情報処理学会論文誌　プログラミング　Vol.7　No.4　8–20 (Aug. 2014)

スタック長の特徴付けによる言語の非DCFL性証明

上里 友弥[1,a)]　南出 靖彦[1,b)]

受付日 2014年1月14日, 採録日 2014年4月18日

概要：本論では，言語が決定性文脈自由言語（DCFL）ではないことを証明するための，決定性プッシュダウンオートマトン（DPDA）に基づく手法を提案する．提案する手法は，計算中でのスタックの高さを特徴付けるもので，以下のようなスタックの変化に対する直感を形式的に扱うことができる：言語 $\{a^n b^n c^n \mid n \geq 1\}$ が何らかの DPDA で受理できたとすると，a の個数と b の個数を比較した結果として，$a^n b^n$ の部分を読み終わった段階のスタックはほとんど空になっているはずである．このとき，続く文字列 c^n を検査することができない．したがって，この言語は DCFL ではない．具体的には，十分に長い繰返し文字列を消費した結果のスタックの内容には繰返し構造があることを示し，繰返し文字列を消費した結果のスタックが一般化順序機械と呼ばれる機械で定義可能であることを示した．この結果を用いて，上の例のように a の個数と b の個数を比較するためには，スタックがほとんど空にならなければならないことを形式化して示した．

キーワード：ポンピング補題，プッシュダウンオートマトン，決定性文脈自由言語

A Characterization of Stack for Proofs of Non-DCFL

Yuya Uezato[1,a)]　Yasuhiko Minamide[1,b)]

Received: January 14, 2014, Accepted: April 18, 2014

Abstract: We propose a new technique based on DPDA (deterministic pushdown automata) for proving that a language is not a DCFL (deterministic context-free language). The following intuitive discussion can be conducted in a formal manner by our technique. If $\{a^n b^n c^n \mid n \geq 1\}$ is accepted by a DPDA, the content of the stack after reading $a^n b^n$-part of the input should be almost empty because b^n is matched against a^n. Then, the DPDA cannot correctly check the rest of the input, c^n. We characterize the content of the stack after reading a sufficiently long repetition of a string and, show that the content of the stack after reading a repeated string is definable by a generalized sequential machine. By using this characterization, we prove that the stack must be almost empty after matching the substring in the input.

Keywords: pumping lemma, pushdown automata, deterministic context-free languages

1. はじめに

本論では，ある言語が決定性文脈自由言語（DCFL）ではないことを証明するための，決定性プッシュダウンオートマトン（DPDA）に基づいた手法を提案する．技術的には，DPDA の計算中におけるスタックの高さに対する特徴付けを行い，文字列の受理判定のために本質的にマッチングを行う必要がある（と思われる）例について，マッチング終了後には，スタックからマッチングで用いられた情報が本質的に失われていることを形式的に示す．このことは，簡単な例を用いて説明すると分かりやすい．言語 $L_1 = \{a^n b^n \mid n \geq 1\}$ は DCFL として知られているが，文字列 $a^i b^j$ について，その前半部と後半部でのマッチによって i と j の比較を行い，その結果として各文字列の受理，非受理が決定される，と直感的には考えられる．したがって，a^i 部を読み終わった時点でのスタックと比べ，b^j 部を読み終わった時点でのスタックは i に相当する量が減少しているはずである．実際，提案する手法ではこの直感を裏

[1] 筑波大学システム情報工学研究科
Department of Computer Science, University of Tsukuba, Tsukuba, Ibaraki 305–8573, Japan
[a)] uezato@score.cs.tsukuba.ac.jp
[b)] minamide@cs.tsukuba.ac.jp

© 2014 Information Processing Society of Japan

付けるような結果，すなわち，言語 L_1 を受理する任意の（正規化がなされた）DPDA は，$a^n b^n$ を読み終わった段階では本質的にスタックを空にしている，ということを証明することができる．

この種のスタックが本質的に空になるという議論を用いると，ある言語が DCFL ではないことを直感的に証明することができる．たとえば，言語 $L_2 = \{a^n b^n c^n \mid n \geq 1\}$ について考える．これは文脈自由言語でさえない例としてよく知られているものである．文字列 $a^i b^j c^k$ の受理，非受理を決定するためには，すなわち $i = j = k$ となっていることを確かめるためにマッチングを避けられない，という直感がある．もし L_2 を受理する DPDA M が存在したとすると，我々の手法を用いると，語 $a^i b^j c^k$ について $a^i b^j$ 部を読み終わった段階でのスタックはほとんど空になってしまっていることが分かる．このとき，ほとんど空になってしまった（すなわち i の情報が残されていない）スタックを用いて，ちょうど c^k が $i = k$ であるかどうかを検査することはできないことから，L_2 が DCFL ではないことが示される．

本論の構成は以下のとおりである．はじめに，2 章で準備として DPDA を定義する．特にここで導入する DPDA の正規形は，DPDA を用いた計算の本質をとらえたようなものであり，本論全体でも正規形だけを考える．続く 3 章で，DPDA が計算を行う対象である入力に規則性があるとそれを読んだ結果得られるスタックにも規則性が生じることを示す．さらに，これを用いて，文字列の繰返しを消費した結果として得られるスタックが，一般化順序機械と呼ばれる形式言語の道具によって定義可能であることを示す．4 章で，この章でも述べたような，文字列に対するマッチングが行われるような場合にはスタックが本質的に使われ，特に $a^n b^n$ を調べる場合には読み終わったあとにはほとんどスタックが空になっているということが，3 章の内容を用いて形式化され，証明される．最後に 5 章において，いくつかの DCFL ではない言語に対して 4 章で得られた結果を用いることで，直感的にその非 DCFL 性を証明する．

1.1 関連研究

文法および導出木をもとにした証明に基づく DCFL に対する反復補題としては，Harrison による iteration theorem [1] や，LR(k) 文法の理論に基づく Yu によるもの [2] がある．一方，DPDA をもとにした（Harrison によるものより弱い形の）反復補題の証明については Kanazawa によるもの [3] がある．

DPDA を用いたスタックの高さに対して着目した非 DCFL 性の証明手法として，Li らによる Kolmogorov 複雑さを用いた手法 [4] がある．ただし，彼らの論文 [4] における証明には明らかな誤りがあり，のちに Glier によって反例が提示され，同時に修正が行われた [5]．我々の提案する手法は，Kolmogorov 複雑さの議論を用いることなく，Glier の行った証明と比べて，いっそうスタックの変化量に対して具体的かつ精緻な特徴付けを行った．結果として，Li らの論文で Future work としてあげられており，Glier による修正された結果で扱うことのできていなかった，言語 $\{a^n b^n c^n \mid n \geq 1\}$ の非 DCFL 性を証明することが可能となった．

技術的に類似した仕事として，1960 年代の Ginsburg らによる結果がある [6]．彼らの論文では，本論文で示した補題 1 と同等の補題 Lemma 4.1 がすでに示されており，これを用いた言語 $L = \left\{a^n b^n, a^n b^{2n} \mid n \geq 1\right\}$ の非 DCFL 性証明についても，本論文における証明と本質的に似た議論をすでに行っている．したがって，補題 1 と当該言語 L の非 DCFL 性証明は，本研究での新たな結果ではない．むしろ，我々は文献 [6] の Lemma 4.1 をさらに推し進め，マッチングによるスタックの消費が本質的に避けられないという，直感的に分かりやすい性質を形式化・証明し，さらに到達可能性解析の考え方を組み合わせることで，より広く適用可能な手法へと発展させることに成功したといえる．

2. 準備

定義 1（決定性プッシュダウンオートマトン（DPDA））．DPDA M は，構造 $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ である．ただし，Q は状態の有限集合，Σ は有限の入力アルファベット，Γ はスタック記号の有限集合，$q_0 \in Q$ は初期状態，$Z_0 \in \Gamma$ はスタックの底を表す記号，$F \subseteq Q$ は最終状態の集合である．δ は遷移関数で，$\delta : Q \times \Gamma \to (\Sigma \to Q \times \Gamma^*) \uplus (\{\varepsilon\} \to Q \times \Gamma^*)$ の型で定義される．遷移関数は DPDA が決定的に，かつつねに次へと遷移できるように以下の 2 つを満たしている：

- $\delta(q, \gamma, \sigma)$ か $\delta(q, \gamma, \varepsilon)$ のどちらかでしか定義されない．両方で定義されていると，入力 σ を読む場合と読まない場合で非決定的に遷移できてしまう．
- σ と ε のどちらで定義されるにしても，遷移先の状態とスタックへの操作がただ 1 つに定まる．

また，δ にはスタック底記号 Z_0 が本質的に取り除かれることはないという制約をつける．すなわち Z_0 が取り除かれるときには，ただちにスタックの底に戻すように次のように定義されている：

$$\delta(p, Z_0, a) = \langle q, Z_0 \gamma_1 \ldots \gamma_n \rangle .$$

■

入力文字を消費しない ε 遷移関係 $\xrightarrow{\varepsilon} \subseteq \mathit{Conf}(M) \times \mathit{Conf}(M)$ と，入力文字を消費する Σ 遷移関係 $\xrightarrow{\sigma} \subseteq \mathit{Conf}(M) \times \mathit{Conf}(M)$ を以下で定義する．ただし，$\mathit{Conf}(M)$ は，DPDA M 上の計算状況全体の集合 $\mathit{Conf}(M) = Q \times \Gamma^*$ である：

$$\langle p, \xi\, \gamma \rangle \xrightarrow{\varepsilon} \langle q, \xi\, \zeta \rangle \iff \delta(p, \gamma, \varepsilon) = \langle q, \zeta \rangle$$

© 2014 Information Processing Society of Japan

$$\langle p, \xi\,\gamma\rangle \xrightarrow{\sigma} \langle q, \xi\,\zeta\rangle \iff \delta(p,\gamma,\sigma) = \langle q, \zeta\rangle$$

すなわち，$\langle p, \xi\,\gamma\rangle \xrightarrow{a} \langle q, \xi\,\zeta\rangle$ は，状態を p から q に変えつつ，スタックのトップにある記号 $\gamma \in \Gamma$ を pop し，新たに記号列 $\zeta \in \Gamma^*$ を push するものである．ある計算状況 $\mathbf{c}$ で Σ 遷移が定義されている，すなわち $\mathbf{c} \xrightarrow{\sigma} \mathbf{c}'$ とする $\sigma \in \Sigma$ と $\mathbf{c}'$ が存在するとき，$\mathbf{c}$ は reading mode にいるといい，特にこのことを read-mode($\mathbf{c}$) と書くことにする．

