上下文无关文法

WANG Hanfei

School of Computer
Wuhan University

September 22, 2015

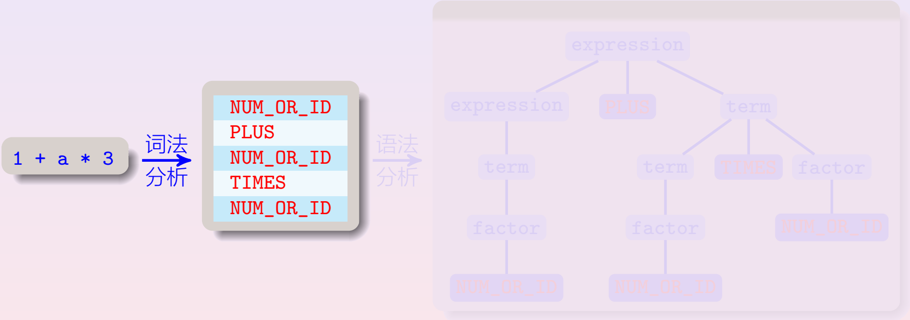
1. 上下文无关文法
   - 递归定义与产生式
   - 上下文无关文法
   - 上下文无关语言
2. 语法树
   - 语法树
   - 语法树与推导的关系
   - 二义性
3. 上下文无关文法的设计
   - 设计举例
   - 正则表达式转换为上下文自由文法
   - 自动机转换为上下文自由文法
   - CFG 的表达能力
4. 二义性的再讨论

XL 语言

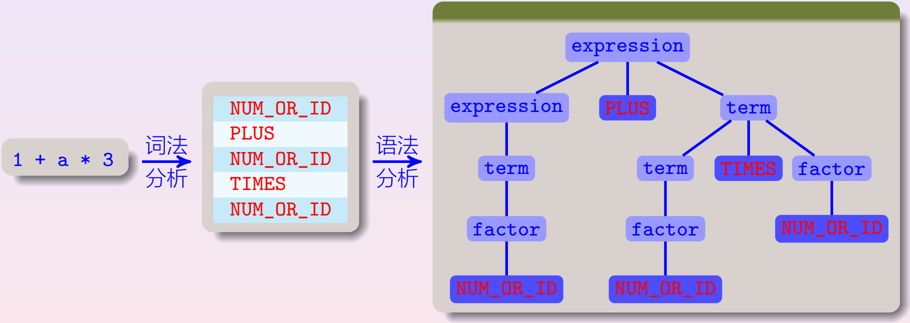
1 + a * 3

词法
分析
NUM_OR_ID
PLUS
NUM_OR_ID
TIMES
NUM_OR_ID
语法
分析
expression
expression
PLUS
term
term
term
TIMES
factor
factor
factor
NUM_OR_ID
NUM_OR_ID
NUM_OR_ID

XL 语言


1 + a * 3
词法
分析
NUM_OR_ID
PLUS
NUM_OR_ID
TIMES
NUM_OR_ID
语法
分析
expression
expression
PLUS
term
term
term
TIMES
factor
factor
factor
NUM_OR_ID
NUM_OR_ID
NUM_OR_ID

XL 语言


1 + a * 3
词法
分析
NUM_OR_ID
PLUS
NUM_OR_ID
TIMES
NUM_OR_ID
语法
分析
expression
expression
PLUS
term
term
term
TIMES
factor
factor
factor
NUM_OR_ID
NUM_OR_ID
NUM_OR_ID
NUM_OR_ID

语法分析 (Parser)

- 形式语言的第二次抽象，对单词再次重组.
  - Ex: $\Sigma = \{\mathtt{NUM\_OR\_ID}, \mathtt{PLUS}, \mathtt{TIMES}, \mathtt{LP}, \mathtt{RP}, \mathtt{SEMI}\}$;
  - $XL \subseteq \Sigma^*$.
- 描述问题：
  - 正则表达式：表达能力太弱，嵌套的括号对不能描述.
  - 数学递归定义：不利于机器对它进行识别.
- 识别问题: $\exists P\ \Sigma^* \longrightarrow \text{Boolean}, s \mapsto \text{true or false}$.
- 语法树的建立.

语法分析 (Parser)

- 形式语言的第二次抽象，对单词再次重组.
  - Ex: $\Sigma = \{\mathrm{NUM\_OR\_ID}, \mathrm{PLUS}, \mathrm{TIMES}, \mathrm{LP}, \mathrm{RP}, \mathrm{SEMI}\}$;
  - $XL \subseteq \Sigma^*$.
- 描述问题：
- 识别问题: $\exists P\ \Sigma^* \longrightarrow \text{Boolean}, s \mapsto \text{true or false}$.
- 语法树的建立.

语法分析 (Parser)

- 形式语言的第二次抽象，对单词再次重组.
  - Ex: Σ = {NUM_OR_ID, PLUS, TIMES, LP, RP, SEMI};
  - $XL \subseteq \Sigma^*$.
- 描述问题：
  - 正则表达式：表达能力太弱，嵌套的括号对不能描述.
  - 数学递归定义：不利于机器对文法的识别.
- 识别问题: $\exists P\ \Sigma^* \longrightarrow$ Boolean, $s \mapsto$ true or false.
- 语法树的建立.

语法分析 (Parser)

- 形式语言的第二次抽象，对单词再次重组.
  - Ex: $\Sigma = \{$NUM_OR_ID, PLUS, TIMES, LP, RP, SEMI$\}$;
  - $XL \subseteq \Sigma^*$.
- 描述问题：
  - 正则表达式：表达能力太弱，嵌套的括号对不能描述.
  - 数学递归定义：不利于机器对文法的识别.
- 识别问题: $\exists P\ \Sigma^* \longrightarrow$ Boolean, $s \mapsto$ true or false.
- 语法树的建立.

语法分析 (Parser)

- 形式语言的第二次抽象，对单词再次重组.
  - Ex: $\Sigma = \{\mathtt{NUM\_OR\_ID}, \mathtt{PLUS}, \mathtt{TIMES}, \mathtt{LP}, \mathtt{RP}, \mathtt{SEMI}\}$;
  - $XL \subseteq \Sigma^*$.
- 描述问题：
  - 正则表达式：表达能力太弱，嵌套的括号对不能描述.
  - 数学递归定义：不利于机器对文法的识别.
- 识别问题: $\exists P\ \Sigma^* \longrightarrow \text{Boolean},\ s \mapsto \text{true or false}$.
- 语法树的建立.

语法分析 (Parser)

- 形式语言的第二次抽象，对单词再次重组.
  - Ex: Σ = {NUM_OR_ID, PLUS, TIMES, LP, RP, SEMI};
  - $XL \subseteq \Sigma^*$.
- 描述问题：
  - 正则表达式：表达能力太弱，嵌套的括号对不能描述.
  - 数学递归定义：不利于机器对文法的识别.
- 识别问题: $\exists P\ \Sigma^* \longrightarrow$ Boolean, $s \mapsto$ true or false.
- 语法树的建立.

语法分析 (Parser)

- 形式语言的第二次抽象，对单词再次重组.
  - Ex: Σ = {NUM_OR_ID, PLUS, TIMES, LP, RP, SEMI};
  - $XL \subseteq \Sigma^*$.
- 描述问题：
  - 正则表达式：表达能力太弱，嵌套的括号对不能描述.
  - 数学递归定义：不利于机器对文法的识别.
- 识别问题: $\exists P\ \Sigma^* \longrightarrow$ Boolean, $s \mapsto$ true or false.
- 语法树的建立.

语法分析 (Parser)

- 形式语言的第二次抽象，对单词再次重组。
  - Ex: $\Sigma = \{\mathtt{NUM\_OR\_ID}, \mathtt{PLUS}, \mathtt{TIMES}, \mathtt{LP}, \mathtt{RP}, \mathtt{SEMI}\}$;
  - $XL \subseteq \Sigma^*$.
- 描述问题：
  - 正则表达式：表达能力太弱，嵌套的括号对不能描述。
  - 数学递归定义：不利于机器对文法的识别。
- 识别问题: $\exists P\, \Sigma^* \longrightarrow \text{Boolean},\ s \mapsto \text{true or false}$.
- 语法树的建立。

语法分析 (Parser)

- 形式语言的第二次抽象，对单词再次重组.
  - Ex: $\Sigma = \{$NUM_OR_ID, PLUS, TIMES, LP, RP, SEMI$\}$;
  - $XL \subseteq \Sigma^*$.
- 描述问题：
  - 正则表达式：表达能力太弱，嵌套的括号对不能描述.
  - 数学递归定义：不利于机器对文法的识别.
- 识别问题: $\exists P\ \Sigma^* \longrightarrow$ Boolean, $s \mapsto$ true or false.
- 语法树的建立.

Example: XL 语言的递归定义

设：$\Sigma = \{$NUM_OR_ID, PLUS, TIMES, LP, RP$\}$, factor, term, expression $\subseteq \Sigma^*$递归定义如下：

1. 归纳基础：NUM_OR_ID $\in$ factor.
2. 归纳条款：$S_1 \in$ expression, then LP S_1 RP $\in$ factor.
   if $S_1 \in$ factor, $S_2 \in$ term, then S_1, S_2 TIMES $S_1 \in$ term.
   if $S_1 \in$ term, $S_2 \in$ expression, then S_1, S_2 PLUS $S_1 \in$ expression.
   其中：S_1和 S_2称为递归定义中的元变量 (metavariable).
3. 由以上规则在有限步生成的字符串构成factor、term和expression集合，称为语言．每个语言中的元素称为语句.

XL 语言的语句 --- 归纳过程

NUM_OR_ID PLUS NUM_OR_ID 是 expression的语句：

Example：XL 语言的递归定义.

设：Σ = {NUM_OR_ID, PLUS, TIMES, LP, RP}，factor，term，expression $\subseteq \Sigma^*$递归定义如下：

1. 归纳基础：NUM_OR_ID $\in$ factor.
2. 归纳条款：$S_1 \in$ expression，then LP S_1 RP $\in$ factor.
   if $S_1 \in$ factor，$S_2 \in$ term，then S_1，S_2 TIMES $S_1 \in$ term.
   if $S_1 \in$ term，$S_2 \in$ expression，then S_1，S_2 PLUS $S_1 \in$ expression.
   其中：S_1和 S_2称为递归定义中的元变量 (metavariable).
3. 由以上规则在有限步生成的字符串构成factor、term和expression集合，称为语言. 每个语言中的元素称为语句.

XL 语言的语句 --- 归纳过程

NUM_OR_ID PLUS NUM_OR_ID 是 expression的语句：

Example: XL 语言的递归定义

设：$\Sigma = \{$NUM_OR_ID, PLUS, TIMES, LP, RP$\}$，factor，term，expression $\subseteq \Sigma^*$递归定义如下：

1. 归纳基础：NUM_OR_ID $\in$ factor.
2. 归纳条款：$S_1 \in$ expression，then LP S_1 RP $\in$ factor.
   if $S_1 \in$ factor，$S_2 \in$ term，then S_1，S_2 TIMES $S_1 \in$ term.
   if $S_1 \in$ term，$S_2 \in$ expression，then S_1，S_2 PLUS $S_1 \in$ expression.
   其中：S_1和 S_2称为递归定义中的元变量 (metavariable).
3. 由以上规则在有限步生成的字符串构成factor、term和expression集合，称为语言。每个语言中的元素称为语句。

XL 语言的语句 --- 归纳过程

NUM_OR_ID PLUS NUM_OR_ID 是 expression的语句：

- NUM_OR_ID 是 factor 的语句，由①；
- NUM_OR_ID 代入归纳条款②中的 S_1 和 S_2 得：NUM_OR_ID PLUS NUM_OR_ID 也是 term 和 expression 的语句.
- 由②，NUM_OR_ID PLUS NUM_OR_ID 是 expression 语言的语句.

Example: XL 语言的递归定义

设：$\Sigma = \{$NUM_OR_ID, PLUS, TIMES, LP, RP$\}$，*factor*，*term*，*expression* $\subseteq \Sigma^*$递归定义如下：

1. 归纳基础：NUM_OR_ID $\in$ *factor*.
2. 归纳条款：$S_1 \in$ *expression*，then LP S_1 RP $\in$ *factor*.
   if $S_1 \in$ *factor*，$S_2 \in$ *term*，then S_1，S_2 TIMES $S_1 \in$ *term*.
   if $S_1 \in$ *term*，$S_2 \in$ *expression*，then S_1，S_2 PLUS $S_1 \in$ *expression*.
   其中：S_1和 S_2称为递归定义中的元变量 (metavariable).
3. 由以上规则在有限步生成的字符串构成*factor*、*term*和*expression*集合，称为语言. 每个语言中的元素称为语句.

XL 语言的语句 --- 归纳过程

NUM_OR_ID PLUS NUM_OR_ID 是 expression的语句：

Example: XL 语言的递归定义

设：Σ = {NUM_OR_ID, PLUS, TIMES, LP, RP}，factor，term，expression $\subseteq \Sigma^*$递归定义如下：

1. 归纳基础：NUM_OR_ID $\in$ factor.
2. 归纳条款：$S_1 \in$ expression，then LP S_1 RP $\in$ factor.
   if $S_1 \in$ factor，$S_2 \in$ term，then S_1，S_2 TIMES $S_1 \in$ term.
   if $S_1 \in$ term，$S_2 \in$ expression，then S_1，S_2 PLUS $S_1 \in$ expression.
   其中：S_1和 S_2称为递归定义中的元变量 (metavariable).
3. 由以上规则在有限步生成的字符串构成factor、term和expression集合，称为语言. 每个语言中的元素称为语句.

XL 语言的语句 --- 归纳过程

NUM_OR_ID PLUS NUM_OR_ID 是 expression的语句：

- NUM_OR_ID 是 factor 的语句，由①.
- NUM_OR_ID 代入到条款②中的 S_1 和 S_2 得：NUM_OR_ID PLUS NUM_OR_ID 也是 term 和 expression 的语句.
- 由②，NUM_OR_ID PLUS NUM_OR_ID 是 expression 语言的语句.

Example: XL 语言的递归定义

设：$\Sigma = \{$NUM_OR_ID, PLUS, TIMES, LP, RP$\}$，factor，term，expression $\subseteq \Sigma^*$递归定义如下：

1. 归纳基础：NUM_OR_ID $\in$ factor.
2. 归纳条款：$S_1 \in$ expression，then LP S_1 RP $\in$ factor.
   if $S_1 \in$ factor，$S_2 \in$ term，then S_1，S_2 TIMES $S_1 \in$ term.
   if $S_1 \in$ term，$S_2 \in$ expression，then S_1，S_2 PLUS $S_1 \in$ expression.
   其中：S_1和 S_2称为递归定义中的元变量 (metavariable).
3. 由以上规则在有限步生成的字符串构成factor、term和expression集合，称为语言. 每个语言中的元素称为语句.

XL 语言的语句 --- 归纳过程

NUM_OR_ID PLUS NUM_OR_ID 是 expression的语句：

- NUM_OR_ID 是 factor 的语句，由①.
- NUM_OR_ID 代入到条款②中的 S_1 和 S_2 得：NUM_OR_ID PLUS NUM_OR_ID 也是 term 和 expression 的语句.
- 由②，NUM_OR_ID PLUS NUM_OR_ID 是 expression 语言的语句.

Example: XL 语言的递归定义

设：Σ = {NUM_OR_ID, PLUS, TIMES, LP, RP}，factor，term，expression $\subseteq \Sigma^*$递归定义如下：

1. 归纳基础：NUM_OR_ID $\in$ factor.
2. 归纳条款：$S_1 \in$ expression，then LP S_1 RP $\in$ factor.
   if $S_1 \in$ factor，$S_2 \in$ term，then S_1，S_2 TIMES $S_1 \in$ term.
   if $S_1 \in$ term，$S_2 \in$ expression，then S_1，S_2 PLUS $S_1 \in$ expression.
   其中：S_1和 S_2称为递归定义中的元变量 (metavariable).
3. 由以上规则在有限步生成的字符串构成factor、term和expression集合，称为语言. 每个语言中的元素称为语句.

XL 语言的语句 --- 归纳过程

NUM_OR_ID PLUS NUM_OR_ID 是 expression的语句：

- NUM_OR_ID 是 factor 的语句，由①.
- NUM_OR_ID 代入到条款②中的 S_1 和 S_2 得：NUM_OR_ID PLUS NUM_OR_ID 也是 term 和 expression 的语句.
- 由②，NUM_OR_ID PLUS NUM_OR_ID 是 expression 语言的语句.

Example: XL 语言的递归定义

设：$\Sigma = \{$NUM_OR_ID, PLUS, TIMES, LP, RP$\}$, factor, term, expression $\subseteq \Sigma^*$递归定义如下：

1. 归纳基础：NUM_OR_ID $\in$ factor.
2. 归纳条款：$S_1 \in$ expression, then LP S_1 RP $\in$ factor.
   if $S_1 \in$ factor, $S_2 \in$ term, then S_1, S_2 TIMES $S_1 \in$ term.
   if $S_1 \in$ term, $S_2 \in$ expression, then S_1, S_2 PLUS $S_1 \in$ expression.
   其中：S_1和 S_2称为递归定义中的元变量 (metavariable).
3. 由以上规则在有限步生成的字符串构成factor、term和expression集合，称为语言. 每个语言中的元素称为语句.

XL 语言的语句 --- 归纳过程

NUM_OR_ID PLUS NUM_OR_ID 是 expression的语句：

- NUM_OR_ID 是 factor 的语句，由①.
- NUM_OR_ID 代入到条款②中的 S_1 和 S_2 得：NUM_OR_ID PLUS NUM_OR_ID 也是 term 和 expression 的语句.
- 由②，NUM_OR_ID PLUS NUM_OR_ID 是 expression 语言的语句.

形式化语言的生成过程

归纳过程

$$\underbrace{\underbrace{\underbrace{\underbrace{\text{NUM\_OR\_ID}}_{\text{factor}}}_{\text{term}}}_{\text{expression}}\ \text{PLUS}\ \underbrace{\underbrace{\underbrace{\underbrace{\text{NUM\_OR\_ID}}_{\text{factor}}}_{\text{term}}\ \text{TIMES}\ \underbrace{\text{NUM\_OR\_ID}}_{\text{factor}}}_{\text{term}}}_{\text{expression}}$$

演绎过程

$$\overbrace{\overbrace{\overbrace{\overbrace{\text{NUM\_OR\_ID}}^{\text{factor}}}^{\text{term}}}^{\text{expression}}\ \text{PLUS}\ \overbrace{\overbrace{\overbrace{\overbrace{\text{NUM\_OR\_ID}}^{\text{factor}}}^{\text{term}}\ \text{TIMES}\ \overbrace{\text{NUM\_OR\_ID}}^{\text{factor}}}^{\text{term}}}^{\text{expression}}$$

形式化语言的生成过程

归纳过程

$$\underbrace{\underbrace{\underbrace{\underbrace{\texttt{NUM\_OR\_ID}}_{\text{factor}}}_{\text{term}}}_{\text{expression}}\ \texttt{PLUS}\ \underbrace{\underbrace{\underbrace{\texttt{NUM\_OR\_ID}}_{\text{factor}}}_{\text{term}}\ \texttt{TIMES}\ \underbrace{\texttt{NUM\_OR\_ID}}_{\text{factor}}}_{\text{term}}}_{\text{expression}}$$

演绎过程

$$\overbrace{\overbrace{\overbrace{\overbrace{\texttt{NUM\_OR\_ID}}^{\text{factor}}}^{\text{term}}}^{\text{expression}}\ \texttt{PLUS}\ \overbrace{\overbrace{\overbrace{\texttt{NUM\_OR\_ID}}^{\text{factor}}}^{\text{term}}\ \texttt{TIMES}\ \overbrace{\texttt{NUM\_OR\_ID}}^{\text{factor}}}^{\text{term}}}^{\text{expression}}$$

Example: XL 语言的递归定义

XL 语言的递归定义

设：Σ = {NUM_OR_ID, PLUS, TIMES, LP, RP}，*factor*，*term*，*expression* $\subseteq \Sigma^*$ 递归定义如下：

1. 归纳基础：NUM_OR_ID $\in$ *factor*.
2. 归纳条款：$S_1 \in$ *expression*，then LP S_1 RP $\in$ *factor*.
   if $S_1 \in$ *factor*，$S_2 \in$ *term*，then S_1，S_2 TIMES $S_1 \in$ *term*.
   if $S_1 \in$ *term*，$S_2 \in$ *expression*，then S_1，S_2 PLUS $S_1 \in$ *expression*.

重写规则 --- 产生式

Example: XL 语言的递归定义

XL 语言的递归定义

设: $\Sigma = \{\mathtt{NUM\_OR\_ID}, \mathtt{PLUS}, \mathtt{TIMES}, \mathtt{LP}, \mathtt{RP}\}$, *factor*, *term*, *expression* $\subseteq \Sigma^*$ 递归定义如下:

1. 归纳基础: $\mathtt{NUM\_OR\_ID} \in$ *factor*.
2. 归纳条款: $S_1 \in$ *expression*, then $\mathtt{LP}\ S_1\ \mathtt{RP} \in$ *factor*.
   if $S_1 \in$ *factor*, $S_2 \in$ *term*, then S_1, $S_2\ \mathtt{TIMES}\ S_1 \in$ *term*.
   if $S_1 \in$ *term*, $S_2 \in$ *expression*, then S_1, $S_2\ \mathtt{PLUS}\ S_1 \in$ *expression*.

重写规则 --- 产生式

1. expression $\longrightarrow$ expression PLUS term.
2. expression $\longrightarrow$ term.
3. term $\longrightarrow$ term TIMES factor.
4. term $\longrightarrow$ factor.
5. factor $\longrightarrow$ NUM_OR_ID.
6. factor $\longrightarrow$ LP expression RP.

Example：XL 语言的递归定义

XL 语言的递归定义

设：$\Sigma = \{$NUM_OR_ID, PLUS, TIMES, LP, RP$\}$，*factor*，*term*，*expression* $\subseteq \Sigma^*$ 递归定义如下：

1. 归纳基础：NUM_OR_ID $\in$ *factor*.
2. 归纳条款：$S_1 \in$ *expression*， then LP S_1 RP $\in$ *factor*.
   if $S_1 \in$ *factor*， $S_2 \in$ *term*， then S_1， S_2 TIMES $S_1 \in$ *term*.
   if $S_1 \in$ *term*， $S_2 \in$ *expression*， then S_1， S_2 PLUS $S_1 \in$ *expression*.

重写规则 --- 产生式

- expression $\longrightarrow$ expression PLUS term.
- expression $\longrightarrow$ term.
- term $\longrightarrow$ term TIMES factor.
- term $\longrightarrow$ factor.
- factor $\longrightarrow$ NUM_OR_ID.
- factor $\longrightarrow$ LP expression RP.

Example: XL 语言的递归定义

XL 语言的递归定义

设：Σ = {NUM_OR_ID, PLUS, TIMES, LP, RP}，factor，term，expression $\subseteq \Sigma^*$ 递归定义如下：

1. 归纳基础：NUM_OR_ID $\in$ factor.
2. 归纳条款：$S_1 \in$ expression，then LP S_1 RP $\in$ factor.
   if $S_1 \in$ factor，$S_2 \in$ term，then S_1，S_2 TIMES $S_1 \in$ term.
   if $S_1 \in$ term，$S_2 \in$ expression，then S_1，S_2 PLUS $S_1 \in$ expression.

元变量用被定义集合名替换！

重写规则 --- 产生式

- expression $\longrightarrow$ expression PLUS term.
- expression $\longrightarrow$ term.
- term $\longrightarrow$ term TIMES factor.
- term $\longrightarrow$ factor;
- factor $\longrightarrow$ NUM_OR_ID .
- factor $\longrightarrow$ LP expression RP.

Example: XL 语言的递归定义

XL 语言的递归定义

设：Σ = {NUM_OR_ID, PLUS, TIMES, LP, RP}，factor，term，expression $\subseteq \Sigma^*$ 递归定义如下：

1. 归纳基础：NUM_OR_ID $\in$ factor.
2. 归纳条款：$S_1 \in$ expression，then LP S_1 RP $\in$ factor.
   if $S_1 \in$ factor，$S_2 \in$ term，then S_1，S_2 TIMES $S_1 \in$ term.
   if $S_1 \in$ term，$S_2 \in$ expression，then S_1，S_2 PLUS $S_1 \in$ expression.

元变量用被定义集合名替换！

重写规则 --- 产生式

- expression $\longrightarrow$ expression PLUS term.
- expression $\longrightarrow$ term.
- term $\longrightarrow$ term TIMES factor.
- term $\longrightarrow$ factor;
- factor $\longrightarrow$ NUM_OR_ID .
- factor $\longrightarrow$ LP expression RP.

Example: XL 语言的递归定义

XL 语言的递归定义

设：Σ = {NUM_OR_ID, PLUS, TIMES, LP, RP}，factor，term，expression $\subseteq \Sigma^*$ 递归定义如下：

1. 归纳基础：NUM_OR_ID $\in$ factor.
2. 归纳条款：$S_1 \in$ expression，then LP S_1 RP $\in$ factor.
   if $S_1 \in$ factor，$S_2 \in$ term，then S_1，S_2 TIMES $S_1 \in$ term.
   if $S_1 \in$ term，$S_2 \in$ expression，then S_1，S_2 PLUS $S_1 \in$ expression.

元变量用被定义集合名替换！

重写规则 --- 产生式

- expression $\longrightarrow$ expression PLUS term.
- expression $\longrightarrow$ term.
- term $\longrightarrow$ term TIMES factor.
- term $\longrightarrow$ factor;
- factor $\longrightarrow$ NUM_OR_ID .
- factor $\longrightarrow$ LP expression RP.

Example: XL 语言的递归定义

XL 语言的递归定义

设：$\Sigma = \{$NUM_OR_ID, PLUS, TIMES, LP, RP$\}$, factor, term, expression $\subseteq \Sigma^*$ 递归定义如下：

1. 归纳基础：NUM_OR_ID $\in$ factor.
2. 归纳条款：$S_1 \in$ expression, then LP S_1 RP $\in$ factor.
   if $S_1 \in$ factor, $S_2 \in$ term, then S_1, S_2 TIMES $S_1 \in$ term.
   if $S_1 \in$ term, $S_2 \in$ expression, then S_1, S_2 PLUS $S_1 \in$ expression.

元变量用被定义集合名替换！

重写规则 --- 产生式

- expression $\longrightarrow$ expression PLUS term.
- expression $\longrightarrow$ term.
- term $\longrightarrow$ term TIMES factor.
- term $\longrightarrow$ factor;
- factor $\longrightarrow$ NUM_OR_ID .
- factor $\longrightarrow$ LP expression RP.

Example: XL 语言的递归定义

XL 语言的递归定义

设: $\Sigma = \{$NUM_OR_ID, PLUS, TIMES, LP, RP$\}$, *factor*, *term*, *expression* $\subseteq \Sigma^*$ 递归定义如下:

1. 归纳基础: NUM_OR_ID $\in$ *factor*.
2. 归纳条款: $S_1 \in$ *expression*, then LP S_1 RP $\in$ *factor*.
   if $S_1 \in$ *factor*, $S_2 \in$ *term*, then S_1, S_2 TIMES $S_1 \in$ *term*.
   if $S_1 \in$ *term*, $S_2 \in$ *expression*, then S_1, S_2 PLUS $S_1 \in$ *expression*.

元变量用被定义集合名替换!

重写规则 --- 产生式

- *expression* $\longrightarrow$ *expression* PLUS *term*.
- *expression* $\longrightarrow$ *term*.
- *term* $\longrightarrow$ *term* TIMES *factor*.
- *term* $\longrightarrow$ *factor*;
- *factor* $\longrightarrow$ NUM_OR_ID .
- *factor* $\longrightarrow$ LP *expression* RP.

Example: XL 语言的递归定义

XL 语言的递归定义

设：$\Sigma = \{\mathtt{NUM\_OR\_ID}, \mathtt{PLUS}, \mathtt{TIMES}, \mathtt{LP}, \mathtt{RP}\}$，factor，term，expression $\subseteq \Sigma^*$ 递归定义如下：

1. 归纳基础：NUM_OR_ID $\in$ factor.
2. 归纳条款：$S_1 \in$ expression，then LP S_1 RP $\in$ factor.
   if $S_1 \in$ factor，$S_2 \in$ term，then S_1，S_2 TIMES $S_1 \in$ term.
   if $S_1 \in$ term，$S_2 \in$ expression，then S_1，S_2 PLUS $S_1 \in$ expression.

元变量用被定义集合名替换！

重写规则 --- 产生式

- expression $\longrightarrow$ expression PLUS term.
- expression $\longrightarrow$ term.
- term $\longrightarrow$ term TIMES factor.
- term $\longrightarrow$ factor;
- factor $\longrightarrow$ NUM_OR_ID .
- factor $\longrightarrow$ LP expression RP.

Example: XL 语言的递归定义

XL 语言的递归定义

设：Σ = {NUM_OR_ID, PLUS, TIMES, LP, RP}，factor，term，expression $\subseteq \Sigma^*$ 递归定义如下：

1. 归纳基础：NUM_OR_ID $\in$ factor.
2. 归纳条款：$S_1 \in$ expression，then LP S_1 RP $\in$ factor.
   if $S_1 \in$ factor，$S_2 \in$ term，then S_1，S_2 TIMES $S_1 \in$ term.
   if $S_1 \in$ term，$S_2 \in$ expression，then S_1，S_2 PLUS $S_1 \in$ expression.

元变量用被定义集合名替换！

重写规则 --- 产生式

- expression $\longrightarrow$ expression PLUS term.
- expression $\longrightarrow$ term.
- term $\longrightarrow$ term TIMES factor.
- term $\longrightarrow$ factor;
- factor $\longrightarrow$ NUM_OR_ID .
- factor $\longrightarrow$ LP expression RP.

