ソフトウェア検証における計算理論的問題

安全性検証および等価性検証の事例

住井 英二郎
東北大学

Disclaimers?

- 自分の研究より分野のご紹介が中心です
- 「釈迦に説法」の部分はご容赦ください

What is Computer Science?

"Computer science is
no more about computers
than astronomy is about telescopes."

—Edsger Dijkstra

計算機科学

＝

計算に関する学問

What is a Program?

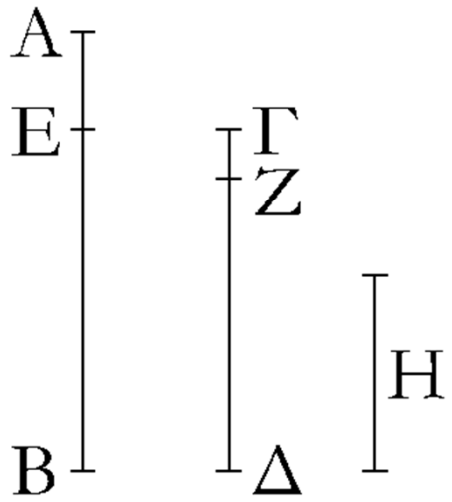
「プログラム」＝「計算の記述」

「プログラミング言語」＝
「計算を記述するための言語」

– ここではJavaやCなどに限らず、「λ計算」
「π計算」などの計算モデルをも含む

β′.

Δύο ἀριθμῶν δοθέντων μὴ πρώτων πρὸς ἀλλήλους τὸ μέγιστον αὐτῶν κοινὸν μέτρον εὑρεῖν.



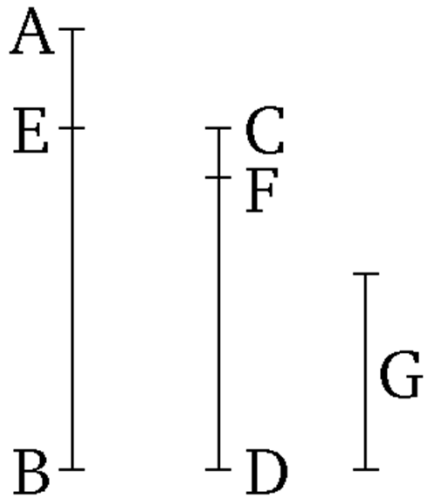
Ἔστωσαν οἱ δοθέντες δύο ἀριθμοὶ μὴ πρῶτοι πρὸς ἀλλήλους οἱ ΑΒ, ΓΔ. δεῖ δὴ τῶν ΑΒ, ΓΔ τὸ μέγιστον κοινὸν μέτρον εὑρεῖν.

Εἰ μὲν οὖν ὁ ΓΔ τὸν ΑΒ μετρεῖ, μετρεῖ δὲ καὶ ἑαυτόν, ὁ ΓΔ ἄρα τῶν ΓΔ, ΑΒ κοινὸν μέτρον ἐστίν. καὶ φανερόν, ὅτι καὶ μέγιστον· οὐδεὶς γὰρ μείζων τοῦ ΓΔ τὸν ΓΔ μετρήσει.

Εἰ δὲ οὐ μετρεῖ ὁ ΓΔ τὸν ΑΒ, τῶν ΑΒ, ΓΔ ἀνθυφαιρουμένου ἀεὶ τοῦ ἐλάσσονος ἀπὸ τοῦ μείζονος λειφθήσεταί τις ἀριθμός, ὃς μετρήσει τὸν πρὸ ἑαυτοῦ. μονὰς μὲν γὰρ οὐ λειφθήσεται· εἰ δὲ μή, ἔσονται οἱ ΑΒ, ΓΔ πρῶτοι πρὸς ἀλλήλους· ὅπερ οὐχ ὑπόκειται. λειφήσεταί τις ἄρα ἀριθμός, ὃς μετρήσει τὸν πρὸ ἑαυτοῦ. καὶ ὁ μὲν ΓΔ τὸν ΒΕ μετρῶν λειπέτω ἑαυτοῦ ἐλάσσονα τὸν ΕΑ, ὁ δὲ ΕΑ τὸν ΔΖ μετρῶν λειπέτω ἑαυτοῦ ἐλάσσονα τὸν ΖΓ, ὁ δὲ ΓΖ τὸν ΑΕ μετρείτω. ἐπεὶ οὖν ὁ ΓΖ τὸν ΑΕ μετρεῖ, ὁ δὲ ΑΕ τὸν ΔΖ μετρεῖ, καὶ

Proposition 2

To find the greatest common measure of two given numbers (which are) not prime to one another.



