第 七 讲

Minimization of DFA
CFG and Derivations

- Minimization of DFA's
- Context-Free Grammars and Derivations

DFA的优化

- 集合上的等价关系与集合的划分
- DFA状态集合上的一个等价关系
- 状态集划分的算法-填表法
- 最优的DFA

DFA的优化

- 集合上的等价关系与集合的划分
- DFA状态集合上的一个等价关系
- 状态集划分的算法-填表法
- 最优的DFA

等价关系与集合划分

设 R 是集合 Q 上的一个等价关系，由 R 产生的所有等价类（或块）的集合构成 Q 的一个划分.

- 等价类 对任何 $a \in Q$，a 所在的等价类用 $[a]$ 表示，定义为

$$[a] = \{x \mid xRa\}$$

- 每一元素都属于唯一的等价类，即满足
  (1) 对任何 $a, b \in Q$，或者 $[a] = [b]$，或者 $[a] \cap [b] = \varnothing$
  (2) $\cup_{a \in Q} [a] = Q$

DFA的优化

- 集合上的等价关系与集合的划分
- DFA状态集合上的一个等价关系
- 状态集划分的算法-填表法
- 最优的DFA

DFA 状态集上的等价关系

设 *DFA* $D = (Q, \Sigma, \delta, q_0, F)$, 定义 Q 上的二元关系 R:

对任何 $p,q \in Q$, pRq *iff* $w \in \Sigma^*$,

$\delta^*(p,w) \in F \Leftrightarrow \delta^*(q,w) \in F$

结论：上述关系 R 是等价关系.

DFA 状态集上的等价关系

R 是等价关系

证明：

1. 自反性：对任何 $q \in Q$, qRq 成立；
2. 对称性：对任何 $p,q \in Q$, $pRq \rightarrow qRp$ 成立；
3. 传递性：对 $\forall p,q,r \in Q$, 设 pRq 和 qRr, 即 $\forall w \in \Sigma^*$, $\delta^*(p,w) \in F \Leftrightarrow \delta^*(q,w) \in F$ 和 $\delta^*(q,w) \in F \Leftrightarrow \delta^*(r,w) \in F$；
   故有 $\delta^*(p,w) \in F \Leftrightarrow \delta^*(r,w) \in F$ 成立.
   所以，qRr 成立

DFA 状态集上的等价关系

若 pRq, 称 p 和 q 等价. 若 p 和 q 不等价，则称 p 和 q 是可区别的(*distinguashable*)

关系 R 对应有限状态集 Q 的一个划分；该划分的每个块是 Q 的一个子集；

- 同一划分块中的所有状态之间都是相互等价的；
- 不同划分块的任何两个状态之间都是可区别的。

DFA 状态集上的等价关系

DFA 的优化

通过合并等价的(或不可区别的)状态.

如何计算上述划分？

DFA的优化

- 集合上的等价关系与集合的划分
- DFA状态集合上的一个等价关系
- 状态集划分的算法-填表法
- 最优的DFA

填 表 法

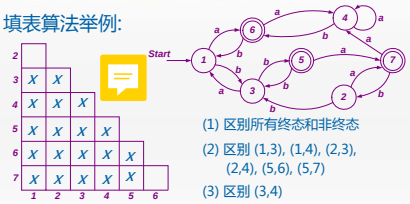
填表算法(*table-filling algorithm*)
基于如下递归地标记可区别的状态偶对的过程:

– 基础
若 p 为终态，q 为非终态，则 p 和 q 标记为可区别的；

– 归纳
设 p 和 q 已标记为可区别的. 若状态 r 和 s 通过输入符号 a 可分别转移到 p 和 q, 即 $\delta(r,a)=p$, $\delta(s,a)=q$, 则 r 和 s 也标记为可区别的；

填表法

填表算法举例:



(1) 区别所有终态和非终态

(2) 区别 (1,3), (1,4), (2,3), (2,4), (5,6), (5,7)

(3) 区别 (3,4)

