計算論 A / 2015年度

第6章

プッシュダウンオートマトン

角川　裕次

もくじ

第6章　プッシュダウンオートマトン
【教科書 p245〜】

＊＊＊ 本日の重要概念 ＊＊＊

プッシュダウンオートマトン (PDA)
PDA が受理する言語クラス ＝ 文脈自由言語のクラス

ϕ(・ω・)メモメモ

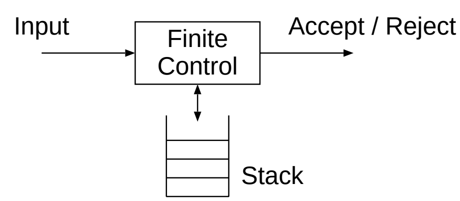
6.1 プッシュダウンオートマトンの定義

6.1.1 直観的な導入

プッシュダウンオートマトン (PDA)

PDA ＝「ε 動作可能な非決定性有限オートマトン」
　　＋「容量無限のスタック」

- スタックはランダムアクセスできない



PDA の特徴

文脈自由言語を認識できる

- 有限オートマトンより強力

文脈自由言語より複雑なものは認識できない

- 認識できないものの例: $\{0^n1^n2^n \mid n \geq 1\}$

重要

PDA が認識する言語のクラス
＝　文脈自由言語のクラス

6.1.2 PDA の形式的な定義

PDA の定義

PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

- Q: 状態の有限集合
- Σ: 入力記号の有限集合
- Γ: 有限のスタックアルファベット
  (スタックに記録できる記号の有限集合)
- δ: 遷移関数 (後のスライドにて)
- q_0: 初期状態
- Z_0: 開始記号
  (これがスタックに1つ置かれた状態でPDAは動作開始)
- F: 受理状態 (最終状態ともいう) の集合

PDA の遷移関数 δ

$(p, \gamma) \in \delta(q, a, X)$

入力

- $q \in Q$: PDA の状態
- $a \in (\Sigma \cup \{\varepsilon\})$: 入力記号 または ε
- $X \in \Gamma$: スタック記号

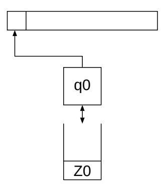
出力

- $p \in Q$: PDA の次の状態
- $\gamma \in \Gamma^*$: スタックアルファベットの列

PDA の動作 (1/3)

初期状態

- 状態は q_0
- スタックには Z_0 のみ

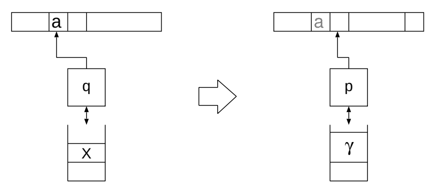


入力を受理する/しないの定義は後ほどで...

PDA の動作 (2/3)

動作: $(p, \gamma) \in \delta(q, \underline{a}, X)$, ただし $a \neq \varepsilon$

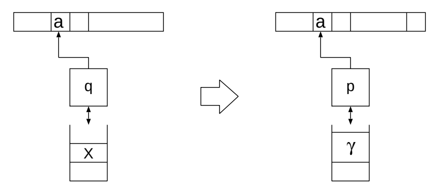
- 状態が q, 入力記号が a, スタックトップが X のとき
- 次の状態を p にする
- 入力記号をひとつ読み進める
- スタックから X をポップして γ をプッシュ



PDA の動作 (3/3)

動作: $(p, \gamma) \in \delta(q, \underline{\varepsilon}, X)$

- 状態が q, スタックトップが X のとき
- 次の状態を p にする
- 入力記号は読み進めない
- スタックから X をポップして γ をプッシュ



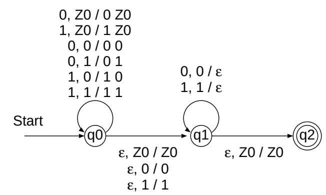
6.1.3 PDA の図表現

PDA の図表現 (1/2)

各状態を円で描く

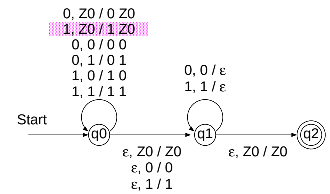
- ただし受理状態は２重の円

初期状態には「出発/Start」ラベルの辺をつける



PDA の図表現 (2/2)