Let AB and CD be the two given numbers (which are) not prime to one another. So it is required to find the greatest common measure of AB and CD.

In fact, if CD measures AB, CD is thus a common measure of CD and AB, (since CD) also measures itself. And (it is) manifest that (it is) also the greatest (common measure). For nothing greater than CD can measure CD.

But if CD does not measure AB then some number will remain from AB and CD, the lesser being continually subtracted, in turn, from the greater, which will measure the (number) preceding it. For a unit will not be left. But if not, AB and CD will be prime to one another [Prop. 7.1]. The very opposite thing was assumed. Thus, some number will remain which will measure the (number) preceding it. And let CD measuring BE leave EA less than itself, and let EA measuring DF leave FC less

プログラムは難しい

プログラムと「数式」の違い：

- 繰り返し・再帰
  ⇒　計算が停止しない可能性
- 外部入出力や内部状態の変化（副作用）
- プログラムを受け取る・プログラムを返すようなプログラム（高階計算、メタ計算）
- 並列・分散計算
- オブジェクトやモジュールによる抽象化

etc.

出典：
朝日新聞

一日で300億円損失

証券 発注ミス

混乱、全面安

みずほ証券による株式誤発注の流れ

ジェイコム（発行済み株式は1万4500株）	
9:22	みずほ証券が 1株61万円を「61万株を1円」で売り注文
初値	67万2000円
9:30	ストップ安 67万2000円に
9:32	約47万株の買い注文（みずほ証券が買い戻し）
9:46	ストップ高 77万2000円に
10:06	終値 77万2000円

正 1株61万円▼誤 1円で61万株

東証、取引障害

新システム初日 午前中2万件

午前全面停止

2520銘柄、売買できず

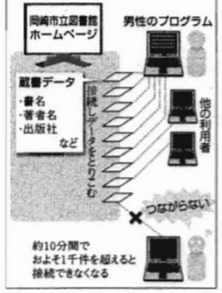
システム障害「午後に再開」

福岡・札幌も

注文殺到し大証も影響

「ソフトにトラブル」

蔵書情報集めたら…「サイバー攻撃」



ソフト自作の男性逮捕 → 起訴猶予

専門家「図書館側に不具合」

報道・システム障害

ウイルスバス

システム障害 羽田・31便欠航

日本航空のチェックインシステムにトラブルが発生、利用客で混雑する羽田空港の第1ターミナル＝3日午前8時37分、東京都大田区、福留庸友撮影

ダム突然開け放水

制御ソフト不具合

長野県阿智村の天竜川水系阿知川にある中部電力の駒場ダムで9日夕方、二つのゲートが突然上がり、ダムにためていた2万立方㍍の水が下流に流れ出した。同川の水位が一時、12㍍上がったが、流域一帯に被害は出ていないという。中電の調査で、ゲートを制御するコンピューターソフトに不具合があることが分かった。国土交通省中部地方整備局は10日、「考えられない事故で災害にもつながりかねない」として調査委員会を設置した。

中電飯田支店によると、9日午後6時すぎ、突然、ゲート開放を告げるチャイムが鳴り、同ダムを管理している平岡ダム管理所の職員が異状に気づき、ゲートの上昇を止めるため電源を切った。

阿智村の水上宗光総務課長は「梅雨や台風の時

プログラム理論・
プログラミング言語理論

- **プログラムとプログラミング言語の意味を数学的・論理学的に定義、性質を証明**
- **（少なくとも）1960年代から研究、最近になって（ようやく）実用段階に**
  - ML, Haskellなど関数型言語（外資系金融機関等、多数）
  - SPINなど古典的モデル検査器（Toyota車等、多数）
  - Astréeなどプログラム解析器（AirBus A340, A380等）
  - SLAMなどソフトウェアモデル検査器（Windows等）
  - IPA「形式手法適用調査」報告書
  - 日経エレクトロニクス特集「ソフトウエアは硬い」
  - 情報処理学会誌特集「フォーマルメソッドの新潮流」etc.

