計算理論　第 7 回
再帰定理

岡本 吉央
okamotoy@uec.ac.jp

電気通信大学

2020 年 11 月 19 日

最終更新：2020 年 11 月 21 日　09:28

概要

この講義の主題

計算理論 (Theory of Computation)
- 計算可能性理論 (Computability Theory)
- 計算複雑性理論 (計算量理論) (Complexity Theory)

講義の進め方

- 前半：計算可能性理論 (担当：岡本)
- 後半：計算複雑性理論 (担当：垂井先生)

スケジュール 前半 (予定)

1. 計算とは何か？ (10/1)
2. 計算モデル (10/8)
3. チャーチ・チューリングの定立 (10/15)
- ⋆ 休み (体育祭) (10/22)
4. コード化 (10/29)
5. 計算可能性 (11/5)
6. 停止性問題 (11/12)
7. 再帰定理 (11/19)
8. 前半のまとめ (11/26)

注意：予定の変更もありうる

今日の話

今日の話

- 再帰定理：WHILE プログラムで再帰を実現する方法
- 再帰定理の応用

(復習) s-m-n 定理

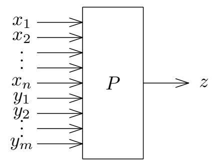
復習：s-m-n 定理

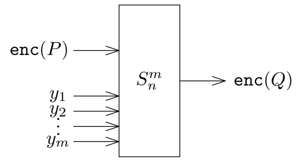
任意の自然数 m, n に対して，
次の性質を持つ $(m+1)$ 変数 WHILE 計算可能関数 $S^m_n : \mathbb{N}^{m+1} \to \mathbb{N}$ が存在

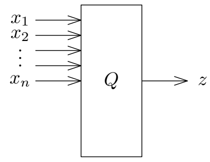
1. 任意の自然数 $e, y_1, \dots, y_m$ に対して
$S^m_n(e, y_1, \dots, y_m)$ は n 入力 GOTO プログラムのコードである
2. 任意の自然数 $e, x_1, \dots, x_n, y_1, \dots, y_m$ に対して
$\mathtt{univ}(S^m_n(e, y_1, \dots, y_m), \mathsf{enc}((x_1, \dots, x_n)))$
$= \mathtt{univ}(e, \mathsf{enc}((x_1, \dots, x_n, y_1, \dots, y_m)))$

(復習) s-m-n 定理：イメージ

$e = \mathsf{enc}(P)$ とする







目次

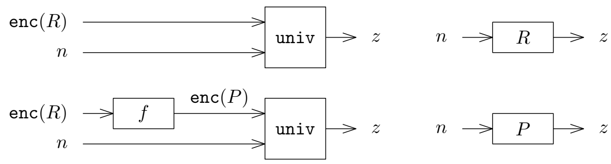
不動点定理

WHILE 計算可能全域関数 $f : \mathbb{N} \to \mathbb{N}$

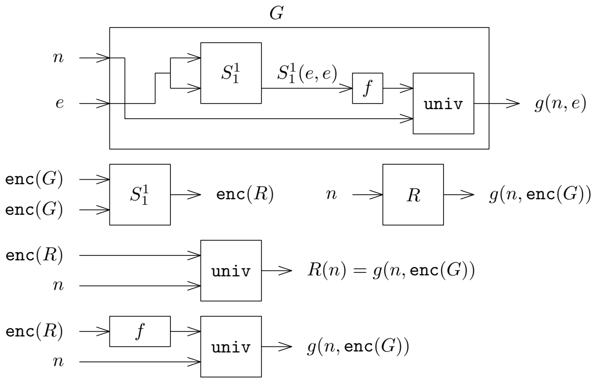
不動点定理

任意の f に対して，次を満たす WHILE プログラム R が存在する
- 任意の n に対して
$\mathtt{univ}(\mathsf{enc}(R), n) = \mathtt{univ}(f(\mathsf{enc}(R)), n)$

定理のイメージ：$f(\mathsf{enc}(R)) = \mathsf{enc}(P)$ とする



不動点定理：証明のイメージ



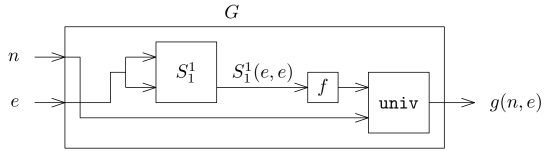
不動点定理：証明 (1)

