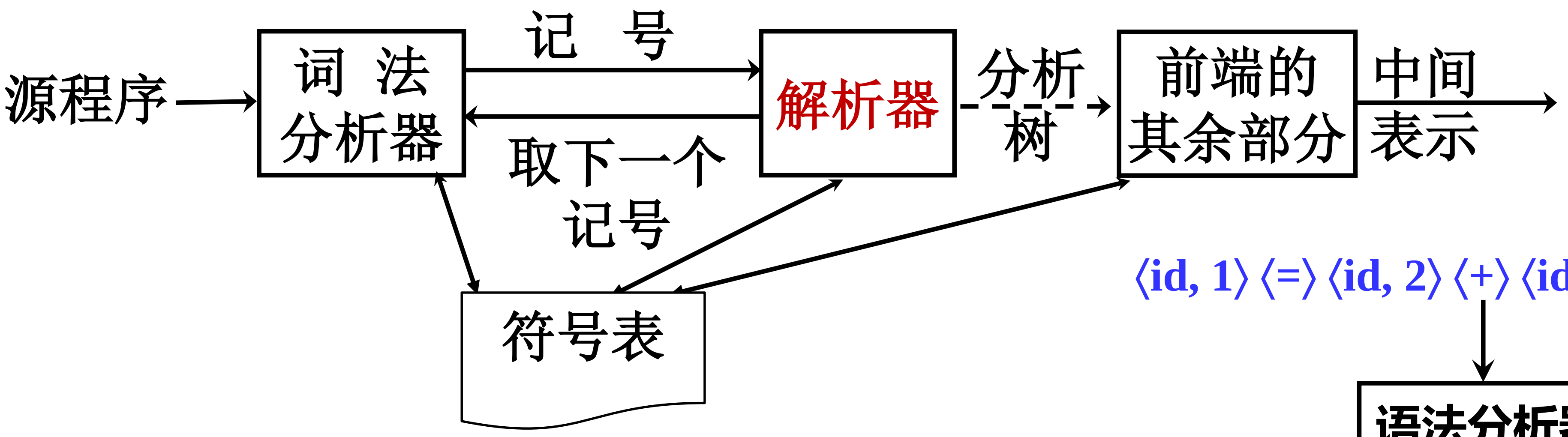
中国科学技术大学
University of Science and Technology of China

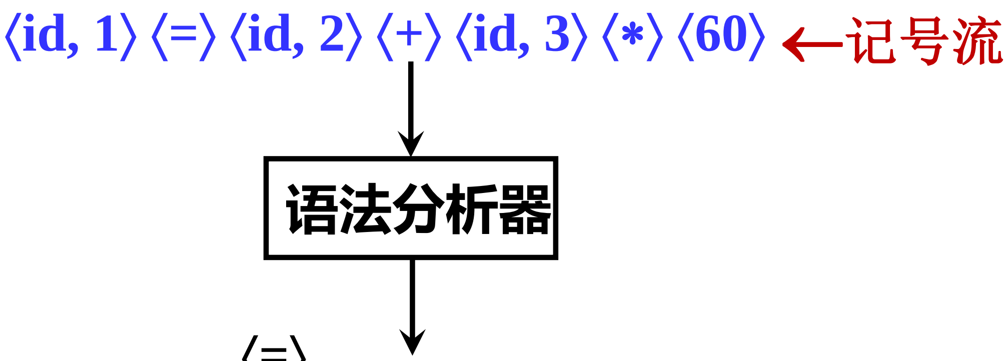
语法分析 I

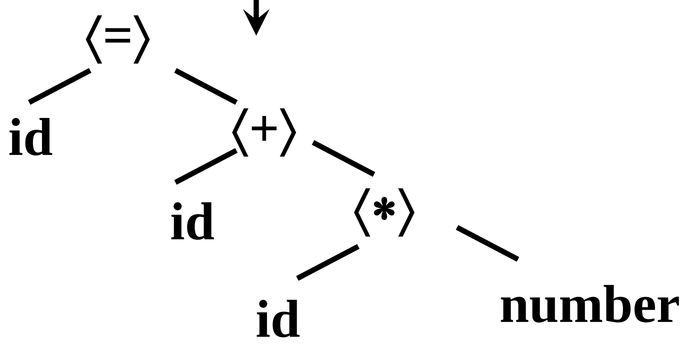
《编译原理和技术(H)》

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)、ALL(*)、SLR(k)、LR(k)、LALR(k)