プログラム検証問題の例

- **型安全性(type safety)：**
  プログラミング言語の意味を
  状態遷移系で定義したとき、
  あるプログラムPが終了状態でなければ、
  次の状態が必ず存在するか？
  - 型安全でない例：C言語の未定義動作
- **プログラム等価性(program equivalence):**
  二つのプログラムP, Qが、
  任意の同じ入力（文脈）に対し、
  同じ出力（動作）をするか？

問題点

ほとんどのプログラム検証問題は決定不能

- 停止性
- 型安全性
  - プログラムPが安全で、プログラムQが安全でなければ、P;Qの安全性はPの停止性の逆に等しい
- プログラム等価性
  - プログラムPと、決して停止しないプログラムQとの等価性は、Pの停止性に等しい

解決策：保守的「近似」

例：停止性問題の保守的「近似解」

- Yes　⇒「必ず停止する」
- No　⇒「停止しないかもしれない」

「近似」の「精度」が問題

（「常にNo」では役に立たない）

Terminator Tackles an Impossible Task

By Rob Knies
August 16, 2006 11:01 PM PT

It is May 28, 1936. A paper entitled *On Computable Numbers With an Application to the Entscheidungsproblem* is received by the London Mathematical Society for publication in its journal.

In the paper, the latest salvo in a mathematical debate that extends back to the dawn of the 20th century, Cambridge academic Alan Turing proves that a general algorithm to determine whether all possible combinations of programs and initial

"Turing proved that, in general, proving program termination is 'undecidable,' " Cook says. "However, this result does not preclude the existence of future program-termination proof tools that work 99.9 percent of the time on programs written by humans. This is the sort of tool that we're aiming to make."

prestigious technical honor the A.M. Turing Award. Is it possible that one of his enduring proofs is about to be upended?

Not quite. But someday, the impact of Turing's result may be considered strictly

Related Links

- Byron Cook
- Alan Turing
- A.M. Turing Award
- Terminator
- SLAM
- Static Driver Verifier

問題点

- **「近似」の「精度」の客観的基準や理論的保証がない**
  - 現状は特定の設定での実験のみ

強い静的型付け (Strong Static Typing)

- ML, Haskellなど関数型言語で発展、Java, C#などにも部分的に導入
- 正しく型付けされたプログラムは未定義動作をしない
  - "Well-typed programs never go wrong" [Milner '78]
- 型付けは論理的規則に基づいて行われる
  - **プログラマは型付け規則を（少なくとも直観的には）理解する必要**

MLの型推論

- **プログラマが型を（理想的には）まったく書かなくても自動的に推論**
  [Hindley '69][Milner '78]

```
- fun compose f g x = g (f x) ;
val compose =
   fn : ('a -> 'b) -> ('b -> 'c) -> 'a -> 'c
```

- **一般には任意のkについてDTIME(2^{n^k})困難**

  k=1の例： let $x_1=(x_0,x_0)$ in let $x_2=(x_1,x_1)$ in
  let $x_3=(x_2,x_2)$ in let $x_4=(x_3,x_3)$ in ...

