5. 文脈自由文法と言語(1): (テキスト5.1)

5.1. 文脈自由文法(CFG; Context Free Grammar)

- 正則言語は言語としては十分な表現能力を持っているとは言えない。

例) $L=\{ 0^n1^n \mid n \geqq 0\}$ …{ε,01,0011,000111,…}

L={括弧の対応が取れている語} …

○ ( ), (( )), ( )( ), (( )( )),…

×),)(,))( ), (( )((, …

これらは正則言語ではない!!

- 上例の後者は現実の「言語」でも必須の能力
  - 複文(文章の入れ子構造)
  - HTML, LaTeX, C, …

5. 文脈自由文法と言語(1): (テキスト5.1)

5.1. 文脈自由文法(CFG; Context Free Grammar)

5.1.1. 直感的な例

回文(Palindrome): 前から読んでも後ろから読んでも同じ

例) たけやぶやけた、だんすがすんだ、いまきらくなくらきまい

$\Sigma=\{0,1\}$ のとき… ε, 0, 1, 00, 11, 000, 010, 101, 111, 0000,…

形式的には… $L_p=\{w \mid w=w^R\}$

L_pは正則言語ではない。

$\Sigma=\{0,1\}$ 上の回文の再帰的定義:

- ε, 0, 1 は回文。
- 回文 w に対して $0w0$, $1w1$ は回文。
- この規則で生成できるものだけが回文。

5. 文脈自由文法と言語(1): (テキスト5.1)

5.1. 文脈自由文法(CFG; Context Free Grammar)

5.1.1. 直感的な例

$\Sigma=\{0,1\}$ 上の回文の再帰的定義:

- ε, 0, 1 は回文。
- 回文 w に対して $0w0$, $1w1$ は回文。
- この規則で生成できるものだけが回文。

回文を生成する文脈自由文法

1. $P \to \varepsilon$
2. $P \to 0$
3. $P \to 1$
4. $P \to 0P0$
5. $P \to 1P1$

5. 文脈自由文法と言語(1): (テキスト5.1)

$$G_p = \{\{P\}, \{0,1\}, A, P\}$$
$$A: \begin{cases} P \to \varepsilon \\ P \to 0 \\ P \to 1 \\ P \to 0P0 \\ P \to 1P1 \end{cases}$$

5.1.2. 文脈自由文法の定義

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

- V: **変数**(または**非終端記号**、**文法概念**)
  - 書き換えるべき記号
- T: **終端記号**
  - 目的とする語を構成するアルファベット
- P: **生成規則**
  - 『非終端記号→非終端記号と終端記号の列』という書き換え規則の集まり
- S: **出発記号**
  - 最初に出発する非終端記号

5. 文脈自由文法と言語(1): (テキスト5.1)

5.1.2. 文脈自由文法の定義

例) L={a,+,(,)から構成される式}

$(a+a)$, $((a+a)+a+a)$, $(((a)))$,…

再帰的な定義:

1. a は式
2. E が式なら、(E) や $E+E$ も式

$G=\{\{P\},\{a,+,(,)\}, A, P\}$

ただし

$$A:\begin{cases} P \to a \\ P \to (P) \\ P \to P+P \end{cases}$$

$P \to a \mid (P) \mid P+P$
と書く場合が多い

5. 文脈自由文法と言語(1): (テキスト5.1)

5.1.3. 文法による導出

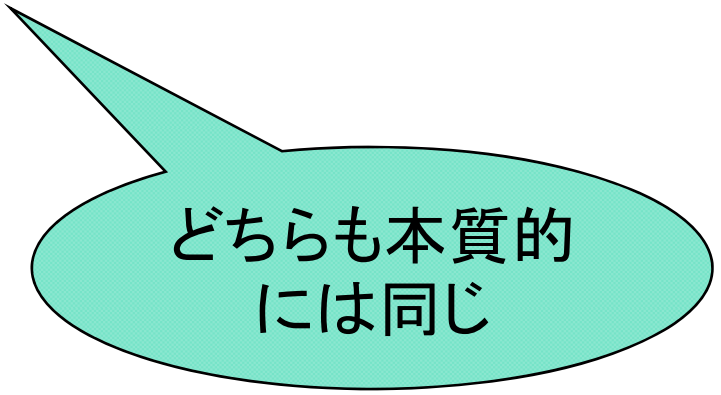
文法と実際に与えられる語について、、、

- 再帰的推論

  文字列(語＝終端記号列)から出発記号(非終端記号)

– 導出

  出発記号(非終端記号)から文字列(語)



5. 文脈自由文法と言語(1): (テキスト5.1)

