

ソフトウェアモデル論(2013年度)
第8回・2013/11/14

桑原 寛明
情報理工学部 情報システム学科

データのコード化

(復習)

- 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
 - $5_{10} \rightarrow 101_2$
 - $A \rightarrow 01000001$
 - $(2, AC) \rightarrow 00101000 00110010 00101100 \dots$
 - 隣接行列でグラフを表現

ソフトウェアモデル論(2013/11/14)

2

チューリング機械のコード化

(復習)

- チューリング機械 $M = (Q, \delta, \Sigma, \Gamma)$
 - 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 \Rightarrow 列挙可能 \Rightarrow コード化可能
- $(Q, \delta, \Sigma, \Gamma)$ をどのようにコード化するか

ソフトウェアモデル論(2013/11/14)

3

Q, Σ, Γ のコード化

(復習)

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

ソフトウェアモデル論(2013/11/14)

4

δ のコード化

(復習)

- 遷移関数は有限集合から有限集合への関数
 - 状態 \times 記号 \rightarrow 状態 \times 記号 \times 移動方向
 - 対応付けは有限通り
- 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

ソフトウェアモデル論(2013/11/14)

5

δ のコード化の例

(復習)

- M_{inc} の遷移関数

状態	記号	遷移関数值
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), ...

ソフトウェアモデル論(2013/11/14)

6

例: M_{inc} のコード化 (復習)

- 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))

ソフトウェアモデル論(2013/11/14) 7

コード化に伴うTMの変形例 (復習)

ソフトウェアモデル論(2013/11/14) 8

合成関数の計算 (復習)

- 合成関数 $g \circ f$ 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

ソフトウェアモデル論(2013/11/14) 9

分岐 (復習)

- 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

ソフトウェアモデル論(2013/11/14) 10

分岐

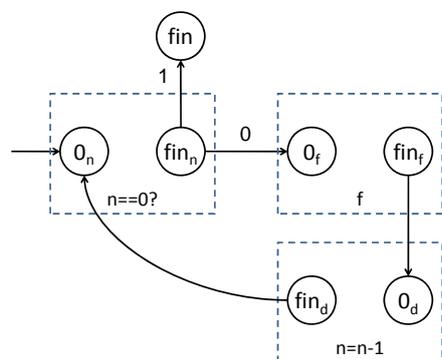
ソフトウェアモデル論(2013/11/14) 11

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

ソフトウェアモデル論(2013/11/14) 12

繰り返し



ソフトウェアモデル論(2013/11/14)

13

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

ソフトウェアモデル論(2013/11/14)

14

チューリング機械計算可能

- 世の中の問題は関数 f として表されるとする
 - 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

計算可能性をチューリング機械を用いて定義

ソフトウェアモデル論(2013/11/14)

15

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

ソフトウェアモデル論(2013/11/14)

16

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。

ソフトウェアモデル論(2013/11/14)

17

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

ソフトウェアモデル論(2013/11/14)

18

万能チューリング機械

- $\text{comp}(p, x)$ 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)

ソフトウェアモデル論(2013/11/14)

19

関数 comp を拡張

- 関数 comp の値が未定義になる場合がある
 - $\text{comp}(p,x)$ 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 $t\text{-comp}$

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

ソフトウェアモデル論(2013/11/14)

20

$t\text{-comp}$ はチューリング機械計算不可能

- $t\text{-comp}$ 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)

ソフトウェアモデル論(2013/11/14)

21

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

ソフトウェアモデル論(2013/11/14)

22

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

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

- 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

ソフトウェアモデル論(2013/11/14)

23

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
- $\text{halt?}(x,x)=1$ ならば停止せず、 $\text{halt?}(x,x)=0$ 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

ソフトウェアモデル論(2013/11/14)

24

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 $\text{halt}?([F],[F])=0$
 - $\text{halt}?([F],[F])=0$ 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 $\text{halt}?([F],[F])=1$
 - $\text{halt}?([F],[F])=1$ 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

ソフトウェアモデル論(2013/11/14)

25

重要なことは

- 計算可能性の定義
 - チャーチの提唱
- 万能チューリング機械が存在する
- 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
 - 停止性判定、文脈自由文法の曖昧性判定、など
 - 計算不可能な関数はたくさんある
 - チューリング機械は可算無限個、関数は非可算個

ソフトウェアモデル論(2013/11/14)

26