第 4 章　上下文无关语言

(Part 1 of 2)

王　鑫

wangx@tju.edu.cn

天津大学　智能与计算学部

Outline

Context-Free Languages

regular languages 正则语言

- finite automata: DFA / NFA
- regular expressions

some simple languages, such as $\{0^n1^n \mid n \geq 0\}$, are ***not*** regular languages.

Context-Free Languages

regular languages 正则语言

- finite automata: DFA / NFA
- regular expressions

some simple languages, such as $\{0^n1^n \mid n \geq 0\}$, are ***not*** regular languages.

context-free languages 上下文无关语言

- pushdown automata 下推自动机

- first used in the study of human languages
- in the specification and compilation of programming languages
  - ***parser***
  - the construction of a parser from a context-free grammar

Outline

Context-Free Grammars 上下文无关文法

例 (context-free grammar: G_1)

$$A \to 0A1$$
$$A \to B$$
$$B \to \#$$

A grammar

- substitution rules, ***productions*** 产生式
- ***variable*** 变元
- ***terminals*** 终结符
- ***start variable*** 起始变元

Context-Free Grammars 上下文无关文法

例 (context-free grammar: G_1)

$$A \to 0A1$$

$$A \to B$$

$$B \to \#$$

G_1 generates the string $000\#111$

Derivation 推导

- The sequence of substitutions to obtain a string is called a ***derivation***
- $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$

Context-Free Grammars 上下文无关文法

Derivation 推导

- $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$

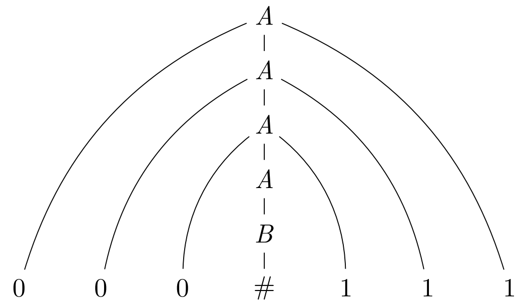
***parse tree* 语法分析树**

Context-Free Grammars 上下文无关文法

Derivation 推导

- $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$

parse tree **语法分析树**



Context-Free Grammars 上下文无关文法

Derivation 推导

- $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$

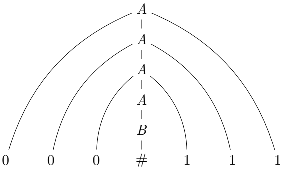
***parse tree* 语法分析树**

Context-Free Grammars 上下文无关文法

Derivation 推导

- $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$

parse tree **语法分析树**



language of the grammar **文法的语言**

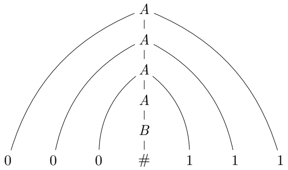
- $L(G_1)$: the language of grammar G_1

Context-Free Grammars 上下文无关文法

Derivation 推导

- $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$

***parse tree* 语法分析树**



***language of the grammar* 文法的语言**

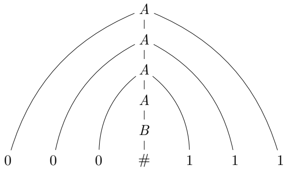
- $L(G_1)$: the language of grammar G_1
- What is $L(G_1)$?

Context-Free Grammars 上下文无关文法

Derivation 推导

- $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$

***parse tree* 语法分析树**



***language of the grammar* 文法的语言**

- $L(G_1)$: the language of grammar G_1
- What is $L(G_1)$? $\{0^n\#1^n \mid n \geq 0\}$

Context-Free Grammars 上下文无关文法

language of the grammar **文法的语言**

- $L(G_1)$: the language of grammar G_1
- What is $L(G_1)$? $\{0^n\#1^n \mid n \geq 0\}$

context-free languages (CFL) **上下文无关语言**

- any language that can be generated by some context-free grammar

Abbreviation

$A \to 0A1$ and $A \to B$

$A \to 0A1 \mid B$

Context-Free Grammars 上下文无关文法

例 (context-free grammar G_2)

