Regular Grammar

Yajun Yang

yjyang@tju.edu.cn

School of Computer Science and Technology
Tianjin University

Outline

Outline

文法

在语言中，一个文法（语法）是一组规则的集合

Example

用下述语法规则定义英语中的某些简单句子

1. <Sentence>→<Noun phrase><Verb phrase>
2. <Noun phrase>→<Article><Noun>
3. <Article>→the|a
4. <Noun>→apple|cat|man
5. <Verb phrase>→<Verb><Noun phrase>|<Verb>
6. <Verb>→eats|sings|runs

The man eats the apple.

文法

Definition

一个文法G是一个四元组

$$G = (V, T, P, S)$$

其中

1. V是变元的有限集。
2. T是终结符的有限集。
3. P是产生式的有限集，其中每个产生式都是$\alpha \to \beta$的形式，$\alpha \in (V \cup T)^+$，且至少有一个V中的符号，$\beta \in (V \cup T)^*$。α称为产生式的左部，β称为产生式的右部。
4. $S \in V$，称为文法G的开始符号。

约定

1. 变元：大写拉丁字母A，B，C，D，E和S，S开始符号（除非另作说明）。
2. 终结符：小写拉丁字母a，b，c，d，e，数字。
3. 终结符串：小写拉丁字母u，v，w，x，y，z等。
4. 变元和终结符共同组成的串：小写希腊字母α，β，γ等。

约定

1. 变元：大写拉丁字母A，B，C，D，E和S，
   S开始符号（除非另作说明）。
2. 终结符：小写拉丁字母a，b，c，d，e，数字。
3. 终结符串：小写拉丁字母u，v，w，x，y，z等。
4. 变元和终结符共同组成的串：小写希腊字母α，β，γ等。

约定

$$\alpha \to \beta_1, \ \alpha \to \beta_2, \ \cdots, \ \alpha \to \beta_n$$

缩写为：$\alpha \to \beta_1 \mid \beta_2 \mid \cdots \mid \beta_n$

例

写一个文法，只写出产生式集合便可

Example

$$S \to 0A1 \mid 10$$

$$0A \to 00A1$$

$$A \to \varepsilon$$

推导

Definition (直接推导)

给定文法$G = (V, T, P, S)$，定义两个字符串之间的关系$\Rightarrow$：
若$\alpha = \alpha_1\alpha_2\alpha_3$，$\gamma = \alpha_1\beta\alpha_3$，并且$\alpha_2 \to \beta$是$P$中的一条产生式，则有$\alpha \Rightarrow \gamma$，此时称由$\alpha$直接推导出$\gamma$

推导

Definition (直接推导)

给定文法$G=(V,T,P,S)$，定义两个字符串之间的关系$\Rightarrow$：若$\alpha=\alpha_1\alpha_2\alpha_3$，$\gamma=\alpha_1\beta\alpha_3$，并且$\alpha_2\to\beta$是$P$中的一条产生式，则有$\alpha\Rightarrow\gamma$，此时称由$\alpha$直接推导出$\gamma$

Definition (推导)

$\Rightarrow$是$(V\cup T)^*$上的二元关系。根据关系闭包的定义，可将$\Rightarrow$扩充为$\overset{*}{\Rightarrow}$，$\alpha\overset{*}{\Rightarrow}\gamma$称为由$\alpha$推导出$\gamma$。

推导

Definition (直接推导)

给定文法$G=(V,T,P,S)$，定义两个字符串之间的关系$\Rightarrow$：
若$\alpha=\alpha_1\alpha_2\alpha_3$，$\gamma=\alpha_1\beta\alpha_3$，并且$\alpha_2\to\beta$是$P$中的一条产生式，则有$\alpha\Rightarrow\gamma$，此时称由$\alpha$**直接推导**出$\gamma$

Definition (推导)

$\Rightarrow$是$(V\cup T)^*$上的二元关系。根据关系闭包的定义，可将$\Rightarrow$ 扩充为$\overset{*}{\Rightarrow}$，$\alpha\overset{*}{\Rightarrow}\gamma$ 称为由α**推导**出γ。

Definition (句型、句子)

若有$S\overset{*}{\Rightarrow}\gamma$，则称$\gamma$为**句型**，当$\gamma\in T^*$时，称$\gamma$为**句子**。

例

文法

Example

$$S \to 0A1 \mid 10$$

$$0A \to 00A1$$

$$A \to \varepsilon$$

有推导

Example

$$S \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000111$$

$$S \overset{*}{\Rightarrow} 000111$$

其中哪些是句型? 哪些是句子?

语言

Definition (语言)

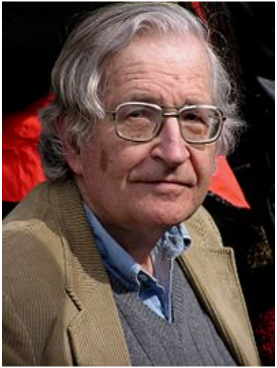
文法$G=(V,T,P,S)$所产生的语言记作$L(G)$，定义为：

$$L(G)=\{w \mid S \overset{*}{\Rightarrow} w \ \wedge \ w \in T^*\}$$

文法G产生的语言$L(G)$，就是由G中开始符号S推导出来的全体终结符串的集合，即句子的集合。

Outline

乔姆斯基



Noam Chomsky

Dec 7, 1928 (age 86)

语言学家、哲学家、认知科学家、政治评论家、社会活动家

MIT语言学与哲学系教授

“现代语言学之父”

乔姆斯基分类

Definition (Chomsky hierarchy)