重写 (Rewriting)

expression
⟹ expression PLUS term
⟹ term PLUS term
⟹ factor PLUS term
⟹ NUM_OR_ID PLUS term
⟹ NUM_OR_ID PLUS term TIMES factor
⟹ NUM_OR_ID PLUS
⟹ NUM_OR_ID PLUS NUM_OR_ID TIMES factor
⟹ NUM_OR_ID PLUS NUM_OR_ID TIMES NUM_OR_ID

expression ⟶ expression PLUS term

重写 (Rewriting)

	expression				
$\Longrightarrow$	expression	PLUS	term		
$\Longrightarrow$	term	PLUS	term		
$\Longrightarrow$	factor	PLUS	term		
$\Longrightarrow$	NUM_OR_ID	PLUS	term		
$\Longrightarrow$	NUM_OR_ID	PLUS	term	TIMES	factor
$\Longrightarrow$	NUM_OR_ID	PLUS	factor	TIMES	factor
$\Longrightarrow$	NUM_OR_ID	PLUS	NUM_OR_ID	TIMES	factor
$\Longrightarrow$	NUM_OR_ID	PLUS	NUM_OR_ID	TIMES	NUM_OR_ID

expression $\longrightarrow$ term

重写 (Rewriting)

	expression				
⟹	expression	PLUS	term		
⟹	term	PLUS	term		
⟹	factor	PLUS	term		
⟹	NUM_OR_ID	PLUS	term		
⟹	NUM_OR_ID	PLUS	term	TIMES	factor
⟹	NUM_OR_ID	PLUS	factor	TIMES	factor
⟹	NUM_OR_ID	PLUS	NUM_OR_ID	TIMES	factor
⟹	NUM_OR_ID	PLUS	NUM_OR_ID	TIMES	NUM_OR_ID

$\boxed{\text{term} \longrightarrow \text{factor}}$

重写 (Rewriting)

	expression				
$\Longrightarrow$	expression	PLUS	term		
$\Longrightarrow$	term	PLUS	term		
$\Longrightarrow$	factor	PLUS	term		
$\Longrightarrow$	NUM_OR_ID	PLUS	term		
$\Longrightarrow$	NUM_OR_ID	PLUS	term	TIMES	factor
$\Longrightarrow$	NUM_OR_ID	PLUS	factor	TIMES	factor
$\Longrightarrow$	NUM_OR_ID	PLUS	NUM_OR_ID	TIMES	factor
$\Longrightarrow$	NUM_OR_ID	PLUS	NUM_OR_ID	TIMES	NUM_OR_ID

factor $\longrightarrow$ ID

重写 (Rewriting)

term $\longrightarrow$ term TIMES factor

	expression				
$\Longrightarrow$	expression	PLUS	term		
$\Longrightarrow$	term	PLUS	term		
$\Longrightarrow$	factor	PLUS	term		
$\Longrightarrow$	NUM_OR_ID	PLUS	term		
$\Longrightarrow$	NUM_OR_ID	PLUS	term	TIMES	factor
$\Longrightarrow$	NUM_OR_ID	PLUS	factor	TIMES	factor
$\Longrightarrow$	NUM_OR_ID	PLUS	NUM_OR_ID	TIMES	factor
$\Longrightarrow$	NUM_OR_ID	PLUS	NUM_OR_ID	TIMES	NUM_OR_ID

重写 (Rewriting)

$\text{term} \longrightarrow \text{factor}$

	expression				
$\Longrightarrow$	expression	PLUS	term		
$\Longrightarrow$	term	PLUS	term		
$\Longrightarrow$	factor	PLUS	term		
$\Longrightarrow$	NUM_OR_ID	PLUS	term		
$\Longrightarrow$	NUM_OR_ID	PLUS	term	TIMES	factor
$\Longrightarrow$	NUM_OR_ID	PLUS	factor	TIMES	factor
$\Longrightarrow$	NUM_OR_ID	PLUS	NUM_OR_ID	TIMES	factor
$\Longrightarrow$	NUM_OR_ID	PLUS	NUM_OR_ID	TIMES	NUM_OR_ID

重写 (Rewriting)

factor $\longrightarrow$ ID

	expression				
$\Longrightarrow$	expression	PLUS	term		
$\Longrightarrow$	term	PLUS	term		
$\Longrightarrow$	factor	PLUS	term		
$\Longrightarrow$	NUM_OR_ID	PLUS	term		
$\Longrightarrow$	NUM_OR_ID	PLUS	term	TIMES	factor
$\Longrightarrow$	NUM_OR_ID	PLUS	factor	TIMES	factor
$\Longrightarrow$	NUM_OR_ID	PLUS	NUM_OR_ID	TIMES	factor
$\Longrightarrow$	NUM_OR_ID	PLUS	NUM_OR_ID	TIMES	NUM_OR_ID

重写 (Rewriting)

factor ⟶ ID

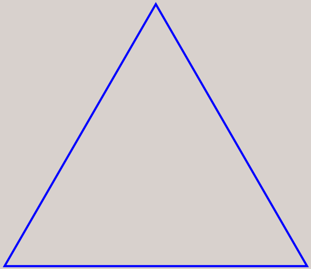
	expression				
⟹	expression	PLUS	term		
⟹	term	PLUS	term		
⟹	factor	PLUS	term		
⟹	NUM_OR_ID	PLUS	term		
⟹	NUM_OR_ID	PLUS	term	TIMES	factor
⟹	NUM_OR_ID	PLUS	factor	TIMES	factor
⟹	NUM_OR_ID	PLUS	NUM_OR_ID	TIMES	factor
⟹	NUM_OR_ID	PLUS	NUM_OR_ID	TIMES	NUM_OR_ID

重写 (Rewriting)

	expression				
$\Longrightarrow$	expression	PLUS	term		
$\Longrightarrow$	term	PLUS	term		
$\Longrightarrow$	factor	PLUS	term		
$\Longrightarrow$	NUM_OR_ID	PLUS	term		
$\Longrightarrow$	NUM_OR_ID	PLUS	term	TIMES	factor
$\Longrightarrow$	NUM_OR_ID	PLUS	factor	TIMES	factor
$\Longrightarrow$	NUM_OR_ID	PLUS	NUM_OR_ID	TIMES	factor
$\Longrightarrow$	NUM_OR_ID	PLUS	NUM_OR_ID	TIMES	NUM_OR_ID

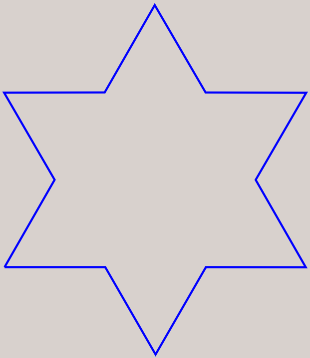
Rewriting Example: L-system (Lindenmayer)

Koch snowflake



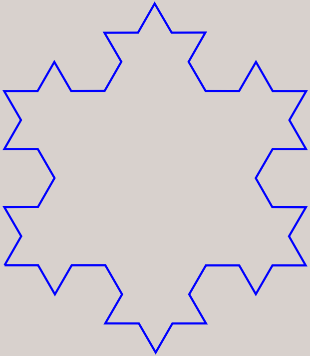
Rewriting Example: L-system (Lindenmayer)

Koch snowflake



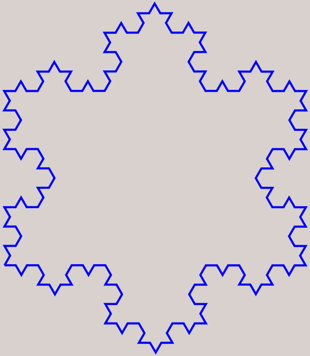
Rewriting Example: L-system (Lindenmayer)

Koch snowflake



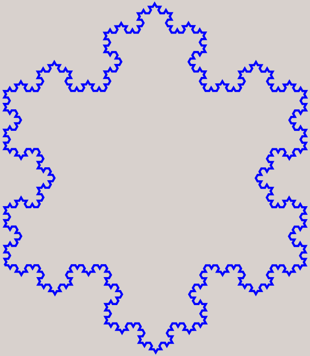
Rewriting Example: L-system (Lindenmayer)

Koch snowflake



Rewriting Example: L-system (Lindenmayer)

Koch snowflake



上下文无关文法 (Context-Free Grammar, CFG)

CFG 定义

$G = \langle T, N, S, P \rangle$，其中：

- T 是有限字母表，其元素称为终结符号 (terminals);
- N 是有限语法成分标记集合，$N \cap T = \emptyset$，称为非终结符号 (non-terminals)，相当于递归定义的元变量;
- $S \in N$ 称为文法开始符号;
- $P = \{S_i \longrightarrow w \mid S_i \in N, w \in (T \cup N)^*\}$ 是一个有限的集合，其元素称为产生式.

Example

```
T = { NUM_OR_ID,  PLUS,
      TIMES,  LP,  RP };
N = {expression, term, factor};
S = expression;
P:
1/ expression -> expression PLUS term
2/               |  term

3/ term -> term TIMES factor
4/         |  factor

5/ factor -> NUM_OR_ID
6/           |  LP expression RP
```

注释

为了书写方便，在后续的讲义中，*expression* 简记为 *exp*，*factor* 简记为 *fac*，NUM_OR_ID 简记为ID.

注释

- Backus-Naur Form (BNF, 源于 Agcol60).
- Terminal = 字母; Non-terminal = 递归定义中的 metavariable.
- 产生式左边 (Left Hand Side, LHS)一定是唯一的一个非终结符.
- 产生式右边(Right Hand Side, RHS)可以是空串，称为空产生式.
- 一个非终结符可以有多个产生式对应，可用 ``|'' 连接其对应的 RHS.

注释

- Backus-Naur Form (BNF, 源于 Agcol60).
- Terminal = 字母; Non-terminal = 递归定义中的 metavariable.
- 产生式左边 (Left Hand Side, LHS)一定是唯一的一个非终结符.
- 产生式右边(Right Hand Side, RHS)可以是空串，称为空产生式.
- 一个非终结符可以有多个产生式对应，可用 ``|'' 连接其对应的 RHS.

注释

- Backus-Naur Form (BNF, 源于 Agcol60).
- Terminal = 字母; Non-terminal = 递归定义中的 metavariable.
- 产生式左边 (Left Hand Side, LHS)一定是唯一的一个非终结符.
- 产生式右边(Right Hand Side, RHS)可以是空串，称为空产生式.
- 一个非终结符可以有多个产生式对应, 可用 ``|'' 连接其对应的 RHS.

注释

- Backus-Naur Form (BNF, 源于 Agcol60).
- Terminal = 字母; Non-terminal = 递归定义中的 metavariable.
- 产生式左边 (Left Hand Side, LHS)一定是唯一的一个非终结符.
- 产生式右边(Right Hand Side, RHS)可以是空串，称为空产生式.
- 一个非终结符可以有多个产生式对应, 可用 ``|'' 连接其对应的 RHS.

注释

- Backus-Naur Form (BNF, 源于 Agcol60).
- Terminal = 字母; Non-terminal = 递归定义中的 metavariable.
- 产生式左边 (Left Hand Side, LHS)一定是唯一的一个非终结符.
- 产生式右边(Right Hand Side, RHS)可以是空串，称为空产生式.
- 一个非终结符可以有多个产生式对应，可用 ``|'' 连接其对应的 RHS.

注释

- Backus-Naur Form (BNF, 源于 Agcol60).
- Terminal = 字母; Non-terminal = 递归定义中的 metavariable.
- 产生式左边 (Left Hand Side, LHS)一定是唯一的一个非终结符.
- 产生式右边(Right Hand Side, RHS)可以是空串， 称为空产生式.
- 一个非终结符可以有多个产生式对应, 可用 \`\`|'' 连接其对应的 RHS.

CFG 所描述的语言 --- 推导

Example of XL

```
T = { ID,  PLUS,
      TIMES,  LP,  RP };
N = {exp, term, fac};
S = exp;
P:
1/ exp -> exp PLUS term
2/      |  term

3/ term -> term TIMES fac
4/       |  fac

5/ fac -> ID
6/      |  LP exp  RP
```

推导过程

	exp	
⇒	exp PLUS term	by 1/
⇒	term PLUS term	by 2/
⇒	fac PLUS term	by 4/
⇒	ID PLUS term	by 5/
⇒	ID PLUS term TIMES fac	by 3/
⇒	ID PLUS fac TIMES fac	by 4/
⇒	ID PLUS ID TIMES fac	by 5/
⇒	ID PLUS ID TIMES ID	by 5/

推导

1. 设 G 是 CFG, $\alpha A \beta \in (T \cup N)^*$, $A \in N$, $A \longrightarrow \gamma \in P$, 则
$$\alpha A \beta \Rightarrow \alpha \gamma \beta$$
是一步推导 (one step derivation)，α 和 β 称推导的上下文环境.
2. $\Rightarrow \subseteq (T \cup N)^* \times (T \cup N)^*$ 是 $(T \cup N)^*$ 上的二元关系.

Example

推导

1. 设 G 是 CFG, $\alpha A \beta \in (T \cup N)^*$, $A \in N$, $A \longrightarrow \gamma \in P$, 则
$$\alpha A \beta \Rightarrow \alpha \gamma \beta$$
是一步推导 (one step derivation)，α 和 β 称推导的上下文环境.
2. $\Rightarrow \subseteq (T \cup N)^* \times (T \cup N)^*$ 是 $(T \cup N)^*$ 上的二元关系.

Example

推导

1. 设 G 是 CFG, $\alpha A\beta \in (T \cup N)^*$, $A \in N$, $A \longrightarrow \gamma \in P$, 则
$$\alpha A\beta \Rightarrow \alpha\gamma\beta$$
是一步推导 (one step derivation), α 和 β 称推导的上下文环境.
2. $\Rightarrow \subseteq (T \cup N)^* \times (T \cup N)^*$ 是 $(T \cup N)^*$ 上的二元关系.

Example

推导

1. 设 G 是 CFG, $\alpha A\beta \in (T\cup N)^*$, $A\in N$, $A\longrightarrow\gamma\in P$, 则
$$\alpha A\beta \Rightarrow \alpha\gamma\beta$$
是一步推导 (one step derivation)，α 和 β 称推导的上下文环境.
2. $\Rightarrow\subseteq (T\cup N)^*\times(T\cup N)^*$ 是 $(T\cup N)^*$ 上的二元关系.

Example

1. $\underbrace{\text{ID PLUS}}_{\alpha}\ \underline{\text{term}}\ \underbrace{\text{TIMES fac}}_{\beta} \Rightarrow \underbrace{\text{ID PLUS}}_{\alpha}\ \underline{\text{fac}}\ \underbrace{\text{TIMES fac}}_{\beta}.$
2. $\underbrace{\text{PLUS}}_{\alpha'}\ \underline{\text{term}}\ \underbrace{\text{TIMES fac}}_{\beta'} \Rightarrow \underbrace{\text{PLUS}}_{\alpha'}\ \underline{\text{fac}}\ \underbrace{\text{TIMES fac}}_{\beta'}.$

推导

1. 设 G 是 CFG, $\alpha A \beta \in (T \cup N)^*$, $A \in N$, $A \longrightarrow \gamma \in P$, 则
$$\alpha A \beta \Rightarrow \alpha \gamma \beta$$
是一步推导 (one step derivation), α 和 β 称推导的上下文环境.
2. $\Rightarrow \subseteq (T \cup N)^* \times (T \cup N)^*$ 是 $(T \cup N)^*$ 上的二元关系.

Example

1. $\underbrace{\mathrm{ID\ PLUS}}_{\alpha}\ \underline{\mathit{term}}\ \underbrace{\mathrm{TIMES}\ \mathit{fac}}_{\beta} \Rightarrow \underbrace{\mathrm{ID\ PLUS}}_{\alpha}\ \underline{\mathit{fac}}\ \underbrace{\mathrm{TIMES}\ \mathit{fac}}_{\beta}$.
2. $\underbrace{\mathrm{PLUS}}_{\alpha'}\ \underline{\mathit{term}}\ \underbrace{\mathrm{TIMES}\ \mathit{fac}}_{\beta'} \Rightarrow \underbrace{\mathrm{PLUS}}_{\alpha'}\ \underline{\mathit{fac}}\ \underbrace{\mathrm{TIMES}\ \mathit{fac}}_{\beta'}$.

推导

1. 设 G 是 CFG, $\alpha A\beta \in (T \cup N)^*$, $A \in N$, $A \longrightarrow \gamma \in P$, 则
$$\alpha A\beta \Rightarrow \alpha\gamma\beta$$
是一步推导 (one step derivation)，α 和 β 称推导的上下文环境.
2. $\Rightarrow \subseteq (T \cup N)^* \times (T \cup N)^*$ 是 $(T \cup N)^*$ 上的二元关系.

Example

1. $\underbrace{\text{ID PLUS}}_{\alpha}\ \underline{\text{term}}\ \underbrace{\text{TIMES fac}}_{\beta} \Rightarrow \underbrace{\text{ID PLUS}}_{\alpha}\ \underline{\text{fac}}\ \underbrace{\text{TIMES fac}}_{\beta}$.
2. $\underbrace{\text{PLUS}}_{\alpha'}\ \underline{\text{term}}\ \underbrace{\text{TIMES fac}}_{\beta'} \Rightarrow \underbrace{\text{PLUS}}_{\alpha'}\ \underline{\text{fac}}\ \underbrace{\text{TIMES fac}}_{\beta'}$.

n 步推导

1. 0 步推导 ($\overset{0}{\Rightarrow}$):
$$\alpha A\beta \overset{0}{\Rightarrow} \alpha A\beta$$
$\overset{0}{\Rightarrow}$ 是 $(T\cup N)^*$ 上的恒等关系.
2. n 步推导 ($\overset{n}{\Rightarrow}$): 设 $\delta \overset{n}{\Rightarrow} \alpha A\beta$ 是 n 步推导，则
$$\delta \overset{n}{\Rightarrow} \alpha A\beta \Rightarrow \alpha\gamma\beta \triangleq \delta \overset{n+1}{\Rightarrow} \alpha\gamma\beta$$
为 $n+1$ 步推导. $\overset{n}{\Rightarrow}$ 是关系 $\Rightarrow$ 的 n 次合成.
3. $\overset{*}{\Rightarrow}$:
$$\overset{*}{\Rightarrow} \triangleq \bigcup_{i=0}^{\infty} \overset{i}{\Rightarrow}$$

Example

n 步推导

1. 0 步推导 ($\overset{0}{\Rightarrow}$):
$$\alpha A\beta \overset{0}{\Rightarrow} \alpha A\beta$$
$\overset{0}{\Rightarrow}$是 $(T \cup N)^*$ 上的恒等关系.

2. n 步推导 ($\overset{n}{\Rightarrow}$): 设 $\delta \overset{n}{\Rightarrow} \alpha A\beta$ 是 n 步推导，则
$$\delta \overset{n}{\Rightarrow} \alpha A\beta \Rightarrow \alpha\gamma\beta \triangleq \delta \overset{n+1}{\Rightarrow} \alpha\gamma\beta$$
为 $n+1$ 步推导. $\overset{n}{\Rightarrow}$是关系 $\Rightarrow$的 n次合成.

3. $\overset{*}{\Rightarrow}$:
$$\overset{*}{\Rightarrow} \triangleq \bigcup_{i=0}^{\infty} \overset{i}{\Rightarrow}$$

Example

n 步推导

1. 0 步推导 ($\overset{0}{\Rightarrow}$):

$$\alpha A \beta \overset{0}{\Rightarrow} \alpha A \beta$$

$\overset{0}{\Rightarrow}$是 $(T \cup N)^*$ 上的恒等关系.

2. n 步推导 ($\overset{n}{\Rightarrow}$): 设 $\delta \overset{n}{\Rightarrow} \alpha A \beta$ 是 n 步推导，则

$$\delta \overset{n}{\Rightarrow} \alpha A \beta \Rightarrow \alpha \gamma \beta \triangleq \delta \overset{n+1}{\Rightarrow} \alpha \gamma \beta$$

为 $n+1$ 步推导. $\overset{n}{\Rightarrow}$是关系 $\Rightarrow$的 n次合成.

3. $\overset{*}{\Rightarrow}$:

$$\overset{*}{\Rightarrow} \triangleq \bigcup_{i=0}^{\infty} \overset{i}{\Rightarrow}$$

Example

n 步推导

1. 0 步推导 ($\overset{0}{\Rightarrow}$):
$$\alpha A\beta \overset{0}{\Rightarrow} \alpha A\beta$$
$\overset{0}{\Rightarrow}$是 $(T\cup N)^*$ 上的恒等关系.
2. n 步推导 ($\overset{n}{\Rightarrow}$): 设 $\delta \overset{n}{\Rightarrow} \alpha A\beta$ 是 n 步推导，则
$$\delta \overset{n}{\Rightarrow} \alpha A\beta \Rightarrow \alpha\gamma\beta \triangleq \delta \overset{n+1}{\Rightarrow} \alpha\gamma\beta$$
为 $n+1$ 步推导. $\overset{n}{\Rightarrow}$是关系 $\Rightarrow$的 n次合成.
3. $\overset{*}{\Rightarrow}$:
$$\overset{*}{\Rightarrow} \triangleq \bigcup_{i=0}^{\infty} \overset{i}{\Rightarrow}$$

Example

n 步推导

1. 0 步推导 ($\overset{0}{\Rightarrow}$):
$$\alpha A\beta \overset{0}{\Rightarrow} \alpha A\beta$$
$\overset{0}{\Rightarrow}$ 是 $(T \cup N)^*$ 上的恒等关系.
2. n 步推导 ($\overset{n}{\Rightarrow}$): 设 $\delta \overset{n}{\Rightarrow} \alpha A\beta$ 是 n 步推导，则
$$\delta \overset{n}{\Rightarrow} \alpha A\beta \Rightarrow \alpha\gamma\beta \triangleq \delta \overset{n+1}{\Rightarrow} \alpha\gamma\beta$$
为 $n+1$ 步推导. $\overset{n}{\Rightarrow}$ 是关系 $\Rightarrow$ 的 n 次合成.
3. $\overset{*}{\Rightarrow}$:
$$\overset{*}{\Rightarrow} \triangleq \bigcup_{i=0}^{\infty} \overset{i}{\Rightarrow}$$

Example

1. $exp \overset{*}{\Rightarrow}$ ID PLUS ID TIMES ID.
2. $term\ exp \overset{*}{\Rightarrow} term$ ID PLUS ID TIMES ID.

n 步推导

1. 0 步推导 ($\overset{0}{\Rightarrow}$):
$$\alpha A\beta \overset{0}{\Rightarrow} \alpha A\beta$$
$\overset{0}{\Rightarrow}$是 $(T \cup N)^*$ 上的恒等关系.
2. n 步推导 ($\overset{n}{\Rightarrow}$): 设 $\delta \overset{n}{\Rightarrow} \alpha A\beta$ 是 n 步推导，则
$$\delta \overset{n}{\Rightarrow} \alpha A\beta \Rightarrow \alpha\gamma\beta \triangleq \delta \overset{n+1}{\Rightarrow} \alpha\gamma\beta$$
为 $n+1$ 步推导. $\overset{n}{\Rightarrow}$是关系 $\Rightarrow$的 n次合成.
3. $\overset{*}{\Rightarrow}$:
$$\overset{*}{\Rightarrow} \triangleq \bigcup_{i=0}^{\infty} \overset{i}{\Rightarrow}$$

Example

1. $exp \overset{*}{\Rightarrow}$ ID PLUS ID TIMES ID.
2. $term\ exp \overset{*}{\Rightarrow} term$ ID PLUS ID TIMES ID.

n 步推导

1. 0 步推导 ($\overset{0}{\Rightarrow}$):
$$\alpha A\beta \overset{0}{\Rightarrow} \alpha A\beta$$
$\overset{0}{\Rightarrow}$是 $(T \cup N)^*$ 上的恒等关系.
2. n 步推导 ($\overset{n}{\Rightarrow}$): 设 $\delta \overset{n}{\Rightarrow} \alpha A\beta$ 是 n 步推导，则
$$\delta \overset{n}{\Rightarrow} \alpha A\beta \Rightarrow \alpha\gamma\beta \triangleq \delta \overset{n+1}{\Rightarrow} \alpha\gamma\beta$$
为 $n+1$ 步推导. $\overset{n}{\Rightarrow}$是关系 $\Rightarrow$的 n次合成.
3. $\overset{*}{\Rightarrow}$:
$$\overset{*}{\Rightarrow} \triangleq \bigcup_{i=0}^{\infty} \overset{i}{\Rightarrow}$$

Example

1. exp $\overset{*}{\Rightarrow}$ ID PLUS ID TIMES ID.
2. term exp $\overset{*}{\Rightarrow}$ term ID PLUS ID TIMES ID.

注释

上下文无关的含义

- 设 $A \longrightarrow \gamma$, 则 $\forall \alpha, \beta \in (T \cup N)^*$, $\alpha A \beta \overset{1}{\Rightarrow} \alpha\gamma\beta$.
- 设 $A \overset{*}{\Rightarrow} u$ 则 $\forall \alpha, \beta \in (T \cup N)^*$, $\alpha \underline{A} \beta \overset{*}{\Rightarrow} \alpha u \beta$.
- $\overset{*}{\Rightarrow}$ 是 $(T \cup N)^*$ 上的关系 $\Rightarrow$ 的自反传递闭包.

Example

注释

上下文无关的含义

- 设 $A \longrightarrow \gamma$, 则 $\forall \alpha, \beta \in (T \cup N)^*, \alpha A \beta \xRightarrow{1} \alpha\gamma\beta$.
- 设 $A \xRightarrow{*} u$ 则 $\forall \alpha, \beta \in (T \cup N)^*, \alpha \underline{A} \beta \xRightarrow{*} \alpha u \beta$.
- $\xRightarrow{*}$ 是 $(T \cup N)^*$ 上的关系 $\Rightarrow$ 的自反传递闭包.

Example

注释

上下文无关的含义

- 设 $A \longrightarrow \gamma$, 则 $\forall \alpha, \beta \in (T \cup N)^*$, $\alpha A \beta \stackrel{1}{\Rightarrow} \alpha \gamma \beta$.
- 设 $A \stackrel{*}{\Rightarrow} u$ 则 $\forall \alpha, \beta \in (T \cup N)^*$, $\alpha \underline{A} \beta \stackrel{*}{\Rightarrow} \alpha u \beta$.
- $\stackrel{*}{\Rightarrow}$ 是 $(T \cup N)^*$ 上的关系 $\Rightarrow$ 的自反传递闭包.

Example

注释

上下文无关的含义

- 设 $A \longrightarrow \gamma$, 则 $\forall \alpha, \beta \in (T \cup N)^*, \alpha A \beta \stackrel{1}{\Rightarrow} \alpha \gamma \beta$.
- 设 $A \stackrel{*}{\Rightarrow} u$ 则 $\forall \alpha, \beta \in (T \cup N)^*, \alpha \underline{A} \beta \stackrel{*}{\Rightarrow} \alpha u \beta$.
- $\stackrel{*}{\Rightarrow}$ 是 $(T \cup N)^*$ 上的关系 $\Rightarrow$ 的自反传递闭包.

Example

注释

上下文无关的含义

- 设 $A \longrightarrow \gamma$, 则 $\forall \alpha, \beta \in (T \cup N)^*$, $\alpha A \beta \overset{1}{\Rightarrow} \alpha\gamma\beta$.
- 设 $A \overset{*}{\Rightarrow} u$ 则 $\forall \alpha, \beta \in (T \cup N)^*$, $\alpha \underline{A} \beta \overset{*}{\Rightarrow} \alpha u \beta$.
- $\overset{*}{\Rightarrow}$ 是 $(T \cup N)^*$ 上的关系 $\Rightarrow$ 的自反传递闭包.

Example

- ID PLUS term TIMES fac $\Rightarrow$ ID PLUS fac TIMES fac
  ID TIMES term TIMES fac $\Rightarrow$ ID TIMES fac TIMES fac.
- 因为 exp $\overset{*}{\Rightarrow}$ ID PLUS ID TIMES ID, 则:
  term PLUS exp $\overset{*}{\Rightarrow}$ term PLUS ID PLUS ID TIMES ID
  产生式如同公理, 推导结果如同定理.

注释

上下文无关的含义

- 设 $A \longrightarrow \gamma$, 则 $\forall \alpha, \beta \in (T \cup N)^*, \alpha A \beta \overset{1}{\Rightarrow} \alpha\gamma\beta$.
- 设 $A \overset{*}{\Rightarrow} u$ 则 $\forall \alpha, \beta \in (T \cup N)^*, \alpha \underline{A} \beta \overset{*}{\Rightarrow} \alpha u \beta$.
- $\overset{*}{\Rightarrow}$ 是 $(T \cup N)^*$ 上的关系 $\Rightarrow$ 的自反传递闭包.

Example

- ID PLUS $\underline{\text{term}}$ TIMES fac $\Rightarrow$ ID PLUS fac TIMES fac
  ID TIMES $\underline{\text{term}}$ TIMES fac $\Rightarrow$ ID TIMES fac TIMES fac.
- 因为 exp $\overset{*}{\Rightarrow}$ ID PLUS ID TIMES ID, 则:
  term PLUS $\underline{\text{exp}}$ $\overset{*}{\Rightarrow}$ term PLUS ID PLUS ID TIMES ID
  产生式如同公理, 推导结果如同定理.

注释

上下文无关的含义

- 设 $A \longrightarrow \gamma$, 则 $\forall \alpha, \beta \in (T \cup N)^*, \alpha A \beta \stackrel{1}{\Rightarrow} \alpha\gamma\beta$.
- 设 $A \stackrel{*}{\Rightarrow} u$ 则 $\forall \alpha, \beta \in (T \cup N)^*, \alpha \underline{A} \beta \stackrel{*}{\Rightarrow} \alpha u \beta$.
- $\stackrel{*}{\Rightarrow}$ 是 $(T \cup N)^*$ 上的关系 $\Rightarrow$ 的自反传递闭包.

Example

- ID PLUS $\underline{\text{term}}$ TIMES fac $\Rightarrow$ ID PLUS fac TIMES fac
  ID TIMES $\underline{\text{term}}$ TIMES fac $\Rightarrow$ ID TIMES fac TIMES fac.
- 因为 exp $\stackrel{*}{\Rightarrow}$ ID PLUS ID TIMES ID, 则:
  term PLUS $\underline{\text{exp}}$ $\stackrel{*}{\Rightarrow}$ term PLUS ID PLUS ID TIMES ID
  产生式如同公理, 推导结果如同定理.

上下文无关语言

定义

- 设 G 是 CFG，G 所生成的语言 $L(G) \subseteq T^*$ 定义为：
$$L(G) = \{\alpha \mid \alpha \in T^* \wedge S \overset{*}{\Rightarrow} \alpha\}$$
α 称为语言 $L(G)$ 的语句。
- $L \subseteq T^*$ 是上下文无关语言 (CFL) iff $\exists$ CGF G and $L = L(G)$.

