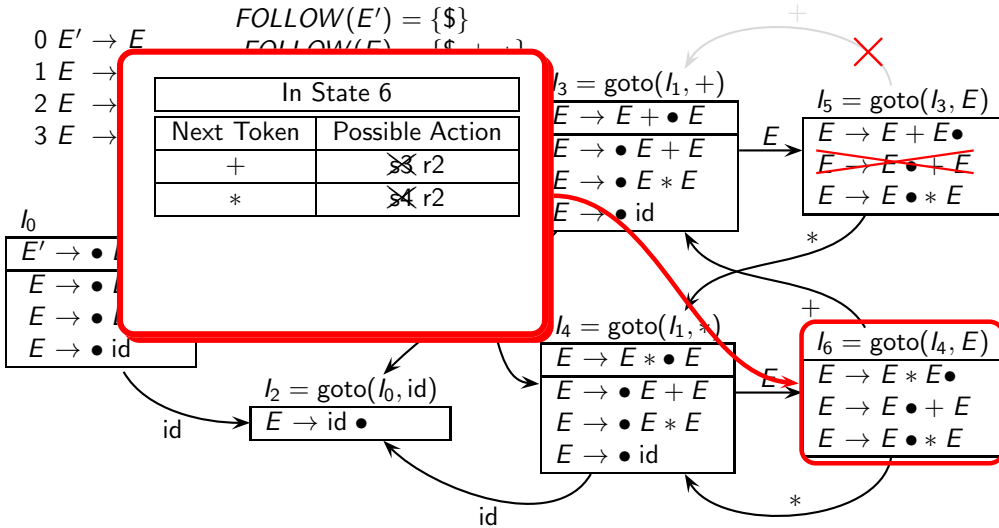
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing



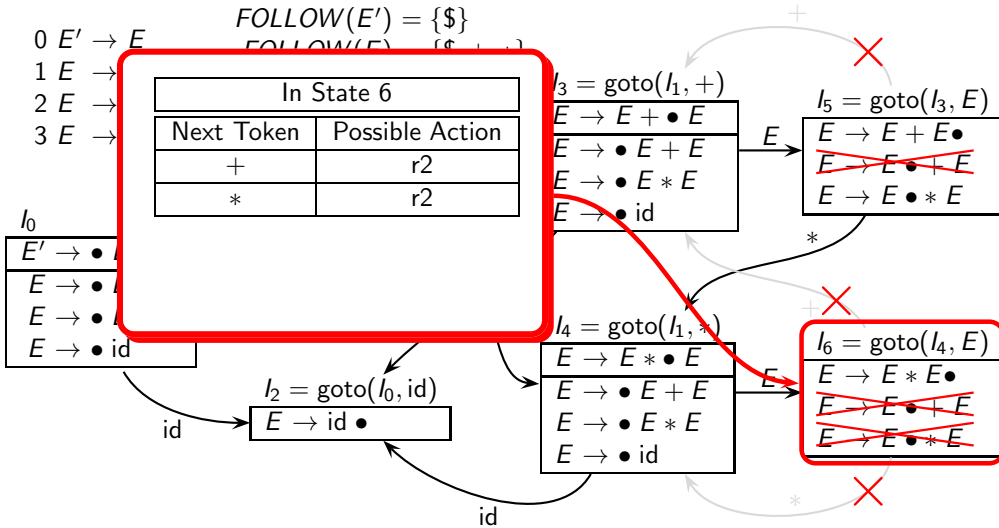


Step 6: Computing LR(0) Item Sets for Expressions Grammar

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





IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:
Syntax Analysis

Section:
Grammars,
Derivations, and Parse
Trees

Shift Reduce Parsing

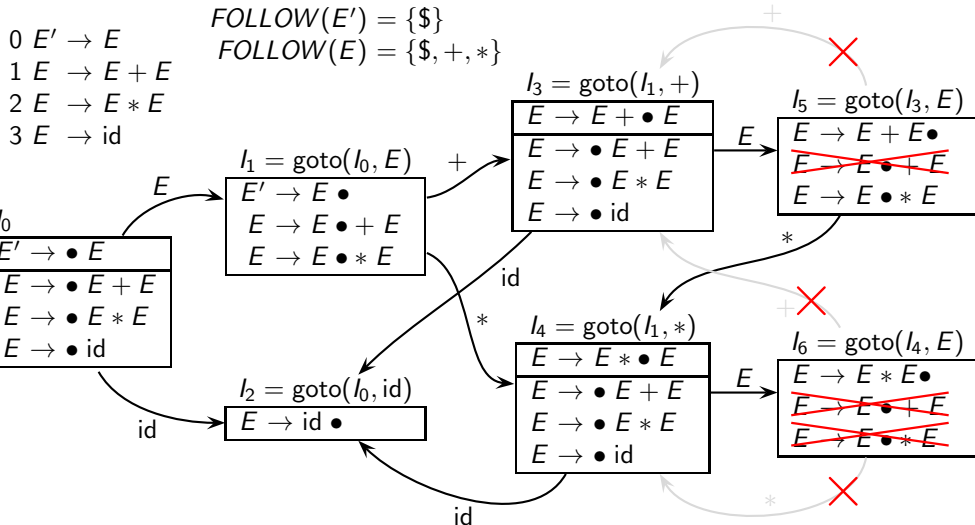
SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing

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





The DFA of Item Sets Accepts Viable Prefixes

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

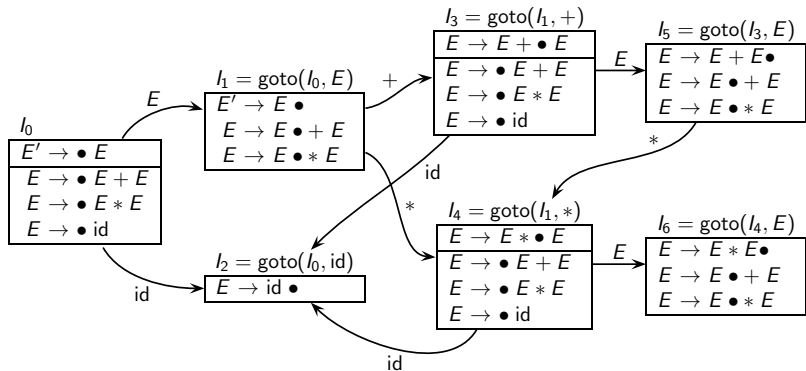
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





The DFA of Item Sets Accepts Viable Prefixes

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:
Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

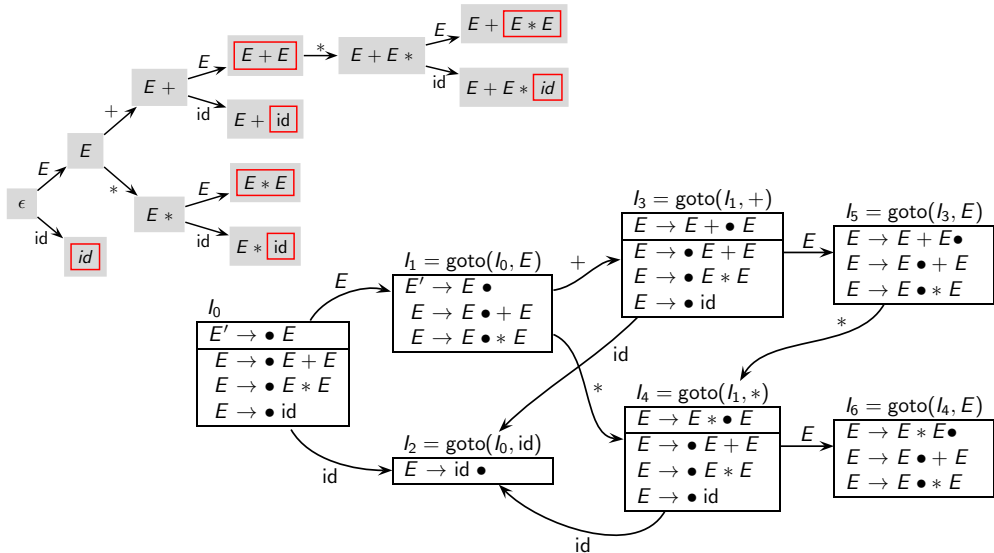
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





Putting it All Together: Constructing the Parsing Table

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

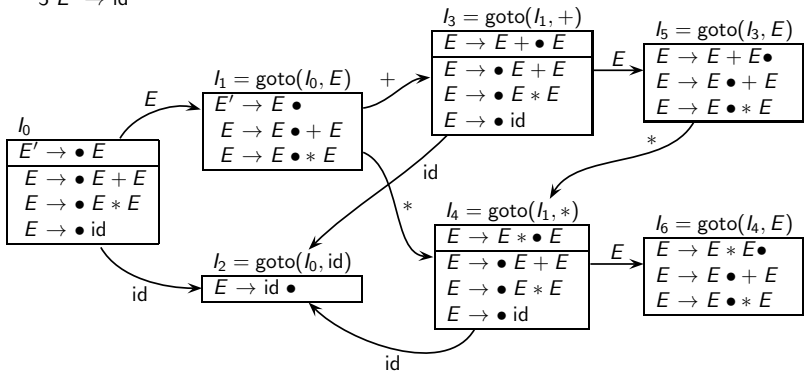
CLR(1) Parsing

LALR(1) Parsing

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

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

	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

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

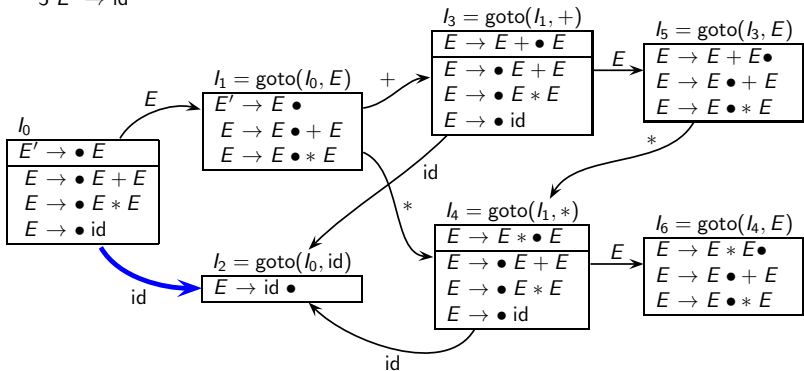
CLR(1) Parsing

LALR(1) Parsing

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

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

	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

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

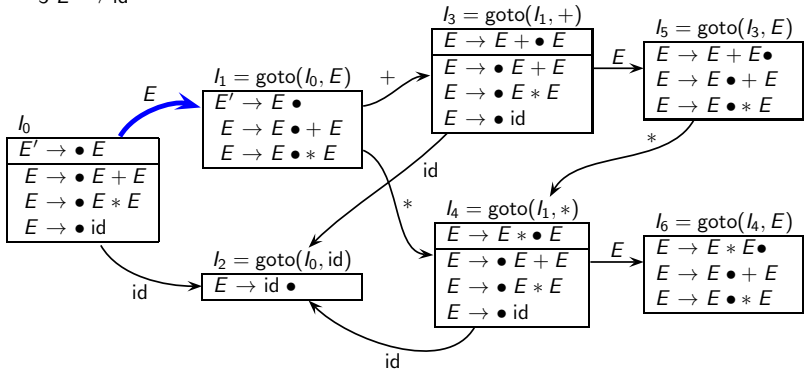
CLR(1) Parsing

LALR(1) Parsing

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

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

	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	r5	
3	s2				c5
4	s2				c6
5		r1	s4	r1	
6		r2	r2	r2	

0 $E' \rightarrow E$

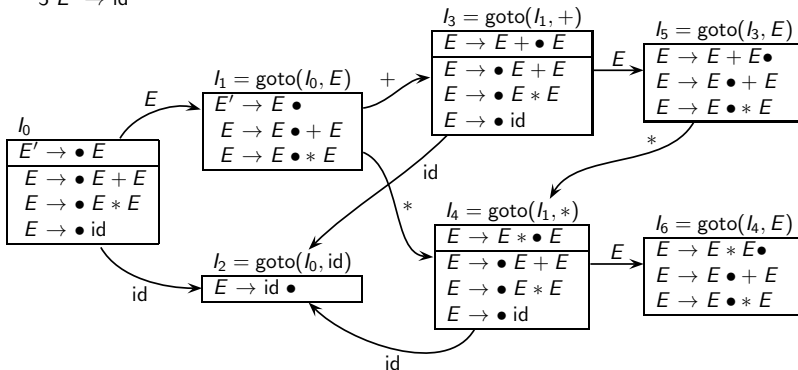
1 $E \rightarrow E + E$

2 $E \rightarrow E * E$

3 $E \rightarrow id$

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

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





Putting it All Together: Constructing the Parsing Table

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

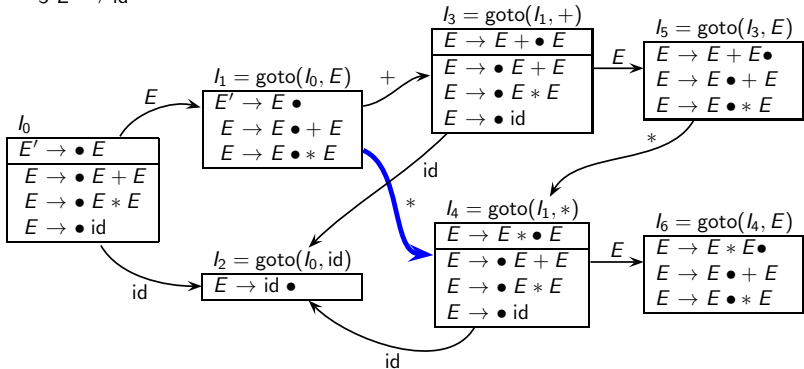
CLR(1) Parsing

LALR(1) Parsing

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

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

	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

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

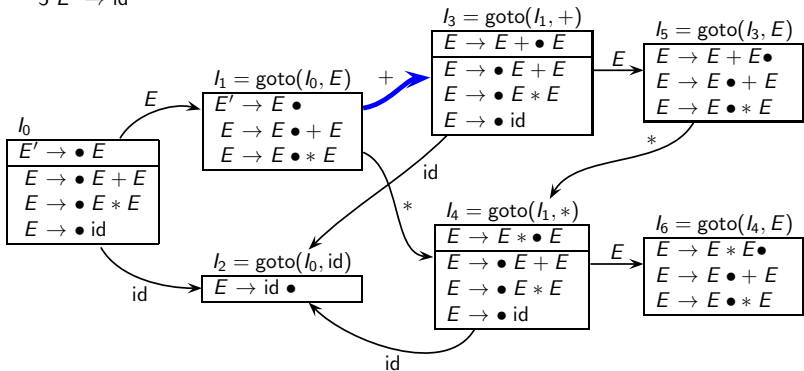
CLR(1) Parsing

LALR(1) Parsing

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

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

	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	

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

Shift Reduce Parsing

SLR(1) Parsing

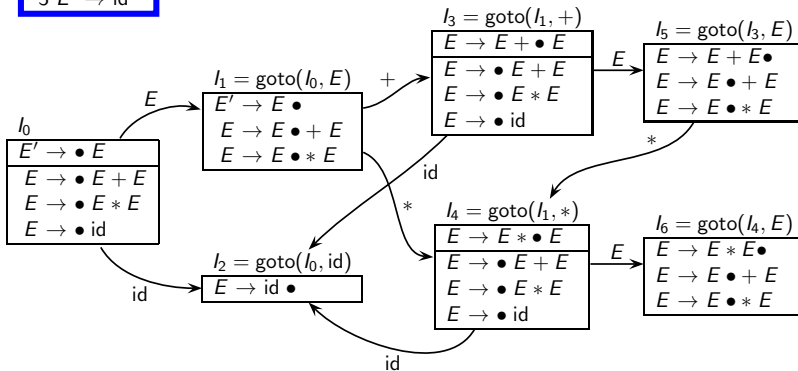
Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing

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

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





Putting it All Together: Constructing the Parsing Table

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

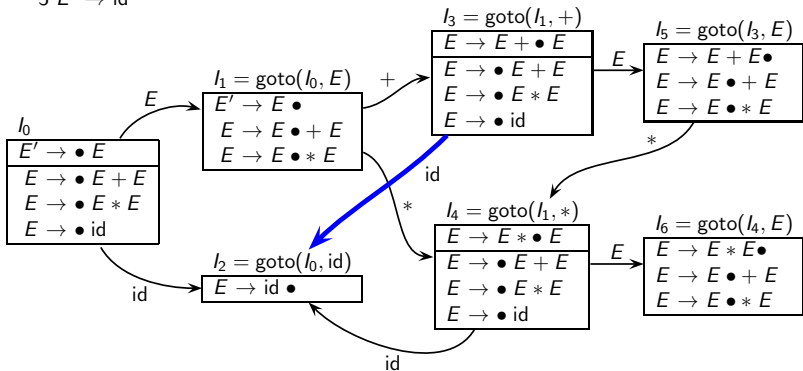
CLR(1) Parsing

LALR(1) Parsing

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

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

	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

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

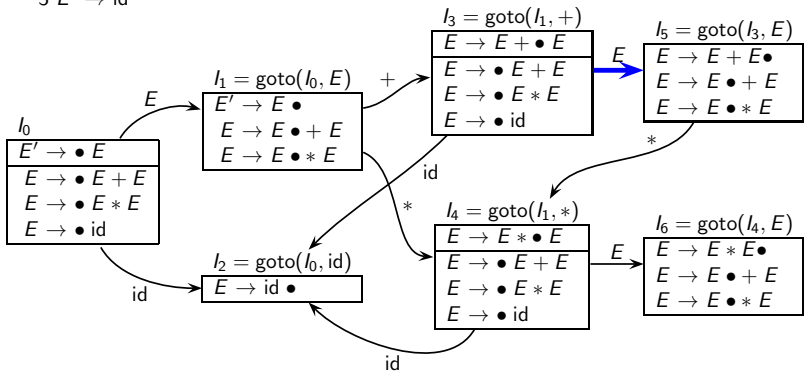
CLR(1) Parsing

LALR(1) Parsing

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

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

	id	+	*	\$	E
0	s2				c1
1		s3	s4	acc	
2		r3	r3	r3	
3	s2				c5
4	s2				c0
5		r1	s4	r1	
6		r2	r2	r2	





Putting it All Together: Constructing the Parsing Table

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

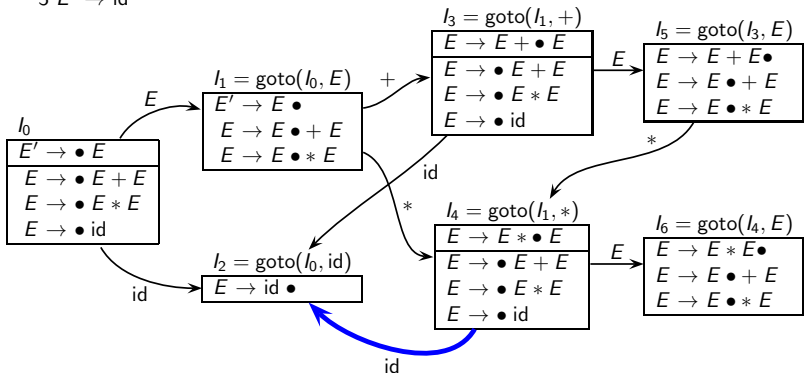
CLR(1) Parsing

LALR(1) Parsing

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

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

	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

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

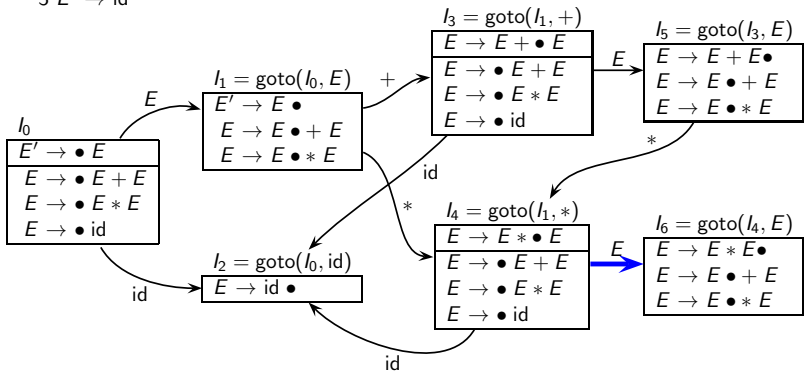
CLR(1) Parsing

LALR(1) Parsing

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

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

	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

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

0 $E' \rightarrow E$

1 $E \rightarrow E + E$

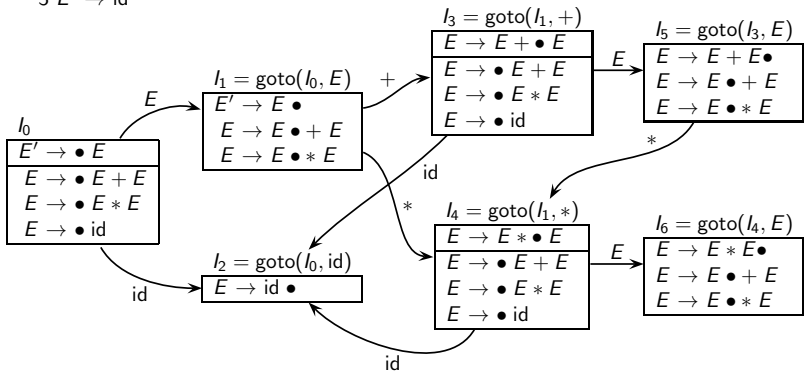
2 $E \rightarrow E * E$

3 $E \rightarrow id$

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

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

	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

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

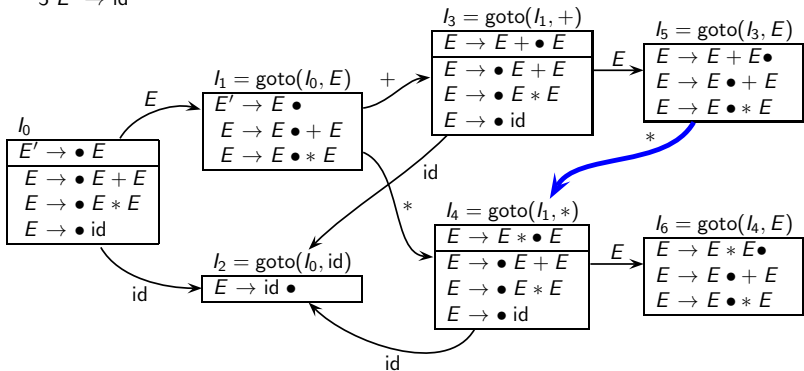
LALR(1) Parsing

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

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

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

	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

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

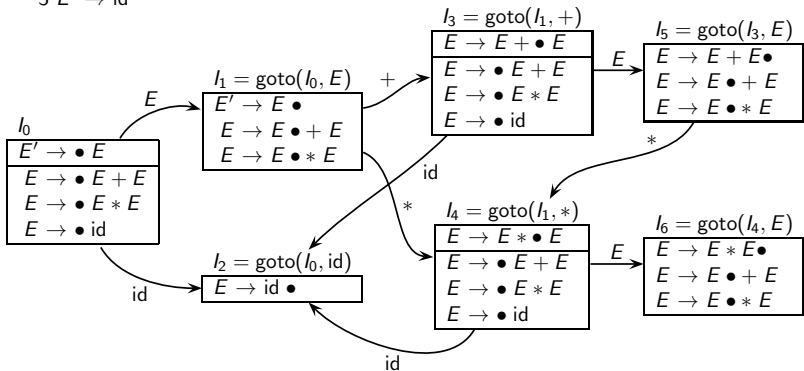
CLR(1) Parsing

LALR(1) Parsing

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

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

	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

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 $E \rightarrow id$

	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 \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	Stack \rightarrow	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



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



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

Conceptual Issues in Parsing



Conceptual Issues in Parsing

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

`parsing-slides-sanyal-part4.pdf`



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



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

CLR(1) Parsing



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

- We illustrate the limitations of SLR(1) parsing by using the pointer assignment grammar given below

$$S \rightarrow L = R \mid R$$
$$L \rightarrow *R \mid \text{id}$$
$$R \rightarrow L$$

- 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



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

$$S' \rightarrow S$$

$$S \rightarrow L = R \mid R$$

$$L \rightarrow *R \mid \text{id}$$

$$R \rightarrow L$$

Computing the FOLLOW Sets for Pointer Assignment Grammar



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

$$\begin{aligned} S' &\rightarrow S && \Rightarrow \text{FOLLOW}(S') \supseteq \{\$\} \\ &&& \text{FOLLOW}(S) \supseteq \text{FOLLOW}(S') \end{aligned}$$

$$S \rightarrow L = R \mid R$$

$$\begin{aligned} L &\rightarrow *R \mid \text{id} \\ R &\rightarrow L \end{aligned}$$

Computing the FOLLOW Sets for Pointer Assignment Grammar



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

$$\begin{aligned} S' &\rightarrow S && \Rightarrow \text{FOLLOW}(S') \supseteq \{\$\} \\ & && \text{FOLLOW}(S) \supseteq \text{FOLLOW}(S') \\ S &\rightarrow L = R \mid R && \Rightarrow \text{FOLLOW}(L) \supseteq \{=\} \\ & && \text{FOLLOW}(R) \supseteq \text{FOLLOW}(S) \\ L &\rightarrow *R \mid \text{id} \\ R &\rightarrow L \end{aligned}$$

Computing the FOLLOW Sets for Pointer Assignment Grammar



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

$$\begin{aligned} S' &\rightarrow S && \Rightarrow \text{FOLLOW}(S') \supseteq \{\$\} \\ & && \text{FOLLOW}(S) \supseteq \text{FOLLOW}(S') \\ S &\rightarrow L = R \mid R && \Rightarrow \text{FOLLOW}(L) \supseteq \{=\} \\ & && \text{FOLLOW}(R) \supseteq \text{FOLLOW}(S) \\ L &\rightarrow *R \mid \text{id} && \Rightarrow \text{FOLLOW}(R) \supseteq \text{FOLLOW}(L) \\ R &\rightarrow L \end{aligned}$$

Computing the FOLLOW Sets for Pointer Assignment Grammar



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

$$\begin{aligned} S' &\rightarrow S && \Rightarrow \text{FOLLOW}(S') \supseteq \{\$\} \\ & && \text{FOLLOW}(S) \supseteq \text{FOLLOW}(S') \\ S &\rightarrow L = R \mid R && \Rightarrow \text{FOLLOW}(L) \supseteq \{=\} \\ & && \text{FOLLOW}(R) \supseteq \text{FOLLOW}(S) \\ L &\rightarrow *R \mid \text{id} && \Rightarrow \text{FOLLOW}(R) \supseteq \text{FOLLOW}(L) \\ R &\rightarrow L && \Rightarrow \text{FOLLOW}(L) \supseteq \text{FOLLOW}(R) \end{aligned}$$

Computing the FOLLOW Sets for Pointer Assignment Grammar



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

$$\begin{aligned} S' &\rightarrow S && \Rightarrow \text{FOLLOW}(S') \supseteq \{\$ \} \\ & && \text{FOLLOW}(S) \supseteq \text{FOLLOW}(S') \\ S &\rightarrow L = R \mid R && \Rightarrow \text{FOLLOW}(L) \supseteq \{=\} \\ & && \text{FOLLOW}(R) \supseteq \text{FOLLOW}(S) \\ L &\rightarrow *R \mid \text{id} && \Rightarrow \text{FOLLOW}(R) \supseteq \text{FOLLOW}(L) \\ R &\rightarrow L && \Rightarrow \text{FOLLOW}(L) \supseteq \text{FOLLOW}(R) \end{aligned}$$

	FOLLOW
S'	$\{\$ \}$
S	$\{\$ \}$
R	$\{=, \$ \}$
L	$\{=, \$ \}$

LR(0) Item Sets for Pointer Assignment Grammar



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

I_0	
$S' \rightarrow \bullet S$	
$S \rightarrow \bullet L = R$	
$S \rightarrow \bullet R$	
$L \rightarrow \bullet * R$	
$L \rightarrow \bullet id$	
$R \rightarrow \bullet L$	



LR(0) Item Sets for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

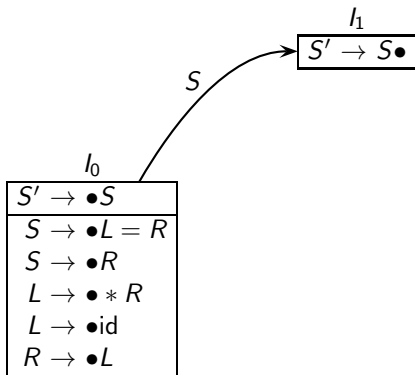
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(0) Item Sets for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

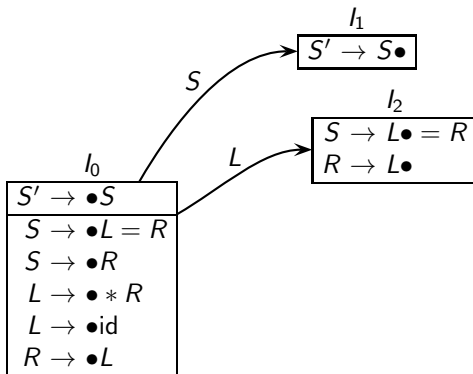
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(0) Item Sets for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

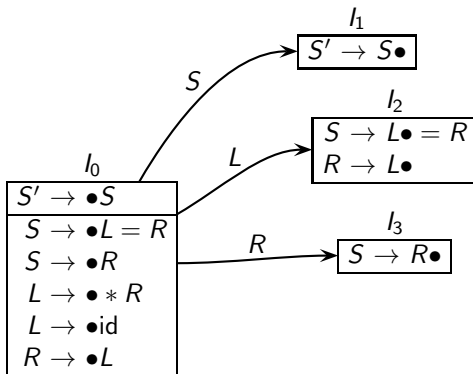
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(0) Item Sets for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

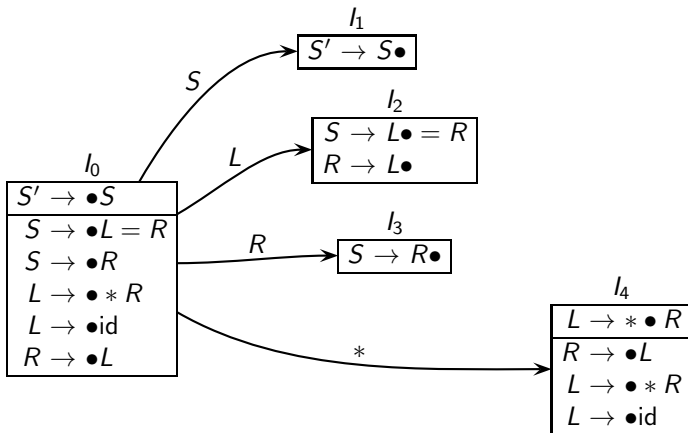
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(0) Item Sets for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

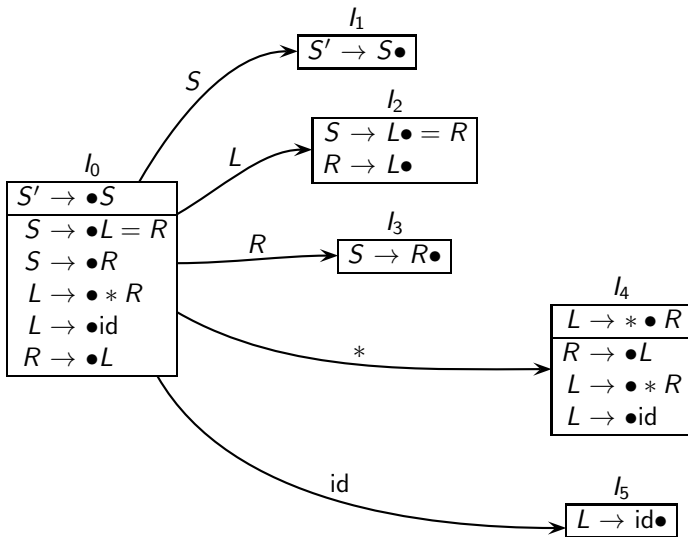
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(0) Item Sets for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

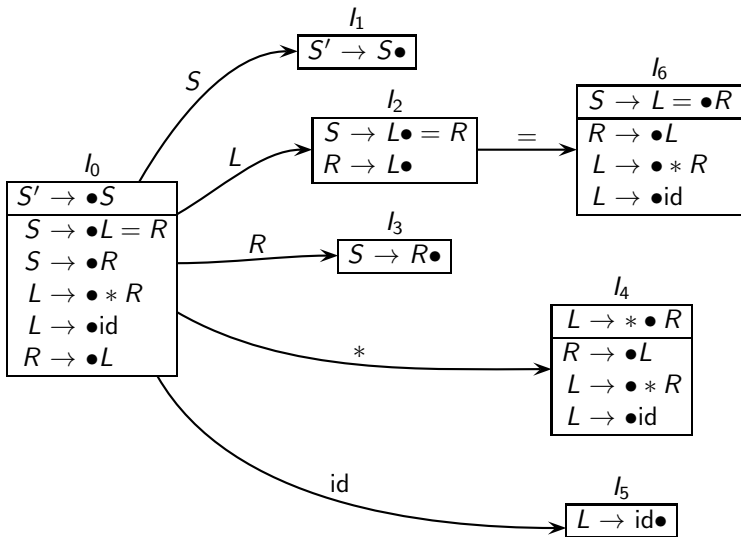
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(0) Item Sets for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

