Syntax Analysis

Amitabha Sanyal

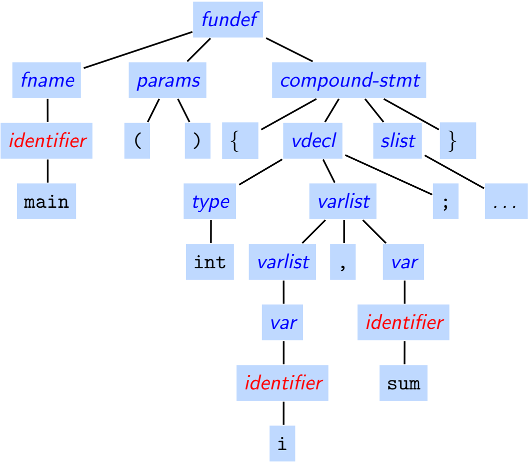
(www.cse.iitb.ac.in/˜as)

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

January 2014

Syntax analysis – example

Syntax analysis discovers the larger structures in a program.




```
main ()
{
  int i,sum;
  sum = 0;
  for (i=1; i<=10; i++)
    sum = sum + i;
  printf("%d\n",sum);
}
```

Parsing

A syntax analyzer or parser

- Ensures that the input program is well-formed by attempting to group tokens according to certain rules. This is syntax checking.

Parsing

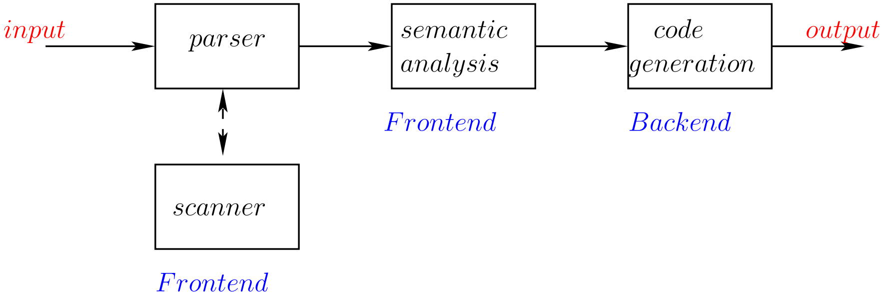
A syntax analyzer or parser

- Ensures that the input program is well-formed by attempting to group tokens according to certain rules. This is syntax checking.
-
    - May also create the hierarchical structure that arises out of such grouping.
    - The tree like representation of the structure is called a *parse tree*.
    - This information is required by subsequent phases.

Place of a parser in a compiler organization

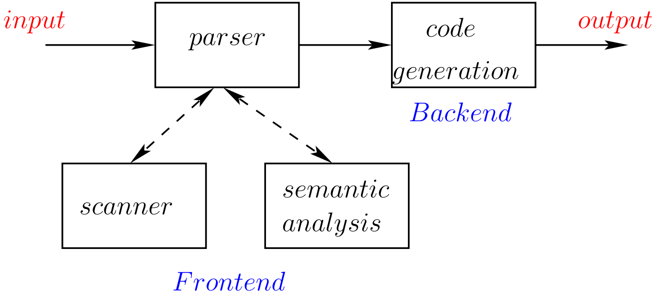
Where is the place of the parser in the overall organization of the compiler?

1. Parser driven syntax tree creation. The parser creates the entire syntax tree and passes control to the later stages.



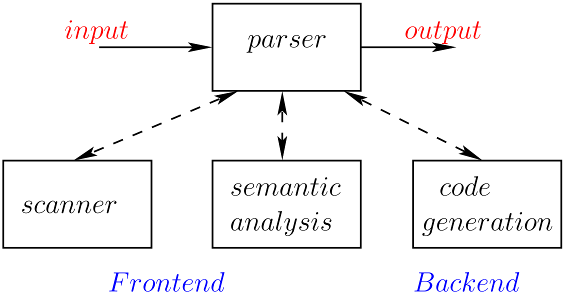
Place of a parser in a compiler organization

2. Parser driven front-end. The parser also does the semantic analysis along with parsing.



Place of a parser in a compiler organization

3. Parser driven compilation. The entire compilation is interleaved along with parsing.



Parser Construction

How are parsers constructed ?

- Till early seventies, parsers (in fact the entire compiler) were written manually.
- A better understanding of parsing algorithms has resulted in tools that can automatically generate parsers.
- Examples of parser generating tools:
  - Yacc/Bison: Bottom-up (LALR) parser generator
  - Antlr: Top-down (LL) scanner cum parser generator. (Terence Parr)
  - PCCTS:Precursor of Antlr (Terence Parr)
  - COCO/R: Lexer and Parser Generators in various languages, generates recursive descent parsers (Hanspeter Mossenbock).
  - Java Compiler Compiler (JavaCC)
  - ...

Specification of syntax

- To check whether a program is well-formed requires a specification of what is a well-formed program:

  1. The specification should be unambiguous.
  2. The specification should be correct and complete. Must cover all the syntactic details of the language
  3. the specification must be convenient to use by both language designer and the implementer

A *context free grammar* meets these requirements.

Context Free Grammar (CFG)

start symbol

$$
\begin{array}{lcl}
\textit{declaration} & \rightarrow & \textit{type idlist } ; \\
\textit{idlist} & \rightarrow & \textbf{id} \mid \textit{idlist} \ , \ \textbf{id} \\
\boxed{\textit{type} \quad \rightarrow \quad \texttt{integer} \mid \texttt{float}}
\end{array}
$$

terminals

non-terminals

production

A CFG G is a 4-tuple (N, T, S, P), where :

1. N is a finite set of nonterminals.
2. T is a finite set of terminals.
3. S is a special nonterminal (from N) called the *start* symbol.
4. P is a finite set of production rules of the form such as $A \rightarrow \alpha$, where A is from N and α from $(N \cup T)^*$

Derivation

What is the language defined by a grammar? To answer this, we need the notion of a *derivation*.

Derivation

What is the language defined by a grammar? To answer this, we need the notion of a *derivation*.

A derivation is traced out as follows:

$$
\begin{array}{ll}
 & \textit{declaration} \\
\Rightarrow & \textit{type idlist}; \\
\Rightarrow & \textbf{integer } \textit{idlist}; \\
\Rightarrow & \textbf{integer } \textit{idlist}, \textbf{ id}; \\
\Rightarrow & \textbf{integer id}, \textbf{ id};
\end{array}
$$

Derivation

What is the language defined by a grammar? To answer this, we need the notion of a *derivation*.

A derivation is traced out as follows:

$$
\begin{array}{ll}
 & \textit{declaration} \\
\Rightarrow & \textit{type idlist}; \\
\Rightarrow & \textbf{integer } \textit{idlist}; \\
\Rightarrow & \textbf{integer } \textit{idlist}, \textbf{id}; \\
\Rightarrow & \textbf{integer id}, \textbf{id};
\end{array}
$$

- A *derivation* is the transformation of a string of grammar symbols by replacing a non-terminal by the corresponding right hand side of a production.
- The set of all possible *terminal strings* that can be derived from the start symbol of a CFG is the language generated by the CFG.

Specification of Syntax by Context Free Grammars

Informal description of variable declarations in C:

- starts with `integer` or `float` as the first token.
- followed by identifier tokens, separated by token **comma**
- followed by token **semicolon**

Question: Can the list of identifier tokens be empty?

$$
\begin{array}{lcl}
\textit{declaration} & \rightarrow & \textit{type idlist} \ ; \\
\textit{idlist} & \rightarrow & \mathbf{id} \mid \textit{idlist} \ , \ \mathbf{id} \\
\textit{type} & \rightarrow & \texttt{integer} \mid \texttt{float}
\end{array}
$$

Illustrates the usefulness of a formal specification.

Question: How does one write a grammar in which the list of identifiers is empty?

Representing programming language constructs using grammars

- Question: How does one write a grammar for assignment statements?
- Question: What language does the following grammar represent?

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

Why the Term "Context Free" ?

1. The only kind of productions permitted are of the form
   non-terminal $\rightarrow$ *sequence of terminals and non-terminals*
2. In a derivation, the replacement is made regardless of the context (symbols surrounding the non-terminal).

As a contrast, observe this context-sensitive grammar.

$$
\begin{array}{lcl}
\mathit{sentence} & \rightarrow & NP \quad VP \\
NP & \rightarrow & \text{the} \quad SN \mid \text{the} \quad PN \\
SN \quad VP & \rightarrow & SN \quad SV \\
PN \quad VP & \rightarrow & PN \quad PV \\
SN \rightarrow & \text{child} & \\
PN \rightarrow & \text{children} & \\
SV \rightarrow & \text{plays} & \\
PV \rightarrow & \text{play} &
\end{array}
$$

Notational Conventions

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

Derivation as a relation

Consider the derivation:

$$
\begin{array}{ll}
 & \textit{declaration} \\
\Rightarrow & \textit{type idlist}; \\
\Rightarrow & \textbf{integer } \textit{idlist}; \\
\Rightarrow & \textbf{integer } \textit{idlist}, \textbf{id}; \\
\Rightarrow & \textbf{integer id}, \textbf{id};
\end{array}
$$

We would like to say:

$$
\begin{array}{l}
\textit{type idlist}; \Rightarrow \textbf{integer } \textit{idlist}; \\
\qquad\qquad\quad \Rightarrow \textbf{integer } \textit{idlist}, \textbf{id}; \\
\qquad\qquad\quad \Rightarrow \textbf{integer id}, \textbf{id}; \\
\textit{type idlist}; \stackrel{+}{\Rightarrow} \textbf{integer id}, \textbf{id};
\end{array}
$$

Derivation as a relation

- $A \to \gamma$ – a production rule
- $\alpha\ A\ \beta$ – a string of grammar symbols
-
  - Replacing A in $\alpha A \beta$ by its RHS (γ) yields $\alpha\gamma\beta$.
  - Formally, this is stated as $\alpha\ A\ \beta$ derives $\alpha\ \gamma\ \beta$ in one step.
  - Symbolically $\alpha\ A\ \beta \Rightarrow \alpha\ \gamma\ \beta$.
- $\alpha_1 \Rightarrow \alpha_2$ means α_1 derives α_2 in one step.
- $\alpha_1 \overset{*}{\Rightarrow} \alpha_2$ means α_1 derives α_2 in zero or more steps. Clearly $\alpha \overset{*}{\Rightarrow} \alpha$ is always true for any α.
- $\alpha_1 \overset{+}{\Rightarrow} \alpha_2$ means α_1 derives α_2 in one or more steps.