状態 q から p への辺のラベル $a, X/\gamma$:
遷移 $(p, \gamma) \in \delta(q, a, X)$ を表す



例: 状態 q_0 でのラベル $\mathbf{1, Z_0/1Z_0}$

- 状態 q_0, 入力記号が $\mathbf{1}$, スタックのトップが $\mathbf{Z_0}$ のとき
- スタックに $\mathbf{1}$ と $\mathbf{Z_0}$ をプッシュ (1が上, $\mathbf{Z_0}$ が下)

6.1.4 PDA の時点表示

時点表示 (q, w, γ)

PDA の現在状態をもれなく表現したもの

- (＝これから後の動作が分かるに十分な情報)

以下の3つ組で時点表示を表現

1. 状態 q
2. 入力の残り w
3. スタックの内容 γ

関係 $\vdash_P$

PDA の一度の動作による時点表示の変化の関係

$(q, aw, X\beta) \vdash_P (p, w, \alpha\beta)$

- $(p, \alpha) \in \delta(q, a, X)$ に対する遷移
- 注: a は ε の場合あり

単に $\vdash$ と表記する場合あり

- P が文脈から明らかな場合

関係 $\vdash_P^*$

PDA の0回以上の動作による時点表示の変化の関係

基礎: $I \vdash^* I$

帰納: $I \vdash K$ かつ $K \vdash^* J$ ならば $I \vdash^* J$

単に $\vdash^*$ と表記する場合あり

- P が文脈から明らかな場合

6.2 PDA の言語

2通りの受理の定義

定義1: 最終状態による受理

- 入力をすべて読み終えた時に受理状態なら入力を受理
- スタック内容はどうでもいい

定義2: 状態空スタックによる受理

1. 入力をすべて読み終えた時に受理状態かつ空スタックなら入力を受理

これら2通りのPDAの受理能力は同等
→ 都合の良い方を使えばいい

6.2.1 最終状態による受理

最終状態で受理する言語

PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

P が最終状態で受理言語 $L(P)$

$$L(P) = \{w \mid \exists q \in F, (q_0, w, Z_0) \vdash_P^* (q, \varepsilon, \alpha)\}$$

最終状態での受理条件

- 入力をすべて読み終えて受理状態に入る
- そのときのスタックの内容 α はなんでもよい

PDA 構成の例 (例6.1, 例6.2)

言語 $L_{wwr} = \{ww^R \mid w$ は $(\mathbf{0}+\mathbf{1})^*$ の中の列$\}$

PDA の動き (概要, 最終状態による受理)

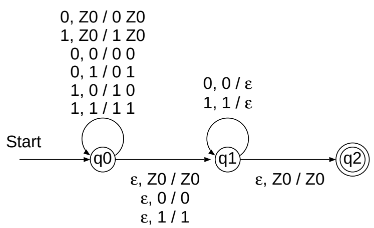
1. w の記号を1つ読む度にそれをスタックにプッシュ
2. w^R の記号を1つ読む度にスタックトップの記号と照合
3. 入力をすべて読み終えたら受理状態に入る

注意点

- w と w^R の境目を表す文字がない (見た目では境目が分からない)
- 非決定的に境目を推測し照合動作に移る

言語 L_{wwr} を最終状態で受理する PDA

言語 $L_{wwr} = \{ww^R \mid w は (0+1)^* の中の列\}$



6.2.2 空スタックによる受理

状態空スタックで受理する言語

PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

P が状態空スタックで受理言語 $N(P)$

$$N(P) = \{w \mid \exists q \in F, (q_0, w, Z_0) \vdash_P^* (q, \varepsilon, \varepsilon)\}$$