划分结果：{1,2}, {3}, {4}, {5}, {6,7}

填表法

填表算法的正确性

如果两个状态没有被填表算法标记，则这两个状态一定是等价的。

证明: 反证法。 假定状态 p 和 q 没有被填表算法标记，但这两个状态不是等价的，即是可区别的。

设字符串 w 可用于区别状态 p 和 q, 即 $\delta^*(p,w)$ 和 $\delta^*(q,w)$ 两个状态中，一个是终态，一个是非终态. 不妨设 p 为终态, q 为非终态。

填表法

显然不可能有 $w=\varepsilon$, 否则, 状态 p 为终态, 而 q 为非终态，依填表算法, p 和 q 第一步就被标记.

设 $w=ax$, 并且 $\delta(p,a)=r$, $\delta(q,a)=s$, 则 r 和 s 也是可区别的, $\delta^*(r,x)$ 为终态, 而 $\delta^*(s,x)$ 为非终态. 但同样 r 和 s 没有被填表算法标记. 同样, $x\neq\varepsilon$.

因 w 的长度有限，该过程不可能一直下去, 终会产生矛盾.

DFA 的优化

- 集合上的等价关系与集合的划分
- DFA状态集合上的一个等价关系
- 状态集划分的算法-填表法
- 最优的DFA

DFA 的优化

步骤：

1. 删除从初态不可到达的状态及与其相关的边, 设所得到的 DFA 为 $A=(Q, \Sigma, \delta, q_0, F)$ ；
2. 使用填表算法找出所有等价的状态偶对；
3. 根据 2 的结果计算当前状态集合的划分块，每一划分块中的状态相互之间等价，而不同划分块中的状态之间都是可区别的. 包含状态 q 的划分块用 $[q]$ 表示.

DFA 的优化

步骤：

构造与 A 等价的 DFA

$$B = (Q_B, \Sigma, \delta_B, [q_0], F_B)$$

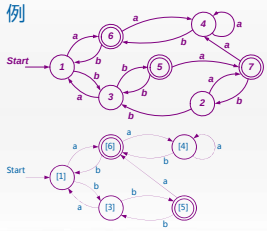
其中

$Q_B=\{[q] \mid q\in Q\}$, $F_B=\{[q] \mid q\in F\}$,

$\delta_B([q],a)=[\delta(q,a)]$

DFA 的优化

例



等价的状态偶对为：
{1, 2}, {6, 7}

划分结果：
{1, 2}, {3}, {4}, {5}, {6, 7}

新的状态集合：
[1], [3], [4], [5], [6]

最优的DFA

给定 *DFA A*, 用上述填表法构造与*A* 等价的*DFA*:

$M = (Q_m, \Sigma, \delta_m, q_{0m}, F_m)$

问题：

是否存在 一个状态数目比 *M* 还少的 *DFA N*, 它接受的语言与 *A* 和 *M* 完全一样?

最优的DFA

若存在 *DFA*

$N = (Q_n, \Sigma, \delta_n, q_{0n}, F_n)$

$L(N) = L(M),\ |Q_n| < |Q_m|$

假定: *M* 和 *N* 之间没有重名的状态.

M 和 *N* 的每一状态都是从初态可达的.

最优的DFA

将 *M* 和 *N* 相并，构造并自动机 “M∪N”，:

$Q_n \cup Q_m$

“$\delta_n \cup \delta_m$”

$q_0 = q_{0n}$ or q_{0m}

$F = \{F_n, F_m\}$

最优的DFA

对 *M* 和 *N* 相并后的自动机， 可以得出:

a) *M* 和 *N* 的初态是不可区分的;

b) 若 *p* 和 *q* 是不可区分的, 则对于 [redacted] 入符号,*p* 和 *q* 的后继状态之间也是不可区分的;

c) *M* 的任一状态至少与 *N* 的一个状态是不可区分的.