Sentential forms and sentences

- The *language* $L(G)$ generated by a grammar G is defined as $\{w \mid S \stackrel{+}{\Rightarrow} w, w \in T^*\}$.

  The language generated by the type declaration grammar is the set of strings consisting of:
  - A type name (**integer** or **float**), followed by
  - a **,** separated list of one or more **id**s, followed by
  - a **;**.

  Strings in $L(G)$ are called *sentences* of G.

Sentential forms and sentences

- A string α, $\alpha \in (N \bigcup T)^*$, such that $S \stackrel{*}{\Rightarrow} \alpha$, is called a *sentential form* of G.
  - *type idlist*;,
    integer *idlist*, **id;**, and
    integer id, id; are all sentential forms.

    However, only **integer id, id;** is a sentence.

Equivalent grammars

- Two grammars are *equivalent*,if they generate the same language.

- The grammars:

$$
\begin{array}{lcl}
\textit{declaration} & \rightarrow & \textit{type idlist} \ ; \\
\textit{idlist} & \rightarrow & \textbf{id} \ | \ \textit{idlist} \ , \textbf{id} \\
\textit{type} & \rightarrow & \texttt{integer} \ | \ \texttt{float}
\end{array}
$$

  and

$$
\begin{array}{lcl}
\textit{declaration} & \rightarrow & \textit{type idlist} \ ; \\
\textit{idlist} & \rightarrow & \textbf{id} \ \textit{commaidlist} \\
\textit{commaidlist} & \rightarrow & , \textbf{id} \ \textit{commaidlist} \ | \ \epsilon \\
\textit{type} & \rightarrow & \texttt{integer} \ | \ \texttt{float}
\end{array}
$$

  are equivalent.

Leftmost and rightmost derivations

- During a derivation, there is choice of non-terminals to expand at each sentential form.

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

Leftmost and rightmost derivations

- During a derivation, there is choice of non-terminals to expand at each sentential form.

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

Leftmost derivation: Expand the leftmost non-terminal.

Leftmost and rightmost derivations

- During a derivation, there is choice of non-terminals to expand at each sentential form.

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

Leftmost derivation: Expand the leftmost non-terminal.

$$\underline{E} \overset{lm}{\Rightarrow} \underline{E} + T$$

Leftmost and rightmost derivations

- During a derivation, there is choice of non-terminals to expand at each sentential form.

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

Leftmost derivation: Expand the leftmost non-terminal.

$$\begin{array}{lcl} \underline{E} & \stackrel{lm}{\Rightarrow} & \underline{E} + T \\ & \stackrel{lm}{\Rightarrow} & \underline{E} + T + T \end{array}$$

Leftmost and rightmost derivations

- During a derivation, there is choice of non-terminals to expand at each sentential form.

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

Leftmost derivation: Expand the leftmost non-terminal.

$$\begin{array}{lcl} \underline{E} & \overset{lm}{\Rightarrow} & \underline{E} + T \\ & \overset{lm}{\Rightarrow} & \underline{E} + T + T \\ & \overset{lm}{\Rightarrow} & \underline{T} + T + T \end{array}$$

Leftmost and rightmost derivations

- During a derivation, there is choice of non-terminals to expand at each sentential form.

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

Leftmost derivation: Expand the leftmost non-terminal.

$$
\begin{array}{lcl}
\underline{E} & \overset{lm}{\Rightarrow} & \underline{E} + T \\
 & \overset{lm}{\Rightarrow} & \underline{E} + T + T \\
 & \overset{lm}{\Rightarrow} & \underline{T} + T + T \\
 & \overset{lm}{\Rightarrow} & \underline{F} + T + T
\end{array}
$$

Leftmost and rightmost derivations

- During a derivation, there is choice of non-terminals to expand at each sentential form.

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

Leftmost derivation: Expand the leftmost non-terminal.

$$\begin{array}{lcl} \underline{E} & \overset{lm}{\Rightarrow} & \underline{E} + T \\ & \overset{lm}{\Rightarrow} & \underline{E} + T + T \\ & \overset{lm}{\Rightarrow} & \underline{T} + T + T \\ & \overset{lm}{\Rightarrow} & \underline{F} + T + T \\ & \overset{lm}{\Rightarrow} & id + \underline{T} + T \end{array}$$

Leftmost and rightmost derivations

- During a derivation, there is choice of non-terminals to expand at each sentential form.

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

Leftmost derivation: Expand the leftmost non-terminal.

$$
\begin{array}{lcl}
\underline{E} & \stackrel{lm}{\Rightarrow} & \underline{E} + T \\
 & \stackrel{lm}{\Rightarrow} & \underline{E} + T + T \\
 & \stackrel{lm}{\Rightarrow} & \underline{T} + T + T \\
 & \stackrel{lm}{\Rightarrow} & \underline{F} + T + T \\
 & \stackrel{lm}{\Rightarrow} & id + \underline{T} + T \\
 & \stackrel{lm}{\Rightarrow} & id + \underline{F} + T
\end{array}
$$

Leftmost and rightmost derivations

- During a derivation, there is choice of non-terminals to expand at each sentential form.

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

Leftmost derivation: Expand the leftmost non-terminal.

$$
\begin{array}{lcl}
\underline{E} & \stackrel{lm}{\Rightarrow} & \underline{E} + T \\
 & \stackrel{lm}{\Rightarrow} & \underline{E} + T + T \\
 & \stackrel{lm}{\Rightarrow} & \underline{T} + T + T \\
 & \stackrel{lm}{\Rightarrow} & \underline{F} + T + T \\
 & \stackrel{lm}{\Rightarrow} & id + \underline{T} + T \\
 & \stackrel{lm}{\Rightarrow} & id + \underline{F} + T \\
 & \stackrel{lm}{\Rightarrow} & id + id + \underline{T}
\end{array}
$$

Leftmost and rightmost derivations

- During a derivation, there is choice of non-terminals to expand at each sentential form.

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

Leftmost derivation: Expand the leftmost non-terminal.

$$
\begin{array}{lcl}
\underline{E} & \overset{lm}{\Rightarrow} & \underline{E} + T \\
 & \overset{lm}{\Rightarrow} & \underline{E} + T + T \\
 & \overset{lm}{\Rightarrow} & \underline{T} + T + T \\
 & \overset{lm}{\Rightarrow} & \underline{F} + T + T \\
 & \overset{lm}{\Rightarrow} & id + \underline{T} + T \\
 & \overset{lm}{\Rightarrow} & id + \underline{F} + T \\
 & \overset{lm}{\Rightarrow} & id + id + \underline{T} \\
 & \overset{lm}{\Rightarrow} & id + id + \underline{F}
\end{array}
$$

Leftmost and rightmost derivations

- During a derivation, there is choice of non-terminals to expand at each sentential form.

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

Leftmost derivation: Expand the leftmost non-terminal.

$$\begin{array}{lcl} \underline{E} & \stackrel{lm}{\Rightarrow} & \underline{E} + T \\ & \stackrel{lm}{\Rightarrow} & \underline{E} + T + T \\ & \stackrel{lm}{\Rightarrow} & \underline{T} + T + T \\ & \stackrel{lm}{\Rightarrow} & \underline{F} + T + T \\ & \stackrel{lm}{\Rightarrow} & id + \underline{T} + T \\ & \stackrel{lm}{\Rightarrow} & id + \underline{F} + T \\ & \stackrel{lm}{\Rightarrow} & id + id + \underline{T} \\ & \stackrel{lm}{\Rightarrow} & id + id + \underline{F} \\ & \stackrel{lm}{\Rightarrow} & id + id + id \end{array}$$

Leftmost and rightmost derivations

- During a derivation, there is choice of non-terminals to expand at each sentential form.

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

Leftmost derivation: Expand the leftmost non-terminal.

$$\begin{array}{lcl} \underline{E} & \overset{lm}{\Rightarrow} & \underline{E} + T \\ & \overset{lm}{\Rightarrow} & \underline{E} + T + T \\ & \overset{lm}{\Rightarrow} & \underline{T} + T + T \\ & \overset{lm}{\Rightarrow} & \underline{F} + T + T \\ & \overset{lm}{\Rightarrow} & id + \underline{T} + T \\ & \overset{lm}{\Rightarrow} & id + \underline{F} + T \\ & \overset{lm}{\Rightarrow} & id + id + \underline{T} \\ & \overset{lm}{\Rightarrow} & id + id + \underline{F} \\ & \overset{lm}{\Rightarrow} & id + id + id \end{array}$$

Rightmost derivation: Expand the rightmost non-terminal.

Leftmost and rightmost derivations

- During a derivation, there is choice of non-terminals to expand at each sentential form.

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

Leftmost derivation: Expand the leftmost non-terminal.

$$
\begin{array}{lcl}
\underline{E} & \overset{lm}{\Rightarrow} & \underline{E} + T \\
 & \overset{lm}{\Rightarrow} & \underline{E} + T + T \\
 & \overset{lm}{\Rightarrow} & \underline{T} + T + T \\
 & \overset{lm}{\Rightarrow} & \underline{F} + T + T \\
 & \overset{lm}{\Rightarrow} & id + \underline{T} + T \\
 & \overset{lm}{\Rightarrow} & id + \underline{F} + T \\
 & \overset{lm}{\Rightarrow} & id + id + \underline{T} \\
 & \overset{lm}{\Rightarrow} & id + id + \underline{F} \\
 & \overset{lm}{\Rightarrow} & id + id + id
\end{array}
$$

Rightmost derivation: Expand the rightmost non-terminal.

