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 9 回：プッシュダウンオートマトン
(日本語訳：大久保 好章，湊 真一)

プッシュダウンオートマトン I

今回の講義も引続きプッシュダウンオートマトンについて議論する．これまでの講義で，正規言語を特徴付ける機械モデルとしての非決定性有限オートマトンについて学んだ．

プッシュダウンオートマトン I

今回の講義も引続きプッシュダウンオートマトンについて議論する．これまでの講義で，正規言語を特徴付ける機械モデルとしての非決定性有限オートマトンについて学んだ．

ここでは，文脈自由言語を特徴付けるためにプッシュダウンオートマトンを用いる．一般に，プッシュダウンオートマトンとは，λ-遷移 (λ-transition) を許す非決定性有限オートマトンの拡張である．

プッシュダウンオートマトン I

今回の講義も引続きプッシュダウンオートマトンについて議論する．これまでの講義で，正規言語を特徴付ける機械モデルとしての非決定性有限オートマトンについて学んだ．

ここでは，文脈自由言語を特徴付けるためにプッシュダウンオートマトンを用いる．一般に，プッシュダウンオートマトンとは，λ-遷移 (λ-transition) を許す非決定性有限オートマトンの拡張である．

ここで，λ-遷移とは，オートマトンが，入力テープ上の空語 λ を読み，その状態を変化させることを意味する．

プッシュダウンオートマトン II

非決定性有限オートマトンに対する拡張は，スタックにある．スタックは，プッシュダウンオートマトンが任意の有限記号列を覚えておくことを可能にする．

プッシュダウンオートマトン II

非決定性有限オートマトンに対する拡張は，スタックにある．スタックは，プッシュダウンオートマトンが任意の有限記号列を覚えておくことを可能にする．

しかし，計算機の一般的なモデル (例えば，チューリング機械やランダムアクセス機械) とは対称的に，スタックへのアクセスは *lifo*-モードのみである．ここで，lifo とは，“last in first out (先入れ後出し)” のことである．

プッシュダウンオートマトン II

非決定性有限オートマトンに対する拡張は，スタックにある．スタックは，プッシュダウンオートマトンが任意の有限記号列を覚えておくことを可能にする．

しかし，計算機の一般的なモデル (例えば，チューリング機械やランダムアクセス機械) とは対称的に，スタックへのアクセスは *lifo*-モードのみである．ここで，lifo とは，“last in first out (先入れ後出し)” のことである．

すなわち，スタックでは，その先頭においてのみリード *(read)*，プッシュ*(push)*，および，ポップ *(pop)* が行なわれる．これは，既に学んだデータ構造のスタックと同様である．

プッシュダウンオートマトン III

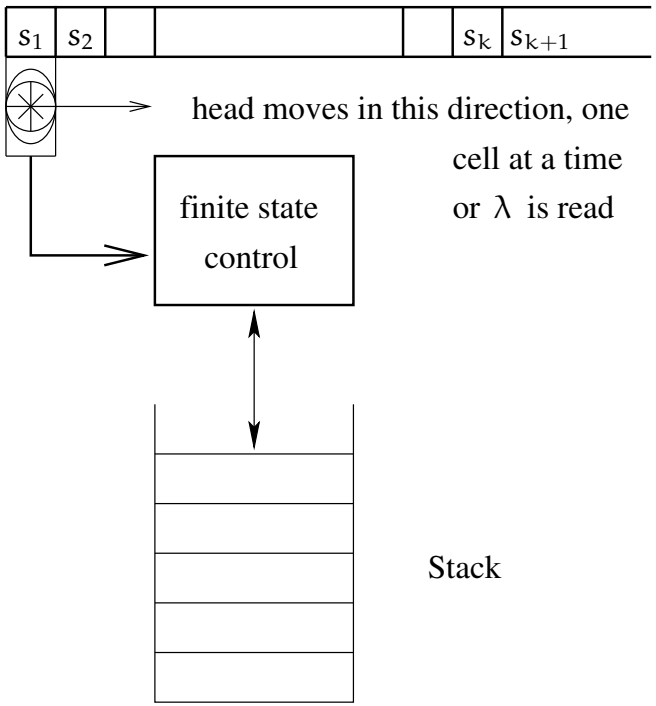

図 1: プッシュダウンオートマトン

プッシュダウンオートマトン IV

形式的には，一回の遷移で，プッシュダウンオートマトンは：

(1) それが読んだ記号 (文字) を入力から消費する．λ が読まれた場合は，入力記号は消費されない．

プッシュダウンオートマトン IV

形式的には，一回の遷移で，プッシュダウンオートマトンは：

(1) それが読んだ記号 (文字) を入力から消費する．λ が読まれた場合は，入力記号は消費されない．

(2) 新たな状態へと移る．ただし，それは現在の状態と同じ場合も異なる場合もある．

プッシュダウンオートマトン IV

形式的には，一回の遷移で，プッシュダウンオートマトンは：

(1) それが読んだ記号 (文字) を入力から消費する．λ が読まれた場合は，入力記号は消費されない．

(2) 新たな状態へと移る．ただし，それは現在の状態と同じ場合も異なる場合もある．

(3) スタックの先頭の記号をある記号列に置き換える．その記号列が λ である場合もあり，これは，スタックのポップ操作に対応する．その記号列が先にスタックの先頭に現れた記号と同じ場合がある．すなわち，スタックには何の変化もおきない．スタックの先頭記号は他の別の記号と置き換えられることもある．この場合，スタックの先頭は変化するが，プッシュもポップも行われない．