因$|Q_n| < |Q_m|$, 则 *M* 中存在两个状态与 *N* 中的同一个状态不可区分. 根据传递性， *M* 的这两个状态是不可区分的, 这与 *M* 的构造过程矛盾.

最优的DFA

结论：

对任何 *DFA A*, *M*是用填表法构造的与 *A* 等价的*DFA*; 则*M* 是与A等价的DFA中状态数目最少的*DFA* .


Symbolic Algorithms for Language Equivalence and Kleene Algebra with Tests

Dexter Kozen

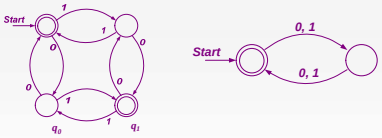
Abstract

1. Introduction

例

最小化下面语言的 *DFA*:

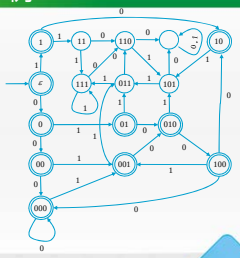
$L = \{\, w \mid w$ 中0 、1数目的奇偶性相同 $\}$



例

最小化 *DFA*:

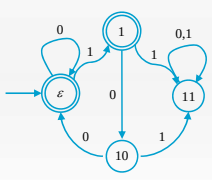
L={w∈{0,1}*|w中长度不超过3的任意 子串最多包含一个1}

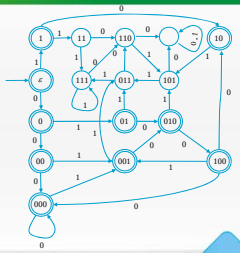


	ε	0	1	00	01	10	000	001	010	100	011	101	110	111	11
0															
1	×	×													
00			×												
01	×	×		×											
10	×	×		×											
000			×												
001	×	×		×											
010	×	×		×											
100			×												
011	×	×	×	×	×	×	×	×	×	×					
101	×	×	×	×	×	×	×	×	×	×					
110	×	×	×	×	×	×	×	×	×	×					
111	×	×	×	×	×	×	×	×	×	×					
11	×	×	×	×	×	×	×	×	×	×					

例

最小化 *DFA*:





上下文无关文法和推导

- 上下文无关文法
- 规约和推导
- 语法分析树
- 规约，推导和语法分析树之间的关系
- 上下文无关语言

上下文无关文法和推导

- 上下文无关文法
- 规约和推导
- 语法分析树
- 规约，推导和语法分析树之间的关系
- 上下文无关语言

上下文无关文法

上下文无关文法(CFG)的定义

上下文无关文法 G = (V, T, S, P)

变量(非终结符)
终 结 符
开 始 变 量
产 生 式

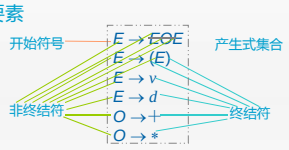
满足: $V \cap T = \varnothing$, $S \in V$

产生式: $A \to \alpha$, 其中 $A \in V$, $\alpha \in (V \cup T)^*$

上下文无关文法

CFG的四个基本要素

1. 终结符的集合
2. 非终结符的集合
3. 开始符号
4. 产生式的集合



形如：
<head> → <body>

上下文无关文法

- 例 CFG G_{exp} = ({E,O}, { (,), +, *, v, d }, E , P).
  其中产式集合 P 为:

  E → EOE | (E) | v | d
  O → + | *

  E → EOE
  E → (E)
  E → v
  E → d
  O → +
  O → *

- *Backus-Naur Form*（巴科斯-瑙尔范式）
  计算机语言语法定义中的形式
  许多场合下采用::= ，而不是→.

例

上下文无关文法 G:

$$S \to aSb$$

$$S \to \varepsilon$$

$$S \Rightarrow aSb \Rightarrow ab$$

可以推导出句子 *ab*. 该文法的语言为：

$$L = \{a^n b^n \mid n \geq 0\}$$

上下文无关文法和推导

- 上下文无关文法
- 规约和推导
- 语法分析树
- 规约，推导和语法分析树之间的关系
- 上下文无关语言

规约和推导

- 字符串是否属于文法的语言的方法：归约和推理
  - 自下而上地推理，称为递归推理，递归推理的过程也称为归约.
  - 自上而下地推理，称为推导.

