計算理論

Thomas Zeugmann

北海道大学 大学院情報科学研究科
アルゴリズム研究室

http://www-alg.ist.hokudai.ac.jp/~thomas/ToC/

第 10 回：$\mathcal{CF}$, PDAs, その先

(日本語訳：湊 真一，大久保 好章)

©Thomas Zeugmann

グライバッハ標準形 I

すべての文脈自由言語は，プッシュダウンオートマトンで受理できることを示したい．そのために，文脈自由言語のもう 1 つの便利な標準形を用いる．これはグライバッハ標準形と呼ばれる．

©Thomas Zeugmann

グライバッハ標準形 I

すべての文脈自由言語は，プッシュダウンオートマトンで受理できることを示したい．そのために，文脈自由言語のもう 1 つの便利な標準形を用いる．これはグライバッハ標準形と呼ばれる．

定義 1

$\lambda \notin L(\mathcal{G})$ なる文脈自由文法もしも $\mathcal{G}$ のすべての生成規則が，$h \to a\alpha$ (ただし $a \in T$ かつ $\alpha \in N^*$) の形をしているならば，これを**グライバッハ標準形 *(Greibach normal form)*** と言う．

©Thomas Zeugmann

グライバッハ標準形 II

すべての文脈自由言語はグライバッハ標準形の文法を持つことを示したい．しかしそれは意外に簡単ではない．そこで，ここでは証明を省略し，テキストに書いてある証明を参照してもらうことにする．そこに例も載っている．

©Thomas Zeugmann

グライバッハ標準形 II

すべての文脈自由言語はグライバッハ標準形の文法を持つことを示したい．しかしそれは意外に簡単ではない．そこで，ここでは証明を省略し，テキストに書いてある証明を参照してもらうことにする．そこに例も載っている．

定理 1

すべての言語 $L \in \mathcal{CF}$ (ただし $\lambda \notin L$) に対して，$L = L(\tilde{\mathfrak{G}})$ かつ $\tilde{\mathfrak{G}}$ がグライバッハ標準形となるような，ある文法 $\tilde{\mathfrak{G}}$ が存在する．

CF から プッシュダウンオートマトンへ I

以上で，プッシュダウンオートマトンに関する基礎的な定理を示す準備ができた．

$\mathcal{CF}$ から プッシュダウンオートマトンへ I

以上で，プッシュダウンオートマトンに関する基礎的な定理を示す準備ができた．

定理 2

すべての言語 $L \in \mathcal{CF}$ について，$L = N(\mathcal{K})$ となるようなプッシュダウンオートマトン $\mathcal{K}$ が存在する．

(証明) $\lambda \notin L$ と仮定する．$\lambda \in L$ の場合の証明に必要な修正は演習問題として残しておく．$\mathcal{G} = [T, N, \sigma, P]$ を，$L = L(\mathcal{G})$ を満たすようなグライバッハ標準形の文脈自由文法とする．

$\mathcal{CF}$ から プッシュダウンオートマトンへ II

我々は次のような記法を必要とする．

終端記号と非終端記号からなる文字列 α は，$\sigma \overset{*}{\Longrightarrow} \alpha$ であるなら

ば，文形式 *(sentential form)* であると言う．

次に，

$$\mathcal{K} = [\{q\}, T, N, \delta, q, \sigma, \emptyset]$$

とする．ただし，$(A \to \alpha\gamma) \in P$ のときは常に
$(q, \gamma) \in \delta(q, a, A)$ であるとする．

$\mathcal{CF}$ から プッシュダウンオートマトンへ II

我々は次のような記法を必要とする．

終端記号と非終端記号からなる文字列 α は，$\sigma \overset{*}{\Longrightarrow} \alpha$ であるなら

ば，文形式 *(sentential form)* であると言う．

次に，

$$\mathcal{K} = [\{q\}, T, N, \delta, q, \sigma, \emptyset]$$

とする．ただし，$(A \rightarrow \alpha\gamma) \in P$ のときは常に
$(q, \gamma) \in \delta(q, a, A)$ であるとする．