状態空スタックでの受理条件

- 入力をすべて読み終えて受理状態に入る
- ただしそのときのスタックは空でないとだめ

6.2.3 空スタックから最終状態へ

空スタック受理PDAを模倣する最終状態受理PDA

$P_N = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$: 任意の空スタック受理PDA

- 任意のものが与えられる

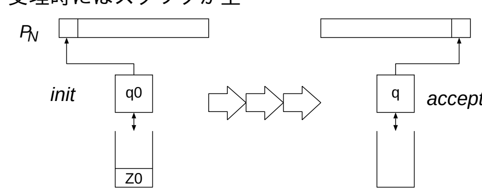
$N(P_N) = L(P_F)$ である最終状態受理PDA P_F が存在

証明方針: P_N の動作を模倣する P_F を構成

P_N の動作の観察

$P_N = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$

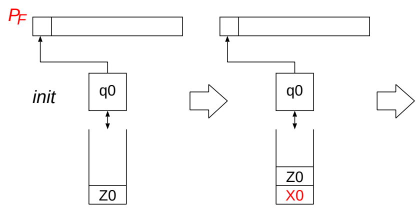
- 動作開始時の状態は q_0 でスタックには Z_0
- 受理時にはスタックが空



P_F の構成方針 (1/3)

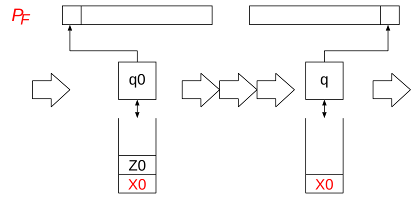
最初, スタックの Z_0 の下に X_0 を置く

- 空スタックを検知できるようにするため



P_F の構成方針 (2/3)

P_N の動作を模倣してゆく

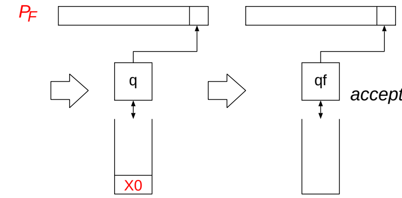


- P_N がスタックの X_0 へアクセスすることはない (P_N はそういう動作になっている)

P_F の構成方針 (3/3)

スタックトップが X_0 なら受理状態 q_f に遷移

- P_N は入力を受理しているので



6.2.4 最終状態から空スタックへ

最終状態受理PDAを模倣する空スタック受理PDA

$P_F = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$: 任意の最終状態受理PDA

- 任意のものが与えられる

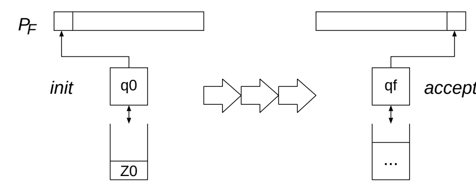
$L(P_F) = N(P_N)$ である
空スタック受理PDA P_N が存在

証明方針: P_F の動作を模倣する P_N を構成

P_F の動作の観察

$P_F = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

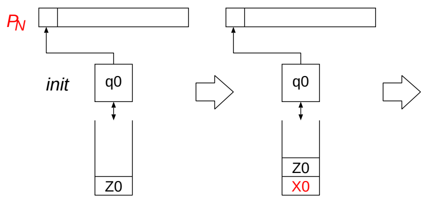
- 動作開始時の状態は q_0 でスタックには Z_0
- 受理時には受理状態 $\in F$
- そのときスタックは空とは限らない



P_N の構成方針 (1/3)

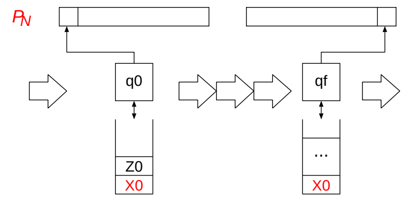
最初にスタックの Z_0 の下に(特別な記号)X_0 を置く

- 空スタックを検知できるようにするため
- 他では使われない記号を X_0 に選ぶ



P_N の構成方針 (2/3)

P_F の動作を模倣してゆく



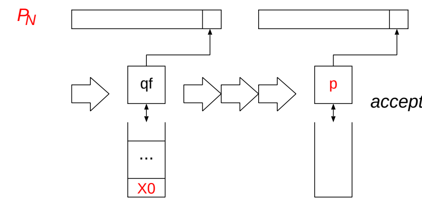
- P_F がスタックの X_0 へアクセスすることはない
  (P_F はそのように作られている)

P_N の構成方針 (3/3)

状態が P_F の受理状態なら(特別な)状態 p に遷移

次にスタックをポップし続ける (X_0 をポップしたら終了)

