計算論理学
COMPUTATIONAL LOGIC

亀山幸義（筑波大学）

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？　→本授業のテーマの1つ

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、$\sqrt{2}$

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！

P^Q が有理数となるような、無理数 P, Q が存在することの構成的な証明 (2012年度受講 林さん)

- log_3 4 (3を底とする4の対数)は無理数
  - これが有理数と仮定する。
  - 1以上の整数m,n に対して log_3 4 = n/m となる。
  - 4^m = 3^n である。
  - 素因数分解の一意性より、この式の整数解はm = n = 0しかないので、矛盾
- $P=\sqrt{3}$、 $Q=$log_3 4 とすると、
  - P^Q = 3^ (1/2・ log_3 4) = 3^(log_3 2) = 2

計算と論理の対応関係のまとめ

対応： 直観主義論理と型付きラムダ計算

対応	直観主義論理	型付きラムダ計算
Curry-Howard の対応	「ならば」だけの論理	単純型付きラムダ計算
	命題論理	＋直積、直和、空型
	一階述語論理	＋依存型(Σ、Π) Martin-Lof の型理論
Girard-Reynolds の対応	二階命題論理	＋多相型、体系F
Barendregt の λ-cube	高階論理	高階型付きラムダ計算、Calculus of Constructions (CoC)
Coquand-Huet	高階論理＋帰納的定義、余帰納的定義	Calculus of Inductive Constructions (Coqの論理)
Griffin, Parigot	古典論理	各ラムダ計算に「制御」を加えたもの

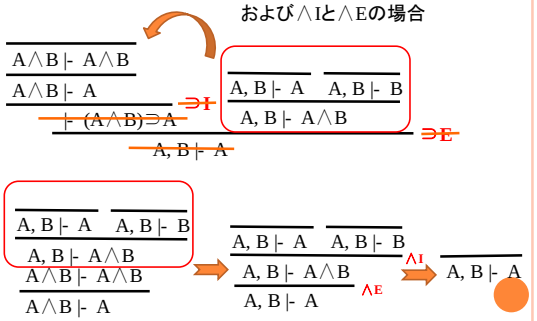
CASE 1: 命題論理との対応

対応	直観主義論理	型付きラムダ計算
Curry-Howard の対応	「ならば」だけの論理	単純型付きラムダ計算
	命題論理	＋直積、直和、空型
	一階述語論理	＋依存型(Σ、Π)

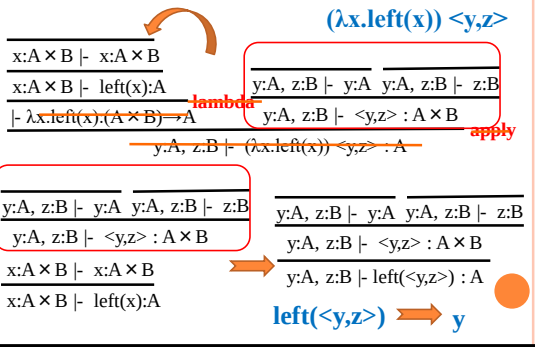
対応	直観主義論理	型付きラムダ計算
命題と型	⊃	→ （関数型）
	∧	× （直積型）
	∨	＋ （直和型）
	⊥	⊥ （空型）
規則	⊃I 規則 ⊃E 規則 …	lambda 規則 apply 規則 …
正規化(証明の無駄の除去)と計算	⊃I とE の正規化	ベータ簡約

証明の無駄の除去： ⊃Iと⊃Eの場合

および∧Iと∧Eの場合



計算： LAMBDAとAPPLYの場合とpairとleftの場合



CASE 2: 一階述語論理との対応

対応	直観主義論理	型付きラムダ計算
Curry-Howard の対応	一階述語論理	単純型付きラムダ計算＋直積、直和、空型 ＋**依存型(Σ型とΠ型)**

対応	直観主義論理	型付きラムダ計算
命題と型	述語 P(x), Q(x,y)	型としての述語
	∀ （すべての）	Π型 （依存積型）
	∃ （ある）	Σ型 （依存和型）

∀x∈Real. ∃y ∈Nat. x<y

f ： Π(x:Real) Σ(y:Nat) LessThan(x,y)

f(x) は、実数x に対して、それより大きな整数yとその証拠を返す関数

命題＝型、 証明＝プログラム、構成的解釈

対応	直観主義論理	型付きラムダ計算
命題と型	命題	型
証明とプログラム	その証明(証拠)	その型を持つ項(プログラム)

