

语法分析

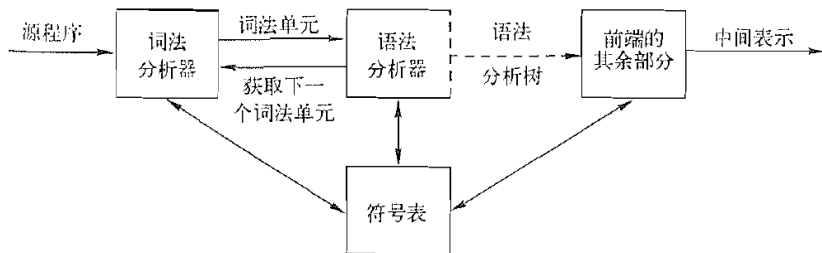
魏恒峰

hfwei@nju.edu.cn

2020 年 11 月 23 日



输入：词法单元流 & 语言的语法规则



输出：语法分析树 (Parse Tree)

语法分析举例

$\langle \text{Stmt} \rangle \rightarrow \langle \text{Id} \rangle = \langle \text{Expr} \rangle ;$
$\langle \text{Stmt} \rangle \rightarrow \{ \langle \text{StmtList} \rangle \}$
$\langle \text{Stmt} \rangle \rightarrow \text{if} (\langle \text{Expr} \rangle) \langle \text{Stmt} \rangle$
$\langle \text{StmtList} \rangle \rightarrow \langle \text{Stmt} \rangle$
$\langle \text{StmtList} \rangle \rightarrow \langle \text{StmtList} \rangle \langle \text{Stmt} \rangle$
$\langle \text{Expr} \rangle \rightarrow \langle \text{Id} \rangle$
$\langle \text{Expr} \rangle \rightarrow \langle \text{Num} \rangle$
$\langle \text{Expr} \rangle \rightarrow \langle \text{Expr} \rangle \langle \text{Optr} \rangle \langle \text{Expr} \rangle$
$\langle \text{Id} \rangle \rightarrow x$
$\langle \text{Id} \rangle \rightarrow y$
$\langle \text{Num} \rangle \rightarrow 0$
$\langle \text{Num} \rangle \rightarrow 1$
$\langle \text{Num} \rangle \rightarrow 9$
$\langle \text{Optr} \rangle \rightarrow >$
$\langle \text{Optr} \rangle \rightarrow +$

		$\langle \text{Stmt} \rangle$
if ($\langle \text{Expr} \rangle$	$\langle \text{Stmt} \rangle$
if ($\langle \text{Expr} \rangle \langle \text{Optr} \rangle \langle \text{Expr} \rangle$	$\langle \text{Stmt} \rangle$
if ($\langle \text{Id} \rangle \langle \text{Optr} \rangle \langle \text{Expr} \rangle$	$\langle \text{Stmt} \rangle$
if ($x \langle \text{Optr} \rangle \langle \text{Expr} \rangle$	$\langle \text{Stmt} \rangle$
if ($x > \langle \text{Expr} \rangle$	$\langle \text{Stmt} \rangle$
if ($x > \langle \text{Num} \rangle$	$\langle \text{Stmt} \rangle$
if ($x > 9$	$\langle \text{Stmt} \rangle$
if ($x > 9$	{ $\langle \text{StmtList} \rangle$ }
if ($x > 9$	{ $\langle \text{StmtList} \rangle \langle \text{Stmt} \rangle$ }
if ($x > 9$	{ $\langle \text{Stmt} \rangle$ }
if ($x > 9$	{ $\langle \text{Id} \rangle = \langle \text{Expr} \rangle ;$ }
if ($x > 9$	{ $x = \langle \text{Expr} \rangle ;$ }
if ($x > 9$	{ $x = \langle \text{Num} \rangle ;$ }
if ($x > 9$	{ $x = 0 ;$ }
if ($x > 9$	{ $x = 0 ; \langle \text{Id} \rangle = \langle \text{Expr} \rangle ;$ }
if ($x > 9$	{ $x = 0 ; y = \langle \text{Expr} \rangle$ }
if ($x > 9$	{ $x = 0 ; y = \langle \text{Expr} \rangle \langle \text{Optr} \rangle \langle \text{Expr} \rangle$ }
if ($x > 9$	{ $x = 0 ; y = \langle \text{Id} \rangle \langle \text{Optr} \rangle \langle \text{Expr} \rangle$ }
if ($x > 9$	{ $x = 0 ; y = y \langle \text{Optr} \rangle \langle \text{Expr} \rangle$ }
if ($x > 9$	{ $x = 0 ; y = y + \langle \text{Expr} \rangle$ }
if ($x > 9$	{ $x = 0 ; y = y + \langle \text{Num} \rangle$ }
if ($x > 9$	{ $x = 0 ; y = y + 1 ;$ }

语法分析阶段的主题之一: 上下文无关文法

```

<Stmt> → <Id> = <Expr> ;
<Stmt> → { <StmtList> }
<Stmt> → if ( <Expr> ) <Stmt>
<StmtList> → <Stmt>
<StmtList> → <StmtList> <Stmt>
<Expr> → <Id>
<Expr> → <Num>
<Expr> → <Expr> <Optr> <Expr>
  <Id> → x
  <Id> → y
  <Num> → 0
  <Num> → 1
  <Num> → 9
  <Optr> → >
  <Optr> → +