Example

上下文无关语言

定义

- 设 G 是 CFG，G 所生成的语言 $L(G) \subseteq T^*$ 定义为：
$$L(G) = \{\alpha \mid \alpha \in T^* \wedge S \stackrel{*}{\Rightarrow} \alpha\}$$
α 称为语言 $L(G)$ 的语句.
- $L \subseteq T^*$ 是上下文无关语言 (CFL) iff $\exists$ CGF G and $L = L(G)$.

Example

- L(XL) = {算术表达式集合}.
- ID PLUS ID TIMES ID 是 XL 语言的语句。
- ID PLUS TIMES ID 不是 XL 语言的语句。

上下文无关语言

定义

- 设 G 是 CFG，G 所生成的语言 $L(G) \subseteq T^*$ 定义为：
$$L(G) = \{\alpha \mid \alpha \in T^* \wedge S \stackrel{*}{\Rightarrow} \alpha\}$$
α 称为语言 $L(G)$ 的语句.
- $L \subseteq T^*$ 是上下文无关语言 (CFL) iff $\exists$ CGF G and $L = L(G)$.

Example

- $L(XL) = \{$ 算术表达式集合 $\}$.

上下文无关语言

定义

- 设 G 是 CFG，G 所生成的语言 $L(G) \subseteq T^*$ 定义为：
$$L(G) = \{\alpha \mid \alpha \in T^* \wedge S \overset{*}{\Rightarrow} \alpha\}$$
α 称为语言 $L(G)$ 的语句.
- $L \subseteq T^*$ 是上下文无关语言 (CFL) iff $\exists$ CGF G and $L = L(G)$.

Example

- $L(XL) = \{$ 算术表达式集合 $\}$.
- ID PLUS ID TIMES ID 是 XL 语言的语句;
- ID PLUS TIMES ID 不是 XL 语言的语句.

上下文无关语言

定义

- 设 G 是 CFG，G 所生成的语言 $L(G) \subseteq T^*$ 定义为：
$$L(G) = \{\alpha \mid \alpha \in T^* \wedge S \overset{*}{\Rightarrow} \alpha\}$$
α 称为语言 $L(G)$ 的语句.
- $L \subseteq T^*$ 是上下文无关语言 (CFL) iff $\exists$ CGF G and $L = L(G)$.

Example

- $L(XL) = \{$ 算术表达式集合 $\}$.
- ID PLUS ID TIMES ID 是 XL 语言的语句;
- ID PLUS TIMES ID 不是 XL 语言的语句.

上下文无关语言

定义

- 设 G 是 CFG，G 所生成的语言 $L(G) \subseteq T^*$ 定义为：
$$L(G) = \{\alpha \mid \alpha \in T^* \wedge S \xRightarrow{*} \alpha\}$$
α 称为语言 $L(G)$ 的语句.
- $L \subseteq T^*$ 是上下文无关语言 (CFL) iff $\exists$ CGF G and $L = L(G)$.

Example

- $L(XL) = \{$ 算术表达式集合$\}$.
- ID PLUS ID TIMES ID 是 XL 语言的语句;
- ID PLUS TIMES ID 不是 XL 语言的语句.

上下文无关语言

定义

- 设 G 是 CFG，G 所生成的语言 $L(G) \subseteq T^*$ 定义为：
$$L(G) = \{\alpha \mid \alpha \in T^* \wedge S \stackrel{*}{\Rightarrow} \alpha\}$$
α 称为语言 $L(G)$ 的语句.
- $L \subseteq T^*$ 是上下文无关语言 (CFL) iff $\exists$ CGF G and $L = L(G)$.

Example

- $L(XL) = \{$ 算术表达式集合$\}$.
- ID PLUS ID TIMES ID 是 XL 语言的语句;
- ID PLUS TIMES ID 不是 XL 语言的语句.

句型

定义及性质

- 设 G 是 CFG, $\alpha \in (T \cup N)^*$ 是 G 的一个句型 (sentential form) iff $S \stackrel{*}{\Rightarrow} \alpha$.
- 语句是特殊的句型.
- 句型中至少有个子串是某一产生式的右边的文法符号串.
- 如果句型不正确，则永远不可能推出语句.
- 句型分析是语法分析的核心.

句型

定义及性质

- 设 G 是 CFG, $\alpha \in (T \cup N)^*$ 是 G 的一个句型 (sentential form) iff $S \overset{*}{\Rightarrow} \alpha$.
- 语句是特殊的句型.
- 句型中至少有个子串是某一产生式的右边的文法符号串.
- 如果句型不正确, 则永远不可能推出语句.
- 句型分析是语法分析的核心.

句型

定义及性质

- 设 G 是 CFG, $\alpha \in (T \cup N)^*$ 是 G 的一个句型 (sentential form) iff $S \stackrel{*}{\Rightarrow} \alpha$.
- 语句是特殊的句型.
- 句型中至少有个子串是某一产生式的右边的文法符号串.
- 如果句型不正确，则永远不可能推出语句.
- 句型分析是语法分析的核心.

句型

定义及性质

- 设 G 是 CFG, $\alpha \in (T \cup N)^*$ 是 G 的一个句型 (sentential form) iff $S \stackrel{*}{\Rightarrow} \alpha$.
- 语句是特殊的句型.
- 句型中至少有个子串是某一产生式的右边的文法符号串.
- 如果句型不正确，则永远不可能推出语句.
- 句型分析是语法分析的核心.

句型

定义及性质

- 设 G 是 CFG, $\alpha \in (T \cup N)^*$ 是 G 的一个句型 (sentential form) iff $S \xRightarrow{*} \alpha$.
- 语句是特殊的句型.
- 句型中至少有个子串是某一产生式的右边的文法符号串.
- 如果句型不正确, 则永远不可能推出语句.
- 句型分析是语法分析的核心.

句型

定义及性质

- 设 G 是 CFG, $\alpha \in (T \cup N)^*$ 是 G 的一个句型 (sentential form) iff $S \overset{*}{\Rightarrow} \alpha$.
- 语句是特殊的句型.
- 句型中至少有个子串是某一产生式的右边的文法符号串.
- 如果句型不正确, 则永远不可能推出语句.
- 句型分析是语法分析的核心.

Example: XL 语言的句型分析

$$
\begin{aligned}
& \underline{exp} \\
\Rightarrow\ & \underline{exp}\ \text{PLUS}\ term \\
\Rightarrow\ & \underline{exp}\ \text{PLUS}\ term\ \text{PLUS}\ term \\
& \cdots \\
\Rightarrow\ & \underline{exp}\ (\text{PLUS}\ term)^n \\
\Rightarrow\ & term\ (\text{PLUS}\ term)^n
\end{aligned}
$$

XL 语言每个非终结符对应的句型

- $(exp|term)(\text{PLUS}\ term)^*$ 是 exp 的句型.
- 即 exp 只能推出 term 或者多 term 中间用 PLUS 链接的串.
- 同样 $(term|fac)(\text{TIMES}\ fac)^*$ 是 term 的句型.
- 即 term 只能是一个 fac 或者多 fac 中间用 TIMES 链接的串.
- fac 的句型是 LP exp RP|ID.

Example: XL 语言的句型分析

$$
\begin{aligned}
& \underline{\text{exp}} \\
\Rightarrow\ & \underline{\text{exp}}\ \text{PLUS term} \\
\Rightarrow\ & \underline{\text{exp}}\ \text{PLUS term PLUS term} \\
& \cdots \\
\Rightarrow\ & \underline{\text{exp}}\ (\text{PLUS term})^n \\
\Rightarrow\ & \text{term}\ (\text{PLUS term})^n
\end{aligned}
$$

XL 语言每个非终结符对应的句型

- $(\text{exp}|\text{term})(\text{PLUS term})^*$ 是 exp 的句型.
- 即 exp 只能推出 term 或者多 term 中间用 PLUS 链接的串.
- 同样 $(\text{term}|\text{fac})(\text{TIMES fac})^*$ 是 term 的句型.
- 即 term 只能是一个 fac 或者多 fac 中间用 TIMES 链接的串.
- fac 的句型是 LP exp RP|ID.

Example: XL 语言的句型分析

$$
\begin{aligned}
& \underline{\text{exp}} \\
\Rightarrow\ & \underline{\text{exp}}\ \text{PLUS}\ \text{term} \\
\Rightarrow\ & \underline{\text{exp}}\ \text{PLUS}\ \text{term}\ \text{PLUS}\ \text{term} \\
& \cdots \\
\Rightarrow\ & \underline{\text{exp}}\ (\text{PLUS}\ \text{term})^n \\
\Rightarrow\ & \text{term}\ (\text{PLUS}\ \text{term})^n
\end{aligned}
$$

XL 语言每个非终结符对应的句型

- $(\text{exp}|\text{term})(\text{PLUS}\ \text{term})^*$ 是 exp 的句型.
- 即 exp 只能推出 term 或者多 term 中间用 PLUS 链接的串.
- 同样 $(\text{term}|\text{fac})(\text{TIMES}\ \text{fac})^*$ 是 term 的句型.
- 即 term 只能是一个 fac 或者多 fac 中间用 TIMES 链接的串.
- fac 的句型是 LP exp RP|ID.

Example: XL 语言的句型分析

$$
\begin{aligned}
& \underline{exp} \\
\Rightarrow\ & \underline{exp}\ \text{PLUS}\ term \\
\Rightarrow\ & \underline{exp}\ \text{PLUS}\ term\ \text{PLUS}\ term \\
& \cdots \\
\Rightarrow\ & \underline{exp}\ (\text{PLUS}\ term)^n \\
\Rightarrow\ & term\ (\text{PLUS}\ term)^n
\end{aligned}
$$

XL 语言每个非终结符对应的句型

- $(exp|term)(\text{PLUS}\ term)^*$ 是 exp 的句型.
- 即 exp 只能推出 term 或者多 term 中间用 PLUS 链接的串.
- 同样 $(term|fac)(\text{TIMES}\ fac)^*$ 是 term 的句型.
- 即 term 只能是一个 fac 或者多 fac 中间用 TIMES 链接的串.
- fac 的句型是 LP exp RP|ID.

Example：XL 语言的句型分析

$$
\begin{aligned}
& \text{exp} \\
\Rightarrow\ & \text{exp}\ \mathrm{PLUS}\ \text{term} \\
\Rightarrow\ & \text{exp}\ \mathrm{PLUS}\ \text{term}\ \mathrm{PLUS}\ \text{term} \\
& \cdots \\
\Rightarrow\ & \text{exp}\ (\mathrm{PLUS}\ \text{term})^n \\
\Rightarrow\ & \text{term}\ (\mathrm{PLUS}\ \text{term})^n
\end{aligned}
$$

XL 语言每个非终结符对应的句型

- $(\text{exp}|\text{term})(\mathrm{PLUS}\ \text{term})^*$ 是 exp 的句型。
- 即 exp 只能推出 term 或者多 term 中间用 PLUS 链接的串。
- 同样 $(\text{term}|\text{fac})(\mathrm{TIMES}\ \text{fac})^*$ 是 term 的句型。
- 即 term 只能是一个 fac 或者多 fac 中间用 TIMES 链接的串。
- fac 的句型是 $\mathrm{LP}\ \text{exp}\ \mathrm{RP}|\mathrm{ID}$.

Example: XL 语言的句型分析

	exp
$\Rightarrow$	exp PLUS term
$\Rightarrow$	exp PLUS term PLUS term
	...
$\Rightarrow$	exp (PLUS term)^n
$\Rightarrow$	term (PLUS term)^n

XL 语言每个非终结符对应的句型

- (exp|term)(PLUS term)* 是 exp 的句型.
- 即 exp 只能推出 term 或者多 term 中间用 PLUS 链接的串.
- 同样 (term|fac)(TIMES fac)* 是 term 的句型.
- 即 term 只能是一个 fac 或者多 fac 中间用 TIMES 链接的串.
- fac 的句型是 LP exp RP|ID.

Example: XL 语言的句型分析

$$
\begin{array}{ll}
 & \underline{exp} \\
\Rightarrow & \underline{exp}\ \text{PLUS}\ term \\
\Rightarrow & \underline{exp}\ \text{PLUS}\ term\ \text{PLUS}\ term \\
 & \cdots \\
\Rightarrow & \underline{exp}\ (\text{PLUS}\ term)^n \\
\Rightarrow & term\ (\text{PLUS}\ term)^n
\end{array}
$$

XL 语言每个非终结符对应的句型

- $(exp|term)(\text{PLUS}\ term)^*$ 是 exp 的句型.
- 即 exp 只能推出 term 或者多 term 中间用 PLUS 链接的串.
- 同样 $(term|fac)(\text{TIMES}\ fac)^*$ 是 term 的句型.
- 即 term 只能是一个 fac 或者多 fac 中间用 TIMES 链接的串.
- fac 的句型是 LP exp RP|ID.

Example: XL 语言的句型分析

$$
\begin{aligned}
& \underline{exp} \\
\Rightarrow\ & \underline{exp}\ \text{PLUS}\ term \\
\Rightarrow\ & \underline{exp}\ \text{PLUS}\ term\ \text{PLUS}\ term \\
& \cdots \\
\Rightarrow\ & \underline{exp}\ (\text{PLUS}\ term)^n \\
\Rightarrow\ & term\ (\text{PLUS}\ term)^n
\end{aligned}
$$

XL 语言每个非终结符对应的句型

- $(exp|term)(\text{PLUS}\ term)^*$ 是 exp 的句型.
- 即 exp 只能推出 term 或者多 term 中间用 PLUS 链接的串.
- 同样 $(term|fac)(\text{TIMES}\ fac)^*$ 是 term 的句型.
- 即 term 只能是一个 fac 或者多 fac 中间用 TIMES 链接的串.
- fac 的句型是 $\text{LP}\ exp\ \text{RP}|\text{ID}$.

XL 语言的运算优先级

- exp PLUS exp 和 exp TIMES term 不是句型.
- term TIMES term 和 term PLUS exp 也不是句型.
- $\underbrace{\underbrace{\underbrace{\text{ID TIMES ID}}_{\text{term}}}_{\text{exp}}\ \text{PLUS ID}}_{\text{exp}}$ $\qquad$ $\underbrace{\text{ID TIMES}\ \underbrace{\text{ID PLUS ID}}_{\text{exp}}}_{\text{error}}$
- 语句 ID TIMES ID PLUS ID 只能是由句型 exp PLUS term 生成的句型 term TIMES fac PLUS term 生成, 即 TIMES 两边的 ID 由同一个非终结符生成. PLUS 两边的 ID 不可能由同一个非终结符生成. 因为生成 PLUS 的非终结符只有 exp, 但 ID TIMES exp 不是句型.
- XL 语言的乘法运算先于加法结合, 即乘法的优先级高于加法.

XL 语言的运算优先级

- exp PLUS exp 和 exp TIMES term 不是句型.
- term TIMES term 和 term PLUS exp 也不是句型.
- $\underbrace{\underbrace{\underbrace{\text{ID TIMES ID}}_{\text{term}}}_{\text{exp}}\ \text{PLUS ID}}_{\text{exp}}$ $\qquad$ $\underbrace{\text{ID TIMES}\ \underbrace{\text{ID PLUS ID}}_{\text{exp}}}_{\text{error}}$
- 语句 ID TIMES ID PLUS ID 只能是由句型 exp PLUS term 生成的句型 term TIMES fac PLUS term 生成. 即 TIMES 两边的 ID 由同一个非终结符生成. PLUS 两边的 ID 不可能由同一个非终结符生成. 因为生成 PLUS 的非终结符只有 exp, 但 ID TIMES exp 不是句型.
- XL 语言的乘法运算先于加法结合，即乘法的优先级高于加法.

XL 语言的运算优先级

- exp PLUS exp 和 exp TIMES term 不是句型.
- term TIMES term 和 term PLUS exp 也不是句型.
- $\underbrace{\underbrace{\underbrace{\text{ID TIMES ID}}_{\text{term}}}_{\text{exp}}\ \text{PLUS ID}}_{\text{exp}}$ $\qquad$ $\underbrace{\text{ID TIMES}\ \underbrace{\text{ID PLUS ID}}_{\text{exp}}}_{\text{error}}$
- 语句 ID TIMES ID PLUS ID 只能是由句型 exp PLUS term 生成的句型 term TIMES fac PLUS term 生成. 即 TIMES 两边的 ID 由同一个非终结符生成. PLUS 两边的 ID 不可能由同一个非终结符生成. 因为生成 PLUS 的非终结符只有 exp, 但 ID TIMES exp 不是句型.
- XL 语言的乘法运算先于加法结合, 即乘法的优先级高于加法.

XL 语言的运算优先级

- exp PLUS exp 和 exp TIMES term 不是句型.
- term TIMES term 和 term PLUS exp 也不是句型.
- $\underbrace{\underbrace{\underbrace{\text{ID TIMES ID}}_{\text{term}}}_{\text{exp}}\ \text{PLUS ID}}_{\text{exp}}$ $\qquad$ $\underbrace{\text{ID TIMES}\ \underbrace{\text{ID PLUS ID}}_{\text{exp}}}_{\text{error}}$
- 语句 ID TIMES ID PLUS ID 只能是由句型 exp PLUS term 生成的句型 term TIMES fac PLUS term 生成. 即 TIMES 两边的 ID 由同一个非终结符生成. PLUS 两边的 ID 不可能由同一个非终结符生成. 因为生成 PLUS 的非终结符只有 exp, 但 ID TIMES exp 不是句型.
- XL 语言的乘法运算先于加法结合, 即乘法的优先级高于加法.

XL 语言的运算优先级

- exp PLUS exp 和 exp TIMES term 不是句型.
- term TIMES term 和 term PLUS exp 也不是句型.
- $\underbrace{\underbrace{\underbrace{\mathrm{ID\ TIMES\ ID}}_{\mathrm{term}}}_{\mathrm{exp}}\ \mathrm{PLUS\ ID}}_{\mathrm{exp}}$ $\qquad$ $\underbrace{\mathrm{ID\ TIMES}\ \underbrace{\mathrm{ID\ PLUS\ ID}}_{\mathrm{exp}}}_{\mathrm{error}}$
- 语句 ID TIMES ID PLUS ID 只能是由句型 exp PLUS term 生成的句型 term TIMES fac PLUS term 生成. 即 TIMES 两边的 ID 由同一个非终结符生成. PLUS 两边的 ID 不可能由同一个非终结符生成. 因为生成 PLUS 的非终结符只有 exp, 但 ID TIMES exp 不是句型.
- XL 语言的乘法运算先于加法结合, 即乘法的优先级高于加法.

XL 语言的运算优先级

- exp PLUS exp 和 exp TIMES term 不是句型.
- term TIMES term 和 term PLUS exp 也不是句型.
- $\underbrace{\underbrace{\underbrace{\text{ID TIMES ID}}_{\text{term}}}_{\text{exp}}\ \text{PLUS ID}}_{\text{exp}}$ $\qquad$ $\underbrace{\text{ID TIMES}\ \underbrace{\text{ID PLUS ID}}_{\text{exp}}}_{\text{error}}$
- 语句 ID TIMES ID PLUS ID 只能是由句型 exp PLUS term 生成的句型 term TIMES fac PLUS term 生成. 即 TIMES 两边的 ID 由同一个非终结符生成. PLUS 两边的 ID 不可能由同一个非终结符生成. 因为生成 PLUS 的非终结符只有 exp, 但 ID TIMES exp 不是句型.
- XL 语言的乘法运算先于加法结合, 即乘法的优先级高于加法.

XL 语言的运算结合律

- $\underbrace{\underbrace{\text{ID PLUS ID}}_{\text{exp}}\ \text{PLUS ID}}_{\text{exp}}$ $\quad$ $\underbrace{\text{ID PLUS }\underbrace{\text{ID PLUS ID}}_{\text{exp}}}_{\text{error}}$
- 原因：没有 PLUS exp 为子串的句型.
- 加法运算左结合.
- 同样乘法运算也是左结合的.

XL 语言的运算结合律

- $\underbrace{\underbrace{\text{ID PLUS ID}}_{\text{exp}}\ \text{PLUS ID}}_{\text{exp}}$ $\qquad$ $\underbrace{\text{ID PLUS}\ \underbrace{\text{ID PLUS ID}}_{\text{exp}}}_{\text{error}}$
- 原因：没有 PLUS exp 为子串的句型.
- 加法运算左结合.
- 同样乘法运算也是左结合的.

XL 语言的运算结合律

- $\underbrace{\underbrace{\text{ID PLUS ID}}_{\text{exp}}\ \text{PLUS ID}}_{\text{exp}}$ $\qquad$ $\underbrace{\text{ID PLUS}\ \underbrace{\text{ID PLUS ID}}_{\text{exp}}}_{\text{error}}$
- 原因：没有 PLUS exp 为子串的句型.
- 加法运算左结合.
- 同样乘法运算也是左结合的.

XL 语言的运算结合律

- $\underbrace{\underbrace{\text{ID PLUS ID}}_{\text{exp}}\ \text{PLUS ID}}_{\text{exp}}$ $\qquad$ $\underbrace{\text{ID PLUS}\ \underbrace{\text{ID PLUS ID}}_{\text{exp}}}_{\text{error}}$
- 原因：没有 PLUS exp 为子串的句型.
- 加法运算左结合.
- 同样乘法运算也是左结合的.

XL 语言的运算结合律

- $\underbrace{\underbrace{\text{ID PLUS ID}}_{\text{exp}}\ \text{PLUS ID}}_{\text{exp}}$ $\quad$ $\underbrace{\text{ID PLUS}\ \underbrace{\text{ID PLUS ID}}_{\text{exp}}}_{\text{error}}$
- 原因：没有 PLUS exp 为子串的句型.
- 加法运算左结合.
- 同样乘法运算也是左结合的.

文法等价

设 G_1, G_2 是两个 CFG, G_1 和 G_2 等价 iff $L(G_1) = L(G_2)$.

Grammar of XL'

```
T = { ID,  PLUS,
      TIMES,  LP,  RP };
N = {exp};
S = exp;
P:
1/ exp -> exp PLUS exp
2/      |  exp TIMES exp
3/      |  LP exp RP
4/      |  ID
```

Grammar of XL''

```
T = { ID,  PLUS,
      TIMES,  LP,  RP };
N = {exp, term, fac};
S = exp;
P:
1/ exp -> term PLUS exp
2/      |  term
3/ term -> fac TIMES term
4/       | fac
5/ fac ->  ID
6/      |  LP exp RP
/* 乘法优先且右结合 */
```

$L(XL) = L(XL') = L(XL'')$.

Example

等价文法

$$G_1 = \begin{cases} S \rightarrow AB \mid CA \\ A \rightarrow a \\ B \rightarrow BC \mid AB \\ C \rightarrow aB \mid b \end{cases} \qquad G_2 = \begin{cases} S \rightarrow CA \\ A \rightarrow a \\ C \rightarrow b \end{cases}$$

- B 不可能推出终结符来，是无用符号.
- 消除 B 即得等价文法 G_2.

Example

等价文法

$$G_1 = \begin{cases} S \rightarrow AB \mid CA \\ A \rightarrow a \\ B \rightarrow BC \mid AB \\ C \rightarrow aB \mid b \end{cases} \qquad G_2 = \begin{cases} S \rightarrow CA \\ A \rightarrow a \\ C \rightarrow b \end{cases}$$

- B 不可能推出终结符来，是无用符号.
- 消除 B 即得等价文法 G_2.

Example

等价文法

$$G_1 = \begin{cases} S \rightarrow AB \mid CA \\ A \rightarrow a \\ B \rightarrow BC \mid AB \\ C \rightarrow aB \mid b \end{cases} \qquad G_2 = \begin{cases} S \rightarrow CA \\ A \rightarrow a \\ C \rightarrow b \end{cases}$$

- B 不可能推出终结符来，是无用符号.
- 消除 B 即得等价文法 G_2.

Example

等价文法

$$G_1 = \begin{cases} S \rightarrow AB \mid CA \\ A \rightarrow a \\ B \rightarrow BC \mid AB \\ C \rightarrow aB \mid b \end{cases} \qquad G_2 = \begin{cases} S \rightarrow CA \\ A \rightarrow a \\ C \rightarrow b \end{cases}$$

- B 不可能推出终结符来，是无用符号.
- 消除 B 即得等价文法 G_2.

Example

等价文法

$$G_1 = \begin{cases} S \rightarrow AB \mid CA \\ A \rightarrow a \\ B \rightarrow BC \mid AB \\ C \rightarrow aB \mid b \end{cases} \qquad G_2 = \begin{cases} S \rightarrow CA \\ A \rightarrow a \\ C \rightarrow b \end{cases}$$

- B 不可能推出终结符来，是无用符号.
- 消除 B 即得等价文法 G_2.

Example

等价文法

$$G_1 = \begin{cases} S \rightarrow AB \mid CA \\ A \rightarrow a \\ B \rightarrow BC \mid AB \\ C \rightarrow aB \mid b \end{cases} \qquad G_2 = \begin{cases} S \rightarrow CA \\ A \rightarrow a \\ C \rightarrow b \end{cases}$$

- B 不可能推出终结符来，是无用符号.
- 消除 B 即得等价文法 G_2.

推导的不确定性

两个不确定性

- 选择句型中的非终结符展开.
- 选择产生式.

Grammar of XL

```
T = { ID, PLUS, TIMES,
      LP, RP };
N = {exp, term, fac};
S = exp;
P:
1/ exp -> exp PLUS term
2/      |  term
3/ term -> term TIMES fac
4/       |  fac
5/ fac -> ID
6/      |  LP exp  RP
```

Example

	exp	
⇒	exp PLUS term	by 1/
⇒	term PLUS term	by 2/
⇒	fac PLUS term	by 4/
⇒	ID PLUS term	by 5/
⇒	ID PLUS term TIMES fac	by 3/
⇒	ID PLUS fac TIMES fac	by 4/
⇒	ID PLUS ID TIMES fac	by 5/
⇒	ID PLUS ID TIMES ID	by 5/
	exp	
⇒	exp PLUS term	by 1/
⇒	exp PLUS term TIMES fac	by 3/
⇒	exp PLUS term TIMES ID	by 5/
⇒	exp PLUS fac TIMES ID	by 4/
⇒	exp PLUS ID TIMES ID	by 5/
⇒	term PLUS ID TIMES ID	by 2/
⇒	fac PLUS ID TIMES ID	by 4/
⇒	ID PLUS ID TIMES ID	by 5/

推导的不确定性

两个不确定性

- 选择句型中的非终结符展开.
- 选择产生式.

Grammar of XL

```
T = { ID, PLUS, TIMES,
      LP, RP };
N = {exp, term, fac};
S = exp;
P:
1/ exp -> exp PLUS term
2/      |  term
3/ term -> term TIMES fac
4/       |  fac
5/ fac -> ID
6/      |  LP exp  RP
```

Example

	exp	
⇒	exp PLUS term	by 1/
⇒	term PLUS term	by 2/
⇒	fac PLUS term	by 4/
⇒	ID PLUS term	by 5/
⇒	ID PLUS term TIMES fac	by 3/
⇒	ID PLUS fac TIMES fac	by 4/
⇒	ID PLUS ID TIMES fac	by 5/
⇒	ID PLUS ID TIMES ID	by 5/

	exp	
⇒	exp PLUS term	by 1/
⇒	exp PLUS term TIMES fac	by 3/
⇒	exp PLUS term TIMES ID	by 5/
⇒	exp PLUS fac TIMES ID	by 4/
⇒	exp PLUS ID TIMES ID	by 5/
⇒	term PLUS ID TIMES ID	by 2/
⇒	fac PLUS ID TIMES ID	by 4/
⇒	ID PLUS ID TIMES ID	by 5/

推导的不确定性

两个不确定性

- 选择句型中的非终结符展开.
- 选择产生式.

Grammar of XL

```
T = { ID, PLUS, TIMES,
      LP, RP };
N = {exp, term, fac};
S = exp;
P:
1/ exp -> exp PLUS term
2/       |  term
3/ term -> term TIMES fac
4/         |  fac
5/ fac -> ID
6/       |  LP exp  RP
```

Example

	exp	
⇒	exp PLUS term	by 1/
⇒	term PLUS term	by 2/
⇒	fac PLUS term	by 4/
⇒	ID PLUS term	by 5/
⇒	ID PLUS term TIMES fac	by 3/
⇒	ID PLUS fac TIMES fac	by 4/
⇒	ID PLUS ID TIMES fac	by 5/
⇒	ID PLUS ID TIMES ID	by 5/
	exp	
⇒	exp PLUS term	by 1/
⇒	exp PLUS term TIMES fac	by 3/
⇒	exp PLUS term TIMES ID	by 5/
⇒	exp PLUS fac TIMES ID	by 4/
⇒	exp PLUS ID TIMES ID	by 5/
⇒	term PLUS ID TIMES ID	by 2/
⇒	fac PLUS ID TIMES ID	by 4/
⇒	ID PLUS ID TIMES ID	by 5/

最左推导与最右推导

定义

- 每次推导都是对句型中的最左边的非终结符展开，称为最左推导，记为：$\overset{*}{\underset{lm}{\Rightarrow}}$ (leftmost);
- 每次推导都是对句型中的最右边的非终结符展开，称为最右推导，记为：$\overset{*}{\underset{rm}{\Rightarrow}}$ (rightmost).

最左推导与最右推导

定义

- 每次推导都是对句型中的最左边的非终结符展开，称为**最左推导**，记为: $\underset{lm}{\overset{*}{\Rightarrow}}$ (leftmost);
- 每次推导都是对句型中的最右边的非终结符展开，称为**最右推导**，记为: $\underset{rm}{\overset{*}{\Rightarrow}}$ (rightmost).

最左推导与最右推导

定义

- 每次推导都是对句型中的最左边的非终结符展开，称为**最左推导**，记为：$\overset{*}{\underset{lm}{\Rightarrow}}$ (leftmost);
- 每次推导都是对句型中的最右边的非终结符展开，称为**最右推导**，记为：$\overset{*}{\underset{rm}{\Rightarrow}}$ (rightmost).

