ソフトウェアモデル論（2010年度）
第8回・2010/11/15

桑原 寛明
情報理工学部 情報システム学科

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 $i+2j+k = n^2+j \leq n^2+n < n^2+2n+1 = (n+1)^2$
- 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

データのコード化

- 入出力記号の集合 Σ を { 0, 1 } に制限する
- 制限しても困ることはない
  - 各記号を2進数やASCIIで表現すればよい
- 例えば
  - $5_{10} \rightarrow 101_2$
  - A → 01000001
  - (2,AC) → 00101000 00110010 00101100 …
  - 隣接行列グラフを表現

チューリング機械のコード化

- チューリング機械 M = (Q, δ, Σ, Γ)
  - Q, Σ, Γ は有限集合
  - δ は有限集合から有限集合への関数
- 有限 ⇒ 列挙可能 ⇒ コード化可能
- (Q, δ, Σ, Γ) をどのようにコード化するか

Q, Σ, Γ のコード化

- 状態は 0, 1, 2, ..., n と名前が付いており n を終了状態とすれば Q は状態数で決定できる
- Σ は { 0, 1 }
- 記号を 0, 1, 2, ..., m（ただし 2 を B とみなす）とすれば Γ は記号数で決定できる

δ のコード化

- 遷移関数は有限集合から有限集合への関数
  - 状態 × 記号 → 状態 × 記号 × 移動方向
  - 有限の対応付け
- δ の引数の状態番号の小さい順、記号番号の小さい順に δ の値を並べる

δ のコード化の例

- M_{inc} の遷移関数

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)

順に並べる

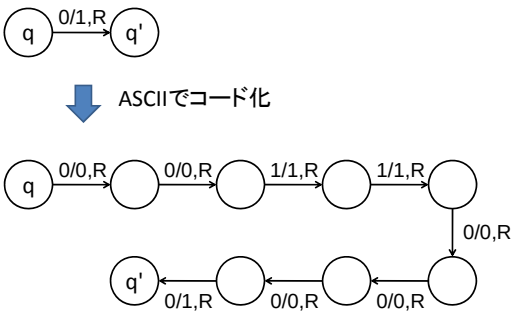
(0,0,R), (0,1,R), (1,B,L), (2,1,L), (1,0,L), ...

例: M_{inc} のコード化

- 状態数 4
  - { 0, 1, 2, fin }
- 記号数 3
  - { 0, 1, B }
- 遷移関数
  - (0,0,R), (0,1,R), (1,B,L), (2,1,L), ..., (fin,B,R)
- よって

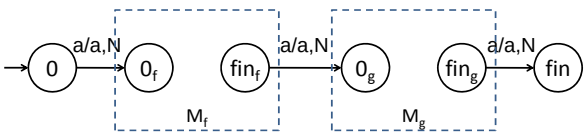
  (4, 3, (0,0,R), (0,1,R), (1,2,L), ..., (3,2,R))

コード化に伴うTMの変形例



合成関数の計算

- 合成関数 g∘f を計算するTM
- 関数 f を計算するTM M_f と g を計算するTM M_g から生成
- 直観的には、M_fの終了状態をM_gの初期状態とみなせばよい



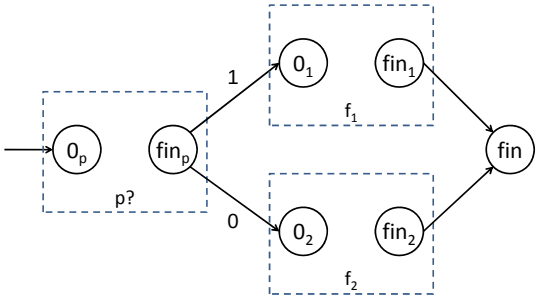
分岐

- 述語 p? と関数 f_1, f_2 に対して以下を計算

$$f(x) = \begin{cases} f_1(x) & \text{if } p?(x) = 1 \\ f_2(x) & \text{if } p?(x) = 0 \end{cases}$$

- p? を計算した後、その結果を見て f_1 を計算する TM と f_2 を計算する TM のいずれの初期状態に遷移するか決める

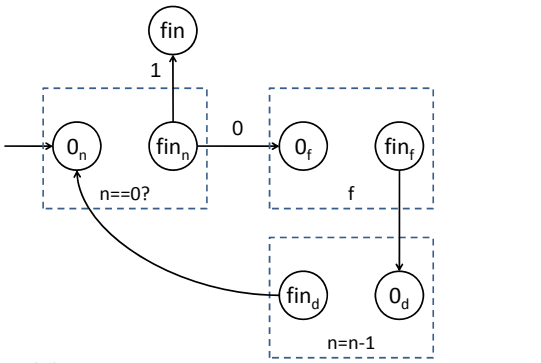
分岐



繰返し

- 関数 f に対して g(x,n) = $f^{(n)}(x)$ を計算する
  - $f^{(n)}$ は f を n 回合成した関数
- n==0 による条件分岐と f の計算、n=n-1 の計算を組み合わせればよい