3.1 上下文无关文法

- □ 正规式与上下文无关文法的比较
- □ 上下文无关文法的一些基本概念
  - ■ 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 $(^n)^n, n \geq 1$

    原因：n不固定，无法表示右括号的个数必须正好与前面左括号的个数一样

  - □ 例2：$\{wcw \mid w$是由a和b组成的串$\}$

    原因：w的长度不固定，c后面的串要依据c前面不定长的串w来确定；

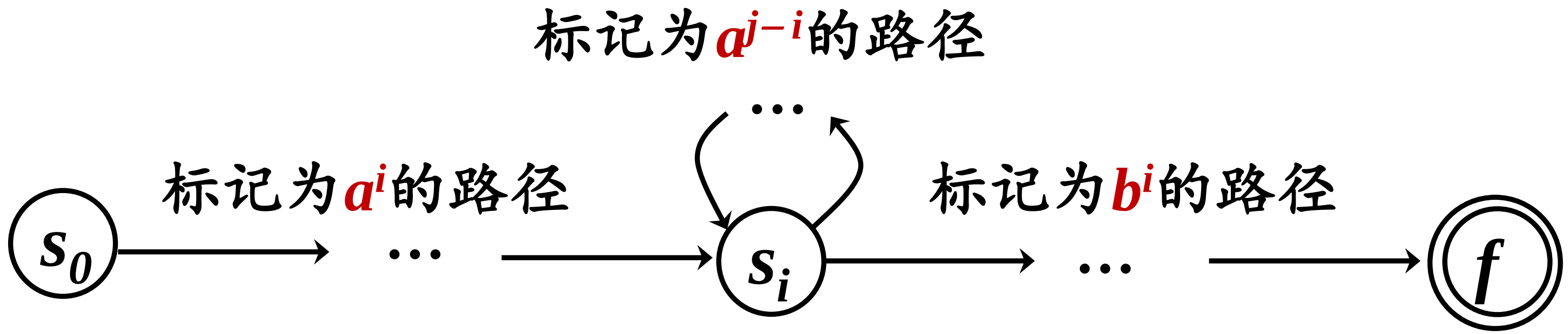
  有限自动机有有限个状态，无法记录访问同一状态的次数

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个（有限个）
- 设 D 读完ε, a, aa, …, a^k 分别到达状态s_0, s_1, …, 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，这与假设相矛盾



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 : $A \to \alpha$， $A \in V_N, \alpha \in (V_T \cup V_N)^*$

有时用 $A ::= \alpha$

- **例 ({id, +, *, –, (,)}, {*expr*, *op*}, *expr*, *P*)** 第一个产生式左部的符号就是文法开始符号

$expr \to expr\ op\ expr$ $expr \to (expr)$ $expr \to -\ expr$

$expr \to \mathrm{id}$ $op \to +$ $op \to *$

CFG的简化表示

表达式的CFG的简化表示

- 引入选择符 |

$$expr \rightarrow expr\ op\ expr \mid (expr) \mid -\ expr \mid \mathbf{id}$$

$$op \rightarrow + \mid *$$

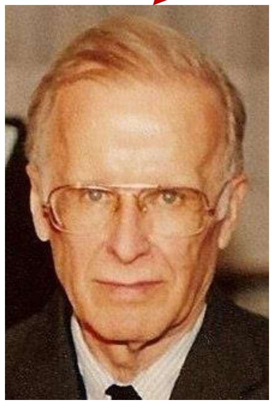
注：+, *是*op*的选择(alternatives)

