第二章 上下文无关文法和语言

§2.1 文法和语言的表示

§2.2 文法和语言的定义

§2.3 句型的分析

§2.4 文法的实用限制和其他表示法

§2.5 文法和语言的Chomsky分类

提要

- 所谓形式化方法，简单地说，就是用一整套带有严格规定的符号体系来描述问题的理论和方法，用形式化方法描述的语言（语法和语义）便是形式语言。
- 本章将从形式语言的角度系统地介绍什么是程序设计语言的文法，文法和语言的关系等问题，本章是本课程的理论基础。

§2·1 文法和语言的表示

语言的定义可采用下列三种方法

1. **枚举法**——把该语言的所有句子列出放在一集合内。（有限个句子时）
2. **有限条规则**——描述语言的全部句子。（有限或无限个句子），即文法表示
3. **装置**——检验和识别句子。（有限或无限个句子）即自动机

§2·2 文法和语言的定义

§2·2·1、 基本概念和术语

1、字母表： 元素的非空有穷集合

元素称符号

例:$\sum=\{0,1\}$

2、符号串： 字母表中的符号所组成的任何有穷序列

特别： 空符号串 ε （不包含任何符号）

3、字母表∑上的符号串的递归定义

(1) ε是∑上的符号串

(2) 若x是∑上的符号串，且a∈∑，

则xa或ax是∑上的符号串

特别：εx = xε= x

(3) 若y是∑上的符号串，

当且仅当y可由（1）和（2）产生。

（1） ε是 Σ上的符号串

（2）若x是 Σ上的符号串，且a ∈ Σ，
则xa或ax是Σ 上的符号串
特别： ε · x = x ε = x

（3）若y是Σ 上的符号串，
当且仅当y可由（1）和（2）产生。

例： ∑ ={b， c} ，
求∑上的 所有符号串

根据1 ε是 ∑上的符号串

根据2 εb和εc 即 b, c 是∑上的符号串

bb， bc， cb， cc是∑上的符号串

bbb， bbc ， bcb ， bcc ， cbb ， cbc ， ccb ， ccc是∑上的符号串.....

∑上的 所有符号串 ε ， b ， c, bb， bc， cb， cc, bbb，
bbc ， bcb ， bcc ， cbb ， cbc ， ccb ， ccc.....

5、符号串的前缀、后缀和子串：

前缀：设x是一符号串，从x的尾部删去若干个（包括0个）符号之后所剩余下的部分称为x的**前缀**；

若x的前缀不是x本身，则称为x的**真前缀**。

后缀：设x是一符号串，从x的头部删去若干个（包括0个）符号之后所剩余下的部分称为x的**后缀**；

若x的后缀不是x本身，则称为x的**真后缀**。

子串：从一个符号串中删去它的一个前缀和一个后缀之后所剩下的部分称为此符号串的**子串**。

若x的子串不是x本身，则称为x的**真子串**

例 设x=abc

x= ε a ε b ε c ε

x的前缀： abc ab a ε

（ε, a, ab为真前缀）

x的后缀： abc bc c ε

（ε, c, bc为真后缀）

x的子串： abc ab bc a b c ε

(ε, a, ab, b, c, bc为真子串)

用法： z=x……（x为z的前缀）

z=……x （x为z的后缀） 感兴趣部分

z=…x….（x 为z的子串）

5、符号串的长度：符号串所含符号的个数

6 、符号串的连接和方幂

连接：设有符号串x,y，把y的符号写在x的符号之后所得的符号串，叫做x与y的连接，记xy

方幂：设有符号串x，则x的n次自身连接称为x的n次方幂，记为x^n

特别:$x^0 = \varepsilon$

7、符号串集合A与B的和与积：

和： $A+B=\{w|w \in A或w \in B\}$

积： $AB=\{xy|x \in A且y \in B\}$

8、符号串集合的方幂：

设有符号串集合A

则定义 $A^0 = \{\varepsilon\}$

$A^1=A$

$A^2=AA$

$A^n=\underbrace{AA\ldots\ldots A}_{n个}$

9、 符号串(符号）集合的正闭包

设A为符号集合，则定义A的正闭包A^+为：

$A^+ = A \cup A^2 \cup \ldots \cup A^n \cup \ldots\ldots$

例：$\sum$={b，c}，

求$\sum$上的所有符号串

ε，b，c, bb，bc，cb，cc, bbb，bbc，bcb，bcc，cbb，bcb，ccb，ccc……

则 A^+={b , c} ∪{bb,bc,cb,cc} ∪ ……

= {b，c, bb，bc，cb，cc, bbb，bbc，bcb，bcc，cbb，bcb，ccb，ccc……}

➢A^+是由A中的元素构成的符号串(除ε)的集合.