このプッシュダウンオートマトン $\mathcal{K}$ は，$\mathcal{G}$ の最左導出をシミュレートする．$\mathcal{G}$ はグライバッハ標準形であるから，最左導出に含まれるすべての文形式は，終端記号の列 x と，それに続く非終端記号の文字列 α からなる．

$\mathcal{CF}$ から プッシュダウンオートマトンへ III

$\mathcal{K}$ は，文形式の接頭辞 x を処理し終わったとき，接尾辞 α をスタックに保持している．形式的には，以下の主張を示せる．

主張 1. 最左導出により $\sigma \overset{*}{\underset{\mathcal{G}}{\Longrightarrow}} x\alpha$ となるための必要十分条件は，

$$(q, x, \sigma) \overset{*}{\underset{\mathcal{K}}{\longrightarrow}} (q, \lambda, \alpha) .$$

$\mathcal{CF}$ から プッシュダウンオートマトンへ IV

十分性の証明から始めよう．数学的帰納法を用いる．すなわち，

$(q, x, \sigma) \xrightarrow[\mathcal{K}]{i} (q, \lambda, \alpha)$ を仮定して，$\sigma \overset{*}{\underset{\mathcal{G}}{\Longrightarrow}} x\alpha$ を示す．帰納法の基

底として，$i = 0$，すなわち $(q, x, \sigma) \xrightarrow[\mathcal{K}]{0} (q, \lambda, \alpha)$ を仮定する．

反射推移的閉包 $\xrightarrow[\mathcal{K}]{*}$ の定義より，これは $(q, x, \sigma) = (q, \lambda, \alpha)$ に

他ならない．したがって，$x = \lambda$ かつ $\alpha = \sigma$ である． 反射推移的

閉包 $\overset{*}{\underset{\mathcal{G}}{\Longrightarrow}}$ の定義より明らかに $\sigma \overset{*}{\underset{\mathcal{G}}{\Longrightarrow}} \sigma$ であることがわかる．

以上で帰納法の基底が示された．

$\mathcal{CF}$ から プッシュダウンオートマトンへ V

帰納ステップとして，$i \geqslant 1$ を仮定し，$x = ya$(ただし $y \in T^*$) とする. ここで，最後から 2 番めのステップを考える．すなわち，

$$(q, ya, \sigma) \xrightarrow[\mathcal{K}]{i-1} (q, a, \beta) \xrightarrow[\mathcal{K}]{1} (q, \lambda, \alpha)\,. \tag{1}$$

もしも，列 (1) の冒頭 i の時点表示において，入力文字列の末尾

から a を削除したとすると，$(q, y, \sigma) \xrightarrow[\mathcal{K}]{i-1} (q, \lambda, \beta)$ となることが

わかる．なぜならば，a が入力文字列から実際に読み込まれるまでは，$\mathcal{K}$ の動作には影響を与えないからである．これに帰納法の仮定を適用すると，以下が得られる．

$$\sigma \overset{*}{\underset{\mathcal{G}}{\Longrightarrow}} y\beta$$

$\mathcal{CF}$ から プッシュダウンオートマトンへ VI

プッシュダウンオートマトン $\mathcal{K}$ が a を読み込む間に，

$(q, a, \beta) \xrightarrow[\mathcal{K}]{1} (q, \lambda, \alpha)$ という動作をすることを考えると，定義より直ちに，ある $A \in N, (A \to a\eta) \in P, \alpha = \eta\gamma$ に対して，
$\beta = A\gamma$ となることがわかる．
したがって，以下にたどり着く．

$$\sigma \overset{*}{\underset{\mathcal{G}}{\Longrightarrow}} y\beta \overset{1}{\underset{\mathcal{G}}{\Longrightarrow}} ya\eta\gamma = x\alpha\,.$$

