



中国科学技术大学
University of Science and Technology of China

语法分析 I

《编译原理和技术》

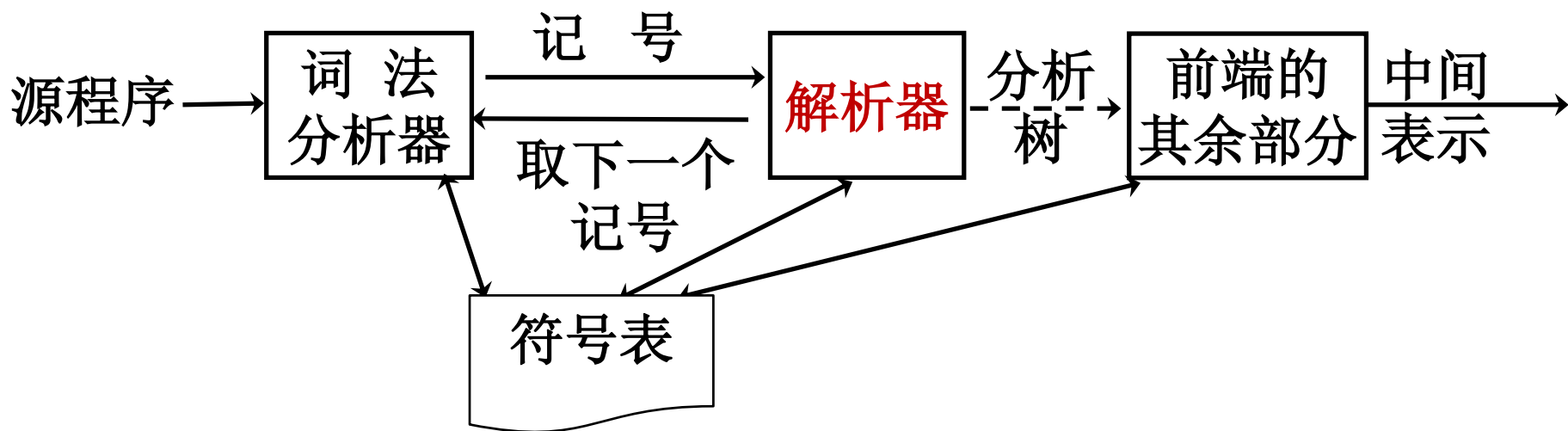
张昱

0551-63603804, yuzhang@ustc.edu.cn

中国科学技术大学
计算机科学与技术学院



本章内容



- 语法的形式描述：上下文无关文法
- 语法分析：自上而下、自下而上
- 语法分析器(parser、syntax analyzer)的自动生成
 - LL(k)、LL(*)、SLR、LR(k)、LALR



