計算論A / 2015年度

文脈依存言語
(今回は教科書の範囲外です)

角川　裕次

もくじ

＊＊＊ 本日の重要概念 ＊＊＊

文脈依存言語, 言語クラスの階層

φ(・ω・)メモメモ

1. 文脈依存文法(CSG)と文脈依存言語(CSL)

文脈依存文法 (Context-Sensitive Grammar; CSG)

文脈依存文法 G は 4つ組 $G = (V, T, P, S)$ で表現

T: 終端記号の集合

- 定義される言語の文字列を構成する記号の有限集合

V: 変数 (非終端記号)の集合

- 文字列の集合を表現する記号

S: 出発記号 (始記号)

- 定義する言語を表す変数

P: 規則 (生成規則)の有限集合

- 言語の再帰的定義を表現
- 規則の形式が文脈自由言語より緩やか
- ※詳しくは次のスライド

規則 (生成規則) の形式

「頭部 → 本体」の形式: $\alpha X \beta \rightarrow \alpha \gamma \beta$

- 例: $X \rightarrow XY0$
- 例: $aXb \rightarrow aX1b$

頭部 $\alpha X \beta$: 置き換えの対象となる列

- $\alpha, \beta \in (V \cup T)^*$
- $X \in V$

本体 $\alpha \gamma \beta$: 置き換える列

- $\gamma \in (V \cup T)^+$　(※長さ1以上であることに注意)
- ※置き換え(生成規則の適用)により列は短くならない

「文脈(context)」とは

生成規則$\alpha X \beta \rightarrow \alpha \gamma \beta$ の形に注目

実際に置き換わるのは X のみ

- α と β は変わらない

X の出現する文脈に応じて置き換えが行われる

- 文脈: X の前後の α と β のこと

生成規則の適用

出発記号より開始

生成規則に従って置き換えてゆく

最終的に終端記号だけの列へ

文脈依存言語 (Context-Sensitive Language; CSL)

文脈依存文法で生成される言語のこと

- L が CSL ならば $L \cup \{\varepsilon\}$ も CSL と定める

文脈依存言語のクラス

- 文脈依存文法で生成される言語のクラス

単調文法

生成規則が $\alpha \rightarrow \beta$ の形をした文法のこと

- ただし $\alpha, \beta \in (V \cup T)^+$ かつ
- $|\alpha| \leq |\beta|$

既知の結果:
単調文法で生成される言語クラス
= 文脈依存言語のクラス

生成規則 $AB \rightarrow BA$ の形の導入

生成規則 $AB \rightarrow BA$ (2変数の前後交換) を使用して良い

- 本来の CSG で許されていない形の生成規則

使用して良い理由

- 新たに変数 X, Y を導入
- 追加生成規則1: $AB \rightarrow XB$
- 追加生成規則2: $XB \rightarrow XY$
- 追加生成規則3: $XY \rightarrow BY$
- 追加生成規則4: $BY \rightarrow BA$
  - ✔ 本来の CSG で許されている形の生成規則のみを使用
- $AB \overset{*}{\underset{G}{\Rightarrow}} BA$ を得る

黒田標準型

CSG の標準型: 黒田標準型 (1/2)

どの生成規則も以下の形に限定した文法

- $X \rightarrow YZ$
- $XY \rightarrow XZ$
- $XY \rightarrow ZY$
- $X \rightarrow Y$
- $X \rightarrow a$

(ただし $X, Y, Z \in V, a \in T$)

(黒田標準系の別形式) 以下の形に限定

- $AB \rightarrow CD$
- $A \rightarrow BC$
- $A \rightarrow B$
- $A \rightarrow a$

(ただし $A, B, C, D \in V, a \in T$)

CSG の標準型: 黒田標準型 (2/2)