これで十分性の証明が完了した．

$\mathcal{CF}$ から プッシュダウンオートマトンへ VII

必要性を証明するために，最左導出で $\sigma \overset{i}{\underset{\mathcal{G}}{\Longrightarrow}} x\alpha$ となることを仮定しよう．我々は i に関する数学的帰納法により，

$(q, x, \sigma) \overset{*}{\underset{\mathcal{K}}{\longrightarrow}} (q, \lambda, \alpha)$ を証明する．帰納法の基底は，やはり $i = 0$ として，先程と同様に示せる．

$\mathcal{CF}$ から プッシュダウンオートマトンへ VIII

帰納ステップとして，$i \geqslant 1$ として，以下を仮定する．

$$\sigma \overset{i-1}{\underset{\mathcal{G}}{\Longrightarrow}} yA\gamma \overset{1}{\underset{\mathcal{G}}{\Longrightarrow}} ya\eta\gamma\,,\ \text{ただし}\ x = ya,\ \alpha = \eta\gamma\,.$$

帰納法の仮定より直ちに

$$(q, y, \sigma) \xrightarrow[\mathcal{K}]{*} (q, \lambda, A\gamma)$$

となり，$(q, ya, \sigma) \xrightarrow[\mathcal{K}]{*} (q, a, A\gamma)$ である．$(A \to a\eta) \in P$ から，

$(q, \eta) \in \delta(q, a, A)$ が言える．したがって，

$$(q, x, \sigma) \xrightarrow[\mathcal{K}]{*} (q, a, A\gamma) \xrightarrow[\mathcal{K}]{1} (q, \lambda, \alpha)\,,$$

となり 必要性が示された．以上で主張 1 が証明された．

©Thomas Zeugmann

$\mathcal{CF}$ から プッシュダウンオートマトンへ IX

定理の証明を完成させるためには，$\alpha = \lambda$ に対する主張 1

$$\sigma \overset{*}{\underset{\mathcal{G}}{\Longrightarrow}} x \text{ の必要十分条件は } (q, x, \sigma) \overset{*}{\underset{\mathcal{K}}{\longrightarrow}} (q, \lambda, \lambda)\,.$$

を書くだけでよい．すなわち，$x \in L(\mathcal{G})$ の必要十分条件は $x \in N(\mathcal{K})$ である．

主定理

ここで，すべてをまとめると，直ちに以下が得られる．

定理 3

L を任意の言語とする．次の 3 つの主張はすべて等価である．

(1) $L \in \mathcal{CF}$.

(2) $L = L(\mathcal{K}_1)$ となるようなプッシュダウンオートマトン $\mathcal{K}_1$ が存在する．

(3) $L = N(\mathcal{K}_1)$ となるようなプッシュダウンオートマトン $\mathcal{K}_1$ が存在する．

©Thomas Zeugmann

所見 I

ここまで講義を進めてきたあとで，次の問いを考えてみよ．すなわち，任意の 2 つの文脈自由文法 $\mathcal{G}_1$, $\mathcal{G}_2$ が与えられたとき，$L(\mathcal{G}_1) \cap L(\mathcal{G}_2) = \emptyset$ であるか否かを判定したい．

©Thomas Zeugmann

所見 I

ここまで講義を進めてきたあとで，次の問いを考えてみよ．すなわち，任意の 2 つの文脈自由文法 $\mathcal{G}_1$, $\mathcal{G}_2$ が与えられたとき，$L(\mathcal{G}_1) \cap L(\mathcal{G}_2) = \emptyset$ であるか否かを判定したい．

あなたは，この作業はそれほど難しくないと思うかも知れない．しかし現実には，すべての文脈自由文法に対してこの問題を解くアルゴリズムを作ることには，これまでに誰も成功していない．おそらくこの問題の背後にもっと深い理由がある．このような問題を扱うためには，その前に，計算可能性という概念を扱わねばならない．