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

Leftmost and rightmost derivations

- During a derivation, there is choice of non-terminals to expand at each sentential form.

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

Leftmost derivation: Expand the leftmost non-terminal.

$$
\begin{array}{lcl}
\underline{E} & \overset{lm}{\Rightarrow} & \underline{E} + T \\
 & \overset{lm}{\Rightarrow} & \underline{E} + T + T \\
 & \overset{lm}{\Rightarrow} & \underline{T} + T + T \\
 & \overset{lm}{\Rightarrow} & \underline{F} + T + T \\
 & \overset{lm}{\Rightarrow} & id + \underline{T} + T \\
 & \overset{lm}{\Rightarrow} & id + \underline{F} + T \\
 & \overset{lm}{\Rightarrow} & id + id + \underline{T} \\
 & \overset{lm}{\Rightarrow} & id + id + \underline{F} \\
 & \overset{lm}{\Rightarrow} & id + id + id
\end{array}
$$

Rightmost derivation: Expand the rightmost non-terminal.

$$
\begin{array}{lcl}
\underline{E} & \overset{rm}{\Rightarrow} & E + \underline{T} \\
 & \overset{rm}{\Rightarrow} & E + \underline{F}
\end{array}
$$

Leftmost and rightmost derivations

- During a derivation, there is choice of non-terminals to expand at each sentential form.

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

Leftmost derivation: Expand the leftmost non-terminal.

$$
\begin{array}{lcl}
\underline{E} & \overset{lm}{\Rightarrow} & \underline{E} + T \\
 & \overset{lm}{\Rightarrow} & \underline{E} + T + T \\
 & \overset{lm}{\Rightarrow} & \underline{T} + T + T \\
 & \overset{lm}{\Rightarrow} & \underline{F} + T + T \\
 & \overset{lm}{\Rightarrow} & id + \underline{T} + T \\
 & \overset{lm}{\Rightarrow} & id + \underline{F} + T \\
 & \overset{lm}{\Rightarrow} & id + id + \underline{T} \\
 & \overset{lm}{\Rightarrow} & id + id + \underline{F} \\
 & \overset{lm}{\Rightarrow} & id + id + id
\end{array}
$$

Rightmost derivation: Expand the rightmost non-terminal.

$$
\begin{array}{lcl}
\underline{E} & \overset{rm}{\Rightarrow} & E + \underline{T} \\
 & \overset{rm}{\Rightarrow} & E + \underline{F} \\
 & \overset{rm}{\Rightarrow} & \underline{E} + id
\end{array}
$$

Leftmost and rightmost derivations

- During a derivation, there is choice of non-terminals to expand at each sentential form.

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

Leftmost derivation: Expand the leftmost non-terminal.

$$
\begin{array}{lcl}
\underline{E} & \stackrel{lm}{\Rightarrow} & \underline{E} + T \\
 & \stackrel{lm}{\Rightarrow} & \underline{E} + T + T \\
 & \stackrel{lm}{\Rightarrow} & \underline{T} + T + T \\
 & \stackrel{lm}{\Rightarrow} & \underline{F} + T + T \\
 & \stackrel{lm}{\Rightarrow} & id + \underline{T} + T \\
 & \stackrel{lm}{\Rightarrow} & id + \underline{F} + T \\
 & \stackrel{lm}{\Rightarrow} & id + id + \underline{T} \\
 & \stackrel{lm}{\Rightarrow} & id + id + \underline{F} \\
 & \stackrel{lm}{\Rightarrow} & id + id + id
\end{array}
$$

Rightmost derivation: Expand the rightmost non-terminal.

$$
\begin{array}{lcl}
\underline{E} & \stackrel{rm}{\Rightarrow} & E + \underline{T} \\
 & \stackrel{rm}{\Rightarrow} & E + \underline{F} \\
 & \stackrel{rm}{\Rightarrow} & \underline{E} + id \\
 & \stackrel{rm}{\Rightarrow} & E + \underline{T} + id
\end{array}
$$

Leftmost and rightmost derivations

- During a derivation, there is choice of non-terminals to expand at each sentential form.

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

Leftmost derivation: Expand the leftmost non-terminal.

$$
\begin{array}{lcl}
\underline{E} & \overset{lm}{\Rightarrow} & \underline{E} + T \\
 & \overset{lm}{\Rightarrow} & \underline{E} + T + T \\
 & \overset{lm}{\Rightarrow} & \underline{T} + T + T \\
 & \overset{lm}{\Rightarrow} & \underline{F} + T + T \\
 & \overset{lm}{\Rightarrow} & id + \underline{T} + T \\
 & \overset{lm}{\Rightarrow} & id + \underline{F} + T \\
 & \overset{lm}{\Rightarrow} & id + id + \underline{T} \\
 & \overset{lm}{\Rightarrow} & id + id + \underline{F} \\
 & \overset{lm}{\Rightarrow} & id + id + id
\end{array}
$$

Rightmost derivation: Expand the rightmost non-terminal.

$$
\begin{array}{lcl}
\underline{E} & \overset{rm}{\Rightarrow} & E + \underline{T} \\
 & \overset{rm}{\Rightarrow} & E + \underline{F} \\
 & \overset{rm}{\Rightarrow} & \underline{E} + id \\
 & \overset{rm}{\Rightarrow} & E + \underline{T} + id \\
 & \overset{rm}{\Rightarrow} & E + \underline{F} + id
\end{array}
$$

Leftmost and rightmost derivations

- During a derivation, there is choice of non-terminals to expand at each sentential form.

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

Leftmost derivation: Expand the leftmost non-terminal.

$$
\begin{array}{lcl}
\underline{E} & \overset{lm}{\Rightarrow} & \underline{E} + T \\
 & \overset{lm}{\Rightarrow} & \underline{E} + T + T \\
 & \overset{lm}{\Rightarrow} & \underline{T} + T + T \\
 & \overset{lm}{\Rightarrow} & \underline{F} + T + T \\
 & \overset{lm}{\Rightarrow} & id + \underline{T} + T \\
 & \overset{lm}{\Rightarrow} & id + \underline{F} + T \\
 & \overset{lm}{\Rightarrow} & id + id + \underline{T} \\
 & \overset{lm}{\Rightarrow} & id + id + \underline{F} \\
 & \overset{lm}{\Rightarrow} & id + id + id
\end{array}
$$

Rightmost derivation: Expand the rightmost non-terminal.

$$
\begin{array}{lcl}
\underline{E} & \overset{rm}{\Rightarrow} & E + \underline{T} \\
 & \overset{rm}{\Rightarrow} & E + \underline{F} \\
 & \overset{rm}{\Rightarrow} & \underline{E} + id \\
 & \overset{rm}{\Rightarrow} & E + \underline{T} + id \\
 & \overset{rm}{\Rightarrow} & E + \underline{F} + id \\
 & \overset{rm}{\Rightarrow} & \underline{E} + id + id
\end{array}
$$

Leftmost and rightmost derivations

- During a derivation, there is choice of non-terminals to expand at each sentential form.

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

Leftmost derivation: Expand the leftmost non-terminal.

$$
\begin{array}{lcl}
\underline{E} & \stackrel{lm}{\Rightarrow} & \underline{E} + T \\
 & \stackrel{lm}{\Rightarrow} & \underline{E} + T + T \\
 & \stackrel{lm}{\Rightarrow} & \underline{T} + T + T \\
 & \stackrel{lm}{\Rightarrow} & \underline{F} + T + T \\
 & \stackrel{lm}{\Rightarrow} & id + \underline{T} + T \\
 & \stackrel{lm}{\Rightarrow} & id + \underline{F} + T \\
 & \stackrel{lm}{\Rightarrow} & id + id + \underline{T} \\
 & \stackrel{lm}{\Rightarrow} & id + id + \underline{F} \\
 & \stackrel{lm}{\Rightarrow} & id + id + id
\end{array}
$$

Rightmost derivation: Expand the rightmost non-terminal.

$$
\begin{array}{lcl}
\underline{E} & \stackrel{rm}{\Rightarrow} & E + \underline{T} \\
 & \stackrel{rm}{\Rightarrow} & E + \underline{F} \\
 & \stackrel{rm}{\Rightarrow} & \underline{E} + id \\
 & \stackrel{rm}{\Rightarrow} & E + \underline{T} + id \\
 & \stackrel{rm}{\Rightarrow} & E + \underline{F} + id \\
 & \stackrel{rm}{\Rightarrow} & \underline{E} + id + id \\
 & \stackrel{rm}{\Rightarrow} & \underline{T} + id + id
\end{array}
$$

Leftmost and rightmost derivations

- During a derivation, there is choice of non-terminals to expand at each sentential form.

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

Leftmost derivation: Expand the leftmost non-terminal.

$$\begin{array}{lcl} \underline{E} & \overset{lm}{\Rightarrow} & \underline{E} + T \\ & \overset{lm}{\Rightarrow} & \underline{E} + T + T \\ & \overset{lm}{\Rightarrow} & \underline{T} + T + T \\ & \overset{lm}{\Rightarrow} & \underline{F} + T + T \\ & \overset{lm}{\Rightarrow} & id + \underline{T} + T \\ & \overset{lm}{\Rightarrow} & id + \underline{F} + T \\ & \overset{lm}{\Rightarrow} & id + id + \underline{T} \\ & \overset{lm}{\Rightarrow} & id + id + \underline{F} \\ & \overset{lm}{\Rightarrow} & id + id + id \end{array}$$

Rightmost derivation: Expand the rightmost non-terminal.

$$\begin{array}{lcl} \underline{E} & \overset{rm}{\Rightarrow} & E + \underline{T} \\ & \overset{rm}{\Rightarrow} & E + \underline{F} \\ & \overset{rm}{\Rightarrow} & \underline{E} + id \\ & \overset{rm}{\Rightarrow} & E + \underline{T} + id \\ & \overset{rm}{\Rightarrow} & E + \underline{F} + id \\ & \overset{rm}{\Rightarrow} & \underline{E} + id + id \\ & \overset{rm}{\Rightarrow} & \underline{T} + id + id \\ & \overset{rm}{\Rightarrow} & \underline{F} + id + id \end{array}$$

Leftmost and rightmost derivations

- During a derivation, there is choice of non-terminals to expand at each sentential form.

