

# 実装モデル検証

平成 21 年度シラバス

2009年2月25日

国立情報学研究所

トップエスイープロジェクト

代表者 本位田 真一

## 1. 講座名

実装モデル検証

## 2. 担当者

田辺 良則、Cyrille Artho

## 3. 本講座の目的

本講座では、ソフトウェアプログラムの検証問題を扱う。ネットワークを介したメッセージ通信を通して動作する分散ソフトウェアの振る舞いを網羅的に検証する技術、および計算機上で扱える程度の規模に検証問題を変換する抽象化技術を習得する。具体的には、Javaプログラムのモデル検査ツールJava PathFinderを使用し、実際のシステム開発に適用する方法を習得する。講義を通してモデル検査技術の理解を深め、実問題への適用能力を体得できる効果が期待できる。

## 4. 本講座のオリジナリティ

Java PathFinder などのソフトウェアプログラムのモデル検査ツールを使用した検証については、従来多くの同様な講座が開講されている。しかしそれらの講座には後述する問題点があったため、受講生が習得した内容を速やかに開発現場で適用するための障害となっていた。本講座では、問題点を解消し、本講座受講後 Java PathFinder ツールを速やかに開発現場で適用できるように配慮している。

表 1 に、既存の講座の問題点と、本講座における解を示す。

表 1 既存の講座の問題点と、本講座における解

既存の講座の問題点	本講座における解
<p>現実的なシステムのプログラムは、通常 GUI や通信機能のための標準ライブラリを使用しているが、この場合、そのままでは <b>Java PathFinder</b> を使用して検証することができないため、そのようなライブラリを使用しない、簡単なプログラムしか扱っておらず、習得した内容を、実際のシステム開発に適用するのが難しい。</p>	<p>現実的なシステムのプログラムの検証に必要な、複数プロセスのプログラムを 1 プロセスのものに変換するノウハウ、また抽象化によりライブラリを使用しないプログラムに変換するノウハウを習得するので、実際のシステム開発にも容易に適用可能である。</p>

## 5. 本講座で扱う難しさ

近年、ソフトウェアの大規模化、複雑化が進んでいる。典型的な例として、家電の制御ソフトウェアを考えると、複数の家電機器がネットワークに接続され、相互に連携して動作することで 1 つの機能を実現する、ネットワーク家電製品の市場が急速に立ち上がりつつある。従来の家電機器の制御ソフトウェアは単体で動作するように実装されており、単体での試験の実施が比較的容易であった。一方、ネットワーク家電では、他の機器との連携動作のためのプロトコルが組み込まれた分散処理型の制御ソフトウェアが中心的役割を担う。ホームネットワークに接続される機器やインターネット上のサービスとの接続関係を常に監視し、時折発生する接続要求や不慮の接続エラーなどに迅速かつ安全に対応できる機能が搭載されていなければならない。このように、大規模化に合わせて、ソフトウェアの安全性に対する要求も高くなっている。ところが現実には、ソフトウェア規模の増大につれ、相互作用の振る舞いが複雑になってしまうなどの原因により、綿密にプログラムを設計したとしても、機器間で資源を奪い合う競合状態のような、論理的に安全でない状態がプログラムの中に混入してしまう可能性が増大している。ソフトウェアの全動作パターンを手で検証する従来型のソフトウェア試験は、膨大な工数を必要とするために、もはや現実的な手段ではなくなっている。さらに、分散ソフトウェアの場合、ネットワークを介したメッセージ通信を通して相互作用を行うが、従来の試験ではソフトウェアの外部からメッセージの交換順序等を制御することは容易ではないため、振る舞いを網羅的に試験できてはいないのが実状である。

## 6. 本講座で習得する技術

本講座は、Java 言語で実装された分散ソフトウェアプログラムに対するソースコードモデル検査技術を扱う。モデル検査器は `Java PathFinder` を使い、このモデル検査器特有のノウハウと、一般のソースコードモデル検査技術の両方が身につくようにする。対象とするプログラムは、主として並行動作を行うものであり、そのようなプログラムの安全性および活性を検証することができるようになる。さらには、異なる `Java VM` 上で動作する複数のプログラムが、ネットワークを介したメッセージ通信により相互作用を行う、分散処理型のソフトウェアまでをカバーする。これには、分散ソフトウェアが利用する通信ライブラリを適切に置換することによって、単一 `Java VM` 上の並行ソフトウェアへと変換する技術が含まれる。また、一般にモデル検査においては、状態爆発と呼ばれる、検証のための探索空間が巨大になってしまう問題への対処が重要であるが、本講座で扱うソースコードモデル検査では、特にこの問題が生じやすい。これに対処するための、探索空間を縮減する手法も習得する。

## 7. 前提知識

本講座の受講生は、以下の項目を習得済みであることが望ましい。

- Java プログラミング
  - ソケットを使用した通信プログラミングを含む
- 以下に関する基礎理論
  - 並行プログラム：非決定性、資源共有・排他制御、通信、安全性、生存性、公平性
  - 状態探索手法：深さ優先探索、幅優先探索
- モデル検査技術の原理
  - 時相論理：構文、意味論(Kripke 構造)、パターン(安全性、生存性など)
  - 遷移系とその抽象化：データ抽象化