<SENTENCE> → <NOUN-PHRASE><VERB-PHRASE>

<NOUN-PHRASE> → <CMPLX-NOUN> | <CMPLX-NOUN><PREP-PHRASE>

<VERB-PHRASE> → <CMPLX-VERB> | <CMPLX-VERB><PREP-PHRASE>

<PREP-PHRASE> → <PREP><CMPLX-NOUN>

<CMPLX-NOUN> → <ARTICLE><NOUN>

<CMPLX-VERB> → <VERB> | <VERB><NOUN-PHRASE>

<ARTICLE> → a | the

<NOUN> → boy | girl | flower

<VERB> → touches | likes | sees

<PREP> → with

a boy sees

the boy sees a flower

a girl with a flower likes the boy

Context-Free Grammars 上下文无关文法

Derivation

<SENTENCE> ⇒ <NOUN-PHRASE><VERB-PHRASE>
⇒ <CMPLX-NOUN><VERB-PHRASE>
⇒ <ARTICLE><NOUN><VERB-PHRASE>
⇒ a <NOUN><VERB-PHRASE>
⇒ a boy <VERB-PHRASE>
⇒ a boy <CMPLX-VERB>
⇒ a boy <VERB>
⇒ a boy sees

Formal Definition of a Context-Free Grammar

定义 (CFG (上下文无关文法))

Formal Definition of a Context-Free Grammar

定义 (CFG (上下文无关文法))

A ***context-free grammar*** (CFG) is a 4-tuple (V, Σ, R, S), where

Formal Definition of a Context-Free Grammar

定义 (CFG (上下文无关文法))

A ***context-free grammar*** (CFG) is a 4-tuple (V, Σ, R, S), where

1. V is a finite set called the ***variables***,

Formal Definition of a Context-Free Grammar

定义 (CFG (上下文无关文法))

A ***context-free grammar*** (CFG) is a 4-tuple (V, Σ, R, S), where

1. V is a finite set called the ***variables***,
2. Σ is a finite set, disjoint from V, called the ***terminals***,

Formal Definition of a Context-Free Grammar

定义 (CFG (上下文无关文法))

A ***context-free grammar*** (CFG) is a 4-tuple (V, Σ, R, S), where

1. V is a finite set called the ***variables***,
2. Σ is a finite set, disjoint from V, called the ***terminals***,
3. R is the finite set of ***rules***, with each rule being a variable and a string of variables and terminals, and

Formal Definition of a Context-Free Grammar

定义 (CFG (上下文无关文法))

A ***context-free grammar*** (CFG) is a 4-tuple (V, Σ, R, S), where

1. V is a finite set called the ***variables***,
2. Σ is a finite set, disjoint from V, called the ***terminals***,
3. R is the finite set of ***rules***, with each rule being a variable and a string of variables and terminals, and
4. $S \in V$ is the ***start variable***.

Formal Definition of a Context-Free Grammar

u, v, w are strings of variables and terminals,

$A \rightarrow w$ is a rule of the grammar

Formal Definition of a Context-Free Grammar

u, v, w are strings of variables and terminals,

$A \rightarrow w$ is a rule of the grammar

- $uAv \Rightarrow uwv$: uAv ***yields*** uwv 直接推导

Formal Definition of a Context-Free Grammar

u, v, w are strings of variables and terminals,

$A \rightarrow w$ is a rule of the grammar

- $uAv \Rightarrow uwv$: uAv ***yields*** uwv **直接推导**
- $u \overset{*}{\Rightarrow} v$: u ***derives*** v **推导**

Formal Definition of a Context-Free Grammar

u, v, w are strings of variables and terminals,

$A \rightarrow w$ is a rule of the grammar

- $uAv \Rightarrow uwv$: uAv ***yields*** uwv **直接推导**
- $u \overset{*}{\Rightarrow} v$: u ***derives*** v **推导**
  - if $u = v$ or