```

语法分析阶段的主题之二: 构建语法分析树

$\langle \text{Stmt} \rangle$			
if ($\langle \text{Expr} \rangle$)	$\langle \text{Stmt} \rangle$
if ($\langle \text{Expr} \rangle$ $\langle \text{Optr} \rangle$ $\langle \text{Expr} \rangle$)	$\langle \text{Stmt} \rangle$
if ($\langle \text{Id} \rangle$ $\langle \text{Optr} \rangle$ $\langle \text{Expr} \rangle$)	$\langle \text{Stmt} \rangle$
if (x $\langle \text{Optr} \rangle$ $\langle \text{Expr} \rangle$)	$\langle \text{Stmt} \rangle$
if (x > $\langle \text{Expr} \rangle$)	$\langle \text{Stmt} \rangle$
if (x > $\langle \text{Num} \rangle$)	$\langle \text{Stmt} \rangle$
if (x > 9)	$\langle \text{Stmt} \rangle$
if (x > 9	{	$\langle \text{StmtList} \rangle$ }
if (x > 9	{	$\langle \text{StmtList} \rangle$ $\langle \text{Stmt} \rangle$ }
if (x > 9	{	$\langle \text{Stmt} \rangle$ $\langle \text{Stmt} \rangle$ }
if (x > 9	{	$\langle \text{Id} \rangle = \langle \text{Expr} \rangle ;$ $\langle \text{Stmt} \rangle$ }
if (x > 9	{	x = $\langle \text{Expr} \rangle ;$ $\langle \text{Stmt} \rangle$ }
if (x > 9	{	x = $\langle \text{Num} \rangle ;$ $\langle \text{Stmt} \rangle$ }
if (x > 9	{	x = 0 $\langle \text{Stmt} \rangle$ }
if (x > 9	{	x = 0 ; $\langle \text{Id} \rangle = \langle \text{Expr} \rangle ;$ }
if (x > 9	{	x = 0 ; y = $\langle \text{Expr} \rangle ;$ }
if (x > 9	{	x = 0 ; y = $\langle \text{Expr} \rangle \langle \text{Optr} \rangle \langle \text{Expr} \rangle ;$ }
if (x > 9	{	x = 0 ; y = $\langle \text{Id} \rangle \langle \text{Optr} \rangle \langle \text{Expr} \rangle ;$ }
if (x > 9	{	x = 0 ; y = y $\langle \text{Optr} \rangle \langle \text{Expr} \rangle ;$ }
if (x > 9	{	x = 0 ; y = y + $\langle \text{Expr} \rangle ;$ }
if (x > 9	{	x = 0 ; y = y + $\langle \text{Num} \rangle ;$ }
if (x > 9	{	x = 0 ; y = y + 1 ; }

语法分析阶段的主题之三: 错误恢复



报错、**恢复**、继续分析



<Context-Free Grammar>

上下文无关文法

Definition (Context-Free Grammar (CFG); 上下文无关文法)

上下文无关文法 G 是一个四元组 $G = (T, N, P, S)$:

- ▶ T 是**终结符号** (Terminal) 集合, 对应于词法分析器产生的词法单元;
- ▶ N 是**非终结符号** (Non-terminal) 集合;
- ▶ P 是**产生式** (Production) 集合;

$$A \in N \longrightarrow \alpha \in (T \cup N)^*$$

头部/左部 (Head) A : **单个**非终结符

体部/右部 (Body) α : 终结符与非终结符构成的串, 也可以是空串 ϵ

- ▶ S 为**开始** (Start) 符号。要求 $S \in N$ 且唯一。

$$G = (\{a, b\}, \{S\}, P, S)$$

$$S \rightarrow aSb$$

$$S \rightarrow \epsilon$$

$$G = (\{(,)\}, \{S\}, P, S)$$

$$S \rightarrow SS$$

$$S \rightarrow (S)$$

$$S \rightarrow ()$$

$$S \rightarrow \epsilon$$

$stmt \rightarrow$ if $expr$ then $stmt$
| if $expr$ then $stmt$ else $stmt$
| other

条件语句文法

悬空 (Dangling)-else 文法

$S \rightarrow \text{if } E \text{ then } S \text{ else } S$

$S \rightarrow \text{begin } S L$

$S \rightarrow \text{print } E$

$L \rightarrow \text{end}$

$L \rightarrow ; S L$

$E \rightarrow \text{num} = \text{num}$

约定: 如果没有明确指定, 第一个产生式的头部就是开始符号

关于终结符号的约定

1) 下述符号是终结符号:

- ① 在字母表里排在前面的小写字母, 比如 a 、 b 、 c 。
- ② 运算符号, 比如 $+$ 、 $*$ 等。
- ③ 标点符号, 比如括号、逗号等。
- ④ 数字 0 、 1 、 \dots 、 9 。
- ⑤ **黑体字符串**, 比如 **id** 或 **if**。每个这样的字符串表示一个终结符号。

关于非终结符号的约定

2) 下述符号是非终结符号:

- ① 在字母表中排在前面的大写字母, 比如 A 、 B 、 C 。
- ② 字母 S 。它出现时通常表示开始符号。
- ③ 小写、斜体的名字, 比如 $expr$ 或 $stmt$ 。

推导 (Derivation)

$$E \rightarrow E + E \mid E * E \mid (E) \mid \text{id}$$

推导即是將某个产生式的左边**替换**成它的右边

每一步推导需要选择**替换哪个非终结符号**, 以及**使用哪个产生式**

推导 (Derivation)

$$E \rightarrow E + E \mid E * E \mid (E) \mid \mathbf{id}$$

$$E \Rightarrow -E \Rightarrow -(E) \Rightarrow -(E + E) \Rightarrow -(\mathbf{id} + E) \Rightarrow -(\mathbf{id} + \mathbf{id})$$

推导 (Derivation)

$$E \rightarrow E + E \mid E * E \mid (E) \mid \mathbf{id}$$

$$E \Longrightarrow -E \Longrightarrow -(E) \Longrightarrow -(E + E) \Longrightarrow -(\mathbf{id} + E) \Longrightarrow -(\mathbf{id} + \mathbf{id})$$

$E \Longrightarrow -E$: 经过一步推导得出

$E \overset{+}{\Longrightarrow} -(\mathbf{id} + E)$: 经过一步或多步推导得出

$E \overset{*}{\Longrightarrow} -(\mathbf{id} + E)$: 经过零步或多步推导得出

推导 (Derivation)

$$E \rightarrow E + E \mid E * E \mid (E) \mid \mathbf{id}$$

$$E \Rightarrow -E \Rightarrow -(E) \Rightarrow -(E + E) \Rightarrow -(\mathbf{id} + E) \Rightarrow -(\mathbf{id} + \mathbf{id})$$

