テクニカルノート

EFSM適合性試験系列生成手法の誤実装検出能力の実験的評価

小原 勝[†,☆] 森 亮憲[†]
樋口昌宏[†] 藤井 護[†]

本稿では拡張有限状態機械（EFSM）モデルの通信プロトコルに対する適合性試験系列の現実的な誤り検出能力の実験的評価について述べる．評価実験に際して，プロトコル仕様を正しく実装したCプログラムからC言語上のフォールトを1個または2個含む実装を生成し，その実装が試験系列を正しく実行するかを観察するシステムを開発した．2つの例プロトコルを対象に，これまでに提案された各種試験系列生成手法による試験系列の誤り検出能力を評価した結果，文献4)の手法は他の手法より誤り検出能力が高いことが確認できた．

A Fault Coverage Evaluation of Test Suite Generation Methods for EFSM

Masaru Kohara,[†,☆] Takanori Mori,[†] Masahiro Higuchi[†] and Mamoru Fujii[†]

In 4), a test suite generation method for communication protocols modeled as Finite-State Machine with Counters, a subclass of Extended Finite-State Machine (EFSM), is proposed. In this paper, a practical evaluation of fault detection power of test suites generated by the method and several other test generation methods. For the evaluation, we developed a mutation system which derives implementations with single or double faults from correct protocol implementation in C program. We examined whether the test suites can detect faults in the derived mutations. The experiment has shown that the fault coverage of the method proposed in 4) is superior to other methods.

1. まえがき

拡張有限状態機械（EFSM）モデルで定義された通信プロトコルに対する適合性試験系列の自動生成手法としてE-UIO法[1)]，ATSG法[2)]，EFTG法[3)]が知られている．これに対し，筆者らはEFSMの部分クラスであるカウンタつき有限状態機械（FSM-C）の遷移条件およびレジスタ代入の正しさを詳細に試験するための系列の生成手法を提案し，実装が仕様と同じモデルとみなせるという仮定のもとで，あらゆる単一誤りを検出可能であることを示した[4)]．ところが，実際の通信プロトコルは順序回路やソフトウェアといった，より能力の高い計算モデルで実現されるのが普通であり，そのような場合，実装モデルでのどのような誤りをどの程度検出できるのかに関する評価は十分でなかった．そこで，提案手法の現実的な誤実装検出能力を評価するための実験を行った．実験では，プロトコルはC言語で実装されるものとし，プロトコルを正しく実装したC言語のソースプログラムから1つまたは2つのフォールトを含んだ実装（ミューテーション）を順次生成するミューテーションシステムを作成した．生成されたすべてのミューテーションに文献4)の手法に基づく試験系列，および従来の手法に基づく試験系列をそれぞれ適用し，誤実装を検出するかを調べた．

2. 準　備

プロトコル機械はFSM-Cとして定義されるものを考える．FSM-Cは，有限状態機械（FSM）に，代入，整定数の加減算，大小比較の機能を持つ整数値レジスタを加えたものであり，1回の入力（出力）で，コマンドと整数値パラメータの組を入力（出力）する．

† 大阪大学大学院基礎工学研究科
Graduate School of Engineering Science, Osaka University
☆ 現在，松下電器産業株式会社
Presently with Matsushita Electric Industrial Co., Ltd.

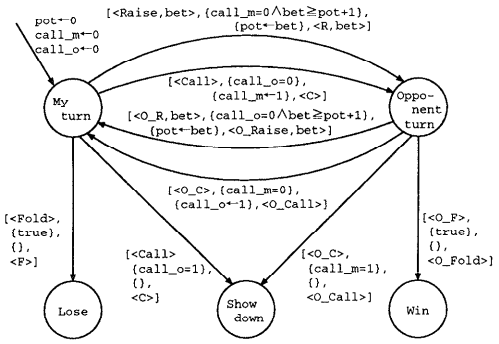

図1 FSM-Cの例
Fig. 1 An example of FSM-C.