Example

最左推导

	exp	
$\underset{lm}{\Rightarrow}$	exp PLUS term	by 1/
$\underset{lm}{\Rightarrow}$	term PLUS term	by 2/
$\underset{lm}{\Rightarrow}$	fac PLUS term	by 4/
$\underset{lm}{\Rightarrow}$	ID PLUS term	by 5/
$\underset{lm}{\Rightarrow}$	ID PLUS term TIMES fac	by 3/
$\underset{lm}{\Rightarrow}$	ID PLUS fac TIMES fac	by 4/
$\underset{lm}{\Rightarrow}$	ID PLUS ID TIMES fac	by 5/
$\underset{lm}{\Rightarrow}$	ID PLUS ID TIMES ID	by 5/

最右推导

	exp	
$\underset{rm}{\Rightarrow}$	exp PLUS term	by 1/
$\underset{rm}{\Rightarrow}$	exp PLUS term TIMES fac	by 3/
$\underset{rm}{\Rightarrow}$	exp PLUS term TIMES ID	by 5/
$\underset{rm}{\Rightarrow}$	exp PLUS fac TIMES ID	by 4/
$\underset{rm}{\Rightarrow}$	exp PLUS ID TIMES ID	by 5/
$\underset{rm}{\Rightarrow}$	term PLUS ID TIMES ID	by 2/
$\underset{rm}{\Rightarrow}$	fac PLUS ID TIMES ID	by 4/
$\underset{rm}{\Rightarrow}$	ID PLUS ID TIMES ID	by 5/

Example

最左推导

	exp	
$\underset{lm}{\Rightarrow}$	$\underline{\mathit{exp}}$ PLUS term	by 1/
$\underset{lm}{\Rightarrow}$	$\underline{\mathit{term}}$ PLUS term	by 2/
$\underset{lm}{\Rightarrow}$	$\underline{\mathit{fac}}$ PLUS term	by 4/
$\underset{lm}{\Rightarrow}$	ID PLUS $\underline{\mathit{term}}$	by 5/
$\underset{lm}{\Rightarrow}$	ID PLUS $\underline{\mathit{term}}$ TIMES fac	by 3/
$\underset{lm}{\Rightarrow}$	ID PLUS $\underline{\mathit{fac}}$ TIMES fac	by 4/
$\underset{lm}{\Rightarrow}$	ID PLUS ID TIMES $\underline{\mathit{fac}}$	by 5/
$\underset{lm}{\Rightarrow}$	ID PLUS ID TIMES ID	by 5/

最右推导

	exp	
$\underset{rm}{\Rightarrow}$	exp PLUS $\underline{\mathit{term}}$	by 1/
$\underset{rm}{\Rightarrow}$	exp PLUS term TIMES $\underline{\mathit{fac}}$	by 3/
$\underset{rm}{\Rightarrow}$	exp PLUS $\underline{\mathit{term}}$ TIMES ID	by 5/
$\underset{rm}{\Rightarrow}$	exp PLUS $\underline{\mathit{fac}}$ TIMES ID	by 4/
$\underset{rm}{\Rightarrow}$	exp PLUS ID TIMES ID	by 5/
$\underset{rm}{\Rightarrow}$	$\underline{\mathit{term}}$ PLUS ID TIMES ID	by 2/
$\underset{rm}{\Rightarrow}$	$\underline{\mathit{fac}}$ PLUS ID TIMES ID	by 4/
$\underset{rm}{\Rightarrow}$	ID PLUS ID TIMES ID	by 5/

最左与最右句型

定义

设 G 是 CFG.

- $\alpha \in (T \cup N)^*$ 是 G 的一个最左句型 (leftmost sentential form) iff $S \overset{*}{\underset{lm}{\Rightarrow}} \alpha$.
- $\alpha \in (T \cup N)^*$ 是 G 的一个最右句型 (rightmost sentential form) iff $S \overset{*}{\underset{rm}{\Rightarrow}} \alpha$.

最左与最右句型

定义

设 G 是 CFG.

- $\alpha \in (T \cup N)^*$ 是 G 的一个最左句型 (leftmost sentential form) iff $S \underset{lm}{\overset{*}{\Rightarrow}} \alpha$.
- $\alpha \in (T \cup N)^*$ 是 G 的一个最右句型 (rightmost sentential form) iff $S \underset{rm}{\overset{*}{\Rightarrow}} \alpha$.

最左与最右句型

定义

设 G 是 CFG.

- $\alpha \in (T \cup N)^*$ 是 G 的一个最左句型 (leftmost sentential form) iff $S \underset{lm}{\overset{*}{\Rightarrow}} \alpha$.
- $\alpha \in (T \cup N)^*$ 是 G 的一个最右句型 (rightmost sentential form) iff $S \underset{rm}{\overset{*}{\Rightarrow}} \alpha$.

Example

最左句型

	exp	
$\underset{lm}{\Rightarrow}$	exp PLUS term	✓
$\underset{lm}{\Rightarrow}$	term PLUS term	✓
$\underset{lm}{\Rightarrow}$	fac PLUS term	✓
$\underset{lm}{\Rightarrow}$	ID PLUS term	✓
$\underset{lm}{\Rightarrow}$	ID PLUS term TIMES fac	✓
$\Rightarrow$	ID PLUS fac TIMES ID	✗

最右句型

	exp	
$\underset{rm}{\Rightarrow}$	exp PLUS term	✓
$\underset{rm}{\Rightarrow}$	exp PLUS term TIMES fac	✓
$\underset{rm}{\Rightarrow}$	exp PLUS term TIMES ID	✓
$\Rightarrow$	term PLUS term TIMES ID	✗

Example

最左句型

	exp	
$\underset{lm}{\Rightarrow}$	$\underline{\text{exp}}$ PLUS term	✓
$\underset{lm}{\Rightarrow}$	$\underline{\text{term}}$ PLUS term	✓
$\underset{lm}{\Rightarrow}$	$\underline{\text{fac}}$ PLUS term	✓
$\underset{lm}{\Rightarrow}$	ID PLUS $\underline{\text{term}}$	✓
$\underset{lm}{\Rightarrow}$	ID PLUS term TIMES $\underline{\text{fac}}$	✓
$\Rightarrow$	ID PLUS fac TIMES ID	✗

最右句型

	exp	
$\underset{rm}{\Rightarrow}$	exp PLUS $\underline{\text{term}}$	✓
$\underset{rm}{\Rightarrow}$	exp PLUS term TIMES $\underline{\text{fac}}$	✓
$\underset{rm}{\Rightarrow}$	$\underline{\text{exp}}$ PLUS term TIMES ID	✓
$\Rightarrow$	term PLUS term TIMES ID	✗

最左与最右句型的相关性质

性质

- 最左推导是最左边的文法符号形成为终结符，最右推导是最右边的文法符号形成为终结符。
- 每条语句既是左句型又是右句型；
- 设 $S \overset{*}{\underset{lm}{\Rightarrow}} \alpha A \beta, \beta \overset{+}{\underset{lm}{\Rightarrow}} \beta'$，则 $\alpha A \beta'$ 一定不是左句型。
- 设 $S \overset{*}{\underset{rm}{\Rightarrow}} \alpha A \beta, \alpha \overset{+}{\underset{rm}{\Rightarrow}} \alpha'$，则 $\alpha' A \beta$ 一定不是右句型。
- 语法分析的过程就是模拟这两种推导，即对这两种句型的识别。

最左与最右句型的相关性质

性质

- 最左推导是最左边的文法符号形成为终结符，最右推导是最右边的文法符号形成为终结符.
- 每条语句既是左句型又是右句型;
- 设 $S \underset{lm}{\overset{*}{\Rightarrow}} \alpha A\beta, \beta \underset{lm}{\overset{+}{\Rightarrow}} \beta'$，则 $\alpha A\beta'$ 一定不是左句型.
- 设 $S \underset{rm}{\overset{*}{\Rightarrow}} \alpha A\beta, \alpha \underset{rm}{\overset{+}{\Rightarrow}} \alpha'$，则 $\alpha' A\beta$ 一定不是右句型.
- 语法分析的过程就是模拟这两种推导，即对这两种句型的识别.

最左与最右句型的相关性质

性质

- 最左推导是最左边的文法符号形成为终结符，最右推导是最右边的文法符号形成为终结符.
- 每条语句既是左句型又是右句型;
- 设 $S \overset{*}{\underset{lm}{\Rightarrow}} \alpha A\beta, \beta \overset{+}{\underset{lm}{\Rightarrow}} \beta'$，则 $\alpha A\beta'$ 一定不是左句型.
- 设 $S \overset{*}{\underset{rm}{\Rightarrow}} \alpha A\beta, \alpha \overset{+}{\underset{rm}{\Rightarrow}} \alpha'$，则 $\alpha' A\beta$ 一定不是右句型.
- 语法分析的过程就是模拟这两种推导，即对这两种句型的识别.

最左与最右句型的相关性质

性质

- 最左推导是最左边的文法符号形成为终结符，最右推导是最右边的文法符号形成为终结符.
- 每条语句既是左句型又是右句型;
- 设 $S \underset{lm}{\overset{*}{\Rightarrow}} \alpha A\beta, \beta \underset{lm}{\overset{+}{\Rightarrow}} \beta'$，则 $\alpha A\beta'$ 一定不是左句型.
- 设 $S \underset{rm}{\overset{*}{\Rightarrow}} \alpha A\beta, \alpha \underset{rm}{\overset{+}{\Rightarrow}} \alpha'$，则 $\alpha' A\beta$ 一定不是右句型.
- 语法分析的过程就是模拟这两种推导，即对这两种句型的识别.

最左与最右句型的相关性质

性质

- 最左推导是最左边的文法符号形成为终结符，最右推导是最右边的文法符号形成为终结符.
- 每条语句既是左句型又是右句型;
- 设 $S \underset{lm}{\overset{*}{\Rightarrow}} \alpha A\beta, \beta \underset{lm}{\overset{+}{\Rightarrow}} \beta'$，则 $\alpha A\beta'$ 一定不是左句型.
- 设 $S \underset{rm}{\overset{*}{\Rightarrow}} \alpha A\beta, \alpha \underset{rm}{\overset{+}{\Rightarrow}} \alpha'$，则 $\alpha' A\beta$ 一定不是右句型.
- 语法分析的过程就是模拟这两种推导，即对这两种句型的识别.

最左与最右句型的相关性质

性质

- 最左推导是最左边的文法符号形成为终结符，最右推导是最右边的文法符号形成为终结符.
- 每条语句既是左句型又是右句型;
- 设 $S \underset{lm}{\overset{*}{\Rightarrow}} \alpha A\beta, \beta \underset{lm}{\overset{+}{\Rightarrow}} \beta'$，则 $\alpha A\beta'$ 一定不是左句型.
- 设 $S \underset{rm}{\overset{*}{\Rightarrow}} \alpha A\beta, \alpha \underset{rm}{\overset{+}{\Rightarrow}} \alpha'$，则 $\alpha' A\beta$ 一定不是右句型.
- 语法分析的过程就是模拟这两种推导. 即对这两种句型的识别.

语法树 --- Example 1/2

最左推导

exp

$\overset{1}{\underset{lm}{\Rightarrow}}$ exp PLUS term

$\overset{2}{\underset{lm}{\Rightarrow}}$ term PLUS term

$\overset{3}{\underset{lm}{\Rightarrow}}$ fac PLUS term

$\overset{4}{\underset{lm}{\Rightarrow}}$ ID PLUS term

$\overset{5}{\underset{lm}{\Rightarrow}}$ ID PLUS term TIMES fac

$\overset{6}{\underset{lm}{\Rightarrow}}$ ID PLUS fac TIMES fac

$\overset{7}{\underset{lm}{\Rightarrow}}$ ID PLUS ID TIMES fac

$\overset{8}{\underset{lm}{\Rightarrow}}$ ID PLUS ID TIMES ID

语法树的建立

语法树 --- Example 1/2

最左推导

exp

$\overset{1}{\underset{lm}{\Rightarrow}}$ exp PLUS term

$\overset{2}{\underset{lm}{\Rightarrow}}$ term PLUS term

$\overset{3}{\underset{lm}{\Rightarrow}}$ fac PLUS term

$\overset{4}{\underset{lm}{\Rightarrow}}$ ID PLUS term

$\overset{5}{\underset{lm}{\Rightarrow}}$ ID PLUS term TIMES fac

$\overset{6}{\underset{lm}{\Rightarrow}}$ ID PLUS fac TIMES fac

$\overset{7}{\underset{lm}{\Rightarrow}}$ ID PLUS ID TIMES fac

$\overset{8}{\underset{lm}{\Rightarrow}}$ ID PLUS ID TIMES ID

语法树的建立

exp

语法树 --- Example 1/2

最左推导

exp

$\xRightarrow[lm]{1}$ exp PLUS term

$\xRightarrow[lm]{2}$ term PLUS term

$\xRightarrow[lm]{3}$ fac PLUS term

$\xRightarrow[lm]{4}$ ID PLUS term

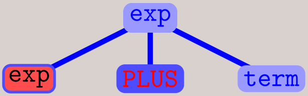
$\xRightarrow[lm]{5}$ ID PLUS term TIMES fac

$\xRightarrow[lm]{6}$ ID PLUS fac TIMES fac

$\xRightarrow[lm]{7}$ ID PLUS ID TIMES fac

$\xRightarrow[lm]{8}$ ID PLUS ID TIMES ID

语法树的建立



语法树 --- Example 1/2

最左推导

exp

$\xRightarrow[lm]{1}$ exp PLUS term

$\xRightarrow[lm]{2}$ term PLUS term

$\xRightarrow[lm]{3}$ fac PLUS term

$\xRightarrow[lm]{4}$ ID PLUS term

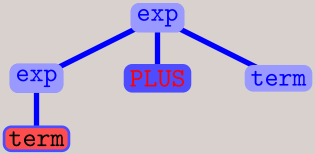
$\xRightarrow[lm]{5}$ ID PLUS term TIMES fac

$\xRightarrow[lm]{6}$ ID PLUS fac TIMES fac

$\xRightarrow[lm]{7}$ ID PLUS ID TIMES fac

$\xRightarrow[lm]{8}$ ID PLUS ID TIMES ID

语法树的建立



语法树 --- Example 1/2

最左推导

exp

$\xrightarrow[lm]{1}$ exp PLUS term

$\xrightarrow[lm]{2}$ term PLUS term

$\xrightarrow[lm]{3}$ fac PLUS term

$\xrightarrow[lm]{4}$ ID PLUS term

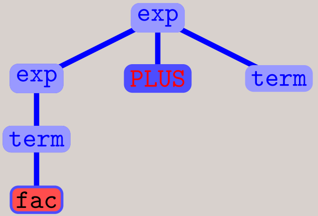
$\xrightarrow[lm]{5}$ ID PLUS term TIMES fac

$\xrightarrow[lm]{6}$ ID PLUS fac TIMES fac

$\xrightarrow[lm]{7}$ ID PLUS ID TIMES fac

$\xrightarrow[lm]{8}$ ID PLUS ID TIMES ID

语法树的建立



语法树 --- Example 1/2

最左推导

exp

$\overset{1}{\underset{lm}{\Rightarrow}}$ exp PLUS term

$\overset{2}{\underset{lm}{\Rightarrow}}$ term PLUS term

$\overset{3}{\underset{lm}{\Rightarrow}}$ fac PLUS term

$\overset{4}{\underset{lm}{\Rightarrow}}$ ID PLUS term

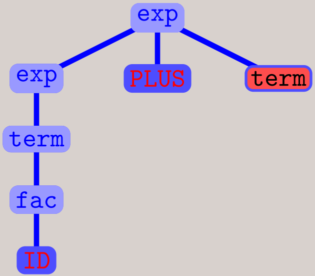
$\overset{5}{\underset{lm}{\Rightarrow}}$ ID PLUS term TIMES fac

$\overset{6}{\underset{lm}{\Rightarrow}}$ ID PLUS fac TIMES fac

$\overset{7}{\underset{lm}{\Rightarrow}}$ ID PLUS ID TIMES fac

$\overset{8}{\underset{lm}{\Rightarrow}}$ ID PLUS ID TIMES ID

语法树的建立



语法树 --- Example 1/2

最左推导

exp

$\overset{1}{\underset{lm}{\Rightarrow}}$ exp PLUS term

$\overset{2}{\underset{lm}{\Rightarrow}}$ term PLUS term

$\overset{3}{\underset{lm}{\Rightarrow}}$ fac PLUS term

$\overset{4}{\underset{lm}{\Rightarrow}}$ ID PLUS term

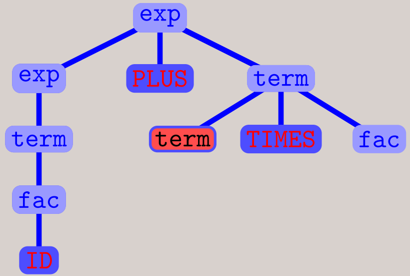
$\overset{5}{\underset{lm}{\Rightarrow}}$ ID PLUS term TIMES fac

$\overset{6}{\underset{lm}{\Rightarrow}}$ ID PLUS fac TIMES fac

$\overset{7}{\underset{lm}{\Rightarrow}}$ ID PLUS ID TIMES fac

$\overset{8}{\underset{lm}{\Rightarrow}}$ ID PLUS ID TIMES ID

语法树的建立



语法树 --- Example 1/2

最左推导

exp

$\overset{1}{\underset{lm}{\Longrightarrow}}$ $\underline{\text{exp}}$ PLUS term

$\overset{2}{\underset{lm}{\Longrightarrow}}$ $\underline{\text{term}}$ PLUS term

$\overset{3}{\underset{lm}{\Longrightarrow}}$ $\underline{\text{fac}}$ PLUS term

$\overset{4}{\underset{lm}{\Longrightarrow}}$ ID PLUS $\underline{\text{term}}$

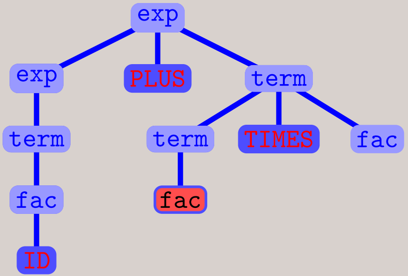
$\overset{5}{\underset{lm}{\Longrightarrow}}$ ID PLUS $\underline{\text{term}}$ TIMES fac

$\overset{6}{\underset{lm}{\Longrightarrow}}$ ID PLUS $\underline{\text{fac}}$ TIMES fac

$\overset{7}{\underset{lm}{\Longrightarrow}}$ ID PLUS ID TIMES $\underline{\text{fac}}$

$\overset{8}{\underset{lm}{\Longrightarrow}}$ ID PLUS ID TIMES ID

语法树的建立



语法树 --- Example 1/2

最左推导

exp

$\xRightarrow[lm]{1}$ exp PLUS term

$\xRightarrow[lm]{2}$ term PLUS term

$\xRightarrow[lm]{3}$ fac PLUS term

$\xRightarrow[lm]{4}$ ID PLUS term

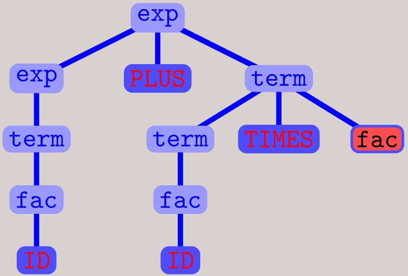
$\xRightarrow[lm]{5}$ ID PLUS term TIMES fac

$\xRightarrow[lm]{6}$ ID PLUS fac TIMES fac

$\xRightarrow[lm]{7}$ ID PLUS ID TIMES fac

$\xRightarrow[lm]{8}$ ID PLUS ID TIMES ID

语法树的建立



语法树 --- Example 1/2

最左推导

exp

$\xRightarrow[lm]{1}$ exp PLUS term

$\xRightarrow[lm]{2}$ term PLUS term

$\xRightarrow[lm]{3}$ fac PLUS term

$\xRightarrow[lm]{4}$ ID PLUS term

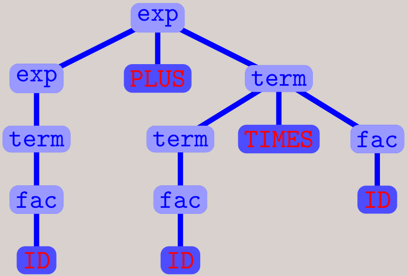
$\xRightarrow[lm]{5}$ ID PLUS term TIMES fac

$\xRightarrow[lm]{6}$ ID PLUS fac TIMES fac

$\xRightarrow[lm]{7}$ ID PLUS ID TIMES fac

$\xRightarrow[lm]{8}$ ID PLUS ID TIMES ID

语法树的建立



语法树 --- Example 2/2

最右推导

$$
\begin{aligned}
& exp \\
\xrightarrow[rm]{1}\ & exp\ \mathrm{PLUS}\ \underline{term} \\
\xrightarrow[rm]{2}\ & exp\ \mathrm{PLUS}\ term\ \mathrm{TIMES}\ \underline{fac} \\
\xrightarrow[rm]{3}\ & exp\ \mathrm{PLUS}\ \underline{term}\ \mathrm{TIMES}\ \mathrm{ID} \\
\xrightarrow[rm]{4}\ & exp\ \mathrm{PLUS}\ \underline{fac}\ \mathrm{TIMES}\ \mathrm{ID} \\
\xrightarrow[rm]{5}\ & \underline{exp}\ \mathrm{PLUS}\ \mathrm{ID}\ \mathrm{TIMES}\ \mathrm{ID} \\
\xrightarrow[rm]{6}\ & \underline{term}\ \mathrm{PLUS}\ \mathrm{ID}\ \mathrm{TIMES}\ \mathrm{ID} \\
\xrightarrow[rm]{7}\ & \underline{fac}\ \mathrm{PLUS}\ \mathrm{ID}\ \mathrm{TIMES}\ \mathrm{ID} \\
\xrightarrow[rm]{8}\ & \mathrm{ID}\ \mathrm{PLUS}\ \mathrm{ID}\ \mathrm{TIMES}\ \mathrm{ID}
\end{aligned}
$$

语法树的建立

语法树 --- Example 2/2

最右推导

$$
\begin{aligned}
& exp \\
\xrightarrow[rm]{1}\ & exp\ \mathrm{PLUS}\ \underline{term} \\
\xrightarrow[rm]{2}\ & exp\ \mathrm{PLUS}\ term\ \mathrm{TIMES}\ \underline{fac} \\
\xrightarrow[rm]{3}\ & exp\ \mathrm{PLUS}\ \underline{term}\ \mathrm{TIMES\ ID} \\
\xrightarrow[rm]{4}\ & exp\ \mathrm{PLUS}\ \underline{fac}\ \mathrm{TIMES\ ID} \\
\xrightarrow[rm]{5}\ & \underline{exp}\ \mathrm{PLUS\ ID\ TIMES\ ID} \\
\xrightarrow[rm]{6}\ & \underline{term}\ \mathrm{PLUS\ ID\ TIMES\ ID} \\
\xrightarrow[rm]{7}\ & \underline{fac}\ \mathrm{PLUS\ ID\ TIMES\ ID} \\
\xrightarrow[rm]{8}\ & \mathrm{ID\ PLUS\ ID\ TIMES\ ID}
\end{aligned}
$$

语法树的建立

exp

语法树 --- Example 2/2

最右推导

exp

$\overset{1}{\underset{rm}{\Rightarrow}}$ exp PLUS term

$\overset{2}{\underset{rm}{\Rightarrow}}$ exp PLUS term TIMES fac

$\overset{3}{\underset{rm}{\Rightarrow}}$ exp PLUS term TIMES ID

$\overset{4}{\underset{rm}{\Rightarrow}}$ exp PLUS fac TIMES ID

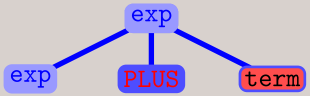
$\overset{5}{\underset{rm}{\Rightarrow}}$ exp PLUS ID TIMES ID

$\overset{6}{\underset{rm}{\Rightarrow}}$ term PLUS ID TIMES ID

$\overset{7}{\underset{rm}{\Rightarrow}}$ fac PLUS ID TIMES ID

$\overset{8}{\underset{rm}{\Rightarrow}}$ ID PLUS ID TIMES ID

语法树的建立



语法树 --- Example 2/2

最右推导

exp

$\xRightarrow[rm]{1} \text{exp PLUS } \underline{\text{term}}$

$\xRightarrow[rm]{2} \text{exp PLUS term TIMES } \underline{\text{fac}}$

$\xRightarrow[rm]{3} \text{exp PLUS } \underline{\text{term}} \text{ TIMES ID}$

$\xRightarrow[rm]{4} \text{exp PLUS } \underline{\text{fac}} \text{ TIMES ID}$

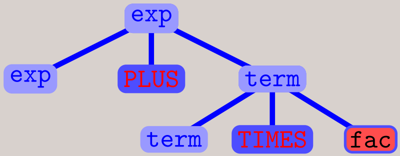
$\xRightarrow[rm]{5} \underline{\text{exp}} \text{ PLUS ID TIMES ID}$

$\xRightarrow[rm]{6} \underline{\text{term}} \text{ PLUS ID TIMES ID}$

$\xRightarrow[rm]{7} \underline{\text{fac}} \text{ PLUS ID TIMES ID}$

$\xRightarrow[rm]{8} \text{ID PLUS ID TIMES ID}$

语法树的建立



语法树 --- Example 2/2

最右推导

exp

$\xRightarrow[rm]{1}$ exp PLUS term

$\xRightarrow[rm]{2}$ exp PLUS term TIMES fac

$\xRightarrow[rm]{3}$ exp PLUS term TIMES ID

$\xRightarrow[rm]{4}$ exp PLUS fac TIMES ID

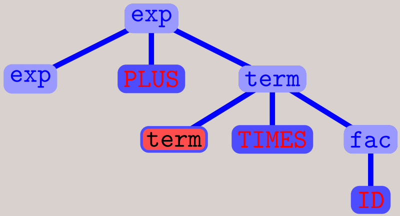
$\xRightarrow[rm]{5}$ exp PLUS ID TIMES ID

$\xRightarrow[rm]{6}$ term PLUS ID TIMES ID

$\xRightarrow[rm]{7}$ fac PLUS ID TIMES ID

$\xRightarrow[rm]{8}$ ID PLUS ID TIMES ID

语法树的建立



语法树 --- Example 2/2

最右推导

exp

$\xRightarrow[rm]{1}$ exp PLUS term

$\xRightarrow[rm]{2}$ exp PLUS term TIMES fac

$\xRightarrow[rm]{3}$ exp PLUS term TIMES ID

$\xRightarrow[rm]{4}$ exp PLUS fac TIMES ID

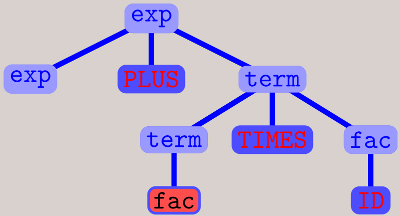
$\xRightarrow[rm]{5}$ exp PLUS ID TIMES ID

$\xRightarrow[rm]{6}$ term PLUS ID TIMES ID

$\xRightarrow[rm]{7}$ fac PLUS ID TIMES ID

$\xRightarrow[rm]{8}$ ID PLUS ID TIMES ID

语法树的建立



语法树 --- Example 2/2

最右推导

exp

$\xRightarrow[rm]{1}$ exp PLUS term

$\xRightarrow[rm]{2}$ exp PLUS term TIMES fac

$\xRightarrow[rm]{3}$ exp PLUS term TIMES ID

$\xRightarrow[rm]{4}$ exp PLUS fac TIMES ID

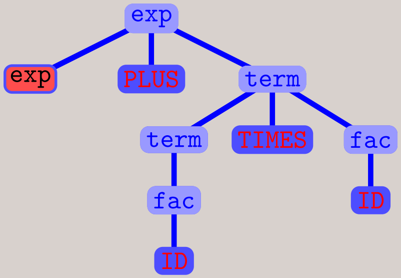
$\xRightarrow[rm]{5}$ exp PLUS ID TIMES ID

$\xRightarrow[rm]{6}$ term PLUS ID TIMES ID

$\xRightarrow[rm]{7}$ fac PLUS ID TIMES ID

$\xRightarrow[rm]{8}$ ID PLUS ID TIMES ID

语法树的建立



语法树 --- Example 2/2

最右推导

exp

$\xRightarrow[rm]{1} \text{exp PLUS } \underline{\text{term}}$

$\xRightarrow[rm]{2} \text{exp PLUS term TIMES } \underline{\text{fac}}$

$\xRightarrow[rm]{3} \text{exp PLUS } \underline{\text{term}} \text{ TIMES ID}$

$\xRightarrow[rm]{4} \text{exp PLUS } \underline{\text{fac}} \text{ TIMES ID}$

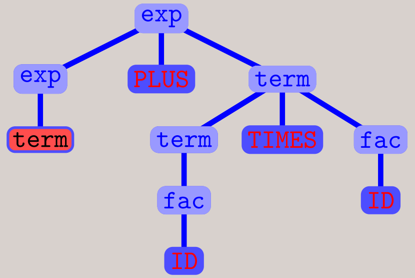
$\xRightarrow[rm]{5} \underline{\text{exp}} \text{ PLUS ID TIMES ID}$

$\xRightarrow[rm]{6} \underline{\text{term}} \text{ PLUS ID TIMES ID}$

$\xRightarrow[rm]{7} \underline{\text{fac}} \text{ PLUS ID TIMES ID}$

$\xRightarrow[rm]{8} \text{ID PLUS ID TIMES ID}$

语法树的建立



语法树 --- Example 2/2

最右推导

exp

$\xRightarrow[rm]{1} \text{exp PLUS } \underline{\text{term}}$

$\xRightarrow[rm]{2} \text{exp PLUS term TIMES } \underline{\text{fac}}$

$\xRightarrow[rm]{3} \text{exp PLUS } \underline{\text{term}} \text{ TIMES ID}$

$\xRightarrow[rm]{4} \text{exp PLUS } \underline{\text{fac}} \text{ TIMES ID}$

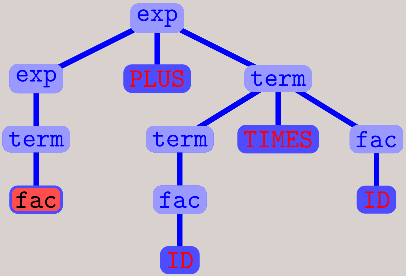
$\xRightarrow[rm]{5} \underline{\text{exp}} \text{ PLUS ID TIMES ID}$

$\xRightarrow[rm]{6} \underline{\text{term}} \text{ PLUS ID TIMES ID}$

$\xRightarrow[rm]{7} \underline{\text{fac}} \text{ PLUS ID TIMES ID}$

$\xRightarrow[rm]{8} \text{ID PLUS ID TIMES ID}$

语法树的建立

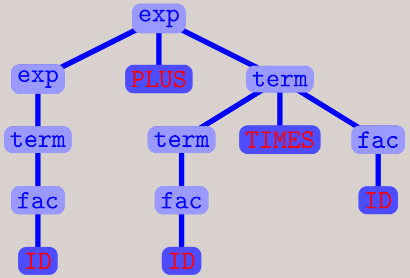


语法树 --- Example 2/2

最右推导

$$
\begin{aligned}
& \mathrm{exp} \\
\xRightarrow[rm]{1}\ & \mathrm{exp}\ \mathtt{PLUS}\ \underline{\mathrm{term}} \\
\xRightarrow[rm]{2}\ & \mathrm{exp}\ \mathtt{PLUS}\ \mathrm{term}\ \mathtt{TIMES}\ \underline{\mathrm{fac}} \\
\xRightarrow[rm]{3}\ & \mathrm{exp}\ \mathtt{PLUS}\ \underline{\mathrm{term}}\ \mathtt{TIMES}\ \mathtt{ID} \\
\xRightarrow[rm]{4}\ & \mathrm{exp}\ \mathtt{PLUS}\ \underline{\mathrm{fac}}\ \mathtt{TIMES}\ \mathtt{ID} \\
\xRightarrow[rm]{5}\ & \underline{\mathrm{exp}}\ \mathtt{PLUS}\ \mathtt{ID}\ \mathtt{TIMES}\ \mathtt{ID} \\
\xRightarrow[rm]{6}\ & \underline{\mathrm{term}}\ \mathtt{PLUS}\ \mathtt{ID}\ \mathtt{TIMES}\ \mathtt{ID} \\
\xRightarrow[rm]{7}\ & \underline{\mathrm{fac}}\ \mathtt{PLUS}\ \mathtt{ID}\ \mathtt{TIMES}\ \mathtt{ID} \\
\xRightarrow[rm]{8}\ & \mathtt{ID}\ \mathtt{PLUS}\ \mathtt{ID}\ \mathtt{TIMES}\ \mathtt{ID}
\end{aligned}
$$

语法树的建立



语法树的定义

递归定义

- 设 S 文法开始符号，则 S 是语法树 (语法分析树，Parse Tree).
- 设 T 是语法树，$A \in N$ 是 T 的树叶，设 $A \to X_1X_2\cdots X_m \in P$，则在 A 下从左到右加上标号分别为 $X_1, X_2, \cdots, X_m$ 的 m 个儿子所得到树是语法树.