任意のCSL L に対し L を生成する黒田標準型が存在

文法の例 1

例1

言語 $L = \{a^n b^n c^n \mid n \geq 1\}$

文脈自由言語ではない

文脈依存文法で生成できることを以下に示す

文法

以下の生成規則を持つ文法で生成

- $S \to aSBC$
- $S \to aBC$
- $aB \to ab$
- $bC \to bc$
- $CB \to BC$
- $bB \to bb$
- $cC \to cc$

導出例

$\underline{S}$
$\Rightarrow a\underline{S}BC$
$\Rightarrow aaB\underline{CB}C$
$\Rightarrow a\underline{aB}BCC$
$\Rightarrow aa\underline{bB}CC$
$\Rightarrow aab\underline{bC}C$
$\Rightarrow aabb\underline{cC}$
$\Rightarrow aabbcc$

文法の例 2

例2

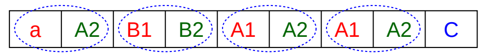
言語 $L = \{w_1 c w_2 \mid w_1, w_2 \in \{a,b\}^*, w_1 = w_2\}$

文脈自由言語ではない

文脈依存文法で生成できることを以下に示す

文法 (1/6)

ステップ1: 以下の文形式をつくる
(以下の例: 最終的に $abaacabaa$ を導出)



- 変数 A_1, B_1: この後それぞれ w_1 の a, b へ
- 変数 A_2, B_2: この後それぞれ w_2 の a, b へ
- 変数 C: c へ

生成規則

- $S \ \rightarrow\ c \mid aA_2T \mid bB_2T$
- $T \ \rightarrow\ A_1A_2T \mid B_1B_2T \mid C$

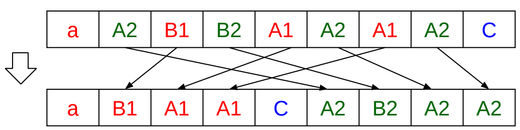
文法 (2/6)

導出例

- $$\begin{aligned} S &\Rightarrow aA_2T \\ &\Rightarrow aA_2B_1B_2T \\ &\Rightarrow aA_2B_1B_2A_1A_2T \\ &\Rightarrow aA_2B_1B_2A_1A_2A_1A_2T \\ &\Rightarrow aA_2B_1B_2A_1A_2A_1A_2C \end{aligned}$$

文法 (3/6)

ステップ2: 変数 A_2, B_2 を変数 C の右側へ(順序は保持)



生成規則

- $A_2A_1 \rightarrow A_1A_2,\quad A_2B_1 \rightarrow B_1A_2,$ — 右へ移動
  $B_2A_1 \rightarrow A_1B_2,\quad B_2B_1 \rightarrow B_1B_2$
- $A_2C \rightarrow CA_2,\quad B_2C \rightarrow CB_2$ — C と交換

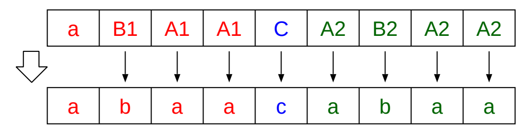
文法 (4/6)

導出例

- $$\begin{aligned} S &\Rightarrow^* aA_2B_1B_2A_1A_2A_1A_2C \\ &\Rightarrow aB_1A_2B_2A_1A_2A_1A_2C \\ &\Rightarrow aB_1A_2A_1B_2A_2A_1A_2C \\ &\Rightarrow aB_1A_2A_1B_2A_2A_2A_1C \\ &\Rightarrow aB_1A_1A_2B_2A_2A_2A_1C \\ &\Rightarrow aB_1A_1A_2B_2A_2A_1A_2C \\ &\Rightarrow aB_1A_1A_2B_2A_1A_2A_2C \\ &\Rightarrow aB_1A_1A_2A_1B_2A_2A_2C \\ &\Rightarrow aB_1A_1A_1A_2B_2A_2A_2C \\ &\Rightarrow aB_1A_1A_1A_2B_2A_2CA_2 \\ &\Rightarrow aB_1A_1A_1A_2B_2CA_2A_2 \\ &\Rightarrow aB_1A_1A_1A_2CB_2A_2A_2 \\ &\Rightarrow aB_1A_1A_1CA_2B_2A_2A_2 \end{aligned}$$

