計算理論の初歩

fujidig

2017 年 9 月

計算理論は自然数の集合に対し計算可能という概念を定め，それについて考察する分野である．この資料ではプログラミング言語に触ったことのある人向けに計算理論の初歩を解説する．この資料の目標はゲーデルの不完全性定理を証明することである．

1 計算理論の基礎

計算可能性の定義の仕方は何通りもあるが，この資料では「C 言語のプログラムによって書ける関数」として定める．ただし，C 言語のフルの機能を想定すると難点があるので以下の機能は制限する:

(1) 整数型以外のデータ型
(2) 入出力
(3) グローバル変数
(4) 標準ライブラリ
(5) その他外部ライブラリ (並列処理や機種依存コマンドなど)

また， ハードウェアのエラーは起きないものとし，メモリは十分にあるものとする．つまり，いつどんな環境で実行しても常に実行結果は一定になるような理想的状況だけを考える．また，現実の C 言語は int 型は有限通りの値しかとることができないが，数学的な取扱いを簡単にするため，我々の C 言語では int 型に絶対値がどんなに大きい整数も扱えることとする．

定義 1. C 言語のプログラム P に対し，それが計算する関数 Φ_P とは,

$$\Phi_P(x) = \begin{cases} y & (P \text{ に } x \text{ を入力したとき } y \text{ が返る}) \\ \text{undefined} & (P \text{ に } x \text{ を入力したとき停止しない}) \end{cases}$$

で定義される関数とする．

定義 2. $f : \mathbb{N} \to \mathbb{N}$ が **partial computable (p.c.)** とは C 言語のプログラムによって計算できる関数であるときをいう．つまり $f = \Phi_P$ となるプログラム P が存在するときである．

x が f の定義域に入っているとき (停止するとき)，$f(x)\downarrow$ と書く．

$f : \mathbb{N} \to \mathbb{N}$ が **computable (計算可能)** とは全域かつ p.c. であるものをいう．全域とは定義域が $\mathbb{N}$ であることを言う．別の言葉でいうと f が計算可能であるのは必ず停止するプログラムで書けることである．

$A \subset \mathbb{N}$ が **computable (計算可能)** とは A の特徴関数 χ_A が計算可能であることを言う．

自然数から自然数への関数全体は連続体濃度を持つ (実数と同じ個数の要素が存在する) のに対し，計算可能な関数は可算無限個しかない．よって計算可能な関数は自然数上の関数のうちごくわずかでしかない．

例 3. 素数全体は計算可能である．なぜなら，次のプログラムによって素数全体の特徴関数を計算できるから．

```
int isprime(int n){
        for(int i = 2; i < n; i++){
                if(n%i == 0){
                        return 0;
                }
        }
        return 1;
}
```

定義 4. プログラム全体を適当な方法で一列に並べ，$e(\in \mathbb{N})$ 番目のプログラムが計算する関数を Φ_e と書く．この列には重複があってよい．

定義 5. 全単射 $\mathbb{N}^2 \to \mathbb{N}$ を一つ固定し，$\mathbb{N}^2 \ni (x, y) \mapsto \langle x, y\rangle \in \mathbb{N}$ とする．ただし $\mathrm{first}(\langle x, y\rangle) = x, \mathrm{second}(\langle x, y\rangle) = y$ なる関数 first, second が計算可能であるようにとる．

命題 6. 次のような $U \in \mathbb{N}$ が存在する．任意の $e, x \in \mathbb{N}$ に対し，

$$\Phi_U(\langle e, x\rangle) = \Phi_e(x).$$

Φ_U のことを universal machine という．任意のプログラム e をエミュレートできるからである．

この証明は実際にプログラム U を構成すればよい．U は「C 言語のプログラムを受け取り，それを実行する C 言語のプログラム」である．このようなプログラムは計算機界隈ではインタプリタと呼ばれるものである．

定義 7. $A \subset \mathbb{N}$ に対し，$A = \varnothing$ またはある f: 計算可能があって $A = \mathrm{range}(f)$ となるとき，A を **computably enumerable (c.e.)** という．

例 8. 平方数の全体は c.e. である．実際，次のプログラムがそれを示す．

```
int square_number(int n){
        return n * n;
}
```

命題 9. $A \subset \mathbb{N}$ に対し次は同値

(1) A は c.e.

