

## ソフトウェアモデル論(2010年度) 第9回・2010/11/22

桑原 寛明  
情報理工学部 情報システム学科

### チューリング機械計算可能

- 関数  $f$  は  $\Sigma^*$  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TMI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(2010/11/22)

2

### 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(2010/11/22)

3

### 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(2010/11/22)

4

### 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(2010/11/22)

5

### 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

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

- $halt?$  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(2010/11/22)

6

## 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
    - チューリング計算機は可算無限個、関数は非可算個

ソフトウェアモデル論(2010/11/22)

7

## 命題論理

ソフトウェアモデル論(2010/11/22)

8

## 命題

- 内容の真偽が確定できる文
  - 加算はチューリング機械計算可能である
  - 1と10は等しい
  - 情報理工学部の学生数は2233人である
- 命題同士が関連することもある
  - 風が吹く
  - 風が吹くと桶屋が儲かる
  - ⇒ 桶屋が儲かる

ソフトウェアモデル論(2010/11/22)

9

## 論理学

- 命題の集まりについて、ある命題の真偽が他の命題の真偽にどのように影響するか、命題間の関連を系統的に調べる学問
- 数理論理学
  - 数学における形式手法、記号的手法を用いて行う論理学

ソフトウェアモデル論(2010/11/22)

10

## 論理学

- 命題論理
  - 命題間の関係
- 述語論理
  - 対象と述語
- 様相論理
  - 可能性、必然性
- 時相論理
  - 時系列

ソフトウェアモデル論(2010/11/22)

11

## 計算機科学における数理論理学の応用

- 論理回路、順序回路
- (論理)プログラミング
- モデル検査
- 定理自動証明  
など

ソフトウェアモデル論(2010/11/22)

12

## 命題変数

- 命題を表す変数
  - 命題が述べる内容には興味がない
  - 興味があるのは命題の真偽と関係
- 特に、原子命題を表す記号
  - 原子命題: 最も基本的な命題

ソフトウェアモデル論(2010/11/22)

13

## 論理式

- 命題を表す記号列
1. 命題変数は論理式である
  2.  $P, Q$  が論理式であれば
    - $(\neg P)$  否定、~でない
    - $(P \wedge Q)$  連言、かつ
    - $(P \vee Q)$  選言、または
    - $(P \rightarrow Q)$  含意、ならば
- は論理式である
1. 1. と 2. から作られるものだけが論理式である

ソフトウェアモデル論(2010/11/22)

14

## 括弧の省略

- 論理演算子の優先度と結合の方向を決めて括弧を省略
- 優先度
  - $\neg, \wedge, \vee, \rightarrow$  の順
- 結合の方向
  - $\wedge, \vee$  は左結合
    - $p \wedge q \wedge r$  は  $((p \wedge q) \wedge r)$  の意味
  - $\rightarrow$  は右結合
    - $p \rightarrow q \rightarrow r$  は  $(p \rightarrow (q \rightarrow r))$  の意味

ソフトウェアモデル論(2010/11/22)

15

## 十分条件、必要条件、逆、裏、対偶

- $P = p \rightarrow q$
- $p$  は  $P$  の十分条件
- $q$  は  $P$  の必要条件
- $P$  の逆
  - $q \rightarrow p$
- $P$  の裏
  - $\neg p \rightarrow \neg q$
- $P$  の対偶
  - $\neg q \rightarrow \neg p$

ソフトウェアモデル論(2010/11/22)

16

## 意味論

- 論理式の意味とは論理式の真理値
  - 真 or 偽
- 以下の2つから決まる
  - 命題変数の真理値
  - 論理演算子 ( $\neg, \wedge, \vee, \rightarrow$ ) の意味

ソフトウェアモデル論(2010/11/22)

17

## 解釈

- 命題変数の真理値を定義する関数
  - true : 真, false : 偽
- 解釈を  $I$ 、命題変数の集合を  $\Sigma$  とすると
  - $I : \Sigma \rightarrow \{ \text{true}, \text{false} \}$
- すべての  $p \in \Sigma$  に対して  $I(p) = \text{true}$  または  $I(p) = \text{false}$

ソフトウェアモデル論(2010/11/22)

18

### 解釈の例

- $\Sigma = \{p, q, r\}$  とすると解釈は8通りあり得る

	p	q	r
$I_1$	true	true	true
$I_2$	true	true	false
$I_3$	true	false	true
$I_4$	true	false	false
$I_5$	false	true	true
$I_6$	false	true	false
$I_7$	false	false	true
$I_8$	false	false	false