最後に，スタックの先頭記号は，二つ以上の記号に置き換えられることもある．この場合は，スタックの先頭記号を (場合によっては) 変え，ひとつ以上の新たな記号をスタックにプッシュする．

例

言語 $L = \{ww^T \mid w \in \{0,1\}^*\}$ がプッシュダウンオートマトンに受理されることを非形式的に示そう．入力として $x \in \{0,1\}^*$ が与えられたとする．

例

言語 $L = \{ww^T \mid w \in \{0,1\}^*\}$ がプッシュダウンオートマトンに受理されることを非形式的に示そう．入力として $x \in \{0,1\}^*$ が与えられたとする．

(1) まだ x の中央にたどり着いていないとの“推測”を表す状態 q_0 から出発する．q_0 にいる間は，一回に一文字を読み，そのコピーをプッシュしてスタックに蓄える．

例

言語 $L = \{ww^T \mid w \in \{0,1\}^*\}$ がプッシュダウンオートマトンに受理されることを非形式的に示そう．入力として $x \in \{0,1\}^*$ が与えられたとする．

(1) まだ x の中央にたどり着いていないとの“推測”を表す状態 q_0 から出発する．q_0 にいる間は，一回に一文字を読み，そのコピーをプッシュしてスタックに蓄える．

(2) 任意の時点で，中央 (すなわち，L の入力記号列が $x = ww^T$ である場合は，w の最後) にたどり着いたことを推測してもよい．この時点で，w はスタック上にあり，特に，w の最右記号はスタックの先頭，最左記号は最後にある．この選択 (推測) を，状態を自発的に q_1 に変化させることにより表す．(つまり，次の入力記号を読む代わりに，λ を読む．)

例 - 続き

(3) 状態 q_1 においては，入力記号とスタックの先頭文字とを比較する．入力から読んだ記号がスタックの先頭文字と一致する場合は，状態 q_1 のまま処理を続け，スタックをポップする．もし異なった場合は，入力記号列を受理せずに終了する．すなわち，この (1 つの) 計算過程は消滅する．

例 - 続き

(3) 状態 q_1 においては，入力記号とスタックの先頭文字とを比較する．入力から読んだ記号がスタックの先頭文字と一致する場合は，状態 q_1 のまま処理を続け，スタックをポップする．もし異なった場合は，入力記号列を受理せずに終了する．すなわち，この (1 つの) 計算過程は消滅する．

(4) もし x の終端にたどり着いて，かつ，スタックが空であった場合は x を受理する．

注意点

明らかに，$x \in L$ である場合は，x の中央を正しく推測することにより，受理を意味する計算過程が得られる．もし，$x \notin L$ である場合は，推測したこととは無関係に，任意の計算過程が受理に至ることは無いであろう．よって，上記のプッシュダウンオートマトンは，L の非決定性受理器である．

注意点

明らかに，$x \in L$ である場合は，x の中央を正しく推測することにより，受理を意味する計算過程が得られる．もし，$x \notin L$ である場合は，推測したこととは無関係に，任意の計算過程が受理に至ることは無いであろう．よって，上記のプッシュダウンオートマトンは，L の非決定性受理器である．

上述した通り，プッシュダウンオートマトンは，自発的な (spontaneous) 遷移を行う際，スタックを更新することが許されている点に注意しよう．

注意点

明らかに，$x \in L$ である場合は，x の中央を正しく推測することにより，受理を意味する計算過程が得られる．もし，$x \notin L$ である場合は，推測したこととは無関係に，任意の計算過程が受理に至ることは無いであろう．よって，上記のプッシュダウンオートマトンは，L の非決定性受理器である．

上述した通り，プッシュダウンオートマトンは，自発的な (spontaneous) 遷移を行う際，スタックを更新することが許されている点に注意しよう．

問

プッシュダウンオートマトンに受理される言語は，形式的にはどのように定義できるだろうか？

プッシュダウンオートマトン V

先の例を見ると，オートマトンは空のスタックと共にその計算を終了した．よって，プッシュダウンオートマトンに受理される言語を，空のスタックで終了する計算が存在するすべての記号列の集合として定義することが自然であろう．