(2) ある計算可能集合 R が存在し，

$$x \in A \iff \exists s \in \mathbb{N}, \langle x, s\rangle \in R$$

となる．

(3) ある p.c. 関数 g が存在し，$A = \mathrm{dom}\, g$.

証明. (1) $\Rightarrow$ (2): $A = \varnothing$ なら $R = \varnothing$ とすればよい. $A = \operatorname{range} f$ のとき

$$R := \{\langle x, s\rangle \mid f(s) = x\}$$

とおけばよい.

(2) $\Rightarrow$ (3): $x \in A \iff \exists s \in \mathbb{N}, \langle x, s\rangle \in R$, R: comp とする. このとき

```
int g(int x){
    int s = 0;
    while(true){
        if(⟨x, s⟩ ∈ R){
            return 0;
        }
        s++;
    }
}
```

とすると，このプログラムが計算する関数 g について，$\operatorname{dom} g = A$ となる. よって (3) が導けた.

(3) $\Rightarrow$ (1): $A = \operatorname{dom} \Phi_e$ とする. $A \neq \varnothing$ とする. すると $a \in A$ を一つとれる.

次のプログラムが計算する関数を f とする.

```
int f(int p){
    int s = first(p);
    int x = second(p);
    if(Φ_{e,s}(x) ↓){
        return x;
    } else{
        return a;
    }
}
```

このとき $\operatorname{range} f = \operatorname{dom} \Phi_e = A$ となる.

ここで $\Phi_{e,s}$ は次で定義される関数である.

$$\Phi_{e,s}(x) = \begin{cases} y & (e \text{ が指すプログラムに } x \text{ を入力したとき実行ステップ数 } s \text{ 以下で止まり出力は } y) \\ \text{undefined} & (e \text{ が指すプログラムに } x \text{ を入力したとき実行ステップ数が } s+1 \text{ 以上}) \end{cases}$$

$\Phi_{e,s}(x) \downarrow$ は計算可能になることに注意. □

命題 10. $A \subset \mathbb{N}$ について次は同値

(1) A は計算可能

(2) A と A^c の両方が c.e.

証明. (1) ⇒ (2): 次のプログラムが計算する関数を g とする.

```
int g(int x){
        if(x ∈ A){
                return 0;
        } else{
                while(true){}
        }
}
```

このとき $\mathrm{dom}\, g = A$ より，A は c.e. である．A^c が c.e. なことの証明も同様.

(2) ⇒ (1): $A = \varnothing, \mathbb{N}$ のときは明らか．$A \neq \varnothing, \mathbb{N}$ とする.

$A = \mathrm{range}\, f, A^c = \mathrm{range}\, g$ とする.

次のプログラムが計算する関数を h とする.

```
int h(int x){
        int i = 0;
        while(true){
                if(f(i) == x){
                        return 1;
                } else if(g(i) == x){
                        return 0;
                }
                i++;
        }
}
```

このとき h は全域関数であり，$h = \chi_A$ となる． □

定義 11.

$$K_0 := \{\langle e, x\rangle \mid \Phi_e(x)\downarrow\}$$
$$K := \{x \in \mathbb{N} \mid \Phi_x(x)\downarrow\}$$

命題 12. K, K_0 は c.e.

証明. $x \in K \iff \exists s \in \mathbb{N}, \Phi_{x,s}(x)\downarrow$ より K は c.e. である．K_0 も同様. □

定理 13. K, K_0 は計算可能でない.

証明. K^c は c.e. でないことを示す．$K^c = \mathrm{dom}(\Phi_e), e \in \mathbb{N}$ とする．すると，

$$e \in K^c \iff \Phi_e(e)\downarrow \iff e \in K$$

となって矛盾．よって K^c は c.e. でないので K は計算可能でない.

$$x \in K \iff \langle x, x\rangle \in K_0$$

であるので K は K_0 に“還元可能”である．よって K_0 が計算可能だとしたら K も計算可能になる．したがって K_0 は計算可能でない． □

この証明は対角線論法を使っている．

$e\backslash x$	0	1	2	3	4	…
0	○	○	×	×	○	…
1	○	×	○	○	○	…
2	×	×	×	×	×	…
3	○	○	○	○	○	…
4	○	×	×	○	×	…
⋮	⋮	⋮	⋮	⋮	⋮	⋱