1 ステップの遷移関係の合成により，以下のような与えられた文字 $\sigma \in \Sigma$ ないし空でない文字列 $w \in \Sigma^+$ を使いきる複数ステップの遷移関係を定義する：

$$\begin{aligned}
\mathbf{c}_0 &\overset{\varepsilon}{\Rightarrow} \mathbf{c}_0 \iff && \text{read-mode}(\mathbf{c}_0),\\
\mathbf{c}_0 &\overset{\sigma}{\Rightarrow} \mathbf{c}_n \iff && \mathbf{c}_0 \xrightarrow{\sigma} \mathbf{c}_1 \xrightarrow{\varepsilon} \mathbf{c}_2 \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} \mathbf{c}_n \wedge\\
& && \text{read-mode}(\mathbf{c}_n),\\
\mathbf{c}_0 &\overset{w}{\Rightarrow} \mathbf{c}_n \iff && \mathbf{c}_0 \xrightarrow{w_{[1]}} \mathbf{c}_1 \xrightarrow{w_{[2]}} \cdots \xrightarrow{w_{[|w|]}} \mathbf{c}_n.
\end{aligned}$$

ただし，文字列 w に対して $|w|$ はその長さを表し，任意の $1 \le i \le |w|$ について $w_{[i]}$ は w の i 番目の文字を表す．また，文字列 $w = w_{[1]}w_{[2]}\ldots w_{[|w|]}$ と 2 変数 $1 \le i \le j \le |w|$ について，$w_{[i:j]} = w_{[i]}\ldots w_{[j]}$ は i 文字目から j 文字目までの部分文字列の切り出しを表す．$w_{[1:|w|]} = w$ が成立する．文字列の末尾から長さ ℓ の部分文字列を切り出すことを表すのに $w_{[-\ell:]}$ を用いる．これは $w_{[-\ell:]} = w_{[(|w|-\ell)+1:|w|]}$ として定義される．また，w から末尾 $w_{[-\ell:]}$ を取り去って先頭から長さ $|w| - \ell$ の部分文字列の切り出しを表すのに $w_{[:-\ell]}$ を用いる．これは $w_{[:-\ell]} = w_{[1:(|w|-\ell)]}$ で定義される．一般に $1 \le \ell \le |w|$ について $w = w_{[:-\ell]}w_{[-\ell:]}$ が成立する．

文字列 w の反転を $w^R = w_{[|w|]}\ldots w_{[2]}w_{[1]}$ で定義する．

$\mathrm{next}_w(i)$ は以下のように定義される関数である：

$$\mathrm{next}_w : \{1,\ldots,|w|\} \to \{1,\ldots,|w|\}$$
$$\mathrm{next}_w(i) = \begin{cases} 1 & \textit{if } i = |w| \\ i+1 & \text{otherwise} \end{cases}$$

w が明らかな場合には next_w のことを単に next と書く．

文字列 x が y の真の接頭辞であることを $x \prec y$ で表し，

$$x \prec y \iff \exists w \in \Sigma^+.\, xw = y$$

で定義し，これを用いて $x \preceq y \iff x \prec y \vee x = y$ も同時に定義しておく．

たとえば $w = \sigma_1\sigma_2\sigma_3\sigma_4\sigma_5$ では，$w_{[-2:]} = w_{[4:5]} = \sigma_4\sigma_5$ であり，$w_{[:-2]} = w_{[1:3]} = \sigma_1\sigma_2\sigma_3$ となる．$w^R = \sigma_5\sigma_4\sigma_3\sigma_2\sigma_1$ であり，$w_{[\mathrm{next}(|w|):2]} = \sigma_1\sigma_2$ となる．

計算状況に対する記法をいくつか導入する：

- 計算状況 $\langle p, \xi\rangle$ に対して，$|\langle p, \xi\rangle|$ をそのスタックの高さ $|\xi|$ として定義する．
- 計算状況 $\langle p, \xi\rangle$ と状態の集合 $S \subseteq Q$ について，$\langle p, \xi\rangle \in S \iff p \in S$ と定義する．

入力を読みきったときの遷移先が一意に決まることを利用して，$M(\mathbf{c}, w) = \mathbf{c}' \iff \mathbf{c} \overset{w}{\Rightarrow} \mathbf{c}'$ という遷移先を求める関数を定義しておく．初期計算状況 $\langle q_0, Z_0\rangle$ から始める場合を，特に $M(w) = M(\langle q_0, Z_0\rangle, w)$ と書く．

M の計算状況 $\mathbf{c}$ から，入力 w 文字を読んだ後に状態の集合 $S \subseteq Q$ に入ることを示す述語 $\mathcal{V}\mathrm{isit}_M$ を導入する．正確には以下で定義される：

$$\mathcal{V}\mathrm{isit}_M(\mathbf{c}, w, S) \iff \exists \mathbf{c}_f.\, M(\mathbf{c}, w) \xrightarrow{\varepsilon}{}^* \mathbf{c}_f \wedge \mathbf{c}_f \in S$$

文脈から M が明らかな場合には $\mathcal{V}$isit とだけ記述する．M が受理する言語 $\mathcal{L}(M) \subseteq \Sigma^*$ は $\mathcal{V}$isit を用いて次で定義される：

$$w \in \mathcal{L}(M) \iff \mathcal{V}\mathrm{isit}(\langle q_0, Z_0\rangle, w, F).$$

すべての DPDA M は以下の命題に示すような扱いやすい等価な正規形 N に，構成的に変換することができる．特に本論では，DPDA として正規形だけを考えることにする．

命題 1.

(1) 必ず次の文字を読むようにできる．すなわち，ε 遷移だけで無限に計算を続けることはない．

(2) ε 遷移では pop だけができる．すなわち，δ を以下のように制限できる：

$$\delta : Q \times \Gamma \to (\Sigma \to Q \times \Gamma^*) \uplus (\{\varepsilon\} \to Q \times \{\varepsilon\})$$

(3) さらに Σ 遷移についても高さはたかだか 1 つしか変えないように制限できる：

$$\delta : Q \times \Gamma \to (\Sigma \to Q \times \Gamma^{\le 2}) \uplus (\{\varepsilon\} \to Q \times \{\varepsilon\})$$

ただし，$\Gamma^{\le n} = \bigcup_{i=0}^{n} \Gamma^i$ である[*1]．

証明のアイディア．任意の DPDA M から，(1) を満たす等価な DPDA M_1 が構成できることはよく知られている [7]．M_1 から (2) を満たす等価な DPDA を構成する際には，M_1 の連続する ε 遷移でスタックに push する記号列のサイズには上限が存在することを用いる [8]．(2)，(3) については，必要な範囲でスタックを圧縮するような操作を行うことになる．詳細な証明は付録において行う．　□

定義 2（一般化順序機械）．一般化順序機械（generalized sequential machine, GSM）G は，構造 $(Q, \Sigma, \Gamma, \delta, q_0, F)$ であり，有限オートマトンを遷移にともなって文字列を出力できるように一般化したものであるといえる．

Q は状態の有限集合，Σ は有限の入力アルファベット，Γ は有限の出力アルファベット，$q_0 \in Q$ は初期状態，$F \subseteq Q$ は

[*1] 文字列の集合上に拡張した結合演算 $X \cdot Y$ および，それによる冪 X^i と Kleene 閉包 X^* は，次で定義している．

$$X \cdot Y = \{xy \mid x \in X, y \in Y\},$$
$$X^0 = \{\varepsilon\},\quad X^{i+1} = X^i \cdot X,\quad X^* = \bigcup_{i=0}^{\infty} X^i.$$

© 2014 Information Processing Society of Japan

最終状態の集合である．有限集合 $\delta \subseteq_{\mathrm{fin}} (Q\times\Sigma)\times(Q\times\Gamma^*)$ は 1 ステップの遷移関係である．

GSM G 上の計算を，遷移関係 $\xrightarrow[w_{out}]{w_{in}}$ の合成によって定義する：

- $q \xrightarrow[\varepsilon]{\varepsilon} q \quad \forall q \in Q$
- $p \xrightarrow[\xi\cdot\zeta]{\sigma\cdot w} q \iff \langle p,\sigma,\xi,r\rangle \in \delta \wedge r \xrightarrow[\zeta]{w} q$

このとき，GSM G の振舞い $\mathcal{B}_G : \Sigma^* \to \mathcal{P}(\Gamma^*)$ を以下のように定義する：

$$\mathcal{B}_G(w_{in}) = \left\{ w_{out} \mid q_0 \xrightarrow[w_{out}]{w_{in}} f, f \in F \right\}.$$

ただし以降では，$\mathcal{B}_G$ と G を区別せずに用い，$G(w_{in})$ と書いて $\mathcal{B}_G(w_{in})$ を表す．

特に，任意の入力 $w_{in} \in \Sigma^*$ について，$|\mathcal{B}_G(w_{in})| \leq 1$ が成立する GSM G を関数的 GSM（Functional GSM）と呼び，関数的 GSM のクラスを FGSM で表す． ■

3. 関数的 GSM を用いた DPDA の振舞いの特徴付け

本章では，FGSM を用いて，入力が文字列の繰返しで表されるときの DPDA の生成するスタックの内容を特徴付ける．

補題 1. 任意の DPDA M と，空でない文字列 $w \in \Sigma^+$ について，w を十分に長く繰り返したものを入力として与えた場合には，スタックの下部に依存せずに計算を進めることができる．すなわち以下が成立する：

$$\begin{aligned}&\exists 1 \leq i \leq |w|.\ \exists n,m,\gamma,\xi,\zeta.\\ &\quad \langle q_0, Z_0\rangle \xrightarrow{w^n w_{[1:i]}} \langle q, \xi\,\gamma\rangle\\ &\quad \wedge \langle q,\gamma\rangle \xrightarrow{w_{[\mathrm{next}(i):|w|]} w^m w_{[1:i]}} \langle q,\zeta\,\gamma\rangle .\end{aligned}$$

証明．背理法で証明する．すなわち，以下を仮定して矛盾を示す：

$$\begin{aligned}&\forall 1 \leq i \leq |w|.\ \forall n,m,\gamma,\xi,\zeta.\\ &\quad \langle q_0, Z_0\rangle \xrightarrow{w^n w_{[1:i]}} \langle q, \xi\,\gamma\rangle\\ &\quad \Rightarrow \langle q,\gamma\rangle \xrightarrow{w_{[\mathrm{next}(i):|w|]} w^m w_{[1:i]}} \langle q,\zeta\,\gamma\rangle \text{ が成立しない.}\end{aligned} \tag{1}$$

w の繰返しを入力として与えたときに，スタックの高さが 1 となる計算状況に，$\langle q_0, Z_0\rangle$ から $(|Q|\cdot|w|)+1$ 回以上到達することができないことを示す．もしそれが可能だとすると，同じ状態で同じ位置 w_j を読む事態が生じる．すなわち，ある $1 \leq j \leq |w|$，$n,m \in \mathbb{N}$，$q \in Q$ で

$$\begin{aligned}&\langle q_0, Z_0\rangle \xrightarrow{w^n w_{[1:j]}} \langle q, Z_0\rangle\\ &\wedge \langle q, Z_0\rangle \xrightarrow{w_{[\mathrm{next}(j):|w|]} w^m w_{[1:j]}} \langle q, Z_0\rangle\end{aligned}$$

