



中山大學
SUN YAT-SEN UNIVERSITY



国家超级计算广州中心
NATIONAL SUPERCOMPUTER CENTER IN GUANGZHOU

DCS²⁹⁰

Compilation Principle 编译原理

第二章：语言 and 文法基础

郑馥丹

zhengfd5@mail.sysu.edu.cn

CONTENTS

目录

01

语言和文法的
直观概念

02

符号和
符号串

03

文法和语言的
形式定义

04

文法的
类型

05

上下文无关文
法及其语法树

06

句型的
分析

07

有关文法实用
中的一些说明

1. 程序设计语言

- 程序设计语言包括：语法和语义
 - 语法(syntax)：是一组**规则**，用它可以形成和产生一个合适的程序
 - 语义(semantics)：定义程序的**意义**
 - ✓ 静态语义：程序在语义上要遵守的规则
 - **数组下标越界**
 - **声明和使用的函数没有定义**
 - **零作除数**
 -
 - ✓ 动态语义：表明程序要做什么

2. 文法[Grammar]的直观概念

- 如何来描述一种语言？

- 如果语言是有穷的（只含有有穷多个句子）：可以将句子逐一列出来表示
- 如果语言是无穷的，要找出语言的有穷表示

- **文法[Grammar]**:

- ✓ 是语言**语法**的描述工具，实现**用有穷的规则把语言的无穷句子集描述出来**
- ✓ 严格定义句子的结构，是判断句子结构合法与否的依据
- 例：“**我是大学生**”是汉语的一个句子

汉语句子的部分构成规则可表示为：

- $\langle \text{句子} \rangle ::= \langle \text{主语} \rangle \langle \text{谓语} \rangle$
- $\langle \text{主语} \rangle ::= \langle \text{代词} \rangle \mid \langle \text{名词} \rangle$
- $\langle \text{代词} \rangle ::= \text{我} \mid \text{你} \mid \text{他}$
- $\langle \text{名词} \rangle ::= \text{王明} \mid \text{大学生} \mid \text{工人} \mid \text{英语}$
- $\langle \text{谓语} \rangle ::= \langle \text{动词} \rangle \langle \text{直接宾语} \rangle$
- $\langle \text{动词} \rangle ::= \text{是} \mid \text{学习}$
- $\langle \text{直接宾语} \rangle ::= \langle \text{代词} \rangle \mid \langle \text{名词} \rangle$

2. 文法[Grammar]的直观概念

• 由规则推导句子

– 方法: 用一条规则, 用 **$::=$ 右端的符号串代替 $::=$ 的左端**

✓ **$\langle \text{句子} \rangle ::= \langle \text{主语} \rangle \langle \text{谓语} \rangle$**

– 表示: 用 “ \Rightarrow ” 表示推导, 其含义是, 使用一条规则, 代替掉 \Rightarrow 左边的某个符号, 产生 \Rightarrow 右端的符号串

– 例如: 句子“我是大学生”的推导过程如下:

$\langle \text{句子} \rangle$

$\Rightarrow \langle \text{主语} \rangle \langle \text{谓语} \rangle$

$\Rightarrow \langle \text{代词} \rangle \langle \text{谓语} \rangle$

$\Rightarrow \text{我} \langle \text{谓语} \rangle$

$\Rightarrow \text{我} \langle \text{动词} \rangle \langle \text{直接宾语} \rangle$

$\Rightarrow \text{我是} \langle \text{直接宾语} \rangle$

$\Rightarrow \text{我是} \langle \text{名词} \rangle$

$\Rightarrow \text{我是大学生}$

汉语句子的部分构成规则可表示为:

- $\langle \text{句子} \rangle ::= \langle \text{主语} \rangle \langle \text{谓语} \rangle$
- $\langle \text{主语} \rangle ::= \langle \text{代词} \rangle \mid \langle \text{名词} \rangle$
- $\langle \text{代词} \rangle ::= \text{我} \mid \text{你} \mid \text{他}$
- $\langle \text{名词} \rangle ::= \text{王明} \mid \text{大学生} \mid \text{工人} \mid \text{英语}$
- $\langle \text{谓语} \rangle ::= \langle \text{动词} \rangle \langle \text{直接宾语} \rangle$
- $\langle \text{动词} \rangle ::= \text{是} \mid \text{学习}$
- $\langle \text{直接宾语} \rangle ::= \langle \text{代词} \rangle \mid \langle \text{名词} \rangle$

CONTENTS

目录

01

语言和文法
的直观概念

02

符号和
符号串

03

文法和语言
的形式定义

04

文法的
类型

05

上下文无关文
法及其语法树

06

句型的
分析

07

有关文法实用
中的一些说明

1. 字母表[Alphabet] (符号集[a set of symbols])

- 定义：字母表是元素的非空有穷集合

例： $\Sigma=\{0,1\}$ $A=\{a,b,c\}$

- 元素也称为符号，字母表也称符号集
- 不同的语言有不同的字母表
- 程序语言的字母表由**字母、数字和若干专用符号**组成

2. 符号串[String]

- 定义：符号串是由字母表中的符号组成的任何有穷序列

例：0,00,10,011是字母表 $\Sigma=\{0,1\}$ 上的符号串

a,ab,aaca是 $A=\{a,b,c\}$ 上的符号串

- 在符号串中，符号是有顺序的，顺序不同，代表不同的符号串 例: ab和ba不同
- 不含任何符号的符号串称为空串，用 ϵ 表示

注意： $\{\epsilon\}$ 并不等于空集合 $\{\}$ —— Φ

- 符号串长度：是符号串中含有符号的个数

例： $|abc|=3$ $|\epsilon|=0$

- 符号串的头尾，固有头和固有尾：如果 $z=xy$ 是一符号串，则 x 是 z 的头， y 是 z 的尾。如果 x 是非空的，则 y 是固有尾；如果 y 非空，则 x 是固有头。

例： $z=abc$ ， z 的头： ϵ ， a ， ab ， abc ，除 abc 外，其他都是固有头； z 的尾： ϵ ， c ， bc ， abc ，除 abc 外，其他的都是固有尾。

头、尾 \leftrightarrow 前缀、后缀

固有头、固有尾 \leftrightarrow 真前缀、真后缀

3. 符号串的运算

- **符号串的连接**: 设 x 、 y 是符号串, 它们的连接是把 y 的符号写在 x 的符号之后得到的符号串 xy

例如 $x="ST"$, $y="abu"$, 则 $xy="STabu"$

显然 $\varepsilon x = x\varepsilon = x$

- **符号串的方幂**: 把符号串 a 自身连接 n 次得到的符号串 $a^n = aa...aa$

例1 $a^0=\varepsilon$, $a^1=a$, $a^2=aa$

例2 $x=AB$, $x^0 = \varepsilon$, $x^1 = AB$, $x^2 = ABAB...$

4. 符号串集合

- 定义：若集合A中所有元素都是某字母表 Σ 上的符号串，则称A为字母表 Σ 上的符号串集合。
- 符号串集合的乘积：符号串集合A和B的乘积定义为：
 - $AB = \{xy \mid x \in A \text{ 且 } y \in B\}$ ，即AB是由A中的串x和B中的串y连接而成的所有串xy组成的集合。

例：若集合 $A = \{ab, cde\}$ $B = \{0, 1\}$

则 $AB = \{ab0, ab1, cde0, cde1\}$

显然 $\{\varepsilon\}A = A\{\varepsilon\} = A$

4. 符号串集合

- 符号串集合的方幂： 设 A 是符号串的集合，则称 A^i 为符号串集 A 的方幂，其中 i 是非负整数。具体定义如下：

$$- A^0 = \{\varepsilon\}, A^1 = A, A^2 = AA, A^k = AA \dots AA (k \text{ 个})$$

例：

若集合 $A = \{a, b\}$ 则 $A^0 = \{\varepsilon\}$ $A^1 = A = \{a, b\}$

$A^2 = AA = \{a, b\}\{a, b\} = \{aa, ab, ba, bb\}$

$A^3 = AAA = \{a, b\}\{a, b\}\{a, b\} = \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$