Formal Definition of a Context-Free Grammar

u, v, w are strings of variables and terminals,

$A \to w$ is a rule of the grammar

- $uAv \Rightarrow uwv$: uAv ***yields*** uwv **直接推导**
- $u \overset{*}{\Rightarrow} v$: u ***derives*** v **推导**
  - if $u = v$ or
  - if a sequence $u_1, u_2, \ldots, u_k$ exists for $k \geq 0$ and

    $u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \cdots \Rightarrow u_k \Rightarrow v$

Formal Definition of a Context-Free Grammar

u, v, w are strings of variables and terminals,

$A \rightarrow w$ is a rule of the grammar

- $uAv \Rightarrow uwv$: uAv ***yields*** uwv **直接推导**
- $u \overset{*}{\Rightarrow} v$: u ***derives*** v **推导**
  - if $u = v$ or
  - if a sequence $u_1, u_2, \ldots, u_k$ exists for $k \geq 0$ and

    $u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \cdots \Rightarrow u_k \Rightarrow v$

The ***language of the grammar*** is $\{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\}$

Formal Definition of a Context-Free Grammar

例 (context-free grammar: G_1)

$G_1 = (V, \Sigma, R, S)$

- $V = \{A, B\}$
- $\Sigma = \{0, 1, \#\}$
- $S = A$
- R:

$$A \to 0A1$$
$$A \to B$$
$$B \to \#$$

Formal Definition of a Context-Free Grammar

例 (context-free grammar: G_3)

$G_3 = (\{S\}, \{a, b\}, R, S)$

- $$S \to aSb \mid SS \mid \varepsilon$$

Formal Definition of a Context-Free Grammar

例 (context-free grammar: G_4)

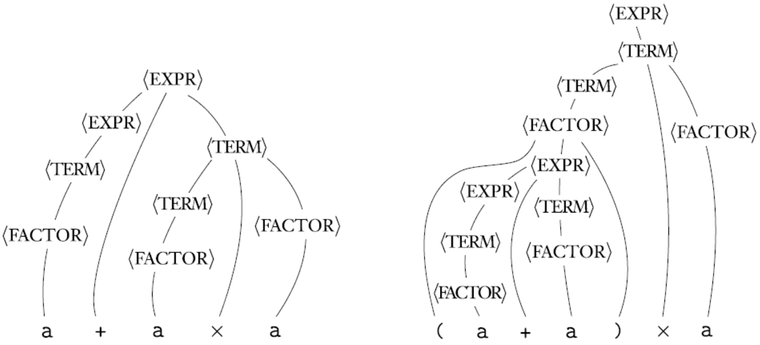
$G_3 = (V, \Sigma, R, <\text{EXPR}>)$

- $V = \{<\text{EXPR}>, <\text{TERM}>, <\text{FACTOR}>\}$
- $\Sigma = \{a, +, \times, (,)\}$
- $$<\text{EXPR}> \to <\text{EXPR}> + <\text{TERM}> \mid <\text{TERM}>$$
  $$<\text{TERM}> \to <\text{TERM}> \times <\text{FACTOR}> \mid <\text{FACTOR}>$$
  $$<\text{FACTOR}> \to (<\text{EXPR}>) \mid a$$

Two strings generated with grammar G_4

- $a + a \times a$
- $(a + a) \times a$

Formal Definition of a Context-Free Grammar



Parse tree for the strings $a + a \times a$ and $(a + a) \times a$

Designing Context-Free Grammars

1. Many CFLs are the union of simpler CFLs

Designing Context-Free Grammars

1. Many CFLs are the union of simpler CFLs

例

$\{0^n1^n \mid n \geq 0\} \cup \{1^n0^n \mid n \geq 0\}$

Designing Context-Free Grammars

1. Many CFLs are the union of simpler CFLs

例

$\{0^n1^n \mid n \geq 0\} \cup \{1^n0^n \mid n \geq 0\}$

- $\{0^n1^n \mid n \geq 0\}$: $S_1 \to 0S_11 \mid \varepsilon$
- $\{1^n0^n \mid n \geq 0\}$: $S_2 \to 1S_20 \mid \varepsilon$