文法 (5/6)

ステップ3: 変数を終端記号へ(左から順に)



生成規則

- $aA_1 \rightarrow aa,\quad aB_1 \rightarrow ab,$ — w_1 の部分
  $bA_1 \rightarrow ba,\quad bB_1 \rightarrow bb,$
- $aC \rightarrow ac,\quad bC \rightarrow bc,$ — 変数 C を終端記号に
- $cA_2 \rightarrow ca,\quad cB_2 \rightarrow cb,$ — w_2 の左端
- $aA_2 \rightarrow aa,\quad aB_2 \rightarrow ab,$ — w_2 の部分
  $bA_2 \rightarrow ba,\quad bB_2 \rightarrow bb$

文法 (6/6)

導出例

- $S \Rightarrow^* aB_1A_1A_1CA_2B_2A_2A_2$
  $\Rightarrow abA_1A_1CA_2B_2A_2A_2$
  $\Rightarrow abaA_1CA_2B_2A_2A_2$
  $\Rightarrow abaaCA_2B_2A_2A_2$
  $\Rightarrow abaacA_2B_2A_2A_2$
  $\Rightarrow abaacaB_2A_2A_2$
  $\Rightarrow abaacabA_2A_2$
  $\Rightarrow abaacabaA_2$
  $\Rightarrow abaacabaa$

2. 線形有界オートマトン (Linear-bounded Automaton; LBA)

線形有界オートマトン(Linear-bounded Automaton; LBA)

CSLを認識するオートマトン

LBAの受理する言語のクラスはCSL のクラスに一致

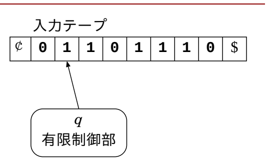
線形有界オートマトン(LBA)の概要

有限制御部

- 状態遷移関数に従って動作 (決定性/非決定性)

テープ

- 入力記号列が書き込まれて与えられる
- 左端記号 ¢ , 右端記号 $ が置かれる
- ヘッドを左右に動かせる
- テープ上の入力記号列を書き換えできる



定義 (1/3)

7つ組 $M = (Q, \Sigma, \Gamma, \delta, q_0, ¢, \$, F)$

- Q : 状態すべての集合 (有限集合)
- Σ : 入力記号 ($\Sigma \subseteq \Gamma$)
- Γ : テープ記号すべての集合 (有限集合)
- δ : M の遷移関数
- q_0 : 初期状態 ($\in Q$)
- $¢, \$$: 左端記号と右端記号 (Γには含まれない)
- F : 受理状態の集合 ($\subseteq Q$)

定義 (2/3): M の遷移関数 δ

δ の入力

- 現在の状態: $q \in Q$
- ヘッドが読む記号: $a \in \Gamma \cup \{¢, \$\}$

δ の出力 (以下の組が0以上 — 非決定性)

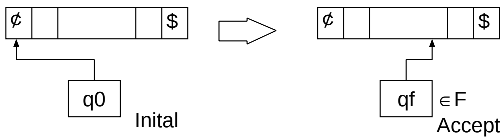
- 次の状態: $p \in Q$
- ヘッド位置の記号の書き換え: $b \in \Gamma \cup \{¢, \$\}$
- ヘッドの移動量: -1 (左), 0 (動かさない), $+1$ (右),

定義 (3/3): M による受理

初期状況より動作を開始

- 初期状態 q_0
- ヘッドは左端に位置 (¢ 上)

受理状態 ($\in F$) に入れば受理



LBA の例 1

$L = \{a^n b^n c^n \mid n \geq 1\}$ (1/2)