5. 集合的闭包[Closure]

- 闭包[Kleene Closure]

- 集合 Σ 的闭包 Σ^* 定义如下: $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \dots$

例: 设有字母表 $\Sigma=\{0, 1\}$

则 $\Sigma^*=\Sigma^0\cup\Sigma^1\cup\Sigma^2\cup\dots$

$=\{\epsilon, 0, 1, 00, 01, 10, 11, 000, \dots\}$

即 Σ^* 表示 Σ 上所有有穷长的串的集合。

- 正闭包[Positive Closure]

- $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \dots$ 称为 Σ 的正闭包。

- Σ^+ 表示 Σ 上的除 ϵ 外的所有有穷长串的集合

$$\Sigma^* = \Sigma^0 \cup \Sigma^+$$

$$\Sigma^+ = \Sigma \Sigma^* = \Sigma^* \Sigma$$

6. 语言[Language]

- 例如: $\Sigma=\{a,b\}$ $\Sigma^*=\{\epsilon,a,b,aa,ab,ba,bb,aaa,aab,\dots\}$
 1. 集合 $\{ab,aabb,aaabbb,\dots,a^n b^n,\dots\}$ 或 $\{w \mid w \in \Sigma^* \text{ 且 } w=a^n b^n, n \geq 1\}$ 为字母表 Σ 上的一个语言。
 2. 集合 $\{a,aa,aaa,\dots\}$ 或 $\{w \mid w \in \Sigma^* \text{ 且 } w=a^n, n \geq 1\}$ 为字母表 Σ 上的一个语言。
 3. $\{\epsilon\}$ 是一个语言。
 4. Φ 即 $\{\}$ 是一个语言。
 5.

Σ^* 上任意字符串的集合
均是该字母表 Σ 上的语言

CONTENTS

目录

01

语言和文法
的直观概念

02

符号和
符号串

03

文法和语言
的形式定义

04

文法的
类型

05

上下文无关文
法及其语法树

06

句型的
分析

07

有关文法实用
中的一些说明

1. 文法的定义

- 文法定义:

- 文法**G**定义为**四元组**(V_N , V_T , P , S)

- ✓ V_N (**Nonterminal**): 非终结符集

- ✓ V_T (**Terminal**): 终结符集

- ✓ P (**Production**): **产生式 (规则)** 集合

- ✓ S (**Start**): 开始符号或识别符号

- V_N , V_T 和 P 是非空有穷集

- $V = V_N \cup V_T$, V 称为文法 G 的字母表

- $V_N \cap V_T = \Phi$

- S 是一个非终结符, 且至少要在一条产生式的左部出现

- 产生式 (规则) :

- 产生式是一个有序对(α, β), 通常写作 $\alpha \rightarrow \beta$ (或 $\alpha ::= \beta$) (读作: α 定义为 β)

$\langle \text{句子} \rangle ::= \langle \text{主语} \rangle \langle \text{谓语} \rangle$

P 中产生式形如: $\alpha \rightarrow \beta$, 其中: $\alpha \in V^+$ 且至少含一个非终结符, $\beta \in V^*$

1. 文法的定义

• 例1：文法 $G=(V_N, V_T, P, S)$ ，其中：

- $V_N = \{\text{句子, 主语, 代词, 名词, 谓语, 动词, 直接宾语}\}$
- $V_T = \{\text{我, 你, 他, 王明, 大学生, 工人, 英语, 是, 学习}\}$

– $P = \{$

$\langle \text{句子} \rangle \rightarrow \langle \text{主语} \rangle \langle \text{谓语} \rangle$

$\langle \text{主语} \rangle \rightarrow \langle \text{代词} \rangle \mid \langle \text{名词} \rangle$

$\langle \text{代词} \rangle \rightarrow \text{我} \mid \text{你} \mid \text{他}$

$\langle \text{名词} \rangle \rightarrow \text{王明} \mid \text{大学生} \mid \text{工人} \mid \text{英语}$

$\langle \text{谓语} \rangle \rightarrow \langle \text{动词} \rangle \langle \text{直接宾语} \rangle$

$\langle \text{动词} \rangle \rightarrow \text{是} \mid \text{学习}$

$\langle \text{直接宾语} \rangle \rightarrow \langle \text{代词} \rangle \mid \langle \text{名词} \rangle$

$\}$

– $S = \text{句子}$

- $\langle \text{句子} \rangle ::= \langle \text{主语} \rangle \langle \text{谓语} \rangle$
- $\langle \text{主语} \rangle ::= \langle \text{代词} \rangle \mid \langle \text{名词} \rangle$
- $\langle \text{代词} \rangle ::= \text{我} \mid \text{你} \mid \text{他}$
- $\langle \text{名词} \rangle ::= \text{王明} \mid \text{大学生} \mid \text{工人} \mid \text{英语}$
- $\langle \text{谓语} \rangle ::= \langle \text{动词} \rangle \langle \text{直接宾语} \rangle$
- $\langle \text{动词} \rangle ::= \text{是} \mid \text{学习}$
- $\langle \text{直接宾语} \rangle ::= \langle \text{代词} \rangle \mid \langle \text{名词} \rangle$

1. 文法的定义

- 例2：文法 $G=(V_N, V_T, P, S)$ ，其中：
 - $V_N = \{S\}$, $V_T = \{0, 1\}$,
 - $P = \{S \rightarrow 0S1, S \rightarrow 01\}$ ，开始符为 S
- 例3：文法 $G=(V_N, V_T, P, S)$ ，其中：
 - $V_N = \{\text{标识符}, \text{字母}, \text{数字}\}$,
 - $V_T = \{a, b, c, \dots, x, y, z, 0, 1, \dots, 9\}$,
 - $P = \{$
 - $\langle \text{标识符} \rangle \rightarrow \langle \text{字母} \rangle$, $\langle \text{标识符} \rangle \rightarrow \langle \text{标识符} \rangle \langle \text{字母} \rangle$
 - $\langle \text{标识符} \rangle \rightarrow \langle \text{标识符} \rangle \langle \text{数字} \rangle$,
 - $\langle \text{字母} \rangle \rightarrow a, \dots, \langle \text{字母} \rangle \rightarrow z$,
 - $\langle \text{数字} \rangle \rightarrow 0, \dots, \langle \text{数字} \rangle \rightarrow 9 \quad \}$,
 - $S = \langle \text{标识符} \rangle$

随堂练习(1)

- 为只包含数字、加号和减号的表达式，例如 $9-2+5$, $3-1$, 7 等构造一个文法

随堂练习(1)

- 为只包含数字、加号和减号的表达式，例如9-2+5,3-1,7等构造一个文法

- **G[式子]:**

- ✓ $\langle \text{式子} \rangle \rightarrow \langle \text{数字} \rangle | \langle \text{式子} \rangle \langle \text{运算符} \rangle \langle \text{式子} \rangle$
- ✓ $\langle \text{数字} \rangle \rightarrow 0 | 1 | 2 | 3 | 4 \dots | 9$
- ✓ $\langle \text{数字} \rangle \rightarrow \langle \text{数字} \rangle \langle \text{数字} \rangle$
- ✓ $\langle \text{运算符} \rangle \rightarrow + | -$

2. 文法的简化表示法

- 简化：通常不用将文法的四元组表示出来，**只写出产生式**
- 约定：
 - 默认第一条产生式的左部的符号是开始符号，或用 **$G[S]$** 表示 **S 是开始符号**；
 - 用**大写字母**（或用尖括号括起来）表示**非终结符**；
 - 用**小写字母**表示**终结符**；
 - 左部相同的产生式 **$A \rightarrow \alpha$** ， **$A \rightarrow \beta$** 可以记为； **$A \rightarrow \alpha | \beta$** ，其中“|”表示“或”
- 例如：

