形式言語とオートマトン チートシート

1. 基本概念と有限オートマトン

重要公式・定義

- **言語の形式的定義**
  - **記号 (Symbol)**: データを表す最小単位。（例: $a, b, 0, 1$）
  - **アルファベット (Σ)**: 記号の有限集合。（例: $\Sigma = a, b$）
  - **語 (Word)**: アルファベットの元の有限個の系列。（例: a, ab, aba）
  - **空語 (ϵ or λ)**: 長さ0の語。
  - Σ^*: Σ上の全ての語の集合（空語を含む）。
  - **言語 (Language)**: あるアルファベット Σ 上の語の集合。Σ^* の部分集合となる。
- **正規表現 (Regular Expression)**
  - 正規言語を代数的に表現するための表記法。
  - **基本演算**:
    - **和集合 (Union)**: $R_1 + R_2$（言語 $L(R_1)$ または $L(R_2)$）
    - **連接 (Concatenation)**: R_1R_2（言語 $L(R_1)$ の語と言語 $L(R_2)$ の語を連結）
    - **クリーネ閉包 (Kleene Star)**: R^*（言語 $L(R)$ の語を0回以上繰り返して連結）
  - **例**: アルファベット $\Sigma = a, b$ 上で、「aで始まりbで終わる」を表す正規表現は $a(a+b)^*b$ と書ける。
  - **クリーネの定理**: ある言語が正規表現で表せることと、その言語が有限オートマトンで受理できることは等価である。
- **決定性有限オートマトン (DFA) の形式的定義**
  - DFAは5項組 $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ で定義される。
    - Q: 状態の有限集合
    - Σ: 入力記号の有限集合（アルファベット）
    - δ: 動作関数 ($Q \times \Sigma \to Q$)
    - q_0: 初期状態 ($q_0 \in Q$)
    - F: 受理状態の有限集合 ($F \subseteq Q$)
- **非決定性有限オートマトン (NFA) の形式的定義**
  - 動作関数 δ の出力先が状態の集合（べき集合）となる。
  - $\delta : Q \times \Sigma \to 2^Q$
- **ϵ-NFA の形式的定義**
  - 動作関数 δ が空語 ϵ による遷移を許す。
  - $\delta : Q \times (\Sigma \cup \epsilon) \to 2^Q$

主要アルゴリズム

- **ϵ-NFAからNFAへの変換（ϵ除去）**

    1. **ϵ閉包の計算**: 各状態 q に対して、状態 q からϵ遷移のみで到達可能な状態の集合 $E(q)$ を求める（q自身も含む）。
    2. 新しい動作関数の定義: 新しい動作関数 $\delta'(q, a)$ は、$E(q)$ 内の各状態から記号 a で遷移可能な全ての状態を求め、さらにその各々のϵ閉包をとったものの和集合となる。 $\delta'(q, a) = \bigcup_{p \in E(q)} E(\delta(p, a))$
    3. **新しい受理状態**: 元の受理状態 F の要素を1つでも含むようなϵ閉包を持つ状態を、新しい受理状態とする。$F' = q \in Q \mid E(q) \cap F \neq \emptyset$

- **NFAからDFAへの変換（決定化 / 部分集合構成法）**

    1. NFAの状態集合 Q_n のべき集合 2^{Q_n} の各要素を、DFAの新しい状態とみなす。
    2. DFAの初期状態は、NFAの初期状態のϵ閉包 $E(q_0)$ とする。
    3. DFAの受理状態は、NFAの受理状態を1つでも含む状態集合とする。
    4. NFAのある状態集合 S から入力記号 a によって遷移可能な状態を全て求め、それを遷移先の状態集合とする。$\delta_D(S, a) = \bigcup_{q \in S} \delta'(q, a)$（$\delta'$は$\epsilon$除去後の動作関数）
    5. この操作を、新しい状態が生成されなくなるまで繰り返す。