$E \Rightarrow -E$: 经过一步推导得出

$E \xRightarrow{+} -(\mathbf{id} + E)$: 经过一步或多步推导得出

$E \xRightarrow{*} -(\mathbf{id} + E)$: 经过零步或多步推导得出

$$E \Rightarrow -E \Rightarrow -(E) \Rightarrow -(E + E) \Rightarrow -(E + \mathbf{id}) \Rightarrow -(\mathbf{id} + \mathbf{id})$$

Definition (Sentential Form; 句型)

如果 $S \xRightarrow{*} \alpha$, 且 $\alpha \in (T \cup N)^*$, 则称 α 是文法 G 的一个**句型**。

$$E \rightarrow E + E \mid E * E \mid (E) \mid \text{id}$$

$$E \Rightarrow -E \Rightarrow -(E) \Rightarrow -(E+E) \Rightarrow -(\text{id} + E) \Rightarrow -(\text{id} + \text{id})$$

Definition (Sentential Form; 句型)

如果 $S \xRightarrow{*} \alpha$, 且 $\alpha \in (T \cup N)^*$, 则称 α 是文法 G 的一个**句型**。

$$E \rightarrow E + E \mid E * E \mid (E) \mid \text{id}$$

$$E \Longrightarrow -E \Longrightarrow -(E) \Longrightarrow -(E+E) \Longrightarrow -(\text{id} + E) \Longrightarrow -(\text{id} + \text{id})$$

Definition (Sentence; 句子)

如果 $S \xRightarrow{*} w$, 且 $w \in T^*$, 则称 w 是文法 G 的一个**句子**。

Definition (文法 G 生成的语言 $L(G)$)

文法 G 的**语言** $L(G)$ 是它能推导出的**所有句子**构成的集合。

$$w \in L(G) \iff S \xRightarrow{*} w$$

关于文法 G 的**两个基本问题**:

- ▶ **Membership 问题**: 给定字符串 $x \in T^*$, $x \in L(G)$?
- ▶ $L(G)$ 究竟是什么?

给定字符串 $x \in T^*$, $x \in L(G)$?
(即, 检查 x 是否符合文法 G)

给定字符串 $x \in T^*$, $x \in L(G)$?

(即, 检查 x 是否符合文法 G)

这就是**语法分析器**的任务:

为输入的词法单元流寻找推导、**构建语法分析树**, 或者报错

根节点是文法 G 的起始符号

$\langle \text{Stmt} \rangle$			
if ($\langle \text{Expr} \rangle$)	$\langle \text{Stmt} \rangle$
if ($\langle \text{Expr} \rangle$	$\langle \text{Optr} \rangle$ $\langle \text{Expr} \rangle$	$\langle \text{Stmt} \rangle$
if ($\langle \text{Id} \rangle$	$\langle \text{Optr} \rangle$ $\langle \text{Expr} \rangle$	$\langle \text{Stmt} \rangle$
if (<u>x</u>	$\langle \text{Optr} \rangle$ $\langle \text{Expr} \rangle$	$\langle \text{Stmt} \rangle$
if (x	> $\langle \text{Expr} \rangle$	$\langle \text{Stmt} \rangle$
if (x	> $\langle \text{Num} \rangle$	$\langle \text{Stmt} \rangle$
if (x	> 9	$\langle \text{Stmt} \rangle$
if (x	> 9	$\langle \text{StmtList} \rangle$
if (x	> 9	{ $\langle \text{StmtList} \rangle$ $\langle \text{Stmt} \rangle$ }
if (x	> 9	{ $\langle \text{Stmt} \rangle$ $\langle \text{Stmt} \rangle$ }
if (x	> 9	{ <u>$\langle \text{Id} \rangle = \langle \text{Expr} \rangle$</u> ; $\langle \text{Stmt} \rangle$ }
if (x	> 9	{ <u>x = $\langle \text{Expr} \rangle$</u> ; $\langle \text{Stmt} \rangle$ }
if (x	> 9	{ x = <u>$\langle \text{Num} \rangle$</u> ; $\langle \text{Stmt} \rangle$ }
if (x	> 9	{ x = 0 ; $\langle \text{Stmt} \rangle$ }
if (x	> 9	{ x = 0 ; <u>$\langle \text{Id} \rangle = \langle \text{Expr} \rangle$</u> ; }
if (x	> 9	{ x = 0 ; y = $\langle \text{Expr} \rangle$; }
if (x	> 9	{ x = 0 ; y = <u>$\langle \text{Expr} \rangle$</u> $\langle \text{Optr} \rangle$ $\langle \text{Expr} \rangle$; }
if (x	> 9	{ x = 0 ; y = <u>$\langle \text{Id} \rangle$</u> $\langle \text{Optr} \rangle$ $\langle \text{Expr} \rangle$; }
if (x	> 9	{ x = 0 ; y = y <u>$\langle \text{Optr} \rangle$</u> $\langle \text{Expr} \rangle$; }
if (x	> 9	{ x = 0 ; y = y + $\langle \text{Expr} \rangle$; }
if (x	> 9	{ x = 0 ; y = y + <u>$\langle \text{Num} \rangle$</u> ; }
if (x	> 9	{ x = 0 ; y = y + 1 ; }

叶子节点是输入的词法单元流

常用的语法分析器以**自顶向下**或**自底向上**的方式构建中间部分

$L(G)$ 是什么?

