B-018

状態遷移を持つオブジェクト間通信のモデル検査技術
Model Checking for Interaction between Objects with States

大貫 智洋† 上野 浩一郎† 磯田 誠†
Tomohiro Onuki, Koichiro Ueno, Makoto Isoda

1. はじめに

近年、情報システムの大規模化・複雑化が進み、設計の不具合を漏れなく検出することが難しくなっている。なかでも分散システムでは、メッセージ通信や状態遷移が生起するタイミングによって膨大な数の動作シナリオが存在する。そのため、全ての動作シナリオを人手により検査し、不具合を検出することは現実的でない。

この課題を解決する技術としてモデル検査の適用研究が盛んである。しかし、従来のモデル検査ツールは、モデルの記述が困難である、検査結果の解析が困難であるなどの課題があり、開発現場に十分に普及していない。

そこで我々は、オブジェクト間通信のモデル化に適した UML をベースとするモデル検査ツール「DESIMOC」を開発した。本稿では、従来技術の課題とその解決策を示した後、適用例を用いて本ツールの評価を行う。

2. 従来記法の課題と解決

2.1 従来のモデル検査の記法の課題

（1）ツール固有の記法に基づく入力の記述

モデル検査ツールは、検査対象システムの仕様を記述したモデルと、システムの満たすべき性質を記述した検査項目を入力とする。一方、入力モデル・検査項目の記述には、ツールに固有の汎用的なプログラム言語が用いられる[1]。そのため、入力モデルや検査項目を正しく記述するためには、ツールに固有のプログラム言語を十分に理解する必要がある。さらに、仕様の変更や検査項目の変更のたびにプログラムを記述する必要があるため、プログラム作成の労力を要する、属人的な誤りが入り込むなどの課題がある[2]。

（2）ツール固有の記法に基づく出力の記述

モデル検査ツールにより検出された不具合は、ツールに固有の記法を用いた図や、プログラムの実行ログ形式で出力される。ツールの記法によっては、不具合の内容を特定することが困難な場合があり、ログを解析する作業が必要とされる[2]。

2.2 UML を用いた記法による課題の解決

これらの課題に対し、我々は UML[3]をベースとした記法に基づくモデル検査ツール「DESIMOC」を開発した。UML で定義される記法の中から、状態遷移を持つオブジェクト間通信のモデル化に必要となる記法だけを選択・拡張して用いることで、モデルの記述を容易なものとする。

なお、本ツールの検査機能にはモデル検査ツール「SPIN」[4]の検査機能を用いており、本ツールの記法と SPIN 固有の記法を相互に変換する処理が行われる。本ツールによるモデル検査のプロセスを図 1 に示す。

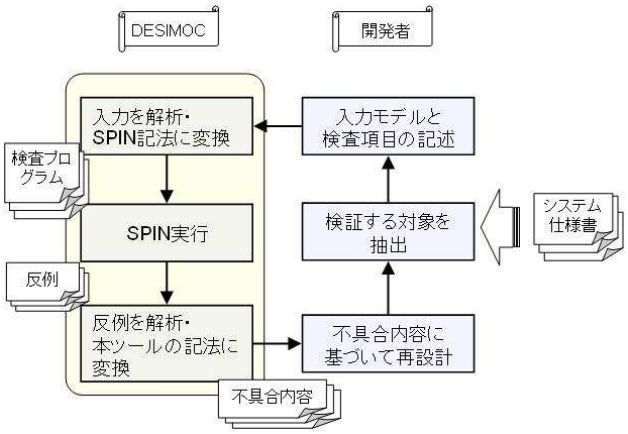

図 1　本ツールによるモデル検査のプロセス

（1）UML を用いた入力記法

入力モデルの記述には、UML シーケンス図と状態マシン図を用いる。シーケンス図ではオブジェクト間の通信を定義し、状態マシン図では各オブジェクトの状態遷移を定義する。本ツールで扱う記述要素は表 1 に示す通りであり、ツール利用者はこれらを適切に組み合わせるだけでモデルを記述することが可能である。

表 1　入力モデルの記述要素

シーケンス図	状態マシン図
・ライフライン	・状態
・メッセージ	・開始擬似状態
― 同期 / 非同期	・終了状態
・複合フラグメント	・遷移
― オルタナティブ	― 外部遷移
― アサーション	― 無条件遷移

一方、検査可能な性質は、以下のとおりである。

①到達可能性：初期状態から特定状態に到達できる
②安全性：デッドロックや無限ループがない
③活性：いつか必ず特定状況が発生する
④公平性：各々の並行処理が必ず進む

以上の性質を検査するため、本ツールではあらかじめ定義した自然言語の定型文と、オブジェクトの状態やメッセージに関する条件式を組み合わせて用いる。例えば、到達可能性を検査するための定型文には、以下のものが定義されている。なお、括弧内には条件式を記述する。

・いつか「p」となる。
・常に「p」となるときがいつかある。

† 三菱電機(株)　情報技術総合研究所

（2）UML を拡張した出力記法

検出した不具合の出力には、入力記法と同様に UML シーケンス図を用いる。さらに、本記法ではシーケンス図を拡張し、オブジェクトの状態を同一図上に表記する。オブジェクト間通信に加え、オブジェクトの状態を明示することで、オブジェクトの状態が原因となる不具合を判別することが可能となるためである。拡張記法により不具合の判別が可能となる例を次節に示す。