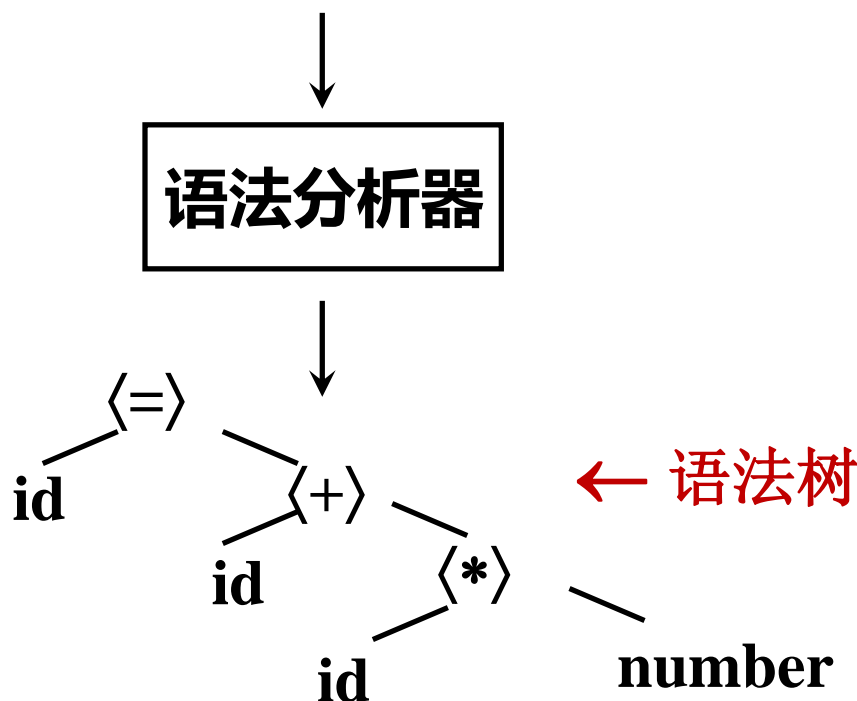
语法分析器

□ 语法树(syntax tree)

■ 源程序的**层次化**语法结构

■ 是程序的一种
中间表示

$\langle \text{id}, 1 \rangle \langle = \rangle \langle \text{id}, 2 \rangle \langle + \rangle \langle \text{id}, 3 \rangle \langle * \rangle \langle 60 \rangle$ ← 记号流





3.1 上下文无关文法

- 正规式的表达能力
- 上下文无关文法
 - 定义、推导、二义性
 - 名词：语言、文法等价、句型、句子



正规式的表达能力不足

□ 正规式的表达能力

- 定义一些简单的语言，能表示给定结构的固定次数的重复或者没有指定次数的重复

例： $a(ba)^5, a(ba)^*$

- 不能用于描述配对或嵌套的结构

例1：配对括号串的集合，如不能表达 $(^n)^n, n \geq 1$

例2： $\{w^c w \mid w \text{ 是由 } a \text{ 和 } b \text{ 组成的串}\}$

原因： n 不固定，后面的串要依据前面不定长的串 w 来确定；

有限自动机无法记录访问同一状态的次数



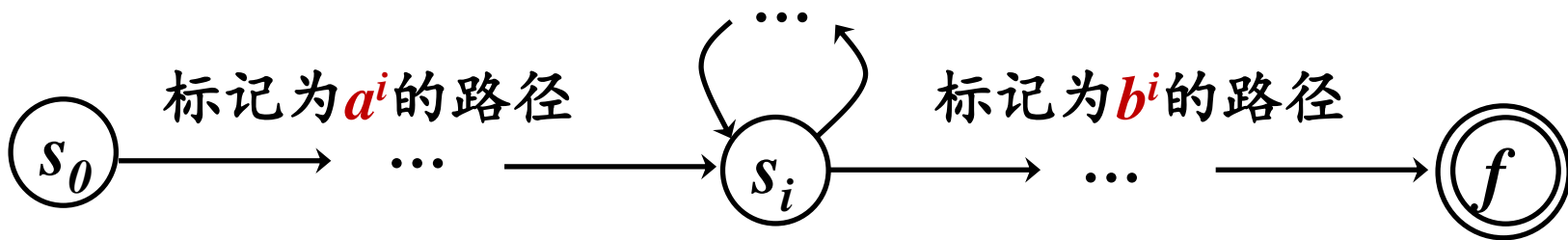
正规式的表达能力不足

例： $L = \{a^n b^n \mid n \geq 1\}$, L 不能用正规式描述

反证法

- 若存在接受 L 的 DFA D , 状态数为 $k+1$ 个 (有限个)
- 设 D 读完 $\varepsilon, a, aa, \dots, a^k$ 分别到达状态 s_0, s_1, \dots, s_k
- 至少有两个状态相同, 例如是 s_i 和 s_j , 则 $a^j b^i$ 属于 L

标记为 a^{j-i} 的路径





上下文无关文法的定义

Context-free **Grammar** (CFG)

注: **Syntax**-语法

□ CFG是四元组 (V_T, V_N, S, P)

V_T : 终结符(terminal, 记号名, 即token的第1元)集合

V_N : 非终结符(nonterminal)集合

S : 开始符号(start symbol), 是一个非终结符

P : 产生式(production)集合

产生式的形式: $A \rightarrow \alpha$, $A \in V_N, \alpha \in (V_T \cup V_N)^*$

有时用 $A ::= \alpha$

■ 例 ($\{\text{id}, +, *, -, (,)\}, \{\text{expr}, \text{op}\}, \text{expr}, P$)