証明：

- 次の部分関数 $g\colon \mathbb{N}^2 \nrightarrow \mathbb{N}$ を考える

$$g(n,e) = \mathtt{univ}(f(S_1^1(e,e)), n)$$

  この g は WHILE 計算可能
- g を計算する WHILE プログラムを G とする
- そして，$\mathsf{enc}(R) = S_1^1(\mathsf{enc}(G), \mathsf{enc}(G))$ を満たす R を考える



不動点定理：証明 (2)

このとき，任意の $n \in \mathbb{N}$ に対して

$$\begin{aligned}
\mathtt{univ}(\mathsf{enc}(R), n) &= \mathtt{univ}(S_1^1(\mathsf{enc}(G), \mathsf{enc}(G)), n) && (\mathsf{enc}(R) \text{ の定義}) \\
&= \mathtt{univ}(\mathsf{enc}(G), \mathsf{enc}((n, \mathsf{enc}(G)))) && (\text{s-m-n 定理}) \\
&= g(n, \mathsf{enc}(G)) && (G \text{ の定義}) \\
&= \mathtt{univ}(f(S_1^1(\mathsf{enc}(G), \mathsf{enc}(G))), n) && (g \text{ の定義}) \\
&= \mathtt{univ}(f(\mathsf{enc}(R)), n) && (\mathsf{enc}(R) \text{ の定義})
\end{aligned}$$

□

目次

再帰の例 (1)

次の関数 $\mathtt{fctl}\colon \mathbb{N} \to \mathbb{N}$ の計算を考える

$$\mathtt{fctl}(n) = \begin{cases} 1 & (n = 0 \text{ のとき}) \\ n \cdot \mathtt{fctl}(n-1) & (n \geq 1 \text{ のとき}) \end{cases}$$

問題点

- fctl の定義に再帰が使われている
- WHILE プログラムで再帰を行なえない (行う方法が分からない)

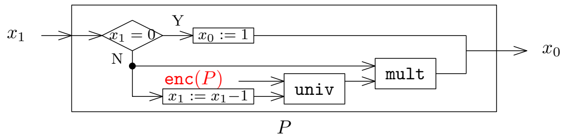
問題点の解決

⇝ WHILE プログラムでも再帰を行なえる (再帰定理)

再帰の例 (2)

- 仮に再帰ができるとして，次の 1 入力プログラム P を考える

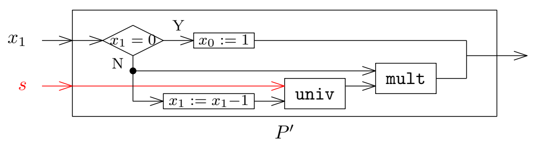
```
IF x1 = 0 THEN x0 := 1
  ELSE x0 := mult(x1, univ(enc(P), enc((x1 − 1))))
END
```



再帰の例 (3)

- 実際は，プログラム P の中で $\mathsf{enc}(P)$ は呼び出せないので，$\mathsf{enc}(P)$ を別の自然数 s にしたプログラム P' を考える (P' への入力は x_1, s)

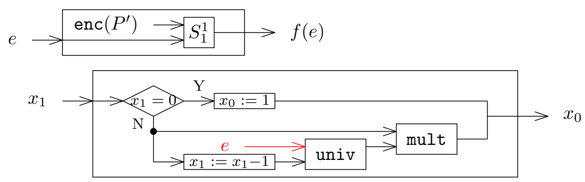
```
IF x1 = 0 THEN x0 := 1
  ELSE x0 := mult(x1, univ(s, enc((x1 − 1))))
END
```



再帰の例 (4)

- このとき，次の 1 変数関数 $f\colon \mathbb{N} \to \mathbb{N}$ を考える

$$f(e) = S_1^1(\mathsf{enc}(P'), e)$$



- f に不動点定理を適用して，プログラム R のコード $\mathsf{enc}(R)$ を得る

$$\mathtt{univ}(\mathsf{enc}(R), n) = \mathtt{univ}(f(\mathsf{enc}(R)), n)$$

今から観察すること

R が fctl を計算する WHILE プログラムになる

再帰の例 (5)