©Thomas Zeugmann

所見 II

一方，我々はこれまでに，正規言語と文脈自由言語について勉強をした．しかし，文脈自由でない言語もすでに見たことがある．すなわち，

$$L = \{a^n b^n c^n \mid n \in \mathbb{N}\}.$$

他にどんな言語族があるのかを質問したくなるのは，自然なことである．時間の制約から，形式言語理論のこれらの部分の概略だけを示すことにする．

©Thomas Zeugmann

文脈依存言語 I

まず，形式的な定義を与える．

定義 2

文法 $\mathcal{G} = [T, N, \sigma, P]$ が，そのすべての生成規則が以下の条件を満たすとき，文脈依存 *(context-sensitive)* 文法であると言う．

(1) $(\alpha \rightarrow \beta) \in P$ という形であって，s_1, s_2, r, h が存在し，それらが $h \in N$, $s_1, s_2 \in (T \cup N)^*$ かつ $r \in (T \cup N)^+$ かつ $\alpha = s_1 h s_2$ かつ $\beta = s_1 r s_2$ を満たすこと，または

(2) $\alpha = h$ かつ $\beta = \lambda$ であって h が P のどの生成規則の右辺にも出現しないこと．

文脈依存言語 II

定義 3

$L = L(\mathcal{G})$ となるような文脈依存文法 $\mathcal{G}$ が存在するならば，言語 L は 文脈依存 ***(context-sensitive)*** 言語であると言う．

©Thomas Zeugmann

文脈依存言語 II

定義 3

$L = L(\mathcal{G})$ となるような文脈依存文法 $\mathcal{G}$ が存在するならば，言語 L は 文脈依存 ***(context-sensitive)*** 言語であると言う．

$\mathcal{CS}$ という記号で，すべての文脈依存言語の族を表現する．文脈依存という名前は非常に直感的である．なぜなら非終端記号の置換（または書き換え）が，接頭辞 s_1 と接尾辞 s_2 により表されたある文脈の中においてのみ可能だからである．上記の定義から直ちに次の観察が得られる．

$\mathcal{CF} \subseteq \mathcal{CS}$.

文脈依存言語 II

定義 3

$L = L(\mathcal{G})$ となるような文脈依存文法 $\mathcal{G}$ が存在するならば，言語 L は **文脈依存 *(context-sensitive)*** 言語であると言う．

$\mathcal{CS}$ という記号で，すべての文脈依存言語の族を表現する．文脈依存という名前は非常に直感的である．なぜなら非終端記号の置換（または書き換え）が，接頭辞 s_1 と接尾辞 s_2 により表されたある文脈の中においてのみ可能だからである．上記の定義から直ちに次の観察が得られる．

$\mathcal{CF} \subseteq \mathcal{CS}$.

演習: $\mathcal{CF} \subset \mathcal{CS}$ を証明せよ．

文脈依存言語 III

$\mathcal{REG}$ や $\mathcal{CF}$ で行ったのと同様のアイデアで，以下が示せる．

定理 4

文脈依存言語は，和集合演算，直積集合演算，クリーニ閉包演算に関して閉じている．

(証明) この証明は演習問題として残しておく．

©Thomas Zeugmann

文脈依存言語 III

$\mathcal{REG}$ や $\mathcal{CF}$ で行ったのと同様のアイデアで，以下が示せる．

定理 4

文脈依存言語は，和集合演算，直積集合演算，クリーニ閉包演算に関して閉じている．

(証明) この証明は演習問題として残しておく．

また，定理 6.4 で示したのと同様の方法で，文脈依存言語が転置演算に関して閉じていることが証明できる．すなわち，次の定理が得られる．

©Thomas Zeugmann

文脈依存言語 III

$\mathcal{REG}$ や $\mathcal{CF}$ で行ったのと同様のアイデアで，以下が示せる．

定理 4

文脈依存言語は，和集合演算，直積集合演算，クリーニ閉包演算に関して閉じている．

(証明) この証明は演習問題として残しておく．

また，定理 6.4 で示したのと同様の方法で，文脈依存言語が転置演算に関して閉じていることが証明できる．すなわち，次の定理が得られる．

定理 5

Σ を任意のアルファベットとし，$L \subseteq \Sigma^*$ とする．このとき，$L \in \mathcal{CS}$ ならば $L^T \in \mathcal{CS}$ もまた成り立つ．