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

Leftmost derivation: Expand the leftmost non-terminal.

$$
\begin{array}{lcl}
\underline{E} & \overset{lm}{\Rightarrow} & \underline{E} + T \\
 & \overset{lm}{\Rightarrow} & \underline{E} + T + T \\
 & \overset{lm}{\Rightarrow} & \underline{T} + T + T \\
 & \overset{lm}{\Rightarrow} & \underline{F} + T + T \\
 & \overset{lm}{\Rightarrow} & id + \underline{T} + T \\
 & \overset{lm}{\Rightarrow} & id + \underline{F} + T \\
 & \overset{lm}{\Rightarrow} & id + id + \underline{T} \\
 & \overset{lm}{\Rightarrow} & id + id + \underline{F} \\
 & \overset{lm}{\Rightarrow} & id + id + id
\end{array}
$$

Rightmost derivation: Expand the rightmost non-terminal.

$$
\begin{array}{lcl}
\underline{E} & \overset{rm}{\Rightarrow} & E + \underline{T} \\
 & \overset{rm}{\Rightarrow} & E + \underline{F} \\
 & \overset{rm}{\Rightarrow} & \underline{E} + id \\
 & \overset{rm}{\Rightarrow} & E + \underline{T} + id \\
 & \overset{rm}{\Rightarrow} & E + \underline{F} + id \\
 & \overset{rm}{\Rightarrow} & \underline{E} + id + id \\
 & \overset{rm}{\Rightarrow} & \underline{T} + id + id \\
 & \overset{rm}{\Rightarrow} & \underline{F} + id + id \\
 & \overset{rm}{\Rightarrow} & id + id + id
\end{array}
$$

Leftmost and rightmost derivations

- For constructing a derivation, there are choices at each sentential form.
  - choice of the non-terminal to be replaced
  - choice of a rule corresponding to the non-terminal.
- Instead of choosing the non-terminal to be replaced, in an arbitrary fashion, it is possible to make an uniform choice at each step.
  - *leftmost derivation:* replace the *leftmost non-terminal* in a sentential form
  - *rightmost derivation:* replace the *rightmost non-terminal* in a sentential form

Parse Trees

What is common to the leftmost derivation and the rightmost derivation shown before?

Leftmost derivation:

$$\begin{array}{lll}
\underline{E} & \overset{lm}{\Rightarrow} & \underline{E} + T \\
& \overset{lm}{\Rightarrow} & \underline{E} + T + T \\
& \overset{lm}{\Rightarrow} & \underline{T} + T + T \\
& \overset{lm}{\Rightarrow} & \underline{F} + T + T \\
& \overset{lm}{\Rightarrow} & id + \underline{T} + T \\
& \overset{lm}{\Rightarrow} & id + \underline{F} + T \\
& \overset{lm}{\Rightarrow} & id + id + \underline{T} \\
& \overset{lm}{\Rightarrow} & id + id + \underline{F} \\
& \overset{lm}{\Rightarrow} & id + id + id
\end{array}$$

Rightmost derivation:

$$\begin{array}{lll}
\underline{E} & \overset{rm}{\Rightarrow} & E + \underline{T} \\
& \overset{rm}{\Rightarrow} & E + \underline{F} \\
& \overset{rm}{\Rightarrow} & \underline{E} + id \\
& \overset{rm}{\Rightarrow} & E + \underline{T} + id \\
& \overset{rm}{\Rightarrow} & E + \underline{F} + id \\
& \overset{rm}{\Rightarrow} & \underline{E} + id + id \\
& \overset{rm}{\Rightarrow} & \underline{T} + id + id \\
& \overset{rm}{\Rightarrow} & \underline{F} + id + id \\
& \overset{rm}{\Rightarrow} & id + id + id
\end{array}$$

Parse Trees

What is common to the leftmost derivation and the rightmost derivation shown before?

Leftmost derivation:

$$
\begin{array}{lcl}
\underline{E} & \overset{lm}{\Rightarrow} & \underline{E} + T \\
 & \overset{lm}{\Rightarrow} & \underline{E} + T + T \\
 & \overset{lm}{\Rightarrow} & \underline{T} + T + T \\
 & \overset{lm}{\Rightarrow} & \underline{F} + T + T \\
 & \overset{lm}{\Rightarrow} & id + \underline{T} + T \\
 & \overset{lm}{\Rightarrow} & id + \underline{F} + T \\
 & \overset{lm}{\Rightarrow} & id + id + \underline{T} \\
 & \overset{lm}{\Rightarrow} & id + id + \underline{F} \\
 & \overset{lm}{\Rightarrow} & id + id + id
\end{array}
$$

Rightmost derivation:

$$
\begin{array}{lcl}
\underline{E} & \overset{rm}{\Rightarrow} & E + \underline{T} \\
 & \overset{rm}{\Rightarrow} & E + \underline{F} \\
 & \overset{rm}{\Rightarrow} & \underline{E} + id \\
 & \overset{rm}{\Rightarrow} & E + \underline{T} + id \\
 & \overset{rm}{\Rightarrow} & E + \underline{F} + id \\
 & \overset{rm}{\Rightarrow} & \underline{E} + id + id \\
 & \overset{rm}{\Rightarrow} & \underline{T} + id + id \\
 & \overset{rm}{\Rightarrow} & \underline{F} + id + id \\
 & \overset{rm}{\Rightarrow} & id + id + id
\end{array}
$$

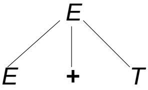
If a non-terminal A is replaced using a production $A \rightarrow \alpha$ in a left-sentential form, then A is also replaced by the same rule in a right-sentential form.

Parse Trees

The commonality of the two derivations is expressed as a parse tree.

Parse Trees

The commonality of the two derivations is expressed as a parse tree.

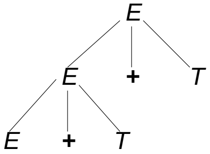


Leftmost derivation:

$$\underline{E} \quad \overset{lm}{\Rightarrow} \quad \underline{E} + T$$

Parse Trees

The commonality of the two derivations is expressed as a parse tree.

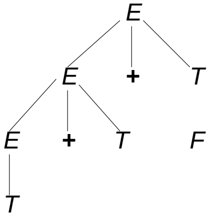


Leftmost derivation:

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

Parse Trees

The commonality of the two derivations is expressed as a parse tree.

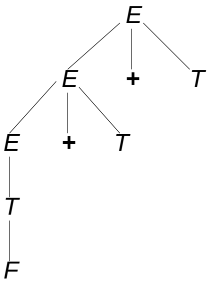


Leftmost derivation:

$$
\begin{array}{lcl}
\underline{E} & \overset{lm}{\Rightarrow} & \underline{E} + T \\
 & \overset{lm}{\Rightarrow} & \underline{E} + T + T \\
 & \overset{lm}{\Rightarrow} & \underline{T} + T + T
\end{array}
$$

Parse Trees

The commonality of the two derivations is expressed as a parse tree.

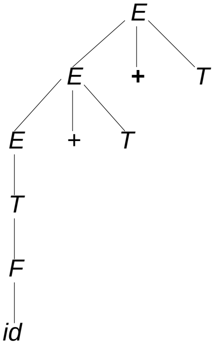


Leftmost derivation:

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

Parse Trees

The commonality of the two derivations is expressed as a parse tree.

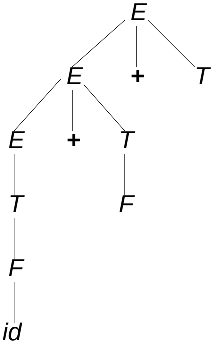


Leftmost derivation:

$$
\begin{array}{lcl}
\underline{E} & \overset{lm}{\Rightarrow} & \underline{E} + T \\
 & \overset{lm}{\Rightarrow} & \underline{E} + T + T \\
 & \overset{lm}{\Rightarrow} & \underline{T} + T + T \\
 & \overset{lm}{\Rightarrow} & \underline{F} + T + T \\
 & \overset{lm}{\Rightarrow} & id + \underline{T} + T
\end{array}
$$

Parse Trees

The commonality of the two derivations is expressed as a parse tree.

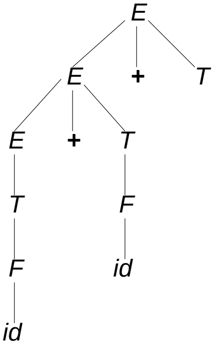


Leftmost derivation:

$$
\begin{array}{lcl}
\underline{E} & \overset{lm}{\Rightarrow} & \underline{E} + T \\
 & \overset{lm}{\Rightarrow} & \underline{E} + T + T \\
 & \overset{lm}{\Rightarrow} & \underline{T} + T + T \\
 & \overset{lm}{\Rightarrow} & \underline{F} + T + T \\
 & \overset{lm}{\Rightarrow} & id + \underline{T} + T \\
 & \overset{lm}{\Rightarrow} & id + \underline{F} + T
\end{array}
$$

Parse Trees

The commonality of the two derivations is expressed as a parse tree.

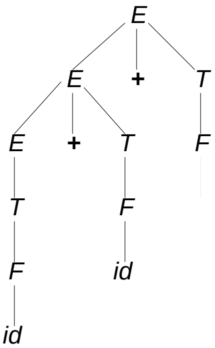


Leftmost derivation:

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

Parse Trees

The commonality of the two derivations is expressed as a parse tree.

