計算の理論 I
文脈自由文法

火曜3校時
大月 美佳

今日の講義内容

1. 文脈自由文法(CFG)
   - 定義
   - 文脈自由言語 (クラスCFL)
   - 構文木と最左導出
- ミニテスト
- レポート出題

レポートについて(再掲)

- メ切：2004年7月13日 講義終了時
- 内容：
  正則表現→ ε-NFA→DFA→最小のDFA
  注意：
  - NFAからいきなり最小化はできない
  - DFAが最小に見えても穴埋めアルゴリズムを使って最小であることを示すこと
- 配点：100点中40点

文脈自由文法の重要性

- プログラミング言語の定義
  Buckus-Naur形式(BNF)記法
  教科書 5.3節 p. 216〜
- 構文解析の概念の定式化
- 言語間の変換の簡単化
- 各種文字処理
- 自然言語処理

文脈自由文法の定義

- **文脈自由文法**(context-free grammar)
  - **非終端アルファベット**N（**有限集合**）
    - **要素** x ∈N **非終端記号**(non-terminal symbol)
  - **終端アルファベット**T（**有限集合**）
    - **要素** x ∈ T **終端記号**(terminal symbol)
  - **開始記号**(start symbol) S∈Q
  - **生成規則**(production) P⊆N×(N∪Σ)*
    - A→α と書く（α = ε のとき ε 生成規則）
    - A→α_1|…| α_n = A→α_1,…, A→α_n

→ **G = (N, T, P, S)**

文脈自由文法の例

1. G=(N, T, P, S)を
   N={S}, T={a, b}, P={S→ab, S→aSb}
   とするとき、Gは文脈自由文法
2. G=(N, T, P, S)を
   N={S, A, B}, T={x, 0, 1},
   P={S→xA, A→0|1B, B→ε|0B|1B}
   とするとき、Gは文脈自由文法

$\Rightarrow_G$

- G=(N, T, P, S)に対して、V=N∪Tとする。
- u, vが次の(1), (2)を満たすとき$u \Rightarrow_G v$と書くことにする。

  (1) u=xAy, v=xαy (x, y, α∈V*, A∈N)。

  (2) A→αはGの生成規則。

導出
(derive)

- $\overset{*}{\Rightarrow}_G$

  $\Rightarrow_G$の反射的かつ推移的閉包
- V*の要素の列$w_0, \ldots, w_n$が

  $$w_0 \Rightarrow_G w_1 \Rightarrow_G \ldots \Rightarrow_G w_n$$

  となっているとき、

  w_0からw_nが導出されるといい、

  $$w_0 \overset{*}{\Rightarrow}_G w_n \quad \text{または} \quad w_0 \overset{n}{\Rightarrow}_G w_n$$

  と書く。(Gは省略可: $\Rightarrow$、$\overset{*}{\Rightarrow}$、$\overset{n}{\Rightarrow}$)

文脈自由言語
(context-free language)

- Gの生成する語w
  開始記号Sより
  終端アルファベットT上の記号列wが
  Gの生成規則によって導出されるとき
- T上の言語L∈T*に対して
  1. GがLを生成する：L=L(G)となるとき。
  2. 文脈自由言語: Lを生成する文脈自由文法が存在する。

文脈自由言語の例 その1

1. Gを例1の文脈自由文法としたとき、
   $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow \ldots \Rightarrow a^{n-1}Sb^{n-1} \Rightarrow a^nb^n$
   となり、
   $L(G)=\{a^nb^n \mid n \geqq 1\}$
   である。
2. Gを例2の文脈自由文法としたとき、
   $S \Rightarrow xA \Rightarrow x0$
   $S \Rightarrow xA \Rightarrow x1B \Rightarrow x10B \Rightarrow x10$
   となり、
   $L(G)=\{xu \mid u=1v,\ v \in \{0, 1\}^* \text{または} u=0\}$

文脈自由言語の例 その2

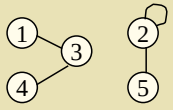
1. G=(N, T, P, S)を
   N={S}, T={), (}, P={S→SS, S→(S), S→()}
   とすると、言語L(G)はかっこの入れ子構造
2. G=(N, T, P, S)を
   N={S}, T={0, 1, Φ, +, *,), (},
   P={S→Φ|0|1|(S+S)|SS|S*}
   とすると、言語L(G)は{0, 1}上の正規表現全体

