ソースコードに基づくモデル検査における反例の理解支援環境の構築

市井誠[†1]　中川雄一郎[†1]　小川秀人[†1]

我々はモデル検査を用いたソフトウェアの不具合解析に取り組んでいる．モデル検査を用いることで不具合シナリオ候補は反例として出力される．これまで，モデル作成工数の削減のため，ソースコードから検証モデルを抽出する POM 基盤を開発してきた．しかし偽反例に対応するための反例の理解の難しさが課題として挙げられていた．本稿では反例とソースコード，モデル間のトレーサビリティおよび可視化に基づく，反例の理解支援環境を提案する．

An Counterexample Comprehension Supporting Environment for Source Code-based Model Checking

MAKOTO ICHII[†1]　YUICHIRO NAKAGAWA[†1]
HIDETO OGAWA[†1]

We are developing developed the Program-Oriented Modeling (POM) platform, which extracts software models from source code in order to reduce the cost for building verification models for software fault analysis using model checking. Users have trouble to interpret counterexamples, that are candidates of target fault scenario, of model checking in order to address specious counterexamples caused by excessive abstraction. In this paper, we propose an environment to support comprehension of counterexamples comprising visualization and traceability among source code, model and counterexample trace.

1. 背景

ソースコードからモデルを抽出する Program Oriented Modeling（POM）基盤の研究を進めている [1)]．POM は使用目的に応じた抽象度のモデルを得る為，変換ルールを構成することで柔軟なモデル抽象化を実現する．これまで，C 言語のソースコードから，モデル検査ツール SPIN[2)]むけの検証モデル（Promela）を抽出するツールを開発してきた．

POM 基盤によるソフトウェア開発支援の 1 つとして，製品開発で発生した不具合の発生シナリオを得る不具合解析を検討している．本不具合解析技術では，発生した不具合を検査性質として定義し，POM 基盤で抽出した検査用モデルに対してモデル検査を行う．モデル検査の結果は反例と呼ばれ，検査性質を満たす（もしくは満たさない）場合のモデルの動作シナリオを示す．モデル検査で得られる反例が，不具合に至るシナリオの候補となる．これにより，タイミングに依存する不具合など，製品に対するテストでは再現が難しい不具合の発生シナリオの解析と対策ができる．

POM 基盤を用いた不具合再現の手順は図 1 のようになる．まずソースコードからモデルを抽出するための抽象化を検討し，それを実現する変換ルールを構成する．POM 基盤はソースコードと変換ルールを入力として実行され，自動的にモデルを抽出する．続いて再現したい不具合をプロパティとして記述し，モデル検査を実施する．

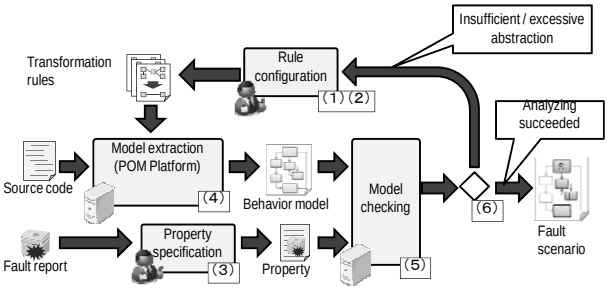

図 1 不具合再現の手順

しかし，ソースコードを抽象化した検査用モデルを検査対象とするため，得られた反例が元のソースコードでは発生しない場合がある．このような反例は偽反例と呼ばれる．偽反例の場合には，POM 基盤による異なる抽象度でのモデル再抽出を行い，モデル検査を再実行する．モデル検査を用いた不具合解析において，反例の内容を理解し，期待する反例であるか，偽反例であるかを判断するコストが大きく，不具合解析実用化の課題の 1 つであった．

反例の理解が困難となる理由は次の 2 つが存在する．(1) Promela/SPIN に由来: SPIN の反例はテキスト列（トレースログ）として表現され，可読性が低い．一方，SPIN 付属の反例可視化ツール（iSpin）のシーケンス図は，大きな反例に対して視認性が低い．(2) 変換・抽象化に由来: POM 基盤により変換されたモデルは，元のプログラム言語（C 言語）の意味論を Promela の意味論で再表現するとともに，状態爆発を防ぐためにソースコードの一部を捨象している．そのため，検査用モデルと元のソースコードとで構造が変

†1 株式会社 日立製作所 横浜研究所
Yokohama Research Laboratory, Hitachi Ltd.