5.1.3. 文法による導出

関係記号[$\Rightarrow$]

αの中にある非終端記号を一つ、文法Gの生成規則に基づいて書き換えたときにβが得られるとき $\alpha \underset{G}{\Rightarrow} \beta$ と書く。Gがわかっているときは$\alpha \Rightarrow \beta$と書く。

関係記号[$\overset{*}{\Rightarrow}$]

基礎: どんな列に対しても$\alpha \overset{*}{\underset{G}{\Rightarrow}} \alpha$

再帰: $\alpha \overset{*}{\underset{G}{\Rightarrow}} \beta$, $\beta \underset{G}{\Rightarrow} \gamma$なら、$\alpha \overset{*}{\underset{G}{\Rightarrow}} \gamma$。

Gがわかっているときは省略する。

5. 文脈自由文法と言語(1): (テキスト5.1)

5.1.3. 文法による導出

例)

文法 $G = \{\{P\},\{a,+,(,)\}, P \rightarrow a \mid (P) \mid P+P, P\}$

に対し、

語 $(a+((a+a)+a))$

の導出は以下の通り:

導出の途中で現れる文字列を文形式と呼ぶ

$$P \Rightarrow (P) \Rightarrow (P+P) \Rightarrow (a+P) \Rightarrow (a+(P)) \Rightarrow (a+(P+P))$$
$$\Rightarrow (a+((P)+P)) \Rightarrow (a+((P+P)+P)) \Rightarrow (a+((a+P)+P))$$
$$\Rightarrow (a+((a+a)+P)) \Rightarrow (a+((a+a)+a))$$
$$P \overset{*}{\Rightarrow} (a+((a+a)+a))$$

5. 文脈自由文法と言語(1): (テキスト5.1)

5.1.4. 最左導出と最右導出

非終端記号が複数あった場合に、どの非終端記号から生成規則を適用するか

- 最左導出…もっとも左にある非終端記号から生成規則を適用
- 最右導出…もっとも右にある非終端記号から生成規則を適用

例) $P \Rightarrow (P) \Rightarrow (P+P) \Rightarrow (a+P) \Rightarrow (a+(P)) \Rightarrow (a+(P+P))$

$\Rightarrow (a+((P)+P)) \Rightarrow (a+((P+P)+P)) \Rightarrow (a+((a+P)+P))$

$\Rightarrow (a+((a+a)+P)) \Rightarrow (a+((a+a)+a))$

5. 文脈自由文法と言語(1): (テキスト5.1)

5.1.5. ある文法の言語

与えられたCFG $G=(V, T, P, S)$ に対して、G によって表現される言語 $L(G)$ は

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

と定義できる。

非終端記号が前後の文脈と関係なく(=Context Free)書き換えられる

言語 L がある CFG G に対して $L(G)=L$ となるとき、L は文脈自由言語あるいは CFL (Context Free Language)と呼ばれる。

5. 文脈自由文法と言語(1): (テキスト5.1)

5.1.5. ある文法の言語

文法 $G_p=(\{P\}, \{0,1\}, P\rightarrow\varepsilon| 0 | 1 | 0P0 | 1P1, P)$ とする。

[定理] $L(G_p)$ は回文の集合である。

[証明] 以下の二つを証明すればよい。

1. $w=w^R$ なら、$P\overset{*}{\Rightarrow}w$であること
2. $P\overset{*}{\Rightarrow}w$ なら $w=w^R$ であること

5.1.5. ある文法の言語

文法 $G_p=(\{P\}, \{0,1\}, P\rightarrow\varepsilon| 0 | 1 | 0P0 | 1P1, P)$ とする。

[定理] $L(G_p)$ は回文の集合である。

[証明] 1. $w=w^R$ なら、$P\overset{*}{\Rightarrow}w$であること

$|w|$に関する帰納法による。

[基礎] $|w|=0$ のときは $w=\varepsilon$, $|w|=1$ のときは $w=0$ か $w=1$ であり、いずれも生成規則で直接生成できる回文である。

[帰納] $|w|=n>1$ として、$|w'|<n$ のときは回文w'はG_pで導出できると仮定。

$w=w^R$ なので、ある文字列xが存在し、

① $w=1x1$ か $w=0x0$ が成立し、かつ

② $x=x^R$ が成立する。

ここで $|x| = |w| - 2 < n$ なので、②と帰納法の仮定より、$P\overset{*}{\Rightarrow}x$ が成立する。

したがって①より

$P\Rightarrow 0P0\overset{*}{\Rightarrow}0x0=w$ または $P\Rightarrow 1P1\overset{*}{\Rightarrow}1x1=w$ が成立。