$n \geq 1$ のとき

$$\begin{aligned}
\color{red}{\texttt{univ}(\mathsf{enc}(R), n)} &= \texttt{univ}(f(\mathsf{enc}(R)), n) && (\text{不動点定理}) \\
&= \texttt{univ}(S_1^1(\mathsf{enc}(P'), \mathsf{enc}(R)), n) && (f \text{ の定義}) \\
&= \texttt{univ}(\mathsf{enc}(P'), \mathsf{enc}((n, \mathsf{enc}(R)))) && (\text{s-m-n 定理}) \\
&= \texttt{mult}(n, \color{red}{\texttt{univ}(\mathsf{enc}(R), n-1)}) && (P' \text{ の定義})
\end{aligned}$$

確かに，再帰ができている

再帰定理

再帰定理

任意の WHILE プログラム P に対して，
「P の中で，P のコード $\mathsf{enc}(P)$ を使う」という糖衣構文を作れる

証明：P の中で $\mathsf{enc}(P)$ を使うとき，それを次の手順で書き換える

1. P の中に現れる $\mathsf{enc}(P)$ を他の記号 s で置き換え，s を P の入力に含める (書き換えたプログラムを P' とする)
2. WHILE 計算可能な 1 変数関数 $f\colon \mathbb{N} \to \mathbb{N}$ を次のように定義する

$$f(e) = S_1^1(\mathsf{enc}(P'), e)$$

3. f に不動点定理を適用し，プログラム R のコード $\mathsf{enc}(R)$ を得る

このとき，R が P と同じ部分関数を計算する (次のページ)

再帰定理 (続き)

実際，

$$\begin{aligned}
\texttt{univ}(\mathsf{enc}(R), n) &= \texttt{univ}(f(\mathsf{enc}(R)), n) && (\text{不動点定理}) \\
&= \texttt{univ}(S_1^1(\mathsf{enc}(P'), \mathsf{enc}(R)), n) && (f \text{ の定義}) \\
&= \texttt{univ}(\mathsf{enc}(P'), \mathsf{enc}((n, \mathsf{enc}(R)))) && (\text{s-m-n 定理})
\end{aligned}$$

一方で，P から P' を作った方法をみると

$$\texttt{univ}(\mathsf{enc}(P), n) = \texttt{univ}(\mathsf{enc}(P'), \mathsf{enc}((n, \mathsf{enc}(P))))$$

つまり，R と P は同じ関数を計算する □

目次

再帰定理の応用：停止性関数の計算不可能性

再帰定理を使って，停止性関数の計算不可能性を証明できる

定義：停止性関数 (復習)

停止性関数 (停止関数) とは $\texttt{isHalting}\colon \mathbb{N}^2 \nrightarrow \mathbb{N}$ で，

$$\texttt{isHalting}(x_1, x_2) = \begin{cases} 1 & (x_1, x_2 \text{ が } \boxed{\text{条件 }*} \text{ を満たすとき}) \\ 0 & (\text{そうではないとき}) \end{cases}$$

$\boxed{\text{条件 }*}$

- ある自然数 k に対して
  x_1 が k 入力 GOTO プログラム P のコードであり，
  x_2 が長さ k のリスト $a = (a_1, \ldots, a_k)$ のコードであり，
  P に a を入力したときに，P が停止する

再帰定理の応用：停止性関数の計算不可能性 (証明)

第 6 回で証明した定理

停止性関数 isHalting は WHILE 計算不可能である

別証明：isHalting を計算する WHILE プログラムがあると仮定

- このとき，再帰定理より，次のような 1 入力プログラム Q が作れる

```
x2 := isHalting(enc(Q), x1);
IF x2 = 0 THEN x0 := 0 ELSE infloop(x1) END
```

- このとき，

$$\begin{aligned}
Q(x_1) \text{ が停止する} &\Rightarrow \texttt{isHalting}(\mathsf{enc}(Q), x_1) = 0 \\
&\Rightarrow Q(x_1) \text{ が停止しない} \\
Q(x_1) \text{ が停止しない} &\Rightarrow \texttt{isHalting}(\mathsf{enc}(Q), x_1) = 1 \\
&\Rightarrow Q(x_1) \text{ が停止する}
\end{aligned}$$

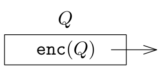
- つまり，どちらの場合でも矛盾 □

クワイン

再帰定理と不動点定理を使うことで，クワインを作れる

クワインとは？

自分自身 (のコード) を出力するプログラム



万能な計算モデルでは，必ずクワインを作れる

目次

今日のまとめ と 次回の予告

今日の話

- 再帰定理：WHILE プログラムで再帰を実現する方法
- 再帰定理の応用

次回は，前半のまとめ と それを踏まえた雑多な話題