文法 $G[S]$:

$S \rightarrow A | SA | SD$

$A \rightarrow a | b | \dots | z$

$D \rightarrow 0 | 1 | \dots | 9$

$V_N = \{S, A, D\}$
 $V_T = \{a, b, \dots, z, 0, 1, \dots, 9\}$
 $P = \{$
 $S \rightarrow A | SA | SD$
 $A \rightarrow a | b | \dots | z$
 $D \rightarrow 0 | 1 | \dots | 9$ }
 S ——开始符号

3. 推导[Derivation]与归约[Reduction]

(1) 直接推导和直接归约

- $\alpha \rightarrow \beta$ 是文法 G 的产生式，若有 v, w 满足： $v = \gamma \alpha \delta, w = \gamma \beta \delta$ ，其中 $\gamma, \delta \in V^*$ ，则称：
 - ✓ v **直接推导** 到 w ，
 - ✓ 或称： w **直接归约** 到 v ，
 - ✓ 记作： $v \Rightarrow w$
- 直接推导：用产生式的右部替换掉产生式的左部
- 直接归约：用产生式的左部替换掉产生式的右部

例 文法 G ： $S \rightarrow 0S1, S \rightarrow 01$ 有直接推导：

0S1	\Rightarrow 00S11	($S \rightarrow 0S1$)
0S1	\Rightarrow 0011	($S \rightarrow 01$)
00S11	\Rightarrow 000S111	($S \rightarrow 0S1$)
000S111	\Rightarrow 00001111	($S \rightarrow 01$)
S	\Rightarrow 0S1	($S \rightarrow 0S1$)

3. 推导[Derivation]与归约[Reduction]

(2) 推导和归约

– 若存在 $v=w_0 \Rightarrow w_1 \Rightarrow \dots \Rightarrow w_n=w, (n>0)$ (即 v 经过**多步**推到到 w) , 则称:

✓ v 推导出 w ,

✓ 或称: w 归约到 v ,

✓ 记为: $v \Rightarrow^+ w$

– 若有 $v \Rightarrow^+ w$, 或 $v=w$, 则记作 $v \Rightarrow^* w$

$$\alpha \Rightarrow 0\beta, \beta \Rightarrow 2\gamma$$

$$\alpha \Rightarrow 0\beta \Rightarrow 02\gamma$$

$$\alpha \Rightarrow^+ 02\gamma$$

$$\alpha \Rightarrow^* 02\gamma$$

例 文法 G : $S \rightarrow 0S1, S \rightarrow 01$

$$\underline{S} \Rightarrow \underline{0S1} \Rightarrow 00\underline{S11} \Rightarrow 000\underline{S111} \Rightarrow 0000\underline{1111}$$

$$S \Rightarrow^+ 00001111$$

$$S \Rightarrow^* 00001111$$

4. 句型、句子、语言的定义

(1) 句型 and 句子

- 句型：由文法**开始符号S推导出的**符号串 α （即 $S \Rightarrow^* \alpha$ ），称为文法G[S]的**句型**。
- 句子：**仅由终结符组成的**句型 α （即 $S \Rightarrow^* \alpha$ ， $\alpha \in V_T^*$ ），称为文法G[S]的**句子**。

- 句子： **仅由终结符组成**的句型 α （即 $S \Rightarrow^* \alpha$, $\alpha \in V_T^*$ ），称为文法 $G[S]$ 的**句子**。

例 文法G: $S \rightarrow 0S1, S \rightarrow 01$

S \Rightarrow **0S1** \Rightarrow **00S11** \Rightarrow **000S111** \Rightarrow **00001111**

则： 句型 句型 句型 句型 句型 句子

句子

4. 句型、句子、语言的定义

(2) 语言的定义

– 语言：文法 $G[S]$ 的一切**句子的集合**称为语言，记做 $L(G)$

例 文法 G ：S \rightarrow 0S1, S \rightarrow 01

$$S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow 0^3S1^3 \Rightarrow \dots \Rightarrow 0^{n-1}S1^{n-1} \Rightarrow 0^n1^n$$

$$L(G) = \{0^n1^n | n \geq 1\}$$

4. 句型、句子、语言的定义

(2) 语言的定义

$$G = (V_N, V_T, P, S), V_N = \{S, B, E\}, V_T = \{a, b, e\}$$

$$(1) S \rightarrow aSBE$$

使用(1) $n-1$ 次, 得到 $S \xRightarrow{*} a^{n-1}S(BE)^{n-1}$

$$(2) S \rightarrow aBE$$

使用(2) 1次, 得到 $S \xRightarrow{*} a^n(BE)^n$

$$(3) EB \rightarrow BE$$

使用(3) 若干次, 得到 $S \xRightarrow{*} a^n B^n E^n$

$$(4) aB \rightarrow ab$$

使用(4) 1次, 得到 $S \xRightarrow{*} a^n b B^{n-1} E^n$

$$(5) bB \rightarrow bb$$

使用(5) $n-1$ 次, 得到 $S \xRightarrow{*} a^n b^n E^n$

$$(6) bE \rightarrow be$$

使用(6) 1次, 得到 $S \xRightarrow{*} a^n b^n e E^{n-1}$

$$(7) eE \rightarrow ee$$

使用(7) $n-1$ 次, 得到 $S \xRightarrow{*} a^n b^n e^n$

所以G产生的语言 $L(G) = \{a^n b^n e^n | n \geq 1\}$

5. 文法的等价

- 若 $L(G1)=L(G2)$ ，即，若两个文法所定义的语言是一样的，则称文法 $G1$ 和 $G2$ 是等价的。

例如：文法 $G1[A]: A \rightarrow 0R \ A \rightarrow 01 \ R \rightarrow A1$

$G2[S]: S \rightarrow 0S1 \ S \rightarrow 01$

所定义的语言都是 0^n1^n

因此，两文法等价。

CONTENTS

目 录

01

语言和文法
的直观概念

02

符号和
符号串

03

文法和语言
的形式定义

04

文法的
类型

05

上下文无关文
法及其语法树

06

句型的
分析

07

有关文法实用
中的一些说明

1. 0型文法

- 通过对产生式施加不同的限制，Chomsky将文法分为**四种类型**：
 - **0型**
 - **1型（上下文有关文法）**
 - **2型（上下文无关文法）**
 - **3型（正规文法）**
- **0型文法(短语文法)**：对任一产生式 $\alpha \rightarrow \beta$ ，都有 $\alpha \in (V_N \cup V_T)^+$ 且至少含有一个非终结符， $\beta \in (V_N \cup V_T)^*$ **(与原文法定义一致)**
- **任何文法都是0型文法。**

2. 1型文法(上下文有关文法[Context-Sensitive Grammar])

• **1型文法（上下文有关文法）**：0型文法的特例

- 设文法 $G=(V_N, V_T, P, S)$ ，对 P 中的任一产生式 $\alpha \rightarrow \beta$ ，都有 $|\beta| \geq |\alpha|$ ，仅 $S \rightarrow \varepsilon$ 除外

例 文法 $G[S]$:

$S \rightarrow aSBE$ $S \rightarrow aBE$ $EB \rightarrow BE$

$aB \rightarrow ab$ $bB \rightarrow bb$ $bE \rightarrow be$ $eE \rightarrow ee$

- 1型文法产生式的一般形式是 $\alpha A \gamma \rightarrow \alpha \beta \gamma$ ， $\alpha, \gamma \in V^*$ ， $A \in V_N$ ， $\beta \in V^+$ ，它表示当 A 的上文为 α 且下文为 γ 时可把 A 替换成 β ，因此称1型文法为**上下文有关文法**。