なお、これらの項目のうち Java プログラミング以外は、全て「基礎理論」講座で習得可能である。

## 8. 講義計画

第1回から第9回までの講義は日本語で、第10回から第15回までの講義は英語で行われる。

### ・ 概要

第1回：概論とツール紹介

第2回：JPF 検証における基本技法

第3回：ソフトウェアの安全性検証(1)

第4回：ソフトウェアの安全性検証(2)

第5回：ソフトウェアの活性検証(1)

第6回：ソフトウェアの活性検証(2)

第7回：スケジューリングの公平性

第8回：抽象化

第9回：グループ演習

第10回：ネットワークソフトウェアの検証の難しさとその解法（概論）

第11回：centralization モデルによる white-box アプローチ

第12回：cache モデルによる black-box アプローチ

第13回：linear cache

第13回：branching cache

第15回：グループ演習

### ・ 詳細

第1回：概論とツール紹介

#### ➤ 問題提起

◇ ソフトウェア開発における課題

◇ モデル検査の位置づけ

#### ➤ ソフトウェア開発におけるモデル検査技術の活用

#### ➤ ツール紹介

◇ JPF、他

#### ➤ 演習

◇ JPF の基本操作

第2回：JPF 検証における基本技法

#### ➤ assertion

#### ➤ listener

#### ➤ property

- 演習

- ✧ Bridge 例題

#### 第 3-4 回：ソフトウェアの安全性検証(1)

- 安全性とは
- 安全性の表現
- JPF における安全性検証

#### 第 5-6 回：ソフトウェアの活性検証(1)

- 活性とは
- 活性の表現
- JPF における活性検証
- 演習

- ✧ ネット家電例題

#### 第 7 回：スケジューリングの公平性

- 2つの公平性
- LTL による公平性の表現
- Java プログラムにおける公平性

#### 第 8 回：抽象化

- 状態爆発問題
- 抽象化技法
- JPF 抽象化ライブラリ

#### 第 9 回：グループ演習

- 並行プログラムに対するモデル検査
- 2人または3人による発表
- 質疑応答および講評

#### 第 10 回：ネットワークソフトウェアの検証の難しさとその解法(概論)

- 外部プロセスのスタブ化
- ロールバック
- 演習問題

- ✧ Daytime server

#### 第 11 回：centralization モデルによる white-box アプローチ

- 単一 VM 上のマルチスレッド化
- 演習

#### 第 12 回：cache モデルによる black-box アプローチ

- Centralization では扱えない問題に対する解法
- 演習

#### 第 13 回：linear cache

- メッセージ交換が決定的である場合のモデル検査技法
- 演習

第14回 : branching cache

- メッセージ交換が非決定的である場合のモデル検査技法
- 演習

第15回

- グループ演習
  - ◇ ネットワークプログラムに対するモデル検査
  - ◇ 2人または3人による発表
  - ◇ 質疑応答および講評

## 9. 教育効果

本講座を受講することにより、実際のシステム開発、特にネットワークソフトウェアの開発に適用可能な、ソフトウェアプログラム検証手法を習得できる。その結果、開発現場において、モデル検査ツールを活用することにより、信頼性の高いシステムを、効率的に開発することができるようになる。

## 10. 使用ツール

Java PathFinder : Java プログラムのモデル検査ツール

- ・ 使用する上での難しさ
  - 複数プロセスで動作したり、通常 GUI や通信機能のための標準ライブラリを使用したりしているような、現実的なプログラムの検証が難しい
  - 検証項目の表現が難しい
  - 検証の結果がエラーの場合、反例をプログラム修正に反映するのが難しい
- ・ 使用上必要なノウハウ
  - 通信の **centralization**
  - ライブラリ抽象化
  - 検証項目表現
    - ◇ 安全性表現ノウハウ
    - ◇ 活性表現ノウハウ
  - 反例分析
- ・ 選択理由、実用性：実用性が最も高い Java プログラムのモデル検査ツール

## 11. 実験及び演習

規模の小さなものから複雑な問題へと順に取り組むことによって、モデル検査技術を段階的に体得する。まず、並行ソフトウェアに対する検証問題を通して、モデル検査技術の基本的素養を学ぶ。次に、分散ソフトウェアに対する検証問題を通して抽象化技術を習得し、モデル検査技術の現実問題への適用能力を養う。

## 12. 評価

演習課題レポート、プレゼン発表、出席日数を総合して評価する。

### 13. 教科書/参考書

- B. Berard, et al, “Systems and Software Verification: Model-Checking Techniques and Tools,” Springer Verlag, 2001.  
モデル検査手法を利用したソフトウェア検証について、入門から実践までの一通りが述べられており、この講義に最適である。
- E. M. Clarke, et. al, “Model Checking,” MIT Press, 2000.  
モデル検査手法の原理を理論的に述べたものであり、上記教科書を補うのに最適である。
- W. Visser, et. al. “Model checking programs,” Automated Software Engineering Journal, vol.10, num.2, 2003.
- F. Lerda and W. Visser. “Addressing dynamic issues of program model checking,” Proceedings of SPIN2001, 2001.  
モデル検査ツール Java PathFinder の原理を理論的に述べたものであり、上記教科書を補うのに最適である。