这是**程序设计语言设计者**需要考虑的问题

$$S \rightarrow SS$$

$$S \rightarrow (S)$$

$$S \rightarrow ()$$

$$S \rightarrow \epsilon$$

$$L(G) =$$

$$S \rightarrow SS$$

$$S \rightarrow (S)$$

$$S \rightarrow ()$$

$$S \rightarrow \epsilon$$

$$L(G) = \{\text{良匹配括号串}\}$$

$$S \rightarrow SS$$

$$S \rightarrow (S)$$

$$S \rightarrow ()$$

$$S \rightarrow \epsilon$$

$$S \rightarrow aSb$$

$$S \rightarrow \epsilon$$

$$L(G) =$$

$$L(G) = \{\text{良匹配括号串}\}$$

$$S \rightarrow SS$$

$$S \rightarrow (S)$$

$$S \rightarrow ()$$

$$S \rightarrow \epsilon$$

$$S \rightarrow aSb$$

$$S \rightarrow \epsilon$$

$$L(G) = \{a^n b^n \mid n \geq 0\}$$

$$L(G) = \{\text{良匹配括号串}\}$$

字母表 $\Sigma = \{a, b\}$ 上的所有回文串 (Palindrome) 构成的语言

字母表 $\Sigma = \{a, b\}$ 上的所有**回文串** (Palindrome) 构成的语言

$$S \rightarrow aSa$$

$$S \rightarrow bSb$$

$$S \rightarrow a$$

$$S \rightarrow b$$

$$S \rightarrow \epsilon$$

字母表 $\Sigma = \{a, b\}$ 上的所有回文串 (Palindrome) 构成的语言

$$S \rightarrow aSa$$

$$S \rightarrow bSb$$

$$S \rightarrow a$$

$$S \rightarrow b$$

$$S \rightarrow \epsilon$$

$$S \rightarrow aSa \mid bSb \mid a \mid b \mid \epsilon$$

$$\{b^n a^m b^{2n} \mid n \geq 0, m \geq 0\}$$

$$\{b^n a^m b^{2n} \mid n \geq 0, m \geq 0\}$$

$$S \rightarrow bSbb \mid A$$

$$A \rightarrow aA \mid \epsilon$$

$$\{x \in \{a, b\}^* \mid x \text{ 中 } a, b \text{ 个数相同}\}$$

$\{x \in \{a, b\}^* \mid x \text{ 中 } a, b \text{ 个数相同}\}$

$$V \rightarrow aVbV \mid bVaV \mid \epsilon$$

$\{x \in \{a, b\}^* \mid x \text{ 中 } a, b \text{ 个数不同}\}$

$\{x \in \{a, b\}^* \mid x \text{ 中 } a, b \text{ 个数不同}\}$

$$S \rightarrow T \mid U$$

$$T \rightarrow VaT \mid VaV$$

$$U \rightarrow VbU \mid VbV$$

$$V \rightarrow aVbV \mid bVaV \mid \epsilon$$

$\{x \in \{a, b\}^* \mid x \text{ 中 } a, b \text{ 个数不同}\}$

$$S \rightarrow T \mid U$$
$$T \rightarrow VaT \mid VaV$$
$$U \rightarrow VbU \mid VbV$$
$$V \rightarrow aVbV \mid bVaV \mid \epsilon$$


练习 (非作业): 证明之



L-System

(注: 这不是上下文无关文法, 但精神上高度一致, 并且更有趣)

variables : A B

constants : + -

start : A

rules : $(A \rightarrow B-A-B)$, $(B \rightarrow A+B+A)$

angle : 60°

A, B : 向右移动并画线

+: 左转

-: 右转

每一步都**并行地**应用**所有**规则

A

$B - A - B$

A

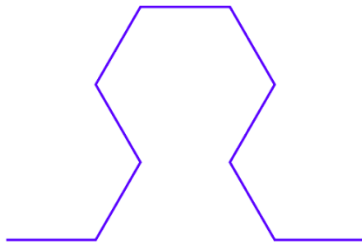
$B - A - B$

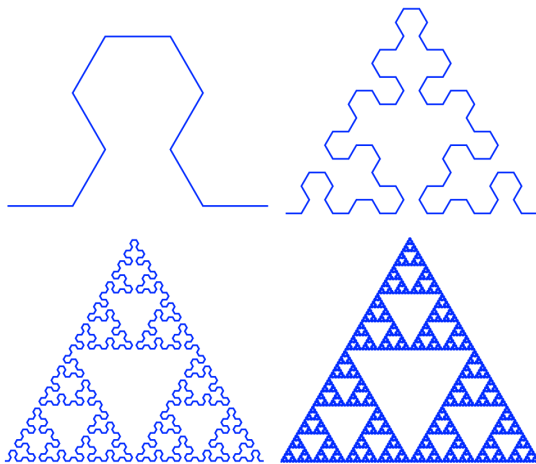
$A + B + A - B - A - B - A + B + A$

A

$B - A - B$

$A + B + A - B - A - B - A + B + A$





Sierpinski arrowhead curve ($n = 2, 4, 6, 8$)

variables : $X Y$

constants : $F + -$

start : FX

rules : $(X \rightarrow X+YF+), (Y \rightarrow -FX-Y)$

angle : 90°

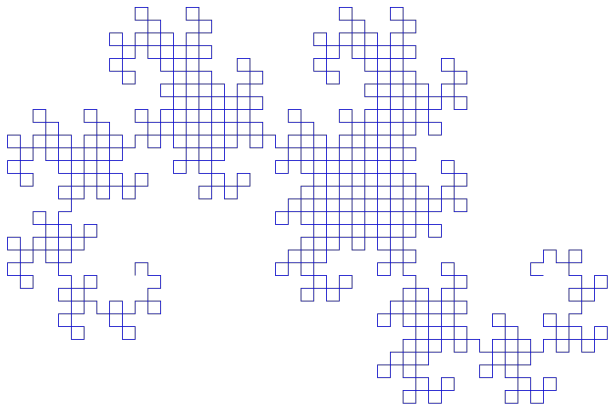
F : 向上移动并画线

$+$: 右转

$-$: 左转

X : 仅用于展开, 在作画时被忽略

每一步都**并行地**应用**所有**规则



Dragon Curve ($n = 10$)

最左 (leftmost) 推导与最右 (rightmost) 推导

$$E \rightarrow E + E \mid E * E \mid (E) \mid \mathbf{id}$$

$$E \xRightarrow{\text{lm}} -E \xRightarrow{\text{lm}} -(E) \xRightarrow{\text{lm}} -(E + E) \xRightarrow{\text{lm}} -(\mathbf{id} + E) \xRightarrow{\text{lm}} -(\mathbf{id} + \mathbf{id})$$

最左 (leftmost) 推导与最右 (rightmost) 推导

$$E \rightarrow E + E \mid E * E \mid (E) \mid \mathbf{id}$$

$$E \xRightarrow{\text{lm}} -E \xRightarrow{\text{lm}} -(E) \xRightarrow{\text{lm}} -(E + E) \xRightarrow{\text{lm}} -(\mathbf{id} + E) \xRightarrow{\text{lm}} -(\mathbf{id} + \mathbf{id})$$