$\text{expr} \rightarrow \text{expr op expr}$ $\text{expr} \rightarrow (\text{expr})$ $\text{expr} \rightarrow - \text{expr}$

$\text{expr} \rightarrow \text{id}$

$\text{op} \rightarrow +$

$\text{op} \rightarrow *$



CFG的简化表示

□ 表达式的CFG的简化表示

■ 引入选择符 |

$$expr \rightarrow expr\ op\ expr \mid (expr) \mid -expr \mid id$$
$$op \rightarrow + \mid *$$

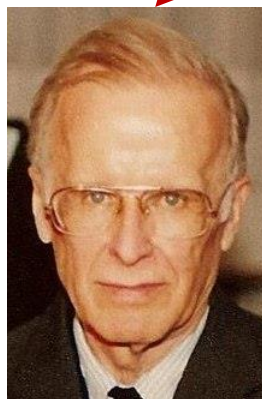
注：+, *是 op 的选择(alternatives)

■ 简化名称

$$E \rightarrow E\ A\ E \mid (E) \mid -E \mid id$$
$$A \rightarrow + \mid *$$

Backus-Naur Form

巴科斯-诺尔范式



John Backus

1977图灵奖

首次在ALGOL 58中实现BNF



Peter Naur

2005图灵奖

在ALGOL 60中发展和简化



图灵奖获得者

□ John Backus (1924-2007)

■ 1977图灵奖获奖成就

- FORTRAN发明组组长
- 提出了BNF

■ 履历

- 弗吉尼亚大学化学(因出勤率低而开除，随和入伍)
- 哥伦比亚大学数学BS
1949, AM 1950
- IBM

■ 获奖演说: Can Programming Be Liberated From the von Neumann Style?

□ Peter Naur (1928-2016)

■ 2005图灵奖获奖成就

- ALGOL 60
- 发展BNF并简化

■ 履历

- 哥本哈根大学天文学BS
1949、博士1957
- 1950-51剑桥: 天气恶劣破坏天文观测计划, 但花很多时间编程以解决天文学中的扰动问题
- 毕业后转向计算机科学
- 获奖演说: Computing vs. Human Thinking



正规式和CFG的比较

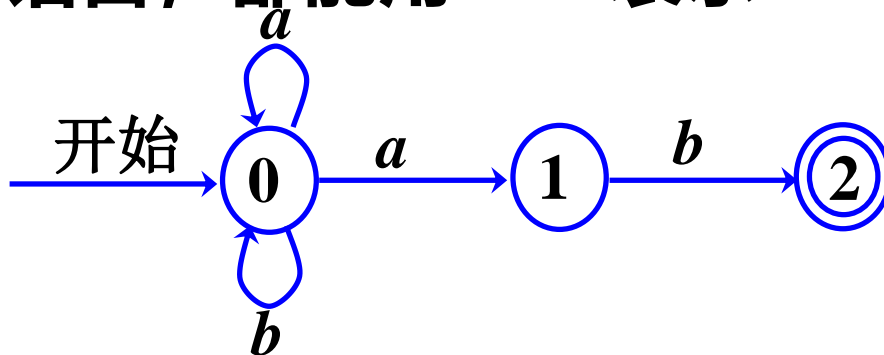
□ 都能表示语言

□ 凡是能用正规式表示的语言，都能用CFG表示

■ 正规式

$(a|b)^*ab$

■ 上下文无关文法CFG



可机械地由NFA变换而得，为每个NFA状态引入一个非终结符，NFA中每条弧对应于产生式的一个分支（选项），

对于接受状态 i ，则引入 $A_i \rightarrow \varepsilon$

$A_0 \rightarrow a A_0 \mid b A_0 \mid a A_1$

$A_1 \rightarrow b A_2$

$A_2 \rightarrow \varepsilon$ （该产生式并不必要）



CFG: 推导(derivation)