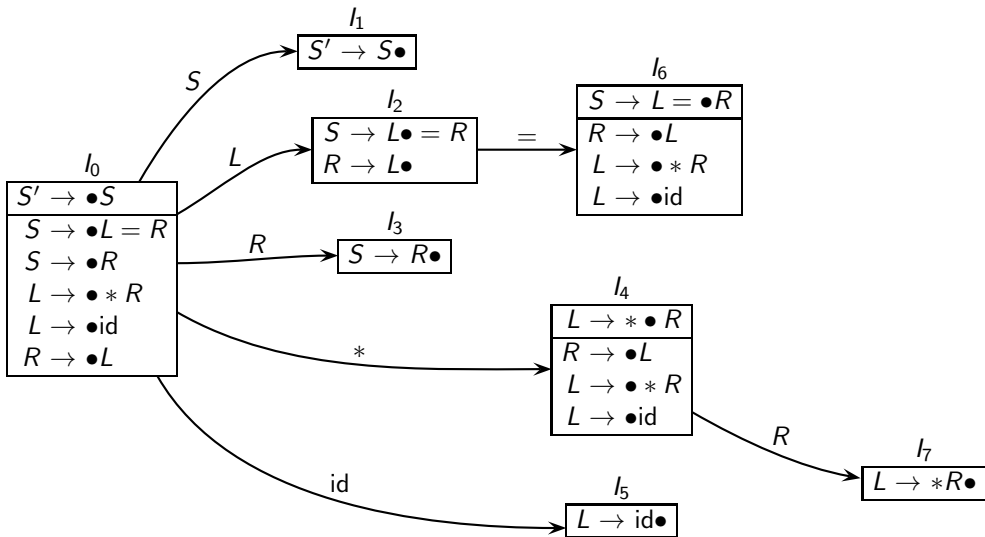
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(0) Item Sets for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

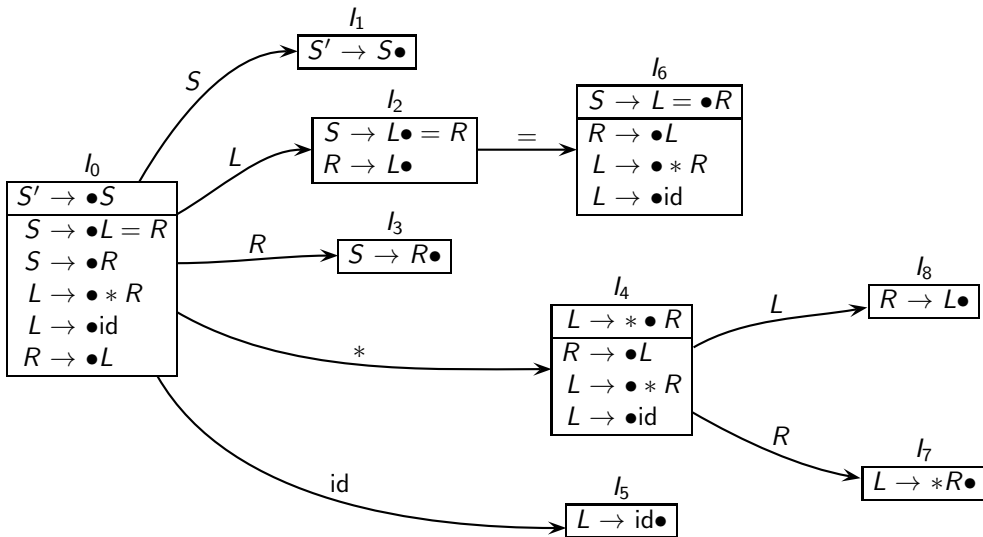
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(0) Item Sets for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

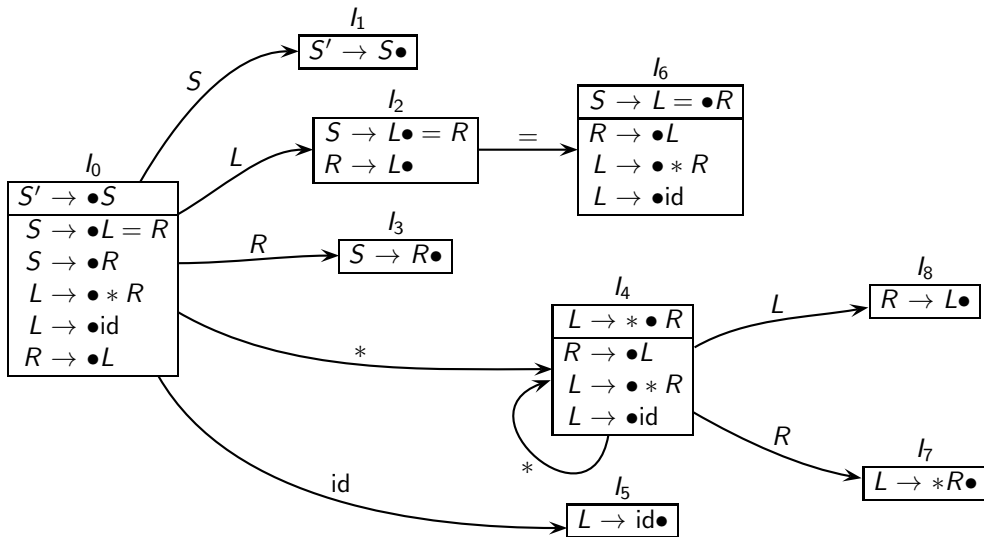
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(0) Item Sets for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

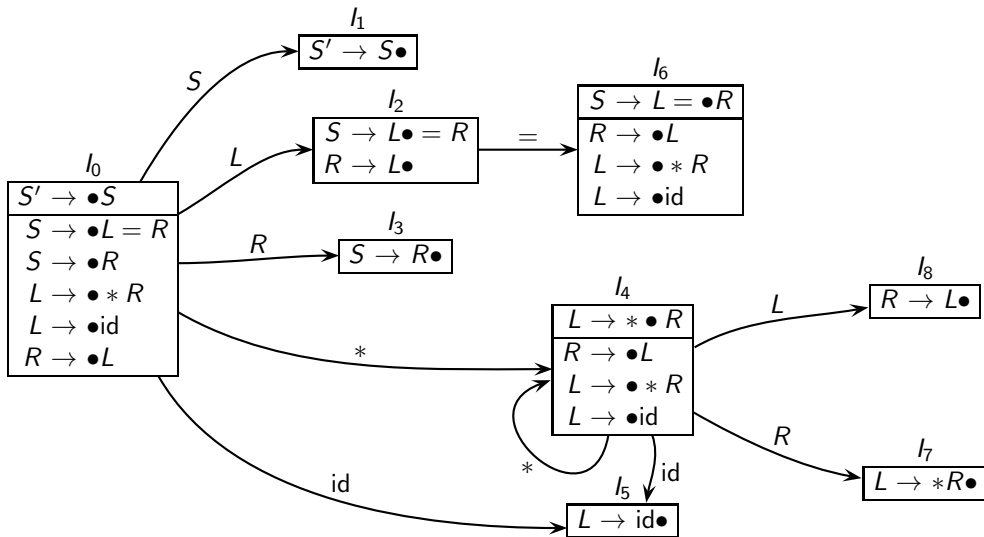
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(0) Item Sets for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

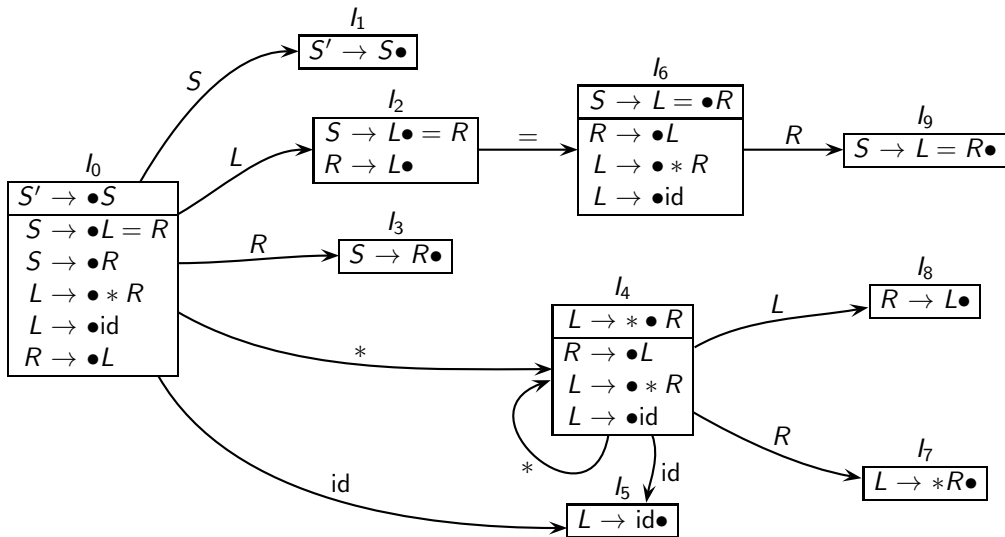
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(0) Item Sets for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:
Syntax Analysis

Section:
Grammars,
Derivations, and Parse
Trees

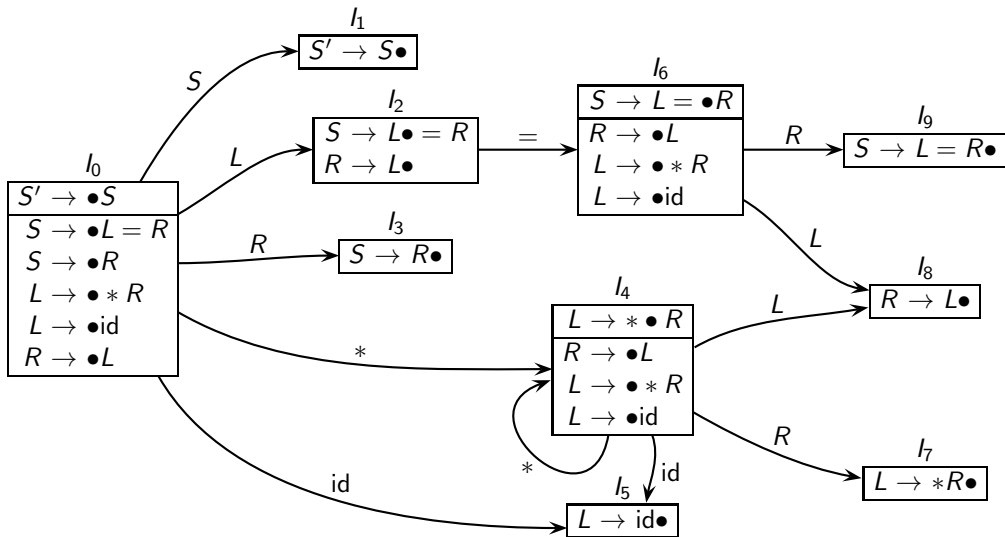
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(0) Item Sets for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

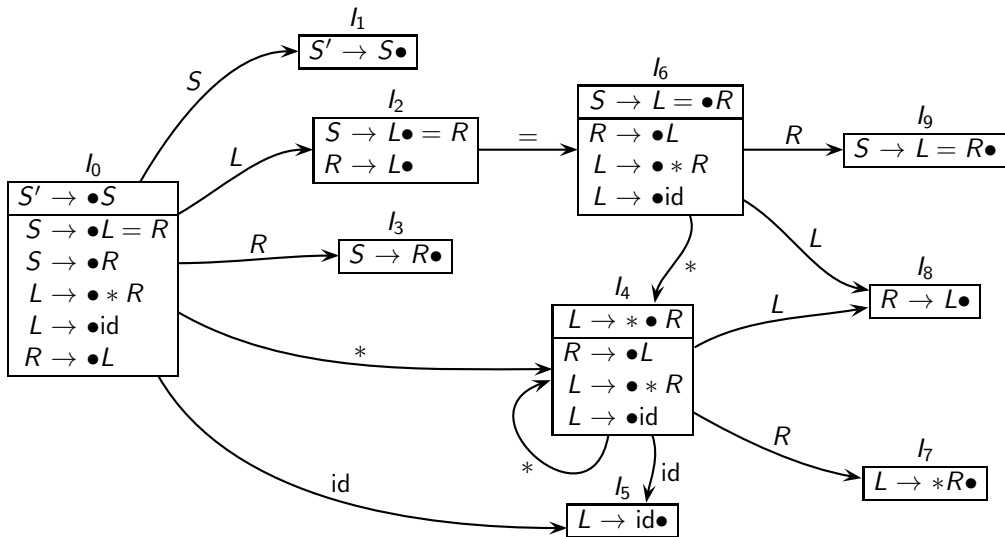
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(0) Item Sets for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

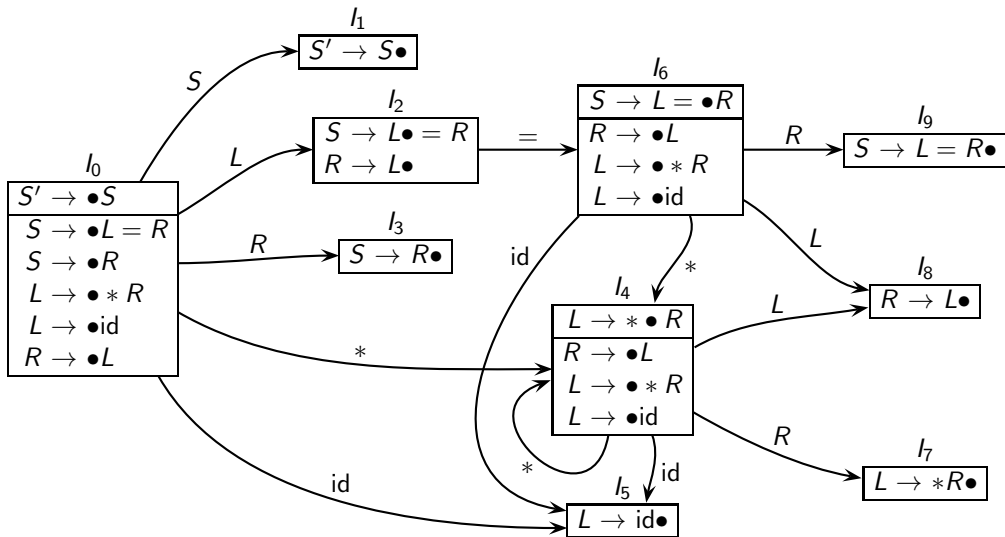
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(0) Item Sets for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

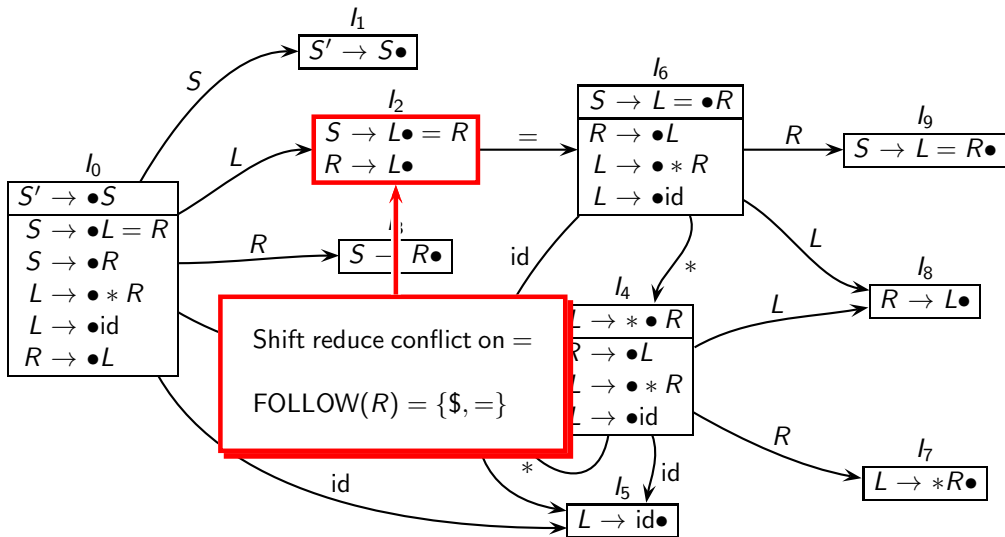
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:
Syntax Analysis