図1に2者によるポーカーゲームで賭けの額（pot）を決めるプロトコルを定義した例を示す．行える操作はRaise（せりあげ），Call（同意），Fold（おり）の3つで，一度Callすると，その後Raiseはできない．図では有限制御部の状態を頂点で，アクションを有向辺で表している．ラベルはそのアクションで実行する動作を[⟨入力コマンド，パラメータ⟩，遷移条件，レジスタ代入式，⟨出力コマンド，パラメータ⟩]という形式で表している．プロトコル機械に入力が与えられると，そのコマンドがアクションの入力コマンドの定義に一致し，各レジスタ値および入力パラメータ値がそのアクションの遷移条件式を満たしている場合，そのアクションを実行する．たとえば，状態My turnで，入力⟨raise, bet⟩を受けた際，自身がまだCallしておらず（call_m = 0），入力パラメータbetの値が現在のせり値potより大きい（bet ≥ pot + 1）とき，potの値をbetにし，せり上げを表すメッセージ⟨R,bet⟩を相手に送り，相手の操作を待つ状態（Opponent-turn）へ遷移することが定義されている．

FSM-Cでは，遷移条件は連立差分不等式で表される．また，レジスタに代入される値，および出力定義式中の出力パラメータ値は$x+c$の形で表されるものに限る．FSM-Cにその状態からの動作が定義されていない入力が行われると，FSM-Cはエラーメッセージを出力して初期状態に遷移する（エラーアクション）．FSM-Cの状態遷移はエラーアクションによる遷移も含めると完全かつ決定性で，その実行時間は有限とする．このような制限のもとでも，シーケンス番号やタイマ制御を用いるプロトコルを簡潔に定義できる．

3. ミューテーションシステム

プロトコル仕様のC言語による実装を考えた場合，たとえば複数のアクションに共通する処理をまとめて記述することも可能であり，その場合ソースプログラム上での単一の誤りが，FSM-Cとして見た場合，多重誤りに相当する．ほかに，制御フローに関する単一誤りもFSM-Cとして多重誤りに相当する場合があり，また，誤りによってFSM-Cでは許されない演算（乗算など）を含んでしまい，IUTがFSM-Cとみなせなくなってしまう場合もありうる．試験系列の現実的な能力を論ずるうえで，これらのフォールトをどの程度検出しうるかを評価することは重要である．

以上の観点から，試験系列の現実的な誤実装検出能力を評価するため，文献5)を基にC言語による実装の際のフォールトモデルを設定した．プロトコルを正しく実装したC言語のソースプログラムから1つまたは2つのフォールトを含んだ実装（ミュータント）を順次生成するミューテーションシステムを作成した．

文献5)ではC言語でのプログラミングの際に起こるバグについて統計をとり，詳細に分類している．このうち，構文誤りなどコンパイラで検出可能なフォールトを除き，以下のフォールトモデルを設定した．

(1) 制御フローの誤り：Cプログラムにおける制御の流れを誤らせる以下のような誤り．これらの誤りはFSM-Cでは多重誤りに相当したり，IUTをFSM-Cとみなせないものとする可能性がある．
- **ループでの誤り**
  - **初期値の誤り**：for文の初期値の誤り．
  - **終端値の誤り**：for文の終了条件を指定する定数の誤り．
  - **ループ終了例外処理の誤り**：ループ中で使われる「break」を「continue」と誤ったもの，またはその逆．
- **述語不良**：if文などでの述語記号（不等号の向き）の誤り．
- **パス不良（括弧の位置の誤り）**：括弧「{」，「}」の位置の誤りのうち，構文誤りにならないもの．

(2) 領域の誤り：変数の領域を表す式における誤り．プログラム中の位置によっては，FSM-Cとして多重誤りに相当する場合がある．
- **境界値の誤り**：境界を表す値の誤り（例：「`x<0`」⇒「`x<1`」）．
- **開閉条件の誤り**：不等号の開閉条件を誤ったもの（例：「`x>=0`」⇒「`x>0`」）．
- **重複指定**：論理演算記号の誤りによるもの（例：「`2<=x && x<=4`」⇒「`2<=x||x<=4`」）．