语法树的定义

递归定义

- 设 S 文法开始符号，则 S 是语法树 (语法分析树，Parse Tree).
- 设 T 是语法树，$A \in N$ 是 T 的树叶，设 $A \to X_1X_2\cdots X_m \in P$，则在 A 下从左到右加上标号分别为 $X_1, X_2, \cdots, X_m$ 的 m 个儿子所得到树是语法树.

语法树的定义

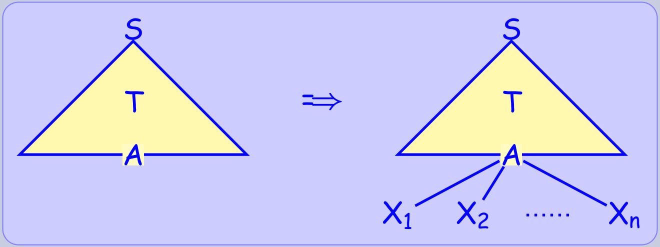
递归定义

- 设 S 文法开始符号，则 S 是语法树 (语法分析树，Parse Tree).
- 设 T 是语法树，$A \in N$ 是 T 的树叶，
  设 $A \to X_1X_2\cdots X_m \in P$，则在 A 下从左到右加上标号分别为 $X_1, X_2, \cdots, X_m$ 的 m 个儿子所得到树是语法树.

语法树的定义

递归定义

- 设 S 文法开始符号，则 S 是语法树 (语法分析树，Parse Tree).
- 设 T 是语法树，$A \in N$ 是 T 的树叶，设 $A \to X_1X_2\cdots X_m \in P$，则在 A 下从左到右加上标号分别为 $X_1, X_2, \cdots, X_m$ 的 m 个儿子所得到树是语法树.



语法树与推导的关系

- 语法树的内部节点一定是非终结符.
- 语法树过滤掉了推导过程对非终结符选择策略.
- 一个推导过程等价于对语法树的语法树的遍历过程，即：$S \overset{*}{\Rightarrow} \alpha$是一个推导 iff $\exists$ 语法树 T, 其叶结点从左到右循序排列为 α.
- 最左推导是前序遍历 (树根 $\rightarrow$ 左儿子 $\rightarrow$ 右儿子). 最右推导是 NRL：树根 $\rightarrow$ 右儿子 $\rightarrow$ 左儿子.
- α 是最左句型 ($S \underset{lm}{\overset{*}{\Rightarrow}} \alpha$) iff 存在一颗语法树，其前序遍历得到的叶结点序列是 α.
- α 是最右句型 ($S \underset{rm}{\overset{*}{\Rightarrow}} \alpha$) iff 存在一颗语法树，其 NRL 遍历得到的叶结点序列是 α.
- 一颗语法树对应唯一的一个最左推导和唯一的一个最右推导.
- 语法分析也可以看成建立语法树的过程.

语法树与推导的关系

- 语法树的内部节点一定是非终结符.
- 语法树过滤掉了推导过程对非终结符选择策略.
- 一个推导过程等价于对语法树的语法树的遍历过程，即: $S \overset{*}{\Rightarrow} \alpha$ 是一个推导 iff $\exists$ 语法树 T, 其叶结点从左到右循序排列为 α.
- 最左推导是前序遍历 (树根 $\rightarrow$ 左儿子 $\rightarrow$ 右儿子). 最右推导是 NRL: 树根 $\rightarrow$ 右儿子 $\rightarrow$ 左儿子.
- α 是最左句型 ($S \overset{*}{\underset{lm}{\Rightarrow}} \alpha$) iff 存在一颗语法树，其前序遍历得到的叶结点序列是 α.
- α 是最右句型 ($S \overset{*}{\underset{rm}{\Rightarrow}} \alpha$) iff 存在一颗语法树，其 NRL 遍历得到的叶结点序列是 α.
- 一颗语法树对应唯一的一个最左推导和唯一的一个最右推导.
- 语法分析也可以看成建立语法树的过程.

语法树与推导的关系

- 语法树的内部节点一定是非终结符.
- 语法树过滤掉了推导过程对非终结符选择策略.
- 一个推导过程等价于对语法树的语法树的遍历过程，即：$S \xRightarrow{*} \alpha$ 是一个推导 iff $\exists$ 语法树 T, 其叶结点从左到右循序排列为 α.
- 最左推导是前序遍历 (树根 $\rightarrow$ 左儿子 $\rightarrow$ 右儿子). 最右推导是 NRL：树根 $\rightarrow$ 右儿子 $\rightarrow$ 左儿子.
- α 是最左句型 ($S \xRightarrow[lm]{*} \alpha$) iff 存在一颗语法树，其前序遍历得到的叶结点序列是 α.
- α 是最右句型 ($S \xRightarrow[rm]{*} \alpha$) iff 存在一颗语法树，其 NRL 遍历得到的叶结点序列是 α.
- 一颗语法树对应唯一的一个最左推导和唯一的一个最右推导.
- 语法分析也可以看成建立语法树的过程.

语法树与推导的关系

- 语法树的内部节点一定是非终结符.
- 语法树过滤掉了推导过程对非终结符选择策略.
- 一个推导过程等价于对语法树的语法树的遍历过程，即：$S \stackrel{*}{\Rightarrow} \alpha$是一个推导 iff $\exists$ 语法树 T, 其叶结点从左到右循序排列为 α.
- 最左推导是前序遍历 (树根 $\rightarrow$ 左儿子 $\rightarrow$ 右儿子). 最右推导是 NRL：树根 $\rightarrow$ 右儿子 $\rightarrow$ 左儿子.
- α 是最左句型 ($S \underset{lm}{\stackrel{*}{\Rightarrow}} \alpha$) iff 存在一颗语法树，其前序遍历得到的叶结点序列是 α.
- α 是最右句型 ($S \underset{rm}{\stackrel{*}{\Rightarrow}} \alpha$) iff 存在一颗语法树，其 NRL 遍历得到的叶结点序列是 α.
- 一颗语法树对应唯一的一个最左推导和唯一的一个最右推导.
- 语法分析也可以看成建立语法树的过程.

语法树与推导的关系

- 语法树的内部节点一定是非终结符.
- 语法树过滤掉了推导过程对非终结符选择策略.
- 一个推导过程等价于对语法树的语法树的遍历过程，即：$S \overset{*}{\Rightarrow} \alpha$是一个推导 iff $\exists$ 语法树 T, 其叶结点从左到右循序排列为 α.
- 最左推导是前序遍历 (树根 $\rightarrow$ 左儿子 $\rightarrow$ 右儿子). 最右推导是 NRL：树根 $\rightarrow$ 右儿子 $\rightarrow$ 左儿子.
- α 是最左句型 ($S \overset{*}{\underset{lm}{\Rightarrow}} \alpha$) iff 存在一颗语法树，其前序遍历得到的叶结点序列是 α.
- α 是最右句型 ($S \overset{*}{\underset{rm}{\Rightarrow}} \alpha$) iff 存在一颗语法树，其 NRL 遍历得到的叶结点序列是 α.
- 一颗语法树对应唯一的一个最左推导和唯一的一个最右推导.
- 语法分析也可以看成建立语法树的过程.

语法树与推导的关系

- 语法树的内部节点一定是非终结符.
- 语法树过滤掉了推导过程对非终结符选择策略.
- 一个推导过程等价于对语法树的语法树的遍历过程，即：$S \overset{*}{\Rightarrow} \alpha$是一个推导 iff $\exists$ 语法树 T, 其叶结点从左到右循序排列为 α.
- 最左推导是前序遍历 (树根 $\rightarrow$ 左儿子 $\rightarrow$ 右儿子). 最右推导是 NRL：树根 $\rightarrow$ 右儿子 $\rightarrow$ 左儿子.
- α 是最左句型 ($S \underset{lm}{\overset{*}{\Rightarrow}} \alpha$) iff 存在一颗语法树，其前序遍历得到的叶结点序列是 α.
- α 是最右句型 ($S \underset{rm}{\overset{*}{\Rightarrow}} \alpha$) iff 存在一颗语法树，其 NRL 遍历得到的叶结点序列是 α.
- 一颗语法树对应唯一的一个最左推导和唯一的一个最右推导.
- 语法分析也可以看成建立语法树的过程.

语法树与推导的关系

- 语法树的内部节点一定是非终结符.
- 语法树过滤掉了推导过程对非终结符选择策略.
- 一个推导过程等价于对语法树的语法树的遍历过程，即：$S \overset{*}{\Rightarrow} \alpha$是一个推导 iff $\exists$ 语法树 T, 其叶结点从左到右循序排列为 α.
- 最左推导是前序遍历 (树根 $\rightarrow$ 左儿子 $\rightarrow$ 右儿子). 最右推导是 NRL：树根 $\rightarrow$ 右儿子 $\rightarrow$ 左儿子.
- α 是最左句型 ($S \underset{lm}{\overset{*}{\Rightarrow}} \alpha$) iff 存在一颗语法树，其前序遍历得到的叶结点序列是 α.
- α 是最右句型 ($S \underset{rm}{\overset{*}{\Rightarrow}} \alpha$) iff 存在一颗语法树，其 NRL 遍历得到的叶结点序列是 α.
- 一颗语法树对应唯一的一个最左推导和唯一的一个最右推导.
- 语法分析也可以看成建立语法树的过程.

语法树与推导的关系

- 语法树的内部节点一定是非终结符.
- 语法树过滤掉了推导过程对非终结符选择策略.
- 一个推导过程等价于对语法树的语法树的遍历过程，即：$S \stackrel{*}{\Rightarrow} \alpha$是一个推导 iff $\exists$ 语法树 T, 其叶结点从左到右循序排列为 α.
- 最左推导是前序遍历 (树根 $\rightarrow$ 左儿子 $\rightarrow$ 右儿子). 最右推导是 NRL：树根 $\rightarrow$ 右儿子 $\rightarrow$ 左儿子.
- α 是最左句型 ($S \underset{lm}{\stackrel{*}{\Rightarrow}} \alpha$) iff 存在一颗语法树，其前序遍历得到的叶结点序列是 α.
- α 是最右句型 ($S \underset{rm}{\stackrel{*}{\Rightarrow}} \alpha$) iff 存在一颗语法树，其 NRL 遍历得到的叶结点序列是 α.
- 一颗语法树对应唯一的一个最左推导和唯一的一个最右推导.
- 语法分析也可以看成建立语法树的过程.

语法树与推导的关系

- 语法树的内部节点一定是非终结符.
- 语法树过滤掉了推导过程对非终结符选择策略.
- 一个推导过程等价于对语法树的语法树的遍历过程，即:
  $S \stackrel{*}{\Rightarrow} \alpha$是一个推导 iff $\exists$ 语法树 T, 其叶结点从左到右循序排列为 α.
- 最左推导是前序遍历 (树根 $\rightarrow$ 左儿子 $\rightarrow$ 右儿子). 最右推导是 NRL: 树根 $\rightarrow$ 右儿子 $\rightarrow$ 左儿子.
- α 是最左句型 ($S \underset{lm}{\stackrel{*}{\Rightarrow}} \alpha$) iff 存在一颗语法树，其前序遍历得到的叶结点序列是 α.
- α 是最右句型 ($S \underset{rm}{\stackrel{*}{\Rightarrow}} \alpha$) iff 存在一颗语法树，其 NRL 遍历得到的叶结点序列是 α.
- 一颗语法树对应唯一的一个最左推导和唯一的一个最右推导.
- 语法分析也可以看成建立语法树的过程.

Example

表达式文法法

$$\begin{array}{rcl} \mathrm{exp} & \rightarrow & \mathrm{exp}\ \mathtt{PLUS}\ \mathrm{exp} \\ & | & \mathrm{exp}\ \mathtt{TIMES}\ \mathrm{exp} \\ & | & \mathtt{LP}\ \mathrm{exp}\ \mathtt{RP} \\ & | & \mathtt{ID} \end{array}$$

两个不同的最左推导

$$\begin{array}{rl} \mathrm{exp} & \underset{lm}{\Rightarrow} \mathrm{exp}\ \mathtt{PLUS}\ \mathrm{exp} \\ & \underset{lm}{\Rightarrow} \mathtt{ID}\ \mathtt{PLUS}\ \mathrm{exp} \\ & \underset{lm}{\Rightarrow} \mathtt{ID}\ \mathtt{PLUS}\ \mathrm{exp}\ \mathtt{TIMES}\ \mathrm{exp} \\ & \underset{lm}{\Rightarrow} \mathtt{ID}\ \mathtt{PLUS}\ \mathtt{ID}\ \mathtt{TIMES}\ \mathrm{exp} \\ & \underset{lm}{\Rightarrow} \mathtt{ID}\ \mathtt{PLUS}\ \mathtt{ID}\ \mathtt{TIMES}\ \mathtt{ID} \end{array}$$

$$\begin{array}{rl} \mathrm{exp} & \underset{lm}{\Rightarrow} \mathrm{exp}\ \mathtt{TIMES}\ \mathrm{exp} \\ & \underset{lm}{\Rightarrow} \mathrm{exp}\ \mathtt{PLUS}\ \mathrm{exp}\ \mathtt{TIMES}\ \mathrm{exp} \\ & \underset{lm}{\Rightarrow} \mathtt{ID}\ \mathtt{PLUS}\ \mathrm{exp}\ \mathtt{TIMES}\ \mathrm{exp} \\ & \underset{lm}{\Rightarrow} \mathtt{ID}\ \mathtt{PLUS}\ \mathtt{ID}\ \mathtt{TIMES}\ \mathrm{exp} \\ & \underset{lm}{\Rightarrow} \mathtt{ID}\ \mathtt{PLUS}\ \mathtt{ID}\ \mathtt{TIMES}\ \mathtt{ID} \end{array}$$

Example

表达式文法

$$
\begin{aligned}
\mathit{exp} &\rightarrow \mathit{exp}\ \mathrm{PLUS}\ \mathit{exp} \\
&\mid \mathit{exp}\ \mathrm{TIMES}\ \mathit{exp} \\
&\mid \mathrm{LP}\ \mathit{exp}\ \mathrm{RP} \\
&\mid \mathrm{ID}
\end{aligned}
$$

两个不同的最左推导

$$
\begin{aligned}
\mathit{exp} &\underset{lm}{\Rightarrow} \underline{\mathit{exp}}\ \mathrm{PLUS}\ \mathit{exp} \\
&\underset{lm}{\Rightarrow} \mathrm{ID}\ \mathrm{PLUS}\ \underline{\mathit{exp}} \\
&\underset{lm}{\Rightarrow} \mathrm{ID}\ \mathrm{PLUS}\ \underline{\mathit{exp}}\ \mathrm{TIMES}\ \mathit{exp} \\
&\underset{lm}{\Rightarrow} \mathrm{ID}\ \mathrm{PLUS}\ \mathrm{ID}\ \mathrm{TIMES}\ \underline{\mathit{exp}} \\
&\underset{lm}{\Rightarrow} \mathrm{ID}\ \mathrm{PLUS}\ \mathrm{ID}\ \mathrm{TIMES}\ \mathrm{ID}
\end{aligned}
$$

$$
\begin{aligned}
\mathit{exp} &\underset{lm}{\Rightarrow} \underline{\mathit{exp}}\ \mathrm{TIMES}\ \mathit{exp} \\
&\underset{lm}{\Rightarrow} \underline{\mathit{exp}}\ \mathrm{PLUS}\ \mathit{exp}\ \mathrm{TIMES}\ \mathit{exp} \\
&\underset{lm}{\Rightarrow} \mathrm{ID}\ \mathrm{PLUS}\ \underline{\mathit{exp}}\ \mathrm{TIMES}\ \mathit{exp} \\
&\underset{lm}{\Rightarrow} \mathrm{ID}\ \mathrm{PLUS}\ \mathrm{ID}\ \mathrm{TIMES}\ \underline{\mathit{exp}} \\
&\underset{lm}{\Rightarrow} \mathrm{ID}\ \mathrm{PLUS}\ \mathrm{ID}\ \mathrm{TIMES}\ \mathrm{ID}
\end{aligned}
$$

Example

表达式文法法

$$
\begin{array}{rcl}
exp & \rightarrow & exp \; \text{PLUS} \; exp \\
 & | & exp \; \text{TIMES} \; exp \\
 & | & \text{LP} \; exp \; \text{RP} \\
 & | & \text{ID}
\end{array}
$$

两个不同的最左推导

$$
\begin{array}{rl}
exp & \underset{lm}{\Rightarrow} \underline{exp} \; \text{PLUS} \; exp \\
 & \underset{lm}{\Rightarrow} \text{ID PLUS} \; \underline{exp} \\
 & \underset{lm}{\Rightarrow} \text{ID PLUS} \; \underline{exp} \; \text{TIMES} \; exp \\
 & \underset{lm}{\Rightarrow} \text{ID PLUS ID TIMES} \; \underline{exp} \\
 & \underset{lm}{\Rightarrow} \text{ID PLUS ID TIMES ID}
\end{array}
$$

$$
\begin{array}{rl}
exp & \underset{lm}{\Rightarrow} \underline{exp} \; \text{TIMES} \; exp \\
 & \underset{lm}{\Rightarrow} \underline{exp} \; \text{PLUS} \; exp \; \text{TIMES} \; exp \\
 & \underset{lm}{\Rightarrow} \text{ID PLUS} \; \underline{exp} \; \text{TIMES} \; exp \\
 & \underset{lm}{\Rightarrow} \text{ID PLUS ID TIMES} \; \underline{exp} \\
 & \underset{lm}{\Rightarrow} \text{ID PLUS ID TIMES ID}
\end{array}
$$

Example

表达式文法法

$$
\begin{array}{lcl}
exp & \rightarrow & exp \ \mathrm{PLUS} \ exp \\
 & \mid & exp \ \mathrm{TIMES} \ exp \\
 & \mid & \mathrm{LP} \ exp \ \mathrm{RP} \\
 & \mid & \mathrm{ID}
\end{array}
$$

两个不同的最左推导

$$
\begin{array}{lcl}
exp & \underset{lm}{\Rightarrow} & \underline{exp} \ \mathrm{PLUS} \ exp \\
 & \underset{lm}{\Rightarrow} & \mathrm{ID} \ \mathrm{PLUS} \ \underline{exp} \\
 & \underset{lm}{\Rightarrow} & \mathrm{ID} \ \mathrm{PLUS} \ \underline{exp} \ \mathrm{TIMES} \ exp \\
 & \underset{lm}{\Rightarrow} & \mathrm{ID} \ \mathrm{PLUS} \ \mathrm{ID} \ \mathrm{TIMES} \ \underline{exp} \\
 & \underset{lm}{\Rightarrow} & \mathrm{ID} \ \mathrm{PLUS} \ \mathrm{ID} \ \mathrm{TIMES} \ \mathrm{ID}
\end{array}
$$

$$
\begin{array}{lcl}
exp & \underset{lm}{\Rightarrow} & \underline{exp} \ \mathrm{TIMES} \ exp \\
 & \underset{lm}{\Rightarrow} & \underline{exp} \ \mathrm{PLUS} \ exp \ \mathrm{TIMES} \ exp \\
 & \underset{lm}{\Rightarrow} & \mathrm{ID} \ \mathrm{PLUS} \ \underline{exp} \ \mathrm{TIMES} \ exp \\
 & \underset{lm}{\Rightarrow} & \mathrm{ID} \ \mathrm{PLUS} \ \mathrm{ID} \ \mathrm{TIMES} \ \underline{exp} \\
 & \underset{lm}{\Rightarrow} & \mathrm{ID} \ \mathrm{PLUS} \ \mathrm{ID} \ \mathrm{TIMES} \ \mathrm{ID}
\end{array}
$$

Example

表达式文法

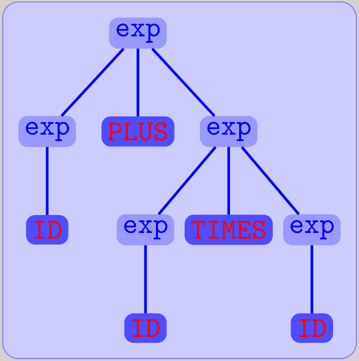
$$
\begin{aligned}
exp &\rightarrow exp \text{ PLUS } exp \\
&\mid exp \text{ TIMES } exp \\
&\mid \text{LP } exp \text{ RP} \\
&\mid \text{ID}
\end{aligned}
$$

两个不同的最左推导

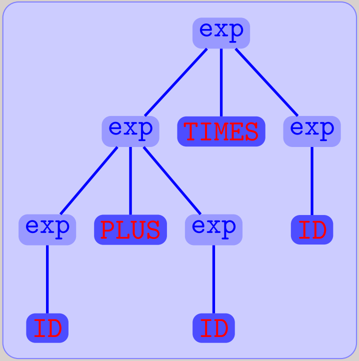
$$
\begin{aligned}
exp &\underset{lm}{\Rightarrow} \underline{exp} \text{ PLUS } exp \\
&\underset{lm}{\Rightarrow} \text{ID PLUS } \underline{exp} \\
&\underset{lm}{\Rightarrow} \text{ID PLUS } \underline{exp} \text{ TIMES } exp \\
&\underset{lm}{\Rightarrow} \text{ID PLUS ID TIMES } \underline{exp} \\
&\underset{lm}{\Rightarrow} \text{ID PLUS ID TIMES ID}
\end{aligned}
$$

$$
\begin{aligned}
exp &\underset{lm}{\Rightarrow} \underline{exp} \text{ TIMES } exp \\
&\underset{lm}{\Rightarrow} \underline{exp} \text{ PLUS } exp \text{ TIMES } exp \\
&\underset{lm}{\Rightarrow} \text{ID PLUS } \underline{exp} \text{ TIMES } exp \\
&\underset{lm}{\Rightarrow} \text{ID PLUS ID TIMES } \underline{exp} \\
&\underset{lm}{\Rightarrow} \text{ID PLUS ID TIMES ID}
\end{aligned}
$$

Example --- 两颗不同的语法树



(a) 乘法运算先结合



(b) 加法运算先结合

二义性 (Ambiguity)

二义性的定义

称 CFG G 是二义性文法 iff 存在一语句有两颗不同的语法树 iff 存在一语句有两个不同的最左 (右) 推导.

注释

- 二义性导致了语法分析的不确定性，同样的语句有不同的理解，在文法设计时要避免；

二义性 (Ambiguity)

二义性的定义

称 CFG G 是**二义性文法** iff 存在一语句有两颗不同的语法树
iff 存在一语句有两个不同的最左 (右) 推导.

注释

- 二义性导致了语法分析的不确定性，同样的语句有不同的理解，在文法设计时要避免：
  - Ex: C 语言数组元素的引用 ``A[ ]'' 和函数调用 ``A( )''.
  - Ex: C 语言的 ``A (B)''：函数调用或者变量说明?
- 二义性文法一定没有确定的语法分析 (无回溯).
- 可以通过特定的分析规则在语法分析过程中解消二义性.
- 其他领域也有二义性，如面向对象的 diamond problem.

二义性 (Ambiguity)

二义性的定义

称 CFG G 是二义性文法 iff 存在一语句有两颗不同的语法树 iff 存在一语句有两个不同的最左(右)推导.

注释

- 二义性导致了语法分析的不确定性，同样的语句有不同的理解，在文法设计时要避免：
  - Ex: C 语言数组元素的引用 ``A[ ]'' 和函数调用 ``A( )''.
  - Ex: C 语言的 ``A (B)'': 函数调用或者变量说明?
- 二义性文法一定没有确定的语法分析(无回溯).
- 可以通过特定的分析规则在语法分析过程中解消二义性.
- 其他领域也有二义性，如面向对象的 diamond problem.

二义性 (Ambiguity)

二义性的定义

称 CFG G 是**二义性文法** iff 存在一语句有两颗不同的语法树 iff 存在一语句有两个不同的最左(右)推导.

注释

- 二义性导致了语法分析的不确定性，同样的语句有不同的理解，在文法设计时要避免：
  - Ex: C 语言数组元素的引用 ``A[ ]'' 和函数调用 ``A( )''.
  - Ex: C 语言的 ``A (B)'': 函数调用或者变量说明?
- 二义性文法一定没有确定的语法分析(无回溯).
- 可以通过特定的分析规则在语法分析过程中解消二义性.
- 其他领域也有二义性，如面向对象的 diamond problem.

二义性 (Ambiguity)

二义性的定义

称 CFG G 是 二义性文法 iff 存在一语句有两颗不同的语法树 iff 存在一语句有两个不同的最左(右)推导.

注释

- 二义性导致了语法分析的不确定性，同样的语句有不同的理解，在文法设计时要避免：
  - Ex: C 语言数组元素的引用 ``A[ ]'' 和函数调用 ``A( )''.
  - Ex: C 语言的 ``A (B)''：函数调用或者变量说明?
- 二义性文法一定没有确定的语法分析(无回溯).
- 可以通过特定的分析规则在语法分析过程中解消二义性.
  - Ex: XL 无二义性文法，而 XL' 是二义文法.
  - 二义性在 XL' 上表现为运算符的优先级别和结合次序.
  - 二义性文法要比无二义性文法要简洁.
- 其他领域也有二义性，如面向对象的 diamond problem.

二义性 (Ambiguity)

二义性的定义

称 CFG G 是**二义性文法** iff 存在一语句有两颗不同的语法树 iff 存在一语句有两个不同的最左(右)推导.

注释

- 二义性导致了语法分析的不确定性，同样的语句有不同的理解，在文法设计时要避免：
  - Ex: C 语言数组元素的引用 ``A[ ]'' 和函数调用 ``A( )''.
  - Ex: C 语言的 ``A (B)'': 函数调用或者变量说明?
- 二义性文法一定没有确定的语法分析 (无回溯).
- 可以通过特定的分析规则在语法分析过程中解消二义性.
  - Ex: XL 无二义性文法，而 XL' 是二义文法.
  - 二义性在 XL' 上表现为运算的优先级别和结合次序.
  - 二义性文法要比无二义性文法要简洁.
- 其他领域也有二义性，如面向对象的 diamond problem.

二义性 (Ambiguity)

二义性的定义

称 CFG G 是二义性文法 iff 存在一语句有两颗不同的语法树 iff 存在一语句有两个不同的最左(右)推导.

注释

- 二义性导致了语法分析的不确定性，同样的语句有不同的理解，在文法设计时要避免：
  - Ex: C 语言数组元素的引用 ``A[ ]'' 和函数调用 ``A( )''.
  - Ex: C 语言的 ``A (B)''：函数调用或者变量说明?
- 二义性文法一定没有确定的语法分析 (无回溯).
- 可以通过特定的分析规则在语法分析过程中解消二义性.
  - Ex: XL 无二义性文法，而 XL' 是二义文法.
  - 二义性在 XL' 上表现为运算的优先级别和结合次序.
  - 二义性文法要比无二义性文法要简洁.
- 其他领域也有二义性，如面向对象的 diamond problem.

二义性 (Ambiguity)

二义性的定义

称 CFG G 是二义性文法 iff 存在一语句有两颗不同的语法树 iff 存在一语句有两个不同的最左 (右) 推导.

注释

- 二义性导致了语法分析的不确定性，同样的语句有不同的理解，在文法设计时要避免：
  - Ex: C 语言数组元素的引用 ``A[ ]'' 和函数调用 ``A( )''.
  - Ex: C 语言的 ``A (B)'': 函数调用或者变量说明?
- 二义性文法一定没有确定的语法分析 (无回溯).
- 可以通过特定的分析规则在语法分析过程中解消二义性.
  - Ex: XL 无二义性文法，而 XL' 是二义文法.
  - 二义性在 XL' 上表现为运算的优先级别和结合次序.
  - 二义性文法要比无二义性文法要简洁.
- 其他领域也有二义性，如面向对象的 diamond problem.