Section:
Grammars,
Derivations, and Parse
Trees

Shift Reduce Parsing

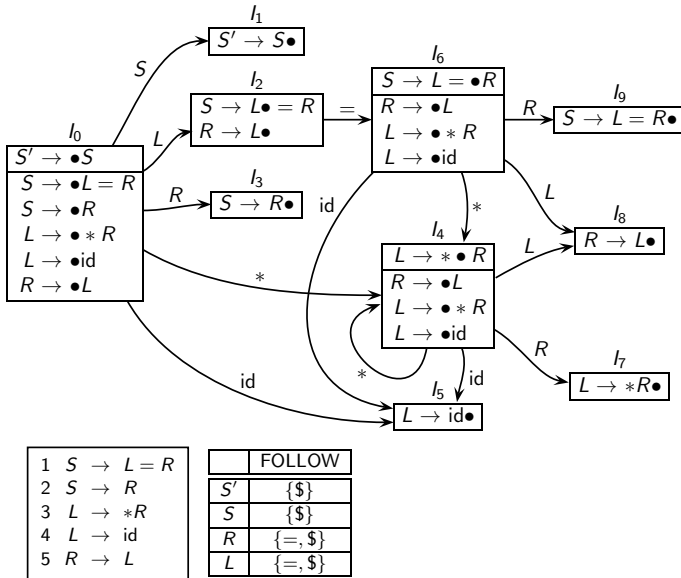
SLR(1) Parsing

Conceptual Issues in
Parsing

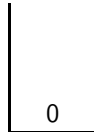
CLR(1) Parsing

LALR(1) Parsing

Limitation of SLR(1) Parsing



Input



Stack



IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:
Syntax Analysis

Section:
Grammars,
Derivations, and Parse
Trees

Shift Reduce Parsing

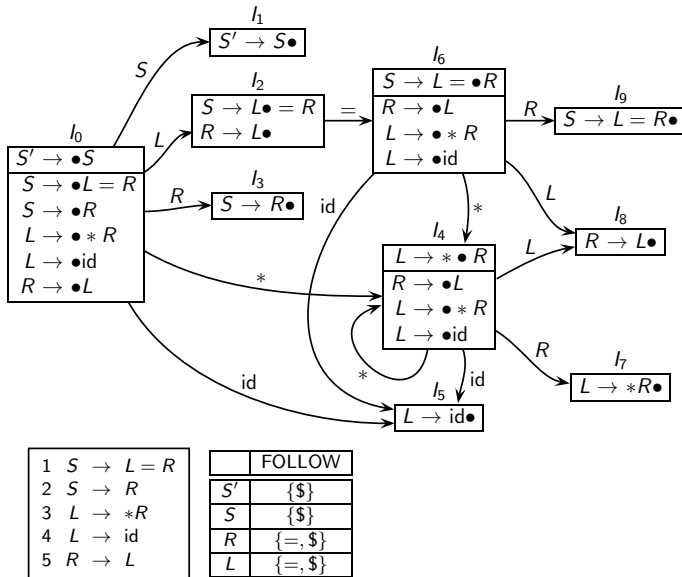
SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing

Limitation of SLR(1) Parsing



Shift 5

Input

id = id\$

0

Stack



IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:
Syntax Analysis

Section:
Grammars,
Derivations, and Parse
Trees

Shift Reduce Parsing

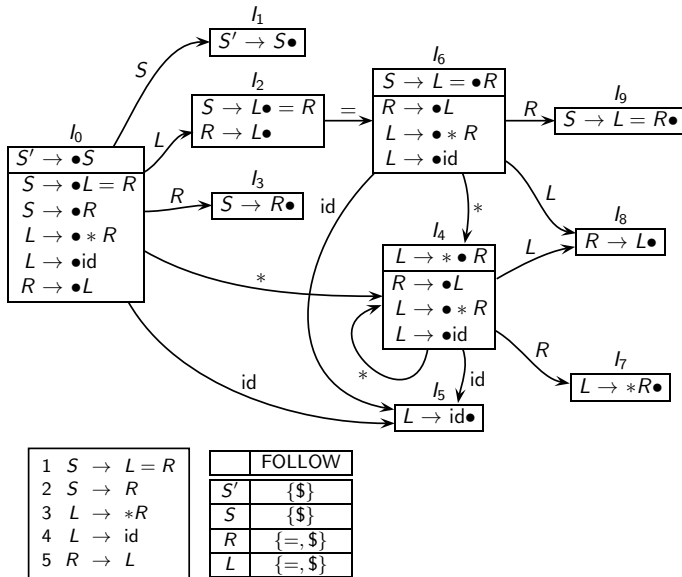
SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing

Limitation of SLR(1) Parsing





IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

Shift Reduce Parsing

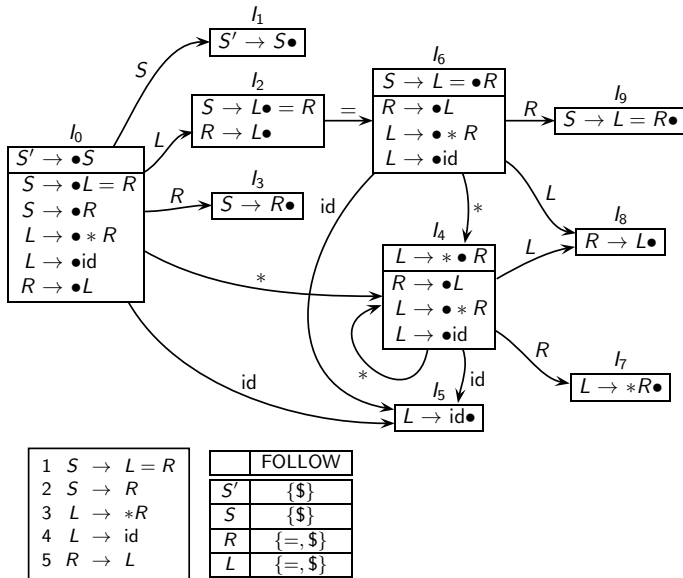
SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing

Limitation of SLR(1) Parsing



Cover by 2

Input

= id\$

L
0

Stack



IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:
Syntax Analysis

Section:
Grammars,
Derivations, and Parse
Trees

Shift Reduce Parsing

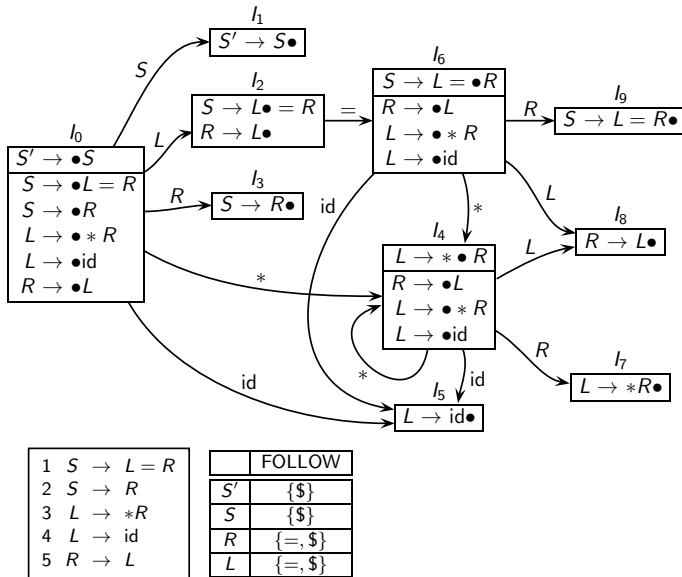
SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing

Limitation of SLR(1) Parsing





Limitation of SLR(1) Parsing

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:
Syntax Analysis

Section:
Grammars,
Derivations, and Parse
Trees

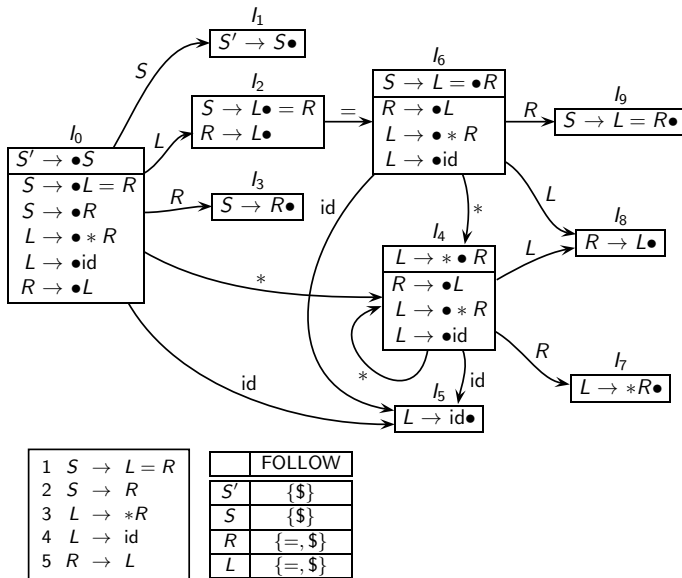
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing



Cover by 3

Input

= id\$

R
0

Stack



IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:
Syntax Analysis

Section:
Grammars,
Derivations, and Parse
Trees

Shift Reduce Parsing

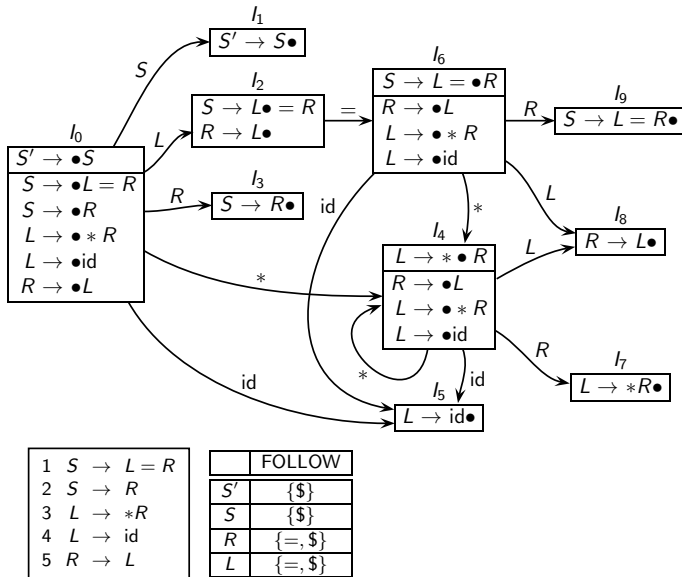
SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing

Limitation of SLR(1) Parsing



Error
No action on =

Input

= id\$

3
R
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: 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



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: 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



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: 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 ' $R =$ '



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 ' $R =$ '
 - Every right sentential form containing ' $R =$ ' begins with a '*' and has a viable prefix '* R '

We will never see '=' after an R without seeing a '*' before the ' R '

$$S \xRightarrow{rm} L = R \xRightarrow{rm} L = L \xRightarrow{rm} L = \text{id} \xRightarrow{rm} \text{id} = \text{id}$$

$$S \xRightarrow{rm} L = R \xRightarrow{rm} L = \text{id} \xRightarrow{rm} *R = \text{id} \xRightarrow{rm} *L = \text{id} \xRightarrow{rm} * \text{id} = \text{id}$$

$$S \xRightarrow{rm} L = R \xRightarrow{rm} L = \text{id} \xRightarrow{rm} *R = \text{id} \xRightarrow{rm} *L = \text{id} \xRightarrow{rm}$$

$$**R = \text{id} \xRightarrow{rm} **L = \text{id} \xRightarrow{rm} ** \text{id} = \text{id}$$

...



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 ' $R =$ '

- Every right sentential form containing ' $R =$ ' begins with a '*' and has a viable prefix '* R '

We will never see '=' after an R without seeing a '*' before the ' R '

$$S \xRightarrow{rm} L = R \xRightarrow{rm} L = L \xRightarrow{rm} L = \text{id} \xRightarrow{rm} \text{id} = \text{id}$$

$$S \xRightarrow{rm} L = R \xRightarrow{rm} L = \text{id} \xRightarrow{rm} *R = \text{id} \xRightarrow{rm} *L = \text{id} \xRightarrow{rm} *\text{id} = \text{id}$$

$$S \xRightarrow{rm} L = R \xRightarrow{rm} L = \text{id} \xRightarrow{rm} *R = \text{id} \xRightarrow{rm} *L = \text{id} \xRightarrow{rm}$$

$$**R = \text{id} \xRightarrow{rm} **L = \text{id} \xRightarrow{rm} **\text{id} = \text{id}$$

...

- '=' is in $\text{FOLLOW}(R)$ only for the RSFs that begin with a '*'
Input ' $\text{id} = \text{id}$ ' does not begin with a '*' so L cannot be reduced to R on '='



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

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



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) 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

- Closure

of the

The goal is to compute different subsets of $\text{FOLLOW}(A)$ 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

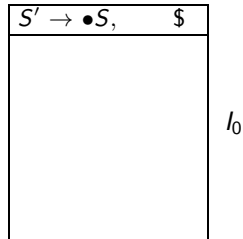
If S is the start symbol, then I_0 contains $S' \rightarrow \bullet S, \$$

- 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





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

$S' \rightarrow \bullet S,$	$\$$
$S \rightarrow \bullet L = R,$	$\$$
$S \rightarrow \bullet R,$	$\$$

I_0



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) 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

$S' \rightarrow \bullet S,$	$\$$
$S \rightarrow \bullet L = R,$	$\$$
$S \rightarrow \bullet R,$	$\$$
$L \rightarrow \bullet * R,$	$=$
$L \rightarrow \bullet \text{id},$	$=$

I_0



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

$S' \rightarrow \bullet S,$	$\$$
$S \rightarrow \bullet L = R,$	$\$$
$S \rightarrow \bullet R,$	$\$$
$L \rightarrow \bullet * R,$	$=$
$L \rightarrow \bullet \text{id},$	$=$
$R \rightarrow \bullet L,$	$\$$
$L \rightarrow \bullet * R,$	$\$$

I_0



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

$S' \rightarrow \bullet S,$	$\$$
$S \rightarrow \bullet L = R,$	$\$$
$S \rightarrow \bullet R,$	$\$$
$L \rightarrow \bullet * R,$	$=$
$L \rightarrow \bullet \text{id},$	$=$
$R \rightarrow \bullet L,$	$\$$
$L \rightarrow \bullet * R,$	$\$$
$L \rightarrow \bullet \text{id},$	$\$$

I_0



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

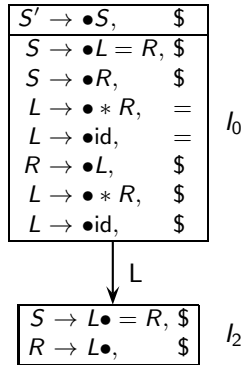
If S is the start symbol, then I_0 contains $S' \rightarrow \bullet S, \$$

- 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





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

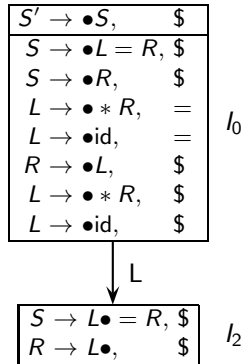
If S is the start symbol, then I_0 contains $S' \rightarrow \bullet S, \$$

- 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



Reduction by $R \rightarrow L \bullet$
only on $\$$ and not on $=$
No shift reduce conflict

LR(1) Sets of Items for Pointer Assignment Grammar



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

I_0

$S' \rightarrow \bullet S,$	$\$$
$S \rightarrow \bullet L = R,$	$\$$
$S \rightarrow \bullet R,$	$\$$
$L \rightarrow \bullet * R,$	$= / \$$
$L \rightarrow \bullet id,$	$= / \$$
$R \rightarrow \bullet L,$	$\$$



LR(1) Sets of Items for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

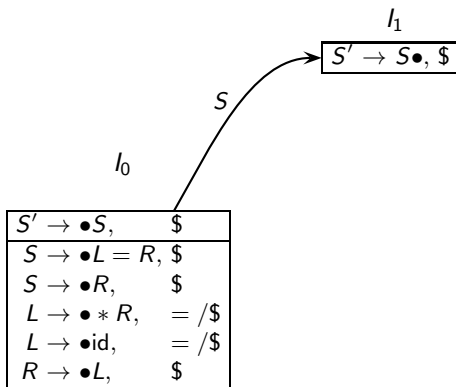
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(1) Sets of Items for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

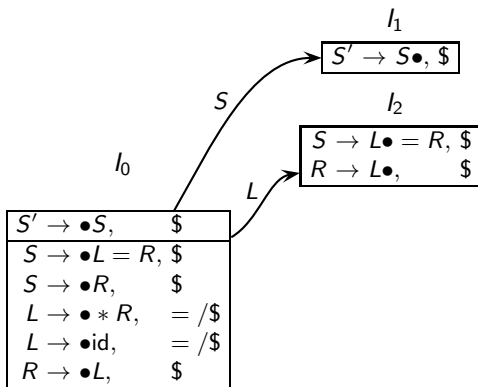
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(1) Sets of Items for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