10、符号串（符号）集合的闭包A^*

设A为符号集合，则定义A的闭包A^*为：

$A^*=A^0 \cup A^+= \{\varepsilon\} \cup A^+$

则$A^*=\{\varepsilon\}\cup\{b,c\}\cup\{bb,bc,cb,cc\}\cup$ ……

= {ε , b , c, bb, bc, cb, cc, bbb, bbc , bcb , bcc , cbb , bcb , ccb , ccc……}

➢A^*由A中的元素构成的所有符号串的集合.

问　题

- 为什么对符号、符号串、符号串集合以及它们的运算感兴趣？

若V为某语言的字母表：V={a, b,…0, 1,…9, +, =, (,)…if, else, for…}

X为单词集：X={if, else, for,…, <标识符>, <常量>, …}, 则 单词 x∈ V^*

句子是定义在单词集X上的符号串, 则句子a ∈X^*

语言是由句子组成的集合，是字母表V上的符号串集合

- 例如：字母表$\sum$ ={a, b}

  $\sum^*$ ={ε , a, b, aa, ab, ba, bb, aaa, aab, …}

那么，集合{ab, aabb, aaabbb, …, a^nb^n ,}或{w|w ∈ V*且w= a^nb^n, n>=1}

为字母表$\sum$上的一个语言

§2·2·2、文法和语言的形式定义

1、文法的形式定义

问题：如何定义句子的合法性？

有穷语言：逐一列举句子

无穷语言：文法是以有穷的集合刻画无穷集合的工具

文法是对语言结构的定义和描述（或称“语法”）

注意：从一组规则可以推导出不同的句子，文法是在形式上对句子结构的定义和描述，而未涉及语义问题

§2·2·2、文法和语言的形式定义

1、文法的形式定义

(1) 规则（产生式）

一有序对（U，x） 记为U::=x或U→x

规则的左部：U是符号（有的文法也可为符号串）

规则的右部：x是有穷符号串

表示：U定义为x

例：S → abc

<主函数> →main（<参数表>）<参数说明><函数体>

(2) 文法G[Z]: 规则的非空有穷集合.

Z: 开始符号（识别符号），至少在一条规则的左部出现。

(3) 字汇表V: 规则左右部中所有符号组成的集合

非终结符号: 需要定义的语法范畴
组成非终结符号集合V_n

终结符号: 规则中不属于V_n的符号
组成终结符号集合V_t

- 非终结符号以< >括起,但当是大写字母是时常省略
- $V=V_n \cup V_t$

(4) 文法的四元组表示

$G=(V_n, V_t, P, Z)$

其中 V_n：非终结符号集

V_t：终结符号集

P：规则的集合

Z：文法的开始符号

- 文法的BNF（巴科斯范式）表示

  规则中有相同的左部时：$V \rightarrow x$

  $V \rightarrow y$

  …

  $V \rightarrow z$

  写成：$V \rightarrow x|y|\ldots|z$　（’|’表达‘或’）
- 元符号：→(，::=，）， | ， < , >
- 元语言: 元符号处理的语言
- 由元符号组成的巴科斯范式(元语言公式)是用以描述算法语言的元语言

例:$G=(V_n,V_t,P,S)$

其中:

$V_n=\{S,A,B\}$

$V_t=\{a,b\}$

S:文法的开始符号

P: $S \rightarrow aB \mid bA$

$A \rightarrow a \mid aS \mid bAA$

$B \rightarrow b \mid bS \mid aBB$

文法BNF表示为

G[S]:

$S \rightarrow aB \mid bA$

$A \rightarrow a \mid aS \mid bAA$

$B \rightarrow b \mid bS \mid aBB$

- 一般规定：第一条规则的左部为开始符号，那么文法有规则的集合就完全确定了。

2、推导的形式定义

(1) **直接推导**：如果$U \to u$是G中的一条规则，

$x, y \in V^*$，

则将规则$U \to u$用于符号串$r=xUy$上

得到符号串$w=xuy$

记为：$xUy \Rightarrow xuy \ (r \Rightarrow w)$

称符号串w是符号串r的**直接推导**,或符号串r直接产生了符号串w,也称w**直接归约**到r.

