文法和语言

以描述和规定语言结构的称为“文法”（也称为“语法”）。
是对语言的语法结构的定义和描述。

符号、符号串及其集合

字母表（字符表）

是元素（字符）的非空有穷集合，由$\sum$表示输入的字符，例如：$\sum = \{a, b, c\}$

字母表中的元素称为符号, 依赖于字母表,符号只有在一定的字母表中才有意义。**字母表中符号的顺序是没有意义的。**

符号串（字）

由$\sum$集合中的符号构成的**有穷序列**称为在$\sum$上的符号串，也叫做字。**符号串中符号的顺序是有意义的。**

空串

不包含任何符号的串。用 ε 表示。

长度

符号串x中所包含的符号个数。用 $|x|$ 表示。

连接

a,b是两个符号串、则a,b的连接，ab就是将b的符号写在a的后面

$$x = ab, y = cd$$

$$xy = abcd, yx = cdab \qquad \text{e.g.1}$$

$$x\epsilon = \epsilon x = x = ab \qquad \text{e.g.2}$$

方幂

同一个符号串的若干次连接。

$$x^0 = \epsilon$$
$$x^1 = x$$
$$x^2 = xx$$
$$x^3 = xxx$$
$$\cdots$$
$$x^n = xxxx\cdots x(n\text{个})$$

例如：

$$x = ab$$
$$x^0 = \epsilon$$
$$x^1 = ab$$
$$x^2 = abab$$
$$x^3 = ababab$$
$$\cdots$$

符号串集合

$\sum^*$表示$\sum$（字母表）上所有字符串的全体，包括空字符串ε。

$$\sum = \{a, b\}$$

$$\sum^* = \{\epsilon, a, b, aa, ab, ba, \cdots\} \tag{e.g.1}$$

符号串集合乘积

两个定义在字母表上的符号串集合U,V，则

$$UV = \{XY | X \in U \& Y \in V\} \tag{defined}$$

例如：

$$\begin{aligned} U &= \{aa, bb\}, V = \{bb, cc\} \\ UV &= \{aabb, aacc, bbbb, bbcc\} \\ VU &= \{bbaa, bbbb, ccaa, ccbb\} \end{aligned} \tag{e.g.1}$$

集合的和（并）-- U+V或U∪V

两个定义在字母表上的符号串集合U,V，则

$$U \bigcup V = \{X | X \in U \text{或} X \in V\} \tag{defined}$$

例如：

$$\begin{aligned} U &= \{aa, bb\}, V = \{bb, cc\} \\ U \bigcup V &= \{aa, bb, cc\} \end{aligned} \tag{e.g.1}$$

$$U \bigcup V = \{aa, bb, cc\}$$

1. U与空集合求和：
   $\phi \bigcup V = V \bigcup \phi = V$
2. U与空集合求积：
   $\phi V = V \phi = \phi$
3. U与空字符串集合求积：
   $\{\epsilon\}V = \{\epsilon\}\phi = V$

符号串方幂

同一个符号串集合的若干次乘运算。

$$U^0 = \{\epsilon\}$$
$$U^1 = U$$
$$U^2 = UU$$
$$U^3 = UUU$$
$$\cdots$$
$$U^n = \prod_{i=1}^{n} U$$

例如：

$$A = \{a, b\}$$
$$A^0 = \{\epsilon\}$$
$$A^1 = \{a, b\}$$
$$A^2 = \{aa, ab, ba, bb\}$$
$$A^3 = \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$$
$$\cdots$$

空集合

不包含任何元素的集合称为空集合，用{ }或ϕ表示.

Warning

1. ε：空字符串；
2. {ε}：空字符集合，含一个ε元素。**A{ε}={ε}A=A，A是一个字符串集合。**
3. ϕ：不含任何元素的空集合。

正闭包（positive closure）

设V为任一集合， V 的正闭包V+为：

$$V^+ = V^1 \bigcup V^2 \bigcup V^3 \bigcup \cdots$$

例：

$$
\text{集合}V=\{a,b\}\text{的正闭包}:V^{+}=V^{1}\bigcup V^{2}\bigcup V^{3}\bigcup\cdots
$$
$$
=\{a,b,aa,ab,ba,bb,aaa,aab,\cdots\}
$$

星闭包（自反闭包或简称闭包）

V为任一集合，V*表示V的星闭包。

$$
V^{*}=V^{0}\bigcup V^{1}\bigcup V^{2}\bigcup V^{3}\bigcup\cdots
$$

即：

$$
V^{*}=V^{0}\bigcup V^{+}
$$
$$
V^{+}=VV^{*}=V^{*}V
$$

思考?