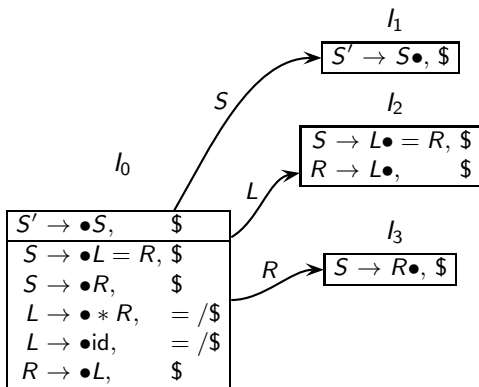
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(1) Sets of Items for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

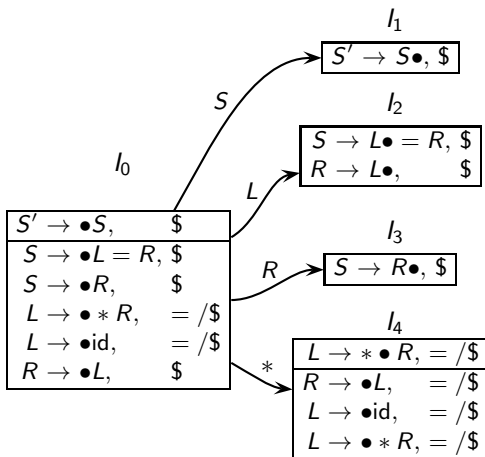
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(1) Sets of Items for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

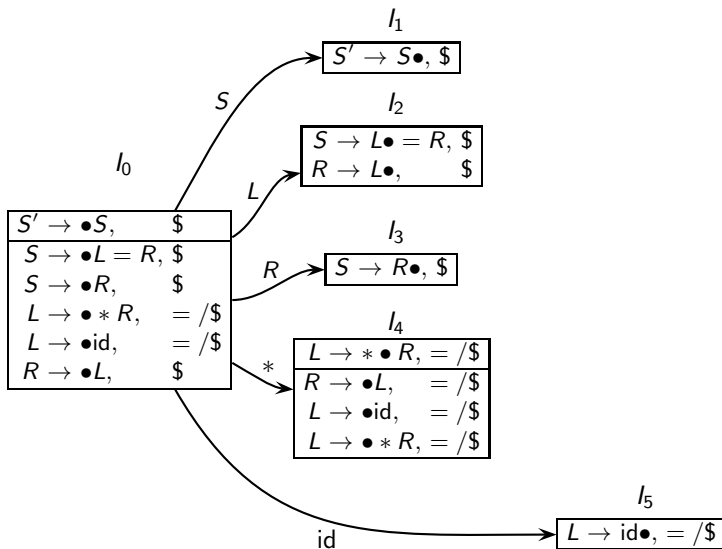
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(1) Sets of Items for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

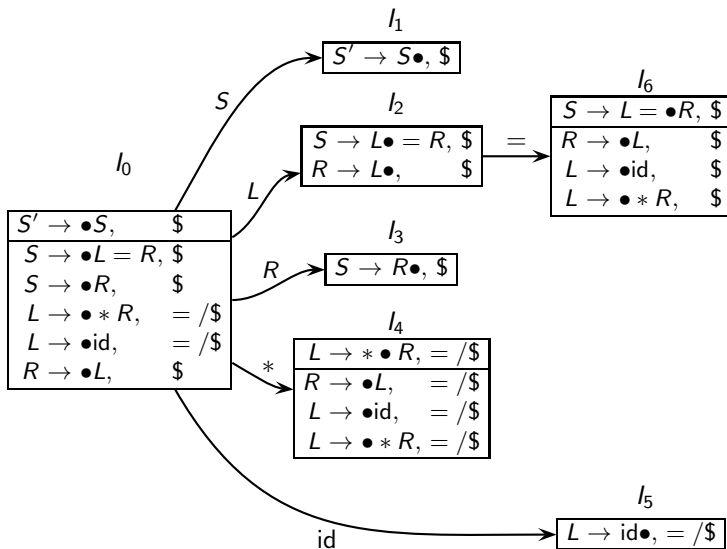
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(1) Sets of Items for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

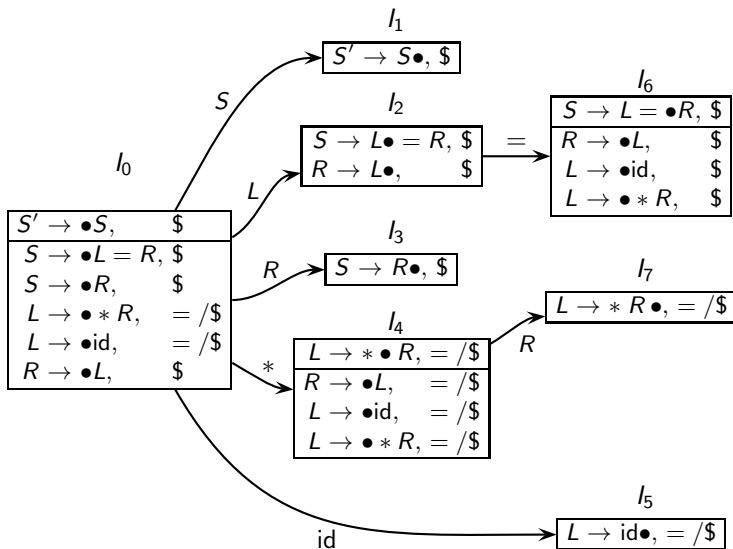
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(1) Sets of Items for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

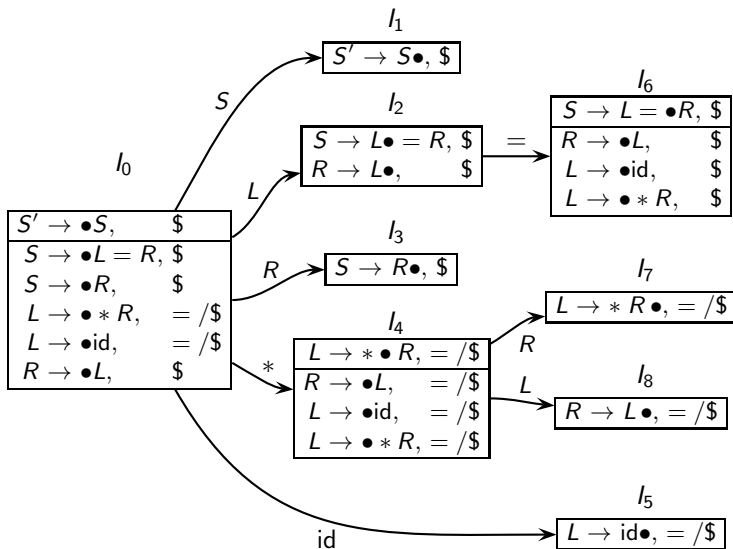
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(1) Sets of Items for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

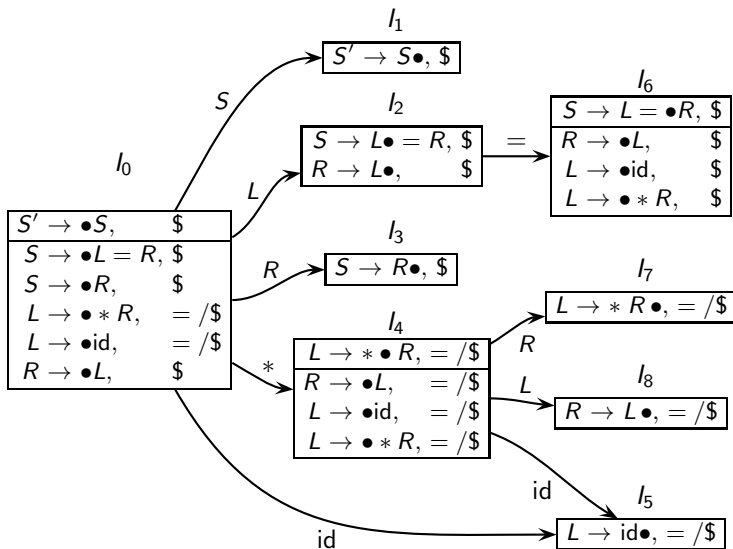
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(1) Sets of Items for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

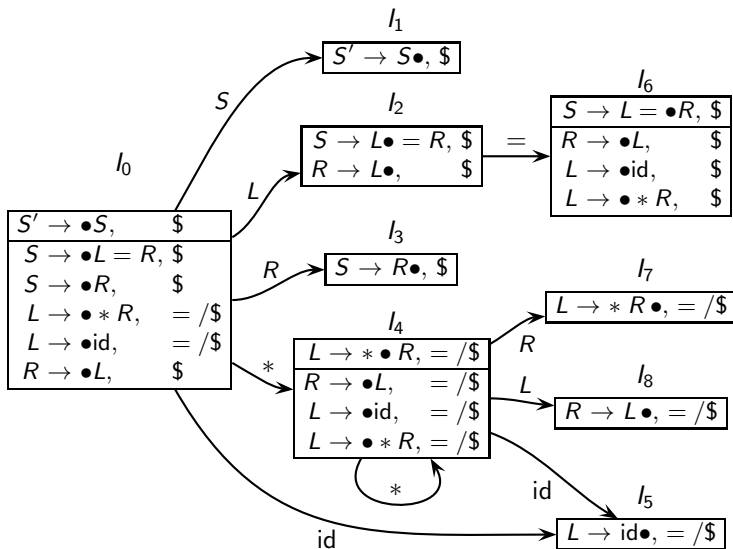
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(1) Sets of Items for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

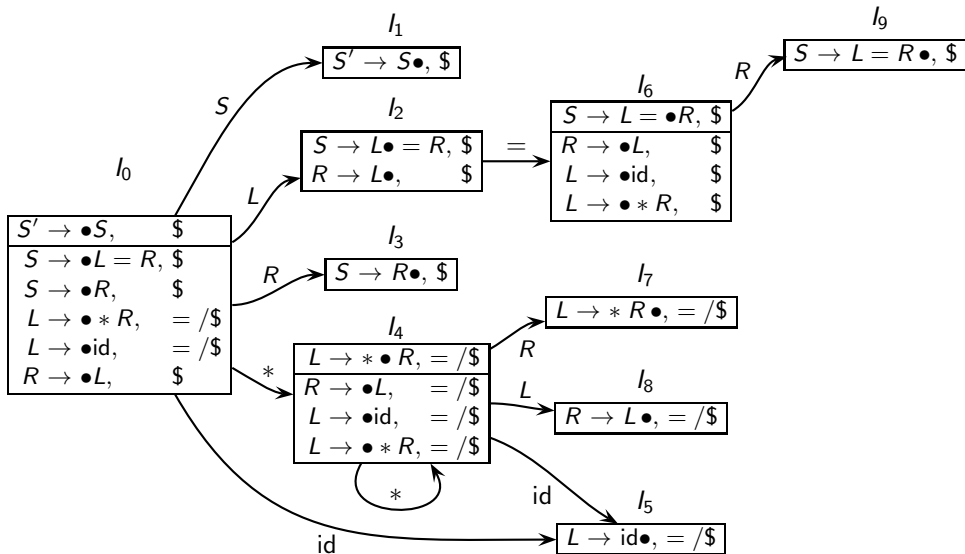
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(1) Sets of Items for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

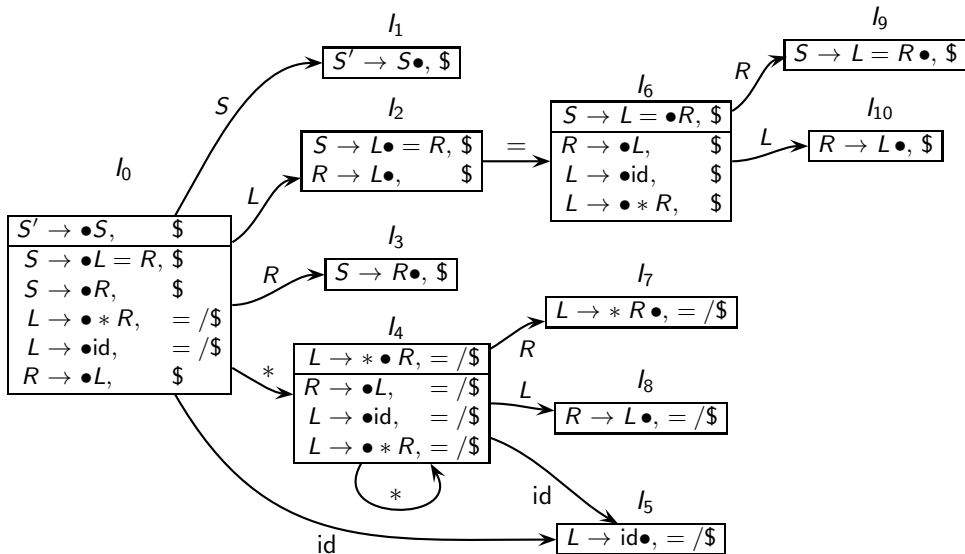
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(1) Sets of Items for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

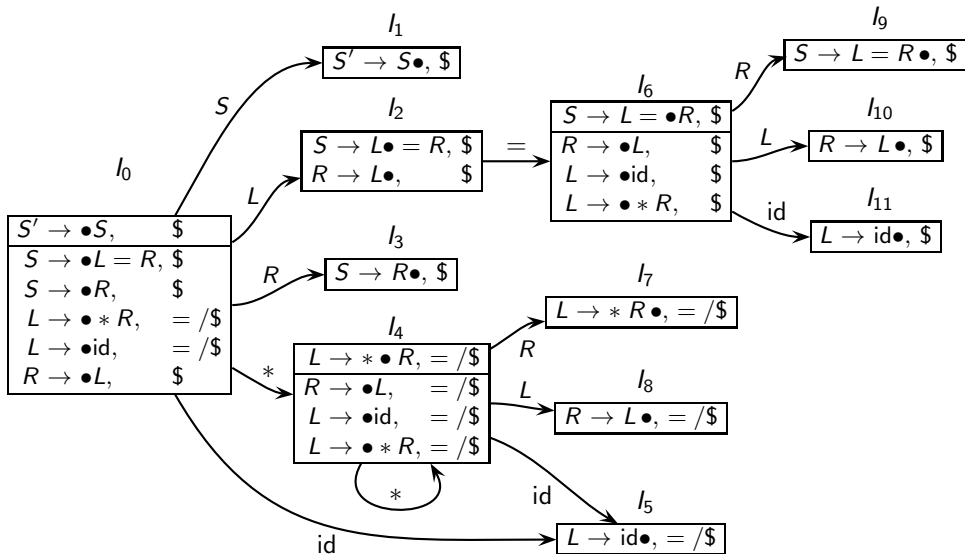
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(1) Sets of Items for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

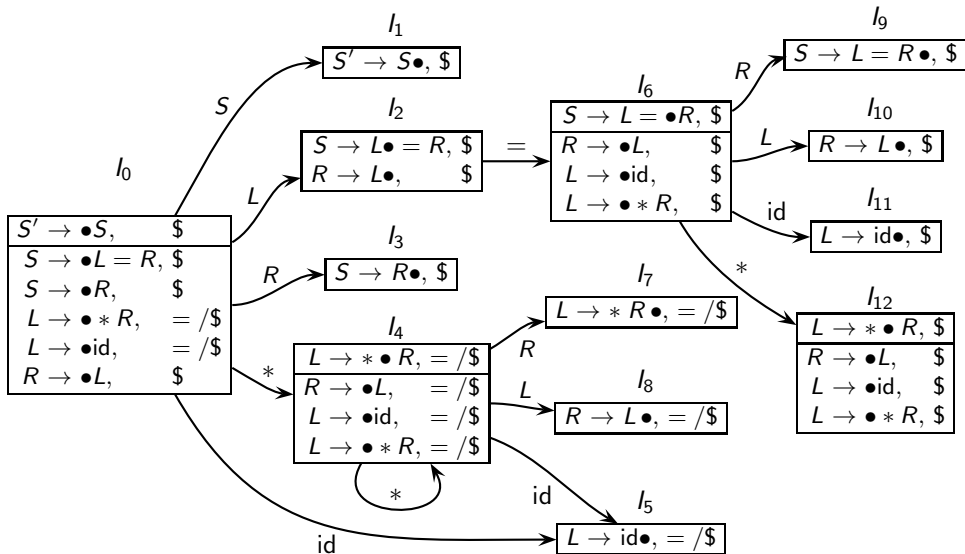
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(1) Sets of Items for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