LBA の動作の概要

- a, b, c を1つづつ消して行く
- 最後に全てが同時に消えればよい

初期状況でのテープ内容

¢	a	a	a	b	b	b	c	c	c	$

一番左の a, b, c をそれぞれ A, B, C に書き換える

- A, B, C： 消去された記号
  (「消えたよ」マークが付いて消えたものとみなす)

¢	A	a	a	B	b	b	C	c	c	$

繰り返す

$L = \{a^n b^n c^n \mid n \geq 1\}$ (2/2)

一番左の a, b, c をそれぞれ A, B, C に書き換える (繰返)

¢	A	a	a	B	b	b	C	c	c	$

⇩

¢	A	A	a	B	B	b	C	B	c	$

⇩

¢	A	A	A	B	B	B	C	C	C	$

- 「a, b, c が同時になくなった」
  — a, b, c が同数なので受理
- 「a, b, c でなくなったものと残ったものがある」
  — a, b, c が同数でないので棄却

LBA の例 2

$L = \{w_1 c w_2 \mid w_1, w_2 \in \{a, b\}^*, w_1 = w_2\}$ (1/2)

LBA の動作の概要

- w_1, w_2 を左から順に1文字づつ照合
- 照合が済んだ文字は A または B に書き換え

初期状況でのテープ内容

¢	a	b	a	a	c	a	b	a	a	$

照合第1回目

- w_1 の左端の未照合文字と w_2 の左端の未照合文字を照合
- 合致しないと入力を棄却

¢	A	b	a	a	c	A	b	a	a	$

繰り返す

$L = \{w_1 c w_2 \mid w_1, w_2 \in \{a,b\}^*, w_1 = w_2\}$ (2/2)

照合を継続...

- w_1 の左端の未照合文字と w_2 の左端の未照合文字を照合
- 合致しないと入力を棄却

¢	A	B	A	A	c	A	B	A	A	$

すべての文字で合致すれば受理

非決定 LBA と決定性 LBA

非決定性 LBA (Nondeterministic LBA; NLBA)

- 複数の動作があるもの
- ひとつの動作でも受理に至れば入力を受理とする

決定性 LBA (Deterministic LBA; DLBA)

- 動作はたかだか1つ

NLBA の受理能力

NLBAの受理する言語クラス

NLBA の受理する言語クラス ＝ 文脈依存言語のクラス

証明 (⇐)

任意の CSG G に対し
G が生成する言語を受理する NLBA M が存在

証明のあらすじ:

- G と等価な黒田標準型の文法 G' が存在
- G' の導出を模倣する NLBA M を構成できる

証明 (⇒)

任意の NLBA M に対し
M の受理する言語を生成する CSG G が存在

証明のあらすじ:

- M の遷移関数をCSGの生成規則の形で表現できる
- M が入力を受理するとき,
  その受理計算に対応する導出が存在
- M が入力を受理しないとき,
  導出は存在しない

3. 文脈依存言語(CSL)の性質

概要

CSL に対する反復補題: まだ知られていない

- CSL でない簡潔な形の言語の例を示すことは難しい

閉包性/非閉包性をいくつか示す

NLBA と DLBA には真に能力差がある

閉包性

CSL は以下の演算のもとで閉じている

- 和集合 $\cup$
- 共通集合 $\cap$
- 差集合 $\setminus$
- 補集合 $\overline{}$
- 連接 $\cdot$
- Kleene 閉包 *
- 反転(鏡像) ^{R}

CSL は以下の演算のもとでは閉じていない

- 準同形写像
- 代入

共通集合 ∩ のもとで閉じていることの証明 (1/4)

L_1, L_2: 任意の CSL

M_1, M_2: それぞれ L_1, L_2 を受理する NLBA

NLBA M : $L_1 \cap L_2$ を受理

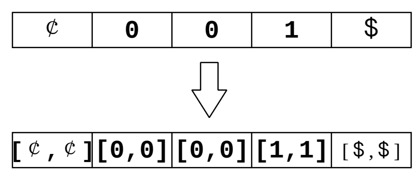
M の概要 (具体的な構成方法は次スライド以降で)