Backus–Naur Form
巴科斯-诺尔，范式

- 简化名称

$$E \rightarrow E\ A\ E \mid (E\ ) \mid -E \mid \mathbf{id}$$

$$A \rightarrow + \mid *$$



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

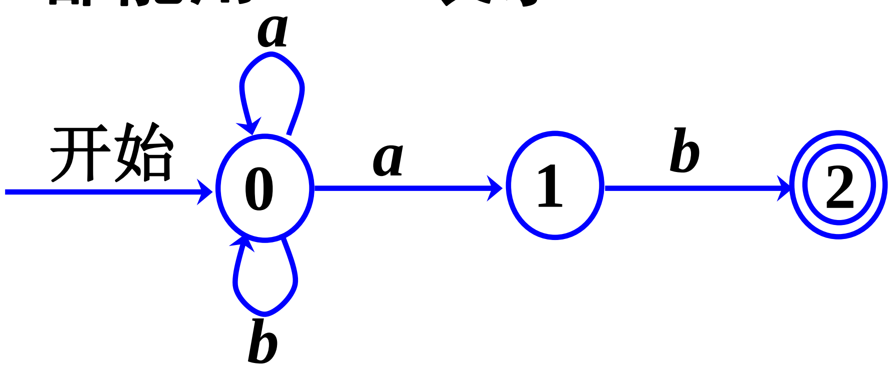
- John Backus (1924-2007)
  - **1977图灵奖获奖成就**
    - Fortran发明组组长
    - 提出了BNF
  - **履历**
    - 弗吉尼亚大学化学(因出勤率低而开除，随后入伍)
    - 哥伦比亚大学数学BS 1949, AM 1950
    - IBM
  - **获奖演说：** Can Programming Be Liberated From the von Neumann Style?

- 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，**NFA的字母表视为终结符集合**

    为每个NFA状态 i 引入一个非终结符A_i ，NFA中每条弧对应于产生式的一个**分支**（**选择**），

    对于接受状态 i，则引入$A_i \to \varepsilon$

    如果状态i有一个a转换到状态j，则引入产生式$A_i \to aA_j$；

    如果是ε转换，则引入$A_i \to A_j$。

    $A_0 \to a\ A_0 \mid b\ A_0 \mid a\ A_1$

    $A_1 \to b\ A_2$

    $A_2 \to \varepsilon$ (该产生式并不必要)

分离词法分析器的理由

- 正规文法是上下文无关文法的特例
- 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

3.1 上下文无关文法

- 正规式与上下文无关文法的比较
- 上下文无关文法的一些基本概念
  - 定义、推导、二义性
  - 名词：语言、文法等价、句型、句子

CFG：推导

推导(derivation)

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

$E \Rightarrow -E$ 读作 E 推导出 $-E$

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

上述代换序列称为从 E 到 -(id+id) 的推导　　-(id+id) 是E 的实例

记法

- 如果 $A \to \gamma$ 是产生式，α 和β 是文法的任意符号串，那么可以说$\alpha A\beta \Rightarrow \alpha\gamma\beta$
- 0步或多步推导 $S \Rightarrow^* \alpha$：1) $\forall\alpha.\ \alpha \Rightarrow^* \alpha$；2) 如果$\alpha \Rightarrow^* \beta$，$\beta \Rightarrow \gamma$，那么 $\alpha \Rightarrow^* \gamma$
- 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，不依赖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产生的语言称为上下文无关语言，它包含：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)： $S \Rightarrow^+ w$，w是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 $E \rightarrow E + E \mid E * E \mid (E) \mid -E \mid \mathbf{id}$

- 最左推导(leftmost derivation)