が成立する．しかしこれは (1) に反する．

直前の議論から，w の十分に長い繰返しを入力として与えると，それ以降はスタックの高さがつねに 2 以上になることが分かる．そのようにする入力を w^l と書くことにする．

w^l 以降で，スタックの高さが 2 となる計算状況には $(|Q|\cdot|\Gamma|\cdot|w|)+1$ 回以上は到達してはならないことが示せる．もし到達したとすると，スタックの高さが 2 になると入力を読まなければならないので

$$\begin{aligned}&\langle q_0, Z_0\rangle \xrightarrow{w^l w^n w_{[1:j]}} \langle q, Z_0\,\gamma\rangle\\ &\wedge \langle q, Z_0\,\gamma\rangle \xrightarrow{w_{[\mathrm{next}(j):|w|]} w^m w_{[1:j]}} \langle q, Z_0\,\gamma\rangle\end{aligned}$$

が成立する．特に $\langle q, Z_0\,\gamma\rangle \xrightarrow{w_{[\mathrm{next}(j):|w|]} w^m w_{[1:j]}} \langle q, Z_0\,\gamma\rangle$ について，スタックの高さが 1 になることはないので，$\langle q,\gamma\rangle \xrightarrow{w_{[\mathrm{next}(j):|w|]} w^m w_{[1:j]}} \langle q,\gamma\rangle$ が成立するが，これは (1) に矛盾．

十分に長い w の繰返しを入力として与えると，それ以降はスタックの高さがつねに 3 以上になることが分かる．この議論の繰返しにより，スタックの高さ $h \geq 2$ へは，$(|Q|\cdot|\Gamma|^{h-1}\cdot|w|)+1$ 回以上到達してはならないことが分かる．したがって，任意の h についてスタックの高さがつねに h 以上であるようにできるので，初期計算状況 $\langle q_0, Z_0\rangle$ から，計算状況 $\langle p, s\rangle$（ただし $|Q|\cdot|\Gamma|\cdot|w| < |s|$）へ到達する遷移がある．すなわち，ある $x \in \mathbb{N}$，$1 \leq y \leq |w|$，$q_{\mathcal{T}} \in Q$ を用いて以下の遷移 $\mathcal{T}$ を固定する：

$$\mathcal{T} : \langle q_0, Z_0\rangle \xrightarrow{w^x w_{[1:y]}} \langle q_{\mathcal{T}}, s\rangle .$$

$\mathcal{T}$ について，次の関数を考える：

$$\begin{aligned}&\mathrm{lastvisit} : \{1,\ldots,|s|\} \to \Sigma^*\\ &\mathrm{lastvisit}(h) = w^n w_{[1:i]}\end{aligned}$$

文字列 $w^n w_{[1:i]}$ はスタックの高さが h となる最後の瞬間を表しているとする．正確には以下の状況を意味するものである：

$$\begin{aligned}&\langle q_0, Z_0\rangle \xrightarrow{w^n w_{[1:i]}} \langle q, \xi\,\gamma\rangle\\ &\wedge \langle q,\gamma\rangle \xrightarrow{w_{[\mathrm{next}(i):|w|]} w^{x-n-1} w_{[1:y]}} \langle q_{\mathcal{T}}, s'\rangle .\end{aligned}$$

ただし $h = |\xi\,\gamma|$，$s = \xi\,s'$ である．

正規化済みの DPDA は

- ε 遷移ではスタックの高さを減らすことしかできない，
- Σ 遷移でも高さをたかだか 1 つ増やすことしかできない，

という 2 条件を満たすことから，上の要件を満たす lastvisit が確かに定義されることが分かる．

さて，

© 2014 Information Processing Society of Japan

$$\mathcal{S} = \left\{ \langle i, p, \gamma, n, j\rangle \;\middle|\; \begin{array}{l} 1 \le i \le |s|, \\ \mathrm{lastvisit}(i) = w^n w_{[1:j]}, \\ M(w^n w_{[1:j]}) = \langle p, \xi\,\gamma\rangle \end{array} \right\}$$

として集合 $\mathcal{S}$ を定義すると $|Q|\cdot|\Gamma|\cdot|w| < |\mathcal{S}|$ が成立するので，ある $i_1, i_2 \in \{1,\ldots,|s|\}$，$p \in Q$，$\gamma \in \Gamma$，$n_1, n_2 \in \mathbb{N}$，$j \in \{1,\ldots,|w|\}$ について $\langle i_1, p, \gamma, n_1, j\rangle, \langle i_2, p, \gamma, n_2, j\rangle \in \mathcal{S}$（ただし $i_1 < i_2$）が成立する．このことは遷移 $\mathcal{T}$ 中に

$$\langle q_0, Z_0\rangle \xrightarrow{w^{n_1} w_{[1:j]}} \langle p, \xi\,\gamma\rangle \wedge$$
$$\langle p, \xi\,\gamma\rangle \xrightarrow{w_{[\mathrm{next}(j):|w|]} w^{n_2-n_1-1} w_{[1:j]}} \langle p, \xi\,\zeta\,\gamma\rangle$$

という部分遷移列が存在することにほかならない．特に，lastvisit の定義より

$$\langle p, \xi\,\gamma\rangle \xrightarrow{w_{[\mathrm{next}(j):|w|]} w^{n_2-n_1-1} w_{[1:j]}} \langle p, \xi\,\zeta\,\gamma\rangle$$

の部分では ξ を 1 度もみることはない．したがって，

$$\langle p, \gamma\rangle \xrightarrow{w_{[\mathrm{next}(j):|w|]} w^{n_2-n_1-1} w_{[1:j]}} \langle p, \zeta\,\gamma\rangle$$

が成立するが，これは (1) に矛盾する．　□

定理 1. 任意の DPDA M について，空でない文字列 $w \in \Sigma^+$ の繰返し w^n を入力として，$\langle q_0, Z_0\rangle$ から計算を行ったあとに得られるスタックの内容は，FGSM で定義可能である．すなわち，以下を成立させる FGSM G が存在する：

$$\forall n \ge 1.\, \mathcal{B}_G(w^n) = \left\{ \xi \mid \langle q_0, Z_0\rangle \xRightarrow{w^n} \langle q, \xi\rangle \right\}.$$

証明.　補題 1 により，ある定数 a，b，c について

$$\langle q_0, Z_0\rangle \xrightarrow{w^b\, w_{[1:a]}} \langle q, \xi\,\gamma\rangle \wedge$$
$$\langle q, \gamma\rangle \xrightarrow{w_{[\mathrm{next}(a):|w|]}\, w^c\, w_{[1:a]}} \langle q, \zeta\,\gamma\rangle \qquad (1)$$

が成立する．したがって，式 (1) によって $n \ge 1$ については，一意に定まる l，k（ただし $k \le c$）を用いて以下のように分解できる：

$$w^n = \begin{cases} w^i & \text{if } 1 \le n \le b \\ XY^l V_k & \text{otherwise} \end{cases}$$

ただし上で $w^b\, w_{[1:a]}$ を X，$w_{[\mathrm{next}(a):|w|]}\, w^c\, w_{[1:a]}$ を Y $w_{[\mathrm{next}(a):|w|]}\, w^k$ を V_k として書いた．

ところで式 (1) を用いて，$n \ge 1$ について $M(\langle q_0, Z_0\rangle, w^n)$ の計算は以下のように表現することができる：

$$M(\langle q_0, Z_0\rangle, w^i) = \langle q, \chi_i\rangle \quad \text{if } 1 \le i \le b$$
$$M(\langle q_0, Z_0\rangle, X) = \langle q, \xi\,\gamma\rangle$$
$$M(\langle q_0, Z_0\rangle, XY^l) = M(\langle q, \xi\,\gamma\rangle, Y^l) = \langle q, \xi\,\zeta^l\,\gamma\rangle$$
$$M(\langle q_0, Z_0\rangle, XY^l V_k) = M(\langle q, \xi\,\zeta^l\,\gamma\rangle, V_k) = \langle r, \xi\,\zeta^l\,\varsigma_k\rangle$$

これを用いて以下の FGSM $H_1, \ldots, H_b, I_0, I_1, \ldots, I_c$ を定義する：

$$\mathcal{B}_{H_i} = \{\langle w^i, \chi_i\rangle\},$$
$$\mathcal{B}_{I_k} = \{\langle XY^l V_k, \xi\zeta^l\varsigma_k\rangle \mid l \in \mathbb{N}\}.$$

これらの FGSM を用いて，目的の G を以下で定義する：

$$G = \bigoplus_{i=1}^{b} H_i \oplus \bigoplus_{i=0}^{c} I_i.$$

GSM A，B について，$A \oplus B$ は A と B から得られる GSM で，$\mathcal{B}_{A\oplus B} = \mathcal{B}_A \cup \mathcal{B}_B$ を満たすものとする．事実，そのような $A \oplus B$ は A，B から容易に構成することができる．2 つの FGSM F_1，F_2 の定義域の共通部分が空であれば，$F_1 \oplus F_2$ もまた FGSM となることに注意すると，いま $H_1, \ldots, H_b, I_0, I_1, \ldots, I_c$ の異なるどの 2 つの定義域についても共通部分が空なので G が FGSM であることが分かった．　□

定理 1 は，入力に繰返しがあるときには，DPDA の計算を模倣するのに本質的にスタックを用いる必要はない，ということを述べている．同様の証明を行うことで，入力アルファベットが 1 文字の文脈自由言語は正規言語であるというよく知られた事実 [9] を，決定性文脈自由言語へ制限したものがただちに得られる：

定理 2. $L \subseteq a^*$ かつ $\mathcal{L}(M) = L$ とする DPDA M が存在するとき，L は正規言語である．特に，M から構成的に L を受理する有限オートマトンを作ることができる．

4. DPDA におけるスタックの消費

定理 1 を用いて，スタックの使用量に関する特徴付けをなす．

まず定理のステートメントを記述するために，述語 use と identify を導入する．

$\mathrm{use}(h, \mathbf{c}, w)$ は計算状況 $\mathbf{c}$ から入力を w として計算すると，必ずスタックの高さが h 以下になることがあるということを意味するもので，以下のように定義される：

$$\mathrm{use}(h, \mathbf{c}, w) = \exists w'.\, \exists \mathbf{c}''.\, w' \preceq w \wedge \mathbf{c} \xRightarrow{w'} \mathbf{c}' \xrightarrow{\varepsilon}{}^{*}\, \mathbf{c}'' \wedge |\mathbf{c}''| \le h.$$

$\mathrm{identify}(\mathbf{c}, y, z, n, S)$ は計算状況 $\mathbf{c}$ から入力を yz^i として計算したときに，はじめて S を訪れるのが yz^n を読んだときであることを意味するもので，以下のように定義される：

$$\mathrm{identify}(\mathbf{c}, y, z, n, S) = \mathcal{V}\mathrm{isit}(\mathbf{c}, yz^n, S) \wedge \forall m < n.\, \neg\mathcal{V}\mathrm{isit}(\mathbf{c}, yz^m, S).$$

identify は，マッチングを形式的に定義したものである．$\mathrm{identify}(M(x^n), y, z, n, S)$ は DPDA M で入力 x^n を読みきった計算状況 $M(x^n)$ から，yz^m でさらに計算を続けた