□ 推导

- 是从文法推出文法所描述的语言中**合法串**集合的动作
- 把产生式看成重写规则，把符号串中的非终结符用其产生式右部的串来代替

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

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

上述代换序列称为从 E 到 $-(\text{id}+\text{id})$ 的**推导**

$-(\text{id}+\text{id})$ 是 E 的**实例**

记法

0步或多步推导 $S \Rightarrow^* \alpha$ 、一步或多步推导 $S \Rightarrow^+ w$



思考

□ **上下文无关**是什么意思？



思考

□ 上下文无关是什么意思？

■ 指对于文法推导的每一步 $\alpha A \beta \Rightarrow \alpha \gamma \beta$

文法符号串 γ 仅依据 A 的产生式推导，而无需依赖 A 的上下文 α 和 β



语言、文法、句型、句子

□ 上下文无关语言

- 由上下文无关文法 G 产生的语言：从**开始符号 S** 出发，经 \Rightarrow^+ 推导所能到达的**所有仅由终结符组成的串**
- **句型(sentential form)**: $S \Rightarrow^* \alpha$, S 是开始符号, α 是由**终结符和/或非终结符**组成的串, 则 α 是文法 G 的句型
- **句子(sentence)**: 仅由**终结符**组成的句型

□ 等价的文法

- 它们产生同样的语言



最左推导与最右推导

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

□ 最左推导(leftmost derivation)

每步代换**最左边**的非终结符

$$\begin{aligned} E &\Rightarrow_{lm} -E \Rightarrow_{lm} -(E) \Rightarrow_{lm} -(E + E) \\ &\Rightarrow_{lm} -(\text{id} + E) \Rightarrow_{lm} -(\text{id} + \text{id}) \end{aligned}$$

□ 最右推导 (rightmost or canonical, 规范推导)

每步代换**最右边**的非终结符

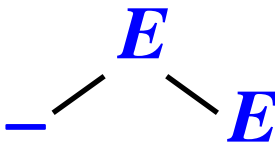
$$\begin{aligned} E &\Rightarrow_{rm} -E \Rightarrow_{rm} -(E) \Rightarrow_{rm} -(E + E) \\ &\Rightarrow_{rm} -(E + \text{id}) \Rightarrow_{rm} -(\text{id} + \text{id}) \end{aligned}$$



分析树(parse tree)

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

-(id+id)最左推导的分析树 (parse tree)

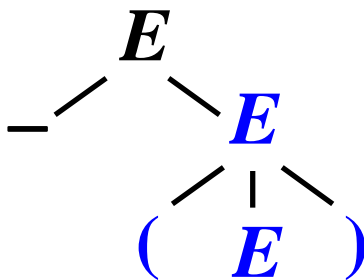




分析树(parse tree)

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

-(id+id)最左推导的分析树 (parse tree)

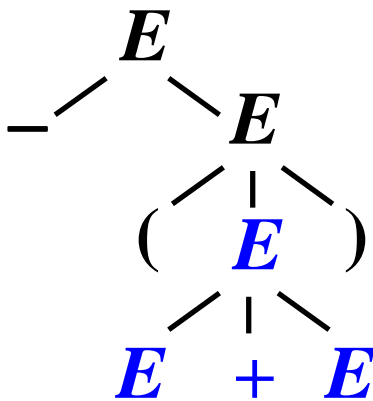




分析树(parse tree)

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

-(id+id)最左推导的分析树 (parse tree)

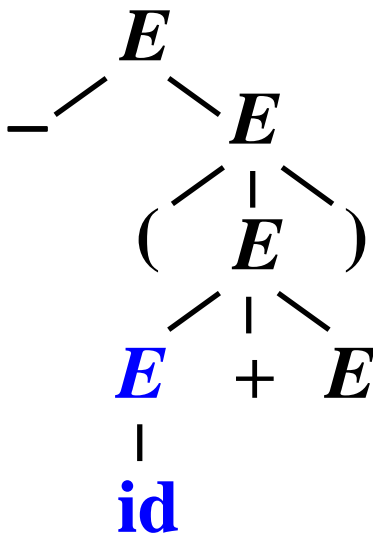




分析树(parse tree)

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

-(id+id)最左推导的分析树 (parse tree)

