マルチメディア通信と分散処理 86－5
グループウェア 26－5
（1998. 1. 29）

EFSMに対する適合性試験系列生成手法のミューテーションシステムを用いた実験評価

小原 勝 樋口 昌宏 藤井 護

大阪大学大学院基礎工学研究科

e-mail: {kohara,higuchi,fujii}@ics.es.osaka-u.ac.jp

我々の研究グループではEFSM（拡張有限状態機械）で記述された通信プロトコルに対する適合性試験系列生成手法を提案している．我々の手法を用いるとFSM-Cという限定されたモデルの下でのあらゆる単一誤りを検出することができる．本研究では我々の手法の，より一般的なフォールトモデルの下でのフォールト検出能力を定量的に評価し，従来の手法と比較するために実験を行った．今回の実験ではプロトコルはC言語を用いて実装されるものとし，統計的分類に基づくCプログラム上のフォールトモデルを用いた．フォールトを一つ含む実装（ミューテーション）を順次生成するシステムを作成し，生成されたミューテーションのフォールトを試験系列が検出できるかどうかを観察した．その結果，我々の手法は従来の手法と比較して非常に高い精度でフォールトを検出できることが確認できた．

Fault Coverage Evaluation of Test Cases Generation Method for EFSM using a Mutation System

Masaru Kohara Masahiro Higuchi Mamoru Fujii

Graduate School of Engineering Science,
Osaka University

We have proposed a test cases generation method for communication protocols modeled as Extended Finite-State Machine(EFSM). Our method can detect any single fault of protocol implementations under a restricted model called FSM-C. In this research, we evaluated fault coverage of our method and the existing methods under a general fault model. We supposed that protocols are implemented by C programs, and used a fault model based on a statistical classification. We implemented a mutation system, which generates implementations which include one fault (mutation). We examined whether the methods can detect faults in the mutations. The experiment has shown that the fault coverage of our method is superior to other methods.

1 まえがき

通信システムの開発において，作成されたシステムがプロトコル仕様通りに動作するかどうかの試験（適合性試験）は重要である．プロトコル適合性試験は試験対象となるシステム（IUT: Implementation Under Test）に試験系列を入力し，その際の出力が仕様通りであるかどうかを観察することによって行われる．適合性試験に用いる試験系列の生成は従来人手により行われていたが，試験効率を上げるためには試験系列を自動で生成する手法を確立する必要がある．

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試験系列の自動生成

手法はこれまでにいくつか提案されてきた．E-UIO 法 [1] を用いると状態の存在及び状態遷移の正しさについて試験する系列を生成することができる．また，ATSG 法 [2] を用いるとこれらに加え，一部のレジスタ操作の正しさを試験する系列を生成することができる．しかし，ATSG 法に基づく試験系列でもレジスタ操作のうち遷移条件の判定の正しさを試験することはできず，さらにレジスタ代入操作についても代入された値が直接または他のレジスタを介して出力に現れない場合には，その正しさを試験することはできない．

我々の研究グループでは EFSM の部分クラスであるカウンタつき有限状態機械（FSM-C）に対して，従来の方法では試験できなかった遷移条件及びレジスタ代入に対する試験系列の生成手法を提案してきた[3]．また，この提案手法に基づく試験系列生成システムを実装し，実際に我々の手法に基づく試験系列が生成できることを確認した．

しかし，これらの手法では実装されたプロトコルも仕様と同じモデルとみなすなどの仮定をおいているため，現実的な有効性は明らかではない．そこで，これらの手法の現実的なフォールト検出能力を定量的に把握し，従来の手法と比較するため実験を行った．本稿ではこの実験の内容とその結果について述べる．今回の実験では，プロトコルは C 言語で実装されるものとした．実験ではまずミューテーションシステムを作成した．ミューテーションシステムはプロトコルを正しく実装した C 言語のソースからフォールトを含んだプロトコル（ミューテーション）を順次生成するシステムである．ミューテーションには文献 [6] で調査された実際の実装の際起こるフォールトを一つずつ含ませた．ミューテーションシステムによって生成されたすべてのミューテーションに対して我々の手法に基づく試験系列，及び従来の手法に基づく試験系列をそれぞれ適用し，フォールトを検出できるかどうかを観察した．また，今回の実験では EFTG 法[4]も比較の対象とした．EFTG 法はデータフローと制御フローに着目して，実行可能な試験系列を生成する手法である．

以降，2. では我々の手法が対象としているプロトコルモデルと試験の形式について，3. では我々の試験系列生成手法について，4. ではミューテーションシステムを用いた実験と評価について述べる．

2 準備

2.1 プロトコル機械モデル

本稿で対象とする通信プロトコルは 図 1 で示す通り，入出力用のゲートを介して通信を行う．入力及び出力はそれぞれコマンドと整数値の組とする．

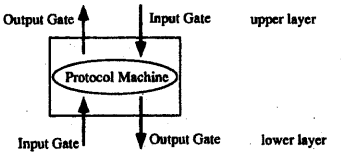

図 1: 入出力ゲートを用いた通信

また，プロトコル機械は FSM-C として定義されるものとする．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プロトコル機械モデルである．FSM-C では，入力，条件判定，出力，レジスタ代入を実行できる．図 2 は FSM-C の例である．