5.1.5. ある文法の言語

文法 $G_p=(\{P\}, \{0,1\}, P\rightarrow\varepsilon| 0 | 1 | 0P0 | 1P1, P)$ とする。

[定理] $L(G_p)$ は回文の集合である。

[証明] 2. $P\overset{*}{\Rightarrow}w$ なら、$w=w^R$ であること

導出の回数に関する帰納法による。

[基礎] 導出の回数が1回のときは $w=\varepsilon, 0, 1$ であり、いずれも回文である。

[帰納] w を導出するために規則をn回$(n>1)$適用したとする。規則を $n-1$回まで適用した場合は回文が生成されると仮定する。

n>1 なので、ある文字列 x が存在し、

①$P\Rightarrow 1P1\overset{*}{\Rightarrow}1x1=w$ か $P\Rightarrow 0P0\overset{*}{\Rightarrow}0x0=w$ が成立し、かつ

②x は P から $n-1$ 回の導出で得られる。

ここで $|x| = |w|-2 < n$ なので、②と帰納法の仮定より、$x=x^R$ が成立する。

したがって①より $w=w^R$ が成立。

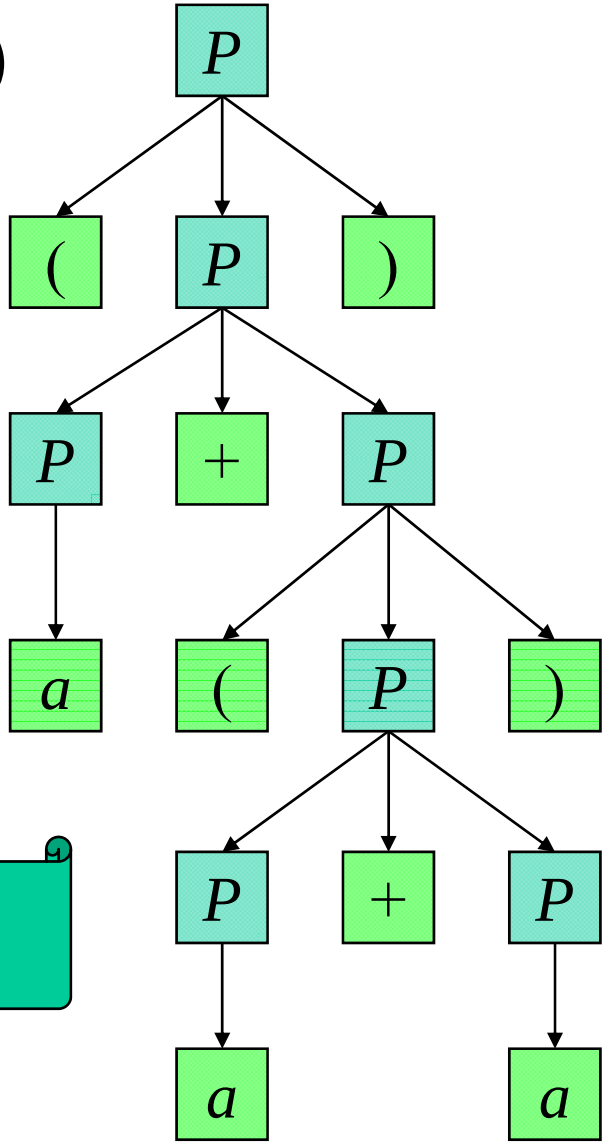
5. 文脈自由文法と言語(2):
(テキスト5.2)