プッシュダウンオートマトン V

先の例を見ると，オートマトンは空のスタックと共にその計算を終了した．よって，プッシュダウンオートマトンに受理される言語を，空のスタックで終了する計算が存在するすべての記号列の集合として定義することが自然であろう．

二番目として，有限オートマトンを議論する際に用いた方法論を利用することができる．つまり，すべての状態の集合から一部を選び，それらを受理状態であると考える．このアプローチをとる場合は，プッシュダウンオートマトンに受理される言語を，受理状態で終了する計算が存在するすべての記号列の集合として定義することが自然であろう．

プッシュダウンオートマトン V

先の例を見ると，オートマトンは空のスタックと共にその計算を終了した．よって，プッシュダウンオートマトンに受理される言語を，空のスタックで終了する計算が存在するすべての記号列の集合として定義することが自然であろう．

二番目として，有限オートマトンを議論する際に用いた方法論を利用することができる．つまり，すべての状態の集合から一部を選び，それらを受理状態であると考える．このアプローチをとる場合は，プッシュダウンオートマトンに受理される言語を，受理状態で終了する計算が存在するすべての記号列の集合として定義することが自然であろう．

両者のアイデアを試してみる．

プッシュダウンオートマトン VI

定義 1

$\mathcal{K} = [Q, \Sigma, \Gamma, \delta, q_0, k_0, F]$ はプッシュダウンオートマトン *(pushdown automaton)* と呼ばれる．ただし，

(1) Q は非空の有限集合 (状態 *(states)* 集合),

(2) Σ はアルファベット (入力アルファベット),

(3) Γ はアルファベット (スタックアルファベット),

(4) $\delta\colon Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \longrightarrow \wp_{fin}(Q \times \Gamma^*)$，遷移関係 *(transition relation)* [a],

(5) $q_0 \in Q$ は 初期状態 *(initial state)*,

(6) k_0 は スタック記号 *(stack symbol)* と呼ばれる．$k_0 \in \Gamma$ であり，スタックは最初，k_0 のみを含む．

(7) $F \subseteq Q$ は，最終状態 *(final states)* の集合．

[a] ここでは，$Q \times \Gamma^*$ の有限部分集合を表すために $\wp_{fin}(Q \times \Gamma^*)$ を用いる．

プッシュダウンオートマトン VII

以下では，入力記号を表すためにアルファベットの小文字を先頭から用い，入力記号列を表すためにアルファベトの小文字を最後から用いる．スタック記号を表すために Γ の大文字を用い，スタック記号の列を表すために小文字のギリシャ文字を用いる．

プッシュダウンオートマトン VII

以下では，入力記号を表すためにアルファベットの小文字を先頭から用い，入力記号列を表すためにアルファベトの小文字を最後から用いる．スタック記号を表すために Γ の大文字を用い，スタック記号の列を表すために小文字のギリシャ文字を用いる．

次に，以下を考える．

$$\delta(q, a, Z) = \{(q_1, \gamma_1), (q_2, \gamma_2), \ldots, (q_m, \gamma_m)\},$$

ここで，$i = 1, \ldots, m$, $a \in \Sigma$, $Z \in \Gamma$ について，$q, q_i \in Q$ であり，かつ，$i = 1, \ldots, m$ について，$\gamma_i \in \Gamma^*$ である．

解釈： $\mathcal{K}$ は状態 q にあり，入力テープ上の a を読み，Z はスタックの先頭記号である．このとき，実行される遷移毎に唯一の (q_i, γ_i)，$i \in \{1, \ldots, m\}$ を非決定的に選択することができる．そして，状態を q_i に変え，$a \neq \lambda$ ならば入力テープ上のヘッドを右へひとつ移動し，Z を γ_i に置き換える．

プッシュダウンオートマトン VIII

γ_i の最右記号が最初にスタックへプッシュされ，次に右から 2 番目の記号がプッシュされ，以下同様に続く，という取り決め (約束) をする．よって，γ_i の最左記号は新たな記号であり，スタックの先頭にある．もし $\gamma_i = \lambda$ であるならば，Z はスタックから取り除かれたと解釈される．

プッシュダウンオートマトン VIII

γ_i の最右記号が最初にスタックへプッシュされ，次に右から 2 番目の記号がプッシュされ，以下同様に続く，という取り決め (約束) をする．よって，γ_i の最左記号は新たな記号であり，スタックの先頭にある．もし $\gamma_i = \lambda$ であるならば，Z はスタックから取り除かれたと解釈される．

もし，$a = \lambda$ ならば，解釈は上記と同様であるが，入力テープ上のヘッドは移動しない．

時点表示

プッシュダウンオートマトンによる計算を形式的に扱うために，ここでは時点表示 *(instantaneous description)* を定義する．時点表示とは，3 つ組 (q, w, γ) であり，$q \in Q, w \in \Sigma^*$ および $\gamma \in \Gamma^*$ である．