对于文法$G = (V, T, P, S)$，按以下标准分为4类：

(1) 若P中的产生式不加另外的限制，
则称G为0型文法（Type-0 Grammar）或无限制文法（Unrestricted Grammar），
简记为UG。

乔姆斯基分类

Definition (Chomsky hierarchy)

对于文法$G = (V, T, P, S)$，按以下标准分为4类：

(2) 若P中的每个产生式都具有如下形式：

$\alpha \to \beta$

且满足 $|\alpha| \leqslant |\beta|$

则称G为1型文法（Type-1 Grammar）或上下文有关文法（Context-Sensitive Grammar），简记为CSG。

乔姆斯基分类

Definition (Chomsky hierarchy)

对于文法$G=(V,T,P,S)$，按以下标准分为4类：

(2) 若P中的每个产生式都具有如下形式：

$\alpha \rightarrow \beta$

且满足 $|\alpha| \leqslant |\beta|$

则称G为1型文法（Type-1 Grammar）或上下文有关文法（Context-Sensitive Grammar），简记为CSG。

上下文有关文法的另一定义

若P中每个产生式都具有如下形式： $\alpha A \beta \rightarrow \alpha \gamma \beta$

其中，$A \in V$，$\alpha, \beta \in (V \cup T)^*, \gamma \in (V \cup T)^+$

且产生式 $S \rightarrow \varepsilon$允许出现，只要$S$不出现在任何产生式的右部。

乔姆斯基分类

Definition (Chomsky hierarchy)

对于文法$G=(V,T,P,S)$，按以下标准分为4类：

(3) 若P中的每个产生式都具有如下形式：

$A \to \beta$

其中，$\beta \in (V \cup T)^*$，$A \in V$

则称G为2型文法（Type-2 Grammar）或上下文无关文法

（Context-Free Grammar），

简记为CFG。

乔姆斯基分类

Definition (Chomsky hierarchy)

对于文法$G = (V, T, P, S)$，按以下标准分为4类：

(4) 若P中的每个产生式都具有如下形式：

$A \to a$

$A \to aB$

其中，$a \in T \cup \{\varepsilon\}$，$A, B \in V$

则称G为3型文法（Type-3 Grammar）或正则文法（Regular Grammar），

简记为RG。

乔姆斯基分类：语言

Definition (语言分类)

1. 由**无限制文法**产生的语言称为递归可枚举语言，简记为REL（Recursively Enumerable Language）。
2. 由**上下文有关文法**产生的语言称为上下文有关语言，简记为CSL（Context-Sensitive Language）。
3. 由**上下文无关文法**产生的语言称为上下文无关语言，简记为CFL（Context-Free Language）。
4. 由**正则文法**产生的语言称为正则语言，简记为RL（Regular Languages）。

Outline

RG和FA的关系

- 联系
  - 正则文法
  - 有穷自动机（DFA和NFA）

通过两个定理，证明RG和FA等价。

RG和FA的关系

Theorem (定理)

设L被某个正则文法G产生，则L可被某个有穷自动机接受。

RG和FA的关系

Theorem (定理)

设L被某个正则文法G产生，则L可被某个有穷自动机接受。

证明

板书

RG和FA的关系

Example (例)

给出正则文法G_1如下：

$$S \to 0B$$

$$B \to 0B \mid 1S \mid 0$$

根据定理给出的方法，构造对应的FA。

RG和FA的关系

Example (例)

给出正则文法G_1如下：

$$S \rightarrow 0B$$

$$B \rightarrow 0B \mid 1S \mid 0$$

根据定理给出的方法，构造对应的FA。

板书

RG和FA的关系

Example (例)

给出正则文法G_2如下：

$$S \to 0A$$

$$A \to 1A \mid B$$

$$B \to 0 \mid \varepsilon$$

根据定理给出的方法，构造对应的FA。

RG和FA的关系

Example (例)

给出正则文法G_2如下：

$$S \to 0A$$

$$A \to 1A \mid B$$

$$B \to 0 \mid \varepsilon$$

根据定理给出的方法，构造对应的FA。

板书

RG和FA的关系

Theorem (定理)

设L被某个*DFA* M接受，则L可被某个正则文法产生。

RG和FA的关系

Theorem (定理)

设L被某个*DFA* M接受，则L可被某个正则文法产生。

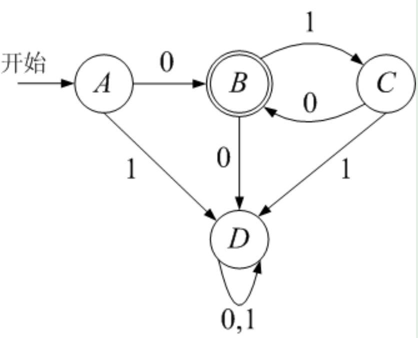
证明

板书

RG和FA的关系

Example (例)

给出一个DFA，如下：

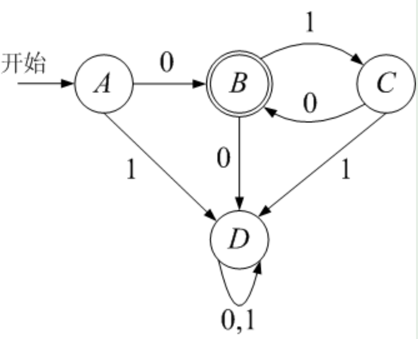


根据定理的构造方法，给出对应的正则文法。

RG和FA的关系

Example (例)

给出一个DFA，如下：



根据定理的构造方法，给出对应的正则文法。板书

正则语言的各种表达形式

- 正则语言
  - 有穷自动机（FA）接受的语言
  - 正则表达式（RE）代表的语言
  - 正则文法（RG）产生的语言

根据需要选择任何一种表达形式。