Designing Context-Free Grammars

1. Many CFLs are the union of simpler CFLs

例

$\{0^n1^n \mid n \geq 0\} \cup \{1^n0^n \mid n \geq 0\}$

- $\{0^n1^n \mid n \geq 0\}$: $S_1 \to 0S_11 \mid \varepsilon$
- $\{1^n0^n \mid n \geq 0\}$: $S_2 \to 1S_20 \mid \varepsilon$

$$S \to S_1 \mid S_2$$
$$S_1 \to 0S_11 \mid \varepsilon$$
$$S_2 \to 1S_20 \mid \varepsilon$$

Designing Context-Free Grammars

2. Constructing a CFG for a language that happens to be regular

Designing Context-Free Grammars

2. Constructing a CFG for a language that happens to be regular

Convert any DFA into an equivalent CFG as follows

- Make a variable R_i for each state q_i of the DFA
- Add the rule $R_i \to aR_j$ to the CFG if $\delta(q_i, a) = q_j$ is a transition in the DFA
- Add the rule $R_i \to \varepsilon$ if q_i is an accept state of the DFA
- Make R_0 the start variable of the grammar, where q_0 is the start state of the machine

Designing Context-Free Grammars

3. Certain context-free languages contain strings with two substrings

- $\{0^n 1^n \mid n \geq 0\}$
- $R \rightarrow uRv$

Designing Context-Free Grammars

3. Certain context-free languages contain strings with two substrings
   - $\{0^n1^n \mid n \geq 0\}$
   - $R \rightarrow uRv$
4. The strings may contain certain structures that appear recursively as part of other (or the same) structures
   - the grammar that generates arithmetic expressions

Ambiguity 二义性

Sometimes a grammar can generate the same string in several different ways

Ambiguity 二义性

Sometimes a grammar can generate the same string in several different ways

Such a string will have several different parse trees and thus several different meanings

Ambiguity 二义性

Sometimes a grammar can generate the same string in several different ways

Such a string will have several different parse trees and thus several different meanings

- If a grammar generates the same string in several different ways, we say that the string is derived ***ambiguously*** in that grammar
- If a grammar generates some string ambiguously, we say that the grammar is ***ambiguous***

Ambiguity

例 (Grammar G_5)

$$\langle\text{EXPR}\rangle \to \langle\text{EXPR}\rangle + \langle\text{EXPR}\rangle \mid \langle\text{EXPR}\rangle \times \langle\text{EXPR}\rangle \mid (\langle\text{EXPR}\rangle) \mid \text{a}$$

Ambiguity

例 (Grammar G_5)

$$<\text{EXPR}>\rightarrow<\text{EXPR}>+<\text{EXPR}> \mid <\text{EXPR}>\times<\text{EXPR}> \mid (<\text{EXPR}>) \mid \text{a}$$

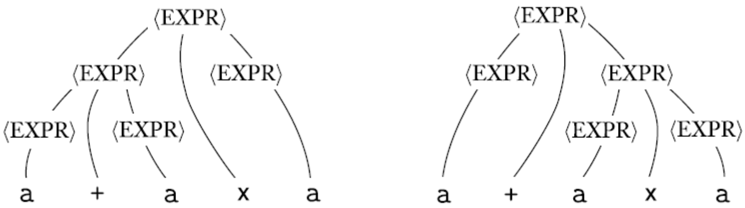
This grammar generates the string a+a×a ***ambiguously***

Ambiguity

例 (Grammar G_5)

<EXPR>→<EXPR>+<EXPR> | <EXPR>×<EXPR> | (<EXPR>) | a

This grammar generates the string a+a×a ***ambiguously***



The two parse trees for the string a+a×a in grammar G_5

Ambiguity

例 (Grammar G_5)

<EXPR>→<EXPR>+<EXPR> | <EXPR>×<EXPR> | (<EXPR>) | a

- This grammar doesn't capture the usual precedence relations and so may group the $+$ before the $\times$ or vice versa.
- Grammar G_4 generates exactly the same language, but every generated string has a unique parse tree.