© 2014 Information Processing Society of Japan

ときに，「はじめて」状態集合 S を訪れるのが $m = n$ となることを表し，直感的には，S について x と z でマッチングしている，ということを意味する．

以下の定理 3 は，入力 $x^n yz^n$ において x と z が状態集合 S についてマッチングする際に，その計算途中で必ずスタックの高さを一様な値 h まで減少させる，という内容を FGSM を用いた抽象的な形で表すものである．

定理 3. 任意の DPDA $M = (Q_M, \Sigma, \Gamma, \delta_M, q_0, Z_0, F_M)$ と FGSM $G = (Q_G, \Sigma, \Gamma, \delta_G, g_0, F_G)$ について以下が成立する：

$$\begin{aligned}&\forall S \subseteq Q.\, \forall p \in Q_M.\, \forall x \in \Sigma^+.\, \forall y \in \Sigma^*.\, \forall z \in \Sigma^+.\\ &(\forall n \geq 1.\, G(x^n) \neq \varnothing)\\ &\Rightarrow (\forall n \geq 1.\, \mathrm{identify}(\langle p, G(x^n)\rangle, y, z, n, S))\\ &\Rightarrow \exists h.\, \forall n \geq 1.\, \mathrm{use}(h, \langle p, G(x^n)\rangle, yz^n).\end{aligned}$$

証明．G は出力をするにあたり必ず入力文字を読むので，以下のことが成立する：

$$\exists c.\, \forall n.\, |G(x^n)| \leq c \cdot n.$$

ここで c は，G で $|x|$ 文字読んだ際に引き起こされる出力の最大長を意味している．

はじめに，$m \geq |Q_G|$ とする入力 x^m についての G の計算で，以下を満たすパスが必ず存在することを確認する：

$$\begin{aligned}&g_0 \xrightarrow[v_1]{x^{m_1}} g \xrightarrow[v_2]{x^{m_2}} g \xrightarrow[v_3]{x^{m_3}} g_f \wedge g_f \in F_G\\ &\wedge\, m = m_1 + m_2 + m_3 \wedge m_1 < |Q_G| \wedge m_2 \geq 1\end{aligned} \tag{1}$$

$m \geq |Q_G|$ と仮定の $\forall n \geq 1.\, G(x^n) \neq \varnothing$ により

$$\mathcal{S} = \left\{ \langle i, j\rangle \;\middle|\; \begin{array}{l} i \geq 0, j \geq 1, g \in G,\\ g_0 \xrightarrow{x^i} g \xrightarrow{x^j} g \xrightarrow{x^{m-i-j}} g_f, g_f \in F_G \end{array} \right\}$$

として定義した集合 $\mathcal{S}$ は空ではない．このとき，$\langle i, j\rangle \in \mathcal{S}$ で，$i < |Q_G|$ を満たすものがなければならないので，この i，j を m_1，m_2 としてとればよい．

仮定を導入し，背理法で証明を行う．すなわち，

$$\forall h.\, \exists n \geq 1.\, \neg\mathrm{use}(h, \langle p, G(x^n)\rangle, yz^n) \tag{†}$$

から矛盾を導く．以下で，仮定の次の式を (⋄) として参照する：

$$\forall n > 1.\, \mathrm{identify}(\langle p, G(x^n)\rangle, y, z, n, S). \tag{⋄}$$

まず式 (†) の h を以下を満たすように選ぶ：

$$h > c \cdot |Q_G|.$$

すると，ある $n \geq 1$ について

$$\forall w'.\, \forall \mathbf{c}''.\, (w' \preceq yz^n \wedge \langle q, G(x^n)\rangle \overset{w'}{\Longrightarrow} \mathbf{c}' \overset{\varepsilon}{\rightarrow}{}^* \mathbf{c}'') \Rightarrow |\mathbf{c}''| > h \tag{2}$$

が成立することが，式 (†) から分かる．このことは，$\langle q, G(x^n)\rangle$ からの入力 yz^n を用いた計算では，つねにスタックの高さが h 以上になることを述べている．したがって，$|G(x^n)| > h > c \cdot |Q_G|$ となり $n > |Q_G|$ でなければならないので，式 (1) のように分解することができる：

$$\begin{aligned}&g_0 \xrightarrow[v_1]{x^{n_1}} g \xrightarrow[v_2]{x^{n_2}} g \xrightarrow[v_3]{x^{n_3}} g_f \wedge g_f \in F_G\\ &\wedge\, n = n_1 + n_2 + n_3 \wedge n_1 < |Q_G| \wedge n_2 \geq 1.\end{aligned}$$

特に $g \xrightarrow[v_2]{x^{n_2}} g$ をポンプして，以下の計算を得る：

$$g_0 \xrightarrow[v_1]{x^{n_1}} g \xrightarrow[v_2]{x^{n_2}} g \xrightarrow[v_2]{x^{n_2}} g \xrightarrow[v_3]{x^{n_3}} g_f$$

このとき，$n_1 < |Q_G|$ より $|v_1| < c \cdot |Q_G| < h$ が成立する．

以下を示す：

$$\mathcal{V}\mathrm{isit}(\langle p, G(x^{n+n_2})\rangle, yz^n, S). \tag{⋆}$$

式 (⋄) から $\mathcal{V}\mathrm{isit}(\langle p, G(x^n)\rangle, yz^n, S)$ が成立し，$G(x^n) = v_1v_2v_3$ であるのでこれを $\mathcal{V}\mathrm{isit}(\langle p, v_1v_2v_3\rangle, yz^n, S)$ と改める．次に，式 (2) により $v_1v_2v_3$ の後ろから $|v_1v_2v_3| - h$ 文字だけが計算を行ううえで必要になるので，任意の $\xi \in \Gamma^*$ について $\mathcal{V}\mathrm{isit}(\langle p, \xi\, v_2\, v_3\rangle, yz^n, S)$ となる．このとき $\mathcal{V}\mathrm{isit}(\langle p, v_1v_2v_2v_3\rangle, yz^n, S)$ が成立し，$G(x^{n+n_2}) = v_1v_2v_2v_3$ であるから，(⋆) が示された．

一方で，(⋄) および $n_2 \neq 0$ より $\mathcal{V}\mathrm{isit}(\langle p, G(x^{n+n_2})\rangle, yz^n, S)$ とは**ならない**．このことと式 (⋆) により矛盾が導かれた．□

定理 3 は，FGSM G による出力（すなわち，スタックの内容）が，ほとんどすべて使われることを示している．しかし，マッチング終了時，すなわち $x^n yz^n$ を読み終わった段階でのスタックの高さについては，何の保証も与えていないことに注意されたい．そこで，さらにこの結果を推し進めて，任意の DPDA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ と $S \subseteq Q$ についてのマッチング終了時において，スタックはほとんど空になるという性質を示す．

定理 4（定理 3 の強化）．

$$\begin{aligned}&\forall S \subseteq Q.\, \forall x \in \Sigma^+.\, \forall y \in \Sigma^*.\, \forall z \in \Sigma^+.\\ &(\forall n \geq 1.\, \mathrm{identify}(M(x^n), y, z, n, S))\\ &\Rightarrow \exists h.\, \forall n.\, |M(M(x^n), yz^n)| \leq h\end{aligned}$$

証明．以下で，仮定の次の式を (⋄) として参照する：

$$\forall n \geq 1.\, \mathrm{identify}(M(x^n), y, z, n, S). \tag{⋄}$$

まず定理 3 の G を，定理 1 の FGSM を用いて具体化することで，以下を得る：

$$\exists h.\, \forall n.\, \mathrm{use}(h, M(x^n), yz^n). \tag{1}$$

このことは，任意の n で入力 yz^n についての計算開始

© 2014 Information Processing Society of Japan

時点のスタックの高さが $|M(x^n)| \leq h$ であるか，もしくは $|M(x^n)| > h$ であったとしてもその計算の途中でスタックの高さを必ず h まで減少させることを表している．

証明の本質は，次の主張を示すことにある：

主張.

$$\begin{aligned}&\exists r.\forall n.\\&(|M(x^n)| \leq h \Rightarrow |yz^n| < r) \wedge (|M(x^n)| > h\\&\quad \Rightarrow \forall \alpha \leq |yz^n| - r.\, M(x^n) \xrightarrow{yz^n{}_{[:-(r+\alpha)]}} |\mathbf{c}| > h).\end{aligned}$$

この主張と式 (1) をあわせると，一様な定数 r が存在し，$M(x^n)$ からの計算がスタックの高さを h 以下にするのは，yz^n が残り r 文字になってからということが分かる．一方で，正規化された DPDA では 1 文字を読むにつきたかだか 1 文字しか push できないことから，

$$\forall n.|M(M(x^n), yz^n)| \leq h + r$$

が成立するので証明終．

($\diamond$) より，明らかに以下の集合 X は有限集合である：

$$X = \{n \mid n \geq 1, |M(x^n)| \leq h\}.$$

もし集合 X が無限集合であったとすると，相異なる $n_1 < n_2$ で，$M(x^{n_1}) = M(x^{n_2})$ を満たすものが存在し，かつ

- identify$(M(x^{n_1}), y, z, n_1, S)$
- identify$(M(x^{n_2}), y, z, n_2, S)$

の両方が成立することになるが，これは矛盾している．

ところで，X の有限性により上限が存在することになるが，これによって

$$\exists r.\forall n.\, |M(x^n)| \leq h \Rightarrow |yz^n| < r$$

は成立する．よって

$$\exists r.\forall n \in Y.\forall \alpha \leq |yz^n| - r.\, M(x^n) \xrightarrow{yz^n{}_{[:-(r+\alpha)]}} |\mathbf{c}| > h$$

を示せばよい．ただし $Y = \{n \mid n \geq 1, |M(x^n)| > h\}$ とし，X が有限集合であることから，Y が無限集合となることに注意する．

そこで以下では，これを背理法によって示す．すなわち，次の式から矛盾を導く：

$$\forall r.\exists n \in Y.\exists \alpha \leq |yz^n| - r.\, M(x^n) \xrightarrow{yz^n{}_{[:-(r+\alpha)]}} |\mathbf{c}| \leq h. \tag{2}$$

$n_i, \beta_i \in \mathbb{N}$（$i = 0, 1, \ldots$）を帰納的に定義する：

$i = 0$ の場合：

式 (2) の r を $|yz|$ で具体化し，n_0，α_0 を得る．このとき，$\beta_0 = |yz| + \alpha_0$ と定義する．

$i = j + 1$ の場合：

式 (2) の r を $|yz^{1+\beta_{i-1}}|$ で具体化し，n_i，α_i を得る．

このとき，$\beta_i = |yz^{1+\beta_{i-1}}| + \alpha_i$ と定義する．

任意の i，j についてただちに分かることは以下の 2 つである：

- $i < j$ について $n_i < n_j$ と $\beta_i < \beta_j$ が成立する．
- $\beta_i \leq |yz^{n_i}|$ が成立する．

