Parsing

Sujit Kumar Chakrabarti

IIITB

Bottom-up Parsing

Example

Grammar

$$
\begin{array}{lcl}
E & \rightarrow & E + T | T \\
T & \rightarrow & T * F | F \\
F & \rightarrow & (E) \\
 & | & \mathbf{id}
\end{array}
$$

Input: 1 * 2

Bottom-up Parsing

Example

id * **id**

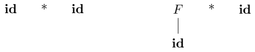
Grammar

$$\begin{array}{lcl} E & \rightarrow & E + T | T \\ T & \rightarrow & T * F | F \\ F & \rightarrow & (E) \\ & | & \mathbf{id} \end{array}$$

Input: 1 * 2

Bottom-up Parsing

Example



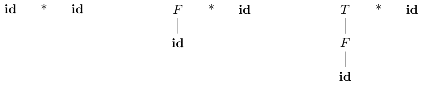
Grammar

$$
\begin{array}{lcl}
E & \rightarrow & E + T | T \\
T & \rightarrow & T * F | F \\
F & \rightarrow & (E) \\
 & | & \textbf{id}
\end{array}
$$

Input: 1 * 2

Bottom-up Parsing

Example



Grammar

$$\begin{array}{lcl} E & \rightarrow & E + T | T \\ T & \rightarrow & T * F | F \\ F & \rightarrow & (E) \\ & | & \mathbf{id} \end{array}$$

Input: 1 * 2

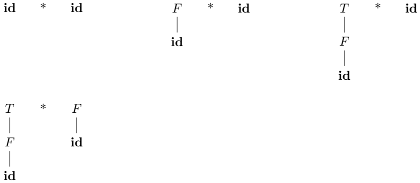
Bottom-up Parsing

Example

Grammar

$$\begin{array}{lcl} E & \rightarrow & E + T | T \\ T & \rightarrow & T * F | F \\ F & \rightarrow & (E) \\ & | & \textbf{id} \end{array}$$

Input: 1 * 2



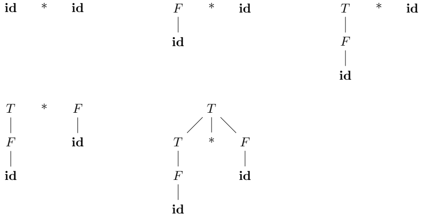
Bottom-up Parsing

Example

Grammar

$$
\begin{array}{lcl}
E & \rightarrow & E + T | T \\
T & \rightarrow & T * F | F \\
F & \rightarrow & (E) \\
  & | & \textbf{id}
\end{array}
$$

Input: 1 * 2



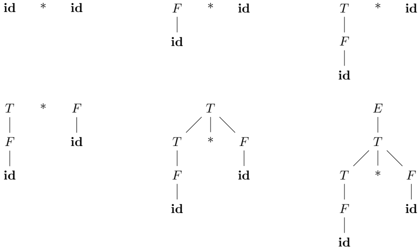
Bottom-up Parsing

Example

Grammar

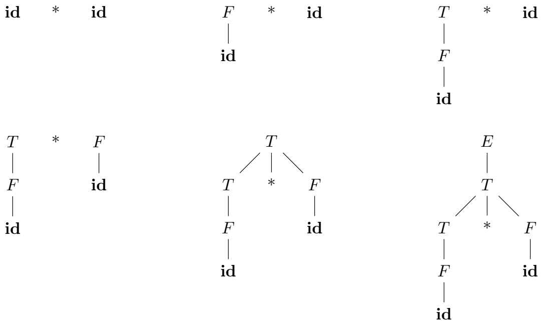
$$\begin{array}{lcl} E & \rightarrow & E + T | T \\ T & \rightarrow & T * F | F \\ F & \rightarrow & (E) \\ & | & \mathbf{id} \end{array}$$

Input: 1 * 2



Bottom-up Parsing

Example

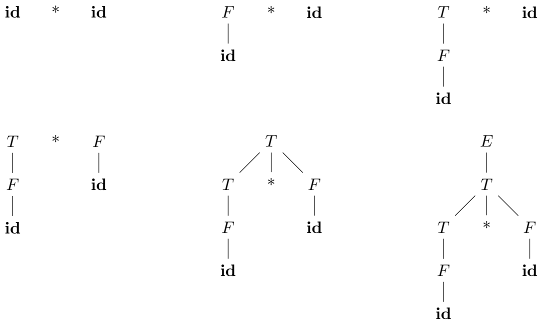


Derivation :

$$E \Rightarrow T \Rightarrow T * F \Rightarrow T * id \Rightarrow F * id \Rightarrow id * id$$