Ambiguity

例 (Grammar G_5)

<EXPR>→<EXPR>+<EXPR> | <EXPR>×<EXPR> | (<EXPR>) | a

- This grammar doesn't capture the usual precedence relations and so may group the $+$ before the $\times$ or vice versa.
- Grammar G_4 generates exactly the same language, but every generated string has a unique parse tree.
- G_4 is unambiguous.
- G_5 is ambiguous.

Ambiguity

例 (Grammar G_5)

<EXPR>→<EXPR>+<EXPR> | <EXPR>×<EXPR> | (<EXPR>) | a

- This grammar doesn't capture the usual precedence relations and so may group the $+$ before the $\times$ or vice versa.
- Grammar G_4 generates exactly the same language, but every generated string has a unique parse tree.
- G_4 is unambiguous.
- G_5 is ambiguous.

What about the grammar G_2? Is it ambiguous?

the girl touches the boy with the flower

Ambiguity

例 (Grammar G_5)

<EXPR>→<EXPR>+<EXPR> | <EXPR>×<EXPR> | (<EXPR>) | a

- This grammar doesn't capture the usual precedence relations and so may group the $+$ before the $\times$ or vice versa.
- Grammar G_4 generates exactly the same language, but every generated string has a unique parse tree.

Ambiguity

例 (Grammar G_5)

<EXPR>→<EXPR>+<EXPR> | <EXPR>×<EXPR> | (<EXPR>) | a

- This grammar doesn't capture the usual precedence relations and so may group the $+$ before the $\times$ or vice versa.
- Grammar G_4 generates exactly the same language, but every generated string has a unique parse tree.
- G_4 is unambiguous.
- G_5 is ambiguous.

Ambiguity

例 (Grammar G_5)

<EXPR>→<EXPR>+<EXPR> | <EXPR>×<EXPR> | (<EXPR>) | a

- This grammar doesn't capture the usual precedence relations and so may group the $+$ before the $\times$ or vice versa.
- Grammar G_4 generates exactly the same language, but every generated string has a unique parse tree.
- G_4 is unambiguous.
- G_5 is ambiguous.

What about the grammar G_2? Is it ambiguous?

the girl touches the boy with the flower

Ambiguity

定义 (leftmost derivation (最左推导))

A derivation of a string w in a grammar G is a ***leftmost derivation*** if at every step the leftmost remaining variable is the one replaced.

Ambiguity

定义 (leftmost derivation (最左推导))

A derivation of a string w in a grammar G is a ***leftmost derivation*** if at every step the leftmost remaining variable is the one replaced.

定义 (ambiguity (二义性))

- A string w is derived ***ambiguously*** in context-free grammar G if it has two or more different leftmost derivations.
- Grammar G is ***ambiguous*** if it generates some string ambiguously.

Ambiguity

定义 (leftmost derivation (最左推导))

A derivation of a string w in a grammar G is a ***leftmost derivation*** if at every step the leftmost remaining variable is the one replaced.

定义 (ambiguity (二义性))

- A string w is derived ***ambiguously*** in context-free grammar G if it has two or more different leftmost derivations.
- Grammar G is ***ambiguous*** if it generates some string ambiguously.