©Thomas Zeugmann

文脈依存言語 IV

さらに，文脈自由言語と異なり，次が成り立つ．

定理 6

文脈依存言語は，共通集合演算に関して閉じている．

文脈依存言語 IV

さらに，文脈自由言語と異なり，次が成り立つ．

定理 6

文脈依存言語は，共通集合演算に関して閉じている．

文脈依存言語のさらに多くの性質を明らかにするため，以下の定義が必要である．

定義 4

ある文法 $\mathcal{G} = [T, N, \sigma, P]$ において，各々の生成規則 $(\alpha \rightarrow \beta) \in P$ に対して $|\alpha| \leqslant |\beta|$ が成り立つとき，$\mathcal{G}$ は**非縮小文法 *(length-increasing)*** であると言う．さらに，P は生成規則 $\sigma \rightarrow \lambda$ を含んでも良いが，その場合は σ は P のどの生成規則の右辺にも出現してはならない．

文脈依存言語 V

文脈依存文法の定義から，直ちに次の系が得られる．

系 7

すべての文脈依存文法は非縮小文法である．

©Thomas Zeugmann

文脈依存言語 V

文脈依存文法の定義から，直ちに次の系が得られる．

系 7

すべての文脈依存文法は非縮小文法である．

しかし，非縮小であっても文脈依存でない文法が存在することが示せる (後述).

©Thomas Zeugmann

文脈依存言語 V

文脈依存文法の定義から，直ちに次の系が得られる．

系 7

すべての文脈依存文法は非縮小文法である．

しかし，非縮小であっても文脈依存でない文法が存在することが示せる (後述)．

それにも関わらず，次の定理を示せる．

定理 8

すべての非縮小文法に対して，それと等価な文脈依存文法が存在する．

©Thomas Zeugmann

文脈依存言語 VI

以上をまとめると，直ちに以下の等価性が得られる．

定理 9

L を任意の言語とする．このとき，以下の 2 つの言明は等価である：

(1) $L = L(\mathfrak{G})$ となるような文脈依存文法 $\mathfrak{G}$ が存在する．

(2) $L = L(\tilde{\mathfrak{G}})$ となるような非縮小文法 $\tilde{\mathfrak{G}}$ が存在する．

©Thomas Zeugmann

文脈依存言語 VII

例. 1

T を任意のアルファベットとする. 文法 $\mathcal{G} = [T, N, \sigma, P]$ を以下の通り定義する.

$$N = \{\sigma\} \cup \{X_i \mid i \in T\} \cup \{A_i \mid i \in T\} \cup \{B_i \mid i \in T\}$$

として, P を以下のような生成規則の集合とする.

1. $\sigma \to i\sigma X_i$
2. $\sigma \to A_i B_i$
3. $B_i X_j \to X_j B_i$
4. $B_i \to i$
5. $A_i X_j \to A_i B_j$
6. $A_i \to i$

ただし $i, j \in T$ である.

©Thomas Zeugmann

文脈依存言語 VIII

生成規則を調べると，$\mathcal{G}$ は非縮小文法であるが，文脈依存文法ではない．なぜならば，生成規則 3 で文脈が崩れているからである．

しかしながら，先ほどの定理より，$L(\mathcal{G})$ は文脈依存言語であることがわかっている．

文脈依存言語 VIII

生成規則を調べると，$\mathcal{G}$ は非縮小文法であるが，文脈依存文法ではない．なぜならば，生成規則 3 で文脈が崩れているからである．

しかしながら，先ほどの定理より，$L(\mathcal{G})$ は文脈依存言語であることがわかっている．

演習: 上の例で与えられた文法 $\mathcal{G}$ が次の言語を生成することを証明せよ．

$$L = \{ww \mid w \in T^+\}$$

文脈依存言語 IX

非縮小文法の概念により，次に述べるように，もう 1 つの好ましい性質を導ける．

定理 10

任意の文脈依存文法 $\mathcal{G} = [T, N, \sigma, P]$ と任意の文字列 $s \in T^*$ を入力として，$s \in L(\mathcal{G})$ であるかどうかを判定するアルゴリズムが存在する．