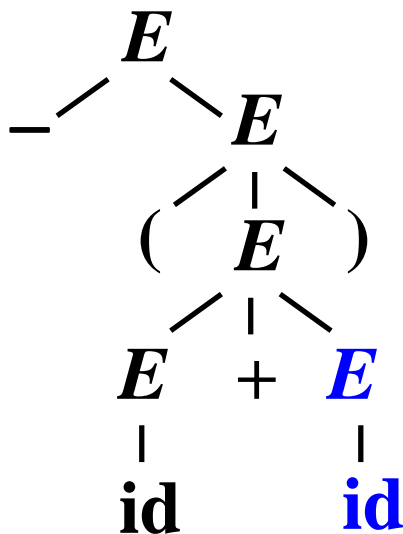




分析树(parse tree)

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

-(id+id)最左推导的分析树 (parse tree)





文法的二义性

文法的某些句子存在**不止一种**最左(最右)推导, 或者**不止一棵**分析树, 则该文法是**二义**的。

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

id*id+id 有两个不同的最左推导

$E \Rightarrow E * E$

$\Rightarrow \text{id} * E$

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

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

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

$E \Rightarrow E + E$

$\Rightarrow E * E + E$

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

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

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



文法的二义性

id*id+id 有两棵不同的分析树

$E \Rightarrow E * E$

$\Rightarrow \text{id} * E$

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

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

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

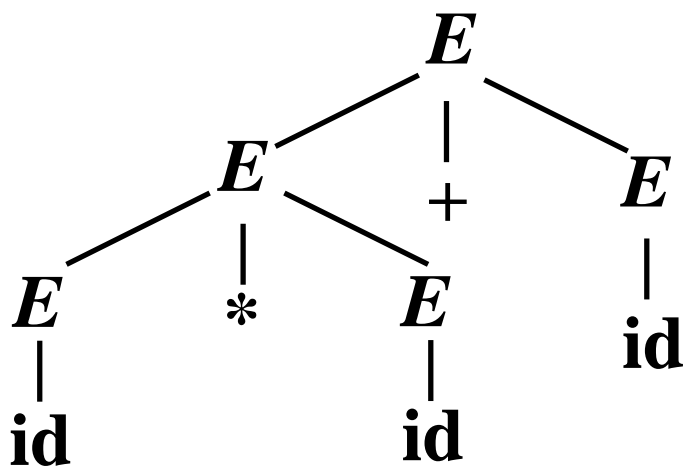
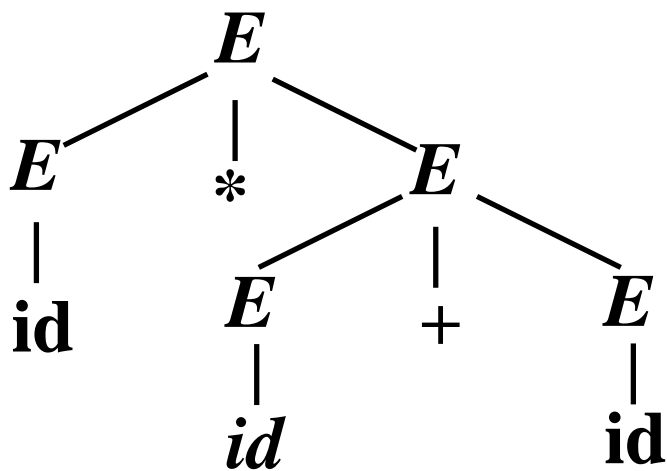
$E \Rightarrow E + E$

$\Rightarrow E * E + E$

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

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

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





3.2 语言 and 文法

- 语言和文法：验证、消除二义
- 词法分析和语法分析的分离
- 非上下文无关文法



验证文法产生的语言

$G : S \rightarrow '(' S ')' S \mid \varepsilon \quad L(G) = \text{配对的括号串的集合}$

□ 按推导步数进行归纳

按任意步推导，推出的是配对括号串

■ 归纳基础(Basis): $S \Rightarrow \varepsilon$

■ 归纳 (Induction)假设: 少于 n 步的推导都产生配对的括号串, 如 $S \Rightarrow^* x, S \Rightarrow^* y$

■ 归纳步骤: n 步的最左推导如下:

$$S \Rightarrow '(' S ')' S \Rightarrow^* '(' x ')' S \Rightarrow^* '(' x ')' y$$