- **DFAの最小化**

    1. DFAの状態を「受理状態」と「非受理状態」の2つのグループに分割する。
    2. 各グループ内で、全ての入力記号に対して遷移先が同じグループに属さない状態があれば、その状態を新しいグループに分割する。
    3. これ以上分割できなくなるまでステップ2を繰り返す。
    4. 最終的に同じグループに残った状態を1つの状態にまとめることで、最簡形のDFAが完成する。

基本用語

- **オートマトン (Automaton)**: 入力に応じて内部状態を遷移させ、結果を出力する計算装置の抽象的な数学モデル。
- **状態遷移図**: オートマトンの状態と遷移を視覚的に表現した図。状態をノード（円）、遷移をエッジ（矢印）で表す。
- **受理 (Accept)**: オートマトンが入力された語を処理し終えたとき、受理状態にあれば、その語は「受理された」という。
- **決定性 (Deterministic)**: ある状態において、次の状態が入力記号によって一意に決まる性質。
- **非決定性 (Non-deterministic)**: ある状態において、次の状態の候補が複数存在しうる性質。

---

2. 正規言語と文脈自由言語

重要公式

- **正規言語の反復補題 (Pumping Lemma for Regular Languages)**
    - 任意の正規言語 L に対して、ある整数 n が存在し、$|z| \geq n$ となるような L の任意の語 z は、$z = uvw$ と分解できる。ここで、$|uv| \leq n, |v| > 0$ であり、全ての $i \geq 0$ に対して $uv^iw \in L$ が成り立つ。
    - **応用**: ある言語が**正規言語でない**ことを証明するために用いる。

- **文脈自由言語の反復補題 (Pumping Lemma for Context-Free Languages)**
    - 任意の文脈自由言語 L に対して、ある整数 n が存在し、$|z| \geq n$ となるような L の任意の語 z は、$z = uvwxy$ と分解できる。ここで、$|vwx| \leq n$, $|vx| \neq \epsilon$ であり、全ての $i \geq 0$ に対して $uv^iwx^iy \in L$ が成り立つ。

主要アルゴリズム

- **有限オートマトン (FA) から正規文法 (RG) への変換**
    - FAの各状態 q を非終端記号 D_q に対応させる。
    - 遷移 $\delta(q, a) = p$ がある場合、生成規則 $D_q \to aD_p$ を追加する。
    - 遷移先 p が受理状態の場合、さらに $D_q \to a$ を追加する。
- **正規文法 (RG) から有限オートマトン (FA) への変換**
    - RGの各非終端記号 A をFAの状態 q_A に対応させ、さらに新しい受理状態 q_F を追加する。
    - 規則 $A \to aB$ がある場合、遷移 $\delta(q_A, a) = q_B$ を追加する。
    - 規則 $A \to a$ がある場合、遷移 $\delta(q_A, a) = q_F$ を追加する。

基本用語

- **形式文法 (Formal Grammar)**: $G = \langle N, \Sigma, P, S \rangle$ の4項組で定義される、言語を生成するための規則の集合。
    - N: 非終端記号の有限集合。
    - Σ: 終端記号の有限集合。
    - P: 書換規則（生成規則）の有限集合。
    - S: 開始記号（初期記号）。
- **導出 (Derivation)**: 開始記号から始めて、生成規則を繰り返し適用し、語を生成する過程。
- **チョムスキー標準形 (Chomsky Normal Form / CNF)**: 全ての生成規則が $A \to BC$ または $A \to a$ の形に限定された文脈自由文法の一形式。
- **正規言語の閉包性**: 正規言語のクラスは、**和集合**、**連接**、**クリーネ閉包**、**補集合**、**共通部分**の演算について閉じている。つまり、正規言語同士をこれらの演算で組み合わせても、結果は必ず正規言語になる。

---

3. プッシュダウンオートマトン (PDA)

重要公式・定義

- **プッシュダウンオートマトンの形式的定義**
    - PDAは7項組 $M = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$ で定義される。
        - Q, Σ, q_0, F: 有限オートマトンと同様。
        - Γ: スタック記号の有限集合。
        - Z_0: スタックの初期記号 ($Z_0 \in \Gamma$)。
        - δ: 動作関数。$\delta : Q \times (\Sigma \cup \epsilon) \times \Gamma \to 2^{Q \times \Gamma^*}$
- **動作関数の表記**: $\delta(q, a, Z) = (p, \alpha)$