©Thomas Zeugmann

文脈依存言語 X

証明の概略: すべての文脈依存文法は非縮小文法でもあるから，$|w_i| < |w_{i+1}|, i = 0, \ldots, n-1$ かつ $\sigma = w_0$ かつ $w_n = s$ (ただし $w_i \in (T \cup N)^+$) であるようなすべての有限系列 $w_0, w_1, \ldots, w_n$ について検査すれば十分である．そのような系列の総数は有限個である．そのような系列の集合を S とする．

©Thomas Zeugmann

文脈依存言語 X

証明の概略: すべての文脈依存文法は非縮小文法でもあるから，$|w_i| < |w_{i+1}|, i = 0, \ldots, n-1$ かつ $\sigma = w_0$ かつ $w_n = s$ (ただし $w_i \in (T \cup N)^+$) であるようなすべての有限系列 $w_0, w_1, \ldots, w_n$ について検査すれば十分である．そのような系列の総数は有限個である．そのような系列の集合を S とする．

すると，検査すべきことは

$$w_i \overset{1}{\underset{G}{\Longrightarrow}} w_{i+1} \quad \text{for all } i = 0, \ldots, n-1 \tag{2}$$

であるかどうかだけである．よって，もしも S の中に (2) を満たす系列が 1 つでも見つかれば，直ちに $s \in L(G)$ とわかる．もしも S のすべての系列が (2) を満たせなければ，$s \notin L(G)$ である． ∎

文脈依存言語 XI

我々は以前に，正規言語は補集合演算について閉じているが，文脈自由言語は補集合演算について閉じていないことを証明した．同じように，文脈依存言語が補集合演算について閉じているかどうかを知るのは興味深いことである．

©Thomas Zeugmann

文脈依存言語 XI

我々は以前に，正規言語は補集合演算について閉じているが，文脈自由言語は補集合演算について閉じていないことを証明した．同じように，文脈依存言語が補集合演算について閉じているかどうかを知るのは興味深いことである．

この問いに答えるのは，たやすいことではない．実際，この問題が解かれるまでに 20 年の歳月がかかった．この答えは「計算量と暗号理論」の講義で述べる予定なので，それまでしばらく我慢しよう．

©Thomas Zeugmann

その先 I

最後に，もしも生成規則の集合に対して何の制約も与えなかったら何が起こるのかを述べておこう．我々はそのような言語の族を $\mathcal{L}_0$ と呼ぶ．ここで 0 は，制約がゼロであるという意味である．

その先 I

最後に，もしも生成規則の集合に対して何の制約も与えなかったら何が起こるのかを述べておこう．我々はそのような言語の族を $\mathcal{L}_0$ と呼ぶ．ここで 0 は，制約がゼロであるという意味である．

このとき，明らかに，$\mathcal{L}_0$ に含まれるすべての言語 L は，次の共通する性質を持つ．すなわち，もしも L を表す文法 $\mathcal{G}$ が与えられたならば，L の要素をすべて，かつ，それらだけをすべて列挙していくことができる．しかし，後に見るように，今回説明した最後の定理は，$\mathcal{L}_0$ には一般化できない．これより直ちに，$\mathcal{CS} \subset \mathcal{L}_0$ が言える．

©Thomas Zeugmann

その先 II

以上すべてをまとめると，有名なチョムスキー階層 (Chomsky Hierarchy) が得られる．すなわち

$$\mathcal{REG} \subset \mathcal{CF} \subset \mathcal{CS} \subset \mathcal{L}_0 \,.$$

©Thomas Zeugmann

Thank you!

©Thomas Zeugmann