  每步代换句型中**最左边**的非终结符

  $$E \Rightarrow_{lm} -E \Rightarrow_{lm} -(E) \Rightarrow_{lm} -(E + E)$$

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

- 最右推导（rightmost or canonical，规范推导）

  每步代换句型中**最右边**的非终结符

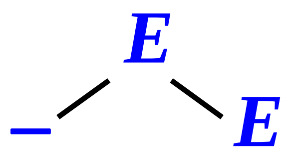
  $$E \Rightarrow_{rm} -E \Rightarrow_{rm} -(E) \Rightarrow_{rm} -(E + E)$$

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

分析树(parse tree)

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

-(id+id)最左推导的**分析树**（ parse tree ）

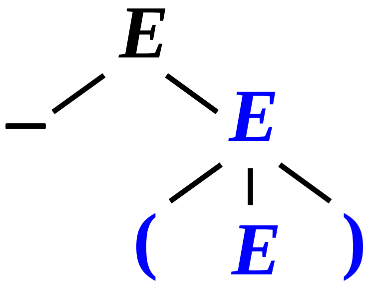


$E \Rightarrow -E$

分析树(parse tree)

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

-(id+id)最左推导的**分析树**（parse tree）

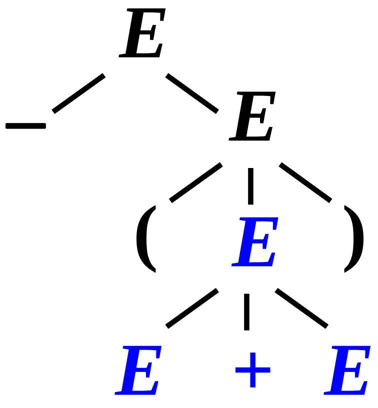


$E \Rightarrow -E \Rightarrow -(E)$

分析树(parse tree)

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

-(id+id)最左推导的**分析树**（parse tree）

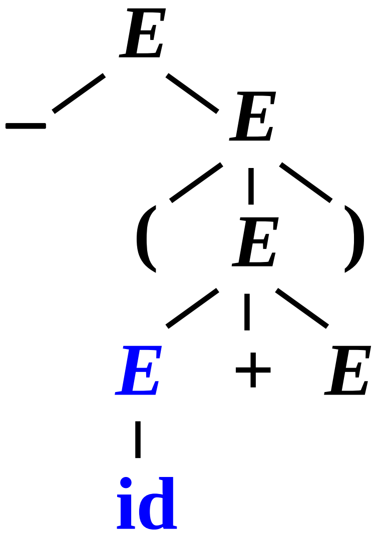


$E \Rightarrow -E \Rightarrow -(E) \Rightarrow -(E + E)$

分析树(parse tree)

例 $E \to E + E \mid E * E \mid (E) \mid -E \mid \textbf{id}$

-(id+id)最左推导的**分析树**（parse tree）

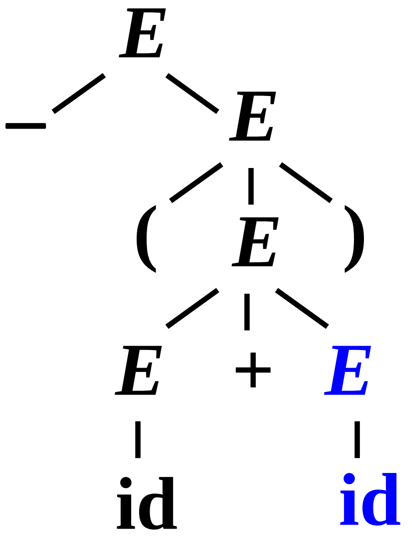


$E \Rightarrow -E \Rightarrow -(E) \Rightarrow -(E+E) \Rightarrow -(\mathrm{id}+E)$

分析树(parse tree)