1. 已知V = { 1 }，求 V+，V*。

$$
V^{+}=\{1,11,111,1111,\cdots\}
$$
$$
V^{*}=\{\epsilon,1,11,111,1111,\cdots\}
$$

2. 已知V = { 0，1 }，求 V0 ，V1，V2。

$$
V^{0}=\{\epsilon\}
$$
$$
V^{1}=\{00,01,10,11\}
$$
$$
V^{2}=\{000,010,100,110,001,011,101,111\}
$$

3. 已知 ϕ，求 ϕ^{*}，ϕ^{+}。

$$
\phi^{+}=\phi
$$
$$
\phi^{*}=\{\epsilon\}
$$

文法和语言的形式定义

语言

语言：是某个字符表$\sum$上的符号串的集合。在一个特定的字符表上定义的，是按一定的规则构成的字符串的集合。

Key：字符表、语言的目标、规则

文法

对语言的结构和定义做形式化描述使用。描述一个语言的语法结构的形式化规则。编译程序的基本依据。

要求文法是准确的、容易理解的和便于分析的。
而且，应当有相当强的描述能力，足以描述各种不同的结构。

产生式（规则）

从“产生语言”出发，定义了语法单位的有限条书写规则，借助这些规则产生语言的全部句子。

符号： $\to$ 定义为产生出；| 定义为或；$\to$ 左边称为产生式的左部、右边称为右部

上下文无关文法

定义的语法单位完全独立于这种范畴可能出现的环境。

定义

文法G是一个四元组$G=(V_N,V_T,S,P)$，其中：

1. V_N:非终结符号的有穷集合。代表所定义的语法单位
2. V_T:终结符号的有穷集合。代表语言的基本符号
3. S：文法的开始符号，$S\in V_N$。代表语言的目标语法单位。
4. P：产生式的（有限）集合。定义语言的语法规则。

其中，非终结符的有穷集合与终结符的有穷集合不产生交集；对于产生式P,其左部一定属于非终结符，右部属于非终结符和终结符的并集。

例

写出下列表达式文法的VN,VT和S。

1. E→E+T | T
2. T→T*F | F
3. F→i | (E)

答

1. VN = ｛E, T, F}
2. VT = ｛+,*,i,(,)}
3. S = E

如果：语言中的所有句子（符号串）均可从文法中推导出来，由文法规则可推出语言的所有句子（符号串），则称：**该文法描述了这个语言**。

说明

给出一个语言，写出描述语言的文法是比较困难的。

同一个语言的文法不一定是唯一的。

要考虑两点：

1. 完备性：要包含语言中所有的句子（符号串）；

2. 最小性：不能含有不属于此语言的句子（符号串）。

推导和语法树

从文法G的开始符号出发，逐步用产生式的右部符号替换左部符号，直至推出给定的句子（输入串），这个过程称为推导。

推导符号：⇒（一步推导）

直接推导

假设x,y是符号串（V*），若用**一次产生式**可以从x推导出y，则称y为x的直接推导，记为：x ⇒y

间接推导

如果用若干次推导，可以从符号串x推出符号串y，则称y为x的推导。记为：$x^+ \Rightarrow y$

最左(右)推导

每次推导总是选择句型中最左（右）边的非终结符，用其产生式的右部来代替它，这样的推导称为最左（右）推导。

句型

设G是一个文法，S是文法的开始符号，α是符号串，如果S间接推导 α，则称α是此文法的一个**句型**。 所有推导过程中能够出现的所有结果都是句型。

句子

只包含终结符的句型。即**句子**

$$S^+ \Rightarrow \alpha, \alpha \in V_T^*, \text{则}\alpha\text{是句子}$$

由文法产生语言示例

语言

由文法G定义的语言记为L(G)，它是从文法的开始符号S推出的所有**句子的集合**。

$$L(G) = \{\alpha | S^+ \Rightarrow \alpha, \alpha \in V_T^*\}$$

文法等价

即需要证明 **文法定义的语言相同**。 因此 **同一种语言可由不同的文法定义**。

由语言构造文法示例

文法小结

1. 一个语言是某个特定字母表上的符号按一定规则构成的符号串（字符串）的集合。
    1. 字母表：规定了语言中所允许出现的符号（终结符）。
    2. 目标：语言定义的目标（文法开始符号）
    3. 规则：指明如何用字母表中的字符构成目标, 也称语法规则（文法中的产生式）
2. 文法的开始符号，至少应在产生式的左部出现一次。
3. 某一语言的文法代表了该语言的形式化定义:
    1. 由开始符号所推出的所有句子都是该语言的句子。
    2. 该语言的所有句子都可以由该文法推出。
4. 同一种语言可用不同的文法定义（文法不是唯一的）

文法的二义性

语法分析树（语法树）

用一张图表示一个句型的推导过程称为语法树（推导）。
表示形式：

1. 根结点：由文法开始符号标记。
2. 内部结点：由文法的非终结符号标记；
3. 叶子结点：由终结符号标记。

二义文法

如果一个文法G存在某个句子，该句子对应两棵不同的语法树，则称该文法是二义的。

弊端

语法结构的不确定，导致语义处理不确定。
同一源程序的目标代码不确定。

注意

文法是二义的并不一定意味着其所对应的语言也是二义的。
对于语言可进行人为规定。
比如，二义的表达式文法，在进行语句分析时可人为规定“*”的优先级大于“+”的优先级，这样就可使其语言无二义。

