2018 年度後期：計算基礎論 (18C5042) 2018 年 10 月 12 日

第 2 回：計算の理論への導入

担当：長尾篤樹

概要: この講義では計算の理論を C 言語の一部を用いたプログラム（教科書上では「関数プログラム」と呼んでおり，本講義でもそれに倣う）を用いている．関数プログラムを用いて計算の理論を議論するために必要な準備を今回の授業で行う．

1 一番大切な定義

定義 1.1 関数が計算可能であるとは，その関数を計算する関数プログラムが存在することである．

例題 1.2 自然数上の 2 変数関数

$$\text{least\_commom\_multiple}(x, y) = z, \qquad \text{ここで } z \text{ は } x \text{ と } y \text{ の最小公倍数}$$

が計算可能であることを示せ

例題 1.3 自然数上の二つの関数 f, g が以下の条件をみたしているとする．

$$f(x) = \begin{cases} 0 & (g(x) \text{ が偶数のとき}) \\ 1 & (g(x) \text{ が奇数のとき}) \end{cases}$$

このとき，g が計算可能であるならが f が計算可能であることを示せ

上記二つの例題はまだ解く事ができない．なぜなら「計算する」「関数プログラム」の定義がされていないからである．何かしらを「示す」「証明する」時には，定義から出発することがほとんどである.[1]「関数プログラム」の定義をどのようにすればよいかを考えるためには，上記二つの例題に対する C 言語プログラムを考えることが重要となる．

1.3 が正しいとするとき，その**対偶 (contradiction)** も正しい．では**逆 (conversion)** や**裏 (Inversion)** はどうだろうか．

2 自然数，表記の約束

前回授業を参照のこと

[1]定義以外にもすでに正しいと示された定理や命題，簡単に確認できる事実等を用いることもある．

3　部分関数

関数は定義域が何かしらの集合で示されればよく，$\mathbb{N}$ 全体等のよく知られている集合と決めつける必要はない．例えば，以下の例 3.1 に示す関数は $\mathbb{N} \times \mathbb{N} \mapsto \mathbb{N}$ であるが，定義域は $\{(x, y) \in \mathbb{N}^2 \mid y \neq 0\}$ である．

例 3.1

$$\text{partial\_func}(x, y) = (x + 1) \div y$$

このような明示的に出力をもたない入力が示された定義域を持つ k 変数関数を，を k 変数部分関数と呼ぶ．k 変数部分関数は以下のようにも表記可能である．

例 3.2

$$\text{half}(x) = \begin{cases} \frac{x}{2} & (x \text{ が偶数のとき}) \\ \text{定義されない} & (x \text{ が奇数のとき}) \end{cases}$$

また，部分関数を表現する際に**関数値対応表**を用いる事がある（教科書 7 ページ上部を参照のこと）．

k 変数部分関数 f に対し，$\vec{n} \in \mathbb{N}^k$ が定義域に入っているということを

$$f(\vec{n}) \downarrow$$

と書き，定義域に入っていないということを

$$f(\vec{n}) \uparrow$$

と書く．これを用いて，k 変数関数 f, g と $\vec{n}, \vec{m} \in \mathbb{N}^k$ に対し，$f(\vec{n}) = g(\vec{m})$ とは，以下の $1., 2.$ どちらかが成り立っているものとする．[2]

1. $(f(\vec{n}) \downarrow \wedge g(\vec{m}) \downarrow) \wedge (f(\vec{n}) \text{ の値} = g(\vec{m}) \text{ の値})$
2. $(f(\vec{n}) \uparrow \wedge g(\vec{m}) \uparrow)$

4　関数プログラム

今回の授業の主題である．関数プログラムは，C 言語プログラムのうち以下の文法や演算子を持つプログラムである．

- 変数の宣言（int 型のみで十分）

[2] 同時には成り立たない．なぜか？

- 代入文 （＝演算子の利用． インクリメントデクリメントも可）
- if 文 （else 節も利用可能）
- while 文（無限ループも可能．ループ中の命令文は空でもよい）
- return 文
- 以下の整数演算子（$+, -, *, /, \%$）（/ は右の値で左の値を割った商を返し，% はその余りを返す）
- 比較演算子（$==, !=, <.>, <=.>=$）
- 論理演算子（$\&\&, ||, !$）

また，if 文と while 文とを組み合わせることで表現可能な loop 文も別に定義する．ループ用の変数宣言を外で行う for 文と同じと思ってよい．

現実の C 言語と異なる点として以下の三つがある．まず，演算子の短絡評価は今回は考えない．すなわち，$0 * f(x)$ とあっても $f(x)$ の計算は必ず行い，$0! = 0 \& \& g(y, z)$ とあっても $g(y, z)$ の計算は必ず行う．さらに，メモリやビット長の制限は設けない．すなわち，2^64 以上の整数値も取り扱う事ができる．最後に，実行時エラーや警告のうち以下のモノは値を返すようにする．宣言のみで初期化や代入が行われていない変数を評価した値は 0，return 文を用いずに関数が終了してしまった場合の関数の返り値も 0，0 で割った商も 0，0 で割った余りは割られる数．

上述の文法や演算子以外の標準的な C 言語の機能は用いる必要がない．例えば，for 文等は用いる文法を組み合わせる事で表現でき，goto 文も書き変え可である．また，計算プログラムは入力と出力とが定められた状態で用いられるため，入出力機構は必要がない．

以上を用い，以下のように関数プログラムを定義する．

定義 4.1 (関数プログラム，k 入力プログラム) *4* 章で示された機能を持つプログラム（を示す文字列）を関数プログラムと呼ぶ．さらに，関数プログラムへの入力が k 個であるとき，それを明示して k 変数関数プログラム，または k 入力プログラムと呼ぶ．

5 計算の定義

定義 4.1 を元に，以下のように**計算 (computation)** を定義する．

定義 5.1 (計算) k を自然数，P を k 入力プログラム[3]，f を自然数上の k 変数部分関数とする．P が f を計算するとは，任意の自然数列 $(n_1, n_2, \ldots, n_k)$ に対し，以下が成り立つことである．

- $f(n_1, n_2, \ldots, n_k)\downarrow$ であるならば，P の入力変数に $n_1, n_2, \ldots, n_k$ の値を与えて P の実行を開始すると，$f(n_1, n_2, \ldots, n_k)$ の値を返り値として出力し実行が終了する．

[3] k 入力プログラムは文字の列なので大文字で示した

- $f(n_1, n_2, \ldots, n_k)\uparrow$であるならば，$P$の入力変数に$n_1, n_2, \ldots, n_k$の値を与えて$P$の実行を開始すると実行は終了せず永遠に止まらない（Pの外部からは永遠に止まらないかどうかは判断ができない）.

ここで「計算する」「計算プログラム，k入力プログラム」の定義が行われたので，計算可能性の定義をもう一度確認する.

定義 5.2 (計算可能性) 自然数上のk変数関数fに対しそれを計算するk入力プログラムが存在するとき，fは計算可能であるという．そのようなk入力プログラムが存在しないとき，fは計算不可能であるという.

これを用い，例題1.2,1.3を解く事が可能となる．さらに，定義5.1,5.2より，以下の定理が導かれる.

定理 5.3 自然数上の部分関数fが計算可能ならば，kを計算する関数プログラムが無数に存在する.

証明は構成的手法で可能である.

レジュメ置き場はこちら．レジュメのパスワードはa-nagaoです.
https://sites.google.com/view/a-nagao/course/theorycomp

URLのQRコード