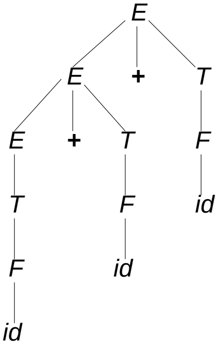


Leftmost derivation:

$$
\begin{array}{lcl}
\underline{E} & \overset{lm}{\Rightarrow} & \underline{E} + T \\
 & \overset{lm}{\Rightarrow} & \underline{E} + T + T \\
 & \overset{lm}{\Rightarrow} & \underline{T} + T + T \\
 & \overset{lm}{\Rightarrow} & \underline{F} + T + T \\
 & \overset{lm}{\Rightarrow} & id + \underline{T} + T \\
 & \overset{lm}{\Rightarrow} & id + \underline{F} + T \\
 & \overset{lm}{\Rightarrow} & id + id + \underline{T} \\
 & \overset{lm}{\Rightarrow} & id + id + \underline{F}
\end{array}
$$

Parse Trees

The commonality of the two derivations is expressed as a parse tree.



Leftmost derivation:

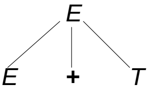
$$
\begin{aligned}
\underline{E} &\stackrel{lm}{\Rightarrow} \underline{E} + T \\
&\stackrel{lm}{\Rightarrow} \underline{E} + T + T \\
&\stackrel{lm}{\Rightarrow} \underline{T} + T + T \\
&\stackrel{lm}{\Rightarrow} \underline{F} + T + T \\
&\stackrel{lm}{\Rightarrow} id + \underline{T} + T \\
&\stackrel{lm}{\Rightarrow} id + \underline{F} + T \\
&\stackrel{lm}{\Rightarrow} id + id + \underline{T} \\
&\stackrel{lm}{\Rightarrow} id + id + \underline{F} \\
&\stackrel{lm}{\Rightarrow} id + id + id
\end{aligned}
$$

Parse Trees

The commonality of the two derivations is expressed as a parse tree.

Parse Trees

The commonality of the two derivations is expressed as a parse tree.

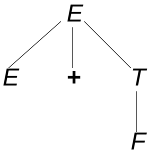


Rightmost derivation:

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

Parse Trees

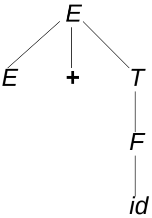
The commonality of the two derivations is expressed as a parse tree.



Rightmost derivation:

$$
\begin{array}{lcl}
\underline{E} & \overset{rm}{\Rightarrow} & E + \underline{T} \\
 & \overset{rm}{\Rightarrow} & E + \underline{F}
\end{array}
$$

The commonality of the two derivations is expressed as a parse tree.

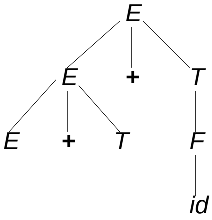


Rightmost derivation:

$$
\begin{array}{lcl}
\underline{E} & \stackrel{rm}{\Rightarrow} & E + \underline{T} \\
 & \stackrel{rm}{\Rightarrow} & E + \underline{F} \\
 & \stackrel{rm}{\Rightarrow} & \underline{E} + id
\end{array}
$$

Parse Trees

The commonality of the two derivations is expressed as a parse tree.

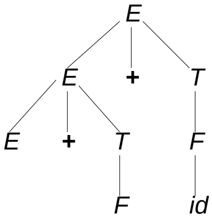


Rightmost derivation:

$$
\begin{array}{lcl}
\underline{E} & \overset{rm}{\Rightarrow} & E + \underline{T} \\
& \overset{rm}{\Rightarrow} & E + \underline{F} \\
& \overset{rm}{\Rightarrow} & \underline{E} + id \\
& \overset{rm}{\Rightarrow} & E + \underline{T} + id
\end{array}
$$

Parse Trees

The commonality of the two derivations is expressed as a parse tree.

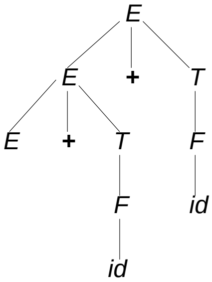


Rightmost derivation:

$$\begin{array}{lcl}
\underline{E} & \overset{rm}{\Rightarrow} & E + \underline{T} \\
 & \overset{rm}{\Rightarrow} & E + \underline{F} \\
 & \overset{rm}{\Rightarrow} & \underline{E} + id \\
 & \overset{rm}{\Rightarrow} & E + \underline{T} + id \\
 & \overset{rm}{\Rightarrow} & E + \underline{F} + id
\end{array}$$

Parse Trees

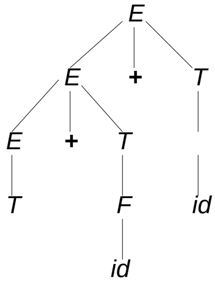
The commonality of the two derivations is expressed as a parse tree.



Rightmost derivation:

$$
\begin{array}{lcl}
\underline{E} & \overset{rm}{\Rightarrow} & E + \underline{T} \\
 & \overset{rm}{\Rightarrow} & E + \underline{F} \\
 & \overset{rm}{\Rightarrow} & \underline{E} + id \\
 & \overset{rm}{\Rightarrow} & E + \underline{T} + id \\
 & \overset{rm}{\Rightarrow} & E + \underline{F} + id \\
 & \overset{rm}{\Rightarrow} & \underline{E} + id + id
\end{array}
$$

The commonality of the two derivations is expressed as a parse tree.

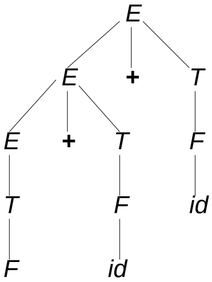


Rightmost derivation:

$$
\begin{array}{lll}
\underline{E} & \overset{rm}{\Rightarrow} & E + \underline{T} \\
 & \overset{rm}{\Rightarrow} & E + \underline{F} \\
 & \overset{rm}{\Rightarrow} & \underline{E} + id \\
 & \overset{rm}{\Rightarrow} & E + \underline{T} + id \\
 & \overset{rm}{\Rightarrow} & E + \underline{F} + id \\
 & \overset{rm}{\Rightarrow} & \underline{E} + id + id \\
 & \overset{rm}{\Rightarrow} & \underline{T} + id + id
\end{array}
$$

Parse Trees

The commonality of the two derivations is expressed as a parse tree.

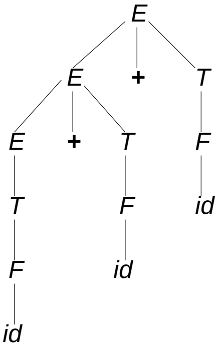


Rightmost derivation:

$$
\begin{array}{lcl}
\underline{E} & \overset{rm}{\Rightarrow} & E + \underline{T} \\
 & \overset{rm}{\Rightarrow} & E + \underline{F} \\
 & \overset{rm}{\Rightarrow} & \underline{E} + id \\
 & \overset{rm}{\Rightarrow} & E + \underline{T} + id \\
 & \overset{rm}{\Rightarrow} & E + \underline{F} + id \\
 & \overset{rm}{\Rightarrow} & \underline{E} + id + id \\
 & \overset{rm}{\Rightarrow} & \underline{T} + id + id \\
 & \overset{rm}{\Rightarrow} & \underline{F} + id + id
\end{array}
$$

Parse Trees

The commonality of the two derivations is expressed as a parse tree.



Rightmost derivation:

$$
\begin{array}{lcl}
\underline{E} & \overset{rm}{\Rightarrow} & E + \underline{T} \\
 & \overset{rm}{\Rightarrow} & E + \underline{F} \\
 & \overset{rm}{\Rightarrow} & \underline{E} + id \\
 & \overset{rm}{\Rightarrow} & E + \underline{T} + id \\
 & \overset{rm}{\Rightarrow} & E + \underline{F} + id \\
 & \overset{rm}{\Rightarrow} & \underline{E} + id + id \\
 & \overset{rm}{\Rightarrow} & \underline{T} + id + id \\
 & \overset{rm}{\Rightarrow} & \underline{F} + id + id \\
 & \overset{rm}{\Rightarrow} & id + id + id
\end{array}
$$

Parse Trees

A *parse tree* is a pictorial form of depicting a derivation.

1. root of the tree is labeled with S
2. each leaf node is labeled by a token or by ϵ
3. an internal node of the tree is labeled by a nonterminal
4. if an internal node has A as its label and the children of this node from left to right are labeled with $X_1, X_2, \ldots, X_n$ then there must be a production

$$A \rightarrow X_1 X_2 \ldots X_n$$

where X_i is a grammar symbol.

Derivations and Parse Trees

The following summarize some interesting relations between the two concepts

- Parse tree filters out the choice of replacements made in the sentential forms.

Derivations and Parse Trees

The following summarize some interesting relations between the two concepts

- Parse tree filters out the choice of replacements made in the sentential forms.
- Given a left (right) derivation for a sentence, one can construct a unique parse tree for the sentence.

Derivations and Parse Trees

The following summarize some interesting relations between the two concepts

- Parse tree filters out the choice of replacements made in the sentential forms.
- Given a left (right) derivation for a sentence, one can construct a unique parse tree for the sentence.
- For every parse tree for a sentence there is a unique leftmost and a unique rightmost derivation.