规约和推导

- 归约过程
  将产生式的右部替换为产生式的左部.
- 推导过程
  将产生式的左部替换为产生式的右部.

规约和推导

- 归约过程举例
  对于CFG $G_{exp}=(\{E,O\},\{(,),+,*,v,d\},E,P)$，$P$为
  (1) $E\to EOE$
  (2) $E\to(E)$
  (3) $E\to v$
  (4) $E\to d$
  (5) $O\to +$
  (6) $O\to *$
  递归推理出字符串 $(v+d)*d$ 的一个归约过程为：

$$(v+d)*d \xrightarrow{(6)} (v+d)Od \xrightarrow{(5)} (vOd)Od \xrightarrow{(4)} (vOE)Od \xrightarrow{(4)} (vOE)OE \xrightarrow{(3)} (EOE)OE \xrightarrow{(1)} (E)OE \xrightarrow{(2)} EOE \xrightarrow{(1)} E$$

规约和推导

- 推导过程举例
  对于CFG $G_{exp}=(\{E,O\},\{(,),+,*,v,d\},E,P)$，$P$为
  (1) $E\to EOE$
  (2) $E\to(E)$
  (3) $E\to v$
  (4) $E\to d$
  (5) $O\to +$
  (6) $O\to *$
  从开始符到字符串 $(v+d)*d$ 的一个推导过程为：

$$E \xrightarrow{(1)} EOE \xrightarrow{(2)} (E)OE \xrightarrow{(1)} (EOE)OE \xrightarrow{(3)} (vOE)OE \xrightarrow{(4)} (vOE)Od \xrightarrow{(4)} (vOd)Od \xrightarrow{(5)} (v+d)Od \xrightarrow{(6)} (v+d)*d$$

规约和推导

- 推导关系
  对于 CFG $G=(V,T,S,P)$, 上述推导过程可用关系 $\Rightarrow$ 描述.
  设 $\alpha,\beta\in(V\cup T)^*$, $A\to\gamma$ 是一个产生式，则定义

$$\alpha A\beta \underset{G}{\Rightarrow} \alpha\gamma\beta .$$

若 G 在上下文中是明确的，则简记为:

$$\alpha A\beta \Rightarrow \alpha\gamma\beta .$$

规约和推导

- 推导传递闭包

上述推导的传递闭包 $\overset{*}{\underset{G}{\Rightarrow}}$，可归纳定义如下：

基础 对任何 $\alpha\in(V\cup T)^*$，满足 $\alpha \overset{*}{\underset{G}{\Rightarrow}} \alpha$.

归纳 设 $\alpha,\beta,\gamma\in(V\cup T)^*$，若 $\alpha \overset{*}{\underset{G}{\Rightarrow}} \beta$，$\beta \underset{G}{\Rightarrow} \gamma$ 成立，则

$$\alpha \overset{*}{\underset{G}{\Rightarrow}} \gamma .$$

最左推导

若推导过程的每一步总是替换出现在最左边的非终结符，则称这样的推导为最左推导.最左推导关系用$\overset{}{\Rightarrow}$表示，其传递闭包用$\overset{+}{\underset{lm}{\Rightarrow}}$表示.