1. L_1 の語か否かを検査
2. L_2 の語か否かを検査
3. 両方の言語の語であれば入力を受理

共通集合 ∩ のもとで閉じていることの証明 (2/4)

ステップ0: 準備

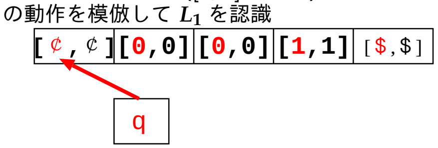
- 入力テープの各コマの内容 a を $[a, a]$ と書き換える
- 左成分は M_1 が処理, 右成分は M_2 が処理



共通集合 ∩ のもとで閉じていることの証明 (3/4)

ステップ1: L_1 の認識

- テープのコマの左成分 ($[a, b]$ での a) のみをアクセス
- M_1 の動作を模倣して L_1 を認識

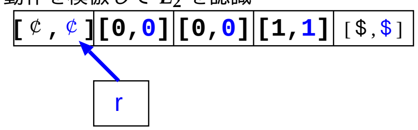


- M_1 が入力を棄却: M は棄却
- M_1 が入力を受理: 次のステップへ

共通集合 ∩ のもとで閉じていることの証明 (4/4)

ステップ2: L_2 の認識

- テープのコマの右成分 ($[a, b]$ での b) のみをアクセス
- M_2 の動作を模倣して L_2 を認識



- M_2 が入力を受理: M は受理
- M_2 が入力を棄却: M は棄却

M が受理 ⇔ 入力語は $L_1 \cap L_2$ の語

4. Chomsky 階層

4つの文法クラス: Chomsky 階層

0型文法: 句構造文法 (Phrase Structure Grammar)
1型文法: 文脈依存文法 (Context-Sensitive Grammar)
2型文法: 文脈自由文法 (Context-Free Grammar)
3型文法: 正規文法 (Regular Grammar)

これら文法のクラスは階層をなす

Chomsky — 形式言語を提唱した言語学者

0型文法: 句構造文法

生成規則の形: $\alpha X \beta \;\rightarrow\; \alpha \gamma \beta$

- 左辺には少なくとも1つの変数がある
- $\alpha, \beta \in (V \cup T)^*$
- $\gamma \in (V \cup T)^*$
  (置き換えにより文形式は短くなっても良い)

例と反例

- ○ $aAaA \rightarrow aBa$
- ○ $aA \rightarrow aaB$
- ○ $A \rightarrow aBbb$
- ○ $A \rightarrow cB$
- ○ $A \rightarrow c$

1型文法: 文脈依存文法

生成規則の形: $\alpha X \beta \;\rightarrow\; \alpha \gamma \beta$

- 左辺には少なくとも1つの変数がある
- $\alpha, \beta \in (V \cup T)^*$
- $\gamma \in (V \cup T)^+$
  (置き換えにより文形式は短くならない)
- 出発記号 S がどの規則の右辺にも現れない場合のみ
  生成規則 $S \rightarrow \varepsilon$ を許す

例と反例

- × $aAaA \rightarrow aBa$
- ○ $aA \rightarrow aaB$
- ○ $A \rightarrow aBbb$
- ○ $A \rightarrow cB$
- ○ $A \rightarrow c$

2型文法: 文脈自由文法

生成規則の形: $X \;\rightarrow\; \gamma$

- 左辺は1つの変数に限る
- $\gamma \in (V \cup T)^+$
  (置き換えにより文形式は短くならない)
- 出発記号 S がどの規則の右辺にも現れない場合のみ
  生成規則 $S \rightarrow \varepsilon$ を許す
  (この定義でも生成される言語クラスは同じ)

例と反例

- × $aAaA \rightarrow aBa$
- × $aA \rightarrow aaB$
- ○ $A \rightarrow aBbb$
- ○ $A \rightarrow cB$
- ○ $A \rightarrow c$