3. 2型文法(上下文无关文法[Context-Free Grammar])

- **2型文法（上下文无关文法[Context-Free Grammar, CFG]**）：1型文法的特例

- 对任一产生式 $\alpha \rightarrow \beta$ ，都有 $\alpha \in V_N$ ， $\beta \in (V_N \cup V_T)^*$

例 文法 $G[S]$ ： $S \rightarrow AB$ $A \rightarrow BS|0$ $B \rightarrow SA|1$

- 2型文法产生式的一般形式是： $A \rightarrow \beta$ ，它表示不管A的上下文如何即可把A替换成 β ，因此被称为上下文无关文法。
- **通常程序设计语言的文法——2型文法**

4. 3型文法 (正规文法[Regular Grammar])

- **3型文法(正规文法)**: 2型文法的特例

- 任一产生式 $\alpha \rightarrow \beta$ 的形式都为 $A \rightarrow aB$ 或 $A \rightarrow a$, 其中 $A, B \in V_N$, $a \in V_T$

例如 文法 $G[S]$: $S \rightarrow 0A | 1B | 0$

$A \rightarrow 0A | 1B | 0S$

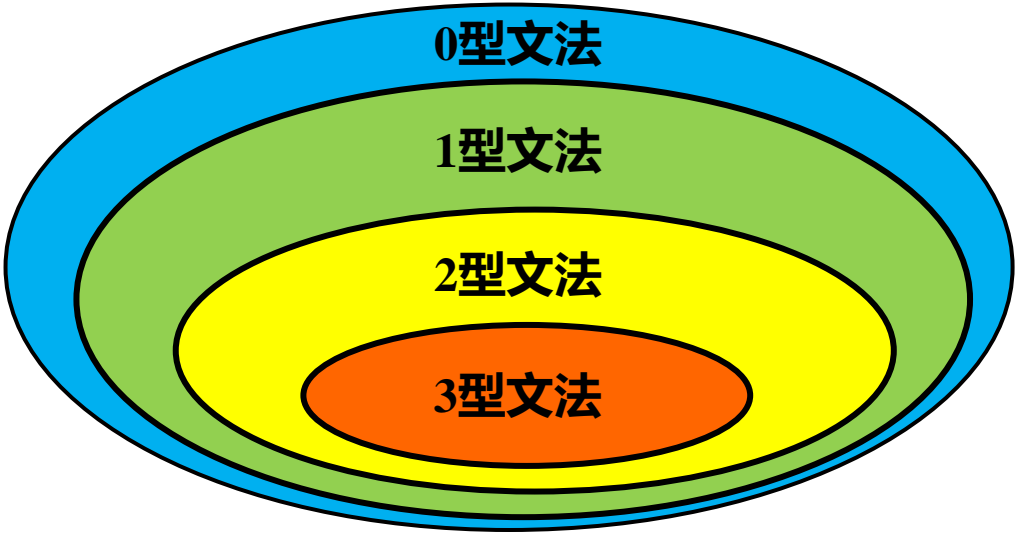
$B \rightarrow 1B | 1 | 0$

- 在程序设计语言中, 3型文法通常用来描述单词的结构

小结

文法类别	产生式形式	说明
0型文法 (短语文法)	$\alpha \rightarrow \beta, \alpha \in V^+, \text{ 且至少含一个非终结符, } \beta \in V^*$	对产生式基本无限制
1型文法 (上下文有关文法)	$\alpha \rightarrow \beta, \beta \geq \alpha \geq 1 \text{ 或 } \alpha A \gamma \rightarrow \alpha \beta \gamma, \alpha, \gamma \in V^*, A \in V_N, \beta \in V^+$	将 A 替换成 β 时, 必须考虑 A 的上下文 α, γ
2型文法 (上下文无关文法)	$A \rightarrow \beta, A \in V_N, \beta \in V^*$	无需考虑 A 在上下文中的出现情况
3型文法 (正规文法)	$A \rightarrow aB \text{ 或 } A \rightarrow a, A, B \in V_N, a \in V_T$	产生式全部是规定的形式

四种文法之间的逐级“包含”关系



CONTENTS

目 录

01

语言和文法
的直观概念

02

符号和
符号串

03

文法和语言
的形式定义

04

文法的
类型

05

上下文无关文
法及其语法树

06

句型的
分析

07

有关文法实用
中的一些说明

1. 上下文无关文法 (2型)

- 上下文无关文法有足够的描述能力描述现今程序设计语言的语法结构
- 因此, 我们只关心上下文无关文法形成的语言中的句子的分析

例1: 算术表达式: $E \rightarrow i | E + E | E * E | (E)$

例2: $\langle \text{赋值语句} \rangle \rightarrow i := E$

$\langle \text{条件语句} \rangle \rightarrow \text{if } \langle \text{条件} \rangle \text{ then } \langle \text{语句} \rangle$

$\quad | \text{if } \langle \text{条件} \rangle \text{ then } \langle \text{语句} \rangle \text{ else } \langle \text{语句} \rangle$

2. 规范推导和规范句型

- 如果在推导的任何一步 $\alpha \rightarrow \beta$ ，其中 α 、 β 是句型，都是对 α 中的最左非终结符进行替换，则称这种推导为**最左推导**；同理，如果是总对最右的非终结符进行替换，则称为**最右推导**
- **最右推导**被称为**规范推导**
- 由规范推导所得的句型称为**规范句型（右句型）**

例 文法G： $E \rightarrow E+T \mid T$

$T \rightarrow T \times F \mid F$

$F \rightarrow (E) \mid i$

句子 $i+i \times i$ 的推导过程如下：

最左推导： $\underline{E} \Rightarrow \underline{E}+T \Rightarrow \underline{T}+T \Rightarrow \underline{F}+T \Rightarrow i+\underline{T} \Rightarrow i+\underline{T} \times F \Rightarrow i+\underline{F} \times F \Rightarrow i+i \times \underline{F} \Rightarrow i+i \times i$

最右推导： $\underline{E} \Rightarrow \underline{E}+\underline{T} \Rightarrow E+T \times \underline{F} \Rightarrow E+T \times i \Rightarrow E+\underline{F} \times i \Rightarrow E+i \times i \Rightarrow \underline{T}+i \times i \Rightarrow \underline{F}+i \times i \Rightarrow i+i \times i$

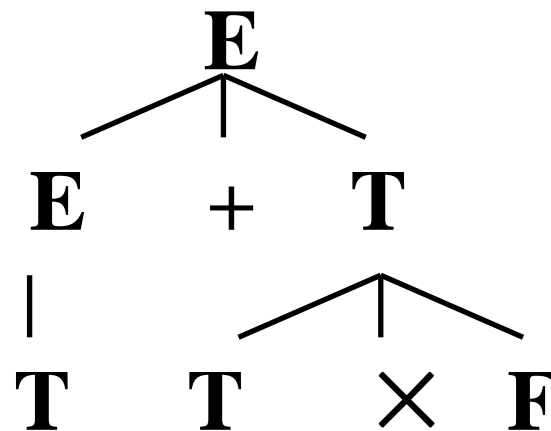
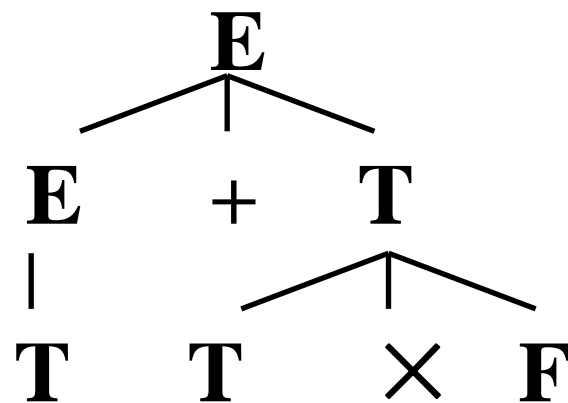
3. 语法树(推导树)[Parse Tree]