Derivations and Parse Trees

The following summarize some interesting relations between the two concepts

- Parse tree filters out the choice of replacements made in the sentential forms.
- Given a left (right) derivation for a sentence, one can construct a unique parse tree for the sentence.
- For every parse tree for a sentence there is a unique leftmost and a unique rightmost derivation.
- *Can a sentence have more than one distinct parse trees, and therefore more than one left (right) derivations?*

Ambiguous Grammars

Consider the grammar:

$$E \rightarrow E + E \;\mid\; E * E \;\mid\; id$$

And consider the sentence:

$$id + id * id$$

Ambiguous Grammars

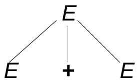
Consider the grammar:

$$E \rightarrow E + E \;\mid\; E * E \;\mid\; id$$

And consider the sentence:

$$id + id * id$$

Parse tree 1:



Leftmost derivation 1:

$$\underline{E} \overset{lm}{\Rightarrow} \underline{E} + E$$

Ambiguous Grammars

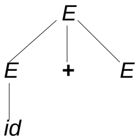
Consider the grammar:

$$E \rightarrow E + E \;\mid\; E * E \;\mid\; id$$

And consider the sentence:

$$id + id * id$$

Parse tree 1:



Leftmost derivation 1:

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

Ambiguous Grammars

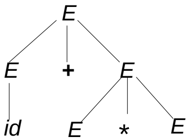
Consider the grammar:

$$E \rightarrow E + E \;\mid\; E * E \;\mid\; id$$

And consider the sentence:

$$id + id * id$$

Parse tree 1:



Leftmost derivation 1:

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

Ambiguous Grammars

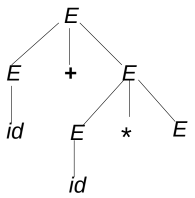
Consider the grammar:

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

And consider the sentence:

$$id + id * id$$

Parse tree 1:



Leftmost derivation 1:

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

Ambiguous Grammars

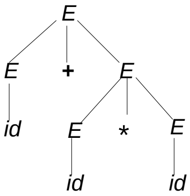
Consider the grammar:

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

And consider the sentence:

$$id + id * id$$

Parse tree 1:



Leftmost derivation 1:

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

Ambiguous Grammars

Consider the grammar:

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

And consider the sentence:

$$id + id * id$$

Ambiguous Grammars

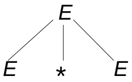
Consider the grammar:

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

And consider the sentence:

$$id + id * id$$

Parse tree 2:



Leftmost derivation

$$\underline{E} \overset{lm}{\Rightarrow} \underline{E} * E$$

2:

Ambiguous Grammars

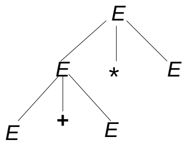
Consider the grammar:

$$E \rightarrow E + E \;\mid\; E * E \;\mid\; id$$

And consider the sentence:

$$id + id * id$$

Parse tree 2:



Leftmost derivation 2:

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

Ambiguous Grammars

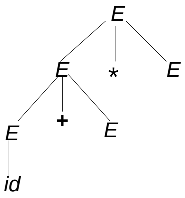
Consider the grammar:

$$E \rightarrow E + E \;\mid\; E * E \;\mid\; id$$

And consider the sentence:

$$id + id * id$$

Parse tree 2:



Leftmost derivation

$$\begin{aligned} \underline{E} &\overset{lm}{\Rightarrow} \underline{E} * E \\ &\overset{lm}{\Rightarrow} \underline{E} + E * E \\ \textit{2:} \quad &\overset{lm}{\Rightarrow} id + \underline{E} * E \end{aligned}$$

Ambiguous Grammars

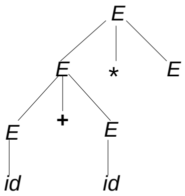
Consider the grammar:

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

And consider the sentence:

$$id + id * id$$

Parse tree 2:



Leftmost derivation

$$\begin{array}{lcl} \underline{E} & \overset{lm}{\Rightarrow} & \underline{E} * E \\ & \overset{lm}{\Rightarrow} & \underline{E} + E * E \\ \textit{2:} & \overset{lm}{\Rightarrow} & id + \underline{E} * E \\ & \overset{lm}{\Rightarrow} & id + id * \underline{E} \end{array}$$

Ambiguous Grammars

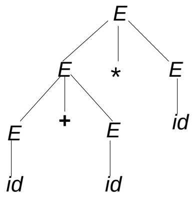
Consider the grammar:

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

And consider the sentence:

$$id + id * id$$

Parse tree 2:



Leftmost derivation

$$\begin{array}{lcl} \underline{E} & \overset{lm}{\Rightarrow} & \underline{E} * E \\ & \overset{lm}{\Rightarrow} & \underline{E} + E * E \\ \textit{2:} & \overset{lm}{\Rightarrow} & id + \underline{E} * E \\ & \overset{lm}{\Rightarrow} & id + id * \underline{E} \\ & \overset{lm}{\Rightarrow} & id + id * id \end{array}$$

There are two parse trees and two leftmostderivations for the sentence.

Ambiguous Grammars

A grammar is *ambiguous*, if there is a sentence for which there are

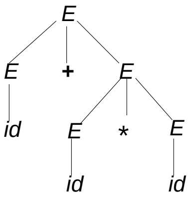
- more than one parse tress, or equivalently
- more than one leftmost derivations, or equivalently
- more than one rightmost derivations.

Ambiguous Grammars

Why is ambiguity an issue?

For the expression grammar, the parse tree represent an implicit parenthesizing of the sentence.

Parse tree 1:



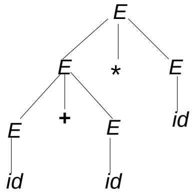
$\Longrightarrow id + (id * id)$

Ambiguous Grammars

Why is ambiguity an issue?

For the expression grammar, the parse tree represent an implicit parenthesizing of the sentence.

Parse tree 2:



$\Longrightarrow (id + id) * id$

And the meanings of the expressions $id + (id * id)$ and $(id + id) * id$ are not the same.

Ambiguous Grammars – A second example

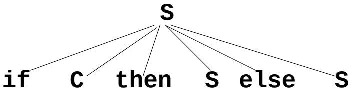
Example:

$$
\begin{array}{lcl}
S & \rightarrow & \texttt{if } C \texttt{ then } S \texttt{ else } S \\
S & \rightarrow & \texttt{if } C \texttt{ then } S \\
S & \rightarrow & \texttt{ass}
\end{array}
$$

Consider the sentence:

`if` C `then if` C `then ass else ass`

First parse tree:



First rightmost derivation:

$$S \rightarrow \texttt{if } C \texttt{ then } S \texttt{ else } \underline{S}$$

Ambiguous Grammars – A second example

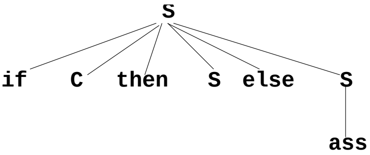
Example:

$$
\begin{array}{lcl}
S & \rightarrow & \texttt{if}\ C\ \texttt{then}\ S\ \texttt{else}\ S \\
S & \rightarrow & \texttt{if}\ C\ \texttt{then}\ S \\
S & \rightarrow & \texttt{ass}
\end{array}
$$

Consider the sentence:

`if` C `then if` C `then ass else ass`

First parse tree:



First rightmost derivation:

$$
\begin{array}{lcl}
S & \rightarrow & \texttt{if}\ C\ \texttt{then}\ S\ \texttt{else}\ \underline{S} \\
S & \rightarrow & \texttt{if}\ C\ \texttt{then}\ \underline{S}\ \texttt{else ass}
\end{array}
$$

Ambiguous Grammars – A second example

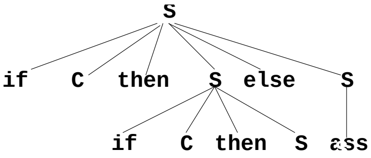
Example:

$$
\begin{array}{lcl}
S & \rightarrow & \texttt{if}\ C\ \texttt{then}\ S\ \texttt{else}\ S \\
S & \rightarrow & \texttt{if}\ C\ \texttt{then}\ S \\
S & \rightarrow & \texttt{ass}
\end{array}
$$

Consider the sentence:

$\texttt{if}\ C\ \texttt{then}\ \texttt{if}\ C\ \texttt{then}\ \texttt{ass}\ \texttt{else}\ \texttt{ass}$

First parse tree:



First rightmost derivation:

$$
\begin{array}{lcl}
S & \rightarrow & \texttt{if}\ C\ \texttt{then}\ S\ \texttt{else}\ \underline{S} \\
S & \rightarrow & \texttt{if}\ C\ \texttt{then}\ \underline{S}\ \texttt{else}\ \texttt{ass} \\
S & \rightarrow & \texttt{if}\ C\ \texttt{then}\ \texttt{if}\ \ C\ \ \texttt{then}\ \ \underline{S}\ \texttt{else}\ \texttt{ass}
\end{array}
$$

Ambiguous Grammars – A second example

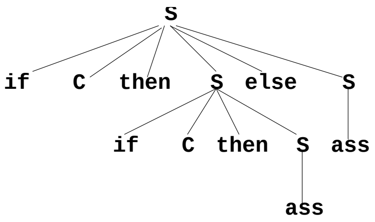
Example:

$$\begin{array}{lll} S & \rightarrow & \texttt{if } C \texttt{ then } S \texttt{ else } S \\ S & \rightarrow & \texttt{if } C \texttt{ then } S \\ S & \rightarrow & \texttt{ass} \end{array}$$

Consider the sentence:

`if` C `then if` C `then ass else ass`

First parse tree:



First rightmost derivation:

$$\begin{array}{lll} S & \rightarrow & \texttt{if } C \texttt{ then } S \texttt{ else } \underline{S} \\ S & \rightarrow & \texttt{if } C \texttt{ then } \underline{S} \texttt{ else ass} \\ S & \rightarrow & \texttt{if } C \texttt{ then if } C \texttt{ then } \underline{S} \texttt{ else ass} \\ S & \rightarrow & \texttt{if } C \texttt{ then if } C \texttt{ then ass else ass} \end{array}$$

Ambiguous Grammars

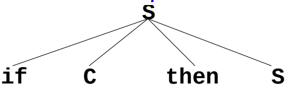
Example:

$$\begin{array}{lcl} S & \rightarrow & \texttt{if}\ C\ \texttt{then}\ S\ \texttt{else}\ S \\ S & \rightarrow & \texttt{if}\ C\ \texttt{then}\ S \\ S & \rightarrow & \texttt{ass} \end{array}$$

Consider the sentence:

`if` C `then if C then ass else ass`

The second parse tree:



The second rightmost derivation:

$$S \rightarrow \texttt{if}\ C\ \texttt{then}\ \underline{S}$$

Ambiguous Grammars

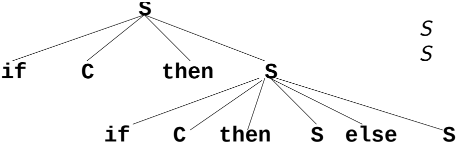
Example:

$$
\begin{array}{lcl}
S & \rightarrow & \texttt{if } C \texttt{ then } S \texttt{ else } S \\
S & \rightarrow & \texttt{if } C \texttt{ then } S \\
S & \rightarrow & \texttt{ass}
\end{array}
$$

Consider the sentence:

`if` C `then if C then ass else ass`

The second parse tree:



The second rightmost derivation:

$$
\begin{array}{lcl}
S & \rightarrow & \texttt{if } C \texttt{ then } \underline{S} \\
S & \rightarrow & \texttt{if } C \texttt{ then if } C \texttt{ then } \underline{S} \texttt{ else } S
\end{array}
$$

Ambiguous Grammars

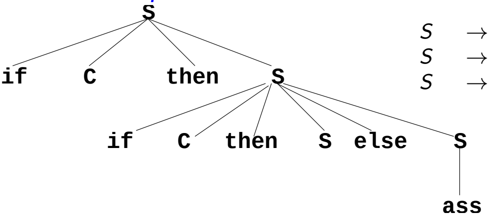
Example:

$$
\begin{array}{lcl}
S & \rightarrow & \texttt{if } C \texttt{ then } S \texttt{ else } S \\
S & \rightarrow & \texttt{if } C \texttt{ then } S \\
S & \rightarrow & \texttt{ass}
\end{array}
$$

Consider the sentence:

`if` C `then if C then ass else ass`

The second parse tree:



The second rightmost derivation:

$$
\begin{array}{lcl}
S & \rightarrow & \texttt{if } C \texttt{ then } \underline{S} \\
S & \rightarrow & \texttt{if } C \texttt{ then if } C \texttt{ then } \underline{S} \texttt{ else } S \\
S & \rightarrow & \texttt{if } C \texttt{ then if } C \texttt{ then } \underline{S} \texttt{ else ass}
\end{array}
$$

Ambiguous Grammars

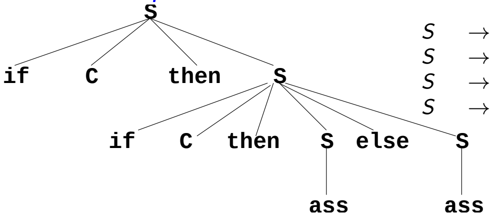
Example:

$$
\begin{array}{lcl}
S & \rightarrow & \texttt{if } C \texttt{ then } S \texttt{ else } S \\
S & \rightarrow & \texttt{if } C \texttt{ then } S \\
S & \rightarrow & \texttt{ass}
\end{array}
$$

Consider the sentence:

`if` C `then if C then ass else ass`

The second parse tree:



The second rightmost derivation:

$$
\begin{array}{lcl}
S & \rightarrow & \texttt{if } C \texttt{ then } \underline{S} \\
S & \rightarrow & \texttt{if } C \texttt{ then if } C \texttt{ then } \underline{S} \texttt{ else } S \\
S & \rightarrow & \texttt{if } C \texttt{ then if } C \texttt{ then } \underline{S} \texttt{ else ass} \\
S & \rightarrow & \texttt{if } C \texttt{ then if } C \texttt{ then ass else ass}
\end{array}
$$

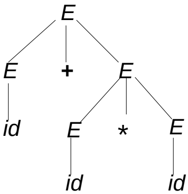
Disambiguation

How does one disambiguate to obtain a single parse tree for a sentence?

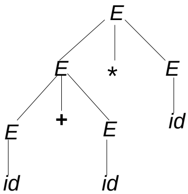
- *Disambiguate during parsing:* Disambiguation rules are incorporated into a parser to choose between possible parse trees.
  - Makes a choice during parse tree construction.
  - Yacc has provisions for such disambiguation.
- *Disambiguate the grammar:* Rewrite the grammar.

Disambiguation by grammar rewriting

- Decide on general rules to choose one of many possible parse trees. As example, choose



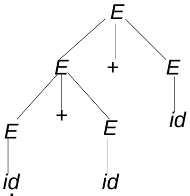
over



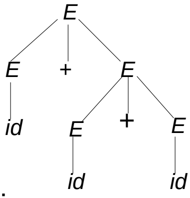
This amounts to giving a higher precedence to $*$ over $+$.

Disambiguation by grammar rewriting

- Similarly, choose:



over



This amounts to saying that + is left associative.

Disambiguation by grammar rewriting

- Consider a sentence $a + b * c * d + d * e$. Denote as T the sub-expressions consisting of products of *id*s or a single *id*.

  Then the expression can be re-written as $T + T + T$

- Because $+$ is left associative, the expression above should be parsed as $(T + T) + T$.

  A grammar which does this is:
  $E \rightarrow E + T \mid T$

Disambiguation by grammar rewriting

- Let F denote either a single *id* or a (E). Then the strings represented by T can be written as $F * F * F$ or a single F.
- A grammar which generates such strings, taking into account the associativity of $*$ is:
  $T \rightarrow T * F \mid F$
- Finally we also have
  $F \rightarrow (E) \mid id$

Disambiguation by grammar rewriting

- Now consider disambiguation of the grammar:

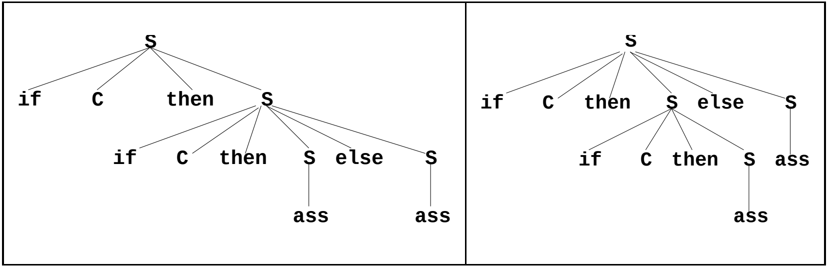
$$\begin{array}{lcl} S & \rightarrow & \texttt{if } C \texttt{ then } S \texttt{ else } S \\ S & \rightarrow & \texttt{if } C \texttt{ then } S \\ S & \rightarrow & \texttt{ass} \end{array}$$

  and the sentence

  `if` C `then if C then ass else ass`

Disambiguation by grammar rewriting

- The parse trees are:



- We choose the first parse tree over the second on the basis of the following rule:

  Every else *should be matched with its closest unmatched* then.

Disambiguation

In other words:
If a then and an else are derived from the same production, then the parse tree between them should have matching then and else.

The following grammar captures this idea:

$$
\begin{aligned}
\mathit{stmt} &\rightarrow \mathit{matched\_stmt} \mid \mathit{unmatched\_stmt} \\
\mathit{matched\_stmt} &\rightarrow \texttt{if } C \texttt{ then } \mathit{matched\_stmt} \texttt{ else } \mathit{matched\_stmt} \\
&\mid \texttt{ass} \\
\mathit{unmatched\_stmt} &\rightarrow \texttt{if } C \texttt{ then } \mathit{stmt} \\
&\mid \texttt{if } C \texttt{ then } \mathit{matched\_stmt} \texttt{ else } \mathit{unmatched\_stmt}
\end{aligned}
$$

Introduction to Parsing

A *parser* for a context free grammar G is a program P that given an input w,

- either verifies that w is a sentence of G and, additionally, may also give the parse tree for w.
- or gives an error message stating that w is not a sentence. May provide some information to locate the error.

Parsing Strategies

Two ways of creating a parse tree:

- *Top-down parsers* – Created from the root down to leaves.
- *Bottom-up parsers* – Created from leaves upwards to the root.

Both the parsing strategies can also be rephrased in terms of derivations.

Example of Bottom Up Parsing

Grammar:

$$D \rightarrow \texttt{var}\ \mathit{list} :\ \mathit{type}\ ;$$
$$\mathit{type} \rightarrow \texttt{integer} \mid \texttt{float}$$
$$\mathit{list} \rightarrow \mathit{list}, \texttt{id} \mid \texttt{id}$$

Input string: `var id,id :  integer;`

Example of Bottom Up Parsing

Grammar:

$$D \rightarrow \texttt{var}\ \mathit{list} : \mathit{type}\ ;$$
$$\mathit{type} \rightarrow \texttt{integer} \mid \texttt{float}$$
$$\mathit{list} \rightarrow \mathit{list}, \texttt{id} \mid \texttt{id}$$

Input string: `var id,id :  integer;`

- The bottom-up parse and the sentential forms produced during the parse are:

Example of Bottom Up Parsing

Grammar:

$$D \rightarrow \texttt{var}\ \textit{list} : \textit{type}\ ;$$
$$\textit{type} \rightarrow \texttt{integer} \mid \texttt{float}$$
$$\textit{list} \rightarrow \textit{list}, \texttt{id} \mid \texttt{id}$$

Input string: `var id,id :  integer;`

- The bottom-up parse and the sentential forms produced during the parse are:

  `var id, id :  integer ;`

Example of Bottom Up Parsing

Grammar:

$$
\begin{aligned}
D &\rightarrow \texttt{var}\ \mathit{list} : \mathit{type}\ ; \\
\mathit{type} &\rightarrow \texttt{integer} \mid \texttt{float} \\
\mathit{list} &\rightarrow \mathit{list}, \texttt{id} \mid \texttt{id}
\end{aligned}
$$

Input string: `var id,id :  integer;`

- The bottom-up parse and the sentential forms produced during the parse are:

$$
\begin{aligned}
& \texttt{var}\ \texttt{id},\ \texttt{id}\ :\ \texttt{integer}\ ; \\
\Rightarrow\ & \texttt{var}\ \mathit{list}\ ,\ \texttt{id}\ :\ \texttt{integer}\ ;
\end{aligned}
$$