例 $E \to E + E \mid E * E \mid (E) \mid -E \mid \mathbf{id}$

-(id+id)最左推导的分析树（parse tree）



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

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 $E \to E + E \mid E * E \mid (E) \mid -E \mid \mathrm{id}$

id*id+id 有两个不同的最左推导

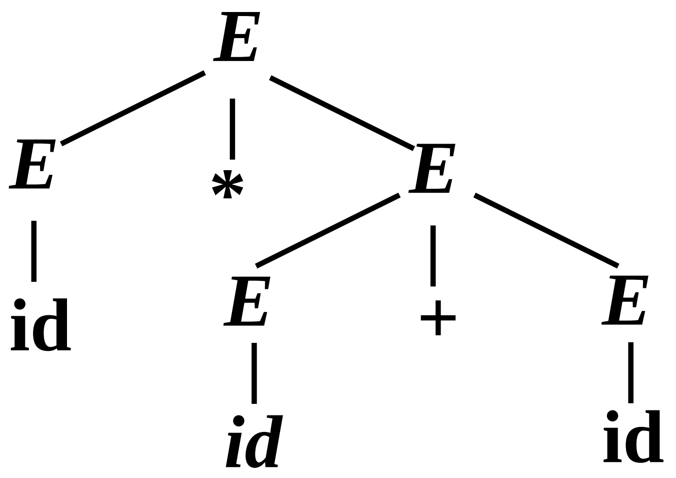
$$
\begin{aligned}
E &\Rightarrow E * E \\
&\Rightarrow \mathrm{id} * E \\
&\Rightarrow \mathrm{id} * E + E \\
&\Rightarrow \mathrm{id} * \mathrm{id} + E \\
&\Rightarrow \mathrm{id} * \mathrm{id} + \mathrm{id}
\end{aligned}
\qquad\qquad
\begin{aligned}
E &\Rightarrow E + E \\
&\Rightarrow E * E + E \\
&\Rightarrow \mathrm{id} * E + E \\
&\Rightarrow \mathrm{id} * \mathrm{id} + E \\
&\Rightarrow \mathrm{id} * \mathrm{id} + \mathrm{id}
\end{aligned}
$$

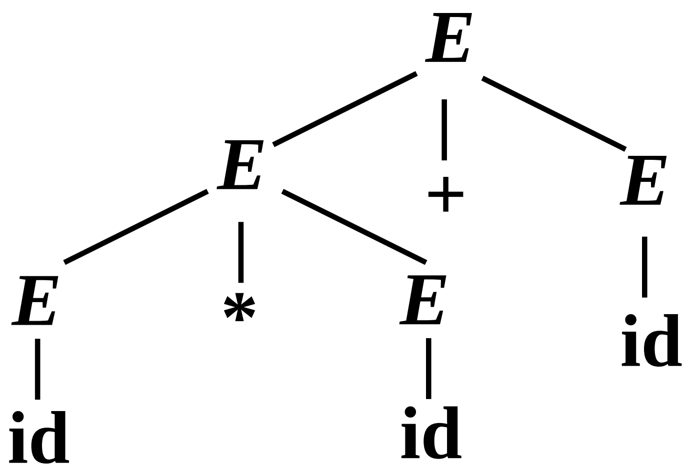
文法的二义性

id*id+id 有两棵不同的分析树

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

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





3.2 语言和文法

- □ 语言和文法：验证、消除二义性
- □ 非上下文无关文法

验证文法产生的语言

文法G : $S \to$ ‘(’S ‘)’ S | ε　　语言$L(G)$ = 配对的括号串的集合

① 从S 推出的是配对的括号串

② 任意配对的括号串可由S 推出

- 证明①：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

文法 $G: S \rightarrow$ '(' S ')' $S \mid \varepsilon$　　语言 $L(G)$ = 配对的括号串的集合

- 证明② ：任意长度的配对括号串均可由 S 推导出来

按串长进行归纳

- 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 ')' y

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

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

$expr \to$ $expr$ + $term$ | $term$

$term \to$ $term$ ∗ $factor$ | $factor$

$factor \to$ id | ($expr$)