文法BNF表示为

G[S]:

S → aB | bA

A →a | aS | bAA

B →b | bS | aBB

例:上述文法G[S]可进行的直接推导

① S=>aB　　(规则S → aB)

U =>u

(规则U → u , x,y均为ε)

② abS=>abbA　(规则S →bA)

xU =>xu　　(规则U → u , x为ab,y为ε)

③ aB=>aaBB　　(规则B →aBB)

xU =>xu　　(规则U → u , x为aaB,y为ε)

文法BNF表示为

G[S]:

S → aB | bA

A →a | aS | bAA

B →b | bS | aBB

例:上述文法G[S]可进行的推导

推导过程	使用规则
S=>aB	S → aB
=>abS	B →bS
=>abbA	S → bA
=>abbbAA	A → bAA
=>abbbaA	A →a
=>abbbaa	A →a

- 只要符号串中存在非终结符号, 推导就能继续, 直至符号串全由终结符号组成, 这也是为什么称终结符和非终结符的原因

(2)推导(长度为n):

设$u_0,u_1,\ldots u_n(n>0)$均$\in V^*$,且有

$r=u_0=>u_1=>\ldots\ldots=>u_{n-1}=>u_n=w$

记为$r =^+>w$ (一步或一步以上)

则称以上序列为长度n的推导,

也称r产生w(w归约为r)

特例: r=w(0步推导)

- 长度n>=0的推导记为 $r=^*>w$; 长度n>0的推导记为 $r=^+>w$

3、语言的形式定义

(1)句型:设有文法G[Z],如果有$Z=^{*}>x$, $x \in V^{*}$,

则称x是文法G的一个句型.

- 凡是由识别符号推导出来的字汇表V上的终结和非终结符号组成的符号串叫做句型。

(2)句子:如有$Z=^{+}>x$, $x \in V_t^{*}$,

则称x是文法G的一个句子.

- 由Z推导的终结符号组成的符号串为句子。

(3)语言L(G[Z]):文法G[Z]产生的所有句子的集合，称文法G[Z]所定义的语言

$$L(G[Z])=\{x|x\in V_t^* \text{且} Z=^+>x\}$$

例:G[<标识符>]

<标识符>→<字母>|<标识符><字母>

|<标识符><数字>

<字母>→a|b|…|z|A|…|Z

<数字>→0|1|2|…|9

问题:符号串“a4y”是不是文法的句子?

结论：

a4y是文法的合法句子

例：G[〈标识符〉]

〈标识符〉→〈字母〉|〈标识符〉〈字母〉

|〈标识符〉〈数字〉

〈字母〉→a|b|...|z|A|...|Z

〈数字〉→0|1|2|...|9

推导过程1

〈标识符〉

⇒〈标识符〉〈字母〉

⇒〈标识符〉y

⇒〈标识符〉〈数字〉y

⇒〈标识符〉4y

⇒〈字母〉4y

⇒a4y

推导过程2

〈标识符〉

⇒〈标识符〉〈字母〉

=>〈标识符〉〈数字〉〈字母〉

⇒〈字母〉〈数字〉〈字母〉

⇒a〈数字〉〈字母〉

⇒a4〈字母〉

⇒a4y

例:G1[A]:A→Bb

B→a　　L(G1)={ab}

G2[A]:A→ab　　L(G2)={ab}

G1≠G2但L(G1)=L(G2)

称G1和G2为等价文法（不同文法，相同语言）.

- 给定文法后，可以确定它的语言，但由语言写出的文法是比较难的，这里形式语言理论可以证明：

1° 给定一文法,就能从结构上唯一确定其语言.