Some context-free languages can be generated only by ambiguous grammars. Such languages are called ***inherently ambiguous*** (固有二义性). e.g., $\{a^i b^j c^k \mid i = j \text{ or } j = k\}$

Chomsky Normal Form 乔姆斯基范式

定义 (Chomsky Normal Form (乔姆斯基范式))

A context-free grammar is in ***Chomsky normal form*** if every rule is of the form

$$A \to BC$$
$$B \to a$$

- where a is any terminal and A, B, and C are any variables,
- except that B and C may not be the start variable.
- permit the rule $S \to \varepsilon$, where S is the start variable.

Chomsky Normal Form 乔姆斯基范式

定理

Any context-free language is generated by a context-free grammar in Chomsky normal form.

Chomsky Normal Form 乔姆斯基范式

定理

Any context-free language is generated by a context-free grammar in Chomsky normal form.

Proof idea:

Chomsky Normal Form 乔姆斯基范式

定理

Any context-free language is generated by a context-free grammar in Chomsky normal form.

Proof idea:

Rules that violate the conditions are replaced with equivalent ones that are satisfactory

1. add a new start variable

Chomsky Normal Form 乔姆斯基范式

定理

Any context-free language is generated by a context-free grammar in Chomsky normal form.

Proof idea:

Rules that violate the conditions are replaced with equivalent ones that are satisfactory

1. add a new start variable
2. eliminate all ε-***rules*** of the form $A \to \varepsilon$

Chomsky Normal Form 乔姆斯基范式

定理

Any context-free language is generated by a context-free grammar in Chomsky normal form.

Proof idea:

Rules that violate the conditions are replaced with equivalent ones that are satisfactory

1. add a new start variable
2. eliminate all ε-***rules*** of the form $A \to \varepsilon$
3. eliminate all ***unit rules*** of the form $A \to B$

Chomsky Normal Form 乔姆斯基范式

定理

Any context-free language is generated by a context-free grammar in Chomsky normal form.

Proof idea:

Rules that violate the conditions are replaced with equivalent ones that are satisfactory

1. add a new start variable
2. eliminate all ε-***rules*** of the form $A \to \varepsilon$
3. eliminate all ***unit rules*** of the form $A \to B$
4. convert the remaining rules into the proper form

Chomsky Normal Form 乔姆斯基范式

Proof

1. add a new start variable

Chomsky Normal Form 乔姆斯基范式

Proof

1. add a new start variable
   - add a new start variable S_0 and the rule $S_0 \to S$, where S was the original start variable.

Chomsky Normal Form 乔姆斯基范式

Proof

1. add a new start variable
   - add a new start variable S_0 and the rule $S_0 \rightarrow S$, where S was the original start variable.
   - This change guarantees that the start variable doesn't occur on the right-hand side of a rule.

Chomsky Normal Form 乔姆斯基范式

Proof

2. eliminate all ε-rules of the form $A \to \varepsilon$

Chomsky Normal Form 乔姆斯基范式

Proof

2. eliminate all ε-rules of the form $A \to \varepsilon$
   - remove an ε-rule $A \to \varepsilon$, where A is not the start variable.

Chomsky Normal Form 乔姆斯基范式

Proof

2. eliminate all ε-rules of the form $A \to \varepsilon$
   - remove an ε-rule $A \to \varepsilon$, where A is not the start variable.
   - for each occurrence of an A on the right-hand side of a rule, add a new rule with that occurrence deleted.

Chomsky Normal Form 乔姆斯基范式

Proof

2. eliminate all ε-rules of the form $A \to \varepsilon$
    - remove an ε-rule $A \to \varepsilon$, where A is not the start variable.
    - for each occurrence of an A on the right-hand side of a rule, add a new rule with that occurrence deleted.
    - $R \to uAv \Rightarrow R \to uv$
    - $R \to uAvAw \Rightarrow R \to uvAw$, $R \to uAvw$, and $R \to uvw$

Chomsky Normal Form 乔姆斯基范式

Proof

2. eliminate all ε-rules of the form $A \to \varepsilon$
   - remove an ε-rule $A \to \varepsilon$, where A is not the start variable.
   - for each occurrence of an A on the right-hand side of a rule, add a new rule with that occurrence deleted.
   - $R \to uAv \Rightarrow R \to uv$
   - $R \to uAvAw \Rightarrow R \to uvAw$, $R \to uAvw$, and $R \to uvw$
   - If we have the rule $R \to A$, add $R \to \varepsilon$ unless the rule $R \to \varepsilon$ was previously removed.