$\mathcal{K} = [Q, \Sigma, \Gamma, \delta, q_0, k_0, F]$ をプッシュダウンオートマトンとする．このとき，以下のように記述する．

$(p, \beta) \in \delta(q, a, Z)$ ならば，

$$(q, aw, Z\alpha) \xrightarrow[\mathcal{K}]{1} (p, w, \beta\alpha)$$

ここで，$a \in \Sigma \cup \{\lambda\}$ であることに注意する．

$\xrightarrow[\mathcal{K}]{1}$ の反射的推移的閉包を $\xrightarrow[\mathcal{K}]{*}$ と書く．

受理のモード

定義 2

$\mathcal{K} = [Q, \Sigma, \Gamma, \delta, q_0, k_0, F]$ をプッシュダウンオートマトンとする．
最終状態により *(via final state)* $\mathcal{K}$ が受理する言語を，次の集合として定義する．

$$L(\mathcal{K}) = \{w \mid \text{ある } p \in F, \gamma \in \Gamma^* \text{ について } (q_0, w, k_0) \xrightarrow[\mathcal{K}]{*} (p, \lambda, \gamma)\}.$$

空スタックにより *(via empty stack)* $\mathcal{K}$ が受理する 言語は次の通りである．

$$N(\mathcal{K}) = \{w \mid \text{ ある } p \in Q \text{ について } (q_0, w, k_0) \xrightarrow[\mathcal{K}]{*} (p, \lambda, \lambda)\}.$$

空スタックによる受理を考える場合は，最終状態集合は関係ないので，常に $F = \emptyset$ とする．

決定性

定義 3

次の条件が成り立つとき，プッシュダウンオートマトン $\mathcal{K} = [Q, \Sigma, \Gamma, \delta, q_0, k_0, F]$ は決定的 ***(deterministic)*** であると言われる．

(1) 各 $q \in Q$ と $Z \in \Gamma$ について，もし $\delta(q, \lambda, Z) \neq \emptyset$ ならば，すべての $a \in \Sigma$ について $\delta(q, a, Z) = \emptyset$ である．

(2) すべての $q \in Q$, $Z \in \Gamma$ と $a \in \Sigma \cup \{\lambda\}$ について，$\mathrm{card}(\delta(q, a, Z)) \leqslant 1$ である．

注意点

後者の定義においては，通常の遷移と自発的な遷移の選択を避けるために条件 (1) を課す必要がある．一方，条件 (2) は，任意の段階で選択がないことを保証する．よって，条件 (2) は，決定性有限オートマトンを定義する際に課した条件と似ている．決定性プッシュダウンオートマトンにより受理される言語は，非決定性プッシュダウンオートマトンに関するそれと同様に定義される．すなわち，最終状態による受理と空スタックによる受理をそれぞれ区別する．

注意点

後者の定義においては，通常の遷移と自発的な遷移の選択を避けるために条件 (1) を課す必要がある．一方，条件 (2) は，任意の段階で選択がないことを保証する．よって，条件 (2) は，決定性有限オートマトンを定義する際に課した条件と似ている．決定性プッシュダウンオートマトンにより受理される言語は，非決定性プッシュダウンオートマトンに関するそれと同様に定義される．すなわち，最終状態による受理と空スタックによる受理をそれぞれ区別する．

有限オートマトンを扱う限りは，決定性有限オートマトンにより受理される言語のクラスは，非決定性有限オートマトンにより受理される言語のクラスと同じである．プッシュダウンオートマトンに関しては，同様な結果は得られないことに注意する．

受理モードの比較 I

定理 1

プッシュダウンオートマトン $\mathfrak{K}$ について $L = L(\mathfrak{K})$ とする．このとき，$L = N(\widetilde{\mathfrak{K}})$ なるプッシュダウンオートマトン $\widetilde{\mathfrak{K}}$ が存在する．

受理モードの比較 I

定理 1

プッシュダウンオートマトン $\mathcal{K}$ について $L = L(\mathcal{K})$ とする．このとき，$L = N(\widetilde{\mathcal{K}})$ なるプッシュダウンオートマトン $\widetilde{\mathcal{K}}$ が存在する．

証明： 明らかに，こうした定理はシミレーションを行うことにより証明される．すなわち，$\mathcal{K}$ が最終状態に到達した際にスタックが空になるよう，$\mathcal{K}$ を修正する．そのために，最終状態に到達すること無しに $\mathcal{K}$ がスタックを空にして受理することが無いよう，新たな状態 q_λ と特別なスタック記号 X_0 を導入する．

受理モードの比較 II

$\mathcal{K} = [Q, \Sigma, \Gamma, \delta, q_0, k_0, F]$ を，$L = L(\mathcal{K})$ なる所与のプッシュダウンオートマトンとする．$L = N(\widetilde{\mathcal{K}})$ なるプッシュダウンオートマトン $\widetilde{\mathcal{K}}$ を構成する必要がある．ここでは以下を考える．

$$\widetilde{\mathcal{K}} = [Q \cup \{q_\lambda, \tilde{q}_0\}, \Sigma, \Gamma \cup \{X_0\}, \tilde{q}_0, X_0, \tilde{\delta}, \emptyset] ,$$