上のように e と x に対して $\Phi_e(x)\downarrow$ かどうかを表にする．

一般に○×の表を作ったとき，表の中の行には現れない○×のリストが存在する．実際，表の対角線から○×の値を拾い，それを反転させた○×のリストを考えればよい．

ところが今考えている表においてはそのような○×のリストとは K^c に属しているかどうかをリストしたものに他ならない．よって $x \in K^c \iff \Phi_e(x)\downarrow$ となるような e は存在しない．すなわち K^c は c.e. ではない．

定義 14. $A, B \subset \mathbb{N}$ に対し，f: 計算可能があり，

$$x \in A \iff f(x) \in B$$

となるとき，$A \leq_m B$ であると定め，A は B に **(多対一) 還元可能**という．

命題 15. $A \leq_m B$ かつ B が計算可能ならば A も計算可能

証明. B の特徴関数を計算する関数を is_in_B とする．以下のプログラムで A の特徴関数を計算できる:

```
int is_in_A(int x){
        return is_in_B(f(x));
}
```

□

2 ペアノ算術

ペアノ算術とは自然数論を捕まえるための形式体系である．形式体系とはいわば数学のミニチュアである．証明という数学の概念を数学的に解析するため，形式化を行う．

一つの形式体系を定義するのは一つのプログラミング言語を定義することに似ている．プログラミング言語を定義するのにまずその文法を定義するように，ここでは形式体系の文法，すなわち項と論理式を定義する．

定義 16. 以下で定められる記号の列を **PA の項**という．

(1) $\bar{0}$ は PA の項である
(2) 変数記号 $x_0, x_1, x_2, \ldots$ は PA の項である
(3) t が PA の項ならば t' も PA の項である
(4) s, t が PA の項ならば $(s + t)$ も PA の項である
(5) s, t が PA の項ならば $(s \times t)$ も PA の項である

例 17.

(1) $\bar{0}''$ は PA の項である
(2) $(x_0 + (\bar{0}'' \times x_1))$ は PA の項である
(3) $\times(\bar{0})+$ は PA の項ではない
(4) (； ´ Д `) は PA の項ではない

定義 18. 以下で定められる記号の列を **PA の論理式**という.

(1) s, t が PA の項ならば $(s = t)$ は PA の論理式である
(2) φ が PA の論理式ならば $(\neg\varphi)$ は PA の論理式である
(3) φ, ψ が PA の論理式ならば $(\varphi \vee \psi)$ は PA の論理式である
(4) φ が PA の論理式で x が変数記号ならば $(\forall x)\varphi$ は PA の論理式である

(1) の形の論理式を**原子論理式**という.
また，以下の省略表現を用意する.

(1) $(\varphi \to \psi)$ と書いたら $((\neg\varphi) \vee \psi)$ を省略したものとする
(2) $(\varphi \wedge \psi)$ と書いたら $(\neg((\neg\varphi) \vee (\neg\psi)))$ を省略したものとする
(3) $(\exists x)\varphi$ と書いたら $(\neg(\forall x)(\neg\varphi))$ を省略したものとする

また，以下，どこに挿入されるべきか明らかな括弧は省略して書く.

例 19.

(1) $x_0 = \bar{0}$ は PA の論理式である
(2) $(\forall x_0)(\forall x_1)(x_0 + x_1 = x_1 + x_0)$ は PA の論理式である
(3) $(\exists x_0)(x_0 = x_0 + 1)$ は PA の論理式である

数学における証明とは公理から出発して論理的に正しい推論を積み上げていき目的の命題を得る行為だといえる．それを反映して，形式体系内での「証明」の概念を定義する.

定義 20. 以下の形をした論理式を**命題論理に関する公理**という

(1) $(\varphi \vee \varphi) \to \varphi$
(2) $\varphi \to (\varphi \vee \psi)$
(3) $(\varphi \vee \psi) \to (\psi \vee \varphi)$
(4) $(\varphi \to \psi) \to (\theta \vee \varphi \to \theta \vee \psi)$

定義 21. 以下の形をした論理式を**量化記号に関する公理**という

(1) $(\forall x)(\varphi \vee \psi) \to (\varphi \vee (\forall x)\psi)$
(2) $(\forall x)\varphi \to [t/x]\varphi$

$[t/x]\varphi$ とは φ の中に自由に現れる x をすべて項 t で置き換えて得られる論理式のことを指す.