验证文法产生的语言

$G : S \rightarrow '(S \text{ })' S \mid \varepsilon \quad L(G) = \text{配对的括号串的集合}$

□ 按串长进行归纳

任意长度的配对括号串均可由 S 推出

- 归纳基础(Basis): $S \Rightarrow \varepsilon$
- 归纳 (Induction)假设: 长度小于 $2n$ 的配对的括号串都可以从 S 推导出来
- 归纳步骤: 考虑长度为 $2n(n \geq 1)$ 的 $w = '(x \text{ })' y$

$$S \Rightarrow '(S \text{ })' S \Rightarrow^* '(x \text{ })' S \Rightarrow^* '(x \text{ })' y$$



表达式的另一种文法

□ 用一种层次的观点看待表达式

id * id * (id+id) + id * id + id

左递归文法
+ 是自左向右结合

□ 无二义的文法

$expr \rightarrow \textcolor{red}{expr} + term \mid term$

$term \rightarrow \textcolor{red}{term} * factor \mid factor$

$factor \rightarrow id \mid (expr)$

如果改成

$expr \rightarrow term + \textcolor{red}{expr} \mid term$
呢?

+ 是自右向左结合



表达式的另一种文法

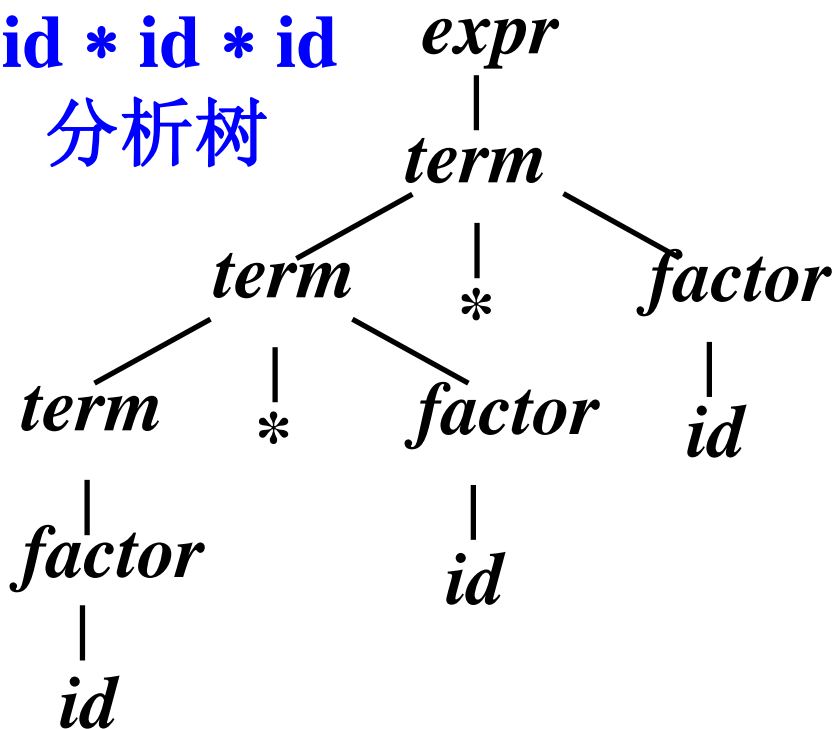
$expr \rightarrow expr + term \mid term$