Bottom-up Parsing

Example



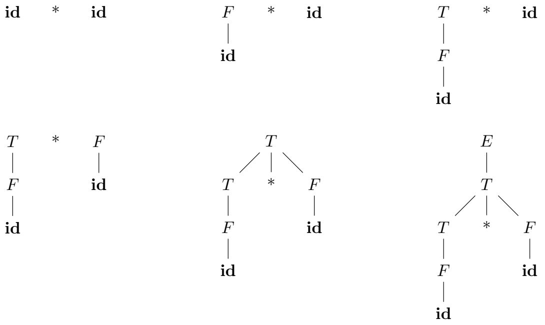
Derivation :

$E \Rightarrow T \Rightarrow T * F \Rightarrow T * id \Rightarrow F * id \Rightarrow id * id$

1. Rightmost derivation

Bottom-up Parsing

Example



Derivation :

$E \Rightarrow T \Rightarrow T * F \Rightarrow T * id \Rightarrow F * id \Rightarrow id * id$

1. Rightmost derivation
2. Reverse rightmost derivation – *Reduction*

Bottom-up Parsing

Right Sentential Form

- **Right Sentential Form.** Strings of grammar symbols that occur anytime during a rightmost derivation from a grammar G
- All *frontiers* during a bottom-up parse are right sentential forms.

Bottom-up Parsing

Handle

- **Handle.** A sub-string of a right-sentential form that appears as a right hand side of a grammar production, and which can be used to reduce the string to its *previous* right-sentential form
- $S \overset{*}{\Rightarrow} \alpha A w \Rightarrow \alpha \beta w$ with a production $A \rightarrow \beta$, then $A \rightarrow \beta$ is a handle.

Bottom-up Parsing

Handle

- **Handle.** A sub-string of a right-sentential form that appears as a right hand side of a grammar production, and which can be used to reduce the string to its *previous* right-sentential form
- $S \overset{*}{\Rightarrow} \alpha A w \Rightarrow \alpha\beta w$ with a production $A \rightarrow \beta$, then $A \rightarrow \beta$ is a handle.
- Bottom-up parsing can be viewed as the process of locating a handle in the n-th right-sentential form of S, and replacing that with the LHS of the production to get the $(n-1)$th right-sentential form.

Shift-Reduce Parsing

Stack	**Input**
$\$$	$w\$$

Shift-Reduce Parsing

$$\begin{array}{cc} \textbf{Stack} & \textbf{Input} \\ \hline \$ & w\$ \\ \multicolumn{2}{c}{\downarrow} \\ \$S & \$ \end{array}$$

Example:

Stack	**Input**	**Action**
$	$\mathbf{id}_1 * \mathbf{id}_2\$$	shift

Shift-Reduce Parsing

Stack	**Input**
$\$$	$w\$$
$\downarrow$	
$\$S$	$\$$

Example:

Stack	**Input**	**Action**
$\$$	$\mathbf{id}_1 * \mathbf{id}_2\$$	shift
$\$\mathbf{id}_1$	$*\mathbf{id}_2\$$	reduce by $F \rightarrow \mathbf{id}$

Shift-Reduce Parsing

$$\begin{array}{cc} \textbf{Stack} & \textbf{Input} \\ \hline \$ & w\$ \\ \multicolumn{2}{c}{\downarrow} \\ \$S & \$ \end{array}$$

Example:

Stack	**Input**	**Action**
$\$$	$\mathbf{id}_1 * \mathbf{id}_2\$$	shift
$\$\mathbf{id}_1$	$*\mathbf{id}_2\$$	reduce by $F \rightarrow \mathbf{id}$
$\$F$	$*\mathbf{id}_2\$$	reduce by $T \rightarrow F$

Shift-Reduce Parsing

$$\begin{array}{cc} \textbf{Stack} & \textbf{Input} \\ \hline \$ & w\$ \\ \multicolumn{2}{c}{\downarrow} \\ \$S & \$ \end{array}$$

Example:

Stack	**Input**	**Action**
$\$$	$\mathbf{id}_1 * \mathbf{id}_2\$$	shift
$\$\mathbf{id}_1$	$*\mathbf{id}_2\$$	reduce by $F \rightarrow \mathbf{id}$
$\$F$	$*\mathbf{id}_2\$$	reduce by $T \rightarrow F$
$\$T$	$*\mathbf{id}_2\$$	shift

Shift-Reduce Parsing

Stack	**Input**
$\$$	$w\$$
$\downarrow$	
$\$S$	$\$$

Example:

Stack	**Input**	**Action**
$\$$	$\mathbf{id}_1 * \mathbf{id}_2\$$	shift
$\$\mathbf{id}_1$	$*\mathbf{id}_2\$$	reduce by $F \to \mathbf{id}$
$\$F$	$*\mathbf{id}_2\$$	reduce by $T \to F$
$\$T$	$*\mathbf{id}_2\$$	shift
$\$T*$	$\mathbf{id}_2\$$	shift

Shift-Reduce Parsing

$$\begin{array}{cc} \textbf{Stack} & \textbf{Input} \\ \hline \$ & w\$ \\ \multicolumn{2}{c}{\downarrow} \\ \$S & \$ \end{array}$$

Example:

Stack	**Input**	**Action**
$\$$	$\mathbf{id}_1 * \mathbf{id}_2\$$	shift
$\$\mathbf{id}_1$	$*\mathbf{id}_2\$$	reduce by $F \rightarrow \mathbf{id}$
$\$F$	$*\mathbf{id}_2\$$	reduce by $T \rightarrow F$
$\$T$	$*\mathbf{id}_2\$$	shift
$\$T*$	$\mathbf{id}_2\$$	shift
$\$T * \mathbf{id}_2$	$\$$	reduce by $F \rightarrow \mathbf{id}$

Shift-Reduce Parsing

Stack	Input
$\$$	$w\$$
$\downarrow$	
$\$S$	$\$$

Example:

Stack	Input	Action
$\$$	$\mathbf{id}_1 * \mathbf{id}_2\$$	shift
$\$\mathbf{id}_1$	$*\mathbf{id}_2\$$	reduce by $F \to \mathbf{id}$
$\$F$	$*\mathbf{id}_2\$$	reduce by $T \to F$
$\$T$	$*\mathbf{id}_2\$$	shift
$\$T*$	$\mathbf{id}_2\$$	shift
$\$T * \mathbf{id}_2$	$\$$	reduce by $F \to \mathbf{id}$
$\$T * F$	$\$$	reduce by $T \to +T * F$

Shift-Reduce Parsing

$$\begin{array}{cc} \textbf{Stack} & \textbf{Input} \\ \hline \$ & w\$ \\ \multicolumn{2}{c}{\downarrow} \\ \$S & \$ \end{array}$$

Example:

Stack	**Input**	**Action**
$\$$	$\mathbf{id}_1 * \mathbf{id}_2\$$	shift
$\$\mathbf{id}_1$	$*\mathbf{id}_2\$$	reduce by $F \rightarrow \mathbf{id}$
$\$F$	$*\mathbf{id}_2\$$	reduce by $T \rightarrow F$
$\$T$	$*\mathbf{id}_2\$$	shift
$\$T*$	$\mathbf{id}_2\$$	shift
$\$T * \mathbf{id}_2$	$\$$	reduce by $F \rightarrow \mathbf{id}$
$\$T * F$	$\$$	reduce by $T \rightarrow +T * F$
$\$T$	$\$$	reduce by $E \rightarrow T$

Shift-Reduce Parsing

Stack	**Input**
$\$$	$w\$$
$\downarrow$	
$\$S$	$\$$

Example:

Stack	**Input**	**Action**
$\$$	$\mathbf{id}_1 * \mathbf{id}_2\$$	shift
$\$\mathbf{id}_1$	$*\mathbf{id}_2\$$	reduce by $F \to \mathbf{id}$
$\$F$	$*\mathbf{id}_2\$$	reduce by $T \to F$
$\$T$	$*\mathbf{id}_2\$$	shift
$\$T*$	$\mathbf{id}_2\$$	shift
$\$T * \mathbf{id}_2$	$\$$	reduce by $F \to \mathbf{id}$
$\$T * F$	$\$$	reduce by $T \to +T * F$
$\$T$	$\$$	reduce by $E \to T$
$\$E$	$\$$	

Shift-Reduce Parsing

Parser Actions

1. **Shift:** Shift the next input symbol onto the stack.
2. **Reduce:** Reduce a contiguous portion of the stack content including the top-of-stack by the LHS of an appropriate grammar production.

Shift-Reduce Parsing

Parser Actions

1. **Shift:** Shift the next input symbol onto the stack.
2. **Reduce:** Reduce a contiguous portion of the stack content including the top-of-stack by the LHS of an appropriate grammar production.
3. **Accept:** Declare success.
4. **Error:** Discover a syntax error.