二义性 (Ambiguity)

二义性的定义

称 CFG G 是二义性文法 iff 存在一语句有两颗不同的语法树 iff 存在一语句有两个不同的最左(右)推导.

注释

- 二义性导致了语法分析的不确定性，同样的语句有不同的理解，在文法设计时要避免：
  - Ex: C 语言数组元素的引用 ``A[ ]'' 和函数调用 ``A( )''.
  - Ex: C 语言的 ``A (B)''：函数调用或者变量说明?
- 二义性文法一定没有确定的语法分析 (无回溯).
- 可以通过特定的分析规则在语法分析过程中解消二义性.
  - Ex: XL 无二义性文法，而 XL' 是二义文法.
  - 二义性在 XL' 上表现为运算的优先级别和结合次序.
  - 二义性文法要比无二义性文法要简洁.
- 其他领域也有二义性，如面向对象的 diamond problem.

二义性 (Ambiguity)

二义性的定义

称 CFG G 是二义性文法 iff 存在一语句有两颗不同的语法树 iff 存在一语句有两个不同的最左 (右) 推导.

注释

- 二义性导致了语法分析的不确定性，同样的语句有不同的理解，在文法设计时要避免：
  - Ex: C 语言数组元素的引用 ``A[ ]'' 和函数调用 ``A( )''.
  - Ex: C 语言的 ``A (B)''：函数调用或者变量说明?
- 二义性文法一定没有确定的语法分析 (无回溯).
- 可以通过特定的分析规则在语法分析过程中解消二义性.
  - Ex: XL 无二义性文法，而 XL' 是二义文法.
  - 二义性在 XL' 上表现为运算的优先级别和结合次序.
  - 二义性文法要比无二义性文法要简洁.
- 其他领域也有二义性，如面向对象的 diamond problem.

二义性 (Ambiguity)

二义性的定义

称 CFG G 是二义性文法 iff 存在一语句有两颗不同的语法树 iff 存在一语句有两个不同的最左 (右) 推导.

注释

- 二义性导致了语法分析的不确定性，同样的语句有不同的理解，在文法设计时要避免：
  - Ex: C 语言数组元素的引用 ``A[ ]'' 和函数调用 ``A( )''.
  - Ex: C 语言的 ``A (B)''： 函数调用或者变量说明?
- 二义性文法一定没有确定的语法分析 (无回溯).
- 可以通过特定的分析规则在语法分析过程中解消二义性.
  - Ex: XL 无二义性文法，而 XL' 是二义文法.
  - 二义性在 XL' 上表现为运算的优先级别和结合次序.
  - 二义性文法要比无二义性文法要简洁.
- 其他领域也有二义性，如面向对象的 diamond problem.

Example 1

$L(G) = \{$以字母 a 和 b 字符串，该串中 a 和 b 的出现次数相同$\}$

分析 $L(G)$ 语句的增长规律：

1. $\varepsilon \in L$.
2. if $w \in L$, then $awb, abw, baw, bwa, wab, wba \in L$.
3. 文法 G:

$$S \to aSb \mid abS \mid baS \mid bSa \mid Sab \mid Sba \mid \varepsilon$$

Example 1

$L(G) = \{$以字母 a 和 b 字符串，该串中 a 和 b 的出现次数相同$\}$

分析 $L(G)$ 语句的增长规律：

1. $\varepsilon \in L$.
2. if $w \in L$, then $awb, abw, baw, bwa, wab, wba \in L$.
3. 文法 G:

$$S \to aSb \mid abS \mid baS \mid bSa \mid Sab \mid Sba \mid \varepsilon$$

Example 1

$L(G) = \{$以字母 a 和 b 字符串，该串中 a 和 b 的出现次数相同$\}$
分析 $L(G)$ 语句的增长规律：

1. $\varepsilon \in L$.
2. if $w \in L$, then $awb, abw, baw, bwa, wab, wba \in L$.
3. 文法 G:

$$S \rightarrow aSb \mid abS \mid baS \mid bSa \mid Sab \mid Sba \mid \varepsilon$$

Example 1

$L(G) = \{$以字母 a 和 b 字符串，该串中 a 和 b 的出现次数相同$\}$

分析 $L(G)$ 语句的增长规律：

1. $\varepsilon \in L$.
2. if $w \in L$, then $awb, abw, baw, bwa, wab, wba \in L$.
3. 文法 G:

$$S \to aSb \mid abS \mid baS \mid bSa \mid Sab \mid Sba \mid \varepsilon$$

验证

验证

文法 G:

$$S \to aSb \mid abS \mid baS \mid bSa \mid Sab \mid Sba \mid \varepsilon$$

则 $a^n b^{2n} a^n \notin L(G)\,(n \geq 2)$，但是该串中 a 和 b 均出现 $2n$ 次.

句型分析:

1. 设第一步推导用 $S \to abS$.
2. 则 $bS \not\Rightarrow a^{n-1}b^{2n}a^n$，不可能.
3. 同理对其他产生式可以得出相同的结论.

验证

验证

文法 G:

$$S \to aSb \mid abS \mid baS \mid bSa \mid Sab \mid Sba \mid \varepsilon$$

则 $a^nb^{2n}a^n \notin L(G)\,(n \geq 2)$，但是该串中 a 和 b 均出现 2n 次.

句型分析:

1. 设第一步推导用 $S \to abS$.
2. 则 $bS \not\Rightarrow a^{n-1}b^{2n}a^n$, 不可能.
3. 同理对其他产生式可以得出相同的结论.

验证

验证

文法 G:

$$S \rightarrow aSb \mid abS \mid baS \mid bSa \mid Sab \mid Sba \mid \varepsilon$$

则 $a^n b^{2n} a^n \notin L(G)\,(n \geq 2)$，但是该串中 a 和 b 均出现 2n 次.

句型分析:

1. 设第一步推导用 $S \rightarrow abS$.
2. 则 $bS \not\Rightarrow a^{n-1}b^{2n}a^n$，不可能.
3. 同理对其他产生式可以得出相同的结论.

验证

验证

文法 G:

$$S \to aSb \mid abS \mid baS \mid bSa \mid Sab \mid Sba \mid \varepsilon$$

则 $a^n b^{2n} a^n \notin L(G)\,(n \geq 2)$，但是该串中 a 和 b 均出现 $2n$ 次.

句型分析:

1. 设第一步推导用 $S \to abS$.
2. 则 $bS \not\Rightarrow a^{n-1} b^{2n} a^n$，不可能.
3. 同理对其他产生式可以得出相同的结论.

语句分析

设 $x_1x_2\ldots x_n \in \{a,b\}^*$，定义函数 $f(i) = |x_1x_2\ldots x_i|_a - |x_1x_2\ldots x_i|_b$，即子串 $x_1x_2\ldots x_i$ 中 a 出现的次数减去 b 的次数，则 f 按步长 1 递增或递减。

$ax_2x_3\cdots x_{2n-1}a \in L$ 具有如下性质：

$$a \quad x_2 \quad x_3 \quad \ldots \quad x_i \quad \ldots \quad x_{2n-1} \quad a$$

$f(2n) = 0$

因此，$\exists s_1, s_2 \in L$, and, $ax_1x_2\cdots x_{2n}a = s_1s_2$.

语句分析

设 $x_1x_2\dots x_n \in \{a,b\}^*$，定义函数 $f(i) = |x_1x_2\dots x_i|_a - |x_1x_2\dots x_i|_b$，即子串 $x_1x_2\dots x_i$ 中 a 出现的次数减去 b 的次数，则 f 按步长 1 递增或递减。

$ax_2x_3\cdots x_{2n-1}a \in L$ 具有如下性质：

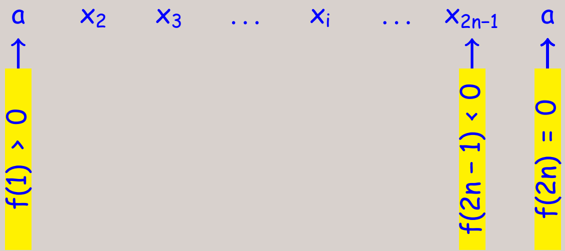
$$a \quad x_2 \quad x_3 \quad \dots \quad x_i \quad \dots \quad \underset{\substack{\uparrow \\ f(2n-1)<0}}{x_{2n-1}} \quad \underset{\substack{\uparrow \\ f(2n)=0}}{a}$$

因此，$\exists s_1, s_2 \in L, \text{and}, ax_1x_2\cdots x_{2n}a = s_1s_2$.

语句分析

设 $x_1x_2\ldots x_n \in \{a,b\}^*$，定义函数 $f(i) = |x_1x_2\ldots x_i|_a - |x_1x_2\ldots x_i|_b$，即子串 $x_1x_2\ldots x_i$ 中 a 出现的次数减去 b 的次数，则 f 按步长 1 递增或递减。

$ax_2x_3\cdots x_{2n-1}a \in L$ 具有如下性质：

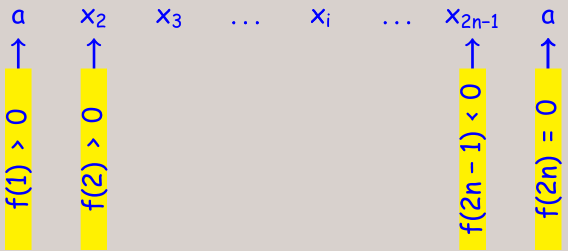


因此，$\exists s_1, s_2 \in L$, and, $ax_1x_2\cdots x_{2n}a = s_1s_2$.

语句分析

设 $x_1x_2\ldots x_n \in \{a,b\}^*$，定义函数 $f(i) = |x_1x_2\ldots x_i|_a - |x_1x_2\ldots x_i|_b$，即子串 $x_1x_2\ldots x_i$ 中 a 出现的次数减去 b 的次数，则 f 按步长 1 递增或递减。

$ax_2x_3\cdots x_{2n-1}a \in L$ 具有如下性质：

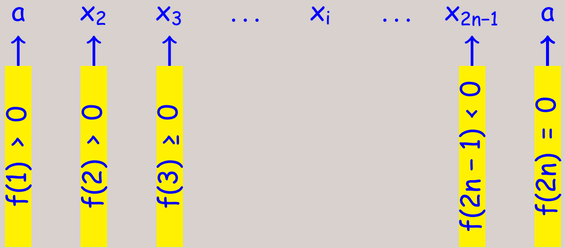


因此，$\exists s_1, s_2 \in L$, and, $ax_1x_2\cdots x_{2n}a = s_1s_2$.

语句分析

设 $x_1x_2\ldots x_n \in \{a,b\}^*$，定义函数 $f(i) = |x_1x_2\ldots x_i|_a - |x_1x_2\ldots x_i|_b$，即子串 $x_1x_2\ldots x_i$ 中 a 出现的次数减去 b 的次数，则 f 按步长 1 递增或递减。

$ax_2x_3\cdots x_{2n-1}a \in L$ 具有如下性质：

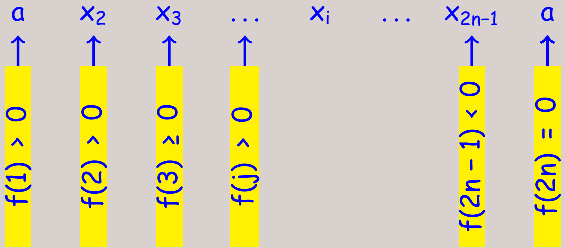


因此，$\exists s_1, s_2 \in L$, and, $ax_1x_2\cdots x_{2n}a = s_1s_2$.

语句分析

设 $x_1x_2\ldots x_n \in \{a,b\}^*$，定义函数 $f(i) = |x_1x_2\ldots x_i|_a - |x_1x_2\ldots x_i|_b$，即子串 $x_1x_2\ldots x_i$ 中 a 出现的次数减去 b 的次数，则 f 按步长 1 递增或递减。

$ax_2x_3\cdots x_{2n-1}a \in L$ 具有如下性质：

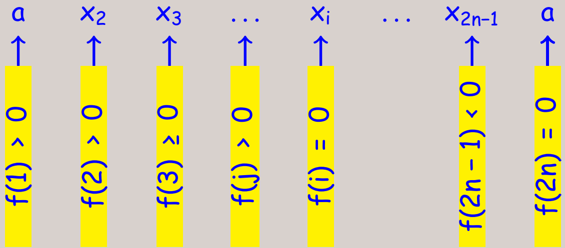


因此，$\exists s_1, s_2 \in L$, and, $ax_1x_2\cdots x_{2n}a = s_1s_2$.

语句分析

设 $x_1x_2\ldots x_n \in \{a,b\}^*$，定义函数 $f(i) = |x_1x_2\ldots x_i|_a - |x_1x_2\ldots x_i|_b$，即子串 $x_1x_2\ldots x_i$ 中 a 出现的次数减去 b 的次数，则 f 按步长 1 递增或递减。

$ax_2x_3\cdots x_{2n-1}a \in L$ 具有如下性质：



因此，$\exists s_1, s_2 \in L$, and, $ax_1x_2\cdots x_{2n}a = s_1s_2$.

语句分析

设 $x_1x_2\ldots x_n \in \{a,b\}^*$，定义函数 $f(i) = |x_1x_2\ldots x_i|_a - |x_1x_2\ldots x_i|_b$，即子串 $x_1x_2\ldots x_i$ 中 a 出现的次数减去 b 的次数，则 f 按步长 1 递增或递减。

$ax_2x_3\cdots x_{2n-1}a \in L$ 具有如下性质：

a	x_2	x_3	$\ldots$	x_i	$\ldots$	x_{2n-1}	a
$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$
$f(1) > 0$	$f(2) > 0$	$f(3) \geq 0$	$f(j) > 0$	$f(i) = 0$	$f(k) \lesseqgtr 0$	$f(2n-1) < 0$	$f(2n) = 0$

因此，$\exists s_1, s_2 \in L$, and, $ax_1x_2\cdots x_{2n}a = s_1s_2$.

语句分析

设 $x_1x_2\ldots x_n \in \{a,b\}^*$, 定义函数 $f(i) = |x_1x_2\ldots x_i|_a - |x_1x_2\ldots x_i|_b$, 即子串 $x_1x_2\ldots x_i$ 中 a 出现的次数减去 b 的次数, 则 f 按步长 1 递增或递减.

$ax_2x_3\cdots x_{2n-1}a \in L$ 具有如下性质:

a	x_2	x_3	$\ldots$	x_i	$\ldots$	x_{2n-1}	a
$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$
$f(1) > 0$	$f(2) > 0$	$f(3) \geq 0$	$f(j) > 0$	$f(i) = 0$	$f(k) \lesseqgtr 0$	$f(2n-1) < 0$	$f(2n) = 0$

因此, $\exists s_1, s_2 \in L, \text{and}, ax_1x_2\cdots x_{2n}a = s_1s_2$.

最后设计

新的文法 G

$$S \rightarrow aSb \mid bSa \mid SS \mid \varepsilon$$

注释

最后设计

新的文法 G

$$S \to aSb \mid bSa \mid SS \mid \varepsilon$$

注释

- 产生式的独立性：原文法 G 的产生式 $S \to abS$ 是新文法 G 的定理：
$$S \Rightarrow \underline{S}S \Rightarrow a\underline{S}bS \Rightarrow abS$$
- 文法是二义文法：
$$S \underset{lm}{\Rightarrow} a\underline{S}b \underset{lm}{\Rightarrow} ab\underline{S}ab \underset{lm}{\Rightarrow} abab$$
$$S \underset{lm}{\Rightarrow} \underline{S}S \underset{lm}{\Rightarrow} a\underline{S}bS \underset{lm}{\Rightarrow} ab\underline{S} \underset{lm}{\Rightarrow} aba\underline{S}b \underset{lm}{\Rightarrow} abab$$
- 等价文法：
$$S \to aSbS \mid bSaS \mid \varepsilon$$

最后设计

新的文法 G

$$S \to aSb \mid bSa \mid SS \mid \varepsilon$$

注释

- 产生式的独立性：原文法 G 的产生式 $S \to abS$ 是新文法 G 的定理：
$$S \Rightarrow \underline{S}S \Rightarrow a\underline{S}bS \Rightarrow abS$$
- 文法是二义文法：
$$S \underset{lm}{\Rightarrow} a\underline{S}b \underset{lm}{\Rightarrow} ab\underline{S}ab \underset{lm}{\Rightarrow} abab$$
$$S \underset{lm}{\Rightarrow} \underline{S}S \underset{lm}{\Rightarrow} a\underline{S}bS \underset{lm}{\Rightarrow} ab\underline{S} \underset{lm}{\Rightarrow} aba\underline{S}b \underset{lm}{\Rightarrow} abab$$
- 等价文法：
$$S \to aSbS \mid bSaS \mid \varepsilon$$

最后设计

新的文法 G

$$S \to aSb \mid bSa \mid SS \mid \varepsilon$$

注释

- 产生式的独立性：原文法 G 的产生式 $S \to abS$ 是新文法 G 的定理：
$$S \Rightarrow \underline{S}S \Rightarrow a\underline{S}bS \Rightarrow abS$$
- 文法是二义文法：
$$\begin{aligned} S &\underset{lm}{\Rightarrow} a\underline{S}b \underset{lm}{\Rightarrow} ab\underline{S}ab \underset{lm}{\Rightarrow} abab \\ S &\underset{lm}{\Rightarrow} \underline{S}S \underset{lm}{\Rightarrow} a\underline{S}bS \underset{lm}{\Rightarrow} ab\underline{S} \underset{lm}{\Rightarrow} aba\underline{S}b \underset{lm}{\Rightarrow} abab \end{aligned}$$
- 等价文法：
$$S \to aSbS \mid bSaS \mid \varepsilon$$

最后设计

新的文法 G

$$S \rightarrow aSb \mid bSa \mid SS \mid \varepsilon$$

注释

- 产生式的**独立性**：原文法 G 的产生式 $S \rightarrow abS$ 是新文法 G 的定理：

$$S \Rightarrow \underline{S}S \Rightarrow a\underline{S}bS \Rightarrow abS$$

- 文法是二义文法：

$$S \underset{lm}{\Rightarrow} a\underline{S}b \underset{lm}{\Rightarrow} ab\underline{S}ab \underset{lm}{\Rightarrow} abab$$

$$S \underset{lm}{\Rightarrow} \underline{S}S \underset{lm}{\Rightarrow} a\underline{S}bS \underset{lm}{\Rightarrow} ab\underline{S} \underset{lm}{\Rightarrow} aba\underline{S}b \underset{lm}{\Rightarrow} abab$$

- 等价文法：

$$S \rightarrow aSbS \mid bSaS \mid \varepsilon$$

解法 2

设

$$S = \{a \text{ 和 } b \text{ 出现的次数相同}\}$$
$$A = \{a \text{ 的次数比 } b \text{ 的次数多 } 1\}$$
$$B = \{b \text{ 的次数比 } a \text{ 的次数多 } 1\}$$

设 $A' = \{a\text{的次数比 }b\text{的次数多 }2\}, x_1x_2\ldots x_n \in A'$,
则 $f(0) = 0 \wedge f(n) = 2$, f 的值从 0 到 n 必须经过 1,
即 $\exists i, f(i) = 1, \therefore x_1x_2\ldots x_i \in A \wedge x_ix_{i+1}\ldots x_n \in A$, 所以 $A' = AA$, 因此语言集合 S, A 和 B 满足下述代数方程组:

$$S = aB \cup bA \cup \{\varepsilon\}$$
$$A = aS \cup bAA$$
$$B = aBB \cup bS$$

故文法 G':

$$S \to aB \mid bA \mid \varepsilon$$
$$A \to aS \mid bAA$$
$$B \to aBB \mid bS$$

解法 3: The unambiguous grammar

设

$$
\begin{aligned}
S &= \{a \text{ 和 } b \text{ 出现的次数相同}\} \\
A &= \{a \text{ 的次数比 } b \text{ 的次数多 } 1, \\
&\quad \text{且 } A \text{ 的任一真前缀 } a \text{ 不比 } b \text{ 多}\} \\
B &= \{b \text{ 的次数比 } a \text{ 的次数多 } 1, \\
&\quad \text{且 } B \text{ 的任一真前缀 } b \text{ 不比 } a \text{ 多}\}
\end{aligned}
$$

因此

$$
\begin{aligned}
S &= aBS \cup bAS \cup \{\varepsilon\} \\
A &= \{a\} \cup bAA \\
B &= aBB \cup \{b\}
\end{aligned}
$$

故文法 G''

$$
\begin{aligned}
S &\rightarrow aBS \mid bAS \mid \varepsilon \\
A &\rightarrow a \mid bAA \\
B &\rightarrow aBB \mid b
\end{aligned}
$$

Example 2

$$L(G) = \{a^n b^n \mid n \in \mathbf{N}\}$$

分析 $L(G)$ 中的串如何递归增长：

1. $\varepsilon \in L$.
2. if $w \in L$, then $awb \in L$.
3. 文法 G:
$$S \to aSb \mid \varepsilon$$
4. 无二义性.

Example 2

$$L(G) = \{a^n b^n \mid n \in \mathbf{N}\}$$

分析 $L(G)$ 中的串如何递归增长：

1. $\varepsilon \in L$.
2. if $w \in L$, then $awb \in L$.
3. 文法 G:

$$S \rightarrow aSb \mid \varepsilon$$

4. 无二义性.

Example 2

$$L(G) = \{a^n b^n \mid n \in \mathbf{N}\}$$

分析 $L(G)$ 中的串如何递归增长：

1. $\varepsilon \in L$.
2. if $w \in L$, then $awb \in L$.
3. 文法 G:

$$S \to aSb \mid \varepsilon$$

4. 无二义性.

Example 2

$$L(G) = \{a^n b^n \mid n \in N\}$$

分析 $L(G)$ 中的串如何递归增长：

1. $\varepsilon \in L$.
2. if $w \in L$, then $awb \in L$.
3. 文法 G:

$$S \rightarrow aSb \mid \varepsilon$$

4. 无二义性.

Example 2

$$L(G) = \{a^n b^n \mid n \in N\}$$

分析 $L(G)$ 中的串如何递归增长：

1. $\varepsilon \in L$.
2. if $w \in L$, then $awb \in L$.
3. 文法 G:

$$S \rightarrow aSb \mid \varepsilon$$

4. 无二义性.

Example 3

$$S = \{a^m b^n \mid m, n \in \mathbf{N} \wedge m \neq n\}$$

对集合 S 分拆，设 $A = \{a^m b^n \mid m > n\}$, $B = \{a^m b^n \mid m < n\}$, 则 $S = A \cup B$

1. 文法 $A \to aAb \mid aA \mid a$ (即 a 在 a 和 b 平衡之后还有多余) 生成集合 A.
2. 文法 $B \to aBb \mid Bb \mid b$ (即 b 在 a 和 b 平衡之后还有多余) 生成集合 B.
3. 文法 G:

$$S \to A \mid B$$
$$A \to aAb \mid aA \mid a$$
$$B \to aBb \mid Bb \mid b$$

4. 二义性?

Example 3

$$S = \{a^m b^n \mid m, n \in \mathbf{N} \wedge m \neq n\}$$

对集合 S 分拆，设 $A = \{a^m b^n \mid m > n\}$, $B = \{a^m b^n \mid m < n\}$, 则 $S = A \cup B$

1. 文法 $A \to aAb \mid aA \mid a$ (即 a 在 a 和 b 平衡之后还有多余) 生成集合 A.
2. 文法 $B \to aBb \mid Bb \mid b$ (即 b 在 a 和 b 平衡之后还有多余) 生成集合 B.
3. 文法 G:

$$\begin{aligned} S &\to A \mid B \\ A &\to aAb \mid aA \mid a \\ B &\to aBb \mid Bb \mid b \end{aligned}$$

4. 二义性?

Example 3

$$S = \{a^m b^n \mid m, n \in \mathbf{N} \wedge m \neq n\}$$

对集合 S 分拆，设 $A = \{a^m b^n \mid m > n\}$, $B = \{a^m b^n \mid m < n\}$,
则 $S = A \cup B$

1. 文法 $A \to aAb \mid aA \mid a$ (即 a 在 a 和 b 平衡之后还有多余) 生成集合 A.
2. 文法 $B \to aBb \mid Bb \mid b$ (即 b 在 a 和 b 平衡之后还有多余) 生成集合 B.
3. 文法 G:

$$\begin{aligned} S &\to A \mid B \\ A &\to aAb \mid aA \mid a \\ B &\to aBb \mid Bb \mid b \end{aligned}$$

4. 二义性?

Example 3

$$S = \{a^m b^n \mid m, n \in \mathbf{N} \wedge m \neq n\}$$

对集合 S 分拆，设 $A = \{a^m b^n \mid m > n\}$, $B = \{a^m b^n \mid m < n\}$,
则 $S = A \cup B$

1. 文法 $A \to aAb \mid aA \mid a$ (即 a 在 a 和 b 平衡之后还有多余) 生成集合 A.
2. 文法 $B \to aBb \mid Bb \mid b$ (即 b 在 a 和 b 平衡之后还有多余) 生成集合 B.
3. 文法 G:

$$\begin{aligned} S &\to A \mid B \\ A &\to aAb \mid aA \mid a \\ B &\to aBb \mid Bb \mid b \end{aligned}$$

4. 二义性?

Example 3

$$S = \{a^m b^n \mid m, n \in \mathbf{N} \wedge m \neq n\}$$

对集合 S 分拆，设 $A = \{a^m b^n \mid m > n\}$, $B = \{a^m b^n \mid m < n\}$,
则 $S = A \cup B$

1. 文法 $A \to aAb \mid aA \mid a$ (即 a 在 a 和 b 平衡之后还有多余) 生成集合 A.
2. 文法 $B \to aBb \mid Bb \mid b$ (即 b 在 a 和 b 平衡之后还有多余) 生成集合 B.
3. 文法 G:

$$\begin{aligned} S &\to A \mid B \\ A &\to aAb \mid aA \mid a \\ B &\to aBb \mid Bb \mid b \end{aligned}$$

4. 二义性?

Example 4

$$S = \{a^m b^n \mid m, n \in \mathbf{N} \land 2m \geq n\}$$

1. 文法 $A \to aAbb \mid \varepsilon$ 生成集合 $\{a^n b^{2n}\}$, 且 $A \subseteq S$.
2. 但是 b 在 S 中可为奇数, 在 S 中最少 b 的出现为语句 ab. 因此加产生式 $A \to ab$,
   即 $L(A) = \{a^m b^n \mid |2m - n| \leq 1\}$.
3. 集合 S 的串在两个 b 平衡一个 a (b 的次数是 a 的两倍) 之后, 还可能有多余的 a.
4. 文法 G:
$$S \to aS \mid A$$
$$A \to aAbb \mid ab \mid \varepsilon$$
5. 删除非终结符 A 后
$$S \to aS \mid aSbb \mid ab \mid \varepsilon$$
6. 二义性?

Example 4

$$S = \{a^m b^n \mid m, n \in \mathbf{N} \wedge 2m \geq n\}$$

1. 文法 $A \to aAbb \mid \varepsilon$ 生成集合 $\{a^n b^{2n}\}$, 且 $A \subseteq S$.
2. 但是 b 在 S 中可为奇数, 在 S 中最少 b 的出现为语句 ab. 因此加产生式 $A \to ab$,
即 $L(A) = \{a^m b^n \mid |2m - n| \leq 1\}$.
3. 集合 S 的串在两个 b 平衡一个 a (b 的次数是 a 的两倍) 之后, 还可能有多余的 a.
4. 文法 G:
$$S \to aS \mid A$$
$$A \to aAbb \mid ab \mid \varepsilon$$
5. 删除非终结符 A 后
$$S \to aS \mid aSbb \mid ab \mid \varepsilon$$
6. 二义性?

Example 4.

$$S = \{a^m b^n \mid m, n \in \mathbf{N} \land 2m \geq n\}$$

1. 文法 $A \to aAbb \mid \varepsilon$ 生成集合 $\{a^n b^{2n}\}$, 且 $A \subseteq S$.
2. 但是 b 在 S 中可为奇数, 在 S 中最少 b 的出现为语句 ab. 因此加产生式 $A \to ab$,
   即 $L(A) = \{a^m b^n \mid |2m - n| \leq 1\}$.
3. 集合 S 的串在两个 b 平衡一个 a (b 的次数是 a 的两倍) 之后, 还可能有多余的 a.
4. 文法 G:
$$S \to aS \mid A$$
$$A \to aAbb \mid ab \mid \varepsilon$$
5. 删除非终结符 A 后
$$S \to aS \mid aSbb \mid ab \mid \varepsilon$$
6. 二义性?

Example 4

$$S = \{a^m b^n \mid m, n \in \mathbf{N} \wedge 2m \geq n\}$$

1. 文法 $A \to aAbb \mid \varepsilon$ 生成集合 $\{a^n b^{2n}\}$, 且 $A \subseteq S$.
2. 但是 b 在 S 中可为奇数, 在 S 中最少 b 的出现为语句 ab. 因此加产生式 $A \to ab$, 即 $L(A) = \{a^m b^n \mid |2m - n| \leq 1\}$.
3. 集合 S 的串在两个 b 平衡一个 a (b 的次数是 a 的两倍) 之后, 还可能有多余的 a.
4. 文法 G:
$$S \to aS \mid A$$
$$A \to aAbb \mid ab \mid \varepsilon$$
5. 删除非终结符 A 后
$$S \to aS \mid aSbb \mid ab \mid \varepsilon$$
6. 二义性?

Example 4.

$$S = \{a^m b^n \mid m, n \in \mathbf{N} \wedge 2m \geq n\}$$

1. 文法 $A \to aAbb \mid \varepsilon$ 生成集合 $\{a^n b^{2n}\}$, 且 $A \subseteq S$.
2. 但是 b 在 S 中可为奇数, 在 S 中最少 b 的出现为语句 ab. 因此加产生式 $A \to ab$, 即 $L(A) = \{a^m b^n \mid |2m - n| \leq 1\}$.
3. 集合 S 的串在两个 b 平衡一个 a (b 的次数是 a 的两倍) 之后, 还可能有多余的 a.
4. 文法 G:

$$S \to aS \mid A$$
$$A \to aAbb \mid ab \mid \varepsilon$$

5. 删除非终结符 A 后

$$S \to aS \mid aSbb \mid ab \mid \varepsilon$$

6. 二义性?