即 G 唯一确定 L(G)

2° 给定一语言,能确定其文法,但这种文法不是唯一的。 即L 确定 G1,或G2…

3° 设$G=(V_n, V_t, P, S)$为一文法,

并设$U\rightarrow xVy$,是P中一产生式,

且$V\rightarrow\beta_1|\beta_2|\beta_3\ldots\ldots|\beta n$ 是P中V的全部产生式

又设 $G1=(V_n,V_t,P_1,S)$ 是其中 P_1 从 P 中删去 $U\rightarrow xVy$,

加入$U\rightarrow x\beta_1 y$, $U\rightarrow x\beta_2 y$,…, $U\rightarrow x\beta n y$,

则L(G1)=L(G)

观察公式$V \rightarrow \beta1|\beta2|\beta3\ldots\ldots|\beta n$两侧

G3[S]:S → A | S-A

A → a | b | c

G4[S]:S → A | A-S

A → a | b | c

符号串a-b-c是G3[S]、G4[S]合法句子，但语义不同。

G3解释为（a-b）-c

G4解释为 a-（b-c）

§2·2·3 递归规则与递归文法

-------用有穷的规则刻划无穷的语言

1、递归规则

形如$U \rightarrow xUy$　　$U \in Vn$，$x,y \in V^*$

左右具有相同的非终结符号的规则

特别:$U \rightarrow Uy$　　$(x=\varepsilon)$左递归规则

$U \rightarrow xU$　　$(y=\varepsilon)$右递归规则

$U \rightarrow xUy$　　$(x,y \neq \varepsilon)$自嵌入递归规则

- 递归规则是对其左部的非终结符号进行递归定义

2.文法的递归性

1)直接递归性:文法中至少包含一条递归规则

2)间接递归性:文法的任一非终结符号经一步以上推导产生的递归性。

3)文法的递归性原则:文法具有直接递归性或间接递归性,否则,文法无递归性。

例1: G[Z]: Z→aZb|ab　　具有直接递归性

例2: G[U]: U →Vx

V →Uy |z　　具有间接递归性

原因: U=>Vx=>Uyx

- 例： G[S]: S →0S1, S →01

  L(G)={ 0^n1^n | n>=1}

- 例： G[S]: (1) S →aSBE

  (2) S → aBE

  (3) EB → BE

  (4) aB → ab

  (5) bB → bb

  (6) bE → be

  (7) eE → ee

  L(G)={ $a^nb^ne^n$ | n>=1}

(1) S →aSBE
(2) S → aBE
(3) EB → BE
(4) aB → ab
(5) bB → bb
(6) bE → be
(7) eE → ee

- S => aSBE (S →aSBE)
  => aaBEBE (S → aBE)
  => aabEBE (aB → ab)
  => aabBEE (EB → BE)
  => aabbEE (bB → bb)
  => aabbeE (bE → be)
  => aabbee (eE → ee)

$S \Rightarrow a^n b^n e^n \quad (n>=1)$?

复习思考题、作业题：

39页， 2-2 (1), (2)

2-3 (1), (2)

§2.3　句型的分析

§2.3.1 规范推导和归约

1、**最左(右)推导**:在任一步推导V=>w中，都是对符号串V的最左(右）非终结符号进行替换，称最左（右）推导。

2、**规范推导**:即最右推导

3、**规范句型**：由规范推导所得的句型.

4、**规范归约**：规范推导的逆过程，称规范归约或最左归约。

例:G[<标识符>]

<标识符>→<字母>|<标识符><字母>

|<标识符><数字>

<字母>→a|b|…|z|A|…|Z

<数字>→0|1|2|…|9

问题:给出句子a4y的规范推导和规范归约.

给出句子a4y的最左推导.

请注意：规范推导和规范归约互为逆过程。

规范推导

〈标识符〉

⇒〈标识符〉〈字母〉

⇒〈标识符〉y

⇒〈标识符〉〈数字〉y

⇒〈标识符〉4y

⇒〈字母〉4y

⇒a4y

问题：如何正确选择规则？

规范归约

a4y

〈≠〈字母〉4y

〈≠〈标识符〉4y

〈≠〈标识符〉〈数字〉y

〈≠〈标识符〉y

〈≠〈标识符〉〈字母〉

〈≠〈标识符〉

问题：如何准确选择可归约串？