ここで，$\tilde{\delta}$ は次の通り定義される：

受理モードの比較 II

$\mathcal{K} = [Q, \Sigma, \Gamma, \delta, q_0, k_0, F]$ を，$L = L(\mathcal{K})$ なる所与のプッシュダウンオートマトンとする．$L = N(\widetilde{\mathcal{K}})$ なるプッシュダウンオートマトン $\widetilde{\mathcal{K}}$ を構成する必要がある．ここでは以下を考える．

$$\widetilde{\mathcal{K}} = [Q \cup \{q_\lambda, \tilde{q}_0\}, \Sigma, \Gamma \cup \{X_0\}, \tilde{q}_0, X_0, \tilde{\delta}, \emptyset] \,,$$

ここで，$\tilde{\delta}$ は次の通り定義される：

$$\begin{aligned}
\tilde{\delta}(\tilde{q}_0, \lambda, X_0) &= \{(q_0, k_0 X_0)\} \\
\tilde{\delta}(q, a, Z) &= \delta(q, a, Z) \text{ for all } q \in Q \setminus F,\ a \in \Sigma \cup \{\lambda\},\ Z \in \Gamma \\
\tilde{\delta}(q, a, Z) &= \delta(q, a, Z) \text{ for all } q \in F,\ a \in \Sigma \text{ and } Z \in \Gamma \\
\tilde{\delta}(q, \lambda, Z) &= \delta(q, \lambda, Z) \cup \{(q_\lambda, \lambda)\} \text{ for all } q \in F,\ Z \in \Gamma \cup \{X_0\} \\
\tilde{\delta}(q_\lambda, \lambda, Z) &= \{(q_\lambda, \lambda)\} \text{ for all } Z \in \Gamma \cup \{X_0\} \,.
\end{aligned}$$

受理モードの比較 III

構成方法から，$\widetilde{\mathcal{K}}$ の開始時，それは $\mathcal{K}$ の初期時点表示で表されるが，自身のスタック記号 X_0 をスタックに付加的にプッシュする．その後，最終状態に到達するまで，$\widetilde{\mathcal{K}}$ は $\mathcal{K}$ を模倣する．$\mathcal{K}$ がある最終状態に到達すると，$\widetilde{\mathcal{K}}$ は $\mathcal{K}$ の模倣を続行するか，あるいは，その状態を q_λ に遷移することができる．もし，$\widetilde{\mathcal{K}}$ が状態 q_λ にあるならば，スタックを空にし，その入力を受理する．

受理モードの比較 IV

$x \in L(\mathcal{K})$ としよう．このとき，次の計算が存在する．

$$\text{ある } q \in F \text{ について } (q_0, x, k_0) \xrightarrow[\mathcal{K}]{*} (q, \lambda, \gamma) .$$

入力 x に対して $\widetilde{\mathcal{K}}$ を考えよう．定義より，$\widetilde{\mathcal{K}}$ は，状態 $\tilde{q}_0$ において，スタック記号 X_0 を伴って動作を開始する．最初の自発的遷移を行い，

$$(\tilde{q}_0, x, X_0) \xrightarrow[\widetilde{\mathcal{K}}]{1} (q_0, x, k_0 X_0) .$$

次に，$\widetilde{\mathcal{K}}$ は $\mathcal{K}$ の動きの各ステップを模倣することができるので，

$$(\tilde{q}_0, x, X_0) \xrightarrow[\widetilde{\mathcal{K}}]{1} (q_0, x, k_0 X_0) \xrightarrow[\widetilde{\mathcal{K}}]{*} (q, \lambda, \gamma X_0)$$

δ の定義における最後の 2 つの遷移を用いて次を得る．

$$(q_0, x, k_0 X_0) \xrightarrow[\widetilde{\mathcal{K}}]{*} (q_\lambda, \lambda, \lambda) , \text{ thus } x \in N(\widetilde{\mathcal{K}}) . \quad \blacksquare$$

受理モードの比較 V

定理 2

プッシュダウンオートマトン $\mathcal{K}$ について $L = N(\mathcal{K})$ とする．このとき，$L = L(\widetilde{\mathcal{K}})$ なるプッシュダウンオートマトン $\widetilde{\mathcal{K}}$ が存在する．

受理モードの比較 V

定理 2

プッシュダウンオートマトン $\mathcal{K}$ について $L = N(\mathcal{K})$ とする．このとき，$L = L(\tilde{\mathcal{K}})$ なるプッシュダウンオートマトン $\tilde{\mathcal{K}}$ が存在する．