さらに各 n_i，β_i について，その定義中で式 (2) を具体化していることから以下が成立する：

$$M(x^{n_i}) \xrightarrow{yz^{n_i}{}_{[:-\beta_i]}} |\mathbf{c}_i| \leq h.$$

また，式 ($\diamond$) を n_i で具体化して次を得る：

$$\text{identify}(M(x^{n_i}), y, z, n_i, S).$$

これらのことをまとめると，任意の i について以下が成立することが分かる：

$$\begin{aligned}\exists \mathbf{c}_i''.\ & M(x^{n_i}) \xrightarrow{yz^{n_i}{}_{[:-\beta_i]}} \mathbf{c}_i \xrightarrow{yz^{n_i}{}_{[-\beta_i:]}} \mathbf{c}_i' \xrightarrow{\varepsilon^*} \mathbf{c}_i'' \in S \wedge\\ & |\mathbf{c}_i| \leq h \wedge \mathbf{c}_i' \text{ までは } S \text{ を訪れない.}\end{aligned} \tag{$\star_1$}$$

スタックの高さがたかだか h 以下の計算状況全体の集合 $\mathcal{C} = \{\mathbf{c} \mid \mathbf{c} \in \mathit{Conf}, |\mathbf{c}| \leq h\}$ は有限集合である．したがって，ある $\mathbf{c} \in \mathcal{C}$ について，

$$[\mathbf{c}] = \left\{ i \;\middle|\; M(x^{n_i}) \xrightarrow{yz^{n_i}{}_{[:-\beta_i]}} \mathbf{c} \right\} \tag{$\star_2$}$$

が無限集合となる．このとき，ある $0 \leq d \leq |z| - 1$ について，

$$\text{Mod}_d = \{i \mid i \in [\mathbf{c}], d \equiv \beta_i \bmod |z|\}$$

が無限集合になる．また，Mod_d が無限集合であることから，

$$\begin{aligned}S_1 &= \left\{ i \;\middle|\; i \in \text{Mod}_d, yz^{n_i}{}_{[:-\beta_i]} \preceq y \right\},\\ S_2 &= \left\{ i \;\middle|\; i \in \text{Mod}_d, y \prec yz^{n_i}{}_{[:-\beta_i]} \right\}\end{aligned}$$

のどちらかの集合が無限集合になる．S_2 が無限集合の場合には，相異なる $i, j \in S_2$ を選ぶと以下が成立する：

$$\begin{aligned}yz^{n_i} &= \overbrace{yz^A z_1 \ldots z_t}^{[:-\beta_i]}\ \overbrace{z_{t+1} \ldots z_{|z|} z^B}^{[-\beta_i:]}\\ yz^{n_j} &= \underbrace{yz^A z_1 \ldots z_t}_{[:-\beta_j]}\ \underbrace{z_{t+1} \ldots z_{|z|} z^B z_1 \ldots z_t z_{t+1} \ldots z_{|z|} z^C}_{[-\beta_j:]}\end{aligned}$$

一方で，S_1 が無限集合の場合には，$|y|$ がたかだか有限であるので，以下を満たす $i, j \in S_1$ が存在しなければならない：

$$\begin{aligned}yz^{n_i} &= \overbrace{y_1 \ldots y_t}^{[:-\beta_i]}\ \overbrace{y_{t+1} \ldots y_{|y|} z^{n_i}}^{[-\beta_i:]}\\ yz^{n_j} &= \underbrace{y_1 \ldots y_t}_{[:-\beta_j]}\ \underbrace{y_{t+1} \ldots y_{|y|} z^{n_i} z^{n_j - n_i}}_{[-\beta_j:]}\end{aligned}$$

© 2014 Information Processing Society of Japan

（ただし，上で y_i などは $y_{[i]}$ を表すものとして用いた．）

このようにして選んだ i，j について以下が成立する：

$$yz^{n_i}_{[-\beta_i:]} \prec yz^{n_j}_{[-\beta_j:]}. \qquad (\star_3)$$

このとき，$(\star_1)$，$(\star_2)$ により，以下のことが成立する：

- $\mathbf{c} \xrightarrow{yz^{n_i}_{[-\beta_i:]}} \mathbf{c}'_i \xrightarrow{\varepsilon}{}^{*} \mathbf{c}''_i \in S$．かつ，
- $\mathbf{c} \xrightarrow{yz^{n_j}_{[-\beta_j:]}} \mathbf{c}'_j$ について $\mathbf{c}'_j$ までは S を訪れない．

これは $(\star_3)$ に反する．したがって主張が成立した．　□

5. 適用例

以下で考える例は，1 つの例外をのぞいて文献 [1], [2], [6], [7] のいずれかで，明示的にその非 DCFL 性が証明されているものである．

5.1 $\{a^n b^n, a^n b^{2n} \mid n \geq 1\}$

言語 $L = \{a^n b^n, a^n b^{2n} \mid n \geq 1\}$ が DCFL ではないことを，我々の結果を用いて証明する．L は，CFL ではあるが DCFL ではない例である．DCFL でないことについては文献 [1], [2], [6] などで証明されている．

この言語が DCFL ではないことに対する直感として，$a^n b^n$ での n の対応を検査した段階でスタックはほとんど空になるが，$a^n b^{2n}$ も受理しなくてはならないので，ほとんど空のスタックでちょうど n を検査することが不可能，というものがある．

L を受理する DPDA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ が存在したとすると，定理 4 を具体化し以下のことを得る：

$$(\forall n \geq 1.\, \mathrm{identify}(M(a^n), \varepsilon, b, n, F))$$
$$\Rightarrow \exists h. \forall n. |M(M(a^n), b^n)| \leq h.$$

仮定が明らかなので $\exists h. \forall n. |M(M(a^n), b^n)| \leq h$ が成立する．これより，任意の $n \geq 1$ について $a^n b^n$ を受理した段階ではスタックの高さはたかだか h しかないことが分かり，すなわち $a^n b^n$ を読み終わった段階でとりうる計算状況は，たかだか有限個となることが分かる．厳密には

$$\mathcal{C} = \{M(a^n b^n) \mid n \in \mathbb{N}\}$$

で定義される計算状況の集合 $\mathcal{C}$ が有限集合となる．したがって，$n_1 < n_2$ とする 2 つの数で $M(a^{n_1} b^{n_1}) = M(a^{n_2} b^{n_2})$ を満たすものがある．一方，言語 L では，任意の n で $a^n b^n$，$a^n b^{2n}$ の両方が受理され，すなわち $\mathrm{identify}(M(a^n b^n), \varepsilon, b, n, F)$ が成立する．これらを合わせると，$n_1 < n_2$ について

- $\mathrm{identify}(M(a^{n_2} b^{n_2}), \varepsilon, b, n_1, F)$
- $\mathrm{identify}(M(a^{n_2} b^{n_2}), \varepsilon, b, n_2, F)$

の両方が成立することになるが，これは identify の定義に反するため，L が DCFL ではないことが分かった．

ここでみた「$\mathcal{C}$ の有限性が，スタックの高さがたかだか h の計算状況 $\mathbf{c}$ で異なる文字列 b^{n_1}，b^{n_2} を特徴付ける（すなわち，それぞれを読み終わって初めて受理状態に入る）ものの存在を示すが，これは identify の定義に反する」という証明の流れは，他の例についても同様に展開される．

5.2 $\{ww^R \mid w \in \{a, b\}^*\}$

言語 $L = \{ww^R \mid w \in \{a, b\}^*\}$ が DCFL ではないことを示す．これは CFL ではあるが，DCFL ではないような例である [1], [7]．

L を受理する DPDA $M = (Q, \Sigma, \Gamma, \delta_0, Z_0, F)$ が存在したとすると，定理 4 を具体化し以下のことを得る：

$$(\forall n \geq 1.\, \mathrm{identify}(M(a^n), bb, a, n, F))$$
$$\Rightarrow \exists h. \forall n. |M(M(a^n), bba^n)| \leq h.$$

仮定が明らかなので $\exists h. \forall n. |M(M(a^n), bba^n)| \leq h$ が成立し，任意の n について $a^n bba^n$ を読み終わった直後のスタックの高さはたかだか h であることが分かった．ところで，任意の偶数 n について，$a^n bba^{\frac{n}{2}} a^{\frac{n}{2}}\, bba^n = a^n bba^n\, bba^n \in L$ であることに注意すると，スタックの高さがたかだか h の計算状況 $\mathbf{c}$ で，異なる bba^{n_1} と bba^{n_2} を特徴付けるものが存在することになる．しかしこれは不可能であり，L が DCFL ではないことが分かった．

5.3 $\{a^n b^n c^n \mid n \geq 1\}$

言語 $L = \{a^n b^n c^n \mid n \geq 1\}$ が DCFL ではないことを，我々の結果を用いて証明する．L は CFL でないよく知られた例である．L を受理する DPDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ が存在したとして，M が $a^n b^n$ を読んだ段階でのスタックがほとんど空であることを示し，そこから続けて c^n の部分を検査することができないことを示したい．

しかし，5.1 節や 5.2 節の例と違って，$a^n b^n$ の部分では受理状態を訪れず，分かりやすい形でマッチングが行われていることを示せない．そこで，この問題に取り組むための準備を行う．

5.3.1 PDA 上の後ろ向き到達可能性解析

直後の議論で用いるために，プッシュダウンオートマトン（PDA）上の入力文字列を無視しない後ろ向き到達可能性解析（backward-reachability analysis）を考える．通常，ソフトウェアモデル検査などで用いる後ろ向き到達可能性解析は，PDA から入力文字列を忘却したプッシュダウンシステム（PDS）に対して行われるが，ここでは，入力文字列が意味を持つような一般化された到達可能性解析を導入する．

任意の PDS は，ある PDA の遷移をすべて ε 遷移にしてしまうことで得られ，このときの後ろ向き到達可能集合 $\mathrm{pre}^*(\mathcal{C}) = \{\mathbf{c} \in \mathit{Conf} \mid \mathbf{c} \rightarrow^* \mathbf{c}', \mathbf{c}' \in \mathcal{C}\}$ は，計算状況の集合 $\mathcal{C} \subseteq \mathit{Conf}$ に到達できるすべての計算状況として定義

© 2014 Information Processing Society of Japan

される．特に $\mathcal{C}$ が正則な集合であれば $\mathrm{pre}^*(\mathcal{C})$ が正則であり構成的に求められることが知られている．ここで計算状況の集合 $\mathcal{C}$ が正則であるとは，Multi-automaton（MA）$\mathcal{A}$ で $\mathcal{L}(\mathcal{A}) = \mathcal{C}$ とするものが存在することを意味する [10]．MA $\mathcal{A} = (A, \Gamma, \Delta, Q, F)$ は，A を状態の有限集合，Γ を有限のアルファベット，$\delta : A \times \Gamma \to A$ を遷移関数，$Q \subseteq A$ は初期状態の集合，$F \subseteq A$ を最終状態の集合とするオートマトンである．$\mathcal{A}(q, \xi) \in F$ であるとき，$\mathcal{A}$ は計算状況 $\langle q, \xi \rangle \in Q \times \Gamma^*$ を受理するといい，$\mathcal{A}$ の受理する計算状況のなす集合 $\mathcal{L}(\mathcal{A})$ が以下で定義される：

$$\mathcal{L}(\mathcal{A}) = \{\langle q_i, \xi \rangle \mid q_i \in Q, \mathcal{A}(q_i, \xi) \in F\} \subseteq Q \times \Gamma^*.$$