对于文法G_{exp}，$(v+d)*d$的一个最左推导为：

$$E \underset{lm}{\overset{(1)}{\Rightarrow}} EOE \underset{lm}{\overset{(2)}{\Rightarrow}} (E)OE \underset{lm}{\overset{(1)}{\Rightarrow}} (EOE)OE \underset{lm}{\overset{(3)}{\Rightarrow}} (vOE)OE$$
$$\underset{lm}{\overset{(5)}{\Rightarrow}} (v+E)OE \underset{lm}{\overset{(4)}{\Rightarrow}} (v+d)OE \underset{lm}{\overset{(6)}{\Rightarrow}} (v+d)*E \underset{lm}{\overset{(4)}{\Rightarrow}} (v+d)*d$$

$E \to EOE$
$E \to (E)$
$E \to v$
$E \to d$
$O \to +$
$O \to *$

最右推导

若推导过程的每一步总是替换出现在最右边的非终结符，则称这样的推导为最右推导. 最右推导关系用$\overset{}{\Rightarrow}$表示，其传递闭包用$\overset{+}{\underset{rm}{\Rightarrow}}$表示.

对于文法G_{exp}，$(v+d)*d$的一个最右推导为：

$$E \underset{rm}{\overset{(1)}{\Rightarrow}} EOE \underset{rm}{\overset{(4)}{\Rightarrow}} EOd \underset{rm}{\overset{(6)}{\Rightarrow}} E*d \underset{rm}{\overset{(2)}{\Rightarrow}} (E)*d \underset{rm}{\overset{(1)}{\Rightarrow}} (EOE)*d$$
$$\underset{rm}{\overset{(4)}{\Rightarrow}} (EOd)*d \underset{rm}{\overset{(5)}{\Rightarrow}} (E+d)*d \underset{rm}{\overset{(3)}{\Rightarrow}} (v+d)*d$$

$E \to EOE$
$E \to (E)$
$E \to v$
$E \to d$
$O \to +$
$O \to *$

句型

设 CFG $G=(V, T, S, P)$, $\alpha \in (V \cup T)^*$ 为 G 的一个句型.

若 $S \overset{*}{\underset{lm}{\Rightarrow}} \alpha$，则 α 是一个左句型;

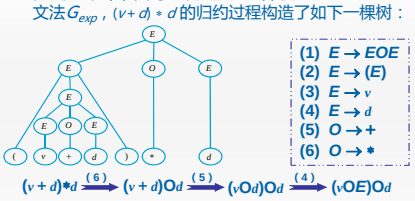
若 $S \overset{*}{\underset{rm}{\Rightarrow}} \alpha$，则 α 是一个右句型.

若句型 $\alpha \in T^*$，则称 α 为一个句子.

上下文无关文法和推导

- 上下文无关文法
- 规约和推导
- 语法分析树
- 规约，推导和语法分析树之间的关系
- 上下文无关语言

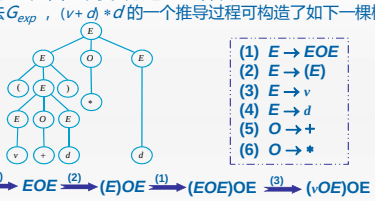
- 归约过程自下而上构造了一棵树
  文法G_{exp}，$(v+d)*d$的归约过程构造了如下一棵树：



(1) $E \to EOE$
(2) $E \to (E)$
(3) $E \to v$
(4) $E \to d$
(5) $O \to +$
(6) $O \to *$

$(v+d)*d \xrightarrow{(6)} (v+d)Od \xrightarrow{(5)} (vOd)Od \xrightarrow{(4)} (vOE)Od$
$\xrightarrow{(4)} (vOE)OE \xrightarrow{(3)} (EOE)OE \xrightarrow{(1)} (E)OE \xrightarrow{(2)} EOE \xrightarrow{(1)} E$

- 推导过程自上而下构造了一棵树
  文法G_{exp}，$(v+d)*d$的一个推导过程可构造了如下一棵树：



(1) $E \to EOE$
(2) $E \to (E)$
(3) $E \to v$
(4) $E \to d$
(5) $O \to +$
(6) $O \to *$

$E \xrightarrow{(1)} EOE \xrightarrow{(2)} (E)OE \xrightarrow{(1)} (EOE)OE \xrightarrow{(3)} (vOE)OE$
$\xrightarrow{(4)} (vOE)Od \xrightarrow{(4)} (vOd)Od \xrightarrow{(5)} (v+d)Od \xrightarrow{(6)} (v+d)*d$