- P_F は入力を受理 $\Rightarrow$ P_N は状態空スタックで受理
- P_F は入力を棄却 $\Rightarrow$ P_N は棄却



ここまでのまとめ

最終状態受理のPDAが受理する言語のクラス
＝ 空スタック受理のPDAが受理する言語のクラス

PDA の定義にどちらを用いても本質的に同じ

- 使いやすい方を選べば良い

6.3 PDA と CFG の等価性

6.3節の内容

> PDA が受理する言語のクラス
> ＝
> 文脈自由言語のクラス

証明は以下の2つで構成

1. 任意の文脈自由文法 G に対し
   G の言語を受理する PDA が存在
   ✔ 文脈自由言語のクラス $\subseteq$ PDA が受理する言語のクラス

2. 任意の PDA P に対し
   P が受理する言語は文脈自由言語
   ✔ PDA が受理する言語のクラス $\subseteq$ 文脈自由言語のクラス

6.3.1 文法から プッシュダウンオートマトンへ

6.3.1 文法からプッシュダウンオートマトンへ

G: 任意の文脈自由文法 (入力)

行なうこと (出力):
G の言語を受理する空スタック受理 PDA P を構成

方針

- P は G の最左導出を模倣
  - ✔ 注: 任意の語に対して最左導出が存在
- 入力 w が導出できれば P は受理

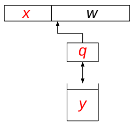
文法 G の言語を受理する PDA の概要

G での導出の途中 $S \overset{*}{\Rightarrow} xy$ を観察

- x は終端記号だけの列
- y は変数/終端記号を含む列 (これから生成規則を適用)

対応する PDA の動作: $(q, Z_0, xw) \vdash^* (q, y, w)$

- x は読み終えた終端記号列
- (最終的には $y \overset{*}{\Rightarrow} w$ を検査)
- y のトップが変数: 生成規則を適用
- y のトップが終端記号: w の文字と照合



文法 G の言語を受理する PDA の構成(1/3)

動作開始時: G の出発記号 S をスタックにプッシュ

文法 G の言語を受理する PDA の構成(2/3)

生成規則を非決定的に推測(適用)する遷移規則

- $\delta(q, \varepsilon, A) = \{(q, \beta) \mid 生成規則 A \to \beta が存在\}$

考え: 可能な生成規則すべてを非決定的に適用

- ヘッドの右方の文字は読めないために
  適用する生成規則を1つに絞れない
- 適用が正しいかどうかは後から照合 (非決定的動作)

文法 G の言語を受理する PDA の構成(3/3)

非決定的な推測の正しさを照合する遷移規則

- $\delta(q, a, a) = \{(q, \varepsilon)\}$ に合致するか照合

考え: 入力ヘッドとスタックトップの記号を照合

入力ヘッドとスタックトップの記号が一致: 動作を継続

- 導出になっている非決定計算だけを継続

一致しない: その非決定計算は棄却として動作を停止

- 適用した生成規則が導出になってないときは一致しない

6.3.2 プッシュダウンオートマトン
から文法へ

6.3.2 プッシュダウンオートマトンから文法へ

P: 任意の空スタック受理PDA (入力)

行なうこと (出力):
P が受理する言語を生成する文法 G を構成

方針

- P の遷移関数の各要素に対応した生成規則を作る

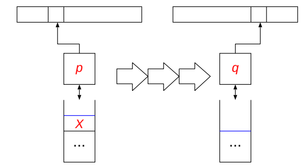
文法の変数

各変数は $[pXq]$ の形

- これでひとつの変数
- 3つの成分: p, X, Q

$[pXq]$ は PDA P の以下の動作に対応

- スタックトップの記号 X が消去される
- X を消去するまでに状態は p から q へ遷移



PDA から文法を構成: 変数

文法の変数は3つ組 $[pXq]$

- 状態p, q とスタック記号X すべてに対して

S : 開始記号

- これだけは3つ組でない

PDA から文法を構成: 生成規則 (1/2)

生成規則 $S \to [q_0, Z_0, p]$

- 各状態 p に対して
- q_0: PDA の初期状態
- Z_0: PDA の初期スタック記号

※ 変数 $[q_0, Z_0, p]$ で $(q_0, Z_0, w) \vdash^* (p, \varepsilon, \varepsilon)$ を表す

