有限オートマトン (12/16 配布)

(決定性) 有限オートマトン $M = (Q, \Sigma, \delta, s, F)$ の形式的定義

- Q : 有限集合 $\cdots$ 状態の集合
- Σ : 有限集合 $\cdots$ 入力文字の集合 ("alphabet")
- $\delta : Q \times \Sigma \longrightarrow Q$: 遷移関数
- $s \in Q \cdots$ 初期状態
- $F \subset Q \cdots$ 受理状態の集合

(決定性) 有限オートマトンによる語の受理

有限オートマトン $M = (Q, \Sigma, \delta, s, F)$ が語 $w = a_1 a_2 \cdots a_n \in \Sigma^*$ を受理 (accept) する
$\Longleftrightarrow \exists r_0, r_1, \ldots, r_n \in Q$:

- $r_0 = s$
- $\delta(r_{i-1}, a_i) = r_i \ (i = 1, \ldots, n)$
- $r_n \in F$

非決定性有限オートマトン $M = (Q, \Sigma, \delta, s, F)$ の形式的定義

- Q : 有限集合 $\cdots$ 状態の集合
- Σ : 有限集合 $\cdots$ alphabet, $\quad \Sigma_\varepsilon := \Sigma \cup \{\varepsilon\}$
- $\delta : Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$: 遷移関数 $\quad \cdots$ 可能な遷移先全体の集合
  ($\mathcal{P}(Q)$ は Q の冪集合)
- $s \in Q \cdots$ 初期状態
- $F \subset Q \cdots$ 受理状態の集合

非決定性有限オートマトンによる語の受理

非決定性有限オートマトン $M = (Q, \Sigma, \delta, s, F)$ が語 $w \in \Sigma^*$ を受理する
$\Longleftrightarrow \exists a_1, a_2, \cdots, a_n \in \Sigma_\varepsilon, \exists r_0, r_1, \ldots, r_n \in Q$:

- $w = a_1 a_2 \cdots a_n$
- $r_0 = s$
- $r_i \in \delta(r_{i-1}, a_i) \ (i = 1, \ldots, n)$
- $r_n \in F$

決定性有限オートマトンと非決定性有限オートマトンとの同等性

与えられた非決定性有限オートマトン $M = (Q, \Sigma, \delta, s, F)$ に対し、M が認識する言語 $L(M)$ を認識する決定性有限オートマトン $\widetilde{M} := (\widetilde{Q}, \Sigma, \widetilde{\delta}, \widetilde{s}, \widetilde{F})$ が次で構成できる :

- $\widetilde{Q} := \mathcal{P}(Q)$: 現在あり得る状態全体の集合
- $\widetilde{\delta} : \widetilde{Q} \times \Sigma \longrightarrow \widetilde{Q}$: $\widetilde{q}$ の何処かから入力 x で遷移できる状態全体

$$(\widetilde{q}, x) \longmapsto \widetilde{\delta}(\widetilde{q}, x) := \bigcup_{i=0}^{\infty} \widetilde{q_i}$$

ここで、$\widetilde{q_0} := \bigcup_{q \in \widetilde{q}} \delta(q, x), \quad \widetilde{q_{i+1}} := \bigcup_{q \in \widetilde{q_i}} \delta(q, \varepsilon)$

- $\widetilde{s} := \bigcup_{i=0}^{\infty} \widetilde{s_i}$: 初期状態 s から入力を何も読まずに遷移できる状態全体

ここで、$\widetilde{s_0} := \{s\}, \ \widetilde{s_{i+1}} := \bigcup_{q \in \widetilde{s_i}} \delta(q, \varepsilon)$

- $\widetilde{F} := \{\widetilde{q} \in \widetilde{Q} \,|\, \widetilde{q} \cup F \neq \emptyset\}$: あり得る状態のどれかが受理状態

主なレポート課題の例 (続き) (12/16 配布)

問 8. (決定性) 有限オートマトン $M = (Q, \Sigma, \delta, s, F)$ に対し、次のものを記述してみよ。(集合・写像などを用いた概念記述の練習)