3型文法: 正規文法

生成規則の形: $X \;\rightarrow\; \gamma$

- 左辺は1つの変数に限る
- $\gamma \in TV$ または $\gamma \in T$
- 出発記号 S がどの規則の右辺にも現れない場合のみ生成規則 $S \rightarrow \varepsilon$ を許す

例と反例

- × $aAaA \rightarrow aBa$
- × $aA \rightarrow aaB$
- × $A \rightarrow aBbb$
- ○ $A \rightarrow cB$
- ○ $A \rightarrow c$

4つの言語クラス

句構造言語 (Phrase Structure Language)

- 句構造文法で生成される言語

文脈依存言語 (Context-Sensitive Language)

- 文脈依存文法で生成される言語

文脈自由言語 (Context-Free Language)

- 文脈自由文法で生成される言語

正規言語 (Regular Language)

- 正規文法で生成される言語

言語と認識機械の関係

句構造言語 ： チューリングマシン TM (次回の講義で)

文脈依存言語 ： 線形有界オートマトン LBA

文脈自由言語 ： プッシュダウンオートマトン PDA

正規言語 ： 有限オートマトン FA

記法: 言語クラスを表す記号の定義

$\mathcal{L}_{ps}$: 句構造言語の全体のクラス

$\mathcal{L}_{cs}$: 文脈依存言語の全体のクラス

$\mathcal{L}_{cf}$: 文脈自由言語の全体のクラス

$\mathcal{L}_{re}$: 正規言語の全体のクラス

言語クラスの階層性

$$\mathcal{L}_{ps} \supsetneq \mathcal{L}_{cs} \supsetneq \mathcal{L}_{cf} \supsetneq \mathcal{L}_{re}$$

4つの言語クラスの間には
真の包含関係がある

階層性の説明: $\mathcal{L}_{cf} \supsetneq \mathcal{L}_{re}$

包含関係 $\mathcal{L}_{cf} \supseteq \mathcal{L}_{re}$ (広いまたは等しい) は自明

- 文法制約の関係より

真の包含関係 $\mathcal{L}_{cf} \supsetneq \mathcal{L}_{re}$ (真に広い):

- $L \in \mathcal{L}_{cf}$ かつ $L \notin \mathcal{L}_{re}$ である言語 L が存在
- L ： FA では受理できないが PDA なら受理できる言語
- 実例1: $L = \{a^n b^n \mid n \geq 0\}$
- 実例2: $L = \{wcw^R \mid w \in \{a,b\}^*\}$

階層性の説明: $\mathcal{L}_{cs} \supsetneq \mathcal{L}_{cf}$

包含関係 $\mathcal{L}_{cs} \supseteq \mathcal{L}_{cf}$ (広いまたは等しい)は自明

- 文法制約の関係より

真の包含関係 $\mathcal{L}_{cs} \supsetneq \mathcal{L}_{cf}$ (真に広い):

- $L \in \mathcal{L}_{cs}$ かつ $L \notin \mathcal{L}_{cf}$ である言語 L が存在
- L ： PDA では受理できないが LBA なら受理できる言語
- 実例1: $L = \{a^n b^n c^n \mid n \geq 0\}$
- 実例2: $L = \{wcw \mid w \in \{a,b\}^*\}$

階層性の説明: $\mathcal{L}_{ps} \supsetneq \mathcal{L}_{cs}$

包含関係 $\mathcal{L}_{ps} \supseteq \mathcal{L}_{cs}$ (広いまたは等しい)は自明

- 文法制約の関係より

真の包含関係 $\mathcal{L}_{ps} \supsetneq \mathcal{L}_{cs}$ (真に広い):

- $L \in \mathcal{L}_{ps}$ かつ $L \notin \mathcal{L}_{cs}$ である言語 L が存在
- L ： LBA では受理できないが TM なら受理できる言語

今日のミニレポート (CLE)

CSL が連接演算・に関して閉じていることを証明せよ
(NLBA の構成法の概要を示せ)