左递归文法
∗ 是自左向右结合

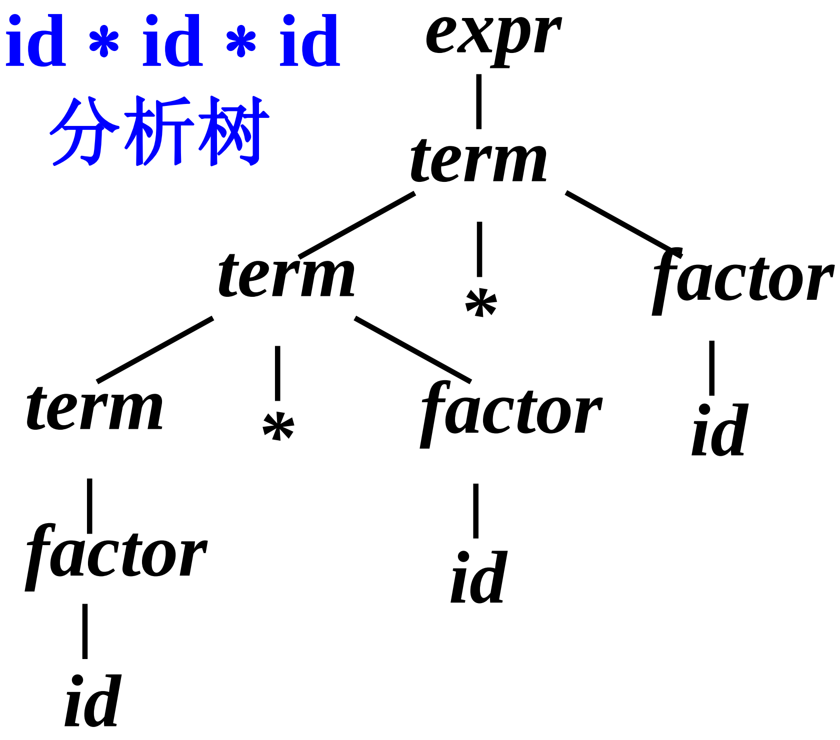
如果改成
$expr \to term + expr \mid term$
呢？
+ 是自右向左结合