なお，(1) では φ の中に x が自由に登場してはならない．また，(2) では項 t の中の変数は $[t/x]\varphi$ の中で量化記号で束縛されてはならない.

定義 22. 以下の形をした論理式を**等号に関する公理**という.

(1) $x = x$
(2) $x = y \to ([y/x_0]\varphi \to [x/x_0]\varphi)$
(3) $x = y \to ([x/x_0]t = [y/x_0]t)$

ただし φ は原子論理式とする.

定義 23. 以下の形をした論理式を**算術に関する公理**という.

(1) $\bar{0} \neq x_0'$
(2) $x_0' = x_1' \to x_0 = x_1$
(3) $x_0 + \bar{0} = x_0$
(4) $x_0 + x_1' = (x_0 + x_1)'$
(5) $x_0 \times \bar{0} = \bar{0}$
(6) $x_0 \times x_1' = x_0 \times x_1 + x_0$
(7) $[0/x]\varphi \to ((\forall x)(\varphi \to [x'/x]\varphi) \to (\forall x)\varphi)$

定義 24. 以上の命題論理に関する公理，量化記号に関する公理，等号に関する公理，算術に関する公理をまとめて単に**公理**という

定義 25. 以下を推論規則という

(1) (Modus Ponens) $\varphi \to \psi$ と φ から ψ を導くこと
(2) (一般化) φ から $(\forall x)\varphi$ を導くこと

定義 26. すべての公理を含み，推論規則で閉じた最小の集合を T_{PA} と書き，その元を PA で**証明可能**な論理式という.

別の言葉で言えば，次のような論理式の列 $\varphi_0, \ldots, \varphi_n$ が存在するような論理式 φ を証明可能という．すなわち，$\varphi = \varphi_n$ であり，任意の $0 \leq i \leq n$ に対して次のどれか少なくとも一つが成立する.

(1) φ_i は公理である
(2) i より小さい j, k が存在して φ_i は φ_j と φ_k から MP によって導ける
(3) i より小さい j が存在して φ_i は φ_j から一般化によって導ける

このような列を φ の**証明**という.

φ が PA で証明可能なとき $\mathrm{PA} \vdash \varphi$ と書く.

例 27. 任意の項 s, t, u について等号の推移律 $s = t \to (t = u \to s = u)$ は証明可能である.

以下が証明である.

(1) $x_1 = x_2 \to (x_2 = x_3 \to x_1 = x_3)$ (等号に関する公理 (2))
(2) $(\forall x_3)(x_1 = x_2 \to (x_2 = x_3 \to x_1 = x_3))$ ((1) に一般化を適用)
(3) $(\forall x_3)(x_1 = x_2 \to (x_2 = x_3 \to x_1 = x_3)) \to (x_1 = x_2 \to (x_2 = u \to x_1 = u))$ (量化子に関する公理 (2))
(4) $x_1 = x_2 \to (x_2 = u \to x_1 = u)$ ((2) と (3) に MP 適用)
(5) $(\forall x_2)(x_1 = x_2 \to (x_2 = u \to x_1 = u))$ ((4) に一般化)
(6) $(\forall x_2)(x_1 = x_2 \to (x_2 = u \to x_1 = u)) \to (x_1 = t \to (t = u \to x_1 = u))$ (量化子に関する公理 (2))
(7) $x_1 = t \to (t = u \to x_1 = u)$ ((5) と (6) に MP 適用)
(8) $(\forall x_1)(x_1 = t \to (t = u \to x_1 = u))$ ((7) に一般化適用)
(9) $(\forall x_1)(x_1 = t \to (t = u \to x_1 = u)) \quad \to (s = t \to (t = u \to s = u))$ (量化子に関する公理 (2))
(10) $s = t \to (t = u \to s = u)$ ((8) と (9) に MP 適用)

例 28. 同様に算術の公理の変数 x_0, x_1 に任意の項を代入した論理式は証明可能である.
同様に等号の公理 (3) の変数に任意の項を代入した $s = t \to [s/x_0]u = [t/x_0]u$ も証明可能である.

例 29. $1 + 1 = 2$ は PA で証明可能である.