$term \rightarrow term * factor \mid factor$

$factor \rightarrow id \mid (expr)$

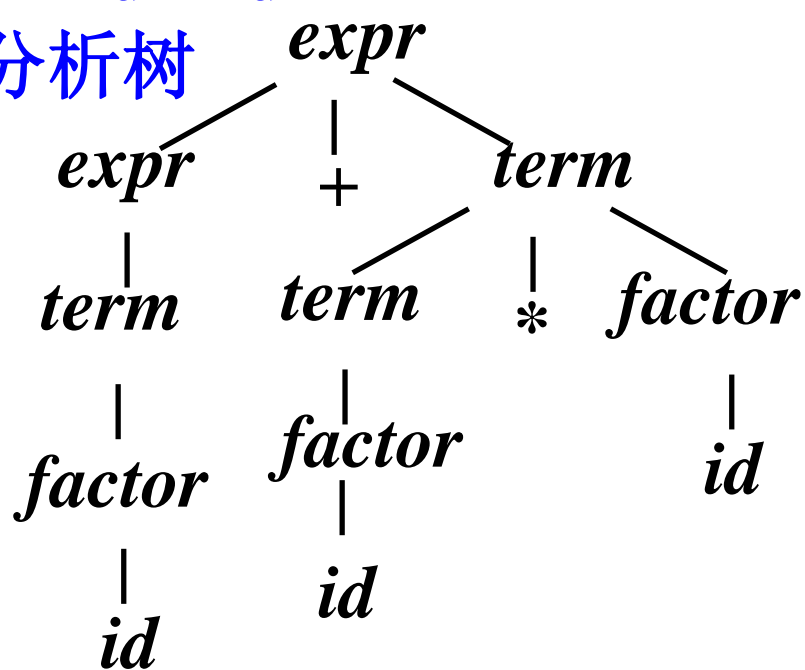
id * id * id

分析树



id + id * id

分析树





消除二义性(Eliminating ambiguity)

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

□ **句型:** if $expr$ then if $expr$ then $stmt$ else $stmt$

有两个最左推导:

$stmt \Rightarrow$ if $expr$ then $stmt$
 \Rightarrow if $expr$ then **if $expr$ then $stmt$ else $stmt$**

$stmt \Rightarrow$ if $expr$ then $stmt$ else $stmt$
 \Rightarrow if $expr$ then **if $expr$ then $stmt$ else $stmt$**



消除二义性

□ 无二义的文法

$$\begin{aligned} stmt &\rightarrow matched_stmt \\ &| unmatched_stmt \end{aligned}$$

else 的就近匹配规则

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



3.2 语言 and 文法

- 语言和文法：验证、消除二义
- 词法分析和语法分析的分离
- 非上下文无关文法



分离词法分析器的理由

□ 为什么要用正规式定义词法

- 词法规则非常简单，不必用上下文无关文法
- 对于词法记号，正规式描述简洁且易于理解
- 从正规式构造出的词法分析器（**DFA**）效率高

□ 分离词法分析和语法分析的好处 (从软件工程看)

- 简化设计，便于编译器前端的模块划分
- 改进编译器的效率
- 增强编译器的可移植性，如输入字符集的特殊性等可以限制在词法分析器中处理



词法分析并入语法分析?

□ 直接从字符流进行语法分析

- **文法复杂化**: 文法中需有反映语言的注释和空白的规则
- **分析器复杂化**: 处理包含注释和空白的分析器, 比注释和空白符已被词法分析器过滤的分析器要复杂得多

□ 分离但在同一遍 (Pass) 中进行

- 是通常编译器的做法



上下文无关文法的优缺点

□ 优点

- 文法给出了精确的、易于理解的语法说明
- 可以给语言定义出层次结构
- 可以基于文法自动产生高效的分析器
- 以文法为基础实现语言便于对语言修改

□ 缺点

- 表达能力不足够，只能描述编程语言中的大部分语法



3.2 语言 and 文法

- 语言和文法：验证、消除二义
- 词法分析和语法分析的分离
- 非上下文无关文法



非上下文无关的语言构造

$$L_1 = \{wcw \mid w \text{ 属于 } (a/b)^*\}$$

用来抽象：标识符的声明应先于其引用

C、Java都不是上下文无关语言

$$L_2 = \{a^n b^m c^n d^m \mid n \geq 0, m \geq 0\}$$

用来抽象：形参个数和实参个数应该相同

$$L_3 = \{a^n b^n c^n \mid n \geq 0\}$$

用来抽象：早先排版描述的一个现象

b e g i n: 5个字母键, 5个回退键, 5个下划线键



形似的上下文无关语言

wcw

$$L_1' = \{wcw^R \mid w \in (a/b)^*\}$$

$$S \rightarrow aSa \mid bSb \mid c$$

$a^n b^m c^n d^m$

$$L_2' = \{a^n b^m c^m d^n \mid n \geq 1, m \geq 1\}$$

$$S \rightarrow aSd \mid aAd$$

$$A \rightarrow bAc \mid bc$$

$a^n b^n c^n$

$$L_2'' = \{a^n b^n c^m d^m \mid n \geq 1, m \geq 1\}$$

$$S \rightarrow AB$$

$$A \rightarrow aAb \mid ab$$

$$B \rightarrow cBd \mid cd$$



形式语言鸟瞰

文法 $G = (V_T, V_N, S, P)$

□ 0型文法: $\alpha \rightarrow \beta, \alpha, \beta \in (V_N \cup V_T)^*, |\alpha| \geq 1$