- 作用：直观地描述上下文无关文法的**句型推导过程**。给定文 $G=(V_N, V_T, P, S)$ ，对于 G 的任何句型都能构造与之关联的语法树

文法 G : $E \rightarrow E+T \mid T$
 $T \rightarrow T \times F \mid F$
 $F \rightarrow (E) \mid i$

$\underline{E} \Rightarrow \underline{E} + \underline{T} \Rightarrow \underline{E} + T \times F \Rightarrow T + T \times F$

$\underline{E} \Rightarrow \underline{E} + T \Rightarrow T + \underline{T} \Rightarrow T + T \times F$



从左到右读出叶子结点得到的符号串，为文法的句型。也把该语法树称为该句型的语法树。

从语法树中看不出句型中的符号被替代的顺序

3. 语法树(推导树)[Parse Tree]

- 语法树定义:

- 给定文法 $G=(V_N, V_T, P, S)$, 若一棵树满足下列4个条件, 则称此树为 G 的语法树:

1. 每个结点都有一个标记, 此标记是 V 的一个符号

- 2. 根的标记是 S**

3. 若一结点 n 至少有一个它自己除外的子孙, 并且有标记 A , 则肯定 $A \in V_N$

4. 如果结点 n 有标记 A , 其直接子孙结点从左到右的次序是 n_1, n_2, \dots, n_k , 其标记分别为 A_1, A_2, \dots, A_k , 那么 $A \rightarrow A_1 A_2 \dots A_k$ 一定是 P 中的一个产生式

随堂练习(2)

$S \rightarrow SS^* | SS+ | a$

通过此文法如何生成串 **aa+a***，并为该串构造**语法树**。

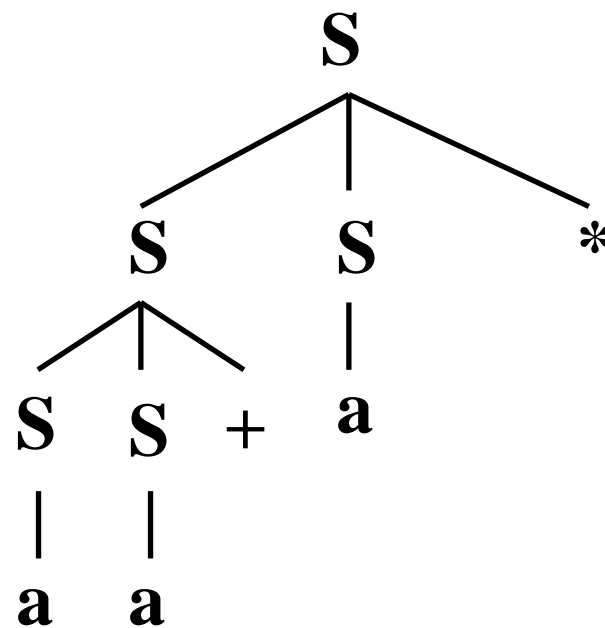
$S \Rightarrow SS^*$

$\Rightarrow SS+S^*$

$\Rightarrow aS+S^*$

$\Rightarrow aa+S^*$

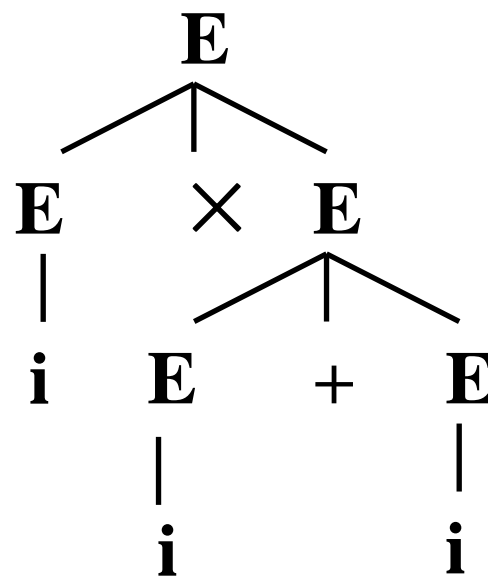
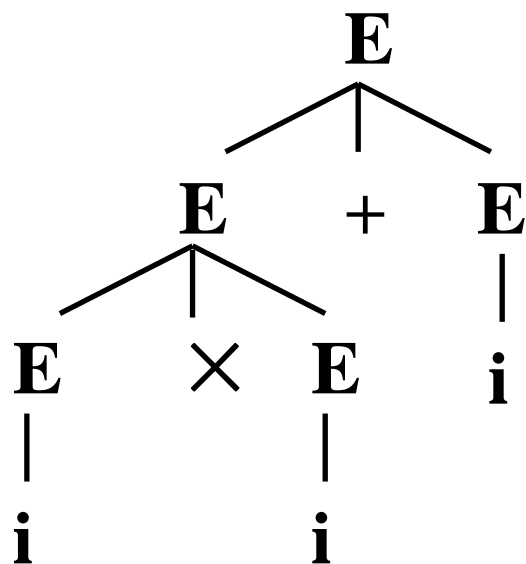
$\Rightarrow aa+a^*$



4. 文法的二义性[Ambiguity]

- 文法G: $E \rightarrow E + E \mid E \times E \mid (E) \mid i$

句子 $i \times i + i$ 对应的语法树



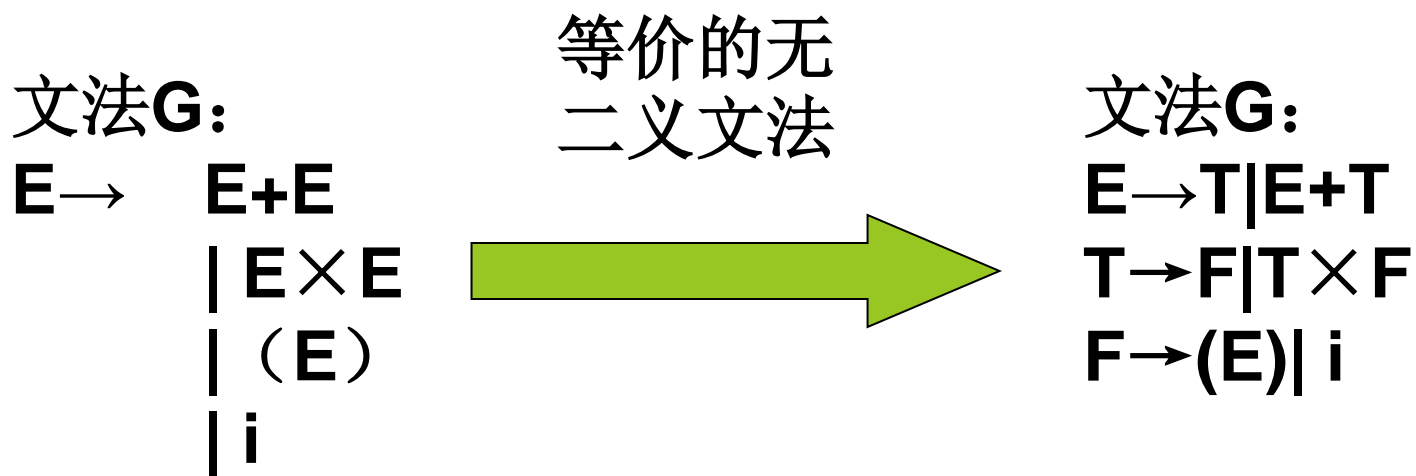
存在两个不同的最左推导:

推导1: $\underline{E} \Rightarrow \underline{E} + E \Rightarrow \underline{E} \times E + E \Rightarrow i \times \underline{E} + E \Rightarrow i \times i + \underline{E} \Rightarrow i \times i + i$