语法分析树

对于 *CFG* $G=(V, T, S, P)$, 语法分析树满足下列条件：

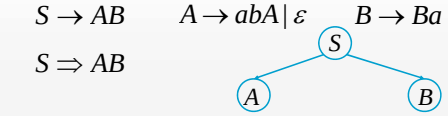
1. 每个内部结点由一个非终结符标记.
2. 每个叶结点或为一个变量，或为一个终结符，或由 ε 来标记. 当标记为 ε 时，它是其父结点唯一的子节点.
3. 若一个内部结点标记为 A, 而其子节点从左至右分别标记为 $X_1, X_2, \ldots, X_k$, 则 $A \to X_1X_2\ldots X_k$ 是 P 中的一个产生式.

注意：只有 $k=1$ 时上述 X_i 才有可能为 ε, 此时结点 A 只有唯一的子节点，且 $A \to \varepsilon$ 是 P 中的一个产生式.

语法分析树

$S \to AB \quad A \to abA \mid \varepsilon \quad B \to Ba \mid b$

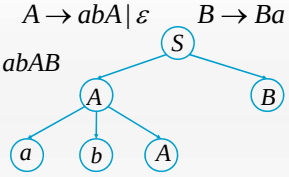
$S \Rightarrow AB$



语法分析树

$S \to AB \quad A \to abA \mid \varepsilon \quad B \to Ba \mid b$

$S \Rightarrow AB \Rightarrow abAB$

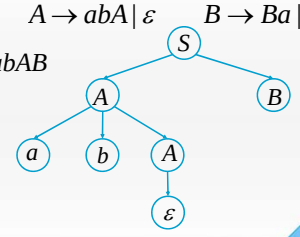


语法分析树

$S \to AB \quad A \to abA \mid \varepsilon \quad B \to Ba \mid b$

$S \Rightarrow AB \Rightarrow abAB$

$\Rightarrow ab\varepsilon B$

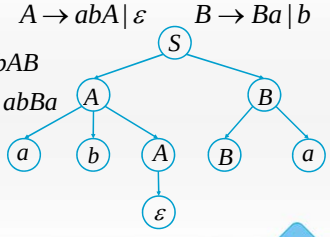


语法分析树

$S \to AB \quad A \to abA \mid \varepsilon \quad B \to Ba \mid b$

$S \Rightarrow AB \Rightarrow abAB$

$\Rightarrow ab\varepsilon B \Rightarrow abBa$



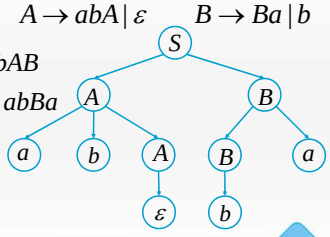
语法分析树

$S \to AB \quad A \to abA \mid \varepsilon \quad B \to Ba \mid b$

$S \Rightarrow AB \Rightarrow abAB$

$\Rightarrow ab\varepsilon B \Rightarrow abBa$

$\Rightarrow abba$



语法分析树

分析树的产物

设 *CFG* $G=(V,T,S,P)$. 将语法分析树的每个叶结点按照从左至右的次序连接起来，得到一个 $(V\cup T)^*$ 中的字符串，称为该语法树的产物.

G 的每个句型都是某个根结点为 S 的分析树的产物；这些分析树中，有些树的产物为句子，所有这些分析树的产物构成了 G 的语言.