ここでは，PDA における後ろ向き到達可能集合

$$\mathrm{pre}^*(\mathcal{C}, L) = \left\{ \mathbf{c} \in \mathit{Conf} \mid \mathbf{c} \xrightarrow{w} \mathbf{c}', \mathbf{c}' \in \mathcal{C}, w \in L \right\}$$

を考え，特に $\mathcal{C}$ と L がともに正則な集合であるときに，$\mathrm{pre}^*(\mathcal{C}, L)$ が正則であり構成的に求められることを示す．$\mathrm{pre}^*(\mathcal{C}, L)$ は，入力語として L の要素だけを用いて $\mathcal{C}$ に入ることができるような計算状況全体を表す．ここで，$\mathrm{pre}^*(\mathcal{C})$ は PDS 上の到達可能集合を表したものであり，$\mathrm{pre}^*(\mathcal{C}, L)$ は PDA 上の到達可能集合を表したものであることに注意されたい．

定理 5. $\mathcal{C}$ と L がともに正則な集合であるときに，$\mathrm{pre}^*(\mathcal{C}, L)$ を受理する有限オートマトンを構成することができる．

証明のアイディア． PDA $P = (Q_P, \Sigma, \Gamma, \delta, q_0, Z_0, F_P)$ と $\mathcal{L}(M) = L$ とする決定性有限オートマトン $M = (Q_M, \Sigma, \delta, m_0, F_M)$ を用いて PDS $P \times M$ を直積構成し，

$$\mathrm{pre}^*_P(\mathcal{C}, L) = \pi(\mathcal{R})$$
$$\text{where } \mathcal{R} = \mathrm{pre}^*_{P \times M}(\{\langle \langle q, f \rangle, \xi \rangle \mid \langle q, \xi \rangle \in \mathcal{C}, f \in F_M\})$$

が成立することを示せば十分である．$P \times M$ は，状態が $Q_P \times Q_M$ であり，スタック記号の集合が Γ となる PDS である．

また，π は以下で定義される射影 $\pi : ((Q_P \times Q_M) \times \Gamma^*) \to (Q_P \times \Gamma^*)$ である：

$$\pi(X) = \{\langle p, w \rangle \mid \langle \langle p, m_0 \rangle, w \rangle \in X\}.$$

実際，正則な計算状況の集合 $\mathcal{R}$ を受理する MA から，π による像 $\pi(\mathcal{R})$ を受理する MA は容易に構成することができる．□

注意

Bouajjani らの論文 [10] では，Multi-automaton の定義では遷移関数ではなく，遷移関係が用いられている．しかし，サブセット構成などのよく知られた手続きで決定化できるので，遷移関数を用いた定義でも本質的に差はない．

また，$\mathcal{A}$ が計算状況 $\langle q, \xi \rangle$ を受理することは，実際にはスタックトップから底までを読むことで定義され，すなわち $\mathcal{A}(q, \xi^R) \in F$ であるとき，となっている．ただし，正規言語は reverse をとる操作について閉じているので，我々のスタックの底からトップまでを読むことでの定義は，元の定義と本質的な差はない．

5.3.2　後ろ向き到達可能性を用いた証明

$a^n b^n$ を読んだ段階でのスタックがほとんど空になっていることは，形式的には $\exists h. \forall n. |M(M(a^n), b^n)| \leq h$ として書ける．これは定理 4 を以下の形で具体化したときの結論になっている：

$$\forall S \subseteq Q. (\forall n \geq 1. \mathrm{identify}(M(a^n), \varepsilon, b, n, S))$$
$$\Rightarrow \exists h. \forall n. |M(M(a^n), b^n)| \leq h.$$

したがって，何らかの状態の部分集合 $S \subseteq Q$ について

$$\forall n \geq 1. \mathrm{identify}(M(a^n), \varepsilon, b, n, S) \tag{1}$$

を証明すれば十分である．式 (1) は，計算状況 $M(a^n)$ に入力 b^n を与えた場合に，ちょうど b^n を読み終わったところで集合 S に入ることを意味しており，$M(a^n)$ は数 n を特徴付けているといえる．

我々は，実際には M について式 (1) を示すのではなく，M をもとにして式 (1) を満たすような M' で，$\mathcal{L}(M) = \mathcal{L}(M')$ とするものを構成する．このような M' を構成するのに，直前で議論した後ろ向き到達可能集合の考えが有効である．

M の最終状態の集合 F と，スタックシンボルの集合 Γ から，定理 5 を用いて正則な計算状況の集合 $\mathcal{OK} = \mathrm{pre}^*(F \times \Gamma^*, c^*)$ を計算する．$\mathbf{c} \in \mathcal{OK}$ ならば，$w_c \in c^*$ で $\mathcal{V}\mathrm{isit}(\mathbf{c}, w_c, F)$ とするものがあることを意味し，特に $\mathcal{V}\mathrm{isit}(M(a^i), b^j, \mathcal{OK})$ ならば，$i = j$ にほかならない．

問題となるのは，現在の計算状況が $\mathcal{OK}$ の要素に入っていることをどのように調べるかという一点である．$\mathcal{OK}$ は正則な計算状況の集合であるので，これを受理する MA $\mathcal{A} = (A, \Gamma, \delta_A, Q, F_A)$ が後ろ向き到達可能性解析により構成される．$\mathcal{A}$ は有限状態機械にすぎないので，これを M' の有限制御部に埋め込み，

$$M' = (Q^0 \uplus Q^1, \Sigma, \Gamma \times A^{|Q|}, \delta', q_0^0, \langle Z_0, \boldsymbol{a}_0 \rangle, F^0 \uplus F^1)$$

とする．ただし，M の状態集合を $Q = \{q_0, q_1, \ldots, q_{|Q|-1}\}$ で表し，タグで拡張した Q を $Q^x = \{q_0^x, q_1^x, \ldots, q_{|Q|-1}^x\}$ で定義する．$\boldsymbol{a}_0 = \langle q_0, q_1, \ldots, q_{|Q|-1} \rangle$ とし，δ' は**図 1** のように定義する．このような構成は，Esparza らによる regular valuation [11] などですでに知られている．$\mathcal{L}(M) = \mathcal{L}(M')$ が成立し，M' は正規化された DPDA になることに注意する．

上で行った議論と構成により，M' が部分集合 Q^1 に入るというのは，$a^n b^n$ を読んだことを検知したことにほかならないので，

$$\forall n \geq 1. \mathrm{identify}(M(a^n), \varepsilon, b, n, Q^1)$$

© 2014 Information Processing Society of Japan

$$\delta'(q_i^0,\langle\gamma,\boldsymbol{a}\rangle,b)=\begin{cases}\left\langle q_j^0,\varepsilon\right\rangle & \text{if } \delta(q_i,\gamma,b)=\langle q_j,\varepsilon\rangle\\ \left\langle q_j^0,\langle\gamma',\boldsymbol{a}\rangle\right\rangle & \text{if } \delta(q_i,\gamma,b)=\langle q_j,\gamma'\rangle \quad \wedge\,\delta_{\mathcal{A}}(\boldsymbol{a},\gamma\,\gamma')[j]\notin F_{\mathcal{A}}\\ \left\langle q_j^0,\langle\gamma',\boldsymbol{a}\rangle\langle\gamma'',\delta_{\mathcal{A}}(\boldsymbol{a},\gamma)\rangle\right\rangle & \text{if } \delta(q_i,\gamma,b)=\langle q_j,\gamma'\,\gamma''\rangle\wedge\,\delta_{\mathcal{A}}(\boldsymbol{a},\gamma\,\gamma'\,\gamma'')[j]\notin F_{\mathcal{A}}\\ \left\langle q_j^1,\langle\gamma',\boldsymbol{a}\rangle\right\rangle & \text{if } \delta(q_i,\gamma,b)=\langle q_j,\gamma'\rangle \quad \wedge\,\delta_{\mathcal{A}}(\boldsymbol{a},\gamma\,\gamma')[j]\in F_{\mathcal{A}}\\ \left\langle q_j^1,\langle\gamma',\boldsymbol{a}\rangle\langle\gamma'',\delta_{\mathcal{A}}(\boldsymbol{a},\gamma)\rangle\right\rangle & \text{if } \delta(q_i,\gamma,b)=\langle q_j,\gamma'\,\gamma''\rangle\wedge\,\delta_{\mathcal{A}}(\boldsymbol{a},\gamma\,\gamma'\,\gamma'')[j]\in F_{\mathcal{A}}\end{cases}$$

$$\delta'(q_i^1,\langle\gamma,\boldsymbol{a}\rangle,b)=\begin{cases}\left\langle q_j^1,\varepsilon\right\rangle & \text{if } \delta(q_i,\gamma,b)=\langle q_j,\varepsilon\rangle\\ \left\langle q_j^1,\langle\gamma',\boldsymbol{a}\rangle\right\rangle & \text{if } \delta(q_i,\gamma,b)=\langle q_j,\gamma'\rangle\\ \left\langle q_j^1,\langle\gamma',\boldsymbol{a}\rangle\langle\gamma'',\delta_{\mathcal{A}}(\boldsymbol{a},\gamma)\rangle\right\rangle & \text{if } \delta(q_i,\gamma,b)=\langle q_j,\gamma'\,\gamma''\rangle\end{cases}$$

$$\delta_{\mathcal{A}}(\langle a_0,\ldots,a_n\rangle,\gamma)=\langle\delta_{\mathcal{A}}(a_0,\gamma),\ldots,\delta_{\mathcal{A}}(a_n,\gamma)\rangle\,.$$

図 1　δ' の定義
Fig. 1　The definition of δ'.