推导2: $\underline{E} \Rightarrow \underline{E} \times E \Rightarrow i \times \underline{E} \Rightarrow i \times \underline{E} + E \Rightarrow i \times i + \underline{E} \Rightarrow i \times i + i$

4. 文法的二义性[Ambiguity]

- 定义：如果一个文法存在某个句子对应**两棵不同的语法树**，则说这个文法是**二义的**
- 二义性文法存在某个句子，它有**两个不同的最左（右）推导**



任何一个二义性的文法，都可以转换成一个等价的无二义性文法。

- 考虑文法 $S \rightarrow aSbS \mid bSaS \mid \varepsilon$ ，试说明此文法是二义性的。
- 提示：可以从对于句子 $abab$ 有两个不同的最左推导来说明。
- 解：
 - $S \Rightarrow aSbS \Rightarrow a\varepsilon bS \Rightarrow a\varepsilon baSbS \Rightarrow a\varepsilon ba\varepsilon bS \Rightarrow a\varepsilon ba\varepsilon b\varepsilon \Rightarrow abab$
 - $S \Rightarrow aSbS \Rightarrow abSaSbS \Rightarrow ab\varepsilon aSbS \Rightarrow ab\varepsilon a\varepsilon bS \Rightarrow ab\varepsilon a\varepsilon b\varepsilon \Rightarrow abab$
 - 存在两个不同的最左推导。
 - 因此文法是二义的。

CONTENTS

目录

01

语言和文法的
直观概念

02

符号和
符号串

03

文法和语言的
形式定义

04

文法的
类型

05

上下文无关文
法及其语法树

06

句型的
分析

07

有关文法实用
中的一些说明

1. 句型分析

- 对于程序设计语言来说，句型分析就是一个识别输入符号串是否为语法上正确的程序的过程。
- **任务**：句型分析就是**识别一个符号串是否为某文法的句型**，是某个推导的构造过程。
- 由文法开始符号 S 推导出的符号串 α （即 $S \Rightarrow \alpha$ ），称为文法 $G[S]$ 的句型。
- **若能根据该文法构造出该符号串的语法树，则该符号串就是该文法的句型。**
- 句型分析算法采用**从左到右的分析算法**，即总是从左到右地识别输入符号串
- 句型的分析算法分类
 - **自上而下分析法 (Top-Down parsing)**
 - **自下而上分析法 (Bottom-Up parsing)**

2. 自上而下的分析方法[Top-Down parsing]

- 定义:

- 从文法的开始符号出发, 反复使用文法的产生式, 寻找与输入符号串匹配的推导。

- 语法树的构造:

- 将文法的开始符号作为语法树的根, 向下逐步建立语法树, 使语法树的末端结点符号串正好是输入符号串。

推导过程: $\underline{S} \Rightarrow cAd \Rightarrow cabd$

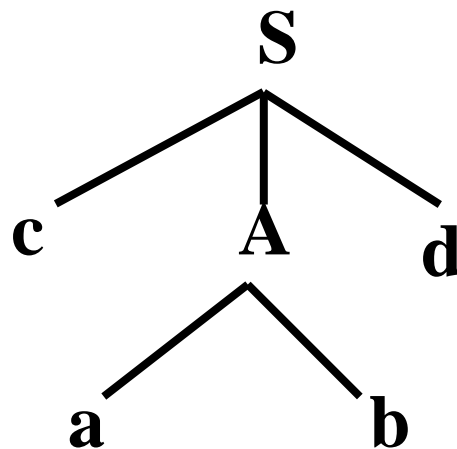
例: 文法G:

$S \rightarrow cAd$

$A \rightarrow ab$

$A \rightarrow a$

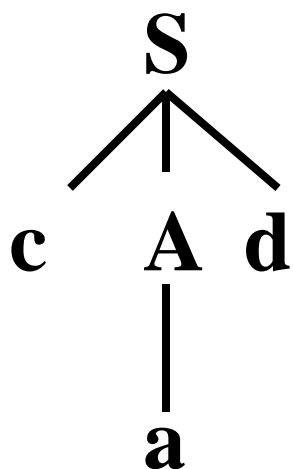
识别输入串 $w=cabd$ 是否为该文法的句子



2. 自上而下的分析方法[Top-Down parsing]

- 自上而下方法的主要问题

例：对输入串cabd自上而下构造语法树的另一过程



不成功!

**不成功的原因是
选错产生式 $A \rightarrow a$**

$S \rightarrow cAd$
 $A \rightarrow ab$
 $A \rightarrow a$

自上而下分析的**主要问题是选择产生式**:

假定要被代换的最左非终结符号是 B ，且有 n 条规则： $B \rightarrow A_1 | A_2 | \dots | A_n$ ，那么如何确定用哪个右部去替代 B ？

3. 自下而上的分析方法 [Bottom-Up parsing]

- 定义:

- 从输入符号串开始, 逐步进行归约, 直至归约到文法的开始符号。

- 语法树的构造:

- 从输入符号串开始, 以它作为语法树的末端结点符号串, 自底向上的构造语法树。

归约过程: cabd |- cAd |- S

用 “|-” 表示归约, 下划线部分为被归约符号

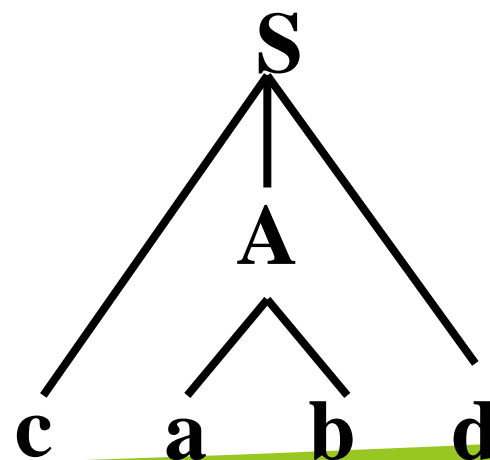
例: 文法G:

$S \rightarrow cAd$

$A \rightarrow ab$

$A \rightarrow a$

识别输入串 $w = cabd$ 是否为该文法的句子



3. 自下而上的分析方法 [Bottom-Up parsing]

- 自下而上分析的主要问题

例：对输入串cabd的两种归约过程

(1) **cabd|-cAd|-S** 归约到开始符

(2) **cabd|-cAbd** 不能归约到开始符

$S \rightarrow cAd$

$A \rightarrow ab$

$A \rightarrow a$

- 在自下而上的分析方法中，每一步都是从当前串中选择一个子串加以归约，该子串暂称“**可归约串**”。
- 自下而上分析的**主要问题**：**如何确定“可归约串”——“句柄”**。

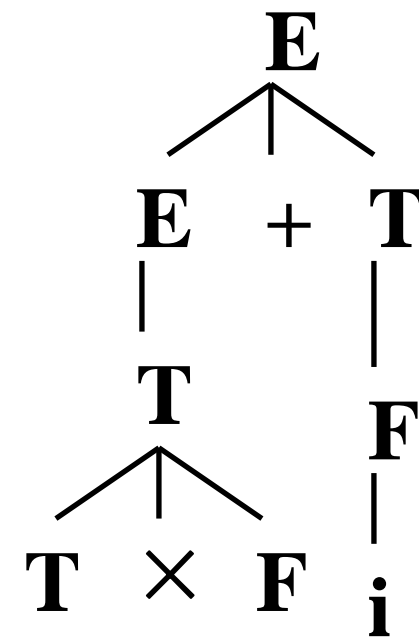
4. 短语[phrase], 直接短语和句柄[handle] (重要!)