- 空スタック受理

PDA から文法を構成: 生成規則 (2/2)

PDAが遷移規則 $(r, Y_1Y_2\cdots Y_k) \in \delta(q, a, X)$ を持つ場合
文法に以下の生成規則を持たせる

$$[qXr_k] \to a[rY_1r_1][r_1Y2r_2]\cdots[r_{k-1}Y_kr_k]$$

- 各状態 $r_1, r_2, ..., r_k$ に対して

この生成規則の考え方

- 終端記号 a を1文字導出
- スタックのトップが $Y_1, Y_2, ..., Y_k$ それぞれで終端記号列を導出

PDA から文法を構成: PDA と文法の関係

一般的性質:

$[qXp] \overset{*}{\Rightarrow} w$ であるときおよびその時のみに限り

$(q, w, X) \vdash^* (p, \varepsilon, \varepsilon)$

結論:

$S \overset{*}{\Rightarrow} w$ であるときおよびその時のみに限り

$(q_0, w, Z_0) \vdash^* (p, \varepsilon, \varepsilon)$

6.4 決定性プッシュダウンオートマトン

決定性プッシュダウンオートマトン (DPDA) (1/2)

受理できる言語は文脈自由言語の部分クラス

- C言語等の主要プログラミング言語が含まれている
- (DPDA で受理できる言語を選んでいる)

コンパイラの構文解析器の正体は DPDA

- 非決定的動作がないので時間効率が良い

応用上とても重要: コンパイラの講義で詳しくやります

決定性プッシュダウンオートマトン (DPDA) (2/2)

例: C言語プログラム

```
void foo(void) {
  static int bar;
  bar += xyzzy();
}
```

使用変数が事前に宣言されている言語 $L = \{wcw\}$

- 前の w : 宣言
- 後ろ w : 使用

→ L は文脈自由言語でない (非決定性 PDA でもだめ)

Q. コンパイラはどうやっているのでしょうか?

- A. コンパイラの講義で勉強してください

6.4.1. 決定性 PDA の定義

決定性 PDA: 可能な動作が高々1つのPDA

$\delta(q, a, X)$ はたかだかひとつの要素を持つ

- 各状態 q, 各記号 a (ε含む), 各スタック記号 X に対し
- (1文字読み動作は高々1通りだけ)

もし $\delta(q, a, X)$ が空でなければ $\delta(q, \varepsilon, X)$ は空

- 各状態 q, 各記号 a に対して
- 状態 q, スタックトップ X のときの動作は1つだけ (1文字読み動作か ε 動作かどちらか1つだけ)

例6.16

言語 $L_{wwr} = \{ww^R \mid w$ は $(0+1)^*$ の中の列$\}$
は決定性PDAでは受理できない

- 境目が分からない
- 非決定性PDAだと受理できる(境目を非決定的に推測)

言語 $L_{wcwr} = \{wcw^R \mid w$ は $(0+1)^*$ の中の列$\}$
は決定性PDAで受理できる

- w の文字をスタックに順次積む
- 境目の c を見たら照合動作に移る
- w^R の文字とスタック上の文字を順次照合

6.4.2 正則言語と決定性PDA

正則言語と決定性PDA (DPDA)の受理する言語のクラス

定理6.17
任意の正則言語 L に対し
L を受理する最終状態 DPDA が存在

証明

- PDAのスタックは使わない
- 内部状態の遷移だけ
- DPDA には決定性オートマトンと同じ動作をさせる

【証明おわり】

6.4.3 DPDA と文脈自由言語

DPDA と文脈自由言語

最終状態受理の DPDA が受理する言語のクラス

- 正則言語のクラスを真に含む
- 文脈自由言語のクラスに真に含まれる

6.4.4 DPDA とあいまいな文法

DPDA の受理する言語は
本質的にあいまいでない文脈自由言語の部分クラス

定理6.20
ある DPDA P について $L = N(P)$ ならば
L はあいまいでない文脈自由言語で記述できる

定理6.21
ある DPDA P について $L = L(P)$ ならば
L はあいまいでない文脈自由言語で記述できる

おわり

教科書280ページ 問6.4.2 (c)