計算論理
COMPUTATIONAL LOGIC

佐藤雅彦（京都大学）
亀山幸義（筑波大学）

講義プラン

- 11月28日
  - 午前(2限)のみ。。。亀山担当
  - 午後。。。佐藤先生担当
- 11月29－30日： 佐藤先生担当
- 12月4日： 亀山担当

- 講義とソフトウェアを使った演習から構成
- 成績評価： 出席、演習、レポート

証明

- 数学では中心的な役割
  - 厳密な証明が与えられていない命題は、ほとんど意味がない。
  - 「フェルマーの定理」は、正確には数百年間「フェルマーの予想」であり、最近ようやく「ワイルズの定理」になった。
  - $x^n + y^n = z^n \quad (n>2,\ xyz \neq 0)$
- コンピュータサイエンスでは？
  - アルゴリズムの正しさの証明 などはある。
  - しかし、一般に、「証明する」ことと「計算する」ことは全く別に見える。
  - 本当にそうか？ →本授業のテーマ

証明する

定理:どんな整数Nに対しても、それより大きな素数Pが存在する。(素数は無限にたくさんある。)

証明

- M=N!+1 とする。(N! は 1・2・…・N)
- (＊)Mが素数なら、P=Mとして終了。
- Mが素数でなければ、1ではない整数A,Bにより、M=AB
- A≦Nならば矛盾するので、A>N である。
- そこで、このNをMとして、(＊)へ。
- これを繰り返すが、Mの値は減少するので、無限に繰り返すことはない。→いつか、Nより大きな素数が見つかる。

計算する

アルゴリズム:整数Nを入力すると、それより大きな素数Pを出力する。

計算

- M=N!+1 とする。
- (＊)Mが素数なら、P=Mとして終了。
- Mが素数でなければ、1ではない整数A,Bにより、M=AB
- A≦Nならば矛盾するので、A>N である。
- そこで、このNをMとして、(＊)へ。
- これを繰り返すが、Mの値は減少するので、無限に繰り返すことはない。→いつか、Nより大きな素数が見つかる。

「証明する」ことは「計算する」こと？

- 「○○という性質を満たすXが存在する」という形の命題に対して
  - その命題を証明する
  - そのXを計算する

  という2つの行為は同じじゃないだろうか？

別の例

p^q が有理数となるような、無理数 p, q が存在する。

- 有理数：分数の形であらわせる数
- 無理数：有理数でない実数。たとえば、√2

別の例：証明する

p^q が有理数となるような、無理数 p, q が存在する。

証明

$a=\sqrt{2}$ とする。a^a は有理数か無理数のどちらか。

(1) 有理数のとき、$p=q=a$ として、証明終了（a は無理数なので）

(2) 無理数のとき、$p=a^a, q=a$ とすると、

$p^q=a^{a\cdot a}=a^2=2$ なので証明終了

別の例：計算する

p^q が有理数となるような、無理数 p, q を計算する。

$a=\sqrt{2}$ とする。a^a は有理数か無理数のどちらか。

(1) 有理数のとき、$p=q=a$ を返せばよい。

(2) 無理数のとき、$p=a^a, q=a$ を返せばよい。

で、結局 $p=a$ ？ $p=a^a$ ？

わからない！

ここまでのまとめ

- 「。。。が存在する」という形の命題の証明は、多くの場合、「。。。を計算する」というアルゴリズム（プログラム）に対応しそうである。
- しかし、対応しないこともある。

- どのようなとき、計算と論理は対応するのだろうか？
- 対応すると、どのような「嬉しさ」があるのだろうか？
- 計算と論理の本質は何だろうか？

計算と論理の対応

- 「存在証明」には、計算する手続きが含まれていることが多いようだ。
  - 素数の例
- 一方、計算とは対応しない証明もあるようだ。。。
  - 無理数・有理数の例

- 計算にぴったり対応する論理体系
  - 構成的論理の体系
- 数学や普通の推論で使う論理体系
  - 古典論理の体系

構成的論理[直観主義論理]

- 古典論理 ＝ 直観主義論理 プラス 以下のもの
  - $A \vee \neg A$
  - $\neg\neg A \Rightarrow A$
  - これらと同値な論理式
- 逆にいえば、これらの式を使わない証明であれば、計算と対応する。
  - かなり多くの証明は、構成的論理の範囲内で記述できることがわかってきた
  - 複雑な数学的証明も、「動かしてみる」ことが可能！

構成的プログラミング [SATO]

- プログラムの仕様

  $(x=yd+r \ \wedge \ r<y)$

  - 自然数xを正の整数yで割った商がdで余りがr
  - 構成的論理での次の式の証明　　から、

    $\forall x.\ \forall y>0.\ \exists d.\ \exists r.\ (x=yd+r \ \wedge \ r<y)$
  - (x,y) が与えられると、上記の性質を満たす (d,r) を計算するプログラム　が得られる。

    f (x,y) = ….　(自然数上の割り算プログラム)
  - 正しさが保証された割り算のプログラムが得られた！

    f(x,y) = (d,r) とおくと、

    どんなx,y に対しても $(x=yd+r \ \wedge \ r<y)$ が成立

構成的プログラミングによる割り算プログラムの導出

構成的論理での次の式の証明

$\forall x.\ \forall y>0.\ \exists d.\ \exists r.\ (x=yd+r \ \wedge \ r<y)$

- x に関する数学的帰納法で証明する。
- x=0のとき、r=d=0とおけば $x=yd+r \wedge r<y$となる。
- x=n+1のとき、帰納法の仮定より

  $n= y\ e + s \ \wedge \ s <y$ となる、e, sがある。

  Case-1. s+1<yのとき、d=e, r=s+1 とすればよい。

  Case-2. s+1=yのとき、d=e+1, r=0とすればよい。

以上より、数学的帰納法により証明された。

構成的プログラミングによる割り算プログラムの導出

構成的論理での次の式の証明から抽出されたプログラム

$\forall x.\ \forall y>0.\ \exists d.\ \exists r.\ (x=yd+r \ \wedge \ r<y)$

- 関数 f(x,y) は (d,r) というタプルを返す再帰的関数。
- x=0のとき、f(x,y)=(0,0)
- x=n+1のとき、

  f(n,y)=(e,s) とすると、

  Case-1. s+1<yのとき、f(x,y)=(e, s+1)

  Case-2. s+1=yのとき、f(x,y)=(e+1, 0)

以上。

「おお、プログラムだ！」「しかも、正しさの証明つき！」

まとめ

- 論理における「証明」も、かなりの程度、「計算」に対応する。
  - 証明を(プログラムのように)動かすことができる。
  - 証明を書けば、その中にプログラムが含まれる。
- 計算と論理は切っても切れない関係にある。
  - 佐藤先生は、計算と論理の基礎を探求し、その過程で、新しいプログラム言語の設計Zが必要、ということに思い当たった。
- プログラム言語 Z
  - 「証明を表すものだけをインスタンスとして持つクラス」が定義できる。
- 既存の道具を我慢しながら使うのではなく、基本概念や基本となる道具立てを1から自分で構築する。