RSA CONFERENCE JAPAN 2010

SEPTEMBER 9-10 | GRAND PRINCE HOTEL AKASAKA

SECURITY DECODED

高信頼・高安全ソフトウェアのための数理的検証手法：最新の研究と応用の現状

住井 英二郎
東北大学 大学院
情報科学研究科 准教授

セッションID: WH-004

本講演の概要と目的

- フォーマルメソッド（数理的・論理的ソフトウェア開発・検証手法）の基礎理論と応用事例のご紹介
  - 「古典」から「最新」まで
  - 例によるツールのデモンストレーションを交えて
    - 「釈迦に説法」の部分はご容赦ください
- 学界と産業界の橋渡しの一助（を目指す）
    - ご質問・ご意見・コメント・情報提供等いただけましたら幸いです

背景：情報処理システムの不具合の社会問題化

ソフトウェアが原因の国内事例（一部）：

- 企業や官公庁のPC 17万台が停止
  （2005年4月・ウイルス対策ソフトウェア）
- 400億円の誤発注の取り消しが失敗
  （2005年12月・証券取引システム）
- コンビニATMの取引2万件が失敗
  （2008年5月・銀行取引システム）
- 航空機遅延・欠航で1万5千人に影響
  （2009年6月・チェックインシステム）

国外事例：患者死亡（セラック25）、ロケット墜落（アリアン5）

フォーマルメソッドとは

数理論理学に基づく
ソフトウェア開発・検証手法
を幅広く指す

- 「形式（的）手法」と訳される（が語感は悪い）
- 欧州で発展、近年は米国や日本でも注目
  - 日経エレクトロニクス2005年12月19日号
    特集「ソフトウェアは硬い」
  - 情報処理2008年5月号
    「特集　フォーマルメソッドの新潮流」 など

フォーマルメソッドの利点と欠点

利点：　厳密である

⇒　曖昧さや漏れがない

欠点：　厳密である

⇒　相応の労力や慣れが必要

- ただしある段階の労力が以降の労力を軽減
  （設計と実装、実装とテスト・デバッグなど）

- 利点＞欠点となる場合に有用
- 「労力」を減らすべく研究が進められている

フォーマルメソッドの代表例（互いに独立ではなく深く関連）

- **形式仕様記述**:
  （日本語や英語ではなく）論理式で仕様を記述
- **定理証明**:
  論理式で書かれた性質を計算機を用いて証明
- **モデル検査**:
  効率的状態探索による自動的全数検証
- **型システム**:
  「型」と呼ばれる特殊な論理式で仕様を記述、
  論理的推論規則で検証

古典的なフォーマルメソッド（VDM，Z記法，B手法等）

- 「事前条件」「事後条件」により仕様を記述
  - プログラム理論（ホーア論理）からの概念
- 仕様から具体的実装（ソースコード）に「詳細化」(refinement)
- 欧州では鉄道・航空システムなど事例多数
  - 参照：IPA「形式手法適用調査」等
- 日本でもFelicaチップファームウェアなど複数事例
  - 参照：前出雑誌記事等

（説明のための単純な）例：max_index関数

- 日本語による仕様記述(?)の例：
  「max_index(a,n)は、長さnの整数配列aに対し、
  最大要素のインデックスを返す」
  - 複数の最大要素があったら？
  - aの長さがnでなかったら？　nが0だったら？
  - インデックスは0ベースか1ベースか？
- 論理式による仕様記述の例：
  事前条件(precondition)　length(a)=n∧n>0
  命令(command)　i := max_index(a,n);
  事後条件(postcondition)
  ∀j. 0≦j<n ⇒ a[j]≦a[i] ∧ (a[j]=a[i] ⇒ j≧i)
  「a[i]はa[0]からa[n-1]までの最初の最大要素」

実装と検証

実装例(C言語風):

```
i=0; for(k=1;k<n;k++) if(a[k]>a[i]) i=k;
```

forループの不変条件(invariant):

事前条件 ∧ $i<k \leqq n$ ∧

$\forall j.\ 0 \leqq j<k \Rightarrow a[j] \leqq a[i] \wedge (a[j]=a[i] \Rightarrow j \geqq i)$

「a[i]はa[0]からa[k-1]までの最初の最大要素」

- この条件をINV(i,k)とおく

証明すべき検証条件(verification condition):

$\forall a,n,i,k.$ (事前条件 ⇒ INV(0,1))

∧ (INV(i,k) ∧ k<n ∧ a[k]>a[i] ⇒ INV(k,k+1))

∧ (INV(i,k) ∧ k<n ∧ $a[k] \leqq a[i]$ ⇒ INV(i,k+1))

∧ (INV(i,k) ∧ $k \geqq n$ ⇒ 事後条件)

一般的知見

- 自然言語による仕様記述は
  曖昧さや漏れが起きやすい
- 論理式による仕様記述は
  曖昧さや漏れをなくす手段（に過ぎない）
  - ∀や∧などは「述語論理」の記号だが、
    そのような記号（や名前）を怖がらなければ、
    むしろ自然言語より簡単なことも多い
- 検証はそれなりに複雑になるので、
  定理証明器などのサポートが有用