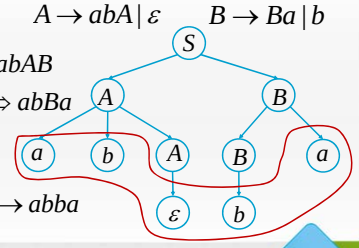
语法分析树

$S\to AB \quad A\to abA\mid\varepsilon \quad B\to Ba\mid b$

$S\Rightarrow AB\Rightarrow abAB$

$\Rightarrow ab\varepsilon B\Rightarrow abBa$

$\Rightarrow abba$



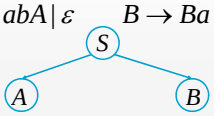
产物 :$ab\varepsilon ba\to abba$

子树

$S\to AB \quad A\to abA\mid\varepsilon \quad B\to Ba\mid b$

$S\Rightarrow AB$

子树

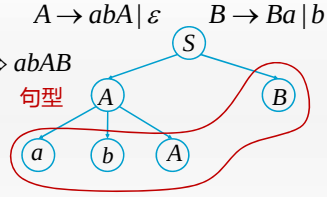


语法分析树

$S\to AB \quad A\to abA\mid\varepsilon \quad B\to Ba\mid b$

$S\Rightarrow AB\Rightarrow abAB$

句型



产物 :$abAB$

上下文无关文法和推导

- 上下文无关文法
- 规约和推导
- 语法分析树
- 规约，推导和语法分析树之间的关系
- 上下文无关语言

规约、推导和语法分析树

设 *CFG* $G=(V,T,S,P)$. 以下命题是相互等价的：

(1) 字符串 $w\in T^*$ 可以归约到非终结符 A；

(2) $A\overset{*}{\Rightarrow}w$；

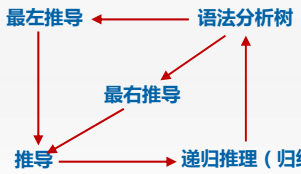
(3) $A\overset{*}{\underset{lm}{\Rightarrow}}w$；

(4) $A\overset{*}{\underset{rm}{\Rightarrow}}w$；

(5) 存在一棵根结点为 A 的分析树，其产物为 w.

规约、推导和语法分析树

证明思路：



规约、推导和语法分析树

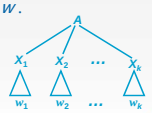
- 从归约到分析树

设 *CFG* $G = (V, T, S, P)$. 若字符串 $w \in T^*$ 可以归约到非终结符 A，则存在一棵根结点为 A 的分析树，其产物为 w.

证明思路：从 w 归约到 A 的步数作归纳.

基础：步数为 1. 一定有产生式 $A \to w$. 存在右上图所示的分析树.

归纳：设步数大于 1，且最后一步归约使用了产生式: $A \to X_1X_2...X_k$. 存在右下图所示的分析树.



规约、推导和语法分析树

- 从分析树到推导

设 *CFG* $G = (V, T, S, P)$. 如果存在一棵根结点为 A 的分析树，其产物为字符串 $w \in T^*$，则 $A \overset{+}{\Rightarrow} w$，$A \overset{+}{\underset{lm}{\Rightarrow}} w$，$A \overset{+}{\underset{rm}{\Rightarrow}} w$.

只证明 $A \overset{+}{\underset{lm}{\Rightarrow}} w$，也证明了 $A \overset{+}{\Rightarrow} w$.

- 从分析树到最左推导

证明思路： 对分析树的高度归纳证明 $A \overset{*}{\underset{lm}{\Rightarrow}} w$.

规约、推导和语法分析树

基础：高度为 1. 分析树一定如右上图所示，必定有产生式 $A \to w$. 因此， $A \underset{lm}{\Rightarrow} w$.

归纳：高度>1 的分析树一定如右图所示，必定有产生式 $A \to X_1X_2...X_k$. 存在 $w_1, w_2, ..., w_k$，w_i 是 X_i 子树的产物或 $w_i = X_i$ $(1 \le i \le k)$，且 $w = w_1w_2...w_k$，由归纳假设：

$X_i \overset{*}{\underset{lm}{\Rightarrow}} w_i \ (1 \le i \le k)$.

在此基础上易证得 $A \overset{*}{\underset{lm}{\Rightarrow}} w$.

规约、推导和语法分析树

- 从推导到归约

设 *CFG* $G = (V, T, S, P)$.
如果对于非终结符 A 和字符串 $w \in T^*$, $A \overset{+}{\Rightarrow} w$,
则 w 可以归约到 A.