(3) 場合分けの不完全性：switch文などで場合分け

が不完全である誤り．

- **場合分けの洩れ**
- **場合分けの重複**

(4) 式の評価の誤り：演算子の誤り．IUT を FSM-C とみなせないものとする可能性がある．

- **四則演算の誤り**：加算記号「+」を乗算記号「*」とするなど誤ったもの．
- **負号の誤り**：必要な「−」が抜けている誤り．

(5) データアクセスの誤り：誤ったオブジェクトアクセス．制御変数に関する誤りは，FSM-C として多重誤りに相当する場合がある．

- 変数名を書き誤ったもの．C プログラム中のすべての変数を対象とする．

定数値を誤ったフォールトは原理的には無限に存在する．そこで，今回の実験では定数値の誤りとして，正しい値より 1 大きく誤ったもの，および 1 小さく誤ったもののみを用いた．これらの 2 つのフォールトはより誤り幅が大きい場合と比較して試験の際に誤りを見逃してしまう場合が多いと考えられる．

4. 例プロトコル

実験評価には以下の 2 つの例プロトコルを用いた．

(1) 文献 3) で用いられているプロトコル コネクションの確立，データ転送，コネクションの解放を行う単純なデータ転送プロトコルで，プロトコル機械の状態数は 8，レジスタ数は 4，状態遷移の数は 20 である．

(2) OSI セションプロトコル[6] OSI セションプロトコルのカーネル機能単位，全二重機能単位，大同期機能単位，小同期機能単位，再同期機能単位，折衝解放機能単位の 6 つの機能単位を選択したもの．プロトコル機械の状態数は 19，レジスタ数は 12，状態遷移の数は 126 である．

評価の対象とした各手法の試験項目と，例プロトコルに対する，各手法に基づく試験系列の総系列長を**表 1** に示す．試験項目に関する表では，それぞれのタイプの誤りについて×は試験項目としないこと，△は試験項目として含むこと，○はさらに，すべての単一誤りを検出することが保証されていることを表している．また試験系列の総系列長とは，それぞれの試験項目に対する試験系列（試験ケース）の試験系列長の総和を表している．

文献 4) の手法は他の手法と比較して試験項目が多いため，系列長がかなり長くなっている．

表 1 各手法の試験項目と試験系列長
Table 1 Test items of test generation methods and their test suite length.

	E-UIO	ATSG	EFTG	文献 4)
試験項目				
状態遷移先	○	○	△	○
レジスタ代入	×	△	×	○
遷移条件	△	△	△	○
総系列長				
(1)	274	114	295	757
(2)	1349	570	生成不能	9783

5. 実験結果

生成した各ミューテーションにそれぞれの試験系列を適用し，ミューテーションに含まれたフォールトを検出できるかどうかを調べた．

5.1 単一フォールトの場合

5.1.1 例プロトコル (1) の実験結果

実験結果を**表 2** に示す．文献 4) の試験系列はすべてのミューテーションのフォールトを検出できたのに対し，他の 3 つの手法による試験系列はいくつかのミューテーションのフォールトを検出できなかった．なお，この実験ではミューテーションシステムはほかに制御フロー，データアクセスに関するフォールトを 1 つ含むミューテーションをそれぞれ，9 個，2 個生成したが，いずれの試験系列も実装誤りと判定しなかった．個々のミューテーションについて調べたところ，プロトコル機械として正しい実装と等価な動作をするプログラムとなっていることが簡単に確認できた．

5.1.2 例プロトコル (2) の実験結果

実験結果を**表 3** に示す．この場合も，文献 4) の試験系列はすべてのミューテーションのフォールトを検出できたが，ATSG 法，E-UIO 法では遷移条件式やレジスタ代入式にかかわるフォールトの多くを検出できなかった．この例でも表 3 に記したもののほか，制御フロー，データアクセスに関するフォールトを 1 つ含むそれぞれ，20 個，6 個のミューテーションについては，いずれの試験系列でも実装誤りと判定しなかったが，それぞれプロトコル機械として正しい実装と等価な動作をすることを確認した．

表 2 検出できなかったフォールト数（単一フォールト，例プロトコル 1）
Table 2 The number of undetected faults (single fault, protocol 1).