$E \xRightarrow{\text{lm}} -E$: 经过一步最左推导得出

$E \xRightarrow{+}_{\text{lm}} -(\mathbf{id} + E)$: 经过一步或多步最左推导得出

$E \xRightarrow{*}_{\text{lm}} -(\mathbf{id} + E)$: 经过零步或多步最左推导得出

最左 (leftmost) 推导与最右 (rightmost) 推导

$$E \rightarrow E + E \mid E * E \mid (E) \mid \mathbf{id}$$

$$E \xRightarrow{\text{lm}} -E \xRightarrow{\text{lm}} -(E) \xRightarrow{\text{lm}} -(E + E) \xRightarrow{\text{lm}} -(\mathbf{id} + E) \xRightarrow{\text{lm}} -(\mathbf{id} + \mathbf{id})$$

$E \xRightarrow{\text{lm}} -E$: 经过一步最左推导得出

$E \xRightarrow[\text{lm}]{+} -(\mathbf{id} + E)$: 经过一步或多步最左推导得出

$E \xRightarrow[\text{lm}]{*} -(\mathbf{id} + E)$: 经过零步或多步最左推导得出

$$E \xRightarrow{\text{rm}} -E \xRightarrow{\text{rm}} -(E) \xRightarrow{\text{rm}} -(E + E) \xRightarrow{\text{rm}} -(E + \mathbf{id}) \xRightarrow{\text{rm}} -(\mathbf{id} + \mathbf{id})$$

Definition (Left-sentential Form; 最左句型)

如果 $S \xRightarrow[\text{lm}]{*} \alpha$, 且 $\alpha \in (T \cup N)^*$, 则称 α 是文法 G 的一个**最左句型**。

$$E \xRightarrow{\text{lm}} -E \xRightarrow{\text{lm}} -(E) \xRightarrow{\text{lm}} -(E + E) \xRightarrow{\text{lm}} -(\text{id} + E) \xRightarrow{\text{lm}} -(\text{id} + \text{id})$$

Definition (Left-sentential Form; 最左句型)

如果 $S \xRightarrow[\text{lm}]{*} \alpha$, 且 $\alpha \in (T \cup N)^*$, 则称 α 是文法 G 的一个**最左句型**。

$$E \xRightarrow{\text{lm}} -E \xRightarrow{\text{lm}} -(E) \xRightarrow{\text{lm}} -(E + E) \xRightarrow{\text{lm}} -(\text{id} + E) \xRightarrow{\text{lm}} -(\text{id} + \text{id})$$

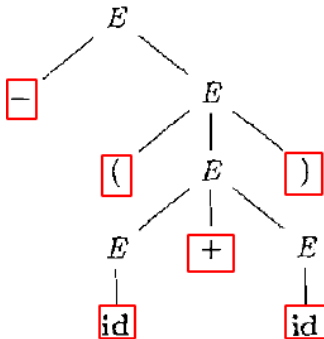
Definition (Right-sentential Form; 最右句型)

如果 $S \xRightarrow[\text{rm}]{*} \alpha$, 且 $\alpha \in (T \cup N)^*$, 则称 α 是文法 G 的一个**最右句型**。

$$E \xRightarrow{\text{rm}} -E \xRightarrow{\text{rm}} -(E) \xRightarrow{\text{rm}} -(E + E) \xRightarrow{\text{rm}} -(E + \text{id}) \xRightarrow{\text{rm}} -(\text{id} + \text{id})$$

语法分析树

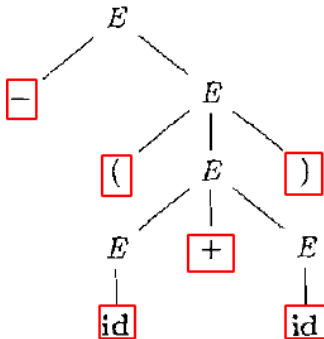
语法分析树是静态的, 它不关心动态的推导顺序



一棵语法分析树对应多个推导

语法分析树

语法分析树是静态的, 它不关心动态的推导顺序



一棵语法分析树对应多个推导

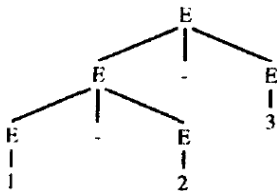
但是, 一棵语法分析树与**最左 (最右) 推导**一一对应

$$E \rightarrow E + E \mid E - E \mid E * E \mid E / E \mid (E) \mid \text{id} \mid \text{number}$$

1 - 2 - 3 的语法树?

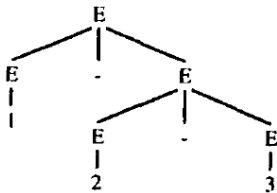
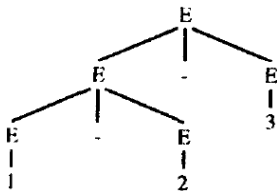
$$E \rightarrow E + E \mid E - E \mid E * E \mid E / E \mid (E) \mid \text{id} \mid \text{number}$$

1 - 2 - 3 的语法树?



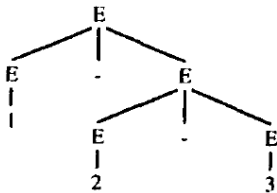
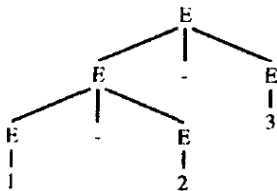
$$E \rightarrow E + E \mid E - E \mid E * E \mid E / E \mid (E) \mid \text{id} \mid \text{number}$$

1 - 2 - 3 的语法树?



$$E \rightarrow E + E \mid E - E \mid E * E \mid E / E \mid (E) \mid \text{id} \mid \text{number}$$

1 - 2 - 3 的语法树?