繰返し



関数を計算したい

- 問題を解く ＝ 関数を計算する
- 問題が関数化されているとする
- 関心があるのは、関数が計算できるかどうか

チューリング機械計算可能

- 関数 f は Σ^* 上の（部分）関数
- f がチューリング機械計算可能であるとは、f を計算するチューリング機械が存在すること
  - f を計算するTMは一通りではない

<u>計算可能性をチューリング機械を用いて定義</u>

その他の計算可能性

- 帰納的関数
- λ計算

これらとチューリング機械の 3 つは等価

チャーチの提唱

あるいはチャーチ＝チューリングの定立

- チューリング機械で記述できる以上の計算能力を持つ計算原理は存在しない。従って、チューリング機械計算可能性を計算可能性の基準として用いるのは妥当である。

万能チューリング機械

- 以下の関数 comp を計算するチューリング機械

$$comp(p,x) = \begin{cases} M(x) & \text{if } p \text{ があるチューリング機械 } M \text{ のコード} \\ 0 & \text{otherwise} \end{cases}$$

- チューリング機械の実装としては万能チューリング機械だけあればよい
  - その他のチューリング機械はコードさえあれば万能チューリング機械で実行できる

万能チューリング機械

- comp(p, x) を計算する
- 1ステップごとに p を調べてシミュレートすればよい

↓TMのコード	↓入力	↓作業領域
(4, 3, (0,0,R), ..., (3,2,R))	1101 ... 01	（状態を記録）

関数 comp を拡張

- 関数 comp の値が未定義になる場合がある
  - comp(p,x) において p が表すチューリング機械 M が計算する関数の定義域に x が含まれない場合
- 未定義にならないように修正
  - 関数 t-comp

$$t\text{-}comp(p,x) = \begin{cases} M(x) & \text{if } p \text{ があるチューリング機械 } M \text{ のコード} \\ & \quad \text{かつ } M(x) \text{ が正常終了} \\ 0 & \text{otherwise} \end{cases}$$

t-comp はチューリング機械計算不可能

- t-comp がチューリング機械計算可能であると仮定すると、矛盾が導かれる
- 詳細は定理 2.7（69ページ）

チューリング機械の停止性

- チューリング機械が関数を計算するとは
  - チューリング機械が停止し、かつ
  - チューリング機械が計算結果を出力する
- チューリング機械 M が関数 f を計算するか判定するには
  - M が停止するか判定し
  - 計算結果が正しいか判定する

チューリング機械の停止性判定

- チューリング機械が停止するか判定

$$halt?(p,x) = \begin{cases} 1 & \text{if } p \text{ があるチューリング機械 } M \text{ のコード} \\ & \quad \text{かつ } M(x) \text{ が正常終了} \\ 0 & \text{otherwise} \end{cases}$$

- halt? は計算不可能
  - halt? を（任意の p と任意の x に対して）計算できるチューリング機械は存在しない

TMの停止性判定の計算不可能性

- halt? を計算する TM H が存在すると仮定する
- halt?(x,x)=1 ならば停止せず、halt?(x,x)=0 ならば 0 を出力し停止する関数 f(x) を考える
- f を計算する TM F を H を使って作ることが可能
- では F([F]) は停止するだろうか？
  - [F] はTM F のコードを表す

TMの停止性判定の計算不可能性

- F([F]) は停止する？
  - 停止するということは f の定義から halt?([F],[F])=0
  - halt?([F],[F])=0 ならば F([F]) は停止しないはず
  ⇒ 矛盾
- F([F]) は停止しない？
  - 停止しないということは f の定義から halt?([F],[F])=1
  - halt?([F],[F])=1 ならば F([F]) は停止するはず
  ⇒ 矛盾

⇒ halt? を計算する TM H は存在しない

重要なことは

- 計算不可能な関数が存在する
  - その関数を計算するチューリング機械が存在しない
  - 計算するアルゴリズムが存在しないと言い換えてもよい
  - 停止性判定、文脈自由文法の曖昧性判定、など
- チューリング計算機は可算無限個、関数は非可算個
  - 計算不可能な関数はたくさんある