    - 状態 q で、入力が a で、スタックの先頭が Z のとき、状態 p に遷移し、Z を α で置き換える（Z をポップし、α をプッシュする）。

基本用語

- **プッシュダウンオートマトン (PDA)**: 有限オートマトンに**スタック**と呼ばれるLIFO (Last-In, First-Out) 方式の外部メモリを追加したモデル。FAでは受理できない $L = a^n b^n \mid n \geq 1$ のような言語を受理できる。
- **PDAの動作原理**: スタックを利用することで、記号の数を数えたり（例: $a^n b^n$）、語の順序を逆転させて照合したり（例: wcw^R）できる。
    - $L = a^n b^n \mid n \geq 1$: a を読むたびにスタックに記号をプッシュし、b を読むたびにポップする。入力終了時にスタックが空になれば受理。
    - $L = wcw^R \mid w \in a, b^*$: c を読むまで入力記号をスタックにプッシュし、c を読んだ後は、入力記号とスタックからポップした記号が一致するかを確認していく。
- **決定性PDA (DPDA) と 非決定性PDA (NPDA)**: FAと異なり、NPDAはDPDAよりも真に能力が高い。$L(\mathrm{DPDA}) \subsetneq L(\mathrm{NPDA})$。例えば、回文言語 ww^R はNPDAでのみ受理可能。

---

4. チューリング機械とチョムスキー階層

主要アルゴリズム

- **非決定性TMの決定性TMによるシミュレート**
    - 非決定性TMの全ての計算経路（分岐）を決定性TMが体系的に探索することでシミュレーションが可能。
    - 3本のテープを使用する: 1本目は元の入力用、2本目は分岐の選択肢を記録、3本目はシミュレーションの作業用。
    - これにより、$L(\mathrm{DTM}) = L(\mathrm{NTM})$ が示される。

基本用語

- **チューリング機械 (Turing Machine / TM)**: 左右に移動可能で読み書きできる無限長の**テープ**を外部メモリとして持つオートマトン。計算可能なあらゆる問題を解く能力を持つとされる計算モデルの基礎。
- **TMの動作原理**: テープ上を自由に行き来できるため、PDAでは不可能な複雑な照合が可能になる。
    - $L = a^n b^n c^n \mid n \geq 1$: テープ上の a, b, c に順番に印をつけながら、数が一致するかを行き来して確認する。
    - $L = ww \mid w \in a, b^*$: 前半部分の記号に印をつけ、後半部分の対応する記号と照合する、という操作を繰り返す。
- **線形拘束オートマトン (Linear Bounded Automaton / LBA)**: テープの使用が入力語の長さに比例する範囲に制限された非決定性チューリング機械。
- **チョムスキー階層 (Chomsky Hierarchy)**: 形式文法を、その生成規則の制約の強さによって4つのクラスに分類したもの。
    - **タイプ0 (句構造文法)**: 制限なし。**チューリング機械**に対応。
    - **タイプ1 (文脈依存文法)**: $|\alpha| \leq |\beta|$（ただし $S \to \epsilon$ を除く）。**線形拘束オートマトン**に対応。
        - **言語例**: $L = a^n b^n c^n \mid n \geq 1$, $L = ww \mid w \in a, b^*$
    - **タイプ2 (文脈自由文法)**: $A \to \beta$。**非決定性プッシュダウンオートマトン**に対応。
        - **言語例**: $L = a^n b^n \mid n \geq 1$, $L = wcw^R \mid w \in a, b^*$

- **タイプ3 (正規文法)**: $A \to aB$ または $A \to a$。**有限オートマトン**に対応。
  - **言語例**: $L = a^n \mid n \geq 0$, $L = (ab)^*$
- 包含関係は $L(\text{タイプ}3) \subset L(\text{タイプ}2) \subset L(\text{タイプ}1) \subset L(\text{タイプ}0)$ となる。