文脈自由文法のクラス

- 文脈自由文法のクラスをCFLと書く
- CFL
  ={L|L=L(G)となる文脈自由文法Gがある}

グラフの定義

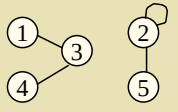
- **グラフ** G=(V, E)
  V: **有限個の頂点**(vertex, node)**の集合**
  E: **頂点の対**((v_1, v_2) **と表記**)**で示される辺**(edge)**の集合**



- **例** (**図**1.1 p.3)
  V={1, 2, 3, 4, 5}
  E={(n, m)|n+m=4 **または、** n+m=7}

道

- **道** (path)、**路**
  - **グラフのある頂点の列** $v_1, v_2, \ldots, v_k$ ($k \geqq 1$)**が道であるというのは、**
  - $(v_1, v_2), (v_2, v_3), \ldots, (v_{k-1}, v_k)$**がいずれも辺であるということ**
- **閉路** (cycle) : $v_i=v_k$**のとき**
- **道の例** (**図**1.1 p.3)
  - 1, 3, 4
  - 2
  - 2, 5

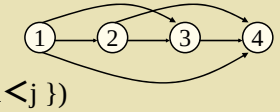


有向グラフ

- **有向グラフ** (directed graph, digraph) G
  - G=(V, E)
  - E**の要素が有向辺**(arc)
  - v→w : v**から**w**へ向かう有向辺**

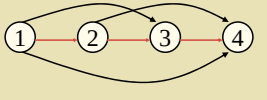
前者 (predecessor)　後者 (successor)

- **例** (**図**1.2 p.3)
  ({ 1, 2, 3, 4 }, { i→j|i＜j })



有向グラフの道

- **有向グラフの道** (path)
  := $v_i \rightarrow v_{i+1}$ ($1 \leqq i < k$) **が有向辺であるような頂点の列** $v_1, v_2, \ldots, v_k$ (**ただし、** $k \geqq 1$)



- **例** (**図**1.2 p.3)
  1→2→3→4 : 1**から**4**への道**

木

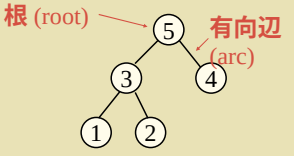
- **木** (tree, ordered directed tree)

  次の性質を持つ有効グラフ

  1. **前者を持たず、各頂点への道が必ず存在する根** (root)**と呼ばれる頂点を一つ持つ**
  2. **根以外の頂点はそれぞれただ一つ前者を持つ**
  3. **各頂点の後者は左から右へ一列に順序つけられている**

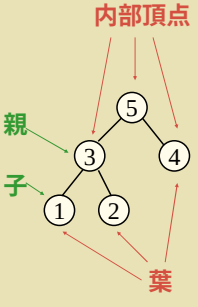
木の書き方

- **根を上に、各有向辺を下に向けて書く**
- **有向辺の矢印は書く必要がない**
- **頂点は**〈**なんらかの**〉**順序に従って左から右に書く**

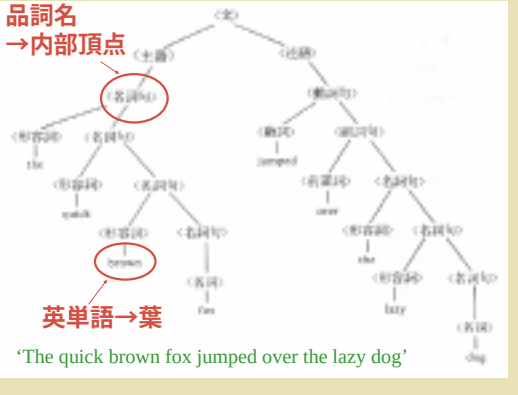


木特有の用語

- **親** (parent, father?) : **前者**
- **子** (child, son?) : **後者**
- **葉** (leaf) : **子を持たない頂点**
- **内部** (interior) **頂点** : **葉でない頂点**
- **先祖** (ancestor) **と子孫** (descendant)
  - **頂点**v_1**から**v_2**への道があるとき(**v_1:**先祖、**v_2:**子孫)**
  - **各頂点は自分自身の先祖かつ子孫**