ソフトウェアモデル論(2010/11/22) 19

### 論理演算子の意味

- 真理値関数によって定義
  - 以下の Not, And, Or, Imp がそれぞれ否定、連言、選言、含意の意味を定義

P	Not(P)	P	Q	And(P,Q)	Or(P,Q)	Imp(P,Q)
true	false	true	true	true	true	true
false	true	true	false	false	true	false
		false	true	false	true	true
		false	false	false	false	true

ソフトウェアモデル論(2010/11/22) 20

### 論理式の意味

- 命題変数の集合  $\Sigma$
- 解釈  $I$  のもとでの論理式  $P$  の真理値  $V_I(P)$ 
  - 解釈は命題変数の真理値を決める

$$V_I(P) = \begin{cases} I(p) & P \text{ が命題変数 } p \in \Sigma \text{ の場合} \\ \text{Not}(V_I(Q)) & P \text{ が } \neg Q \text{ の場合} \\ \text{And}(V_I(Q), V_I(R)) & P \text{ が } Q \wedge R \text{ の場合} \\ \text{Or}(V_I(Q), V_I(R)) & P \text{ が } Q \vee R \text{ の場合} \\ \text{Imp}(V_I(Q), V_I(R)) & P \text{ が } Q \rightarrow R \text{ の場合} \end{cases}$$

ソフトウェアモデル論(2010/11/22) 21

### モデル

- 論理式集合  $\Phi$
- 解釈  $I$
- $I$  が  $\Phi$  のモデルである
  - iff  $\Phi$  に含まれるすべての論理式  $P \in \Phi$  について  $I(P) = \text{true}$
- $\models \Phi$  あるいは  $I \models \Phi$  と書く

ソフトウェアモデル論(2010/11/22) 22

### 論理的帰結

- 論理式集合  $\Phi$
- 論理式  $P$
- $P$  は  $\Phi$  の論理的帰結である
  - iff 全ての解釈  $I$  について、 $I$  が  $\Phi$  のモデルならば  $I$  は  $P$  モデルでもある
- $\Phi \models P$  と書く

ソフトウェアモデル論(2010/11/22) 23

### トートロジー(恒真式)

- 論理式  $P$
- $P$  はトートロジーである
  - iff 全ての解釈  $I$  に対して  $I \models P$
- 任意の解釈のもとで真になる
- 例えば、 $p \vee \neg p$  はトートロジー

ソフトウェアモデル論(2010/11/22) 24

### 充足可能

- 論理式集合  $\Phi$  に対してモデルが存在するならば  $\Phi$  は充足可能である
- 論理式  $P$  に対してモデルが存在するならば  $P$  は充足可能である
- 充足可能でなければ充足不能と

ソフトウェアモデル論(2010/11/22)

25

### 恒偽式

- 充足不能な論理式
- どのような解釈のもとでも偽である
- 例えば、 $p \wedge \neg p$  は恒偽式

ソフトウェアモデル論(2010/11/22)

26

### 論理的帰結の判定

- 論理式  $P$  が論理式集合  $\Phi$  の論理的帰結であるか判定したい
- 手順？
  1. すべての解釈について  $\Phi$  のモデルであるか調べる
  2.  $\Phi$  のモデルであるすべての解釈のもとで  $P$  が真であるか調べる
- $\Phi$  と  $P$  に含まれる命題変数が  $n$  種類ならば  $2^n$  通りの解釈について調べなければならない  
⇒ 効率が悪い

ソフトウェアモデル論(2010/11/22)

27

### 証明系

- 論理式が論理式集合の論理的帰結であることを、論理式(の列)に対する機械的な操作のみによって調べる方法
  - 論理式の意味(真理値)を考えない
  - 判定アルゴリズムの一種とみなしてもよい
- 証明系は、論理式(の集合)から別の論理式を導出する推論規則の集合として定義される

ソフトウェアモデル論(2010/11/22)

28

### 証明系の種類

- 演繹系
  - 前提の論理式集合から結論の論理式が導出されるまで推論を繰り返す証明系
  - 例えば、LK や自然演繹など
- 反駁系
  - 前提の論理式集合に結論の否定を加えて推論し、否定的な結果が得られたら成功とする証明系
  - 例えば、分解証明系など

ソフトウェアモデル論(2010/11/22)

29