- **だが「現実的プログラム」では効率的**

問題点（と一つの「解決策」）

- 「現実的プログラム」をどう特徴づけるか？
  - "Order"と"arity"をboundすればO(nα(n))
    [McAllester '96, '03]
    - そのboundは本当に「現実的」か？
      (cf. type-level programming)

高階再帰スキーム(HORS)の検証

- 文脈自由（無限木）文法の拡張の一種
  - 一種の制限された関数型言語と見ることもできる

  例：S→Ac，Ax→a(x,A(bx))
  ⇒　a(c,a(bc,a(bbc,a(bbbc,a(bbbbc,...)))))
- あるHORSの生成する木が
  ある種の論理式（様相μ計算／MSO）を
  満たすかどうかは決定可能

  例：上の木の有限パスは必ずcで終わるか？(Yes)
  bの下にaが現れることはあるか？(No)
  - 一般にはk-EXPTIME完全[Ong '06]
  - 「実用上」は効率的なアルゴリズム[小林'09]

問題点

- 「実用上」をどう特徴づけるか？
  - 未解決

モデル検査

- ハードウェア記述言語やC言語等のプログラムが assertionや時相論理式等を満たすかどうか静的検証
- 整数など無限領域を、ある論理式を満たすかどうかで分割して有限近似
- **CEGAR [Clarke et al.]:** まず分割せずに疑似実行してみて、assertionが失敗したらその実行パスの**条件が充足可能か判定**、充足不能であればその条件で分割、これを繰り返す

  ⇒ SMT (SAT modulo theory)を高速に解く必要、「実用上は効率的な」アルゴリズムが存在

  - DPLL(T) [Nieuwenhuis他'06] 等

CounterExample-Guided Abstraction Refinement

例：if(x>0){y:=1}else{y:=0}; ...;
if(x>0){assert(y=1)}

まずx, y ∈ Intとして疑似実行

⇒ assert(y=1)が成立するかどうか
わからないので検証失敗

⇒ そのときの実行パスの条件x>0∧y=1∧y<>1
は充足不能なので、Intをx>0とx<=0,
y=1とy<>1に分割して疑似実行

⇒ x>0のときは必ずy=1になるので検証成功

プログラム等価性

双模倣(bisimulation)：

二つのプログラムPとQが双模倣的

⇔

- Pが入出力αを実行してP'になるならば、
  Qも入出力αを実行してQ'になり、
  P'とQ'も双模倣的
- Qが入出力αを実行してQ'になるならば、
  Pも入出力αを実行してP'になり、
  P'とQ'も双模倣的
  - 正確には余帰納法(coinduction)で定義

双模倣判定問題の計算量

- **正則(regular)なプロセス**：多項式時間
- **「文脈自由(context-free)」なプロセス (BPA, BPP)**：「冗長でない」(normed) プロセスの場合は多項式時間、一般には PSPACE困難・DEXPTIME以下

etc.

「実用上効率的」な判定アルゴリズムは（おそらく）あまり研究されていない

蛇足：環境双模倣 [住井他'04-]

- **通常の双模倣は高階計算や情報隠蔽（抽象データ型など）の扱いに論理的難点**
  - 余帰納的定義の母関数が単調にならない等
- **環境双模倣**：二つのプログラムPとQが**環境Eの下で**双模倣的　⇔
  - Pがvを出力してP'になるならば、Qはwを出力してQ'になり、P'とQ'は**Eを(v,w)で拡張した環境の下で**双模倣的
  - **Eから合成できる任意の(v,w)について、**Pがvを入力してP'になるならば、Qはwを出力してQ'になり、P'とQ'はEの下で双模倣的

適用言語例

住井らを中心とする研究：

暗号λ計算，多相λ計算，状態付き多相λ計算，
状態・メモリ解放付きλ計算，高階適用π計算，
高階分散π計算（２件），etc.

他研究者らを中心とする研究：

名前呼びλ計算，名前生成付きλ計算，
状態付きλ計算（２件），高階π計算，
プロトタイプベース・オブジェクト計算，
クラスベース・オブジェクト計算，etc.

環境双模倣判定問題の計算量？

- 一般には決定不能
- 有限プロセスなど、決定可能かもしれない場合の計算量も（まだ）まったく手つかず

結論

- 一般的に、プログラム検証問題の
  計算量クラスは非常に高いことが多い

- しかし「実用上」「現実的」には
  効率的に解けるケースも出てきている

- そのようなケースの理論的特徴づけが
  望まれるが、必ずしもよくわかっていない