Shift-Reduce Parsing

Reductions happen only at the top of the stack

1. $S \underset{rm}{\overset{*}{\Rightarrow}} \alpha A z \Rightarrow \alpha\beta B y z \underset{rm}{\Rightarrow} \alpha\beta\gamma y z$
2. $S \underset{rm}{\overset{*}{\Rightarrow}} \alpha B x A z \Rightarrow \alpha B x y z \underset{rm}{\Rightarrow} \alpha\gamma x y z$

Shift-Reduce Parsing

Reductions happen only at the top of the stack

1. $S \overset{*}{\underset{rm}{\Rightarrow}} \alpha A z \Rightarrow \alpha\beta B y z \underset{rm}{\Rightarrow} \alpha\beta\gamma y z$
2. $S \overset{*}{\underset{rm}{\Rightarrow}} \alpha B x A z \Rightarrow \alpha B x y z \underset{rm}{\Rightarrow} \alpha\gamma x y z$

Derivations to the left happen later. Reductions to the left happen earlier.

Shift-Reduce Parsing

Condition for a Successful Parse

Shift-Reduce Parsing

Condition for a Successful Parse

Viable Prefix

A prefix of a right-sentential form.

Shift-Reduce Parsing

Condition for a Successful Parse

Viable Prefix

A prefix of a right-sentential form.

Conflicts

1. **Shift-Reduce**
2. **Reduce-Reduce**

Shift-Reduce Parsing

Conflicts

Example

$$\begin{array}{lcl} \mathit{stmt} & \rightarrow & \text{if } \mathit{expr} \text{ then } \mathit{stmt} \\ & | & \text{if } \mathit{expr} \text{ then } \mathit{stmt} \text{ else } \mathit{stmt} \\ & | & \text{other} \end{array}$$

Bottom-up Parsing

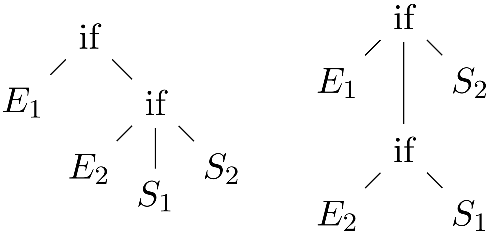
Conflict – Example

input: if E_1 then if E_2 then S_1 else S_2

Bottom-up Parsing

Conflict – Example

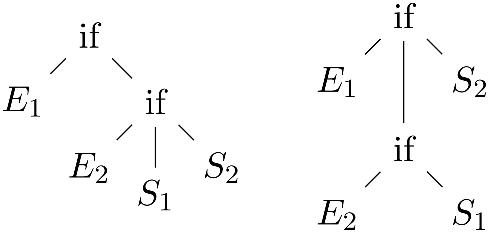
input: if E_1 then if E_2 then S_1 else S_2



Bottom-up Parsing

Conflict – Example

input: if E_1 then if E_2 then S_1 else S_2



- $\underline{\text{if } E_1 \text{ then } \underline{\text{if } E_2 \text{ then } S_1} \text{ else } S_2}$
- $\underline{\text{if } E_1 \text{ then } \underline{\text{if } E_2 \text{ then } S_1 \text{ else } S_2}}$

Bottom-up Parsing

LR Parsing

Properties

- Table driven
- LR grammars

Bottom-up Parsing

LR Parsing

Properties

- Table driven
- LR grammars

Advantages

- LR grammars are quite general.
- Syntax errors are detected early.
- $LR(k) \supset LL(k)$

Bottom-up Parsing

LR Parsing

Properties

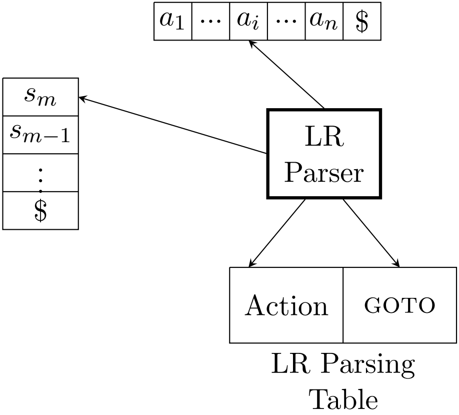
- Table driven
- LR grammars

Advantages

- LR grammars are quite general.
- Syntax errors are detected early.
- $LR(k) \supset LL(k)$

SLR Parsing

Parser Architecture


a_1
$\cdots$
a_i
$\cdots$
a_n
$
s_m
s_{m-1}
$\vdots$
$
LR Parser
Action
GOTO
LR Parsing Table