Definition (二义性(Ambiguous) 文法)

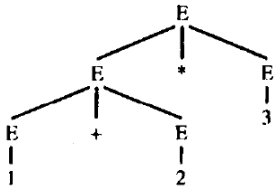
如果 $L(G)$ 中的**某个**句子有一个**以上**语法树/最左推导/最右推导,
则文法 G 是二义性的。

$$E \rightarrow E + E \mid E - E \mid E * E \mid E / E \mid (E) \mid \mathbf{id} \mid \mathbf{number}$$

$1 + 2 * 3$ 的语法树?

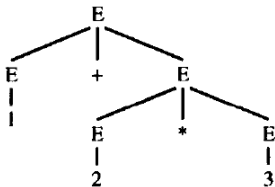
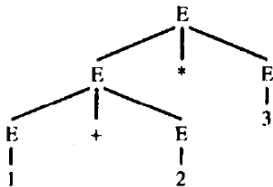
$$E \rightarrow E + E \mid E - E \mid E * E \mid E / E \mid (E) \mid \text{id} \mid \text{number}$$

1 + 2 * 3 的语法树?



$$E \rightarrow E + E \mid E - E \mid E * E \mid E / E \mid (E) \mid \text{id} \mid \text{number}$$

1 + 2 * 3 的语法树?



$stmt \rightarrow$ if $expr$ then $stmt$
 | if $expr$ then $stmt$ else $stmt$
 | other

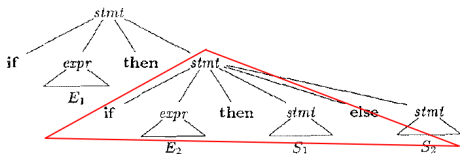
“悬空-else” 文法

if E_1 then if E_2 then S_1 else S_2

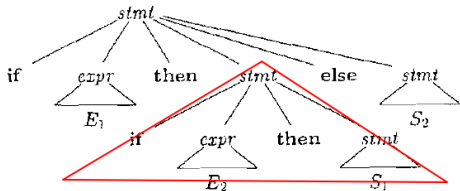
$stmt \rightarrow$ if $expr$ then $stmt$
 $\quad \mid$ if $expr$ then $stmt$ else $stmt$
 $\quad \mid$ other

“悬空-else” 文法

if E_1 then if E_2 then S_1 else S_2



if E_1 then (if E_2 then S_1 else S_2)

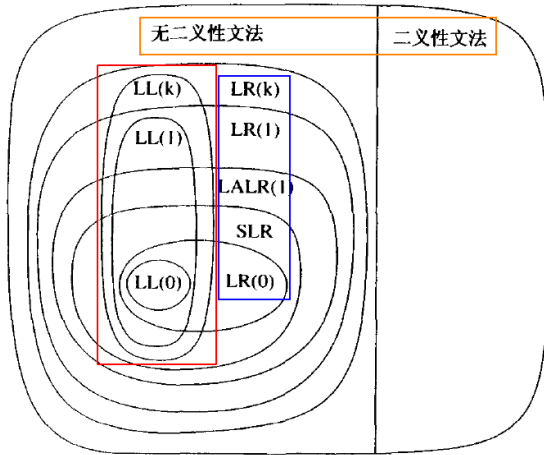


if E_1 then (if E_2 then S_1) else S_2

二义性文法

不同的语法分析树产生不同的语义





所有语法分析器都要求文法是**无二义性**的

二义性文法

Q: 如何**识别**二义性文法?

Q: 如何**消除**文法的二义性?

二义性文法

Q : 如何**识别**二义性文法?



Q : 如何**消除**文法的二义性?

这是**不可判定**的问题

二义性文法

Q : 如何**识别**二义性文法?



Q : 如何**消除**文法的二义性?

LEARN BY EXAMPLES

这是**不可判定**的问题

$$E \rightarrow E + E \mid E - E \mid E * E \mid E / E \mid (E) \mid \text{id} \mid \text{number}$$

四则运算均是**左结合**的

优先级: 括号最先, 先乘除后加减

二义性表达式文法以**相同的方式**处理所有的算术运算符

要消除二义性, 需要**区别对待**不同的运算符

$$E \rightarrow E + E \mid E - E \mid E * E \mid E / E \mid (E) \mid \text{id} \mid \text{number}$$

四则运算均是**左结合**的

优先级: 括号最先, 先乘除后加减

二义性表达式文法以**相同的方式**处理所有的算术运算符

要消除二义性, 需要**区别对待**不同的运算符

将运算的“先后”顺序信息编码到语法树的“层次”结构中

$$E \rightarrow E + E \mid \mathbf{id}$$

$$E \rightarrow E + E \mid \mathbf{id}$$

$$E \rightarrow E + T$$

$$T \rightarrow \mathbf{id}$$

左结合文法

$$E \rightarrow E + E \mid \mathbf{id}$$

$$E \rightarrow E + T$$

$$T \rightarrow \mathbf{id}$$

左结合文法

$$E \rightarrow T + E$$

$$T \rightarrow \mathbf{id}$$

右结合文法

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

$$E \rightarrow E + T$$

$$T \rightarrow \text{id}$$

左结合文法

$$E \rightarrow T + E$$

$$T \rightarrow \text{id}$$

右结合文法

使用左 (右) 递归实现左 (右) 结合

$$E \rightarrow E + E \mid E * E \mid (E) \mid \mathbf{id}$$

$$E \rightarrow E + E \mid E * E \mid (E) \mid \mathbf{id}$$

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

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

$$F \rightarrow (E) \mid \mathbf{id}$$

括号最先, 先乘后加文法

$$E \rightarrow E + E \mid E - E \mid E * E \mid E / E \mid (E) \mid \mathbf{id} \mid \mathbf{number}$$

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

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

$$F \rightarrow (E) \mid \mathbf{id} \mid \mathbf{number}$$

无二义性的表达式文法

E : 表达式(*expression*); T : 项(*term*) F : 因子(*factor*)

$$E \rightarrow E + E \mid E - E \mid E * E \mid E / E \mid (E) \mid \mathbf{id} \mid \mathbf{number}$$

