モデル検査用記述に基づいたテストケースの生成法

Nguyen Tam Thi Minh(0710023)

北陸先端科学技術大学院大学 情報科学研究科

2009 年 2 月 5 日

キーワード: Spin、Promela、テストケース、W-方法、整合性.

1 はじめに

設計工程の品質を保証するため、モデル検査などを用いて設計モデルの検証を行う研究が行われている。しかし、検証した設計モデルに基づいて実装する際、設計モデルで検証した性質が失われてしまう恐れがある。設計検証の結果を崩さないように実装するためには、設計と実装の整合性をとる必要がある。この整合性を保証する一つの解決策として、設計モデルと実装が整合しているかテストを行う。このテストはテストケースを生成し、それに基づいて行われる。

そこで、本研究は、設計モデルと実装の整合性をテストするため、設計モデルからテストケースを自動生成する手法を提案する。設計モデルはモデル検査により正しさが検証されていることを前提とする。今回は、モデル検査ツール Spin（Simple Promela INterpreter）[2] を使用するので、設計モデルは Promela（PROcess MEta LAnguage）言語で記述されている。よって、Promela 記述からテストケースを自動生成する手法を提案する。

2 関連研究

関連研究として適合テスト方法とテストケース自動生成法について調査を行った。適合テスト方法に関しては、W-方法 [1] を調査した。テストケース自動生成法に関しては、モデルベーステスト（MBT 手法）およびモデル検査の反例に基づいたテストケース生成法を調査した。

W-方法は、オートマトン理論に基づいて仕様と実装の整合性をテストする手法である。W-方法では、仕様と実装の両方が決定性有限オートマトン（FSM）でモデル化されることを前提している。よって、仕様と実装の整合テストとは、２つの FSM の整合性をテストすることを意味する。また、W-方法で使われる FSM には、完全に規定され、ミニマルであり、全ての状態が到達可能な状態であるという前提がある。これらの前提は、W-方法の実用性をを制限する。

MBT 手法は、テスト対象のシステム（System Under Test または SUT）のモデルに基づいて、テストケースを導出する手法である。このモデルは、正しさが保証されていない。また、設計者は、モデルを作成するので、設計者の考えによっては異なるモデルが作成されたり、作業途中で誤りが入る可能性がある。そのため、テスト結果を評価するときに、テスト結果が良くなければ、テストケース生成アルゴリズムだけではなく、モデルも修正する場合がある。

モデル検査の反例に基づいた方法は、変異分析（Mutation Analysis）の適用に基づいて、テストケースを生成する手法である。変異分析は、モデル検査のモデルとモデル検査で検証される性質の不一致を生成する手法である。一つの不一致をモデル検査で処理すると、反例が出力される。この反例は、初期状態から検証される性質に違反する状態までの到達可能状態シーケンスであり、テストケースとして使われる。しかし、モデルにおいて全ての到達可能な状態シーケンスを網羅する反例の集合を生成することは困難である。つまり、モデル検査の反例に基づいた方法に基づいて生成されるテストケースの集合は完全であると言えない。

3 アプローチ

本研究は、W-方法を参考にして、Promela 記述と実装の整合性をテストする枠組みを提案する。すなわち、Promela 記述と実装の両方が有限オートマトン (FSM) に変換され、Promela 記述と実装の整合性をテストすることは、２つの FSM の整合性をテストすることになる。また、Spin で表現される探索機能に基づいて、Promela 記述からテストケースを自動生成する手法を提案する。そのため、Promela 記述と実装を表現する FSM は、Promela 記述に基づいて定義される。

提案した手法を評価するため、キッチンタイマ [3] を題材として小規模な実験を行なう。このキッチンタイマは、Promela による設計モデルが存在し、設計モデルの正しさは Spin によって既に検証されている。

4 提案手法

4.1 Promela 記述と実装の整合性をテストする枠組み

Promela 記述と実装の整合性をテストする枠組みは、３つの重要なステップから成る。

ステップ１. Promela 記述から FSM D に変換する。このステップは、Promela 記述から FSM への変換規則を提供する。また、今回、本研究は内部プロセスが一つしかない Promela 記述のみに着目する。

ステップ２. 実装から FSM M に変換する。本研究は、実装がホワイトボックスであり、内部状態が見えることを前提とする。また、この実装は、Ｃコードで表現される。内