解决方法

1. 允许文法有二义性，但在分析过程中使用某些规则，在出现二义时根据这些规则来确定哪棵语法树是正确的（如规定算符的优先级）。
2. 直接修改文法，构造一个等价的无二义性的文法。
    1. 优先性技术
    2. 规定左结合或右结合

关于文法的几点限制

不能有有害的产生式（会带来二义性且是不必要的），如：
S→Pa
P→ P|b
对句子ba有无数种推导方法

不能有多余的产生式
多余产生式是指：
①在推导文法的所有句子中始终用不到的产生式；
②在推导过程中，一旦使用了此产生式，则将永远无法再推出任何终结符号串。

文法的分类

0型文法（短语文法，无约束文法）

文法G中含有产生式：

$$\alpha \to \beta, \alpha \in V^*V_NV^*，\beta \in V^*$$

即这种文法的特点是产生左部可以是一个符号串，但至少含有一个非终结符，产生式右部可以为空。
用图灵机识别。

1型文法（上下文有关文法）

文法G中含有产生式：

$$\alpha \to \beta, \alpha \in V^*V_NV^*，\beta \in V^*且|\alpha| < |\beta|(\alpha \to \epsilon除外)$$

用线性界线自动机识别。

2型文法（上下文无关文法）

$$A \to \beta, A \in V_N, \beta \in V^*$$

产生式的左部只能为单个非终结符。
用确定下推自动机识别。

3型文法（正规文法）

如果产生式只有下述两种形式:

$$
A \to \alpha B \\
A \to \alpha \\
A, B \in V_N, \alpha \in V_T^*
$$

右线性文法

$$
A \to B\alpha \\
A \to \alpha \\
A, B \in V_N, \alpha \in V_T^*
$$

左线性文法
用有限自动机（具有有穷描述的某种机器）识别。

作业

1.令文法G1为：
N → D|ND
D →0|1|2|… |9
（1）文法描述的语言L(G)是什么？
  L(G)是0-9构成的字符串集合
（2）给出句子568的最左推导和最右推导。
最左推导:

$$
N \Rightarrow ND \Rightarrow NDD \Rightarrow DDD \Rightarrow 5DD \Rightarrow 56D \Rightarrow 568
$$

最右推导:

$$
N \Rightarrow ND \Rightarrow N8 \Rightarrow ND8 \Rightarrow N68 \Rightarrow D68 \Rightarrow 568
$$

2.令文法G2为：
  E → T|E+T|E-T
  T→ F|T*F|T/F
  F →(E) |i
（1）给出该文法的VN、VT和S。

$$
V_N = \{E, T, F\} \\
V_T = \{*, /, +, -, (,), i\} \\
S = E
$$

（2）给出i+ii，i(i+i)的最左推导、最右推导和语法树。
最左推导 i+i*i:

$$
\begin{gathered}
E \Rightarrow E+T \Rightarrow T+T \Rightarrow F+T \Rightarrow \\
i+T \Rightarrow i+T*F \Rightarrow i+F*F \Rightarrow \\
i+i*F \Rightarrow i+i*i
\end{gathered}
$$

最右推导 i+i*i:

$$
\begin{gathered}
E \Rightarrow E+T*F \Rightarrow E+T*F \Rightarrow E+T*i \Rightarrow \\
E+F*i \Rightarrow E+i*i \Rightarrow T+i*i \Rightarrow \\
T+i*i \Rightarrow i+i*i
\end{gathered}
$$

最左推导 i*(i+i)：

$$
\begin{gathered}
E \Rightarrow T \Rightarrow T*F \Rightarrow F*F \Rightarrow \\
i*F \Rightarrow i*(E) \Rightarrow i*(E+T) \Rightarrow \\
i*(T+T) \Rightarrow i*(F+T) \Rightarrow i*(i+T) \Rightarrow i*(i+i)
\end{gathered}
$$

最右推导 i*(i+i)：

$$
\begin{gathered}
E \Rightarrow T \Rightarrow T*F \Rightarrow T*(E) \Rightarrow T*(E+T) \Rightarrow T*(E+F) \\
\Rightarrow T*(E+i) \Rightarrow T*(F+i) \Rightarrow T*(i+i) \\
\Rightarrow F*(i+i) \Rightarrow i*(i+i)
\end{gathered}
$$

3.证明下面的文法是二义的：
S → iSeS | iS | i
证明:
最右推导 iiiei

$$
\begin{gathered}
S \Rightarrow iSeS \Rightarrow iSei \Rightarrow iiSei \Rightarrow iiiei \\
S \Rightarrow iSeS \Rightarrow iiSeS \Rightarrow iiSei \Rightarrow iiiei
\end{gathered}
$$

故存在两个最右推导能推导出语句，故语法为二义的。