(1) 語 $w \in \Sigma^*$ を読んだ後の状態を与える関数 $\widetilde{\delta} : \Sigma^* \to Q$ (ヒント: 語の長さ $|w|$ に関する帰納的定義を用いよ)
(2) より一般に、状態 $q \in Q$ にいる所から出発して語 $w \in \Sigma^*$ を読んだ後の状態を与える関数 $\widetilde{\delta} : Q \times \Sigma^* \to Q$ (記号の濫用だが、$\widetilde{\delta}(s, w)$ を単に $\widetilde{\delta}(w)$ と書く、ということにして上問との整合性を取ることにする)
(3) M が認識する言語 $L(M)$
(4) より一般に、語 $w \in \Sigma^*$ を読んだ後に、続けて読めば受理される語全体の成す集合 (これも 1 つの言語) を与える関数 $S_M : \Sigma^* \to \mathcal{P}(\Sigma^*)$
(5) その他、議論に必要または便利と思われる概念を定式化せよ。

問 9. 前問の S_M と同様に、言語 L に対して、語 $w \in \Sigma^*$ の後に連接すると L の元になる語全体の成す集合を $S_L(w)$ とする。

(1) 関数 $S_L : \Sigma^* \to \mathcal{P}(\Sigma^*)$ を記述せよ。(前問との整合性としては $S_M = S_{L(M)}$)
(2) $a \in \Sigma$ に対し、$\ell_a : \Sigma^* \to \Sigma^*$ (resp. r_a) を、語に左 (resp. 右) から文字 a を連接させる関数とする (これもきちんと記述せよ)。$\ell_a^{-1}(S_L(w)) = S_L(r_a(w))$ を示せ。
(3) L が有限オートマトンで認識可能 $\iff \mathrm{Im}S_L$ が有限集合
(ヒント: $\Longrightarrow$ 側: $L = L(M)$ とするとき、$\widetilde{\delta}(w_1) = \widetilde{\delta}(w_2) \Longrightarrow S_L(w_1) = S_L(w_2)$. $\Longleftarrow$ 側: $\mathrm{Im}S_L$ を状態集合とする有限オートマトンを構成せよ)

問 10. $\Sigma = \{a, b\}$ を alphabet とする次の言語について、(a) 正規表現で表せ。(b) 生成規則で表せ。(c) 受理する非決定性有限オートマトンを構成せよ。(d) 受理する決定性有限オートマトンを構成せよ。(e) 上記の有限オートマトンを模倣するプログラムを作成せよ (言語は何でも良い)。

(1) a で始まり b で終わる
(2) a, b の片方しか現われない
(3) a が偶数個
(4) その他、適当に非自明で興味深い例を考えよ

問 11. $\Sigma = \{a, b\}$ を alphabet とする次の言語について、(a) 生成規則で表せ。(b) 受理する (非決定性) プッシュダウンオートマトンを構成せよ。(c) 上記のプッシュダウンオートマトンを模倣するプログラムを作成せよ (言語は何でも良い)。

(1) $\{a^n b^n | n \geq 0\}$
(2) 偶数文字の回文
(3) a, b が同数現われる
(4) その他、適当に非自明で興味深い例を考えよ

問 12. $\Sigma = \{a, b, +, -, \times, /, (,)\}$ から成る "文法に適っている" 数式を、

(1) 生成規則で表せ。
(2) 受理する (非決定性) プッシュダウンオートマトンを構成せよ。
(3) 上記のプッシュダウンオートマトンを模倣するプログラムを作成せよ。

但し、簡単のため、考える上で多少の制約・許容を行なって良い ($+, -$ を単項演算子としては使わないとか、普通なら付けない括弧が付いても気にしないとか)。

問 13. 例えば、ローマ字仮名変換のように、入力を 1 文字づつ読込みながら適宜出力をしてゆくような処理を定式化するために、"出力付き有限オートマトン" とでも言うべき**有限状態変換器 (finite state transducer)** と呼ばれる計算モデル (処理モデル) を考えてみよう。

(1) この計算モデルを定式化せよ。
(2) その下で、ローマ字仮名変換を実装してみよ (これを実行する "出力付き有限オートマトン" を構成せよ)。例えば、次のような制限下でも良い。
- 入力文字集合 $\Sigma = \{\mathtt{a}, \mathtt{o}, \mathtt{k}, \mathtt{r}, \mathtt{y}\}$
- 出力文字集合 $\Gamma =$ { あ, お, か, こ, ら, ろ, や, よ, き, り, ゃ, ょ, っ}