証明： ここでも証明はシミュレーションにより行われる．プッシュダウンオートマトン $\tilde{\mathcal{K}}$ は，$\mathcal{K}$ がそのスタックを空にしたことを検出するまで，$\mathcal{K}$ を模倣するであろう．もしそうなった場合は，$\tilde{\mathcal{K}}$ はある最終状態に遷移した後に停止する．形式的な記述はテキストにある． ∎

PDA と $\mathcal{CF}$ I

これまでは，プッシュダウンオートマトンとそれらの受理に関する振舞だけを扱ってきた．如何なる言語がプッシュダウンオートマトンに受理されるかは，まだ明らかではない．このことは次の定理により明らかとなる．ここで，導出の各段階において，最左非終端記号に生成規則が適用されるとき，その導出は最左導出であると言われることを思い出して欲しい．

PDA と $\mathcal{CF}$ II

定理 3

$\mathcal{K} = [Q, \Sigma, \Gamma, \delta, q_0, k_0, \emptyset]$ をプッシュダウンオートマトンとし，$L = N(\mathcal{K})$ とする．このとき，L は文脈自由である．

証明 I

証明： $\mathcal{K} = [Q, \Sigma, \Gamma, \delta, q_0, k_0, \emptyset]$ をプッシュダウンオートマトンとする．$L = N(\mathcal{K})$ が文脈自由であることを証明するためには，$L = L(\mathcal{G})$ なる文脈自由文法 $\mathcal{G}$ を構成する必要がある．
$\mathcal{G} = [\Sigma, N, \sigma, P]$ とし，N を次の通り定義する．N の要素を $[q, A, p]$ で表す．ただし，$p, q \in Q$ および $A \in \Gamma$ である．さらに N は記号 σ を含むとする．次に，生成規則の集合を定義する必要がある．P は次の規則を含むとする．

(1) 各 $q \in Q$ について，$\sigma \rightarrow [q_0, k_0, q]$,

(2) $q_1, \ldots, q_{m+1} \in Q$，および，$m > 0$ において $(q_1, B_1 B_2 \cdots B_m) \in \delta(q, a, A)$ なる $A, B_1, \ldots, B_m \in \Gamma$ について，
$[q, A, q_{m+1}] \rightarrow a[q_1, B_1, q_2][q_2, B_2, q_3] \cdots [q_m, B_m, q_{m+1}]$.

$m = 0$ の場合，生成規則は $[q, A, q_1] \rightarrow a$ である．

証明 II

この証明を理解するためには，$\mathcal{G}$ の非終端記号と生成規則が，$\mathcal{G}$ における記号列 x の最左導出が，入力 x が与えられたプッシュダウンオートマトン $\mathcal{K}$ の模倣となるように定義されていることを理解するとよい．特に，$\mathcal{G}$ における最左導出の任意の時点に現れる非終端記号は，文法が既に生成したのと同じ入力を $\mathcal{K}$ が観測した時点でのスタック上の記号に対応している．別の言い方をすると，ここで意図していることは，もし x が $\mathcal{K}$ に対し，q から始まり p で終るある動作列によって，スタックから A を消去させるならば，かつ，そのときに限り，$[q, A, p]$ が x を導出することである．

証明 III

$L(\mathcal{G}) = N(\mathcal{K})$ を示すために，次を帰納的に証明する．

$$[q, A, p] \overset{*}{\underset{\mathcal{G}}{\Longrightarrow}} x \quad \text{if and only if} \quad (q, x, A) \overset{*}{\underset{\mathcal{K}}{\longrightarrow}} (p, \lambda, \lambda)\,. \tag{1}$$

最初に，i に関する 帰納法により次を示す．

$(q, x, A) \overset{i}{\underset{\mathcal{K}}{\longrightarrow}} (p, \lambda, \lambda)$ ならば $[q, A, p] \overset{*}{\underset{\mathcal{G}}{\Longrightarrow}} x$.

証明 IV

帰納法の基底として，$i = 1$ とする．$(q, x, A) \xrightarrow[\mathcal{K}]{1} (p, \lambda, \lambda)$ を得るためには，$(p, \lambda) \in \delta(q, x, A)$ が成り立たなければならない．結果として，$x = \lambda$ あるいは $x \in \Sigma$ のいずれかである．P の構成法から，両者の場合において $([q, A, p] \to x) \in P$ であることがわかる．よって，$[q, A, p] \overset{1}{\underset{\mathcal{G}}{\Longrightarrow}} x$ である．これが帰納法の基底の証明となる．

証明 V

いま，$i > 0$ と仮定する．$x = ay$ かつ

$$(q, ay, A) \xrightarrow[\mathcal{K}]{1} (q_1, y, B_1B_2 \cdots B_n) \xrightarrow[\mathcal{K}]{i-1} (p, \lambda, \lambda)\,.$$