Example 4.

$$S = \{a^m b^n \mid m, n \in \mathbf{N} \wedge 2m \geq n\}$$

1. 文法 $A \to aAbb \mid \varepsilon$ 生成集合 $\{a^n b^{2n}\}$, 且 $A \subseteq S$.
2. 但是 b 在 S 中可为奇数, 在 S 中最少 b 的出现为语句 ab. 因此加产生式 $A \to ab$,
   即 $L(A) = \{a^m b^n \mid |2m - n| \leq 1\}$.
3. 集合 S 的串在两个 b 平衡一个 a (b 的次数是 a 的两倍) 之后, 还可能有多余的 a.
4. 文法 G:
   $$S \to aS \mid A$$
   $$A \to aAbb \mid ab \mid \varepsilon$$
5. 删除非终结符 A 后
   $$S \to aS \mid aSbb \mid ab \mid \varepsilon$$
6. 二义性?

Example 4

$$S = \{a^m b^n \mid m, n \in \mathbf{N} \wedge 2m \geq n\}$$

1. 文法 $A \to aAbb \mid \varepsilon$ 生成集合 $\{a^n b^{2n}\}$, 且 $A \subseteq S$.
2. 但是 b 在 S 中可为奇数, 在 S 中最少 b 的出现为语句 ab. 因此加产生式 $A \to ab$, 即 $L(A) = \{a^m b^n \mid |2m - n| \leq 1\}$.
3. 集合 S 的串在两个 b 平衡一个 a (b 的次数是 a 的两倍) 之后, 还可能有多余的 a.
4. 文法 G:
$$S \to aS \mid A$$
$$A \to aAbb \mid ab \mid \varepsilon$$
5. 删除非终结符 A 后
$$S \to aS \mid aSbb \mid ab \mid \varepsilon$$
6. 二义性?

Example 5

$L(G) = \{$由 $<$ 和 $>$ 组成的嵌套括号对$\}$.

分析 $L(G)$ 的语句如何递归增长：

1. $\varepsilon \in L$;
2. if $w \in L$, then $<w>$, $ww \in L$.
3. 二义性文法 G_1:
$$S \to < S > \mid SS \mid \varepsilon$$
4. 无二义文法 G_2:
$$S \to < S > S \mid \varepsilon$$
5. 上述两文法等价等价.
6. 将 $<$ 看成 start tag, $>$ 看成 end tag, 即是 XML 语言的文法.

Example 5

$L(G) = \{$由 < 和 > 组成的嵌套括号对$\}$.

分析 $L(G)$ 的语句如何递归增长：

1. $\varepsilon \in L$;
2. if $w \in L$, then $< w >$, $ww \in L$.
3. 二义性文法 G_1:
$$S \to < S > \mid SS \mid \varepsilon$$
4. 无二义文法 G_2:
$$S \to < S > S \mid \varepsilon$$
5. 上述两文法等价等价.
6. 将 < 看成 start tag, > 看成 end tag, 即是 XML 语言的文法.

Example 5

$$L(G) = \{\text{由} < \text{和} > \text{组成的嵌套括号对}\}.$$

分析 $L(G)$ 的语句如何递归增长：

1. $\varepsilon \in L$;
2. if $w \in L$, then $< w >$, $ww \in L$.
3. 二义性文法 G_1:
$$S \to < S > \mid SS \mid \varepsilon$$
4. 无二义文法 G_2:
$$S \to < S > S \mid \varepsilon$$
5. 上述两文法等价等价.
6. 将 < 看成 start tag, > 看成 end tag, 即是 XML 语言的文法.

Example 5

$L(G) = \{$由 $<$ 和 $>$ 组成的嵌套括号对$\}$.

分析 $L(G)$ 的语句如何递归增长：

1. $\varepsilon \in L$;
2. if $w \in L$, then $< w >$, $ww \in L$.
3. 二义性文法 G_1:
$$S \to < S > \mid SS \mid \varepsilon$$
4. 无二义文法 G_2:
$$S \to < S > S \mid \varepsilon$$
5. 上述两文法等价等价.
6. 将 $<$ 看成 start tag, $>$ 看成 end tag, 即是 XML 语言的文法.

Example 5

$$L(G) = \{\text{由} < \text{和} > \text{组成的嵌套括号对}\}.$$

分析 $L(G)$ 的语句如何递归增长：

1. $\varepsilon \in L$;
2. if $w \in L$, then $< w >, ww \in L$.
3. 二义性文法 G_1:
$$S \to < S > \mid SS \mid \varepsilon$$
4. 无二义文法 G_2:
$$S \to < S > S \mid \varepsilon$$
5. 上述两文法等价等价.
6. 将 $<$ 看成 start tag, $>$ 看成 end tag, 即是 XML 语言的文法.

Example 5

$L(G) = \{$由 $<$ 和 $>$ 组成的嵌套括号对$\}$.

分析 $L(G)$ 的语句如何递归增长：

1. $\varepsilon \in L$;
2. if $w \in L$, then $<w>$, $ww \in L$.
3. 二义性文法 G_1:
$$S \to <S> \mid SS \mid \varepsilon$$
4. 无二义文法 G_2:
$$S \to <S>S \mid \varepsilon$$
5. 上述两文法等价等价.
6. 将 $<$ 看成 start tag, $>$ 看成 end tag, 即是 XML 语言的文法.

Example 5

$L(G) = \{$由 $<$ 和 $>$ 组成的嵌套括号对$\}$.

分析 $L(G)$ 的语句如何递归增长：

1. $\varepsilon \in L$;
2. if $w \in L$, then $< w >$, $ww \in L$.
3. 二义性文法 G_1：
$$S \rightarrow < S > \mid SS \mid \varepsilon$$
4. 无二义文法 G_2:
$$S \rightarrow < S > S \mid \varepsilon$$
5. 上述两文法等价等价.
6. 将 $<$ 看成 start tag, $>$ 看成 end tag, 即是 XML 语言的文法.

Example 6 --- 描述文法所表达的语言

Example

$$S \to aS \mid Sb \mid a \mid b$$

分析句型:

1. $S \stackrel{*}{\Rightarrow} a^nS \stackrel{*}{\Rightarrow} a^nSb^m \stackrel{*}{\Rightarrow} a^pSb^q \in L$;
2. 由此看出：在句型中 a 只能出现在 S的左边，b 只能在 S 的右边.
3. $a, b \in L(G), \varepsilon \notin L(G)$
4. $L(G) = \{a^mb^n \mid m, n \in N, m+n > 0\}$

Example 6 --- 描述文法所表达的语言

Example

$$S \to aS \mid Sb \mid a \mid b$$

分析句型:

1. $S \overset{*}{\Rightarrow} a^n S \overset{*}{\Rightarrow} a^n S b^m \overset{*}{\Rightarrow} a^p S b^q \in L$;
2. 由此看出：在句型中 a 只能出现在 S的左边，b 只能在 S 的右边.
3. $a, b \in L(G)$, $\varepsilon \notin L(G)$
4. $L(G) = \{a^m b^n | m, n \in N, m + n > 0\}$

Example 6 --- 描述文法所表达的语言

Example

$$S \to aS \mid Sb \mid a \mid b$$

分析句型:

1. $S \overset{*}{\Rightarrow} a^nS \overset{*}{\Rightarrow} a^nSb^m \overset{*}{\Rightarrow} a^pSb^q \in L$;
2. 由此看出: 在句型中 a 只能出现在 S的左边, b 只能在 S 的右边.
3. $a, b \in L(G)$, $\varepsilon \notin L(G)$
4. $L(G) = \{a^mb^n \mid m, n \in N, m + n > 0\}$

Example 6 --- 描述文法所表达的语言

Example

$$S \rightarrow aS \mid Sb \mid a \mid b$$

分析句型:

1. $S \stackrel{*}{\Rightarrow} a^nS \stackrel{*}{\Rightarrow} a^nSb^m \stackrel{*}{\Rightarrow} a^pSb^q \in L$;
2. 由此看出: 在句型中 a 只能出现在 S的左边, b 只能在 S 的右边.
3. $a, b \in L(G), \varepsilon \notin L(G)$
4. $L(G) = \{a^m b^n \mid m, n \in N, m + n > 0\}$

Example 6 --- 描述文法所表达的语言

Example

$$S \to aS \mid Sb \mid a \mid b$$

分析句型:

1. $S \overset{*}{\Rightarrow} a^nS \overset{*}{\Rightarrow} a^nSb^m \overset{*}{\Rightarrow} a^pSb^q \in L$;
2. 由此看出: 在句型中 a 只能出现在 S的左边, b 只能在 S 的右边.
3. $a, b \in L(G)$, $\varepsilon \notin L(G)$
4. $L(G) = \{a^mb^n \mid m, n \in \mathbf{N}, m + n > 0\}$

Example 7 --- 正则表达式的文法设计 (1/2)

$L(G) = \{$由 a 和 b 组成的正则表达式$\}$

分析 $L(G)$中的串如何递归增长：

1. ε, $\emptyset$, a, b $\in L$.
2. if $r, s \in L$, then $r|s$, $r\,s$, r^*, $(r) \in L$
3. 二义性文法 G_1:

$$S \to S|S \mid SS \mid S* \mid (S) \mid a \mid b \mid \varepsilon \mid \emptyset$$

Example 7 --- 正则表达式的文法设计 (1/2)

$L(G) = \{$由 a 和 b 组成的正则表达式$\}$

分析 $L(G)$中的串如何递归增长：

1. ε, $\emptyset$, a, $b \in L$.
2. if $r, s \in L$, then $r|s$, $r\,s$, r^*, $(r) \in L$
3. 二义性文法 G_1:

$$S \to S|S \mid SS \mid S* \mid (S) \mid a \mid b \mid \varepsilon \mid \emptyset$$

Example 7 --- 正则表达式的文法设计 (1/2).

$$L(G) = \{\text{由 } a \text{ 和 } b \text{ 组成的正则表达式}\}$$

分析 $L(G)$中的串如何递归增长：

1. ε, $\emptyset$, a, $b \in L$.
2. if $r, s \in L$, then $r|s$, $r\,s$, r^*, $(r) \in L$
3. 二义性文法 G_1:

$$S \to S|S \mid SS \mid S* \mid (S) \mid a \mid b \mid \varepsilon \mid \emptyset$$

Example 7 --- 正则表达式的文法设计 (1/2).

$L(G) = \{$由 a 和 b 组成的正则表达式$\}$

分析 $L(G)$中的串如何递归增长：

1. ε, $\emptyset$, a, $b \in L$.
2. if $r, s \in L$, then $r|s$, $r\,s$, r^*, $(r) \in L$
3. 二义性文法 G_1:

$$S \to S|S \mid SS \mid S* \mid (S) \mid a \mid b \mid \varepsilon \mid \emptyset$$

Example 7 --- 正则表达式的文法设计 (2/2).

对语法成分进行分类: F 记录字母或括号内容. K 记录 Kleene Closure. C 记录连接运算. S 记录并运算:

1. ε, $\emptyset$, a, b, (S) $\in F$;
2. $\forall f \in L(F)$, $f \in L(K)$; if $k \in L(K)$ then $k^* \in L(K)$;
3. $\forall k \in L(K)$, $k \in L(C)$; if $c \in L(C)$, $k \in L(K)$ then $ck \in L(C)$;
4. $\forall c \in L(C)$, $c \in L(S)$; if $s \in L(S)$, $c \in L(C)$ then $s|c \in L(S)$;
5. 无二义运算左结合的文法 G_2:

$$
\begin{aligned}
S &\to S|C \mid C \\
C &\to CK \mid K \\
K &\to K^* \mid F \\
F &\to a \mid b \mid \varepsilon \mid \emptyset \mid (S)
\end{aligned}
$$

Example 7 --- 正则表达式的文法设计 (2/2)

对语法成分进行分类：F 记录字母或括号内容，K 记录 Kleene Closure，C 记录连接运算，S 记录并运算：

1. ε, $\emptyset$, a, b, $(S) \in F$;
2. $\forall f \in L(F), f \in L(K)$; if $k \in L(K)$ then $k^* \in L(K)$;
3. $\forall k \in L(K), k \in L(C)$; if $c \in L(C), k \in L(K)$ then $ck \in L(C)$;
4. $\forall c \in L(C), c \in L(S)$; if $s \in L(S), c \in L(C)$ then $s|c \in L(S)$;
5. 无二义运算左结合的文法 G_2:

$$\begin{aligned} S &\to S|C \mid C \\ C &\to CK \mid K \\ K &\to K^* \mid F \\ F &\to a \mid b \mid \varepsilon \mid \emptyset \mid (S) \end{aligned}$$

Example 7 --- 正则表达式的文法设计 (2/2)

对语法成分进行分类：F 记录字母或括号内容. K 记录 Kleene Closure. C 记录连接运算. S 记录并运算：

1. ε, $\emptyset$, a, b, (S) $\in F$;
2. $\forall f \in L(F), f \in L(K)$; if $k \in L(K)$ then $k^* \in L(K)$;
3. $\forall k \in L(K), k \in L(C)$; if $c \in L(C), k \in L(K)$ then $ck \in L(C)$;
4. $\forall c \in L(C), c \in L(S)$; if $s \in L(S), c \in L(C)$ then $s|c \in L(S)$;
5. 无二义运算左结合的文法 G_2:

$$
\begin{aligned}
S &\to S|C \mid C \\
C &\to CK \mid K \\
K &\to K^* \mid F \\
F &\to a \mid b \mid \varepsilon \mid \emptyset \mid (S)
\end{aligned}
$$

Example 7 --- 正则表达式的文法设计 (2/2)

对语法成分进行分类：F 记录字母或括号内容. K 记录 Kleene Closure. C 记录连接运算. S 记录并运算：

1. ε, $\emptyset$, a, b, $(S) \in F$;
2. $\forall f \in L(F)$, $f \in L(K)$; if $k \in L(K)$ then $k^* \in L(K)$;
3. $\forall k \in L(K)$, $k \in L(C)$; if $c \in L(C)$, $k \in L(K)$ then $ck \in L(C)$;
4. $\forall c \in L(C)$, $c \in L(S)$; if $s \in L(S)$, $c \in L(C)$ then $s|c \in L(S)$;
5. 无二义运算左结合的文法 G_2:

$$S \to S|C \mid C$$
$$C \to CK \mid K$$
$$K \to K^* \mid F$$
$$F \to a \mid b \mid \varepsilon \mid \emptyset \mid (S)$$

Example 7 --- 正则表达式的文法设计 (2/2)

对语法成分进行分类：F 记录字母或括号内容，K 记录 Kleene Closure，C 记录连接运算，S 记录并运算：

1. ε, $\emptyset$, a, b, $(S) \in F$;
2. $\forall f \in L(F), f \in L(K)$; if $k \in L(K)$ then $k^* \in L(K)$;
3. $\forall k \in L(K), k \in L(C)$; if $c \in L(C), k \in L(K)$ then $ck \in L(C)$;
4. $\forall c \in L(C), c \in L(S)$; if $s \in L(S), c \in L(C)$ then $s|c \in L(S)$;
5. 无二义运算左结合的文法 G_2:

$$S \rightarrow S|C \mid C$$
$$C \rightarrow CK \mid K$$
$$K \rightarrow K^* \mid F$$
$$F \rightarrow a \mid b \mid \varepsilon \mid \emptyset \mid (S)$$

Example 7 --- 正则表达式的文法设计 (2/2)

对语法成分进行分类: F 记录字母或括号内容. K 记录 Kleene Closure. C 记录连接运算. S 记录并运算:

1. ε, $\emptyset$, a, b, (S) $\in F$;
2. $\forall f \in L(F)$, $f \in L(K)$; if $k \in L(K)$ then $k^* \in L(K)$;
3. $\forall k \in L(K)$, $k \in L(C)$; if $c \in L(C)$, $k \in L(K)$ then $ck \in L(C)$;
4. $\forall c \in L(C)$, $c \in L(S)$; if $s \in L(S)$, $c \in L(C)$ then $s|c \in L(S)$;
5. 无二义运算左结合的文法 G_2:

$$
\begin{aligned}
S &\to S|C \mid C \\
C &\to CK \mid K \\
K &\to K^* \mid F \\
F &\to a \mid b \mid \varepsilon \mid \emptyset \mid (S)
\end{aligned}
$$

正则表达式转换为上下文自由文法

对应关系

1. $r^* \Longleftrightarrow S \rightarrow S_r S | \varepsilon$;
2. $r|s \Longleftrightarrow S \rightarrow S_r | S_s$;
3. $rs \Longleftrightarrow S \rightarrow S_r S_s$;

$(a|b)^*abb$

$$S \rightarrow aS|bS|abb \quad \Longleftrightarrow \quad \left\{ \begin{array}{lll} S & \rightarrow & aS|bS|aA \\ A & \rightarrow & bB \\ B & \rightarrow & bC \\ C & \rightarrow & \varepsilon \end{array} \right.$$

正则表达式转换为上下文自由文法

对应关系

1. $r^* \iff S \to S_r S | \varepsilon;$
2. $r|s \iff S \to S_r | S_s;$
3. $rs \iff S \to S_r S_s;$

$(a|b)^*abb$

$$S \to aS|bS|abb \quad \iff \quad \left\{ \begin{array}{lll} S & \to & aS|bS|aA \\ A & \to & bB \\ B & \to & bC \\ C & \to & \varepsilon \end{array} \right.$$

正则表达式转换为上下文自由文法

对应关系

1. $r^* \Longleftrightarrow S \rightarrow S_r S | \varepsilon;$
2. $r|s \Longleftrightarrow S \rightarrow S_r | S_s;$
3. $rs \Longleftrightarrow S \rightarrow S_r S_s;$

$(a|b)^*abb$

$$S \rightarrow aS|bS|abb \quad \Longleftrightarrow \quad \begin{cases} S & \rightarrow & aS|bS|aA \\ A & \rightarrow & bB \\ B & \rightarrow & bC \\ C & \rightarrow & \varepsilon \end{cases}$$

正则表达式转换为上下文自由文法

对应关系

1. $r^* \Longleftrightarrow S \rightarrow S_r S | \varepsilon$;
2. $r|s \Longleftrightarrow S \rightarrow S_r | S_s$;
3. $rs \Longleftrightarrow S \rightarrow S_r S_s$;

$(a|b)^*abb$

$$S \rightarrow aS|bS|abb \quad \Longleftrightarrow \quad \begin{cases} S & \rightarrow & aS|bS|aA \\ A & \rightarrow & bB \\ B & \rightarrow & bC \\ C & \rightarrow & \varepsilon \end{cases}$$

- 形如后者的文法称为正则文法或右线性文法。

正则表达式转换为上下文自由文法

对应关系

1. $r^* \iff S \to S_r S | \varepsilon$;
2. $r|s \iff S \to S_r | S_s$;
3. $rs \iff S \to S_r S_s$;

$(a|b)^*abb$

$$S \to aS|bS|abb \iff \begin{cases} S & \to & aS|bS|aA \\ A & \to & bB \\ B & \to & bC \\ C & \to & \varepsilon \end{cases}$$

- 形如后者的文法称为正则文法或右线性文法。
- 正则语言都能用 CFG 描述，所以 CFG 的表达能力强于前者。

正则表达式转换为上下文自由文法

对应关系

1. $r^* \iff S \to S_r S|\varepsilon$;
2. $r|s \iff S \to S_r|S_s$;
3. $rs \iff S \to S_r S_s$;

$(a|b)^*abb$

$$S \to aS|bS|abb \quad \iff \quad \begin{cases} S & \to & aS|bS|aA \\ A & \to & bB \\ B & \to & bC \\ C & \to & \varepsilon \end{cases}$$

- 形如后者的文法称为正则文法或右线性文法.
- 正则语言都能用 CFG 描述，所以 CFG 的表达能力强于前者.

正则表达式转换为上下文自由文法

对应关系

1. $r^* \Longleftrightarrow S \rightarrow S_r S | \varepsilon;$
2. $r|s \Longleftrightarrow S \rightarrow S_r | S_s;$
3. $rs \Longleftrightarrow S \rightarrow S_r S_s;$

$(a|b)^*abb$

$$S \rightarrow aS|bS|abb \quad \Longleftrightarrow \quad \left\{ \begin{array}{lcl} S & \rightarrow & aS|bS|aA \\ A & \rightarrow & bB \\ B & \rightarrow & bC \\ C & \rightarrow & \varepsilon \end{array} \right.$$

- 形如后者的文法称为正则文法或右线性文法.
- 正则语言都能用 CFG 描述，所以 CFG 的表达能力强于前者.

为什么要两级抽象

Example --- 基于字母的表达式文法

$$
\begin{aligned}
exp &\rightarrow exp + exp \mid exp \times exp \mid id \\
id &\rightarrow letter\ rest \\
letter &\rightarrow a \mid b \mid \ldots \mid z \\
rest &\rightarrow letter\ rest \mid digit\ rest \mid \varepsilon \\
digit &\rightarrow 0 \mid 1 \mid \ldots \mid 9
\end{aligned}
$$

为什么要两级抽象

Example --- 基于字母的表达式文法

$$
\begin{aligned}
exp &\rightarrow exp + exp \mid exp \times exp \mid id \\
id &\rightarrow letter\ rest \\
letter &\rightarrow a \mid b \mid \ldots \mid z \\
rest &\rightarrow letter\ rest \mid digit\ rest \mid \varepsilon \\
digit &\rightarrow 0 \mid 1 \mid \ldots \mid 9
\end{aligned}
$$

- 上下文无关文法的表达能力比正则表达式强，因此形式语言的识别完全可以一级抽象"字母 $\longrightarrow$ 语法"，不需要"字母 $\longrightarrow$ 单词 $\longrightarrow$ 语法"二级抽象。
- 文法的识别远比正则表达式复杂。
- 采用二级抽象可以显著降低语法分析的负担。
- Extended Backus-Naur Form(EBNF)允许产生式右边为正则表达式：

$$exp \longrightarrow term\ (+\ term)^*$$

  这样只用一级抽象就可以同时描述词法和语法，如 XML 规范，语法自动生成工具 ANTLR(http://www.antlr.org/) 等采用 EBNF，但在识别时还是分解成两级结构。

为什么要两级抽象

Example --- 基于字母的表达式文法

$$\begin{aligned} exp &\longrightarrow exp + exp \mid exp \times exp \mid id \\ id &\longrightarrow letter\ rest \\ letter &\longrightarrow a \mid b \mid \ldots \mid z \\ rest &\longrightarrow letter\ rest \mid digit\ rest \mid \varepsilon \\ digit &\longrightarrow 0 \mid 1 \mid \ldots \mid 9 \end{aligned}$$

- 上下文无关文法的表达能力比正则表达式强，因此形式语言的识别完全可以一级抽象“字母 ⟶ 语法”，不需要“字母 ⟶ 单词 ⟶ 语法”二级抽象。
- 文法的识别远比正则表达式复杂。
- 采用二级抽象可以显著降低语法分析的负担。
- Extended Backus-Naur Form(EBNF)允许产生式右边为正则表达式：
$$exp \longrightarrow term\ (+\ term)^*$$
这样只用一级抽象就可以同时描述词法和语法，如 XML 规范，语法自动生成工具 ANTLR(http://www.antlr.org/) 等采用 EBNF，但在识别时还是分解成两级结构。

为什么要两级抽象

Example --- 基于字母的表达式文法

$$\begin{aligned}
exp &\rightarrow exp + exp \mid exp \times exp \mid id \\
id &\rightarrow letter\ rest \\
letter &\rightarrow a \mid b \mid \ldots \mid z \\
rest &\rightarrow letter\ rest \mid digit\ rest \mid \varepsilon \\
digit &\rightarrow 0 \mid 1 \mid \ldots \mid 9
\end{aligned}$$

- 上下文无关文法的表达能力比正则表达式强，因此形式语言的识别完全可以一级抽象“字母 $\longrightarrow$ 语法”，不需要“字母 $\longrightarrow$ 单词 $\longrightarrow$ 语法”二级抽象。
- 文法的识别远比正则表达式复杂。
- 采用二级抽象可以显著降低语法分析的负担。
- Extended Backus-Naur Form(EBNF)允许产生式右边为正则表达式：
  $$exp \rightarrow term\ (+\ term)^*$$
  这样只用一级抽象就可以同时描述词法和语法，如 XML 规范，语法自动生成工具 ANTLR(http://www.antlr.org/) 等采用 EBNF，但在识别时还是分解成两级结构。

为什么要两级抽象

Example --- 基于字母的表达式文法

$$
\begin{aligned}
exp &\rightarrow exp + exp \mid exp \times exp \mid id \\
id &\rightarrow letter\ rest \\
letter &\rightarrow a \mid b \mid \ldots \mid z \\
rest &\rightarrow letter\ rest \mid digit\ rest \mid \varepsilon \\
digit &\rightarrow 0 \mid 1 \mid \ldots \mid 9
\end{aligned}
$$

- 上下文无关文法的表达能力比正则表达式强，因此形式语言的识别完全可以一级抽象“字母 $\longrightarrow$ 语法”，不需要“字母 $\longrightarrow$ 单词 $\longrightarrow$ 语法”二级抽象。
- 文法的识别远比正则表达式复杂。
- 采用二级抽象可以显著降低语法分析的负担。
- Extended Backus-Naur Form(EBNF)允许产生式右边为正则表达式：

$$exp \rightarrow term\ (+\ term)^*$$

  这样只用一级抽象就可以同时描述词法和语法，如 XML 规范，语法自动生成工具 ANTLR(http://www.antlr.org/) 等采用 EBNF，但在识别时还是分解成两级结构。

为什么要两级抽象

Example --- 基于字母的表达式文法

$$
\begin{aligned}
exp &\rightarrow exp + exp \mid exp \times exp \mid id \\
id &\rightarrow letter\ rest \\
letter &\rightarrow a \mid b \mid \ldots \mid z \\
rest &\rightarrow letter\ rest \mid digit\ rest \mid \varepsilon \\
digit &\rightarrow 0 \mid 1 \mid \ldots \mid 9
\end{aligned}
$$

- 上下文无关文法的表达能力比正则表达式强，因此形式语言的识别完全可以一级抽象"字母 $\longrightarrow$ 语法"，不需要"字母 $\longrightarrow$ 单词 $\longrightarrow$ 语法"二级抽象。
- 文法的识别远比正则表达式复杂。
- 采用二级抽象可以显著降低语法分析的负担。
- Extended Backus-Naur Form(EBNF)允许产生式右边为正则表达式：

$$exp \rightarrow term\ (+\ term)^*$$

这样只用一级抽象就可以同时描述词法和语法，如 XML 规范，语法自动生成工具 ANTLR(http://www.antlr.org/) 等采用 EBNF，但在识别时还是分解成两级结构。

自动机转换为上下文自由文法

对应关系

1. 每个状态对应一个非终结符，开始状态对应文法开始符号.
2. if $(i) \xrightarrow{a} (j)$, then: $S_i \to aS_j$.
3. 如果(i)是接受状态，则增加产生式：$S_i \to \varepsilon$;

Example

自动机转换为上下文自由文法

对应关系

1. 每个状态对应一个非终结符，开始状态对应文法开始符号.
2. if $i \xrightarrow{a} j$, then: $S_i \rightarrow aS_j$.
3. 如果 i 是接受状态，则增加产生式：$S_i \rightarrow \varepsilon$;

Example

自动机转换为上下文自由文法

对应关系

1. 每个状态对应一个非终结符，开始状态对应文法开始符号.
2. if $i \xrightarrow{a} j$, then: $S_i \to aS_j$.
3. 如果i是接受状态，则增加产生式：$S_i \to \varepsilon$;

Example

自动机转换为上下文自由文法

对应关系

1. 每个状态对应一个非终结符，开始状态对应文法开始符号.
2. if $i \xrightarrow{a} j$, then: $S_i \to aS_j$.
3. 如果 i 是接受状态，则增加产生式：$S_i \to \varepsilon$;

Example

自动机转换为上下文自由文法

对应关系

1. 每个状态对应一个非终结符，开始状态对应文法开始符号.
2. if $(i) \xrightarrow{a} (j)$, then: $S_i \rightarrow aS_j$.
3. 如果(i)是接受状态，则增加产生式：$S_i \rightarrow \varepsilon$;

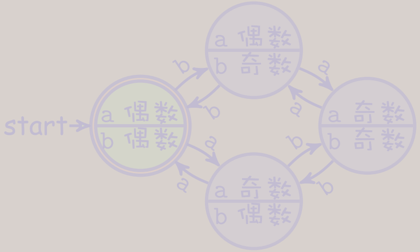
Example

自动机转换为上下文自由文法

对应关系

1. 每个状态对应一个非终结符，开始状态对应文法开始符号。
2. if $(i) \xrightarrow{a} (j)$, then: $S_i \rightarrow aS_j$.
3. 如果(i)是接受状态，则增加产生式：$S_i \rightarrow \varepsilon$;

Example



$\Longleftrightarrow$

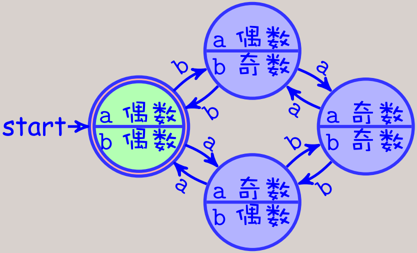
$$\left\{\begin{array}{lcl} A & \rightarrow & aD|bB|\varepsilon \\ B & \rightarrow & aC|bA \\ C & \rightarrow & aB|bD \\ D & \rightarrow & aA|bC \end{array}\right.$$

自动机转换为上下文自由文法

对应关系

1. 每个状态对应一个非终结符，开始状态对应文法开始符号.
2. if $i \xrightarrow{a} j$, then: $S_i \to aS_j$.
3. 如果 i 是接受状态，则增加产生式：$S_i \to \varepsilon$;

Example



$\Longleftrightarrow$

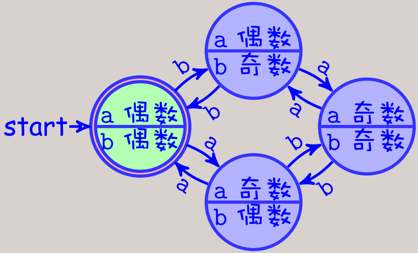
$$\begin{cases} A \to aD \mid bB \mid \varepsilon \\ B \to aC \mid bA \\ C \to aB \mid bD \\ D \to aA \mid bC \end{cases}$$

自动机转换为上下文自由文法

对应关系

1. 每个状态对应一个非终结符，开始状态对应文法开始符号.
2. if $i \xrightarrow{a} j$, then: $S_i \to aS_j$.
3. 如果 i 是接受状态，则增加产生式：$S_i \to \varepsilon$;

Example



$\Longleftrightarrow$

$$\begin{cases} A \to aD \mid bB \mid \varepsilon \\ B \to aC \mid bA \\ C \to aB \mid bD \\ D \to aA \mid bC \end{cases}$$

Syntactic sugar

Syntactic sugar is extra syntax in a programming language that makes the code easier to understand or write.