論理式の構成的解釈
　論理式が「正しい」とは、その論理式の証拠が(計算によって)得られること
　A∧B の証拠は、Aの証拠とBの証拠から構成される。
　A∨B の証拠は、「Aの証拠を持っている」という情報と、Aの証拠か「Bの証拠をもっている」という情報と、Bの証拠
　A⊃B の証拠は、(誰かが)Aの証拠をもってきたら、(それを使って)Bの証拠を返すことができるもの
構成的解釈は、Curry-Howard対応と、密接に対応する
　例： A⊃B の証拠は、A型の項をもらうと、B型の項を返す関数

命題＝型、 証明＝プログラム、構成的解釈

対応	直観主義論理	型付きラムダ計算
命題と型	命題	型
証明とプログラム	その証明(証拠)	その型を持つ項(プログラム)

例1． (A∧B)⊃(B∧A) は構成的解釈のもとで「正しい」
　誰かが A∧Bの証拠をもってきたら、B∧Aの証拠を返せるから
　　ーー λx. <right(x), left(x)> が証拠となる
例2． A∨(A⊃⊥) は構成的解釈のもとで「正しい」とは言えない
　Aの証拠を出すことはできないし、A⊃⊥の証拠も無理。

構成的プログラミング [SATO]

- プログラムの仕様
  $(x=y*d+r \ \wedge \ r<y)$
  - 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、
    $\forall x.\ \forall y>0.\ \exists d.\ \exists r.\ (x=y*d+r \ \wedge \ r<y)$
  - (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 $(x=y*d+r \ \wedge \ r<y)$ 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
　$\forall x.\ \forall y>0.\ \exists d.\ \exists r.\ (x=y*d+r \ \wedge \ r<y)$

- 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=y*d+r∧r<yとなる。
- x=n+1のとき、帰納法の仮定より
  n= y e + s ∧ 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
　$\forall x.\ \forall y>0.\ \exists d.\ \exists r.\ (x=y*d+r \ \wedge \ r<y)$

- 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！」

構成的プログラミング (プログラム抽出)

プログラムが満たすべき入出力仕様を論理式として書く。
　入力x,y
　出力d,r
　入出力仕様
　$\forall x.\ \forall y>0.\ \exists d.\ \exists r.\ (x=y*d+r \ \wedge \ r<y)$
それを直観主義論理(構成的論理)で証明する

すると、その仕様を満たすプログラムf が得られる。
　fの正しさ $\forall x.\ \forall y>0.\ (x=y*f1(x,y)+f2(x,y) \wedge f2(x,y)<y)$
　の証明も得られる。

Coqシステム

- CoC (Calculus of Constructions)
  - 単純型付きラムダ計算、依存型、高階論理を包含した論理体系(および対応する型システム)
- Coq のベースとなる型システム・論理(Calculus of Inductive Constructions)
  - ユーザが自由に帰納的定義等をできるようにしたCoCの拡張
- いずれにしても、型理論に基づいている
  - 論理でいえば、直観主義論理(構成的論理)
  - つまり、プログラム抽出ができる!
- あれれ、では Coq では A∨¬A も証明できないの?
  - それを仮定して、証明をすることはできる。

古典論理への拡張

- 古典論理は、通常の意味ではCurry-Howardの対象外
- しかし、プログラム言語の世界に「制御」をいれれば、ある程度対応することが知られている。
  - コントロールオペレータ
    - JavaやMLの exception (例外)、C の setjmp, longjmp
    - SchemeやSML/NJ の call/cc (第一級継続)
    - OCaml/delimcc や Scala の shift/reset (第一級限定継続)
- Curry-Howardの楽園のような世界ではないので、精密な理論展開が必要

ここまでのまとめ

論理と計算

- 論理における「証明」は、「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。
- 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。

形式的体系と型システム

- 形式的体系とは何か?
- 型システムの基本は?
- 型推論とは?

まとめ

もう少し「メタ」な視点で考えると。。。

- 計算機プログラムが、人間の頭脳による論理的な生産物である以上、両者は切っても切れない関係にある。
  - プログラムの検証のための論理 は当然だが、それだけでなく、
  - プログラム、プログラミング言語のモデル化、設計、理解。。。
- Curry-Howardの対応(およびその拡張)は、プログラミング言語の設計の指針となり得る。