としよう．記号列 y は $y = y_1y_2 \cdots y_n$ と表すことができる．ここで，y_j は，場合によっては長い動作列の後に，スタックから B_j をポップする働きを有する．つまり，y_1 を，その終端においてスタックが初めて長さ $n-1$ の列になる y の接頭辞とする．また，y_2 は y_1 に続く y の記号列であり，y_2 の終端においてスタックが初めて長さ $n-2$ となるものとする．以下，同様に続く．
この配置を次に示す．

証明 VI

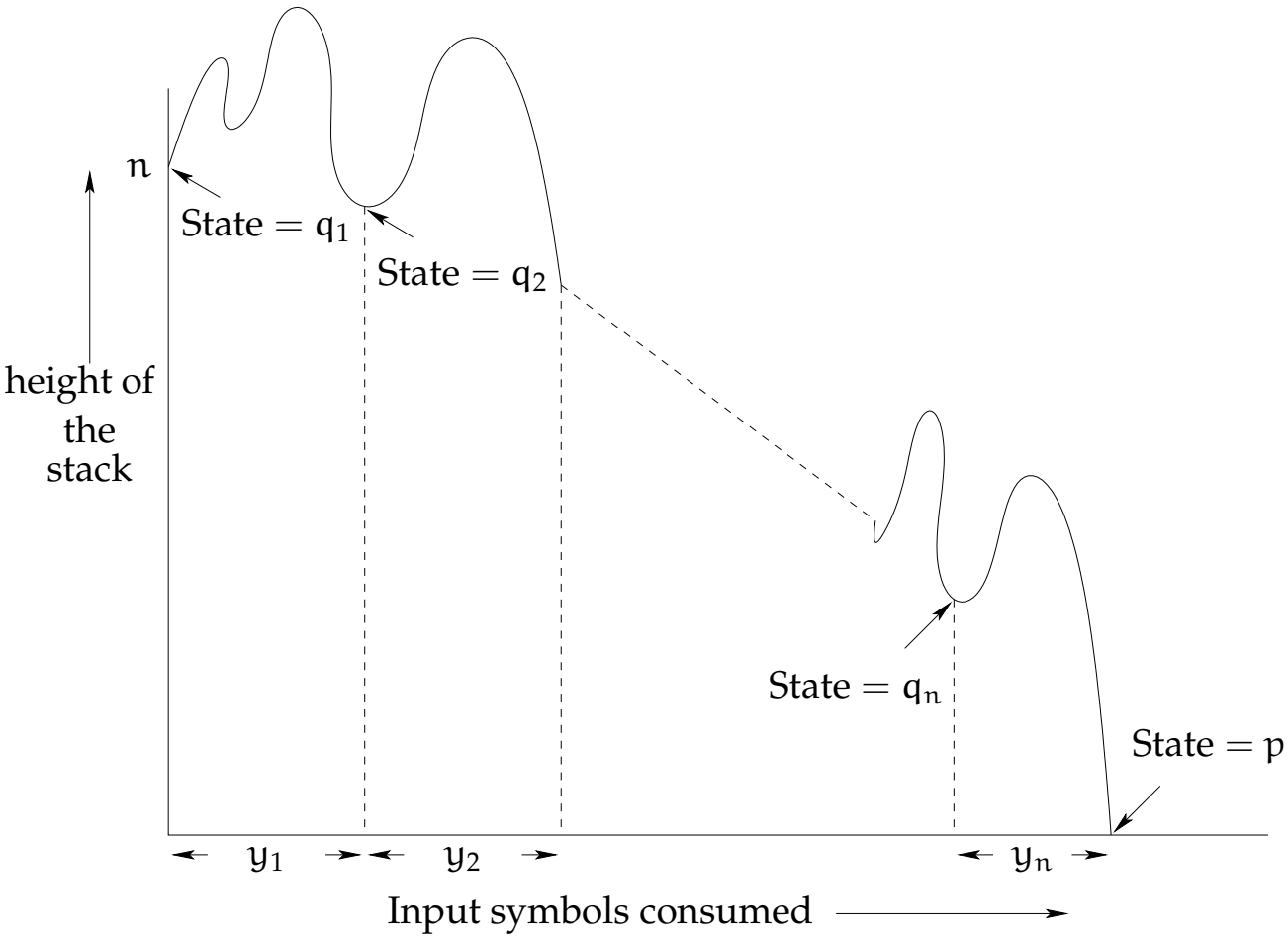

図 2: 消費した入力記号の関数としてのスタックの高さ

証明 VII

$\mathcal{K}$ が y_1 を読み込む間中，B_1 が最後から n 番目のスタック記号である必要はない．なぜなら，もし B_1 がスタックの先頭にあり，ひとつ以上の記号と置き換えられる場合は，変化するかもしれないからである．しかし，y_1 が読まれている間は，$B_2B_3\cdots B_n$ のいずれも決してスタックの先頭になることはない．このように，y_1 が処理されている間は，$B_2B_3\cdots B_n$ のいずれも変化しない，あるいは，計算に影響を及ぼさない．一般には，$y_1\cdots y_{j-1}$ が読まれている間，B_j はスタック上にそのまま残る．