Chomsky Normal Form 乔姆斯基范式

Proof

2. eliminate all ε-rules of the form $A \to \varepsilon$
   - remove an ε-rule $A \to \varepsilon$, where A is not the start variable.
   - for each occurrence of an A on the right-hand side of a rule, add a new rule with that occurrence deleted.
   - $R \to uAv \Rightarrow R \to uv$
   - $R \to uAvAw \Rightarrow R \to uvAw$, $R \to uAvw$, and $R \to uvw$
   - If we have the rule $R \to A$, add $R \to \varepsilon$ unless the rule $R \to \varepsilon$ was previously removed.
   - repeat these steps until we eliminate all ε-rules not involving the start variable

Chomsky Normal Form 乔姆斯基范式

Proof

3. eliminate all unit rules of the form $A \to B$

Chomsky Normal Form 乔姆斯基范式

Proof

3. eliminate all unit rules of the form $A \to B$
   - remove a unit rule $A \to B$

Chomsky Normal Form 乔姆斯基范式

Proof

3. eliminate all unit rules of the form $A \to B$
   - remove a unit rule $A \to B$
   - whenever a rule $B \to u$ appears, we add the rule $A \to u$ unless this was a unit rule previously removed.

Chomsky Normal Form 乔姆斯基范式

Proof

3. eliminate all unit rules of the form $A \to B$
    - remove a unit rule $A \to B$
    - whenever a rule $B \to u$ appears, we add the rule $A \to u$ unless this was a unit rule previously removed.
    - repeat these steps until we eliminate all unit rules.

Chomsky Normal Form 乔姆斯基范式

Proof

4. convert the remaining rules into the proper form

Chomsky Normal Form 乔姆斯基范式

Proof

4. convert the remaining rules into the proper form
   - replace each rule $A \to u_1 u_2 \cdots u_k$,
     with the rules $A \to u_1 A_1$, $A_1 \to u_2 A_2$, $A_2 \to u_3 A_3$, $\cdots$, and $A_{k-2} \to u_{k-1} u_k$
     - where $k \geq 3$
     - each u_i is a variable or terminal symbol
     - A_i's are new variables

Chomsky Normal Form 乔姆斯基范式

Proof

4. convert the remaining rules into the proper form
   - replace each rule $A \to u_1 u_2 \cdots u_k$,
     with the rules $A \to u_1 A_1$, $A_1 \to u_2 A_2$, $A_2 \to u_3 A_3$, $\cdots$, and $A_{k-2} \to u_{k-1} u_k$
     - where $k \geq 3$
     - each u_i is a variable or terminal symbol
     - A_i's are new variables
   - replace any terminal u_i in the preceding rule(s) with the new variable U_i and add the rule $U_i \to u_i$

Chomsky Normal Form

例

Let G_6 be the following CFG and convert it to Chomsky normal form by using the conversion procedure just given.

1. The original CFG G_6 is shown on the left. The result of applying the first step to make a new start variable appears on the right.

$$S \to ASA \mid aB$$
$$A \to B \mid S$$
$$B \to b \mid \varepsilon$$

Chomsky Normal Form

例

Let G_6 be the following CFG and convert it to Chomsky normal form by using the conversion procedure just given.

1. The original CFG G_6 is shown on the left. The result of applying the first step to make a new start variable appears on the right.