5.2. 構文木

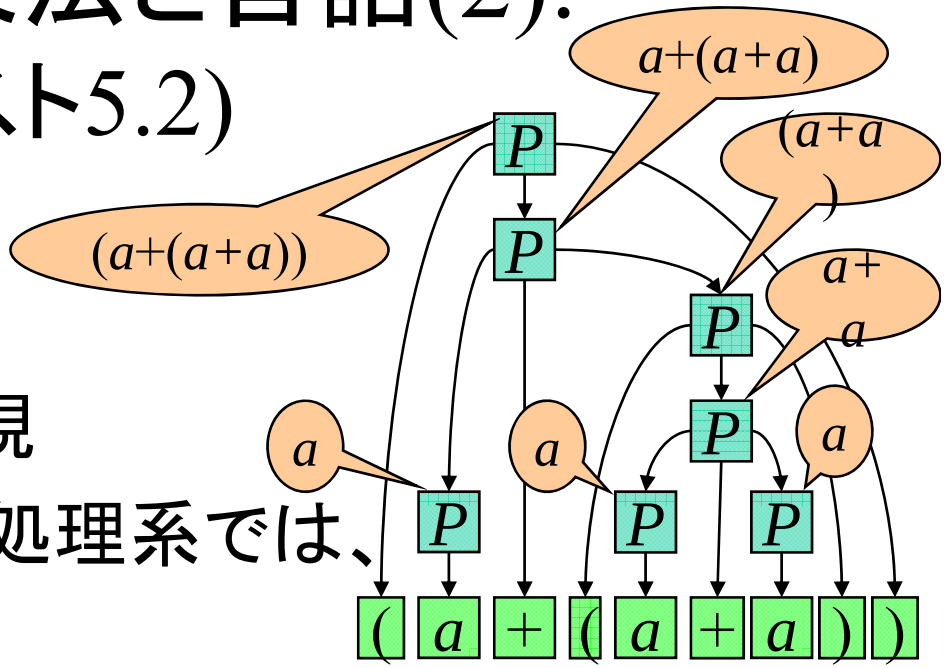
導出のプロセスは木構造で表現されることが多い。

例) $P \Rightarrow (P) \Rightarrow (P+P)$
$\Rightarrow (a+P) \Rightarrow (a+(P))$
$\Rightarrow a+(P+P))$
$\Rightarrow (a+(a+P))$
$\Rightarrow (a+(a+a))$

構文木



5. 文脈自由文法と言語(2): (テキスト5.2)



5.2. 構文木

- 「語」の導出過程を表現
- コンパイラなどの言語処理系では、
  - 式
  - 命令列

などの構造を表現する標準的なデータ構造

✓ 与えられた語に対する構文木の構成

は言語処理系では必須の機能

5. 文脈自由文法と言語(2): (テキスト5.2)

5.2. 構文木

- 曖昧な文法＝語に対する構文木が複数個存在する文法

⇒プログラミング言語では許されない

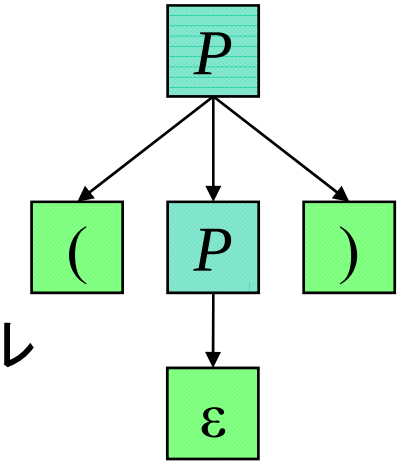
⇒自然言語処理でも大きな問題

例) Time flies like an arrow.

「時は矢のように飛ぶ」のか？

「時蝿は矢を好む」のか？

5.2.1. 構文木の構成



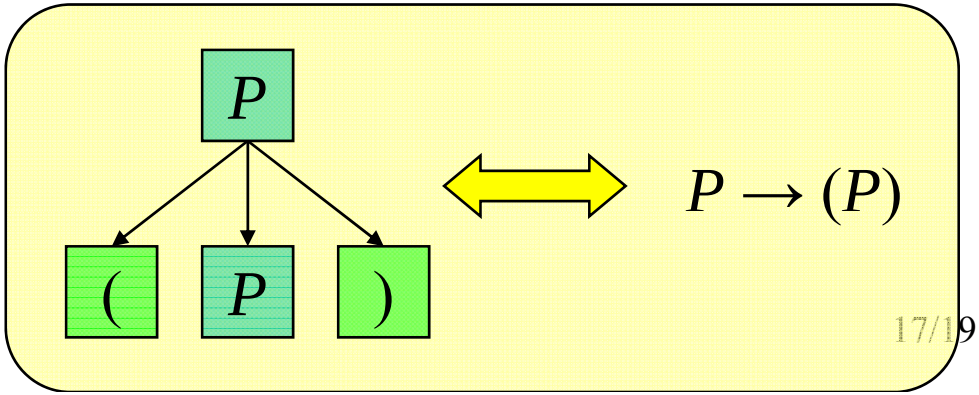
文法 $G=(V,T,P,S)$ に対して、Gの構文木とは、
以下の条件を満たす木:

1. 葉でない頂点にはV(非終端記号)がラベル
2. 葉のラベルは次のどれか一つ
   1. Tの要素(導出が終わっている頂点)
   2. Vの要素(まだ導出途中の頂点)
   3. ε(その葉が親の唯一の子供のとき)
3. 葉でない頂点のラベルがAで、子のラベルが左から

$$X_1, X_2, \ldots, X_k$$

なら、G は以下の生成規則を持つ

$$A \rightarrow X_1 X_2 \ldots X_k$$



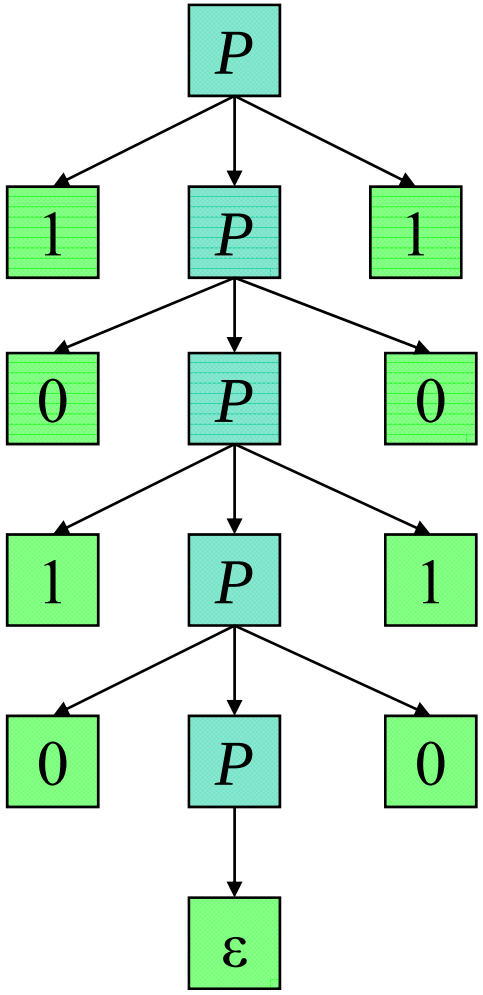
5. 文脈自由文法と言語(2): (テキスト5.2)

5.2.1. 構文木の構成

例)

$G_p = \{\{P\}, \{0,1\}, A, P\}$

$A: P \rightarrow \varepsilon \mid 0 \mid 1 \mid 0P0 \mid 1P1$



10100101の構文木

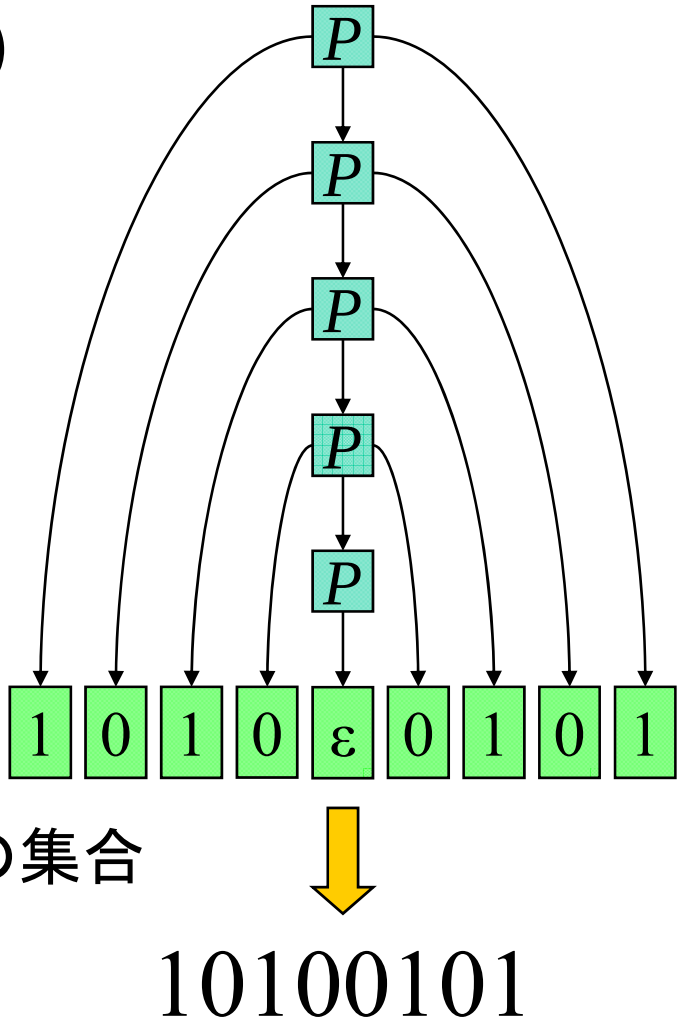
5. 文脈自由文法と言語(2): (テキスト5.2)

5.2.2. 構文木の成果

構文木が

1. 根のラベルが出発記号
2. 葉のラベルがすべて終端記号かεのとき、葉のラベルを左から並べた文字列を構文木の成果と言う。

(c.f. $a\varepsilon=\varepsilon a = a$)



[観測] 文法 G によって導出される語の集合
＝文法 G の成果である語の集合