化している部分があり，反例の示す動作シーケンスが，元のソースコードと直接対応付かない．

2. 反例理解支援環境

本稿では，反例理解にかかるコストを削減し，不具合解析の実用性をさらに向上するため，次の様な方針で反例理解支援の方式を検討した: (1) 大規模な反例のトレースログの理解性向上のため，利用者の指定する条件を用いて反例の一部を表示もしくは隠蔽する(フィルタリング)．また，反例をシーケンス図として表示することで直観的な理解を助ける（図表現）．(2) 反例中のステップが，システム動作全体を表現しているソースコード/モデルのどの部分に対応するかを把握しやすくするため，反例・ソースコード・モデルの相互のトレーサビリティを実現する．

2.1 可視化による理解性向上

トレースログを，変数や関数に基づいてフィルタリングする．具体的には，指定した変数が指定した値になっているステップや，指定した関数内/外のステップといった指定により，トレースログをフィルタリングもしくはハイライト表示（図 2）する．また，トレースログに表示する変数も指定可能とすることで，状態変数など，不具合に密接に関わる変数に注目しやすくする．

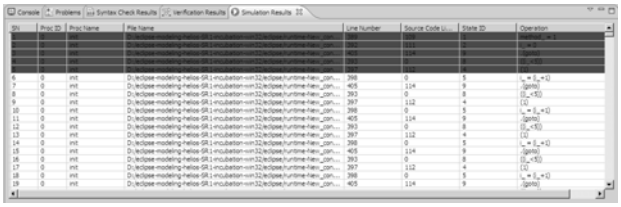

図 2 トレースログのハイライト

また，プロセス間の通信および関数呼出，変数への代入を表現したシーケンス図を出力する（図 3）．表示する要素はユーザが指定することができ，関心のある挙動に絞り込んだ表示をおこなうことができる．

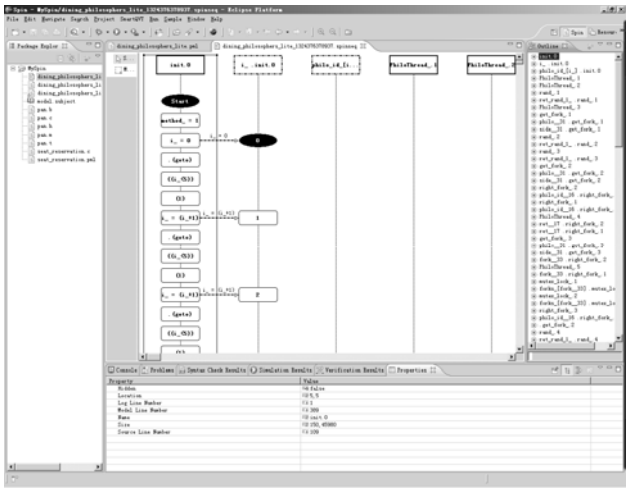

図 3 反例のシーケンス表示

2.2 トレーサビリティ

反例・検査用モデル・ソースコードそれぞれの間でのトレーサビリティを提供する（図 4）．ウィンドウ上の領域それぞれに，ソースコード（C)・検査モデル（Promela)・トレースが表示されている．反例トレースのステップを選択することで，ソースコードおよび検査モデルの該当行がハイライト表示される．逆に，ソースコード行（モデル行）を指定することで，対応するモデル行（ソースコード行）と，反例トレース中のステップがハイライト表示される．また，シーケンス図を表示している場合には，シーケンス図も含めた 4 者間で対応付けをおこなう．

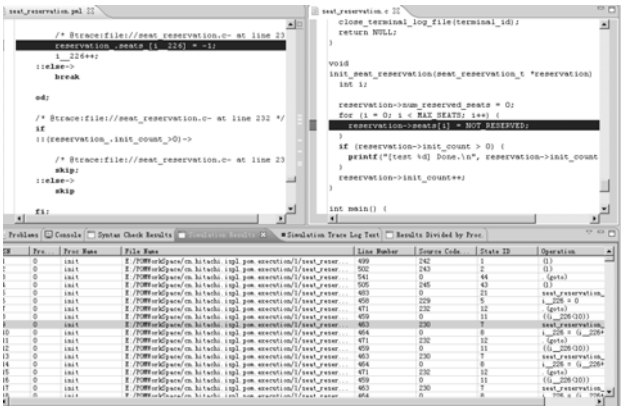

図 4 トレーサビリティ表示

本機能は，出力されるモデル（Promela）に元のソースコードの位置情報をコメント行として埋め込み，反例表示時にモデルのコードを解析し，コメントから位置情報を抽出することで実現される．

本理解支援環境は，POM 基盤とのシームレスな連携を実現するため， SPIN 実行機能を含む Eclipse プラグインとして開発した．これにより，図 1 に示す不具合再現の手順，すなわち，ソースコードからのモデル抽出およびモデル検査の実行，反例の理解の一連の流れを，異なるツールへの切り替えなしに行うことができる．

3. まとめと今後の課題

不具合解析に対するモデル検査の応用における課題として反例の理解の困難さに注目し，理解支援環境を構築した．今後は適用実験により本環境の評価および改善を進めていく予定である．

参考文献

1) 市井，中川，小川, “メタモデルを用いたソースコードからのモデル自動抽出手法の提案”, 情処研報, SE-174-1, pp.1—8, 2011
2) 中島, “SPIN モデル検査”, 近代科学社, 2008