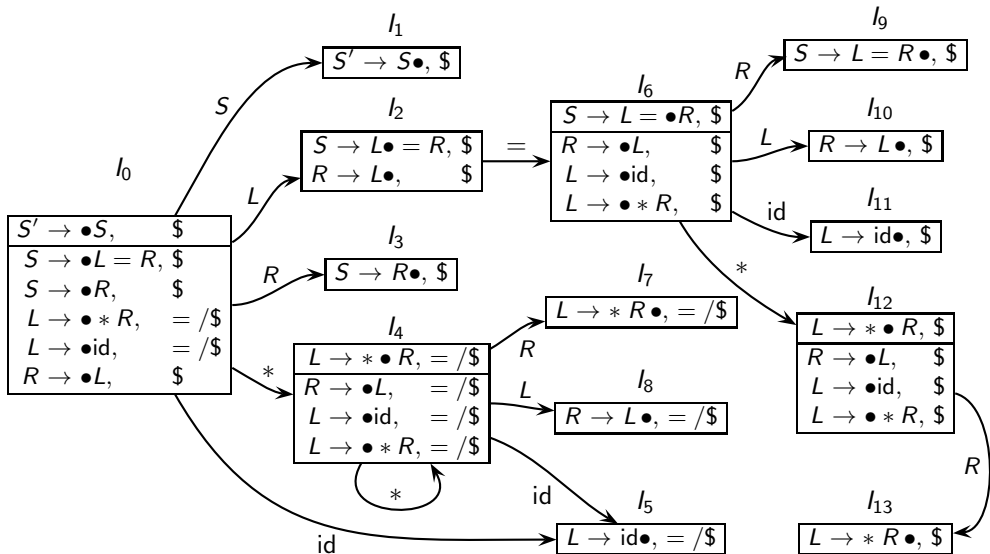
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(1) Sets of Items for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

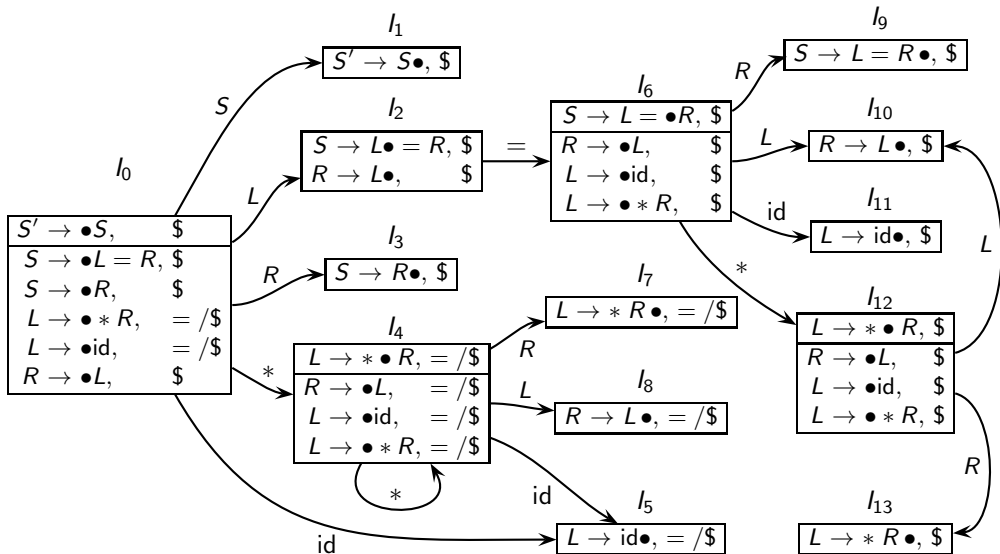
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(1) Sets of Items for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

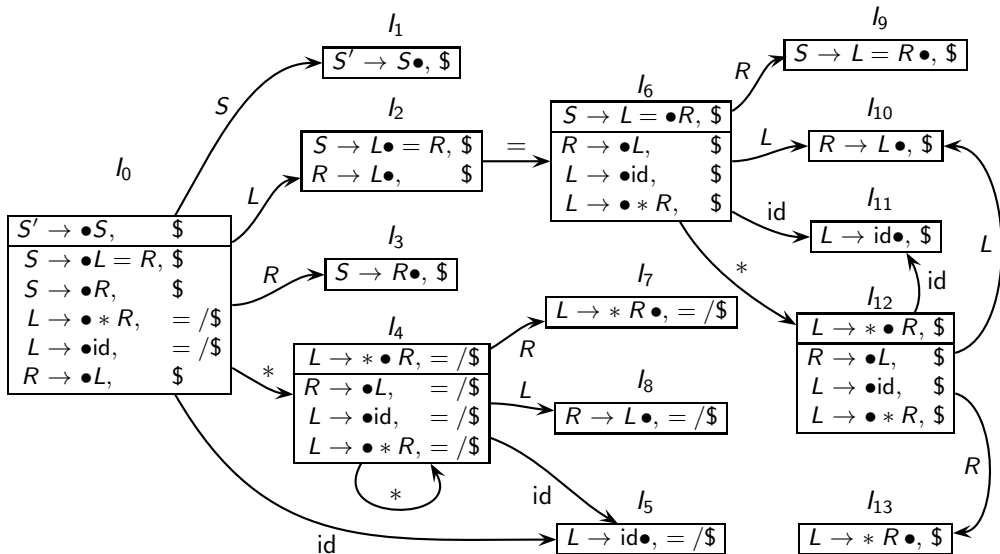
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(1) Sets of Items for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

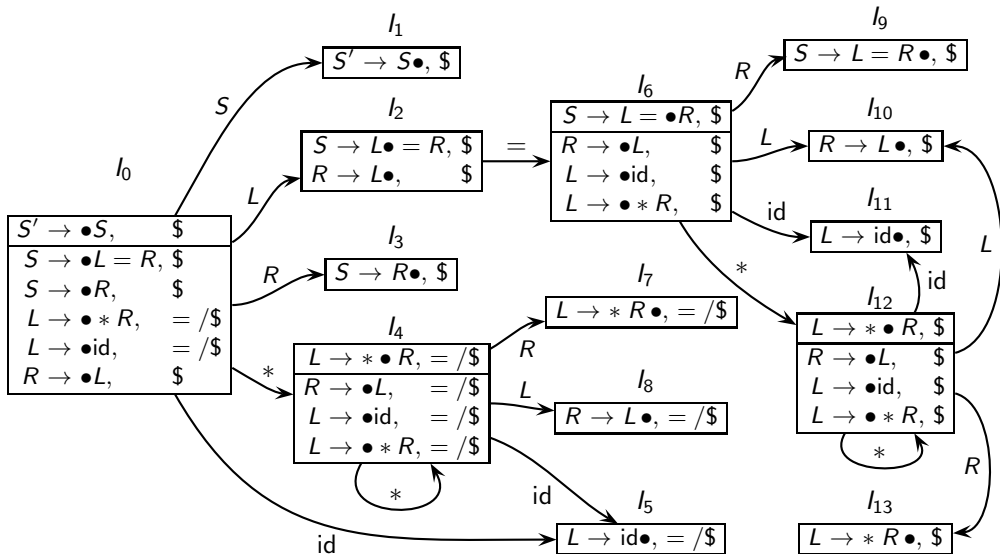
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LR(1) (aka CLR(1)) Parsing Table for Pointer Assignment Grammar

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

- 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

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

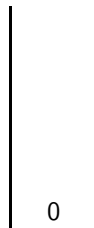
CLR(1) Parsing

LALR(1) Parsing

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



Stack



LR(1) (aka CLR(1)) Parsing for the Pointer Assignment Grammar

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

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

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

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 $L \rightarrow id$
- 5 $R \rightarrow L$

Input

= id\$

Reduce by 4

5
id
0

Stack



LR(1) (aka CLR(1)) Parsing for the Pointer Assignment Grammar

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

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\$

Cover by 2

L
0

Stack



LR(1) (aka CLR(1)) Parsing for the Pointer Assignment Grammar

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

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 6

2
L
0

Stack



LR(1) (aka CLR(1)) Parsing for the Pointer Assignment Grammar

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

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

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

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 4

11
id
6
=
2
L
0

Stack



LR(1) (aka CLR(1)) Parsing for the Pointer Assignment Grammar

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

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

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

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 5

10
L
6
=
2
L
0

Stack



LR(1) (aka CLR(1)) Parsing for the Pointer Assignment Grammar

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

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

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

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

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

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

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

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



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

$A \rightarrow aBe$

$A \rightarrow aCd$

$A \rightarrow bBd$

$A \rightarrow bCe$

$B \rightarrow f$

$C \rightarrow f$

Another Example of LR(1) (aka CLR(1)) Parsing



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

$A \rightarrow aBe$

$A \rightarrow aCd$

$A \rightarrow bBd$

$A \rightarrow bCe$

$B \rightarrow f$

$C \rightarrow f$

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

$A \rightarrow aBe$

$A \rightarrow aCd$

$A \rightarrow bBd$

$A \rightarrow bCe$

$B \rightarrow f$

$C \rightarrow f$

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

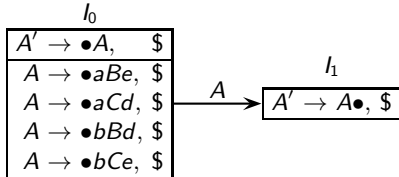
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





Another Example of LR(1) (aka CLR(1)) Parsing

$A \rightarrow aBe$

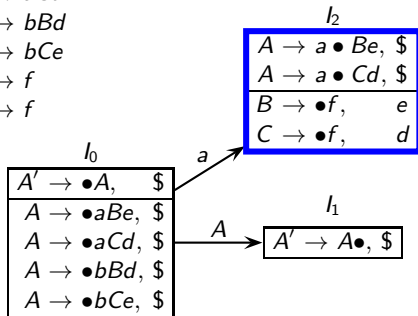
$A \rightarrow aCd$

$A \rightarrow bBd$

$A \rightarrow bCe$

$B \rightarrow f$

$C \rightarrow f$



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 e and p is $\$$
If we expect to see a string derivable from B in this state, the string must be followed by
 $\text{FIRST}(\beta p) = \text{FIRST}(e\$) = e$
- 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
 $\text{FIRST}(\beta p) = \text{FIRST}(d\$) = d$