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

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

$$F \rightarrow (E) \mid \mathbf{id} \mid \mathbf{number}$$

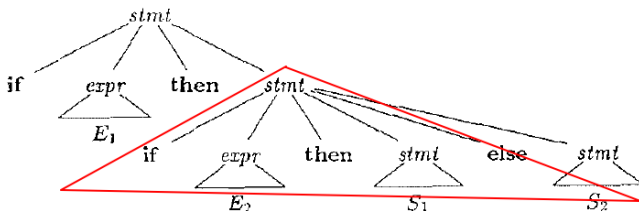
无二义性的表达式文法

E : 表达式(*expression*); T : 项(*term*) F : 因子(*factor*)

将运算的“先后”顺序信息编码到语法树的“层次”结构中

$stmt \rightarrow$ if $expr$ then $stmt$
 $\quad \mid$ if $expr$ then $stmt$ else $stmt$
 $\quad \mid$ other

if E_1 then if E_2 then S_1 else S_2



“每个else与最近的尚未匹配的then匹配”

$$\begin{aligned}
 stmt &\rightarrow \text{if } expr \text{ then } stmt \\
 &\quad | \text{ if } expr \text{ then } stmt \text{ else } stmt \\
 &\quad | \text{ other}
 \end{aligned}$$

$stmt$	\rightarrow	$matched_stmt$
	$ $	$open_stmt$
$matched_stmt$	\rightarrow	$\text{if } expr \text{ then } matched_stmt \text{ else } matched_stmt$
	$ $	other
$open_stmt$	\rightarrow	$\text{if } expr \text{ then } stmt$
	$ $	$\text{if } expr \text{ then } matched_stmt \text{ else } open_stmt$

基本思想: **then** 与 **else** 之间的语句必须是“已匹配的”

我也看不懂啊

~~“我不想去上课啊妈妈”~~

“清醒一点！你是老师啊！”



我们要证明**两**件事情



我们要证明**两**件事情

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



我们要证明**两**件事情

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

G' 是无二义性的

$$\begin{aligned}
 stmt &\rightarrow \text{if } expr \text{ then } stmt \\
 &\quad | \text{ if } expr \text{ then } stmt \text{ else } stmt \\
 &\quad | \text{ other}
 \end{aligned}$$

$stmt$	\rightarrow	$matched_stmt$
	$ $	$open_stmt$
$matched_stmt$	\rightarrow	$\text{if } expr \text{ then } matched_stmt \text{ else } matched_stmt$
	$ $	other
$open_stmt$	\rightarrow	$\text{if } expr \text{ then } stmt$
	$ $	$\text{if } expr \text{ then } matched_stmt \text{ else } open_stmt$

$$\begin{aligned}
 stmt &\rightarrow \text{if } expr \text{ then } stmt \\
 &\quad | \text{ if } expr \text{ then } stmt \text{ else } stmt \\
 &\quad | \text{ other}
 \end{aligned}$$

$stmt$	\rightarrow	$matched_stmt$
	$ $	$open_stmt$
$matched_stmt$	\rightarrow	$\text{if } expr \text{ then } matched_stmt \text{ else } matched_stmt$
	$ $	other
$open_stmt$	\rightarrow	$\text{if } expr \text{ then } stmt$
	$ $	$\text{if } expr \text{ then } matched_stmt \text{ else } open_stmt$

$$L(G') \subseteq L(G)$$

$$L(G) \subseteq L(G')$$

$$\begin{aligned}
 stmt &\rightarrow \text{if } expr \text{ then } stmt \\
 &\quad | \text{ if } expr \text{ then } stmt \text{ else } stmt \\
 &\quad | \text{ other}
 \end{aligned}$$

$$\begin{aligned}
 stmt &\rightarrow matched_stmt \\
 &\quad | open_stmt \\
 matched_stmt &\rightarrow \text{if } expr \text{ then } matched_stmt \text{ else } matched_stmt \\
 &\quad | \text{ other} \\
 open_stmt &\rightarrow \text{if } expr \text{ then } stmt \\
 &\quad | \text{ if } expr \text{ then } matched_stmt \text{ else } open_stmt
 \end{aligned}$$

$$L(G') \subseteq L(G)$$

$$L(G) \subseteq L(G')$$

对推导步数作数学归纳

G' 是无二义性的

$stmt$	\rightarrow	$matched_stmt$
	$ $	$open_stmt$
$matched_stmt$	\rightarrow	$if\ expr\ then\ matched_stmt\ else\ matched_stmt$
	$ $	$other$
$open_stmt$	\rightarrow	$if\ expr\ then\ stmt$
	$ $	$if\ expr\ then\ matched_stmt\ else\ open_stmt$

G' 是无二义性的

$stmt$	\rightarrow	$matched_stmt$
	$ $	$open_stmt$
$matched_stmt$	\rightarrow	$if\ expr\ then\ matched_stmt\ else\ matched_stmt$
	$ $	$other$
$open_stmt$	\rightarrow	$if\ expr\ then\ stmt$
	$ $	$if\ expr\ then\ matched_stmt\ else\ open_stmt$

每个句子对应的语法分析树是唯一的

G' 是无二义性的

$stmt$	\rightarrow	$matched_stmt$
	$ $	$open_stmt$
$matched_stmt$	\rightarrow	$if\ expr\ then\ matched_stmt\ else\ matched_stmt$
	$ $	$other$
$open_stmt$	\rightarrow	$if\ expr\ then\ stmt$
	$ $	$if\ expr\ then\ matched_stmt\ else\ open_stmt$

每个句子对应的**语法分析树**是唯一的

只需证明: 每个非终结符的“**展开**”方式是唯一的

G' 是无二义性的

$stmt$	\rightarrow	$matched_stmt$
	$ $	$open_stmt$
$matched_stmt$	\rightarrow	$if\ expr\ then\ matched_stmt\ else\ matched_stmt$
	$ $	$other$
$open_stmt$	\rightarrow	$if\ expr\ then\ stmt$
	$ $	$if\ expr\ then\ matched_stmt\ else\ open_stmt$