証明 VIII

状態 $q_2, q_3, \ldots, q_{n+1}$ が存在する．ここで $q_{n+1} = p$ である．ただし，i より少ない動作により，

$$(q_j, y_j, B_j) \xrightarrow[\mathcal{K}]{*} (q_{j+1}, \lambda, \lambda)$$

である．q_j は，スタックの長さが初めて $n - j + 1$ となった際に到達する状態である．よって，帰納法の仮定を適用し，次を得る．

$$1 \leqslant j \leqslant n \text{ について } [q_j, B_j, q_{j+1}] \overset{*}{\underset{\mathcal{G}}{\Longrightarrow}} y_j \,.$$

もとの動作

$(q, ay, A) \xrightarrow[\mathcal{K}]{1} (q_1, y, B_1 B_2 \cdots B_n)$ を考えると，次のことがわかる．

$$[q, A, p] \;\Rightarrow\; a[q_1, B_1, q_2][q_2, B_2, q_3] \cdots [q_n, B_n, q_{n+1}] \,.$$

故に $[q, A, p] \overset{*}{\underset{\mathcal{G}}{\Longrightarrow}} a y_1 y_2 \cdots y_n = x$ となり，(1) の ($\Leftarrow$) が示される．

証明 IX

(1) の必要性を示すために，$[q, A, p] \overset{i}{\underset{\mathcal{G}}{\Longrightarrow}} x$ を仮定する．i に関す

る帰納法により $(q, x, A) \overset{*}{\underset{\mathcal{K}}{\longrightarrow}} (p, \lambda, \lambda)$ を証明する．

帰納法の基底として，再び $i = 1$ の場合を考える．もし

$[q, A, p] \overset{1}{\underset{\mathcal{G}}{\Longrightarrow}} x$ ならば，$([q, A, p] \to x) \in P$ であり，故に，

$(p, \lambda) \in \delta(q, x, A)$ である．

証明 X

次に，帰納のために，次を仮定する．

$$[q, A, p] \Rightarrow a[q_1, B_1, q_2] \cdots [q_n, B_n, q_{n+1}] \overset{i-1}{\underset{\mathcal{G}}{\Longrightarrow}} x\,,$$

ここで，$q_{n+1} = p$. x を $x = ax_1 \cdots x_n$ と表す．ただし，

$j = 1, \ldots, n$ について，$[q_j, B_j, q_{j+1}] \overset{*}{\underset{\mathcal{G}}{\Longrightarrow}} x_j$ である．さらに，各導出のステップは i より少ない．よって，帰納法の仮定を適用し次を得る．

$$(q_j, x_j, B_j) \overset{*}{\underset{\mathcal{K}}{\longrightarrow}} (q_{j+1}, \lambda, \lambda) \text{ for } j = 1, \ldots, n\,.$$

証明 XI

もし，上記の時点表示の列にいてそれぞれのスタックの最後に $B_{j+1}\cdots B_n$ を挿入すると，次を得る．

$$(q_j, x_j, B_jB_{j+1}\cdots B_n) \xrightarrow[\mathcal{K}]{*} (q_{j+1}, \lambda, B_{j+1}\cdots B_n)\,. \tag{2}$$

さらに，$[q, A, p]$ からの x の導出の最初のステップより，

$$(q, x, A) \xrightarrow[\mathcal{K}]{1} (q_1, x_1x_2\cdots x_n, B_1B_2\cdots B_n)$$

は，$\mathcal{K}$ の正当な動作であることがわかる．よって，この動作と $j = 1, 2, \ldots, n$ における (2) から，直ちに次を得る．

$$(q, x, A) \xrightarrow[\mathcal{K}]{*} (p, \lambda, \lambda)\,.$$

このことは (1) の必要性を証明している．

証明 XII

この証明は，$q = q_0$ および $A = k_0$ のもとでの (1) が，次のことを述べているとの観測をもって完結する．

$[q_0, k_0, p] \overset{*}{\underset{\mathcal{G}}{\Longrightarrow}} x$ ならば，かつ，そのときに限り $(q_0, x, k_0) \overset{*}{\underset{\mathcal{K}}{\longrightarrow}} (p, \lambda, \lambda)$．

P の構成における規則 (1) と共に，この観測は，ある状態 p について，もし $(q_0, x, k_0) \overset{*}{\underset{\mathcal{K}}{\longrightarrow}} (p, \lambda, \lambda)$ ならば，かつ，そのときに限り $\sigma \overset{*}{\underset{\mathcal{G}}{\Longrightarrow}} x$ であることを述べている．よって，$x \in N(\mathcal{K})$ ならば，かつ，そのときに限り $x \in L(\mathcal{G})$ であることがわかる． ∎

Thank you!