ソフトウェアモデル論（2012年度）
第6回・2012/11/02

桑原 寛明
情報理工学部 情報システム学科

有限オートマトンから正規表現への変換（復習）

- 有限オートマトン $M = (\{1,\dots,n\}, \Sigma, \delta, 1, F)$
- r_{ij}^{k} を求める
  - 状態 i から状態 j へ k 以下の状態のみを通過して到達する記号列の正規表現
  - r_{ij}^{0} から順に帰納的に
- $r_{1f1}^{n} + \dots + r_{1fl}^{n}$ が初期状態から受理状態へ到達する記号列の正規表現
  - { f1, ..., fl } は受理状態の集合

r_{ij}^{0} を求める （復習）

- 状態 i から状態 j へ直接到達
- $\delta(i,a) = j$ を満たす a の集合を $\{ a_1, \dots, a_l \}$ とする
  - そのような a がなければ $\emptyset$
- $i \neq j$ ならば $r_{ij}^{0} = a_1 + \dots + a_l$
- $i = j$ ならば $r_{ij}^{0} = a_1 + \dots + a_l + \varepsilon$

r_{ij}^{k} を求める （復習）

- $r_{ij}^{k} = r_{ik}^{k-1} (r_{kk}^{k-1})^* r_{kj}^{k-1} + r_{ij}^{k-1}$
- 状態 k を通過するかしないかの2択
  - 通過する
    1. i から k-1 以下を通過して初めて k に到達
    2. k-1 以下のみ通過してまた k に到達（を繰り返す）
    3. k から k-1 以下を通過して j に到達
  - 通過しない
    - i から k-1 以下の状態のみ通過して j に到達

正規言語 （復習）

- 正規文法が生成する言語
- 有限オートマトンが受理できる言語
- 正規表現で表現できる言語

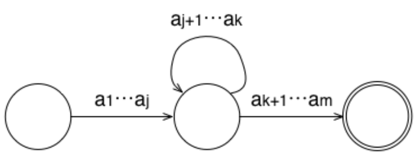
ある言語が正規言語か否か判定するにはどうすればよいか

有限オートマトンが長い語を受理する場合 （復習）

- 初期状態から受理状態に到達する間にループが存在する
  - 同じ状態を2度以上通過する

部屋割り原理
鳩ノ巣原理

- 状態数が n で語の長さが n 以上だと…



反復補題 (復習)

- 正規言語 L に対して n が存在して、|z|≥n なる任意の z ∈ L について以下を満たすように z を uvw に分解できる
  1. |uv| ≤ n
  2. |v| ≥ 1
  3. 0 以上の任意の i について $uv^iw \in L$
- 条件を満たす分解が1つでもあればよい

反復補題は必要条件

- 反復補題は正規言語の必要条件
  - 反復補題が成り立たなければ正規言語ではない
- 反復補題を満たす正規言語でない言語も存在する

レポートその5

- L = { a^m | m = n^2, nは自然数 } が正規言語でないことを反復補題を用いて証明
- L が正規言語であると仮定して、反復補題が成り立たないことを示す
- 反復補題が成り立たないとは…
  - 適当な正整数 n に対して |z|≥n なる z を選び、z を z=uvw(ただし|uv|≤n、|v|≥1)となるように u, v, w に分解すると、どのように分解しても $uv^iw \notin L$ となる i が存在する

レポートその5

- L が正規言語であると仮定する
- 適当な正整数 n に対し a^m(ただしm=n^2)を選ぶ
- a^m=uvw に分解
  - ただし u=a^i、v=a^j、w=a^k
  - また i≥0、j≥1、k≥0、i+j≤n、i+j+k=n^2
- 反復補題より $uv^2w \in L$ のはずである
- 本当にそうか？

レポートその5

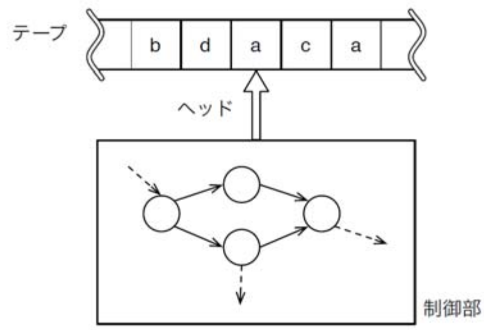
- $|uv^2w| = |a^ia^{2j}a^k| = i+2j+k$ である
- j≥1 および i+j+k=n^2 より $i+2j+k = n^2+j > n^2$
- i≥0 および i+j≤n より j≤n
- よって $i+2j+k = n^2+j \le n^2+n < n^2+2n+1 = (n+1)^2$
- つまり $n^2 < |uv^2w| < (n+1)^2$ なので $uv^2w \notin L$
- これは $uv^2w \in L$ に反する
- よって L は正規言語ではない