部状態のデータからFSM Mに変換する。実装から唯一のFSMに変換するため、内部状態データの取り出し方について基準化する。

ステップ３．２つのFSM DとMの整合性をテストする。DとMの整合性は、$M = D$を意味する。そこで、本研究は、$I_M = I_D$と$S(M) = S(D)$を前提とし、W-方法を参考にして$M = D$をテストする手法を提案する。ここで、I_MはMの入力記号の集合、I_DはDの入力記号の集合、$S(M)$はMの状態集合、$S(D)$はDの状態集合である。

4.2 Promela記述からのテストケース自動生成

Promelaからのテストケース自動生成のプロセスは、３つのステップを含む。

1) Spinでの探索木の枝にラベル（入力記号、出力記号、次の状態）を付ける。すなわち、Promela記述に（入力記号、出力記号、次の状態）を印刷するCコードを埋め込む。

2) オプション`-DCHECK`を用いて、Cコードを埋め込んだPromela記述をコンパイルする。このコンパイルは、SpinがPromela記述を検証する各ステップを詳しく記述する基準的な出力を印刷する。この出力もSpinが探索木を走査する各ステップを詳しく記述する。なお、このコンパイルの情報は、ファイル`dcheck.out`に保存される。

3) ファイル`dcheck.out`からテストケースを自動生成する。本研究はテストケースを自動生成するプログラム`TestcaseGenerator`を作成した。このプログラムは、ファイル`dcheck.out`を処理し、テストケースの集合を生成する。

5 実験と考察

Promela記述からのテストケース自動生成のプロセスによって、キッチンタイマのテストケースを生成した。キッチンタイマのテストケースは、1155個あった。

そして、キッチンタイマのPromela記述に基づいて、キッチンタイマをCコードで実装した。$I_M = I_D$の前提を成立させるため、キッチンタイマのCプログラムでは、Promela記述で使われる３つのメッセージを３つのイベントとして対応付けた。一方、$S(M) = S(D)$の前提を検討するため、キッチンタイマを３つの実装パターンで実装した。

1) パターン１（$L_{Imp} = L_{Promela}$、$V_{Imp} = V_{Promela}$）は、最も簡単な場合である。このパターンでは、Promela記述と同じように実装する。Promela記述で表現されること以外を実装しないため、$S(M) = S(D)$が簡単に成立する。

2) パターン２（$L_{Imp} > L_{Promela}$、$V_{Imp} = V_{Promela}$）は、実際のキッチンタイマを実装するパターンである。このパターンでは、実装の余分ラベルを表現する文字列を

Promela 記述のラベルと同じように変換すれば、$S(M) = S(D)$ が成立する。また、余分ラベルに関する遷移の入出力記号も、Promela 記述の入出力記号と同じように変換される。

3) パターン３（$L_{Imp} = L_{Promela}$、$V_{Imp} > V_{Promela}$）は、最も複雑な場合である。このパターンでは、Promela の一つの状態は複数の状態で実装される。今回は、隣接した複数の状態について検討した。$S(M) = S(D)$ を成立させるため、特別なマークと表現を定義した。テストするとき、これらのマークと表現を見つけると、これらの対応関係によって、テストを行った。

今回、本研究は以上のパターンについて検討した。それぞれのパターンに対して、本研究は、$S(M) = S(D)$ が成立するための解法を提案した。今後は、ほかの実装パターンを検討する。

6　まとめ

本研究では、設計モデルからテストケースを自動生成する手法を提案し、このテストケースを用いて実装と設計モデルの整合性をテストする枠組みを提案した。また、この設計モデルは、Promela で記述され、正しさが Spin により検証されている。そのため、本研究で提案した手法は、Promela 記述からテストケースを自動生成する手法と言っても良い。また、この手法は、W-方法を参考にして提案された。提案手法を評価するため、キッチンタイマを用いて小規模な実験を行った。

本研究は、ソフトウェアテストにテストケース生成法の一つを提供した。本研究のテストケース生成法は、MBT 手法やW－方法の問題点を解消する。また、本提案手法は、Spin の探索機能を用いて、Promela 記述からテストケースを生成することを自動化している。このモデル検査とテストケース生成法の連結により、提案手法の実用性が高くなると言える。

参考文献

[1] T. Chow, “ Testing software design modeled by finite state machines ”, *IEEE Trans. Software Eng.*, vol. SE-4, pp. 178-187, Mar, 1978.

[2] Holzmann, G. J.(2004). *The SPIN Model Checker: Primer and Reference Manual*. Boston: Addison-Wesley.

[3] 啓樹穴田 (2007, December 3). モデルベース開発とは. 日経エレクトロニクス: 組み込みアカデミー, pp. 133-140.