が成立し，ただちに

$$\exists h.\,\forall n.\,|M'(M'(a^n),b^n)|\le h$$

が導かれる．

これより，任意の $n\ge 1$ について a^nb^n を読み終わった段階ではスタックの高さはたかだか h しかないことが分かる．すると，1 つの計算状況 $\mathbf{c}$ で異なる 2 つの c^{n_1}，c^{n_2} を特徴付ける必要があるが，これは不可能であり，L が DCFL でないことが分かった

5.4　$\left\{\boldsymbol{w}\in\{\boldsymbol{a},\boldsymbol{b}\}^*\mid \boldsymbol{w}=\boldsymbol{uv},|\boldsymbol{u}|=|\boldsymbol{v}|,\boldsymbol{v}\text{ は }\boldsymbol{b}\text{ を含む}\right\}$

言語 $L=\left\{w\in\{a,b\}^*\mid w=uv,|u|=|v|,v\text{ は }b\text{ を含む}\right\}$ が DCFL ではないことを示す．これは CFL ではあるが，DCFL ではない例である [2].

a^nb に注目すると，a^nba^n から先はどんな $w_a\in a^*$ を読んだとしても受理されることはないが，$l<n$ については a^nba^l からある $w_a\in a^*$ を読むことで受理できることを利用する．すなわち，L を受理する DPDA $M=(Q,\Sigma,\Gamma,\delta,q_0,F)$ について $\mathrm{pre}^*(F\times\Gamma^*,a^*)$ の補集合 $\mathcal{F}ail=(\mathrm{pre}^*(F\times\Gamma^*,a^*))^{\complement}$ を用いることにする．

計算状況の正則集合 $\mathcal{F}ail$ は，これ以降 $w_a\in a^*$ を読むだけでは受理できない位置，すなわち a^nba^n の最後まで読んだことを検知するために用いる．$\mathcal{F}ail$ を用いて 5.4 節と同様に構成した M' について

$$\forall n\ge 1.\,\mathrm{identify}(M'(a^n),b,a,n,Q^1)$$

が成立するので，定理 4 より

$$\exists h.\,\forall n.\,|M'(M'(a^n),ba^n)|\le h$$

が導かれる．

これより，任意の $n\ge 1$ について $a^{2n}ba^{2n}$ を読み終わった段階ではスタックの高さはたかだか h しかないことが分かる．一方で，$a^{2n}ba^{2n}\,ba^{2n}\in L$ となることから，スタックの高さがたかだか h の計算状況 $\mathbf{c}$ で，異なる 2 つの ba^{2n_1}，ba^{2n_2} を特徴付けるものが存在する．しかしこのことは不可能であり，L が DCFL でないことが分かった．

5.5　$\left\{\boldsymbol{x}\#\boldsymbol{y}\boldsymbol{x}^{\boldsymbol{R}}\boldsymbol{z}\mid \boldsymbol{x},\boldsymbol{y},\boldsymbol{z}\in\{\boldsymbol{a},\boldsymbol{b}\}^*\right\}$

言語 $L=\left\{x\#yx^Rz\mid x,y,z\in\{a,b\}^*\right\}$ は，文字列 x についてのパターンマッチを行うような言語であり，これは DCFL ではないが CFL ではあるような例である．この例は，Li らによる論文 [4] から引用した．

L を受理する DPDA $M=(Q,\Sigma,\Gamma,\delta,q_0,Z_0,F)$ が存在したとする．$a^n\#a^{n-1}$ を読み終わった後に，スタックがほとんど空になっているということを示す．すなわち，正則な計算状況の集合 $\mathcal{OK}=\mathrm{pre}^*(F\times\Gamma^*,a)$ を考える．これは，a をあと 1 文字読めば受理状態に入るということを意図しているものである．5.4 節と同様にして $\mathcal{OK}$ を用いて M' を構成すると

$$\forall n\ge 1.\,\mathrm{identify}(M'(a^n),a\#,a,n,Q^1)$$

が成立する．したがって，定理 4 により任意の n について $a^{n+1}\#a^n$ を読み終わった段階でのスタックの高さはたかだか h である．$a^{n+1}\#a^n\notin L$ だが，$a^{n+1}\#a^nba^{n+1}\in L$ であることに注意する．すると，スタックの高さがたかだか h の計算状況 $\mathbf{c}$ で，ba^{n_1+1} と ba^{n_2+1} を特徴付けるものが存在しなければならない．ところがこれは不可能なので，L が DCFL ではないことが証明された．

6.　おわりに

本論文では，正規化された DPDA については，入力が文字列の繰返しになっているときに，その計算によって得られるスタックの内容が関数的 GSM によって定義可能であることを示した．また，関数的 GSM によるスタックの表現には，ある種の繰返し構造が現れており，これを用いることによって，部分文字列のマッチングを行うような計算を行う場合には本質的なスタックの消費が避けられず，結果としてスタックがほとんど空になるということを意味する定理を示した．さらに，スタックがほとんど空になってしまうという議論を基に，いくつかの言語が DCFL でないことを証明した．

今後の課題として，特に定理 3 や 4 の形が具体的になり

© 2014 Information Processing Society of Japan

すぎているので，これを一般化したいと考えている．計算中の $x^n yz^n$ という部分列について，これを最後まで読んだところではじめて特別な状況に移る場合には本質的にスタックが消費されるということを述べているが，ちょうど x^n と z^n で n 個の対応がついていなくても，たとえば定数程度の誤差は問題にならないはずであり，いくらかの緩和は可能であると考えている．また，スタックが本質的に空になっていることを示した後にも，2 つの文字列 w_1，w_2 で $w_1 \prec w_2$ とするものを見つけ，それぞれをちょうど読み終わったときにだけ特殊な状況に移るはずであるという条件を整えたうえで，矛盾が生じるという議論を行っている．これについても一般化し，容易に扱えるようにしたい．

言語の非 DCFL 性の証明について，スタックがほとんど空になってしまうという議論では，扱いきれないものがあると考えられる．たとえば，$L_1 = \{a^p \mid p \text{ は素数}\}$，$L_2 = \left\{a^n b^{n^2} \mid n \geq 1\right\}$，$L_3 = \left\{a^i b^j c^k \mid k \leq i \text{ または } k \leq j\right\}$ といった言語の非 DCFL 性を証明するにあたっては，今回の手法は有効ではないと考える．このような言語を扱うには，スタックの高さに関する特徴付けではなく，DPDA が生成するスタックの内容にもう少し迫った特徴付けを行う必要があると考えている．

一方で，L_1，L_2 が文脈自由言語でないということは，文脈自由文法を利用して容易に証明される標準的な反復補題を用いると簡単に示すことができる．そこで，このような文法を用いたことで得られる反復補題の本質的な意味を，今回のように機械における直感的な性質として表現したい．

参考文献

[1] Harrison, M.A.: *Introduction to Formal Language Theory*, Addison-Wesley Longman Publishing Co., Inc. (1978).

[2] Yu, S.: A Pumping Lemma for Deterministic Context-free Languages, *Information Processing Letters*, Vol.31, No.1, pp.47–51 (1989).

[3] Kanazawa, M.: Machine-Based Approach to Pumping (2013), available from ⟨http://makotokanazawa.blogspot.jp/2013/12/machine-based-approach-to-pumping.html⟩.

[4] Li, M. and Vitányi, P.: A New Approach to Formal Language Theory by Kolmogorov Complexity, *SIAM Journal on Computing*, Vol.24, No.2, pp.398–410 (1995).

[5] Glier, O.: Kolmogorov Complexity and Deterministic Context-Free Languages, *SIAM J. Comput.*, Vol.32, No.5, pp.1389–1394 (2003).

[6] Ginsburg, S. and Greibach, S.: Deterministic context free languages, *Information and Control*, Vol.9, No.6, pp.620–648 (1966).

[7] Hopcroft, J.E. and Ullman, J.D.: *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley (1979).

[8] Autebert, J.-M., Berstel, J. and Boasson, L.: Context-free Languages and Pushdown Automata, *Handbook of Formal Languages*, Vol.1, Rozenberg, G. and Salomaa, A. (Eds.), pp.111–174, Springer-Verlag New York, Inc. (1997).

[9] Ginsburg, S. and Rice, H.G.: Two Families of Languages Related to ALGOL, *J. ACM*, Vol.9, No.3, pp.350–371 (1962).

[10] Bouajjani, A., Esparza, J. and Maler, O.: Reachability analysis of pushdown automata: Application to model-checking, *CONCUR '97: Concurrency Theory*, Lecture Notes in Computer Science, Vol.1243, pp.135–150, Springer Berlin/Heidelberg (1997).

[11] Esparza, J., Kučera, A. and Schwoon, S.: Model checking LTL with regular valuations for pushdown systems, *Information and Computation*, Vol.186, No.2, pp.355–376 (2003).

付　　録

A.1　DPDA の正規化について

本付録では，準備の章で命題 1 として述べたとおりに，任意の DPDA にはそれと等価な正規形が存在することを示す．正規形が存在するということ自体は文献 [1], [7] といった標準的な教科書に Exercise として証明なしで登場し，また Σ 遷移での制限を考慮しないものについては文献 [8] において証明が与えられているといった程度である．以下では明示的にモードによる受理という概念を導入しているが，これは文献 [8] などにも現れたよくある考え方である．

通常の DPDA は $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ の構造であるが，この付録においては，最終状態の集合 $F \subseteq Q$ ではなく，最終モードの集合を代わりに用いる DPDA M で $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, G)$ という構造を考える．最終モードの集合は $G \subseteq Q \times \Gamma$ である．このとき，最終モードの集合を用いて言語を定義することができる：

$$w \in \mathcal{L}(M) \Longleftrightarrow M(\langle q_0, Z_0\rangle, w) \xrightarrow{\varepsilon}{}^{*} \langle q, \xi\,\gamma\rangle \wedge \langle q, \gamma\rangle \in G.$$

命題 2. モードによる受理を行う DPDA M から，同じ言語を受理する，状態による受理を行う DPDA M' を構成することができる．

証明．M' 側では M 側のスタックトップの記号を状態に入れてしまう，というのが証明のアイディアである．各遷移関係は以下のように模倣される．

Σ 遷移で push する場合：

$$M : \langle p, \gamma_1\rangle \xrightarrow{\sigma} \langle q, \gamma_2\gamma_3\rangle$$
$$M' : \langle p_{\gamma_1}, \gamma\rangle \xrightarrow{\sigma} \langle q_{\gamma_3}, \gamma\,\gamma_2\rangle \ \forall\gamma \in \Gamma$$

Σ 遷移でスタックトップシンボルの入れ替えをする場合：

$$M : \langle p, \gamma_1\rangle \xrightarrow{\sigma} \langle q, \gamma_2\rangle$$
$$M' : \langle p_{\gamma_1}, \gamma\rangle \xrightarrow{\sigma} \langle q_{\gamma_2}, \gamma\rangle \ \forall\gamma \in \Gamma$$

Σ 遷移で pop する場合：

$$M : \langle p, \gamma\rangle \xrightarrow{\sigma} \langle q, \varepsilon\rangle$$
$$M' : \langle p_\gamma, \gamma'\rangle \xrightarrow{\sigma} \langle q_{\gamma'}, \varepsilon\rangle \ \forall\gamma' \in \Gamma$$

© 2014 Information Processing Society of Japan

スタック底シンボルが見えている場合の Σ 遷移：

$M : \langle p, Z_0\rangle \xrightarrow{\sigma} \langle q, Z_0\,\gamma\rangle$
$M' : \langle p_{Z_0}, \bot\rangle \xrightarrow{\sigma} \langle q_\gamma, \bot\, Z_0\rangle$
または
$M : \langle p, Z_0\rangle \xrightarrow{\sigma} \langle q, Z_0\rangle$
$M' : \langle p_{Z_0}, \bot\rangle \xrightarrow{\sigma} \langle q_{Z_0}, \bot\rangle$