短语文法

□ 1型文法: $|\alpha| \leq |\beta|$, 但 $S \rightarrow \varepsilon$ 可以例外

上下文有关文法

□ 2型文法: $A \rightarrow \beta, A \in V_N, \beta \in (V_N \cup V_T)^*$

上下文无关文法

□ 3型文法: $A \rightarrow aB$ 或 $A \rightarrow a, A, B \in V_N, a \in V_T$

正规文法



上下文有关文法

例 $L_3 = \{a^n b^n c^n \mid n \geq 1\}$ 的上下文有关文法

$S \rightarrow aSBC$ $S \rightarrow aBC$ $CB \rightarrow BC$ $aB \rightarrow ab$

$bB \rightarrow bb$ $bC \rightarrow bc$ $cC \rightarrow cc$

$a^n b^n c^n$ 的推导过程如下:

$S \Rightarrow^* a^{n-1} S(BC)^{n-1}$ 用 $S \rightarrow aSBC$ $n-1$ 次

$S \Rightarrow^+ a^n (BC)^n$ 用 $S \rightarrow aBC$ 1 次

$S \Rightarrow^+ a^n B^n C^n$ 用 $CB \rightarrow BC$ 交换相邻的 CB

$S \Rightarrow^+ a^n b B^{n-1} C^n$ 用 $aB \rightarrow ab$ 1 次

$S \Rightarrow^+ a^n b^n C^n$ 用 $bB \rightarrow bb$ $n-1$ 次

$S \Rightarrow^+ a^n b^n c C^{n-1}$ 用 $bC \rightarrow bc$ 1 次

$S \Rightarrow^+ a^n b^n c^n$ 用 $cC \rightarrow cc$ $n-1$ 次



例题1 写等价的非二义文法

下面的二义文法描述命题演算公式的语法，
为它写一个等价的非二义文法

$$S \rightarrow S \text{ and } S \mid S \text{ or } S \mid \text{not } S \mid p \mid q \mid '(' S ')'$$

解答

非二义文法的产生式如下：

$$E \rightarrow E \text{ or } T \mid T$$

$$T \rightarrow T \text{ and } F \mid F$$

$$F \rightarrow \text{not } F \mid '(' E ') ' \mid p \mid q$$



例题1 写等价的非二义文法

下面的二义文法描述命题演算公式的语法，
为它写一个等价的非二义文法

$$S \rightarrow S \text{ and } S \mid S \text{ or } S \mid \text{not } S \mid p \mid q \mid '(' S ')'$$

解答

非二义文法的产生式如下：

$$E \rightarrow E \text{ or } T \mid T$$

$$T \rightarrow T \text{ and } F \mid F$$

$$F \rightarrow \text{not } E \mid '(' E ') \mid p \mid q \quad ?$$

not p and q有两种不同的最左推导

not p and q
not p and q



例题2 写等价的不同文法

设计一个文法：字母表 $\{a, b\}$ 上 a 和 b 的个数相等的所有串的集合

□ 二义文法： $S \rightarrow a S b S \mid b S a S \mid \varepsilon$
 $aabababab$ $aabbabab$

□ 二义文法： $S \rightarrow a B \mid b A \mid \varepsilon$
 $A \rightarrow a S \mid b A A$
 $B \rightarrow b S \mid a B B$
 $aabababab$ $aabbabab$ $aabbabab$

□ 非二义文法： $S \rightarrow a B S \mid b A S \mid \varepsilon$
 $A \rightarrow a \mid b A A$
 $B \rightarrow b \mid a B B$
 $a a b b a b a b$
 $a B S$



中国科学技术大学
University of Science and Technology of China

下期预告：自上而下的分析