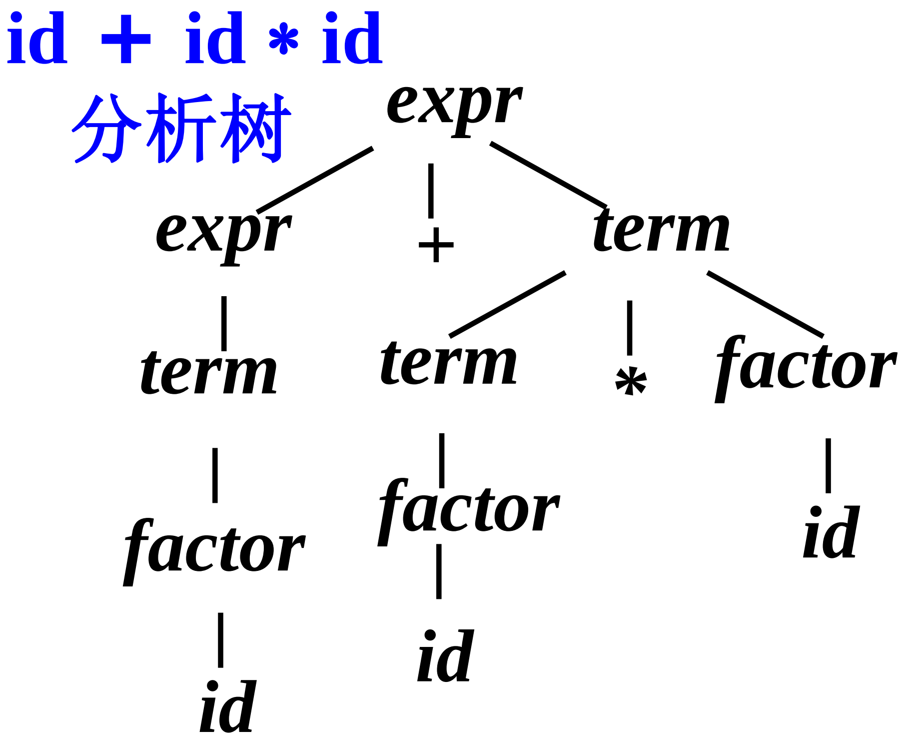
表达式的另一种文法

$$expr \rightarrow expr + term \mid term$$

$$term \rightarrow term * factor \mid factor$$

$$factor \rightarrow \mathbf{id} \mid (expr)$$





消除二义性(Eliminating ambiguity)

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

- 句型：**if** *expr* **then if** *expr* **then** *stmt* **else** *stmt*

有两个最左推导：

$$\begin{aligned} stmt &\Rightarrow \textbf{if } expr \textbf{ then } stmt \\ &\Rightarrow \textbf{if } expr \textbf{ then } \textbf{if } expr \textbf{ then } stmt \textbf{ else } stmt \end{aligned}$$

$$\begin{aligned} stmt &\Rightarrow \textbf{if } expr \textbf{ then } stmt \textbf{ else } stmt \\ &\Rightarrow \textbf{if } expr \textbf{ then } \textbf{if } expr \textbf{ then } stmt \textbf{ else } stmt \end{aligned}$$

消除二义性

无二义的文法

else 的就近匹配规则

$$stmt \rightarrow matched\_stmt$$
$$\quad | \ unmatched\_stmt$$

$$matched\_stmt \rightarrow \text{if } expr \text{ then } matched\_stmt \text{ else } matched\_stmt$$
$$\quad | \ \text{other}$$

$$unmatched\_stmt \rightarrow \text{if } expr \text{ then } stmt$$
$$\quad | \ \text{if } expr \text{ then } matched\_stmt \text{ else } unmatched\_stmt$$

3.2 语言和文法

- 语言和文法：验证、消除二义性
- 非上下文无关文法

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$A \rightarrow a$或$A \rightarrow \varepsilon$， $A, B \in V_N , a \in V_T$

  正规文法

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

非上下文无关的语言构造

$L_1 = \{wcw \mid w属于(a \mid b)^*\}$

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

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

$$S \to aSa \mid bSb \mid c$$

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

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

$$S \to aSd \mid aAd$$

$$A \to bAc \mid bc$$

$a^n b^n c^n$

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

$$S \to AB$$

$$A \to aAb \mid ab$$

$$B \to cBd \mid cd$$

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

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

$bB \to bb$　　$bC \to bc$　　$cC \to cc$

$a^n b^n c^n$的推导过程如下：

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

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

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

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

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

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

$S \Rightarrow^+ a^n b^n c^n$　　用$cC \to cc$　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

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

解答

非二义文法的产生式如下：

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

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

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

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

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

解答

非二义文法的产生式如下：

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

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

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

not p and q有两种不同的最左推导

$\text{not } \underline{\text{p and q}}$

$\underline{\text{not p}} \text{ and q}$

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： $S \to a\ S\ b\ S \mid b\ S\ a\ S \mid \varepsilon$

  aabbabab　　aabbabab

- 二义文法： $S \to a\ B \mid b\ A \mid \varepsilon$

  $A \to a\ S \mid b\ A\ A$

  $B \to b\ S \mid a\ B\ B$

  aabbabab　aabbabab　aabbabab

- 非二义文法： $S \to a\ B\ S \mid b\ A\ S \mid \varepsilon$

  $A \to a \mid b\ A\ A$

  $B \to b \mid a\ B\ B$

  a abb abab

  a B　S

下期预告：自上而下的分析