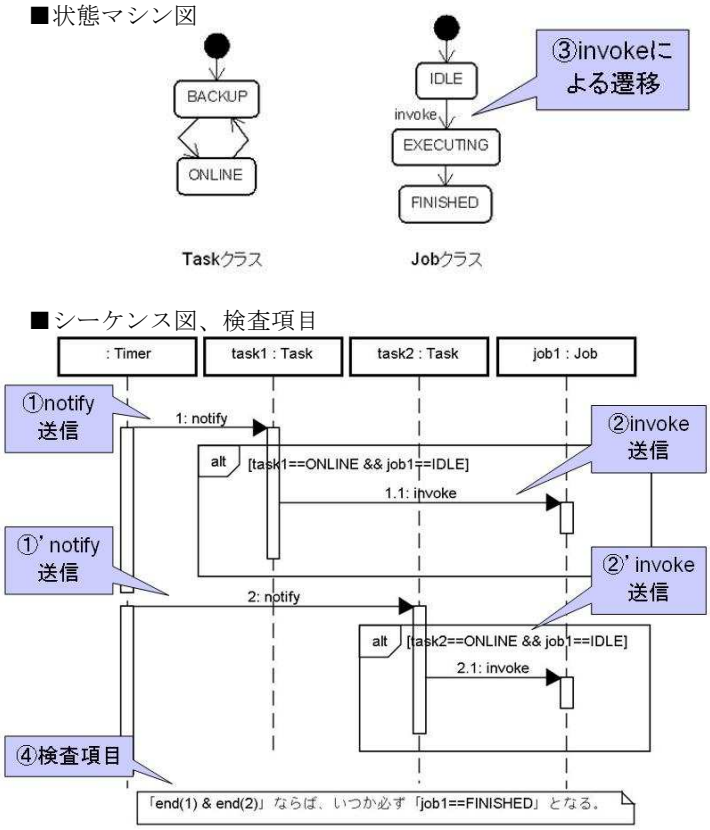

図 2　入力モデルと検査項目

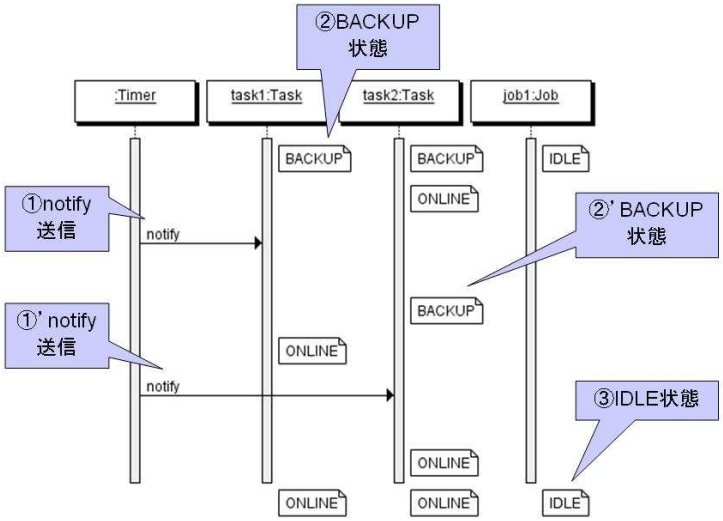

図 3　検査結果

3. タイマーによるジョブ起動システムへの適用

3.1 入力の記述

本ツールの効果を確認するために、開発済システムから動作仕様を抜粋・単純化して適用した。

検証対象は、「分散システムにおけるタイマーによるジョブ起動」の仕様である（図 2）。システムの構成要素は Timer、task1、task2、job1 である。①Timer が task1 と task2 に notify を送信すると、②各 task は ONLINE 状態であれば job1 に invoke を送信する。③job1 は、invoke を受け付けると EXECUTING 状態に遷移する。④検査項目は「各 task2 が notify を受信し実行終了した後ならば、いつか必ず job1 は FINISHED になる」ことである。

なお、検査項目は UML シーケンス図のノートに記述されており、自然言語の定型文と括弧内の条件式を組み合わせて記述される。

3.2 検査結果

本ツールの出力した反例の一つを図 3 に示す。この出力は、①Timer が notify を送信した際、②task1 と task2 がいずれも BACKUP であるため、job1 に invoke を送信せず、③job1 の内部状態が IDLE のまま終了する不具合を示している。本記法によって、状態に起因する不具合の内容と原因を判別することが可能となったことが分かる。

3.3 ツールの評価

本ツールの記法によって、プログラムを用いずに入力モデル・検査項目を記述し、モデル検査を実行することが出来た。また、拡張記法により、オブジェクトの状態が原因となる不具合を判別することが出来た。これらの結果から、固有の記法を用いる従来ツールに比べ、導入の容易な機能を実現できたと考えられる。

4. おわりに

本稿では、UML を用いた記法によりオブジェクト間通信をモデル化し、モデル検査を行うツール DESIMOC について報告した。

現在、我々はいくつかのシステムを適用し本ツールの評価を進めている。現段階では、オルタナティブとアサーションを除く複合フラグメントが未定義であるなど、入出力の記法に改善すべき点が残る。多様なシステムのモデル化に取り組み、不足する記述要素を追加することが今後の課題となる。

参考文献

[1]中島 震: SPIN モデル検査, 近代科学社, 2008.
[2]篠崎 孝一, 太田 弘, 早水 公二, 星野 光勇, 今村 哲典, 吉田 雅昭: モデル検査の実用化課題と支援ソフトウェアの開発, 第三回システム検証の科学技術シンポジウム, 2006.
[3]OMG Unified Modeling LanguageTM (OMG UML), Superstructure, Version 2.2, OMG formal/2009-02-02, Object Management Group ,2009.
[4]Gerard J. Holzmann,“The Model Checker SPIN”, IEEE Trans. Soft. Engin., Vol.23, No.5, pp.279–295, 1997.