Another Example of LR(1) (aka CLR(1)) Parsing

$A \rightarrow aBe$

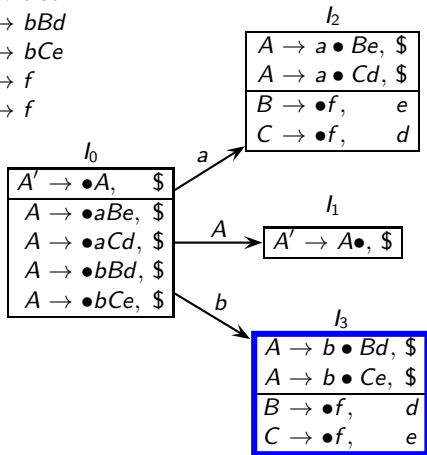
$A \rightarrow aCd$

$A \rightarrow bBd$

$A \rightarrow bCe$

$B \rightarrow f$

$C \rightarrow f$



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

$A \rightarrow aBe$

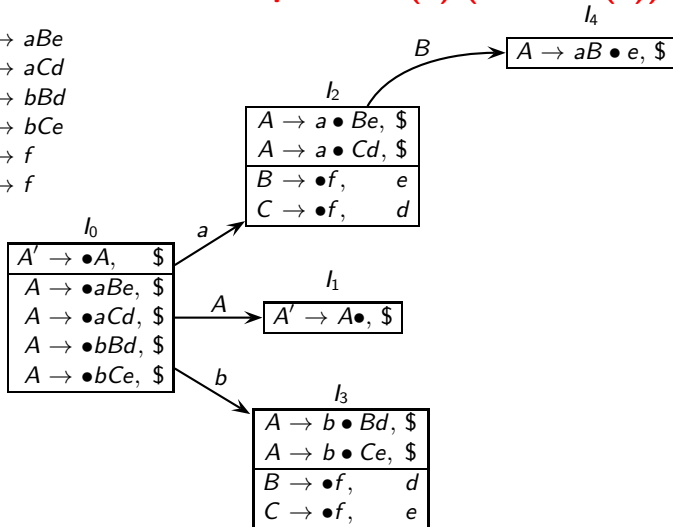
$A \rightarrow aCd$

$A \rightarrow bBd$

$A \rightarrow bCe$

$B \rightarrow f$

$C \rightarrow f$



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



Another Example of LR(1) (aka CLR(1)) Parsing

$A \rightarrow aBe$

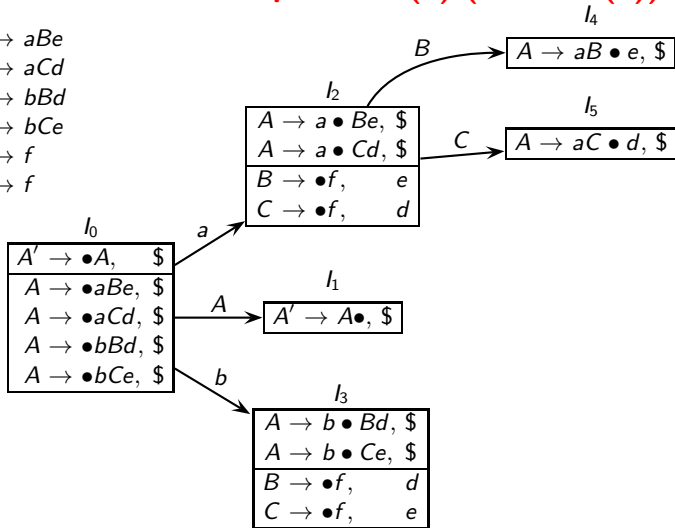
$A \rightarrow aCd$

$A \rightarrow bBd$

$A \rightarrow bCe$

$B \rightarrow f$

$C \rightarrow f$



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



Another Example of LR(1) (aka CLR(1)) Parsing

$A \rightarrow aBe$

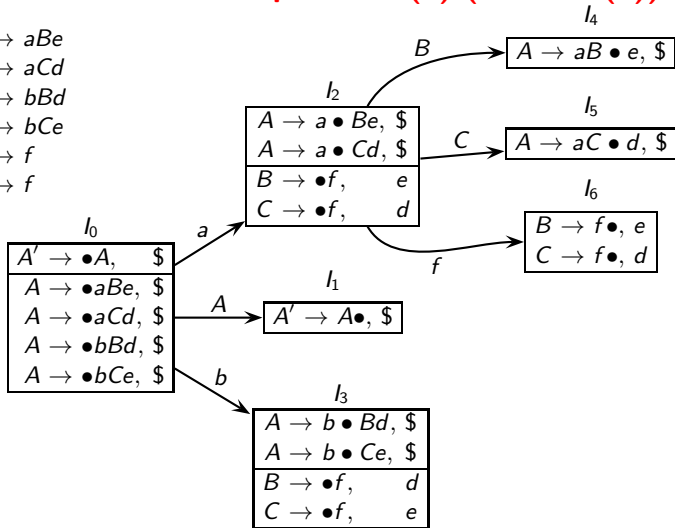
$A \rightarrow aCd$

$A \rightarrow bBd$

$A \rightarrow bCe$

$B \rightarrow f$

$C \rightarrow f$



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



Another Example of LR(1) (aka CLR(1)) Parsing

$A \rightarrow aBe$

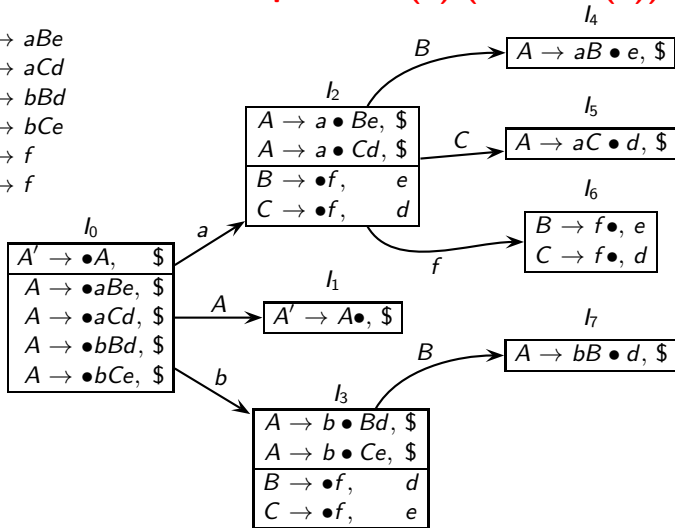
$A \rightarrow aCd$

$A \rightarrow bBd$

$A \rightarrow bCe$

$B \rightarrow f$

$C \rightarrow f$



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



Another Example of LR(1) (aka CLR(1)) Parsing

$A \rightarrow aBe$

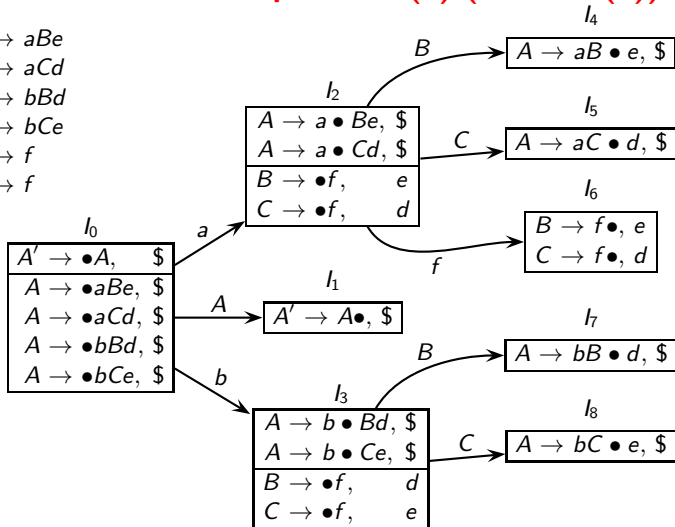
$A \rightarrow aCd$

$A \rightarrow bBd$

$A \rightarrow bCe$

$B \rightarrow f$

$C \rightarrow f$



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



Another Example of LR(1) (aka CLR(1)) Parsing

$A \rightarrow aBe$

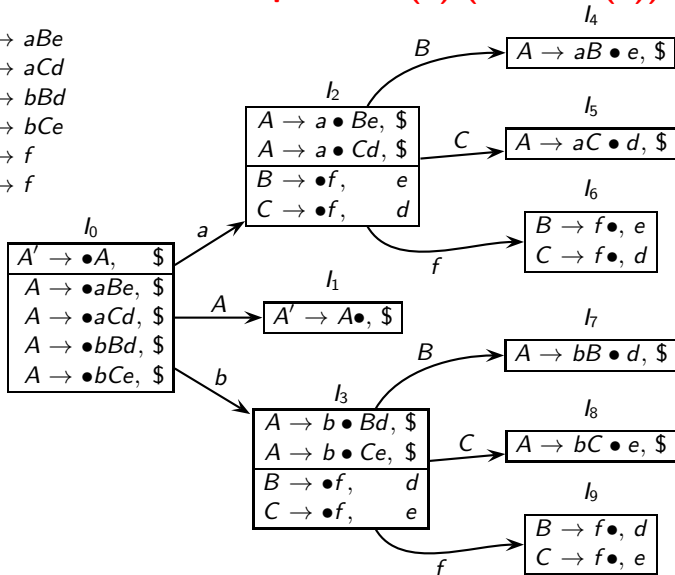
$A \rightarrow aCd$

$A \rightarrow bBd$

$A \rightarrow bCe$

$B \rightarrow f$

$C \rightarrow f$



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



Another Example of LR(1) (aka CLR(1)) Parsing

$A \rightarrow aBe$

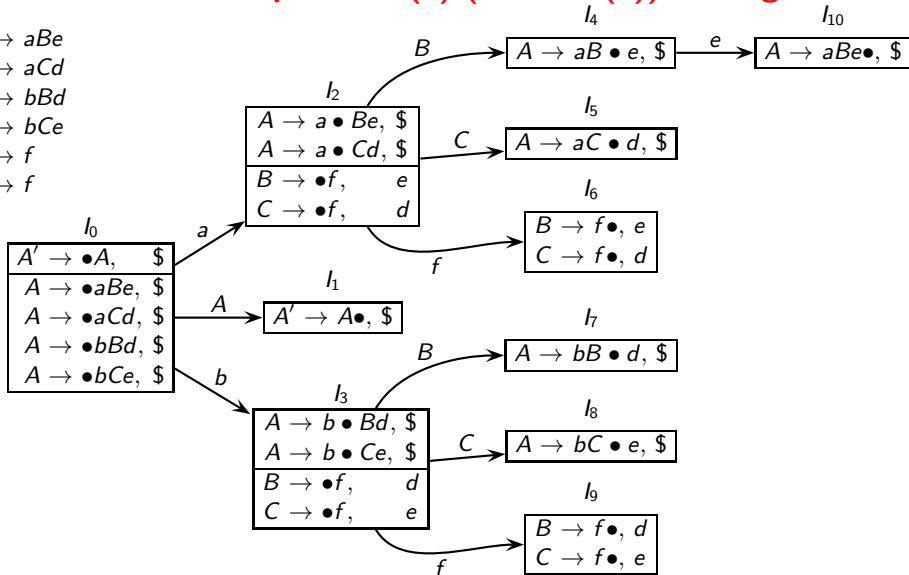
$A \rightarrow aCd$

$A \rightarrow bBd$

$A \rightarrow bCe$

$B \rightarrow f$

$C \rightarrow f$



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



Another Example of LR(1) (aka CLR(1)) Parsing

$A \rightarrow aBe$

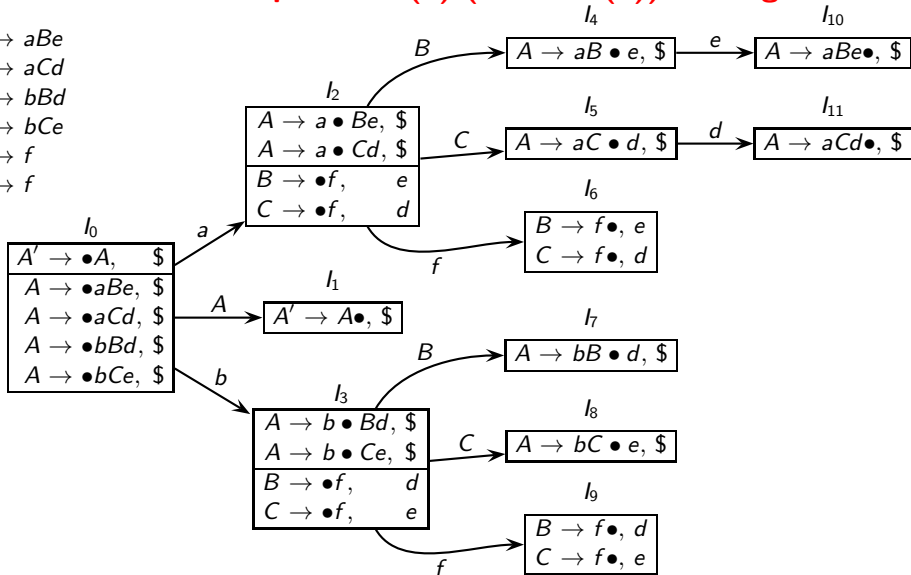
$A \rightarrow aCd$

$A \rightarrow bBd$

$A \rightarrow bCe$

$B \rightarrow f$

$C \rightarrow f$



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



Another Example of LR(1) (aka CLR(1)) Parsing

$A \rightarrow aBe$

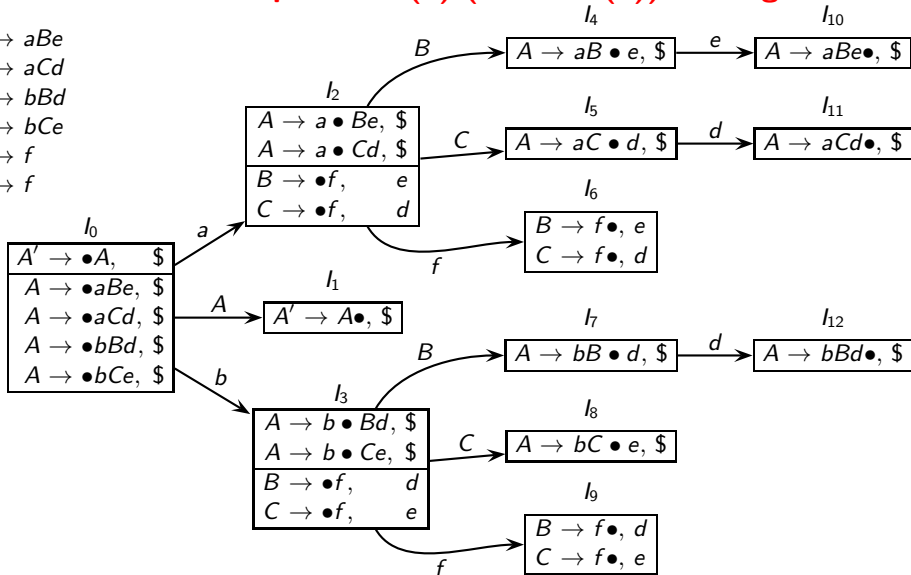
$A \rightarrow aCd$

$A \rightarrow bBd$

$A \rightarrow bCe$

$B \rightarrow f$

$C \rightarrow f$



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



Another Example of LR(1) (aka CLR(1)) Parsing

$A \rightarrow aBe$

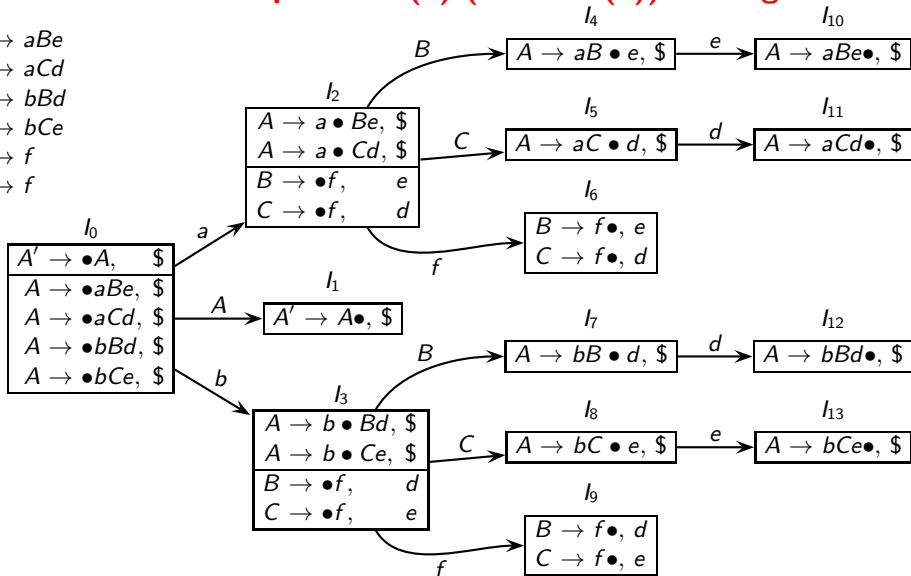
$A \rightarrow aCd$

$A \rightarrow bBd$

$A \rightarrow bCe$

$B \rightarrow f$

$C \rightarrow f$



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



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



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

LALR(1) Parsing



LALR(1) Parsing

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

- 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

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

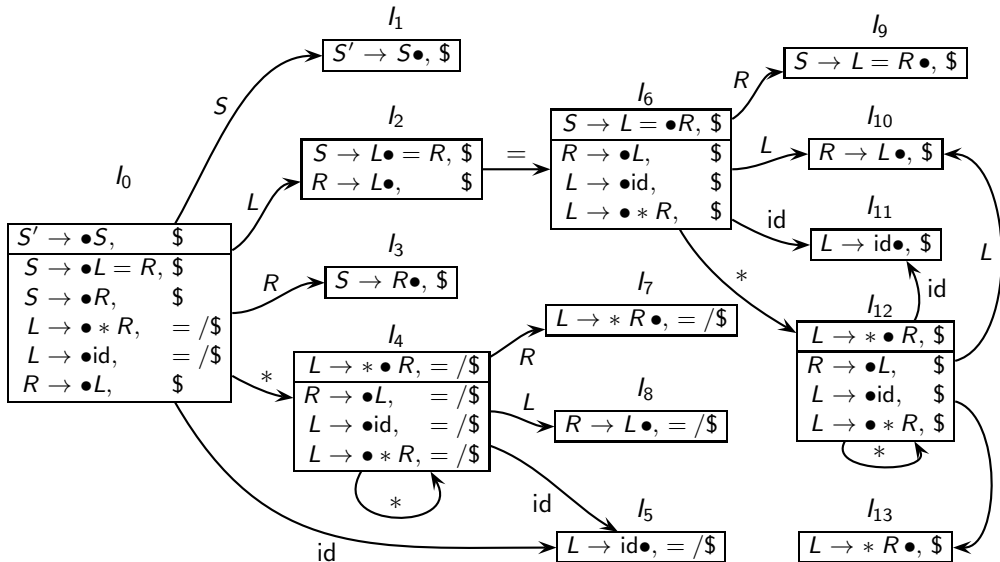
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LALR(1) Parsing for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:
Syntax Analysis

Section:
Grammars,
Derivations, and Parse
Trees

Shift Reduce Parsing

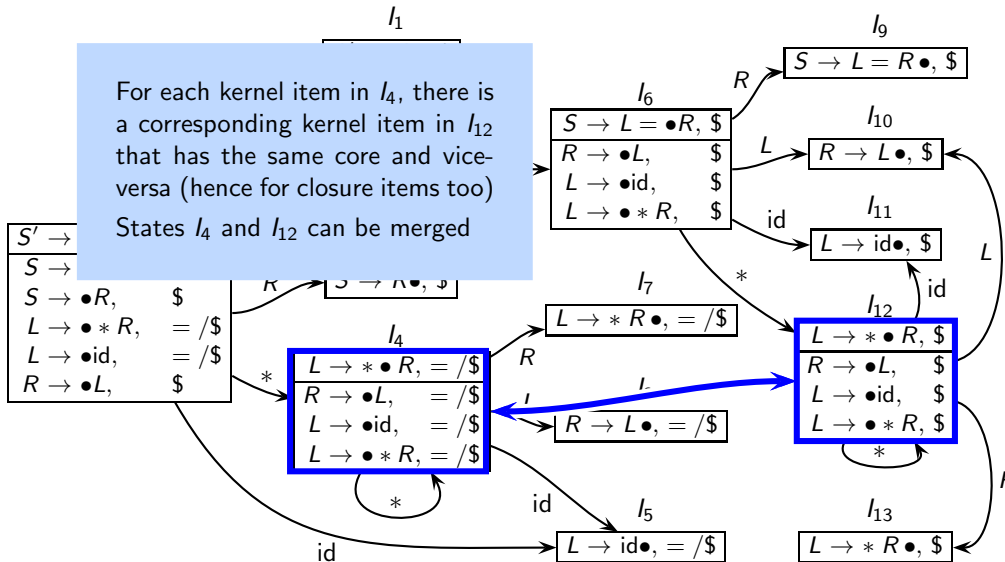
SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing

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

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:
Syntax Analysis

Section:
Grammars,
Derivations, and Parse
Trees

Shift Reduce Parsing

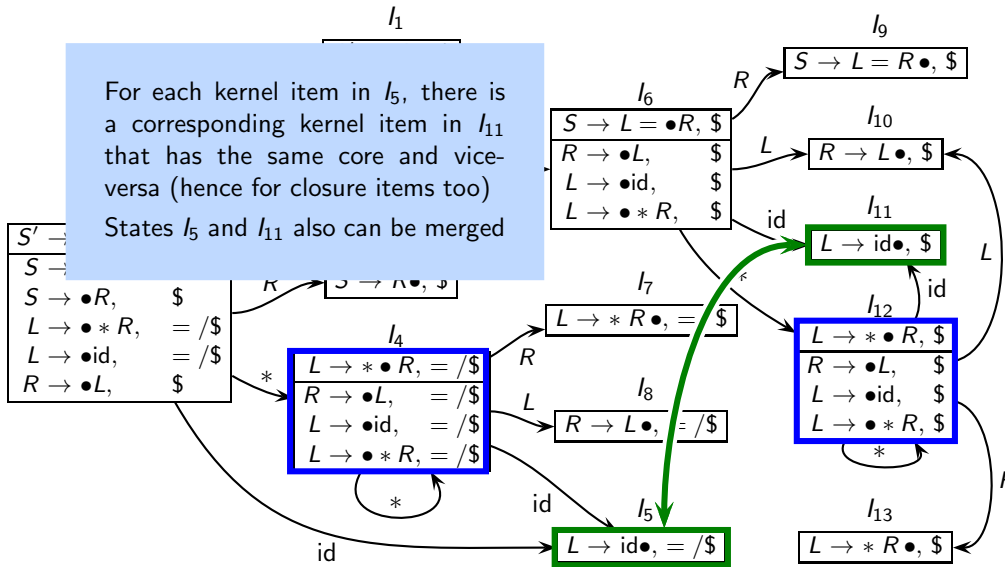
SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing

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

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:
Syntax Analysis

Section:
Grammars,
Derivations, and Parse
Trees

Shift Reduce Parsing

SLR(1) Parsing

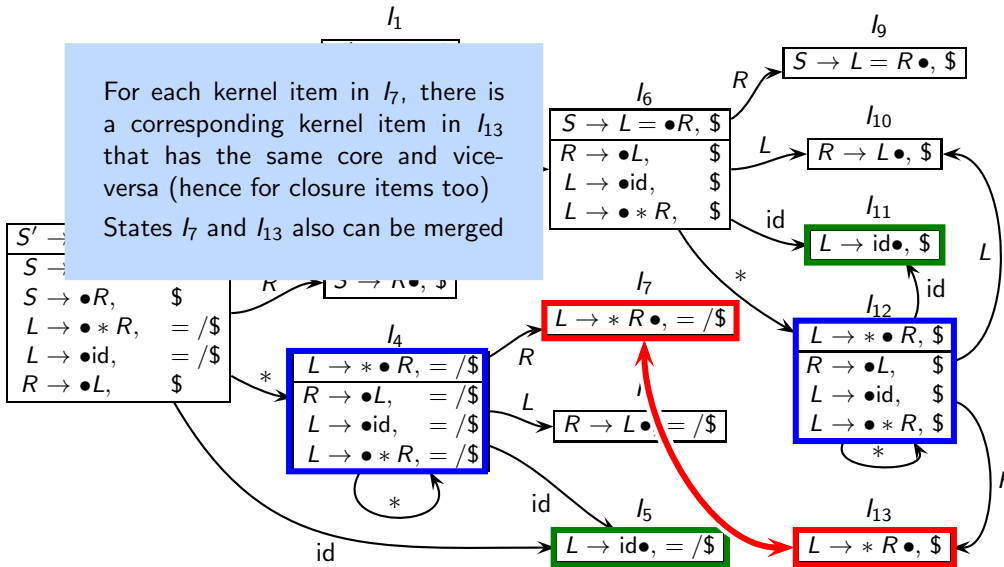
Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing

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





LALR(1) Parsing for Pointer Assignment Grammar

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:
Syntax Analysis

Section:
Grammars,
Derivations, and Parse
Trees

Shift Reduce Parsing

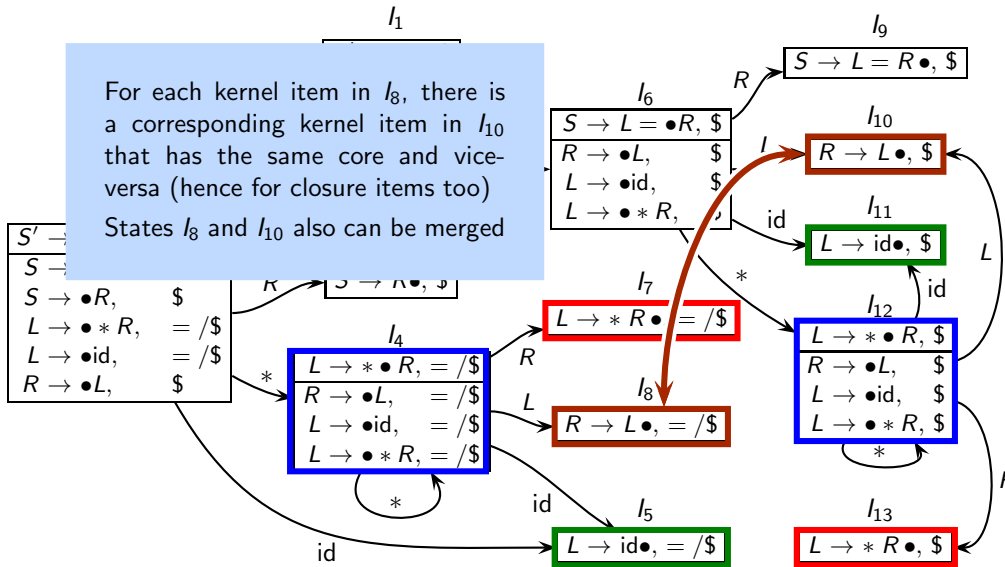
SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing

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

IIT Bombay
cs302: Implementation
of Programming
Languages

Topic:

Syntax Analysis

Section:

Grammars,
Derivations, and Parse
Trees

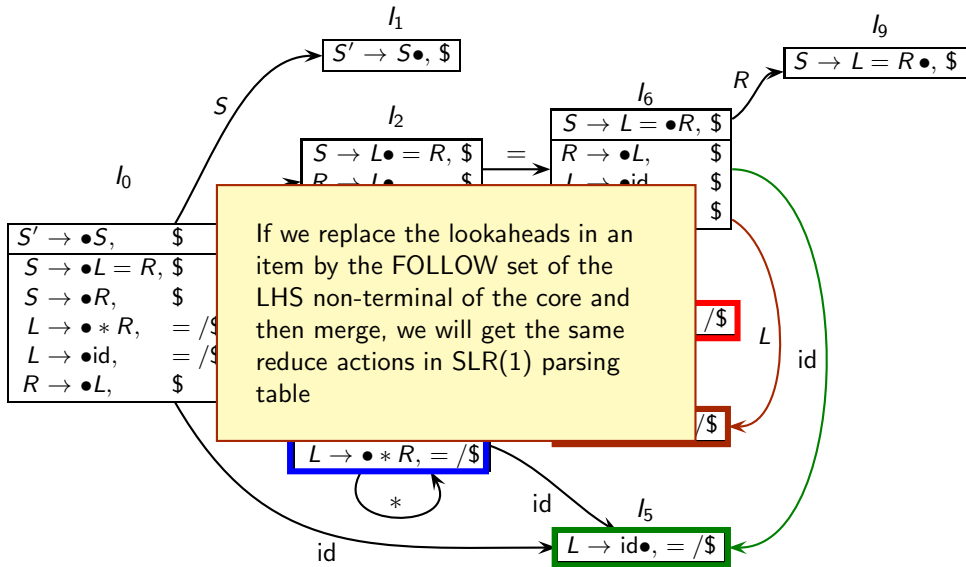
Shift Reduce Parsing

SLR(1) Parsing

Conceptual Issues in
Parsing

CLR(1) Parsing

LALR(1) Parsing





LALR(1) Parsing Table for Pointer Assignment Grammar

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

- 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



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

- Can merging of LR(1) states introduce shift-reduce conflict?
- Can merging of LR(1) states introduce reduce-reduce conflict?



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

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)



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

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 $I_i :$
$$\begin{array}{l} A \rightarrow \alpha \bullet a\beta, \quad p \\ B \rightarrow \gamma \bullet, \quad q \end{array}$$
 and $I_j :$
$$\begin{array}{l} A \rightarrow \alpha \bullet a\beta, \quad r \\ B \rightarrow \gamma \bullet, \quad s \end{array}$$
 where p, q, r, s are arbitrary terminals

So that the merged state is $I_{ij} :$
$$\begin{array}{l} A \rightarrow \alpha \bullet a\beta, \quad p/r \\ B \rightarrow \gamma \bullet, \quad q/s \end{array}$$



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

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 $I_i : \begin{array}{l} A \rightarrow \alpha \bullet a\beta, \quad p \\ B \rightarrow \gamma \bullet, \quad q \end{array}$ and $I_j : \begin{array}{l} A \rightarrow \alpha \bullet a\beta, \quad r \\ B \rightarrow \gamma \bullet, \quad s \end{array}$ where p, q, r, s are arbitrary terminals

So that the merged state is $I_{ij} : \begin{array}{l} A \rightarrow \alpha \bullet a\beta, \quad p/r \\ B \rightarrow \gamma \bullet, \quad q/s \end{array}$

- For a shift-reduce conflict in I_{ij} , either q or s must be a .



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

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 $I_i : \begin{array}{l} A \rightarrow \alpha \bullet a\beta, \quad p \\ B \rightarrow \gamma \bullet, \quad q \end{array}$ and $I_j : \begin{array}{l} A \rightarrow \alpha \bullet a\beta, \quad r \\ B \rightarrow \gamma \bullet, \quad s \end{array}$ where p, q, r, s are arbitrary terminals

So that the merged state is $I_{ij} : \begin{array}{l} A \rightarrow \alpha \bullet a\beta, \quad p/r \\ B \rightarrow \gamma \bullet, \quad q/s \end{array}$

- For a shift-reduce conflict in I_{ij} , either q or s must be a .
 - If q is a , then I_i is $\begin{array}{l} A \rightarrow \alpha \bullet a\beta, \quad p \\ B \rightarrow \gamma \bullet, \quad a \end{array}$ and thus I_i has a shift-reduce conflict



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

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 $I_i : \begin{array}{l} A \rightarrow \alpha \bullet a\beta, \quad p \\ B \rightarrow \gamma \bullet, \quad q \end{array}$ and $I_j : \begin{array}{l} A \rightarrow \alpha \bullet a\beta, \quad r \\ B \rightarrow \gamma \bullet, \quad s \end{array}$ where p, q, r, s are arbitrary terminals

So that the merged state is $I_{ij} : \begin{array}{l} A \rightarrow \alpha \bullet a\beta, \quad p/r \\ B \rightarrow \gamma \bullet, \quad q/s \end{array}$

- For a shift-reduce conflict in I_{ij} , either q or s must be a .
 - If q is a , then I_i is $\begin{array}{l} A \rightarrow \alpha \bullet a\beta, \quad p \\ B \rightarrow \gamma \bullet, \quad a \end{array}$ and thus I_i has a shift-reduce conflict
 - If s is a , then I_j is $\begin{array}{l} A \rightarrow \alpha \bullet a\beta, \quad r \\ B \rightarrow \gamma \bullet, \quad a \end{array}$ 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 : $A \rightarrow \alpha \bullet a \beta$, r and I_j : $B \rightarrow \gamma \bullet$, a be arbitrary LR(1) items, r, s are terminals.

So that the

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

- For a shift-reduce conflict to exist in I_{ij}
 - If q is r , then I_i has a shift-reduce conflict
 - If s is a , then I_j is $B \rightarrow \gamma \bullet$, a and thus I_j has a shift-reduce conflict



Can Merging LR(1) Sets of Items Introduce Reduce-Reduce Conflict?

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

- Let $I_i :$
$$\begin{array}{l} A \rightarrow \alpha \bullet, \quad p \\ B \rightarrow \alpha \bullet, \quad q \end{array}$$
 and $I_j :$
$$\begin{array}{l} A \rightarrow \alpha \bullet, \quad r \\ B \rightarrow \alpha \bullet, \quad s \end{array}$$

So that the merged state is $I_{ij} :$
$$\begin{array}{l} A \rightarrow \alpha \bullet, \quad p/r \\ B \rightarrow \alpha \bullet, \quad q/s \end{array}$$



Can Merging LR(1) Sets of Items Introduce Reduce-Reduce Conflict?

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

- Let $I_i :$
$$\begin{array}{l} A \rightarrow \alpha \bullet, \quad p \\ B \rightarrow \alpha \bullet, \quad q \end{array}$$
 and $I_j :$
$$\begin{array}{l} A \rightarrow \alpha \bullet, \quad r \\ B \rightarrow \alpha \bullet, \quad s \end{array}$$

So that the merged state is $I_{ij} :$
$$\begin{array}{l} A \rightarrow \alpha \bullet, \quad p/r \\ B \rightarrow \alpha \bullet, \quad q/s \end{array}$$

- 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?

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

- Let $I_i :$

A
 B

So that the

- For a reduce
or I_j ,

- $p = s.$
- $r = q.$

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

ce conflict in I_i

and I_j

This is also possible without a reduce-reduce conflict in I_i and I_j



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

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



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

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, $\mathbb{G}(LALR(1)) \subset \mathbb{G}(LR(1))$



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

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?



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

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 $L(G)$



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

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 $L(G)$

- 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 $L(G) = L(G')$



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

Example of Reduce-Reduce Conflict Caused by Merging LR(1) Sets of Items

$A \rightarrow aBe$

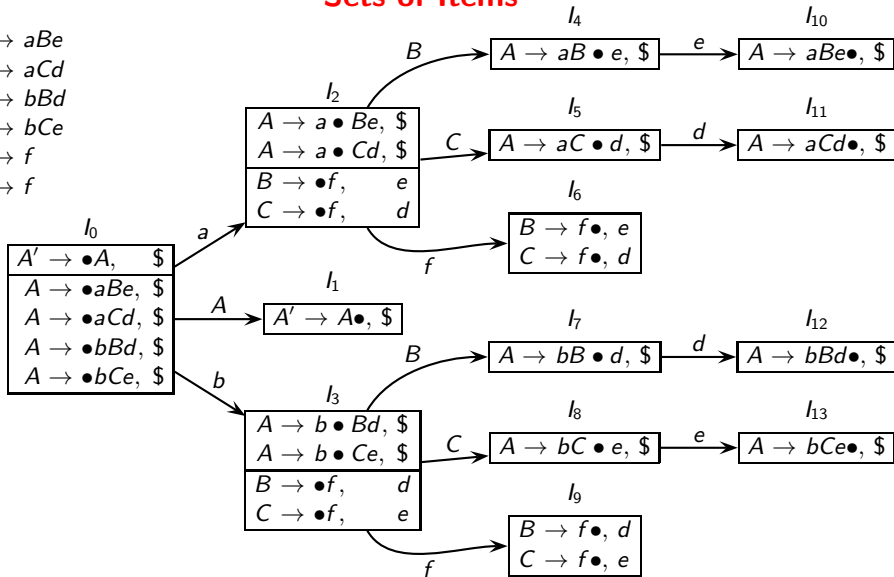
$A \rightarrow aCd$

$A \rightarrow bBd$

$A \rightarrow bCe$

$B \rightarrow f$

$C \rightarrow f$





Example of Reduce-Reduce Conflict Caused by Merging LR(1) Sets of Items

$A \rightarrow aBe$

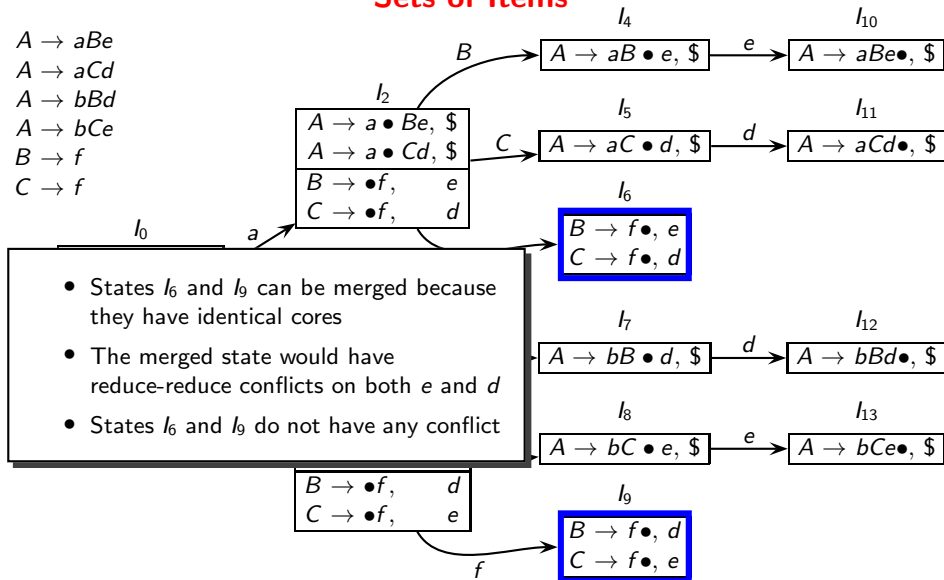
$A \rightarrow aCd$

$A \rightarrow bBd$

$A \rightarrow bCe$

$B \rightarrow f$

$C \rightarrow f$





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

A Practical Example of Reduce-Reduce Conflict in LR(1) Parsing

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

program → func_decl var_decl
program → var_decl func_decl
var_decl → data_type ID ;
data_type → INT
func_decl → return_type ID ()
return_type → INT
return_type → VOID

State I_0 contains the following items

data_type → • INT, ID
return_type → • INT, ID

The transition on INT gives the following set of items showing a reduce-reduce conflict on ID

data_type → INT •, ID
return_type → INT •, ID



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

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
<code>program → func_decl var_decl</code>	<code>program → func_decl var_decl</code>
<code>program → var_decl func_decl</code>	<code>program → var_decl func_decl</code>
<code>var_decl → data_type ID ;</code>	<code>var_decl → INT ID ;</code>
<code>data_type → INT</code>	<code>func_decl → INT ID ()</code>
<code>func_decl → return_type ID ()</code>	<code>func_decl → VOID ID ()</code>
<code>return_type → INT</code>	
<code>return_type → VOID</code>	



A Summary of Bottom Up Parsing Methods

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

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



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

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)