(1) $0' + 0 = 0'$ (算術に関する公理 (3) と例 28)
(2) $(0' + 0 = 0') \to (0' + 0)' = 0''$ (代入に関する公理 (3) と例 28)
(3) $(0' + 0)' = 0''$ ((1) と (2) に MP)
(4) $0' + 0' = (0' + 0)'$ (算術に関する公理 (4) と例 28)
(5) $0' + 0' = (0' + 0)' \to ((0' + 0)' = 0'' \to 0' + 0' = 0'')$ (例 27)
(6) $0' + 0' = 0''$ ((4) と (5) に MP 適用)

定義 30. ある閉論理式 φ が存在して φ と $\neg\varphi$ の両方が PA で証明可能なとき，PA は**矛盾している**という.
そうでないとき PA は**無矛盾**であるという.
任意の閉論理式 φ に対して，φ と $\neg\varphi$ のどちらか一方が PA で証明可能なとき，PA は**完全**であるという.
ここで閉論理式とは，自由変数のない論理式である.

定理 31 (ゲーデルの第一不完全性定理)**.** PA は，無矛盾であるならば不完全である.

命題 32 (表現定理)**.** A を計算可能集合とする．このとき PA の論理式 $\varphi(x)$ が存在して

$$x \in A \Rightarrow \mathrm{PA} \vdash \varphi(\bar{x})$$
$$x \notin A \Rightarrow \mathrm{PA} \vdash \neg\varphi(\bar{x})$$

となる.

ここではしないが，表現定理を証明するには次のようにする:

- 「帰納的関数」と呼ばれる関数のクラスを用意してそれが計算可能関数のクラスと一致することを示す
- 表現定理を帰納的関数の構成に関する帰納法で示す

命題 33 (半表現定理)**.** PA を無矛盾とする．A を c.e. set とする．このとき PA の論理式 $\varphi(x)$ が存在して

$$x \in A \iff \mathrm{PA} \vdash (\exists s)\varphi(\bar{x}, s)$$

となる．

半表現定理は表現定理より従う．ω 無矛盾性というものを仮定すれば証明は自然にできるが，ただの無矛盾性の場合は証明はやっかいである．

定義 34. PA の記号の有限列に適当な方法で自然数 (コード，ゲーデル数) を割り当てる．ただし符号化した自然数から元の有限列の要素を得る関数が計算可能となるようにする．たとえば 2 進法を使ったり素因数分解を使う．

PA の記号列の列に対しても同様に自然数を割り当てる．

論理式 φ に対してそのゲーデル数を $\mathrm{gn}(\varphi)$ と書く．

命題 35. 自然数 n に対して

$$\mathrm{isfml}(n) \iff n \text{ がある論理式のゲーデル数}$$

と定義すると isfml は計算可能な述語である．

自然数 p, n に対して

$$\begin{aligned}\mathrm{proves}(p, n) \iff\ & p \text{ がある証明のゲーデル数で } n \text{ がある論理式のゲーデル数であり} \\ & p \text{ は } n \text{ が指す論理式の証明を指している}\end{aligned}$$

と定義すると proves は計算可能な述語である．

具体的にこれらを計算するプログラムを書くのが面倒だが難しくはない．

T_{PA} に属する各論理式のゲーデル数をとった集合も同じく T_{PA} と書くことにする．

定理 36. PA を無矛盾とする．T_{PA} は計算可能でない．

証明. K が c.e. であることと半表現定理より，$K \leq_m T_{\mathrm{PA}}$ である．よって命題 15 より T_{PA} は計算可能でない． □

第一不完全性定理の証明．もし PA が完全であれば，次のようにして PA で証明可能な論理式か判定するプログラムが書ける．それは T_{PA} は計算可能でないことに反する．

```
int isprovable(int x){
        if(! isfml(x)) return 0;
        int n = 0;
        while(true){
                if(proves(n, x)) return 1;
                if(proves(n, neg(x))) return 0;
                n++;
        }
}
```

参考文献

[1] 鹿島亮 (2008)『C 言語による計算の理論』サイエンス社

[2] S.Barry Cooper (2002) “ Computability Thoery ” Chapman and Hall/CRC

[3] 照井一成 (2015)『コンピュータは数学者になれるのか？』 青土社

[4] 河村彰星 (2017) 『はじめての計算可能性』
http://www.graco.c.u-tokyo.ac.jp/~kawamura/t/summer_H29/

[5] 坪井明人　(2012)『数理論理学の基礎・基本』 牧野書店