Example

Syntactic sugar

Syntactic sugar is extra syntax in a programming language that makes the code easier to understand or write.

Example

- C: $E \rightarrow E[E]$ is an extra pruduction for simply $*(E+E)$. so `2["abc"]` is correct C expression.
- C#:

```
public class Student {
  public string Name { get; set; }
}
```

- Syntactic sugar is popular in functional programming languages. e.g. Haskell is much sweeter.

sweet	unsweet
``a `mappend` b``	`mappend a b`
`(+ 2)`	`\x -> x + 2`
`(x, y)`	`(,) x y`
`[1, 2, 3]`	`(:) 1 ((:) 2 ((:) 3 []))`

- ``sweet or salty'' is always a debate in programming language designs.

Syntactic sugar

Syntactic sugar is extra syntax in a programming language that makes the code easier to understand or write.

Example

- C: $E \to E[E]$ is an extra pruduction for simply $*(E + E)$. so `2["abc"]` is correct C expression.
- $C\#$:

```
public class Student {
  public string Name { get; set; }
}
```

- Syntactic sugar is popular in functional programming languages. e.g. Haskell is much sweeter.

sweet	unsweet
``a `mappend` b``	`mappend a b`
`(+ 2)`	`\x -> x + 2`
`(x, y)`	`(,) x y`
`[1, 2, 3]`	`(:) 1 ((:) 2 ((:) 3 []))`

- \`\`sweet or salty'' is always a debate in programming language designs.

Syntactic sugar

Syntactic sugar is extra syntax in a programming language that makes the code easier to understand or write.

Example

- C: $E \to E[E]$ is an extra pruduction for simply $*(E + E)$. so `2["abc"]` is correct C expression.
- $C\#$:

```
public class Student {
  public string Name { get; set; }
}
```

- Syntactic sugar is popular in functional programming languages. e.g. Haskell is much sweeter.

sweet	unsweet
``a `mappend` b``	`mappend a b`
`(+ 2)`	`\x -> x + 2`
`(x, y)`	`(,) x y`
`[1, 2, 3]`	`(:) 1 ((:) 2 ((:) 3 []))`

- ``sweet or salty'' is always a debate in programming language designs.

Syntactic sugar

Syntactic sugar is extra syntax in a programming language that makes the code easier to understand or write.

Example

- C: $E \to E[E]$ is an extra pruduction for simply $*(E+E)$. so `2["abc"]` is correct C expression.
- C#:

```
public class Student {
  public string Name { get; set; }
}
```

- Syntactic sugar is popular in functional programming languages. e.g. Haskell is much sweeter.

sweet	unsweet
`` a `mappend` b ``	`mappend a b`
`(+ 2)`	`\x -> x + 2`
`(x, y)`	`(,) x y`
`[1, 2, 3]`	`(:) 1 ((:) 2 ((:) 3 []))`

- ``sweet or salty'' is always a debate in programming language designs.

Syntactic sugar

Syntactic sugar is extra syntax in a programming language that makes the code easier to understand or write.

Example

- C: $E \to E[E]$ is an extra pruduction for simply $*(E + E)$. so `2["abc"]` is correct C expression.
- C#:

```
public class Student {
  public string Name { get; set; }
}
```

- Syntactic sugar is popular in functional programming languages. e.g. Haskell is much sweeter.

sweet	unsweet
``a `mappend` b``	`mappend a b`
`(+ 2)`	`\x -> x + 2`
`(x, y)`	`(,) x y`
`[1, 2, 3]`	`(:) 1 ((:) 2 ((:) 3 []))`

- ``sweet or salty'' is always a debate in programming language designs.

CFG 的表达能力

Pumping lemma

Let L be a context-free language. There is $p \in \mathbb{N}^{+}$ such that for any sentence z with the length $|z| \geq p$, there are $u, v, w, x, y \in \Sigma^{*}$ with the following properties:

- $z = uvwxy$.
- $vx \neq \varepsilon$.
- $|vwx| < p$.
- $uv^{n}wx^{n}y \in L$ forall $n \in \mathbb{N}$.

Example

CFG 的表达能力

Pumping lemma

Let L be a context-free language. There is $p \in \mathbb{N}^+$ such that for any sentence z with the length $|z| \geq p$, there are $u, v, w, x, y \in \Sigma^*$ with the following properties:

- $z = uvwxy$.
- $vx \neq \varepsilon$.
- $|vwx| < p$.
- $uv^nwx^ny \in L$ forall $n \in \mathbb{N}$.

Example

CFG 的表达能力

Pumping lemma

Let L be a context-free language. There is $p \in \mathbb{N}^+$ such that for any sentence z with the length $|z| \geq p$, there are $u, v, w, x, y \in \Sigma^*$ with the following properties:

- $z = uvwxy$.
- $vx \neq \varepsilon$.
- $|vwx| < p$.
- $uv^n wx^n y \in L$ forall $n \in \mathbb{N}$.

Example

CFG 的表达能力

Pumping lemma

Let L be a context-free language. There is $p \in \mathbb{N}^+$ such that for any sentence z with the length $|z| \geq p$, there are $u, v, w, x, y \in \Sigma^*$ with the following properties:

- $z = uvwxy$.
- $vx \neq \varepsilon$.
- $|vwx| < p$.
- $uv^nwx^ny \in L$ forall $n \in \mathbb{N}$.

Example

- $\{a^nb^nc^n \mid n \in \mathbb{N}\}$ 不是 CFL，但 $\{a^mb^nc^p \mid m, n, p \in \mathbb{N}\}$ 是 CFL.
- $\{ww \mid w \in \{a, b\}^*\}$ 不是 CFL，但 $\{ww^R \mid w \in \{a, b\}^*\}$ 是 CFL.

CFG 的表达能力

Pumping lemma

Let L be a context-free language. There is $p \in \mathbf{N}^+$ such that for any sentence z with the length $|z| \geq p$, there are $u, v, w, x, y \in \Sigma^*$ with the following properties:

- $z = uvwxy$.
- $vx \neq \varepsilon$.
- $|vwx| < p$.
- $uv^n wx^n y \in L$ forall $n \in \mathbf{N}$.

Example

- $\{a^n b^n c^n \mid n \in \mathbf{N}\}$ 不是 CFL，但 $\{a^m b^n c^p \mid m, n, p \in \mathbf{N}\}$ 是 CFL.
- $\{ww \mid w \in \{a, b\}^*\}$ 不是 CFL，但 $\{ww^R \mid w \in \{a, b\}^*\}$ 是 CFL.

CFG 的表达能力

Pumping lemma

Let L be a context-free language. There is $p \in \mathbf{N}^+$ such that for any sentence z with the length $|z| \geq p$, there are $u, v, w, x, y \in \Sigma^*$ with the following properties:

- $z = uvwxy$.
- $vx \neq \varepsilon$.
- $|vwx| < p$.
- $uv^n wx^n y \in L$ forall $n \in \mathbf{N}$.

Example

- $\{a^n b^n c^n \mid n \in \mathbf{N}\}$ 不是 CFL，但 $\{a^m b^n c^p \mid m, n, p \in \mathbf{N}\}$ 是 CFL.
- $\{ww \mid w \in \{a, b\}^*\}$ 不是 CFL, 但 $\{ww^R \mid w \in \{a, b\}^*\}$ 是 CFL.

CFG 的表达能力

Pumping lemma

Let L be a context-free language. There is $p \in \mathbb{N}^+$ such that for any sentence z with the length $|z| \geq p$, there are $u, v, w, x, y \in \Sigma^*$ with the following properties:

- $z = uvwxy$.
- $vx \neq \varepsilon$.
- $|vwx| < p$.
- $uv^nwx^ny \in L$ forall $n \in \mathbb{N}$.

Example

- $\{a^nb^nc^n \mid n \in \mathbb{N}\}$ 不是 CFL，但 $\{a^mb^nc^p \mid m, n, p \in \mathbb{N}\}$ 是 CFL.
- $\{ww \mid w \in \{a, b\}^*\}$ 不是 CFL，但 $\{ww^R \mid w \in \{a, b\}^*\}$ 是 CFL.

CFG 的表达能力

Pumping lemma

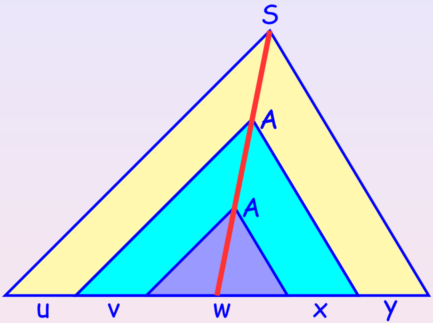
Let L be a context-free language. There is $p \in \mathbb{N}^+$ such that for any sentence z with the length $|z| \geq p$, there are $u, v, w, x, y \in \Sigma^*$ with the following properties:

- $z = uvwxy$.
- $vx \neq \varepsilon$.
- $|vwx| < p$.
- $uv^n wx^n y \in L$ forall $n \in \mathbb{N}$.

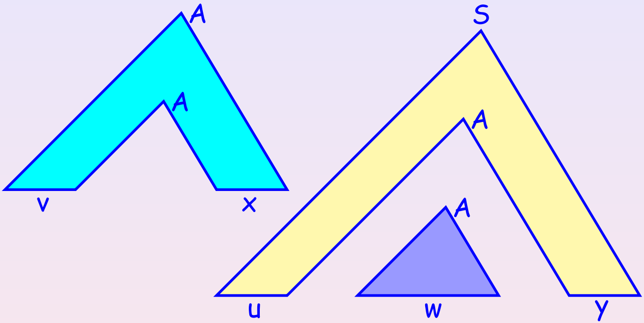
Example

- $\{a^n b^n c^n \mid n \in \mathbb{N}\}$ 不是 CFL，但 $\{a^m b^n c^p \mid m, n, p \in \mathbb{N}\}$ 是 CFL.
- $\{ww \mid w \in \{a, b\}^*\}$ 不是 CFL, 但 $\{ww^R \mid w \in \{a, b\}^*\}$ 是 CFL.

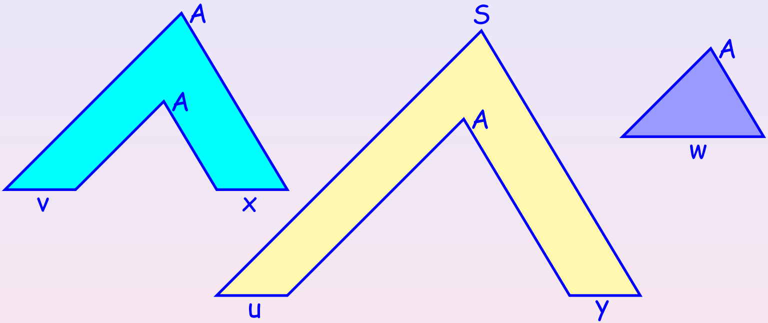
Illustration



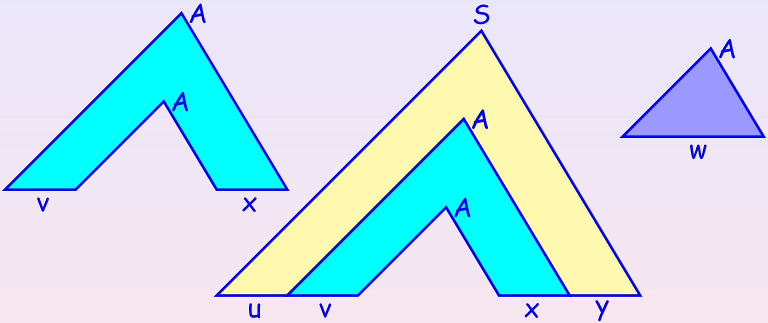
Illustration



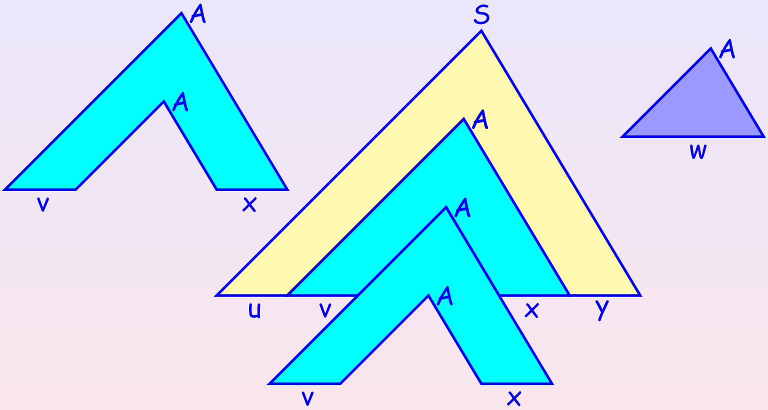
Illustration



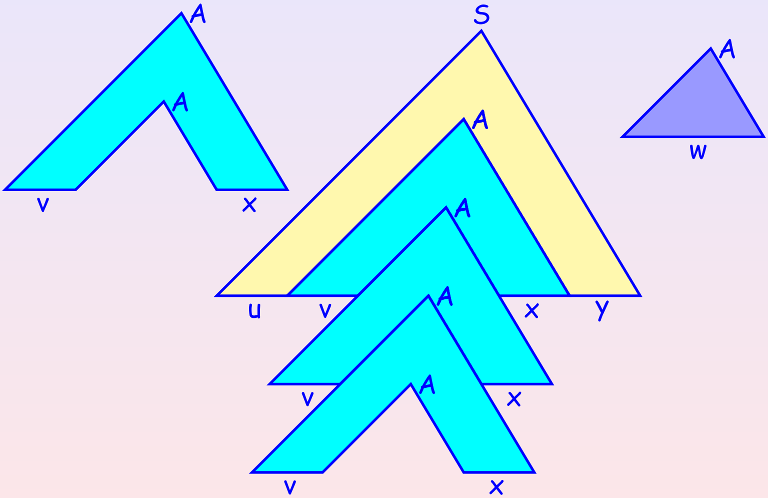
Illustration



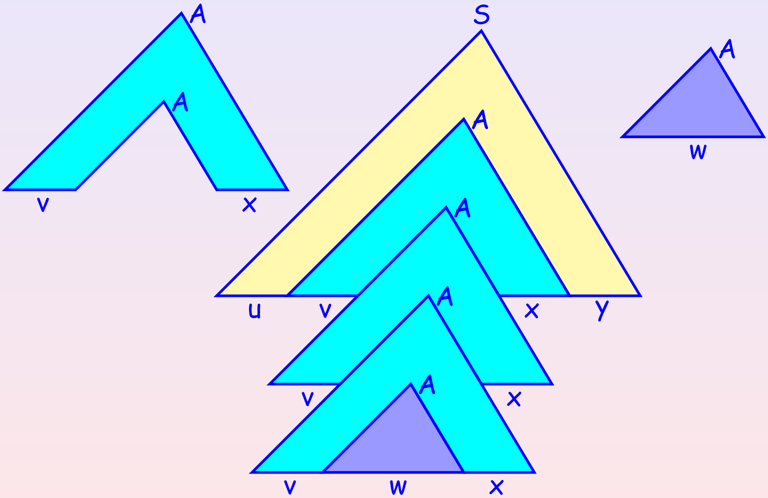
Illustration



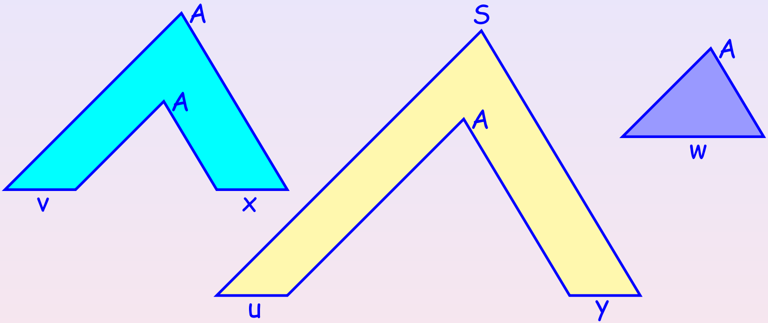
Illustration



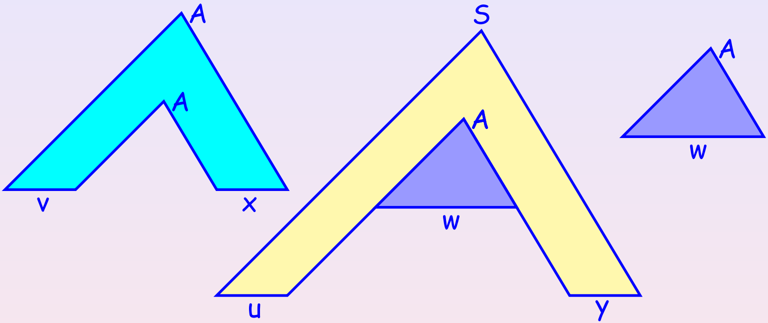
Illustration



Illustration



Illustration



else 的挂靠

最早出现在 Algol60 的设计中.

if-then-else 语句

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

dangling-else

当 else 与 if 不平衡出现时, else 如何挂靠 if 产生二义性.

else 的挂靠

最早出现在 Algol60 的设计中.

if-then-else 语句

$$
\begin{array}{rcl}
\textit{stmt} & \rightarrow & \texttt{if}\ \textit{expr}\ \texttt{then}\ \textit{stmt} \\
 & | & \texttt{if}\ \textit{expr}\ \texttt{then}\ \textit{stmt}\ \texttt{else}\ \textit{stmt} \\
 & | & \textit{other}
\end{array}
$$

dangling-else

当 else 与 if 不平衡出现时，else 如何挂靠 if 产生二义性.

- 平衡出现：

$$\texttt{if}\ E_1\ \texttt{then}\ S1\ \texttt{else}\ \texttt{if}\ E_2\ \texttt{then}\ S_2\ \texttt{else}\ S_3$$

- 不平衡出现：

$$\texttt{if}\ E_1\ \texttt{then}\ \texttt{if}\ E_2\ \texttt{then}\ S_2\ \texttt{else}\ S_3$$

else 可以挂靠第一个 if，也可以挂靠第二个 if.

else 的挂靠

最早出现在 Algol60 的设计中.

if-then-else 语句

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

dangling-else

当 else 与 if 不平衡出现时, else 如何挂靠 if 产生二义性.

- 平衡出现:

$$\texttt{if}\ E_1\ \texttt{then}\ S1\ \texttt{else}\ \texttt{if}\ E_2\ \texttt{then}\ S_2\ \texttt{else}\ S_3$$

- 不平衡出现:

$$\texttt{if}\ E_1\ \texttt{then}\ \texttt{if}\ E_2\ \texttt{then}\ S_1\ \texttt{else}\ S_2$$

else 可以挂靠第一个 if, 也可以是第二个 if.

else 的挂靠

最早出现在 Algol60 的设计中.

if-then-else 语句

$$
\begin{aligned}
stmt \;\rightarrow\; & \textbf{if}\ expr\ \textbf{then}\ stmt \\
\mid\; & \textbf{if}\ expr\ \textbf{then}\ stmt\ \textbf{else}\ stmt \\
\mid\; & other
\end{aligned}
$$

dangling-else

当 else 与 if 不平衡出现时, else 如何挂靠 if 产生二义性.

- 平衡出现:

$$\texttt{if}\ E_1\ \texttt{then}\ S1\ \texttt{else}\ \texttt{if}\ E_2\ \texttt{then}\ S_2\ \texttt{else}\ S_3$$

- 不平衡出现:

$$\texttt{if}\ E_1\ \texttt{then}\ \texttt{if}\ E_2\ \texttt{then}\ S_1\ \texttt{else}\ S_2$$

else 可以挂靠第一个 if, 也可以是第二个 if.

else 的挂靠

最早出现在 Algol60 的设计中.

if-then-else 语句

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

dangling-else

当 else 与 if 不平衡出现时，else 如何挂靠 if 产生二义性.

- 平衡出现：

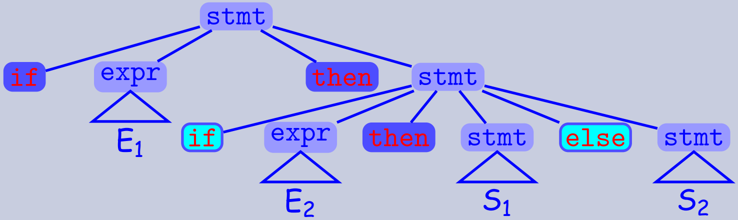
$$\texttt{if}\ E_1\ \texttt{then}\ S1\ \texttt{else}\ \texttt{if}\ E_2\ \texttt{then}\ S_2\ \texttt{else}\ S_3$$

- 不平衡出现：

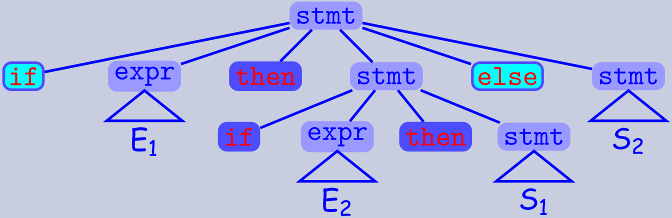
$$\texttt{if}\ E_1\ \texttt{then}\ \texttt{if}\ E_2\ \texttt{then}\ S_1\ \texttt{else}\ S_2$$

else 可以挂靠第一个 if，也可以是第二个 if.

两颗不同的分析树



(a) else 挂靠最近的 if



(b) else 挂靠较远的 if

解消二义性

让 else 挂靠最近的 if.

消除二义性后的文法

$$
\begin{aligned}
stmt &\rightarrow m\_stmt \mid u\_stmt \\
m\_stmt &\rightarrow \textbf{if}\ expr\ \textbf{then}\ m\_stmt\ \textbf{else}\ m\_stmt \\
&\mid\ other \\
u\_stmt &\rightarrow \textbf{if}\ expr\ \textbf{then}\ stmt \\
&\mid\ \textbf{if}\ expr\ \textbf{then}\ m\_stmt\ \textbf{else}\ u\_stmt
\end{aligned}
$$

Example

$$
\underbrace{\underbrace{\textbf{if}\ E_1\ \textbf{then}\ \underbrace{\underbrace{\textbf{if}\ E_2\ \textbf{then}\ S_1\ \textbf{else}\ S_2}_{m\_stmt}}_{stmt}}_{u\_stmt}}_{stmt}
$$

解消二义性

让 else 挂靠最近的 if.

消除二义性后的文法

$$
\begin{array}{lcl}
\mathit{stmt} & \rightarrow & \mathit{m\_stmt} \mid \mathit{u\_stmt} \\
\mathit{m\_stmt} & \rightarrow & \texttt{if}\ \mathit{expr}\ \texttt{then}\ \mathit{m\_stmt}\ \texttt{else}\ \mathit{m\_stmt} \\
 & \mid & \mathit{other} \\
\mathit{u\_stmt} & \rightarrow & \texttt{if}\ \mathit{expr}\ \texttt{then}\ \mathit{stmt} \\
 & \mid & \texttt{if}\ \mathit{expr}\ \texttt{then}\ \mathit{m\_stmt}\ \texttt{else}\ \mathit{u\_stmt}
\end{array}
$$

Example

$$
\underbrace{\underbrace{\texttt{if}\ E_1\ \texttt{then}\ \underbrace{\underbrace{\texttt{if}\ E_2\ \texttt{then}\ S_1\ \texttt{else}\ S_2}_{\mathit{m\_stmt}}}_{\mathit{stmt}}}_{\mathit{u\_stmt}}}_{\mathit{stmt}}
$$

解消二义性

让 else 挂靠最近的 if.

消除二义性后的文法

$$\begin{array}{lcl}
\text{stmt} & \to & \text{m_stmt} \mid \text{u_stmt} \\
\text{m_stmt} & \to & \texttt{if}\ \text{expr}\ \texttt{then}\ \text{m_stmt}\ \texttt{else}\ \text{m_stmt} \\
 & \mid & \text{other} \\
\text{u_stmt} & \to & \texttt{if}\ \text{expr}\ \texttt{then}\ \text{stmt} \\
 & \mid & \texttt{if}\ \text{expr}\ \texttt{then}\ \text{m_stmt}\ \texttt{else}\ \text{u_stmt}
\end{array}$$

Example

$$\underbrace{\underbrace{\texttt{if}\ E_1\ \texttt{then}\ \underbrace{\underbrace{\texttt{if}\ E_2\ \texttt{then}\ S_1\ \texttt{else}\ S_2}_{\text{m\_stmt}}}_{\text{stmt}}}_{\text{u\_stmt}}}_{\text{stmt}}$$

解消二义性

不能解消二义性的文法

$$
\begin{aligned}
\mathit{stmt} &\rightarrow \textbf{if}\ \mathit{expr}\ \textbf{then}\ \mathit{stmt} \\
&\mid\ \mathit{m\_stmt} \\
\mathit{m\_stmt} &\rightarrow \textbf{if}\ \mathit{expr}\ \textbf{then}\ \mathit{m\_stmt}\ \textbf{else}\ \mathit{stmt} \\
&\mid\ \mathit{other}
\end{aligned}
$$

m_stmt 的 else 部分可能会导致不平衡

$$
\underbrace{\textbf{if}\ E_1\ \textbf{then}\ \underbrace{\textbf{if}\ E_2\ \textbf{then}\ S_1\ \textbf{else}\ \underbrace{\textbf{if}\ E_3\ \textbf{then}\ S_2}_{\mathit{stmt}}}_{\mathit{m\_stmt}}\ \textbf{else}\ S_3}_{\mathit{m\_stmt}}
$$

$$
\underbrace{\textbf{if}\ E_1\ \textbf{then}\ \underbrace{\textbf{if}\ E_2\ \textbf{then}\ S_1\ \textbf{else}\ \underbrace{\textbf{if}\ E_3\ \textbf{then}\ S_2\ \textbf{else}\ S_3}_{\mathit{m\_stmt}}}_{\mathit{m\_stmt}}}_{\mathit{stmt}}
$$

解消二义性

不能解消二义性的文法

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

m_stmt 的 else 部分可能会导致不平衡

$$\underbrace{\texttt{if}\ E_1\ \texttt{then}\ \underbrace{\texttt{if}\ E_2\ \texttt{then}\ S_1\ \texttt{else}\ \underbrace{\texttt{if}\ E_3\ \texttt{then}\ S_2}_{\mathit{stmt}}}_{\mathit{m\_stmt}}\ \texttt{else}\ S_3}_{\mathit{m\_stmt}}$$

$$\underbrace{\texttt{if}\ E_1\ \texttt{then}\ \underbrace{\texttt{if}\ E_2\ \texttt{then}\ S_1\ \texttt{else}\ \underbrace{\texttt{if}\ E_3\ \texttt{then}\ S_2\ \texttt{else}\ S_3}_{\mathit{m\_stmt}}}_{\mathit{m\_stmt}}}_{\mathit{stmt}}$$

CFG 和自动机的关系

- 程序设计语言一般是 CFL.
- DFA 不能识别 CFL.
- CFG 与不确定的下推自动机等价.
- 不确定的下推自动机与确定的下推自动机不等价.
- 无回溯的语法分析器仅能识别部分的 CFL.

CFG 和自动机的关系.

- 程序设计语言一般是 CFL.
- DFA 不能识别 CFL.
- CFG 与不确定的下推自动机等价.
- 不确定的下推自动机与确定的下推自动机不等价.
- 无回溯的语法分析器仅能识别部分的 CFL.

CFG 和自动机的关系

- 程序设计语言一般是 CFL.
- DFA 不能识别 CFL.
- CFG 与不确定的下推自动机等价.
- 不确定的下推自动机与确定的下推自动机不等价.
- 无回溯的语法分析器仅能识别部分的 CFL.

CFG 和自动机的关系

- 程序设计语言一般是 CFL.
- DFA 不能识别 CFL.
- CFG 与不确定的下推自动机等价.
- 不确定的下推自动机与确定的下推自动机不等价.
- 无回溯的语法分析器仅能识别部分的 CFL.

CFG 和自动机的关系

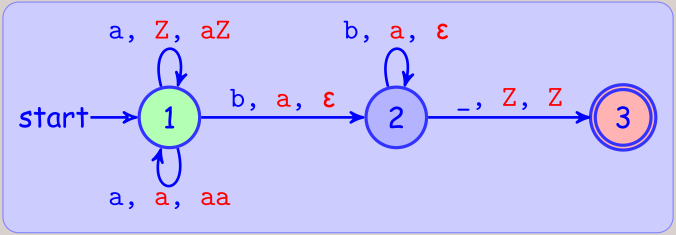
- 程序设计语言一般是 CFL.
- DFA 不能识别 CFL.
- CFG 与不确定的下推自动机等价.
- 不确定的下推自动机与确定的下推自动机不等价.
- 无回溯的语法分析器仅能识别部分的 CFL.

CFG 和自动机的关系

- 程序设计语言一般是 CFL.
- DFA 不能识别 CFL.
- CFG 与不确定的下推自动机等价.
- 不确定的下推自动机与确定的下推自动机不等价.
- 无回溯的语法分析器仅能识别部分的 CFL.

Example

识别语言 $\{a^nb^n \mid n \in N\}$ 的下推自动机



本章小节

1. 上下文无关文法
   - 递归定义与产生式
   - 上下文无关文法
   - 上下文无关语言
2. 语法树
   - 语法树
   - 语法树与推导的关系
   - 二义性
3. 上下文无关文法的设计
   - 设计举例
   - 正则表达式转换为上下文自由文法
   - 自动机转换为上下文自由文法
   - CFG 的表达能力
4. 二义性的再讨论