実際のツールによるデモンストレーション

- VDM++（仕様・実装の記述・テストツール）
  - 検証条件の生成（一部）・テストが可能
    - 網羅的ではない
- Coq（対話的定理証明器）
  - 検証条件の網羅的証明が可能
    - 自動的ではない
- CVC3（自動的定理証明器）
  - 検証条件の網羅的・自動的証明が可能
    - 証明に失敗することもある

モデル検査：
網羅的・効率的・自動的な状態探索

- （多くの）テストと異なり全数検査
- 探索アルゴリズムを工夫

  例: CEGAR (Counter-Example Guided Abstraction Refinement)

  状態を合併（抽象化）して探索空間削減、
  検証が失敗したら、失敗した条件で分割（詳細化）
- 多くは時相論理(temporal logic)で仕様記述
  - 「X until Y」（条件Yを満たす状態になるまでは常に条件Xを満たす）など
  - 時相論理式ではなくassert文も記述可能

古典的モデル検査器の例: SPIN Model Checker

- 専用の言語(Promela)でモデルを記述
  - 一種の並行プログラミング言語
- 有限状態のシステムのみ検証可能
  - 無限状態は有限状態で近似

- （デモンストレーション）

CEGARつきモデル検査器の例：BLASTおよびSLAM

- アサーションつきC言語プログラムが対象
- CEGARにより効率的・自動的に検査
  - 無限状態でも網羅的に検査
  - ただしCEGARが失敗する場合は検査も失敗する
- （デモンストレーション）

現状のまとめ

- 自動化と一般性のトレードオフ
  - 一般には「決定不能問題」なので
    完全な両立は理論的に不可能
- 定理証明器・モデル検査器の進歩により
  「できるだけ一般的かつ自動的」に
  - 一般技術者向けの（特に日本語の）
    ツール・ドキュメントはまだまだ未整備
  - 国内（名古屋）の小規模企業でCoqによる
    仕様記述・検証とコード自動生成・納品の事例も

最近の研究

- ACM POPL 2010（米計算機科学会・プログラミング言語基礎理論シンポジウム）よりごく簡潔に紹介
  - ホーア論理を始め、フォーマルメソッドの基礎はプログラミング言語理論による部分が大きい
    - 「プログラミング言語」といってもJavaやCに限らず「λ計算」「π計算」などの抽象的理論も扱うため
  - 論文投稿207件中39件採択（日本から2件）
    - 採択率約19%だが、もっと厳しい年もある
    - 数件以外はプログラム検証・解析が主題

POPL 2010論文紹介(1/5)

- Atig他：弱い共有メモリモデル下での並列プログラムの検証
- Attiya他：並列プログラムの直列化可能性の検証
- Godefroid他：プログラムの「必ず成り立つ」性質と「成り立つかもしれない」性質を同時に解析する手法
- Chaudhuri他：プログラムの入力の小さな変化に対し出力が大きく変化しないか解析

POPL 2010論文紹介(2/5)

- Tristan他：コンパイラにおけるパイプライン最適化の検証器（の検証）
- Chlipala：定理証明器(Coq)で検証された、副作用つき関数型言語のコンパイラ
- Myreen：定理証明器(HOL)で検証された、JavaからのJITコンパイラ
- 寺内：「依存型」（関数の論理的仕様の一種）をプログラムから自動推論する手法

POPL 2010論文紹介(3/5)

- Dreyer他：状態を持つ抽象データ型の実装間の等価性を証明するための論理体系
- Magill他：ヒープ（メモリ）上のデータ構造操作を整数演算に変換して検証する手法
- Jost他：関数型プログラムの時間・メモリ消費量を静的に自動解析する手法
- Malecha他：定理証明器(Coq)で検証された関係データベースの実装

POPL 2010論文紹介(4/5)

- Podelski他：ヒープ（メモリ）上のデータ構造の不変条件を自動推論する手法
- Nanevski他：分離論理（ヒープに関する論理体系の一種）をCoq上で表現
- Srivastava他：ある種の仕様からプログラムを自動生成する手法
- Vechev他：並列プログラムの同期コードを仕様から自動生成する手法

POPL 2010論文紹介(5/5)

- Broberg他：プログラム中の機密情報の流れを静的に検証する手法の一つ
- Bhargaven他：暗号プロトコルのセキュリティを型システムで検証する手法の一つ
- 小林他：ある種の木構造変換プログラム（XML処理など）の性質を静的に検証する手法

結論にかえて

- 「定理証明やモデル検査はまだ難しすぎる」と思われた方は、OCamlやHaskellなどの

  関数型言語

  を是非ともお試しください
  - フォーマルメソッドというほどではないが密接な関連
    - Coqも強力な関数型言語の一種
  - 外資系金融機関を中心に既に実用化