

Java PathFinder によるモデル検査における GUI の提案

横山 翔一[†] 栗原 正仁[‡]

北海道大学 大学院情報科学研究科[§]

1 はじめに

有限状態並行システムの検証を自動的に行う技術として、モデル検査手法が注目されてきている。しかし、従来のモデル検査ツール [1] は、これを扱うために独自のシステム記述言語の習得が必要であったり、検証結果の出力が分かりにくくエラーの原因が特定しにくいなど、そのユーザビリティの向上が課題の一つとして挙げられている。ここでは、Java プログラムを対象とするモデル検査ツールである Java PathFinder [2] の機能を拡張することで、並行システムの振舞いをインタラクティブに理解しやすい形で表示し、さらにデバッガとして有用性の高いツールの実現を目指す。

2 Java PathFinder

2.1 ツールの概要

Java PathFinder (JPF) は、実行可能な Java バイトコード・プログラムを検証するためのモデル検査ツールである。JPF は Java プログラムの全ての潜在的な実行経路を体系的に探索することによって、デッドロックのようなエラーがプログラムに内在しないか、またプログラムが与えられた仕様を満たすかについて自動的に検証を行う。具体的には、JPF は一種の Java バーチャル・マシンであり、逐次 Java バイトコード命令を実行し、命令の実行毎にプログラムの状態を定義し保持する。その際に状態のマッチングとバックトラックを駆使し、可能性のある全ての実行経路を網羅的に探索する。検証の際には On-the-fly Partial Order Reduction という技術を用いて、実行時に他のスレッドに影響を与えない連続したバイトコード列

を自動的にグルーピングして 1 つの状態遷移として扱うことにより、状態数の削減を実現する。また検証の結果エラーが検出された場合には、エラー状態へ至るまでの実行経路を出力する機能を備える。対象プログラムに対する仕様は、Java のクラスとして定義する。

2.2 ユーザ・インタフェース

JPF では、コマンドライン・インタフェースに加えて、静的な画像によるグラフ描画を用いた GUI が提供されている。JPF による検証の後に、その結果として得られる対象プログラムの状態遷移図を表示させることが可能である。状態遷移図は状態ノードと遷移ノードによって構成され、ユーザはシステムの状態遷移の様子を視覚的に理解することができる。

しかし、モデル検査は並行システムに対する検証をその特徴とするものの、ここで提供される GUI 機能では対象プログラム全体の状態遷移図しか知ることができず、内部で並行動作する各スレッドの振舞いを知ることはできない。また、内部状態や遷移に関する情報はデバッグの際に有用と考えられるが、それらの多くの情報を静的なグラフ上に表示すると、グラフが煩雑で見づらくなるという欠点が挙げられる。

3 提案システム

3.1 提案システムの概要

本システムは、JPF による検証結果として得られる状態遷移図を、インタラクティブに操作可能な GUI アプリケーションの形で表示する。表示されたグラフは、マウス操作によって拡大・縮小、視点移動を行うことができ、ユーザは容易にグラフを調べることが可能である。

3.2 各スレッド遷移図の抽出

本システムでは検証の際に、プログラムの実行中に生成される各スレッドの振舞いを自動的に抽

Extension of GUI in Model Checking with
Java PathFinder

[†]Shoichi Yokoyama

[‡]Masahito Kurihara

[§]Graduate School of Information Science and
Technology, Hokkaido University

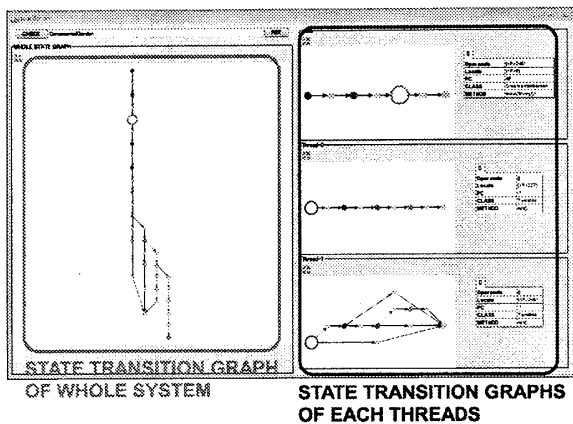


Fig. 1: 提案システムによるグラフ表示

出し、システム全体の状態遷移図に加えて各スレッドの状態遷移図を個別に表示する。さらに、これらのシステム全体の状態遷移図と各スレッドの状態遷移図の双方間の対応関係を示す機能を備える。ユーザがシステム全体の状態遷移図中の状態ノードをクリックすると、その状態に対応する各スレッドの状態遷移図上の状態ノードがマークされる。また、この逆の参照も可能となっている。

3.3 内部情報の表示

本システムでは、特にデバッグの際に有用となる機能として、グラフ上の各ノードに対応するプログラムの内部状態に関する情報を、クリック操作によってインタラクティブに表示する機能を提供する。遷移ノードをクリックした際には、選択された遷移で実行されるバーチャル・マシン命令とそれに対応するソースコード上の記述が表示される。また状態ノードをクリックした場合には、各スレッドの状態遷移図上において、選択された状態に対応するスタック・フレーム内部の情報、つまりメソッドのローカル変数配列や、処理対象データを保持するオペランド・スタック、該当メソッドのクラス実行時コンスタント・プールへの参照などの情報を表示する。

さらに、ユーザが Java プログラムの振舞いを理解する上で最も注目するものの 1 つに、ソースコード上で定義される変数の値が挙げられる。このため本システムでは、状態ノードが選択された際にその状態における変数値を表示する機能を提供する。ソースコード上で定義されている変数の中で、ユーザが興味のあるものを検証の際に指定

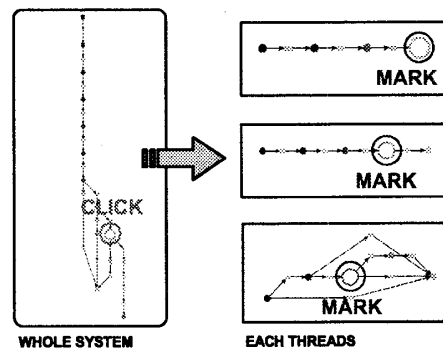


Fig. 2: システム全体と各スレッド間の対応関係の表示

することにより、グラフ上でその変数値の遷移を知ることができる。

これら様々なプログラムの内部情報を提示することにより、検証の結果エラーが発見された場合には、その原因を究明するためのデバッガとしても本システムを有効に活用することができる。

4 まとめ

主に静的な画像形式で結果を表示する従来の JPF に対して、本研究ではインタラクティブに操作可能な表示方法を実装したことによって、並行システムの動的な振舞いをより直感的に理解しやすい形で示すと同時に、システムに関するより多くの情報を分かりやすい形でユーザに提供することができた。また、各種の内部状態の情報を表示する機能を設けたことによって、よりデバッガとしての有用性の高いツールを実現することができた。

今後は、仕様の記述に関するインタフェースの改良や、出力される状態遷移グラフに対するユーザ定義による抽象化について検討したい。

参考文献

- [1] 例えば, Gerard J.Holzmann:“THE SPIN MODEL CHECKER”, Addison-Wesley Pub (2003) .
- [2] W.Visser, K.Havelund, G.Brat, S.Park and F.Lerda:“Model Checking Programs”, Automated Software Engineering Journal, Volume 10, Number 2 (2003) .