ε 遷移についてもまったく同様にすればよい．

このようにすると，$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, G)$ から $M' = (Q', \Sigma, \Gamma \cup \{\bot\}, \delta', q_{0Z_0}, \bot, F)$ とすれば $\mathcal{L}(M) = \mathcal{L}(M')$ が成立することが分かる．ただし，$Q' = \{q_\gamma \mid q \in Q, \gamma \in \Gamma\}, F = \{q_\gamma \mid \langle q, \gamma\rangle \in G\}$ である． □

注意 上の証明では，正規化済みのモードによる受理を行う DPDA M から，正規化済みの状態による受理を行う DPDA M' が構成されることに注意する．

命題 3. 状態による受理を行う DPDA M から，必ず次の文字を読む（ε 遷移でループしない），モードによる受理を行う DPDA M_1 を構成することができる．

証明のアイディア． M_1 を満たす DPDA の構成には，標準的な教科書にある ε 遷移でのループを除去する構成をそのまま用いればよい [7]．具体的には，$\langle p, \gamma\rangle \xrightarrow[M]{\varepsilon}{}^{+} \langle p, \zeta\,\gamma\rangle$ が成立するかどうかが決定可能ということが本質である．これを満たす $\langle p, \gamma\rangle$ について，$\delta_{M_1}(p, \gamma, \varepsilon) = \langle s, \gamma\rangle$ とする．特に，$\langle p, \gamma\rangle$ からのループ中に最終状態を訪れるならば $\langle p, \gamma\rangle$ を最終モードに加える．状態 s は sink state であり，任意の $\gamma \in \Gamma$，$\sigma \in \Sigma$ について $\delta_{M_1}(s, \gamma, \sigma) = \langle s, \gamma\rangle$ と定義する． □

命題 4. 命題 3 によって得たモードによって受理を行う DPDA M_1 から ε 遷移では pop だけができるモードによって受理を行う DPDA M_2 を構成することができる．

証明. $M_1 = (Q, \Sigma, \Gamma, \delta_{M_1}, q_0, Z_0, G)$ について，

$$\mathfrak{s}(\langle p, \gamma\rangle) = \begin{cases} \varepsilon & \text{if } \langle p, \gamma\rangle \xrightarrow[M_1]{\varepsilon}{}^{*} \langle q, \varepsilon\rangle \\ \xi & \text{if } \langle p, \gamma\rangle \xrightarrow[M_1]{\varepsilon}{}^{*} \langle q, \xi\rangle \wedge \text{read-mode}(\langle q, \xi\rangle) \end{cases}$$

と定義する．M_1 は ε 遷移に留まり続けることがないので，$\mathfrak{s}$ は任意の $\mathbf{c} \in Q \times \Gamma$ について定義され，特に

$$sup = \max\left(\left\{\, |\mathfrak{s}(\mathbf{c})| \;\middle|\; \mathbf{c} \in Q \times \Gamma \right\}\right)$$

を定義することができる．このとき，以下のようにして等価なモードによる受理をする DPDA $M_2 = (Q', \Sigma, \Gamma, \delta_{M_2}, q_0, Z_0, G')$ を定義する．

M_1 におけるモード $\langle p, \gamma\rangle \in Q \times \Gamma$ が reading mode である場合は $\delta_{M_2}(p, \gamma, \sigma) = \delta_{M_1}(p, \gamma, \sigma)$ として定義する．

M_1 におけるモード $\langle p, \gamma\rangle$ が reading mode でないとすると，$\langle p, \gamma\rangle$ からの ε 遷移列では以下のどちらかが生じる：

(a) $\langle p, \gamma\rangle \xrightarrow[M_1]{\varepsilon}{}^{*} \langle q, \varepsilon\rangle$
(b) $\langle p, \gamma\rangle \xrightarrow[M_1]{\varepsilon}{}^{*} \langle q, \xi\rangle \wedge \text{read-mode}(\langle q, \xi\rangle)$

(a) については，$\delta_{M_2}(p, \gamma, \varepsilon) = \langle q, \varepsilon\rangle$ とすればよい．

(b) については，$\delta_{M_2}(p, \gamma, \varepsilon) = \langle \langle q, \xi\rangle, \varepsilon\rangle$ としておく．これは状態側に本来スタックに push するべき記号列を記憶させたものである．Σ 遷移では当然 push が許されているので，一時的に記憶したスタックもあわせて push するようにする．すなわち，

$$\delta_{M_2}(\langle p, \varepsilon\rangle, \gamma, \sigma) = \delta_{M_1}(p, \gamma, \sigma)$$
$$\delta_{M_2}(\langle p, \xi\gamma'\rangle, \gamma, \sigma) = \langle q, \gamma\,\xi\,\zeta\rangle \;\; \textit{if } \delta_{M_1}(p, \gamma', \sigma) = \langle q, \zeta\rangle .$$

注意するべきは，ε 遷移中では長さがたかだか sup の push しか行わないので，M_2 の状態の集合 $Q' = Q \uplus (Q \times \Gamma^{\le b})$ はたかだか有限で済む．

最後に M_2 の最終モードの集合であるが，これは

$$G_{M_2} = \mathcal{E} \cup \left\{ \langle \langle p, \xi\,\gamma\rangle, \gamma'\rangle \mid \xi\,\gamma \in \Gamma^{\le sup}, \gamma' \in \Gamma, \langle p, \gamma\rangle \in \mathcal{E} \right\}$$

ただし，$\mathcal{E} = \left\{ \langle p, \gamma\rangle \mid \langle p, \gamma\rangle \xrightarrow[M_1]{\varepsilon}{}^{*} \langle q, \xi\gamma'\rangle, \langle q, \gamma'\rangle \in G_{M_1} \right\}$ である． □

命題 5. 命題 4 を満たすモードによる受理を行う DPDA M_2 から，さらに Σ 遷移でもたかだか 1 つしか push しないような，モードによる受理を行う DPDA M_3 を構成することができる．

証明. $M_2 = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, G_{M_2})$ からスタックシンボルを拡張し $M_3 = (Q, \Sigma, \lfloor\Gamma\rfloor, \delta', q_0, \lfloor Z_0\rfloor, G_{M_3})$ を構成する．スタック記号の集合を

$$\lfloor\Gamma\rfloor = \left\{ \lfloor s\rfloor \mid s \in \bigcup_{i=1}^{sup} \Gamma^i \right\}$$

として定義する．$\lfloor s\rfloor$ は，長さがたかだか sup の文字列 s を *1* **文字**として見ることを意味する．また sup は M_2 が Σ 遷移で 1 度に push する記号列の最大長を指す．

M_2 におけるモード $\langle p, \gamma\rangle$ が reading mode かつ $\delta_{M_2}(p, \gamma, \sigma) = \langle q, \zeta\rangle$ であるならば，

$$\delta'(p, \lfloor\xi\,\gamma\rfloor, \sigma) = \begin{cases} \langle q, \lfloor\xi\rfloor\rangle & \textit{if } \zeta = \varepsilon \\ \langle q, \lfloor\xi\rfloor\lfloor\zeta\rfloor\rangle & \text{otherwise} \end{cases}$$

と定義する．

M_2 におけるモード $\langle p, \gamma\rangle$ が reading mode ではない場合を考える．以下では，M_3 の各計算状況に対応する M_2 の計算状況における ε 遷移を考える．まず，2 つ以上の記号を残して reading mode に入る場合，たとえば：

$M_3 : \langle p, \lfloor\gamma_1\gamma_2 \ldots \gamma\rfloor\rangle$
$M_2 : \langle p, \gamma_1\gamma_2 \ldots \gamma\rangle \xrightarrow{\varepsilon}{}^{*} \langle q, \gamma_1\gamma_2\rangle \wedge \text{read-mode}(\langle q, \gamma_1\gamma_2\rangle)$

© 2014 Information Processing Society of Japan

については

$$\delta'(p, \lfloor\gamma_1\gamma_2\ldots\gamma\rfloor, \sigma) = \begin{cases} \langle r, \lfloor\gamma_1\rfloor\rangle & \text{if } \delta(q,\gamma_2,\sigma) = \langle r,\varepsilon\rangle \\ \langle r, \lfloor\gamma_1\rfloor\lfloor\zeta\rfloor\rangle & \text{if } \delta(q,\gamma_2,\sigma) = \langle r,\zeta\rangle \end{cases}$$

として δ' を定義する．

また，1 つだけ記号を残して reading mode に入る場合，すなわち：

$$M_3 : \langle p, \lfloor\gamma_1\gamma_2\ldots\gamma\rfloor\rangle$$
$$M_2 : \langle p, \gamma_1\gamma_2\ldots\gamma\rangle \xrightarrow{\varepsilon}{}^{*} \langle q,\gamma_1\rangle \wedge \text{read-mode}(\langle q,\gamma_1\rangle)$$

については

$$\delta'(p, \lfloor\gamma_1\gamma_2\ldots\gamma\rfloor, \sigma) = \begin{cases} \langle r, \lfloor\gamma_1\rfloor\rangle & \text{if } \delta(q,\gamma_1,\sigma) = \langle r,\varepsilon\rangle \\ \langle r, \lfloor\zeta\rfloor\rangle & \text{if } \delta(q,\gamma_1,\sigma) = \langle r,\zeta\rangle \end{cases}$$

として δ' を定義する．

最後に，ε 遷移でその部分のスタックをすべて pop してしまう場合：

$$M_3 : \langle p, \lfloor\gamma_1\gamma_2\ldots\gamma\rfloor\rangle$$
$$M_2 : \langle p, \gamma_1\gamma_2\ldots\gamma\rangle \xrightarrow{\varepsilon}{}^{*} \langle q,\varepsilon\rangle$$

については，$\delta'(p, \lfloor\gamma_1\gamma_2\ldots\gamma_n\rfloor, \varepsilon) = \langle q,\varepsilon\rangle$ と定義する．

M_3 の受理モードは以下で定義する．

$$G_{M_3} = \{\langle q, \lfloor\xi\,\gamma\rfloor\rangle \mid \lfloor\xi\,\gamma\rfloor \in \lfloor\Gamma\rfloor, \langle q,\gamma\rangle \in \mathcal{E}\}.$$

ただし，$\mathcal{E} = \left\{ \langle p,\gamma\rangle \mid \langle p,\gamma\rangle \xrightarrow[M_2]{\varepsilon}{}^{*} \langle q,\xi\gamma'\rangle, \langle q,\gamma'\rangle \in G_{M_2} \right\}$ である． □

系 1. 任意の DPDA M から，正規化された DPDA N を構成することができる．

証明. 命題 5 と命題 2（の**注意**）から明らか． □

上里 友弥　（学生会員）

2013 年筑波大学情報学群情報科学類卒業．同年筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻進学．形式言語理論を用いたプログラムの自動検証に興味を持つ．

南出 靖彦　（正会員）

1993 年京都大学大学院理学研究科数理解析専攻修士課程修了．同年同大学数理解析研究所助手．1999 年筑波大学電子・情報工学系講師．2004 年筑波大学大学院システム情報工学研究科講師．2007 年同准教授．現在，筑波大学システム情報系准教授．博士（理学）．プログラミング言語およびソフトウェア検証に興味を持つ．

© 2014 Information Processing Society of Japan