チューリング機械

チューリング機械

- Alan Turing, 1930's
- 計算を機械的動作としてとらえたモデル

概念図



有限オートマトンとの違い

- 制限なし
  - テープへの書き込みも可能
  - 1マス読んだらヘッドを
    - 右へ1マス移動
    - 左へ1マス移動
    - 移動しない

チューリング機械の動作

1. ヘッドの位置のマスの記号を読む
2. 読んだ記号に従って状態を遷移する
   - 終了状態へ到達したら終了
3. ヘッドの位置のマスに記号を書く
4. ヘッドの位置を
   a. 1マス右へ移動する
   b. 1マス左へ移動する
   c. 移動しない
5. 1. へ戻る

チューリング機械の定義

$M = (Q, \delta, \Sigma, \Gamma)$

Q : 状態の有限集合 $(\neq \emptyset)$

　$q_0 \in Q$: 初期状態

　$q_{fin} \in Q$: 終了状態

δ : 遷移関数

　$(Q - \{q_{fin}\} \times \Gamma) \to Q \times \Gamma \times \{L, R, N\}$

Σ : 入出力記号の有限集合

Γ : テープ記号の有限集合

テープ記号 Γ

- テープのマスに書くことができる記号のすべて
- 空白のマスを表す記号 B を含む

入出力記号 Σ

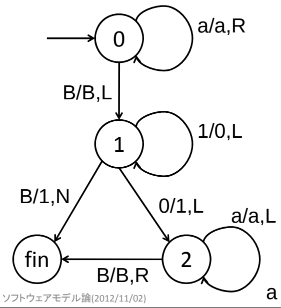
- チューリング機械の入出力に利用できる記号のすべて
- Γ の部分集合
- B は入出力に使えない
- 以下では Σ = { 0, 1 } とする

状態遷移関数 δ

- $\delta : (Q\text{-}\{q_{fin}\}) \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, N\}$
- $\delta(q, a) = (q', b, L)$
  - 状態 q で記号 a を読んだ場合、
    1. 状態 q' に遷移し
    2. 記号 b を書込み（a を上書きし）
    3. ヘッドを左へ1マス移動する
  - R ならば右へ1マス移動
  - N ならば移動しない
- 終了状態の遷移先はない

例：関数 inc(x) = x + 1 を計算するTM

M_{inc} = ({ 0, fin, 1, 2}, δ, { 0, 1 }, { 0, 1, B })



a = 0 or 1

状態	記号	遷移関数値
0	0	(0, 0, R)
0	1	(0, 1, R)
0	B	(1, B, L)
1	0	(2, 1, L)
1	1	(1, 0, L)
1	B	(fin, 1, N)
2	0	(2, 0, L)
2	1	(2, 1, L)
2	B	(fin, B, R)

チューリング機械の計算状況

- 計算中のチューリング機械の様子は、制御部の状態、テープの内容、ヘッドの位置で決まる

(q, ω, ω')

- q : 制御部の状態
- ω : ヘッドより左側のテープの内容
- ω' : ヘッドから右側のテープの内容
  （ヘッド位置含む）

初期状況

(q_0, ...BB, xBB...)

- x ∈ Σ* が入力
- 例えば、(q_0, ...BB, 111000BB...)
  - (q_0, B, 111000B) あるいは
  - (q_0, ε, 111000) とも書く

終了状況

(q_{fin}, ω, ω')

- 終了状態に到達
- 正常終了状況
  - (q_{fin}, ...BB, yBB...)
  - y が出力

チューリング機械の計算動作

- 計算状況を遷移関数に従って変えること
- q ∈ Q, u,v ∈ Σ*, a,b ∈ Σ とすると

$$(q, ub, av) \vdash \begin{cases} (q', u, ba'v) & \text{if } \delta(q,a) = (q', a', L) \\ (q', uba', v) & \text{if } \delta(q,a) = (q', a', R) \\ (q', ub, a'v) & \text{if } \delta(q,a) = (q', a', N) \end{cases}$$

- ⊢ が1回の計算動作を表す

例：M_{inc} の計算動作

- 入力が101の場合

(0, B, 101B) ⊢ (0, B1, 01B) ⊢ (0, B10, 1B)

⊢ (0, B101, B) ⊢ (1, B10, 1B) ⊢ (1, B1, 00B)

⊢ (2, B, 110B) ⊢ (2, B, B110B) ⊢ (fin, B, 110B)

チューリング機械の計算

- 入力 x に対するチューリング機械の計算
  - x に対する初期状況から計算動作を繰り返す過程の総称
- 計算列
  - 計算動作に伴って変化する計算状況の列
- 計算の正常終了
  - 正常終了状況に到達

関数を計算するチューリング機械

- 関数 f を計算するチューリング機械 M
  - f は Σ* 上の1変数（1引数）関数
- x ∈ dom(f) ならば M に x を入力して実行すると f(x) を出力して正常終了
- x ∉ dom(f) ならば M に x を入力して実行すると正常終了しない
- 結果を M(x) と書く（出力以外に終了状況を含む）

例：inc(x) 再び

- 正確には

$$inc(x) = \begin{cases} n+1 \text{ の 2 進数表記} & \text{if } x \text{ がある } n \in \mathbb{N} \text{ の 2 進数表記} \\ \text{未定義} & \text{otherwise} \end{cases}$$

- x が正しい2進数でなければ未定義
  - 正しくない（定義域に含まれない）入力の場合は正常終了しない