每个句子对应的语法分析树是唯一的

只需证明: 每个非终结符的“展开”方式是唯一的

$$L(matched_stmt) \cap L(open_stmt) = \emptyset$$

G' 是无二义性的

$stmt$	\rightarrow	$matched_stmt$
	$ $	$open_stmt$
$matched_stmt$	\rightarrow	$if\ expr\ then\ matched_stmt\ else\ matched_stmt$
	$ $	$other$
$open_stmt$	\rightarrow	$if\ expr\ then\ stmt$
	$ $	$if\ expr\ then\ matched_stmt\ else\ open_stmt$

每个句子对应的语法分析树是唯一的

只需证明: 每个非终结符的“展开”方式是唯一的

$$L(matched_stmt) \cap L(open_stmt) = \emptyset$$

$$L(matched_stmt_1) \cap L(matched_stmt_2) = \emptyset$$

G' 是无二义性的

$stmt$	\rightarrow	$matched_stmt$
	$ $	$open_stmt$
$matched_stmt$	\rightarrow	$if\ expr\ then\ matched_stmt\ else\ matched_stmt$
	$ $	$other$
$open_stmt$	\rightarrow	$if\ expr\ then\ stmt$
	$ $	$if\ expr\ then\ matched_stmt\ else\ open_stmt$

每个句子对应的语法分析树是唯一的

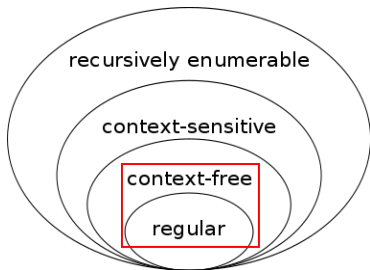
只需证明: 每个非终结符的“展开”方式是唯一的

$$L(matched_stmt) \cap L(open_stmt) = \emptyset$$

$$L(matched_stmt_1) \cap L(matched_stmt_2) = \emptyset$$

$$L(open_stmt_1) \cap L(open_stmt_2) = \emptyset$$

为什么不使用优雅、强大的**正则表达式**描述程序设计语言的语法？



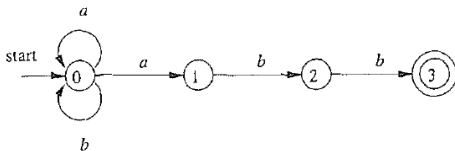
正则表达式的表达能力**严格弱于**上下文无关文法

每个正则表达式 r 对应的语言 $L(r)$ 都可以使用上下文无关文法来描述

$$r = (a|b)^*abb$$

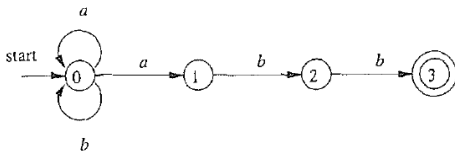
每个正则表达式 r 对应的语言 $L(r)$ 都可以使用上下文无关文法来描述

$$r = (a|b)^*abb$$



每个正则表达式 r 对应的语言 $L(r)$ 都可以使用上下文无关文法来描述

$$r = (a|b)^*abb$$



$$\begin{aligned} A_0 &\rightarrow aA_0 \mid bA_0 \mid aA_1 \\ A_1 &\rightarrow bA_2 \\ A_2 &\rightarrow bA_3 \\ A_3 &\rightarrow \epsilon \end{aligned}$$

此外, 若 $\delta(A_i, \epsilon) = A_j$, 则添加 $A_i \rightarrow A_j$

$$S \rightarrow aSb$$

$$S \rightarrow \epsilon$$

$$L = \{a^n b^n \mid n \geq 0\}$$

该语言**无法**使用正则表达式来描述

Theorem

$L = \{a^n b^n \mid n \geq 0\}$ 无法使用正则表达式描述。

Theorem

$L = \{a^n b^n \mid n \geq 0\}$ 无法使用正则表达式描述。

反证法

Theorem

$L = \{a^n b^n \mid n \geq 0\}$ 无法使用正则表达式描述。

反证法

假设存在正则表达式 $r: L(r) = L$

Theorem

$L = \{a^n b^n \mid n \geq 0\}$ 无法使用正则表达式描述。

反证法

假设存在正则表达式 $r: L(r) = L$

则存在**有限**状态自动机 $D(r): L(D(r)) = L$; 设其状态数为 k

Theorem

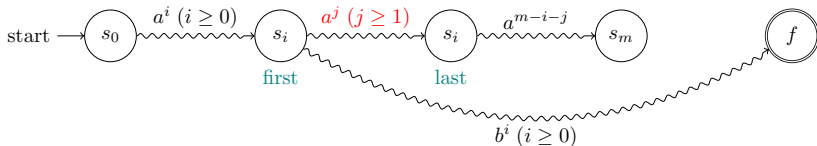
$L = \{a^n b^n \mid n \geq 0\}$ 无法使用正则表达式描述。

反证法

假设存在正则表达式 $r: L(r) = L$

则存在有限状态自动机 $D(r): L(D(r)) = L$; 设其状态数为 k

考虑输入 $a^m (m > k)$



Theorem

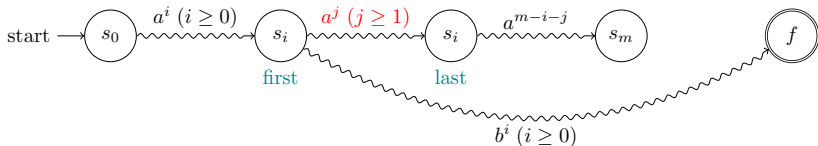
$L = \{a^n b^n \mid n \geq 0\}$ 无法使用正则表达式描述。

反证法

假设存在正则表达式 $r: L(r) = L$

则存在有限状态自动机 $D(r): L(D(r)) = L$; 设其状态数为 k

考虑输入 $a^m (m > k)$



$D(r)$ 也能接受 $a^{i+j}b^i$; 矛盾!

$$L = \{a^n b^n \mid n \geq 0\}$$

Pumping Lemma for Regular Languages

$$L = \{a^n b^n \mid n \geq 0\}$$

Pumping Lemma for Regular Languages

$$L = \{a^n b^n c^n \mid n \geq 0\}$$

Pumping Lemma for Context-free Languages

Thank
You!



Office 926

hfwei@nju.edu.cn