	生成数	E-UIO	ATSG	EFTG	文献 4)
制御フロー	29	0	0	2	0
領域	9	0	0	2	0
場合分け	25	0	0	0	0
式の評価	10	0	0	1	0
データアクセス	96	3	3	9	0

表3 検出できなかったフォールト数
(単一フォールト，例プロトコル2)
Table 3 The number of undetected faults (single fault, protocol 2).

	生成数	E-UIO	ATSG	文献4)
制御フロー	135	11	11	0
領域	192	11	11	0
場合分け	70	0	0	0
式の評価	98	11	13	0
データアクセス	354	39	41	0

表4 検出できなかったフォールト数
(二重フォールト，例プロトコル1)
Table 4 The number of undetected faults (double fault, protocol 1).

生成数	E-UIO	ATSG	EFTG	文献4)
9926	34	163	34	3

表5 検出できなかったフォールト数
(二重フォールト，例プロトコル2)
Table 5 The number of undetected faults (double fault, protocol 2).

生成数	E-UIO	ATSG	文献4)
9991	116	102	0

5.2 二重フォールトの場合

単一フォールトとして生成したフォールトからランダムに2つのフォールトを選ぶ方法で，10,000個の2つのフォールトを持つミューテーションを生成した．生成したミューテーション中，例プロトコル(1)で74個，例プロトコル(2)で9個，のミューテーションについてはいずれの試験系列でも実装誤りと判定しなかったが，それぞれプロトコル機械として正しい実装と等価な動作をすることが確認できた．

5.2.1 例プロトコル(1)の実験結果

実験結果を**表4**に示す．文献4)の試験系列で検出できなかった3個のミューテーションのうち2個は，1つの差分不等式中に演算子「−」を「∗」と誤ったフォールトを含む2つのフォールトが存在する場合であった．残りの1個は2つのアクションに1つずつフォールトを含んでいるもので，それぞれのフォールトを検出すべき試験系列として他方のアクションの実行を含むものを選んでいるため，2つのフォールトが相殺しあうことにより選ばれた試験系列を正しく実行していた．

5.2.2 例プロトコル(2)での実験結果

実験結果を**表5**に示す．このプロトコルでは，ミューテーションに含まれるフォールトが2つになっても文献4)の手法は正しい実装と等価でないすべてのミューテーションのフォールトを検出できた．

6. あとがき

今回の実験で，文献4)の手法に基づく試験系列は他の試験系列と比較して，高い誤り検出能力を持つことが確認できた．この手法は，IUTもFSM-Cとみなせると仮定した場合に十分な誤実装検出能力を持つように，フォールトの検出範囲を広くとり，細かく試験を行っている．このため，現実の実装の際に生じる多くのフォールトが検出可能になっていると考えられる．

今後の課題としては，ハードウェアによる実装を考慮し，VHDLなどの高レベル記述言語におけるフォールトモデルに基づく誤実装検出能力を評価するなど，より多角的に試験系列の能力を評価することなどが考えられる．

参考文献

1) 李 湘東，東野輝夫，樋口昌宏，谷口健一：拡張有限状態機械モデルで書かれた通信プロトコルの適合性試験系列の自動生成の一手法，電子情報通信学会論文誌（B-I），Vol.J79-B-I, No.4, pp.137–147 (1996).
2) Wang, C.J. and Liu, M.T.: Axiomatic Test Sequence Generation for Extended Finite State Machines, *Proc 12th Int'l Conf. on Distributed Computing Systems*, pp.252–259 (1992).
3) Bourfir, C., Dssouli, R., Aboulhamid, E. and Rico, N.: Automatic executable test case generation for extended finite state machine protocols, *Proc International Workshop on Testing Communication Systems*, pp.75–90 (1997).
4) 樋口昌宏，小原 勝，中石敬治，藤井 護：通信プロトコル適合性試験におけるレジスタ操作に対する試験系列の生成手法，情報処理学会論文誌，Vol.39, No.4, pp.1067–1076 (1998).
5) ボリス．B：ソフトウェアテスト技法，小野間彰，山浦恒央（訳），日経BP出版センター．
6) ISO: Information Processing System − Open Systems Interconnection − Basic Connection Oriented Session Protocol Specification, IS 8327 (1987).

(平成11年1月5日受付)
(平成11年5月7日採録)