Example of Bottom Up Parsing

Grammar:

$$D \rightarrow \texttt{var}\ list : type\ ;$$
$$type \rightarrow \texttt{integer} \mid \texttt{float}$$
$$list \rightarrow list,\ \texttt{id} \mid \texttt{id}$$

Input string: `var id,id :  integer;`

- The bottom-up parse and the sentential forms produced during the parse are:

$$
\begin{array}{ll}
 & \texttt{var id, id : integer ;} \\
\Rightarrow & \texttt{var}\ list\ \texttt{, id : integer ;} \\
\Rightarrow & \texttt{var}\ list : \texttt{integer;}
\end{array}
$$

Example of Bottom Up Parsing

Grammar:

$$D \rightarrow \texttt{var}\ \textit{list} :\ \textit{type}\ ;$$
$$\textit{type} \rightarrow \texttt{integer} \mid \texttt{float}$$
$$\textit{list} \rightarrow \textit{list}, \texttt{id} \mid \texttt{id}$$

Input string: `var id,id : integer;`

- The bottom-up parse and the sentential forms produced during the parse are:

$$\begin{array}{rl} & \texttt{var id, id : integer ;} \\ \Rightarrow & \texttt{var}\ \textit{list}\ \texttt{, id : integer ;} \\ \Rightarrow & \texttt{var}\ \textit{list} :\ \texttt{integer;} \\ \Rightarrow & \texttt{var}\ \textit{list} :\ \textit{type}\ ; \end{array}$$

Example of Bottom Up Parsing

Grammar:

$$D \rightarrow \texttt{var}\ list : type\ ;$$
$$type \rightarrow \texttt{integer} \mid \texttt{float}$$
$$list \rightarrow list, \texttt{id} \mid \texttt{id}$$

Input string: `var id,id :  integer;`

- The bottom-up parse and the sentential forms produced during the parse are:

$$
\begin{array}{ll}
 & \texttt{var id, id : integer ;} \\
\Rightarrow & \texttt{var}\ list\ \texttt{, id : integer ;} \\
\Rightarrow & \texttt{var}\ list : \texttt{integer;} \\
\Rightarrow & \texttt{var}\ list : type\ ; \\
\Rightarrow & D
\end{array}
$$

Example of Bottom Up Parsing

Grammar:

$$D \rightarrow \texttt{var}\ \mathit{list} : \mathit{type}\ ;$$
$$\mathit{type} \rightarrow \texttt{integer} \mid \texttt{float}$$
$$\mathit{list} \rightarrow \mathit{list}, \texttt{id} \mid \texttt{id}$$

Input string: `var id,id :  integer;`

- The bottom-up parse and the sentential forms produced during the parse are:

$$\begin{array}{ll} & \texttt{var}\ \texttt{id},\ \texttt{id}\ :\ \texttt{integer}\ ; \\ \Rightarrow & \texttt{var}\ \mathit{list}\ ,\ \texttt{id}\ :\ \texttt{integer}\ ; \\ \Rightarrow & \texttt{var}\ \mathit{list} : \texttt{integer;} \\ \Rightarrow & \texttt{var}\ \mathit{list} : \mathit{type}\ ; \\ \Rightarrow & D \end{array}$$

- The sentential forms happen to be a *right most derivation in the reverse order.*

Example of Bottom Up Parsing

Here is bottom up parsing, viewed in terms of parse tree construction:

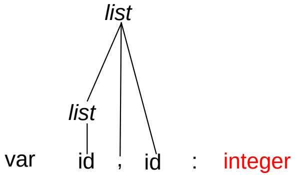
var id , id : integer

Example of Bottom Up Parsing

Here is bottom up parsing, viewed in terms of parse tree construction:

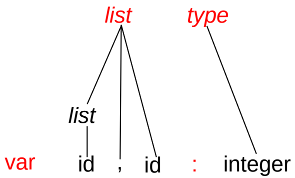
Example of Bottom Up Parsing

Here is bottom up parsing, viewed in terms of parse tree construction:



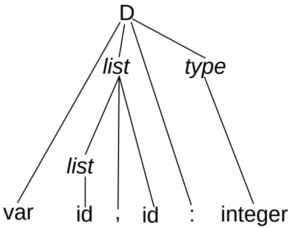
Example of Bottom Up Parsing

Here is bottom up parsing, viewed in terms of parse tree construction:



Example of Bottom Up Parsing

Here is bottom up parsing, viewed in terms of parse tree construction:



Principles of Bottom Up Parsing - Handles

The basic steps of a bottom-up parser are

1. to identify a *substring* within a *rightmost sentential* form which matches the rhs of a rule.
2. when this substring is replaced by the lhs of the matching rule, it must produce the previous rm-sentential form.

Such a substring is called a *handle* .

Principles of Bottom Up Parsing - Handles

The basic steps of a bottom-up parser are

1. to identify a *substring* within a *rightmost sentential* form which matches the rhs of a rule.
2. when this substring is replaced by the lhs of the matching rule, it must produce the previous rm-sentential form.

Such a substring is called a *handle* .

```
var id , id :  integer ;
```

Principles of Bottom Up Parsing - Handles

The basic steps of a bottom-up parser are

1. to identify a *substring* within a *rightmost sentential* form which matches the rhs of a rule.
2. when this substring is replaced by the lhs of the matching rule, it must produce the previous rm-sentential form.

Such a substring is called a *handle* .

$$\begin{array}{ll} & \texttt{var}\ \underline{\texttt{id}}\ ,\ \texttt{id}\ :\ \texttt{integer}\ ; \\ \Rightarrow & \texttt{var}\ \underline{\textit{list}\ ,\ \texttt{id}}\ :\ \texttt{integer}\ ; \end{array}$$

Principles of Bottom Up Parsing - Handles

The basic steps of a bottom-up parser are

1. to identify a *substring* within a *rightmost sentential* form which matches the rhs of a rule.
2. when this substring is replaced by the lhs of the matching rule, it must produce the previous rm-sentential form.

Such a substring is called a *handle* .

$$
\begin{array}{ll}
 & \texttt{var}\ \underline{\texttt{id}}\ ,\ \texttt{id}\ :\ \ \texttt{integer}\ ; \\
\Rightarrow & \texttt{var}\ \underline{\textit{list}\ ,\ \texttt{id}}\ :\ \ \texttt{integer}\ ; \\
\Rightarrow & \texttt{var}\ \textit{list}\ :\ \ \underline{\texttt{integer}}\ ;
\end{array}
$$

Principles of Bottom Up Parsing - Handles

The basic steps of a bottom-up parser are

1. to identify a *substring* within a *rightmost sentential* form which matches the rhs of a rule.
2. when this substring is replaced by the lhs of the matching rule, it must produce the previous rm-sentential form.

Such a substring is called a *handle* .

$$
\begin{array}{ll}
 & \texttt{var}\ \underline{\texttt{id}}\ ,\ \texttt{id}\ :\ \texttt{integer}\ ; \\
\Rightarrow & \texttt{var}\ \underline{\textit{list}\ ,\ \texttt{id}}\ :\ \texttt{integer}\ ; \\
\Rightarrow & \texttt{var}\ \textit{list}\ :\ \underline{\texttt{integer}}\ ; \\
\Rightarrow & \underline{\texttt{var}\ \textit{list}\ :\ \textit{type}}\ ;
\end{array}
$$

Principles of Bottom Up Parsing - Handles

The basic steps of a bottom-up parser are

1. to identify a *substring* within a *rightmost sentential* form which matches the rhs of a rule.
2. when this substring is replaced by the lhs of the matching rule, it must produce the previous rm-sentential form.

Such a substring is called a *handle* .

$$
\begin{array}{ll}
 & \texttt{var}\ \underline{\texttt{id}}\ \texttt{,}\ \texttt{id}\ \texttt{:}\ \ \texttt{integer}\ \texttt{;} \\
\Rightarrow & \texttt{var}\ \underline{\mathit{list}\ \texttt{,}\ \texttt{id}}\ \texttt{:}\ \ \texttt{integer}\ \texttt{;} \\
\Rightarrow & \texttt{var}\ \mathit{list}\ \texttt{:}\ \ \underline{\texttt{integer}}\ \texttt{;} \\
\Rightarrow & \underline{\texttt{var}\ \mathit{list} : \mathit{type}}\ ; \\
\Rightarrow & D
\end{array}
$$

Handle – Definition

A *handle* of a right sentential form γ, is

- a production rule $A \rightarrow \beta$, and
- an occurrence of a sub-string β in γ

such that when the occurrence of β is replaced by A in γ, we get the previous right sentential form in a rightmost derivation of γ.

Handle – Definition

A *handle* of a right sentential form γ, is

- a production rule $A \rightarrow \beta$, and
- an occurrence of a sub-string β in γ

such that when the occurrence of β is replaced by A in γ, we get the previous right sentential form in a rightmost derivation of γ.

Formally, if

$$S \overset{*rm}{\Rightarrow} \alpha\, A\, \mathtt{w} \overset{rm}{\Rightarrow} \alpha\, \beta\, \mathtt{w}$$

then the rule $A \rightarrow \beta$ and the occurrence β is the handle in $\alpha\, \beta\, \mathtt{w}$.

Handle – Definition

A *handle* of a right sentential form γ, is

- a production rule $A \rightarrow \beta$, and
- an occurrence of a sub-string β in γ

such that when the occurrence of β is replaced by A in γ, we get the previous right sentential form in a rightmost derivation of γ.

Formally, if

$$S \overset{*rm}{\Rightarrow} \alpha\, A\, \mathtt{w} \overset{rm}{\Rightarrow} \alpha\, \beta\, \mathtt{w}$$

then the rule $A \rightarrow \beta$ and the occurrence β is the handle in $\alpha\, \beta\, \mathtt{w}$.

Only terminal symbols can appear to the right of a handle in a rightmost sentential form. Why?

Handles

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