- 设 $\alpha\beta\delta$ 是文法 $G[S]$ 中的一个句型, 如果有 $S \Rightarrow^* \alpha A \delta$ 且 $A \Rightarrow^+ \beta$, 则称 β 是句型 $\alpha\beta\delta$ 相对于非终结符 A 的**短语[phrase]**。
- 特别的如有 $A \Rightarrow \beta$, 则称 β 是句型 $\alpha\beta\delta$ 相对于规则 $A \rightarrow \beta$ 的**直接短语 (简单短语)**。
- 一个句型的最左直接短语称为该句型的**句柄[Handle]**。
- **句柄就是“可归约串”。**
- 注意: 在短语的定义中包括了三个条件, 都必须满足:
 - (1) $\alpha\beta\delta$ 是文法的一个句型;
 - (2) $S \Rightarrow^* \alpha A \delta$;
 - (3) $A \Rightarrow^+ \beta$ 。
 - ✓ (1) (2) 说明 $\alpha\beta\delta$ 、 $\alpha A \delta$ 都必须是句型;
 - ✓ (2) (3) 说明, 将 $\alpha\beta\delta$ 中的 β 归约 A 后, 得到的 $\alpha A \delta$ 一定要是句型。假如符号串 β , 将其归约成 A 后得到的符号串不能由开始符号推出, 则 β 不是短语。

4. 短语[phrase], 直接短语和句柄[handle] (重要!)

例: 文法 $G[E]: E \rightarrow E + T \mid T, T \rightarrow T \times F \mid F, F \rightarrow (E) \mid i$ 的一个句型是 $T \times F + i$, 相应的语法树见右图:

1. 因为 $E \Rightarrow^* T + i$ 且 $T \Rightarrow T \times F$, 所以 $T \times F$ 是句型相对于 T 的短语, 且是相对于 $T \rightarrow T \times F$ 的直接短语;
2. 因为 $E \Rightarrow^* T \times F + F$ 且 $F \Rightarrow i$, 所以 i 是句型相对于 F 的短语, 且是相对于 $F \rightarrow i$ 的直接短语;
3. 因为 $E \Rightarrow^* E$ 且 $E \Rightarrow^+ T \times F + i$, 所以 $T \times F + i$ 是句型相对于 E 的短语;
4. $T \times F$ 是最左直接短语, 即句柄。



注意: 虽然 $F + i$ 是句型 $T \times F + i$ 的一部分, 但**不是短语**, 因为尽管有 $E \Rightarrow^+ F + i$, 但是**不存在从文法开始符 $E \Rightarrow^* T \times E$ 的推导**

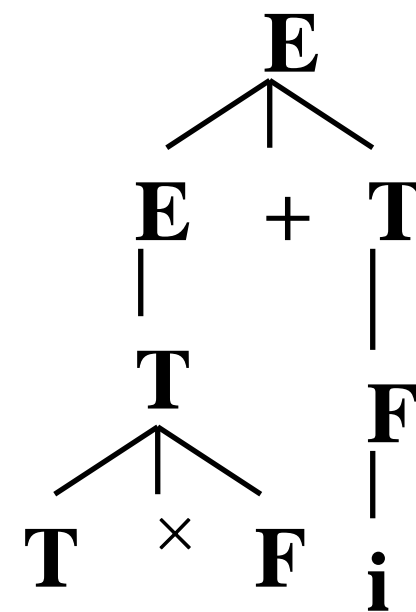
若有 $S \Rightarrow^* \alpha A \delta$ 且 $A \Rightarrow^+ \beta$, 则称 β 是句型 $\alpha \beta \delta$ 相对于非终结符 A 的短语。

5. 短语与语法树

- 从句型的语法树上很容易找出句型的短语——语法树中每棵子树的末端结点构成相对于子树根的短语

对于句型 $T \times F + i$ 来说:

- ✓ 五棵子树对应五个**短语** (其中2个重复): $T \times F$, i , $T \times F + i$
- ✓ 两层子树的末端结点构成直接短语, 两棵两层子树对应两个**直接短语**: $T \times F$, i
- ✓ 位于最左边的两层子树的末端结点构成**句柄**: $T \times F$



5. 短语与语法树

文法G[E]:

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

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

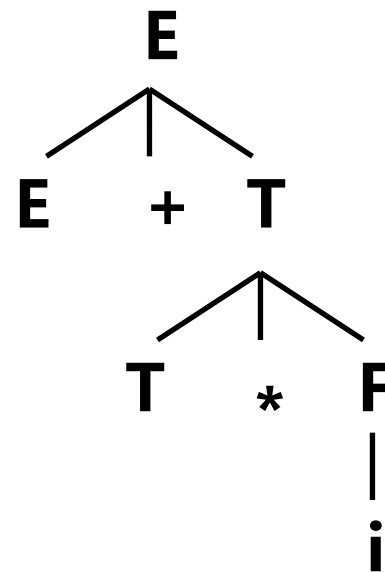
□ $F \rightarrow (E) \mid i$ 的一个句型是 $E + T * i$

3棵子树，对应3个短语：

□ $E + T * i$

□ $T * i$

□ i (直接短语、句柄)



5. 短语与语法树

文法 $G[E]: E \rightarrow E + T \mid T, T \rightarrow T \times F \mid F, F \rightarrow (E) \mid i$ 的一个句型是 $F \times i + i$

➤ 7棵子树,对应有7个短语:

$F \times i + i$ ($E = * > \underline{E}, E = * > F \times i + i$)

$F \times i$ ($E = * > \underline{E} + T, E = * > F \times i$)

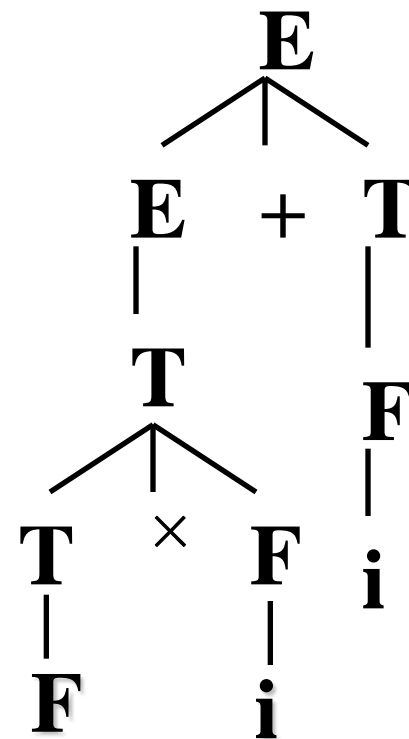
$F \times i$ ($E = * > \underline{T} + T, T = * > F \times i$)

句柄 $\leftarrow F$ ($E = * > \underline{T} \times i + i, T = * > F$) (直接短语)

i ($E = * > F \times \underline{F} + i, F = * > i$) (直接短语)

i ($E = * > F \times i + \underline{T}, T = * > i$)

i ($E = * > F \times i + \underline{F}, F = * > i$) (直接短语)



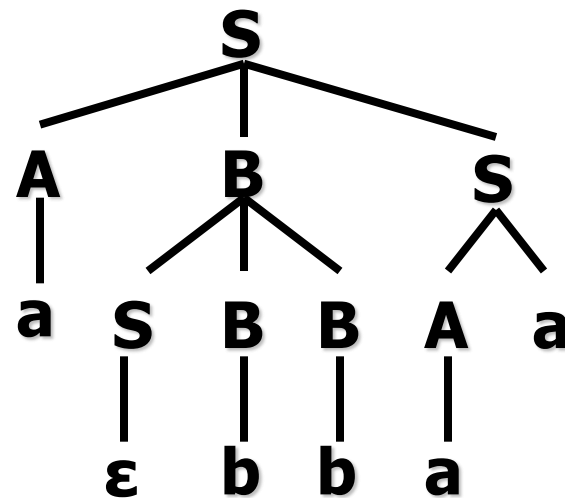
5. 短语与语法树

- 小结：找短语、直接短语、句柄的方法：
 - 找短语、直接短语、句柄时，可以通过**语法树**来进行。
 - 句型或句子才有所谓的短语等，所以要**先确定句型或句子**，然后再找它的短语、直接短语、句柄。
 - **要求某一句型的句柄等，语法树只需要分析到该句型。**