证明思路:
对推导 $A \overset{*}{\Rightarrow} w$ 的步数用归纳法.

基础
步数为 1.
一定有产生式 $A \to w$. w 可以归约到 A.

规约、推导和语法分析树

归纳: 设步数大于 1.
第一步使用了产生式 $A \to X_1X_2...X_k$. 该推导：

$A \Rightarrow X_1X_2...X_k \overset{*}{\Rightarrow} w$.

可以将 w 分成 $w = w_1w_2...w_k$，其中

(*a*) 若 X_i 为终结符，则 $w_i = X_i$.
(*b*) 若 X_i 为非终结符，则 $X_i \overset{+}{\Rightarrow} w_i$.

由归纳假设，w_i 可以归约到 X_i.

这样，w_i 或者为 X_i，或者可以归约到 X_i，使用产生式 $A \to X_1X_2...X_k$，得出 w 可以归约到 A.

上下文无关文法和推导

- 上下文无关文法
- 规约和推导
- 语法分析树
- 规约，推导和语法分析树之间的关系
- 上下文无关语言

上 下 文 无 关 语 言

- 上下文无关文法的语言

  设 *CFG* $G = (V, T, S, P)$，定义 G 的语言为

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

- 上下文无关语言

  如果一个语言 L 是某个 *CFG* G 的语言，即

  $L(G) = L$

  则 L 是上下文无关语言.

上 下 文 无 关 语 言

证明给定语言 L 是某个文法 G 的语言

一般步骤

- 如果 $w \in L$ 那么 $w \in L(G)$.
- 如果 $w \in L(G)$ 那么 $w \in L$.

对于前者，多数情况下可以归纳于 w 的长度 $|w|$；

对于后者，一般情况下可以归纳于推导 w 的步数.

上 下 文 无 关 语 言

例 回文(Palindromes)的文法 G_{pal} 为:

$G_{pal} = (\{S\}, \{0,1\}, S, P)$,

P 为产生式：

$S \rightarrow 0S0 \mid 1S1 \mid 0 \mid 1 \mid \varepsilon$

则 $L(G_{pal})$ 是字母为{0,1}上的回文集合：

$$L = \{w \mid w \in \{0,1\}^*, w = w^R\}$$

上 下 文 无 关 语 言

证明:

(1) 设 w 是一个回文，则 $w \in L(G_{pal})$ 。

对 w 的长度 $|w|$ 作归纳。

(2) 设 $w \in L(G_{pal})$ ，则 w 是一个回文。

对推导 w 的步数作归纳。

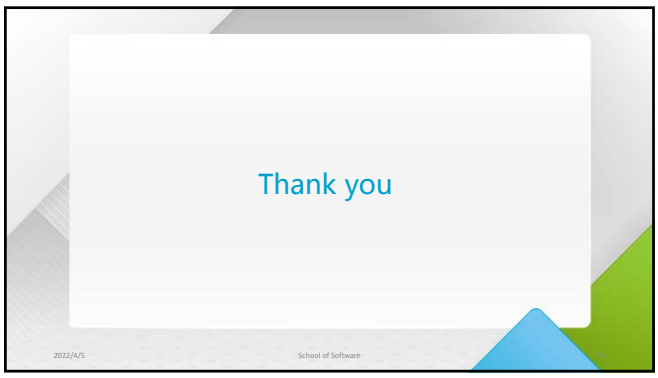
课后练习

✧ **必做题:**

P-164 *Ex.4.4.2*
P-180 !*Ex.5.1.1(c)*
P-180 *Ex.5.1.2(c)*
P-180 **! Ex.5.1.5*
P-180 !*Ex.5.1.6 (b)*
P-191 *Ex.5.2.1*

思考题:

P-180 !*Ex.5.1.1(d)*
P-180 !*Ex.5.1.3*
P-191 !*Ex.5.2.2*


Thank you
2022/4/5
School of Software