構文木



構文木
(parse tree)

- 文脈自由文法G=(N, T, P, S)に対して、構文木を帰納的に定義する。
- ここで、V=N∪Tとする。

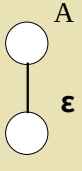
構文木の定義(1)

(1) 各A∈Vに対して、記号Aをラベルとする1つの頂点のみからなる木

は構文木である。

構文木の定義(2)

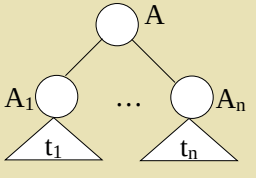
(2) A→ ε に対して、



は構文木である。

構文木の定義(3)

(3) $t_1, \ldots t_n$を根のラベルが、$A_1, \ldots, A_n \in V$である構文木とする。Gの生成規則$A \rightarrow A_1 \ldots A_n$に対して、Aを根とする木



は構文木である。

構文木の定義(4)

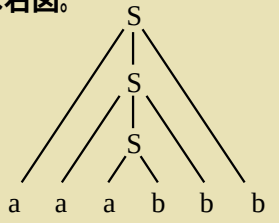
(4) 上の(1)〜(3)の規則を使って定義される有限の木のみを構文木という。

構文木の例

G=(N, T, P, S)を

N={S}, T={a, b}, P={S→ab, S→aSb}

としたとき、構文木は右図。



最左導出
(leftmost derivation)

- 導出$u \overset{*}{\Rightarrow} v$の各ステップにおいて一番左にある非終端記号が置き換えられているとき、その導出は最左導出である。
  - G=(N, T, P, S)を

    N={S}, T={), (}, P={S→SS, S→(S), S→()}

    とすると、

    S⇒(S)⇒((S))⇒((SS))⇒((()S)⇒((()()))

    は最左導出。

走査
(traverse)

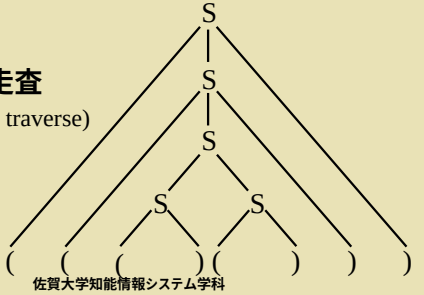
- 根から始めてどのように走査するか

  $A \overset{*}{\Rightarrow} u$

  最左導出

  →左側順走査

  (preorder traverse)



補題

- G=(N, T, P, S)を文脈自由文法とする。導出 $u \Rightarrow v$があるならば、uからvへの最左導出がある。

  証明

  $u=w_1A_1w_2\ldots w_kA_kw_{k+1}$($A_i \in N$, $w_i \in T^*$)とする。
  このとき、vの分解$v= w_1v_1w_2\ldots w_kv_kw_{k+1}$があって、
  $A_i \Rightarrow v_i$ ($1 \leqq i \leqq k$)となる。
  そこで、$A_i \Rightarrow v_i$に対する構文木を取り、その左側順走査を考えれば、A_iからv_iへの最左導出が得られる。
  それらの最左導出を、非終端記号の順に適用していけば求める最左導出が得られる。

最右導出

(rightmost derivation)

- 最左導出と反対に右から置き換えていく導出

  G=(N, T, P, S)を
  N={S}, T={), (}, P={S→SS, S→(S), S→()}
  とすると、
  S⇒(S)⇒((S))⇒((SS))⇒((S ()))⇒((()()))
  は最左導出。

曖昧性

- 曖昧
  ある語に二つ以上の構文木があるとき
  ⇔
  – 二つ以上の最左導出をもつ
  または
  – 二つ以上の最右導出をもつ
  とき
- 本質的に曖昧→4.7 (p.128)

ミニテストと次回内容

- ミニテスト
  教科書･資料を見ても、友達と相談しても良い
- ミニテストを提出すること
  出したら帰って良し
- 次回(7/6)内容
  プッシュダウンオートマトン(PDA)