求句子abbaa的短语、直接短语、句柄。

8棵子树，对应8个短语：

- abbaa
- a (直接短语)
- bb → (句柄)
- aa
- ϵ (直接短语)
- b (直接短语)
- b (直接短语)
- a (直接短语)



CONTENTS

目录

01

语言和文法
的直观概念

02

符号和
符号串

03

文法和语言
的形式定义

04

文法的
类型

05

上下文无关文
法及其语法树

06

句型的
分析

07

有关文法实用
中的一些说明

1. 有关文法的实用限制

• 有害规则和多余规则

- 有害规则： $U \rightarrow U$ ，无用且引起文法的二义性
- 多余规则：所有句子推导都用不到的规则，表现在：
 - ✓ 不可到达：文法中某些非终结符不在任何规则的右部出现，所以任何句子的推导中都无法用到它。
 - ✓ 不可终止：不可推导出终结符号串的非终结符。

例：文法 $G[S]$ ：

多余规则为：

(1) $S \rightarrow Be$ ~~(2) $B \rightarrow Ce$~~ (3) $B \rightarrow Af$

(4) $A \rightarrow Ae$ (5) $A \rightarrow e$ ~~(6) $C \rightarrow Cf$~~

~~(7) $D \rightarrow f$~~

• 不可到达 (7)

• 不可终止 (6)

• (2) 也是不可终止的

1. 有关文法的实用限制

- 为避免有害规则和多余规则：
 - 非终结符 A 必须在某句型中出现。即有 $S \Rightarrow^* \alpha A \beta$, 其中 α, β 均属于 $(V_N \cup V_T)^*$
 - 必须能从 A 推出终结符号串 t 来。即 $A \Rightarrow^* t$, 其中 $t \in V_T^*$

2. 上下文无关文法中的 ε 规则

- 上下文无关文法中允许使用 $A \rightarrow \varepsilon$ 产生式, $A \rightarrow \varepsilon$ 称为 ε 规则

第二章课后作业

- 1. 令文法 $G[E]$ 为:

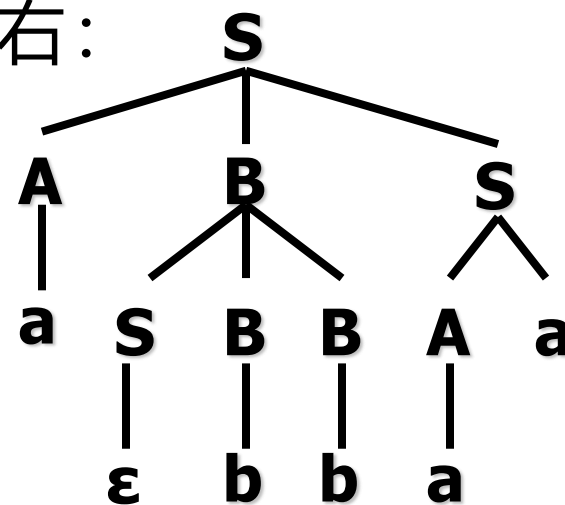
$$E \rightarrow T | E + T | E - T$$

$$T \rightarrow F | T * F | T / F$$

$$F \rightarrow (E) | i$$

证明 $E + T * F$ 是它的一个规范句型，指出这个句型的所有短语、直接短语和句柄。

- 2. 一个上下文无关文法生成句子abb aa的唯一语法树如右:



- (1) 给出该句子相应的最左推导和最右推导。
- (2) 该文法的产生式集合 P 可能有哪些元素?
- (3) 找出该句子的所有短语、简单短语和句柄。

第二章课后作业

- 提交要求：

- 文件命名：学号-姓名-第二章作业；
- 文件格式：.pdf文件；
- 手写版、电子版均可；若为手写版，则拍照后转成pdf提交，但**须注意将照片旋转为正常角度，且去除照片中的多余信息**；电子版如word等转成pdf提交；
- 提交到超算习堂（第二章作业）处；
- 提交ddl：**3月11日晚上12:00**；
- **重要提示：不得抄袭！**

课前/课后 复习/提问

1. 根据右侧的文法推导出句子“他是王明”

〈句子〉

⇒ 〈主语〉 〈谓语〉

⇒ 〈代词〉 〈谓语〉

⇒ 他 〈谓语〉

⇒ 他 〈动词〉 〈直接宾语〉

⇒ 他 是 〈直接宾语〉

⇒ 他是 〈名词〉

⇒ 他是 王明

- $\langle \text{句子} \rangle ::= \langle \text{主语} \rangle \langle \text{谓语} \rangle$
- $\langle \text{主语} \rangle ::= \langle \text{代词} \rangle \mid \langle \text{名词} \rangle$
- $\langle \text{代词} \rangle ::= \text{我} \mid \text{你} \mid \text{他}$
- $\langle \text{名词} \rangle ::= \text{王明} \mid \text{大学生} \mid \text{工人} \mid \text{英语}$
- $\langle \text{谓语} \rangle ::= \langle \text{动词} \rangle \langle \text{直接宾语} \rangle$
- $\langle \text{动词} \rangle ::= \text{是} \mid \text{学习}$
- $\langle \text{直接宾语} \rangle ::= \langle \text{代词} \rangle \mid \langle \text{名词} \rangle$

2. 进行如下字符串运算

(1) $x = \text{"abc"}, y = \text{"defg"}$, 则 $xy = ?$

$xy = \text{"abcdefg"}$

(2) $x = \text{"abc"}$, 则 $x^2 = ?$

$x^2 = \text{"abcabc"}$

(3) $A = \{a, b, c\}$ $B = \{d, e\}$, 则 $AB = ?$

$AB = \{ad, ae, bd, be, cd, ce\}$

(4) $B = \{d, e\}$, 则 $B^2 = ?$

$B^2 = BB = \{d, e\} \{d, e\} = \{dd, de, ed, ee\}$

3. 文法G: $S \rightarrow 0S1$, $S \rightarrow 01$

由文法G推导出句子0000011111

$\underline{S} \Rightarrow 0\underline{S}1$

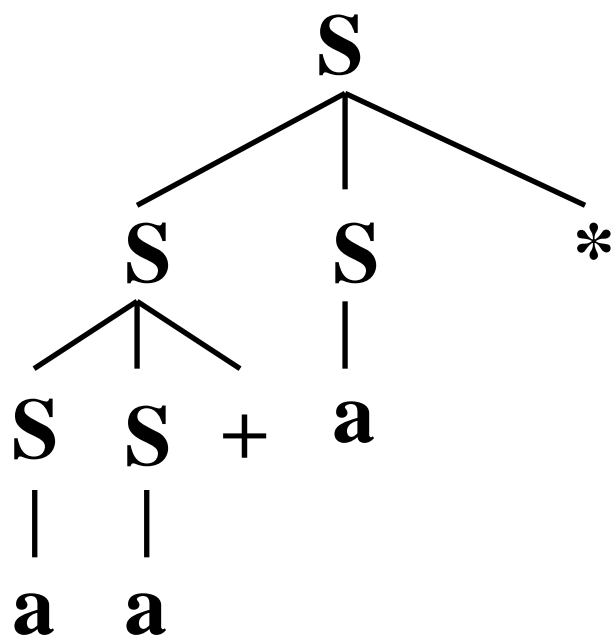
$\Rightarrow 00\underline{S}11$

$\Rightarrow 000\underline{S}111$

$\Rightarrow 0000\underline{S}1111$

$\Rightarrow 0000011111$

- 4. 找出句型 $aa+a^*$ 的短语、直接短语和句柄



5棵子树，对应5个短语：

✓ $aa+a^*$

✓ $aa+$

✓ a (直接短语)

✓ a (直接短语) (句柄)

✓ a (直接短语)