$S \to ASA \mid aB$	$S_0 \to S$
$A \to B \mid S$	$S \to ASA \mid aB$
$B \to b \mid \varepsilon$	$A \to B \mid S$
	$B \to b \mid \varepsilon$

Chomsky Normal Form

例

Let G_6 be the following CFG and convert it to Chomsky normal form by using the conversion procedure just given.

2. Remove ε-rules $B \to \varepsilon$, shown on the left, and $A \to \varepsilon$, shown on the right.

$$S_0 \to S$$
$$S \to ASA \mid aB \mid a$$
$$A \to B \mid S \mid \varepsilon$$
$$B \to b \mid \varepsilon$$

Chomsky Normal Form

例

Let G_6 be the following CFG and convert it to Chomsky normal form by using the conversion procedure just given.

2. Remove ε-rules $B \to \varepsilon$, shown on the left, and $A \to \varepsilon$, shown on the right.

$$\begin{aligned} S_0 &\to S \\ S &\to ASA \mid aB \mid a \\ A &\to B \mid S \mid \varepsilon \\ B &\to b \mid \varepsilon \end{aligned} \qquad \begin{aligned} S_0 &\to S \\ S &\to ASA \mid aB \mid a \mid SA \mid AS \mid S \\ A &\to B \mid S \mid \varepsilon \\ B &\to b \end{aligned}$$

Chomsky Normal Form

例

Let G_6 be the following CFG and convert it to Chomsky normal form by using the conversion procedure just given.

3. (a) Remove unit rules $S \rightarrow S$, shown on the left, and $S_0 \rightarrow S$, shown on the right.

$S_0 \rightarrow S$

$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS \mid S$

$A \rightarrow B \mid S$

$B \rightarrow b$

Chomsky Normal Form

例

Let G_6 be the following CFG and convert it to Chomsky normal form by using the conversion procedure just given.

3. (a) Remove unit rules $S \to S$, shown on the left, and $S_0 \to S$, shown on the right.

$S_0 \to S$	$S_0 \to S \mid ASA \mid aB \mid a \mid SA \mid AS$
$S \to ASA \mid aB \mid a \mid SA \mid AS \mid S$	$S \to ASA \mid aB \mid a \mid SA \mid AS$
$A \to B \mid S$	$A \to B \mid S$
$B \to b$	$B \to b$

Chomsky Normal Form

例

Let G_6 be the following CFG and convert it to Chomsky normal form by using the conversion procedure just given.

3. (b) Remove unit rules $A \to B$ and $A \to S$.

$S_0 \to ASA \mid aB \mid a \mid SA \mid AS$

$S \to ASA \mid aB \mid a \mid SA \mid AS$

$A \to B \mid S \mid b$

$B \to b$

Chomsky Normal Form

例

Let G_6 be the following CFG and convert it to Chomsky normal form by using the conversion procedure just given.

3. (b) Remove unit rules $A \to B$ and $A \to S$.

$$S_0 \to ASA \mid aB \mid a \mid SA \mid AS$$
$$S \to ASA \mid aB \mid a \mid SA \mid AS$$
$$A \to B \mid S \mid b$$
$$B \to b$$

$$S_0 \to ASA \mid aB \mid a \mid SA \mid AS$$
$$S \to ASA \mid aB \mid a \mid SA \mid AS$$
$$A \to S \mid b \mid ASA \mid aB \mid a \mid SA \mid AS$$
$$B \to b$$

Chomsky Normal Form

例

Let G_6 be the following CFG and convert it to Chomsky normal form by using the conversion procedure just given.

4. Convert the remaining rules into the proper form by adding additional variables and rules. The final grammar in Chomsky normal form is equivalent to G_6.

$$S_0 \to AA_1 \mid UB \mid a \mid SA \mid AS$$
$$S \to AA_1 \mid UB \mid a \mid SA \mid AS$$
$$A \to b \mid AA_1 \mid UB \mid a \mid SA \mid AS$$
$$A_1 \to SA$$
$$U \to a$$
$$B \to b$$

Outline

Outline