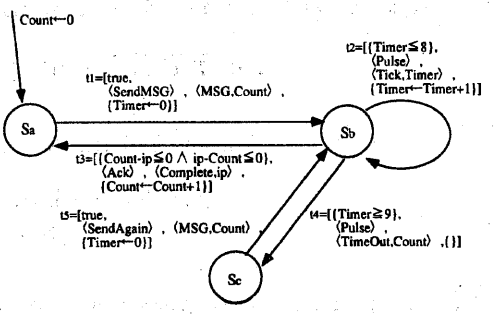

図 2: FSM-C モデルの例

図 2では，有限制御部の状態を頂点で，状態遷移を有向辺で表している．ラベルはその状態遷移で実行するアクションを [{ 遷移条件 },⟨ 入力コマンド ⟩,⟨ 出力定義式 ⟩,{ レジスタ代入式 }] という形式で表している．また，遷移条件やレジスタ代入式で参照される入力パラメータは ip で表している．プロトコル機械に入力が与えられると，その時の各レジスタ値及び入力パラメータ値が遷移条件式を満たしており，入力コマンドがアクションの定義を満たしている場合，そのアクションを実行する．アクションの実行に

おいては，レジスタ値の更新，有限制御部の状態の遷移，出力がそれぞれ行われる．

FSM-C では，遷移条件式は連立差分不等式で表されているものとする．ここで，差分不等式とは $x-y \leq c$ の形で表される式である．また，出力値及びレジスタに代入される値は $x+c$ あるいは 単に c と表されるものに限られる．以上において，x, y はレジスタあるいは入力パラメータを表し，c は整定数を表している．

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なわれると，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．このアクションをエラーアクションと呼ぶ．本稿で試験対象とする 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であるものとする．

上記のような制約の下でも，一連番号やタイムアウト制御を行なうプロトコルを簡潔に定義できる．

2.2 適合性試験の形式

通常，IUT の内部状態は外部から観察できない．そこで，適合性試験はブラックボックス試験という方法で行なわれる．ブラックボックス試験は，IUT に入力系列を与え，その出力が仕様通りのものであるかを調べることによって行なわれる．このため，適合性試験においては試験系列が重要な役割を果たす．

EFSM に対する試験系列の一つとして状態を同定する E-UIO 系列[1] がある．これは FSM において一つの状態を同定するための UIO 系列を拡張したものである．EFSM において一つの状態を識別するためには，その状態に対する UIO 系列の前にその UIO 系列を実行可能とするようなレジスタ値を設定する先行系列を連接しなければならない．UIO 系列にこのような先行系列を連接した系列が E-UIO 系列である．我々が提案している手法では，系列を実行した後の有限制御部の状態が仕様通りの状態であることを確認する必要があるため，この E-UIO 系列を併せて用いる．

3 試験系列生成手法

我々の手法[3]を用いると，IUT が FSM-C とみなすことができるとき，状態遷移先，遷移条件の判定，レジスタ代入におけるレジスタ参照，定数割り当てに関する単一誤りを検出できる．ここでは，その手法によって生成される試験系列の満たす性質について簡単に述べる．

この手法においては，プロトコル仕様は下記の条件 1 を満たし，IUT は FSM-C とみなすことができるものとする．

[条件 1] エラーアクション以外の任意の二つのアクション $t_i, t_j (i \neq j)$ について，その終状態及び出力コマンドが共に同じものはない． □

FSM-C では，レジスタ操作として遷移条件の判定とレジスタ代入があり，それぞれを試験する系列を別々に生成する．

3.1 遷移条件式に対する試験系列

あるアクション t の遷移条件が $d_1 \wedge d_2 \wedge \cdots \wedge d_k$（$d_i$ は一つの差分不等式）であるとき，$C_t = \{d_1, d_2, \cdots, d_k\}$ と表すものとする．アクション定義中の始状態が s で入力コマンドが CMD であるアクションの集合を $T_{s,CMD}$ とし，$C_{s,CMD} = \bigcup_{t_j \in T_{s,CMD}} C_{t_j}$ とおく．このとき，すべての $d_i (\in C_{s,CMD})$ の判定が正しく実装されていることを示すことによって，すべての $C_{t_j} (t_j \in T_{s,CMD})$ が正しく実装されていることを示すことができる．

以下では試験対象を $C_{s,CMD}$ に含まれる不等式 $(d_i : x-y \leq c)$ であるとする．また，一般性を失うことなく，$d_i \in C_a$ なるエラーアクションでないアクション a が存在するものとする．

3.1.1 定数 c が正しいことを試験する系列

遷移条件式 $d_i : x-y \leq c$ の定数 c の正しさを試験するためには次の二つの系列を用いる．一つは c より小さな値 c' を用いて $x-y \leq c'$ と実装されている場合に実行できない入出力系列（IOC_G）で，もう一つは c より大きな値 c'' を用いて $x-y \leq c''$ と実装されている場合に実行できない入出力系列（IOC_L）である．IOC_G, IOC_L には，以下の条件を満たすあるアクション a に対して，次のような系列を用いる．

- a の始状態は s.
- a の遷移条件は不等式 $d_i : x-y \leq c$ を含む.
- a はエラーアクションではない.

(IOC_G) 初期状態から実行可能な入出力系列であって，その最後の遷移は a によるものである．また，最後の遷移を実行する前の状態において，$x-y=c$ が成り立つ．

(IOC_L)　初期状態から実行可能な入出力系列であって，その最後の遷移はa以外のアクションによるものである．また，最後の遷移を実行する前の状態において，aの遷移条件式のうちのd_i以外の各遷移条件式および，$x-y=c+1$が成り立つ．

さらに，IOC_G，IOC_Lが正しく実行されたことを出力から確認できない場合には，それぞれの終状態を同定する系列（E-UIO 系列）を連接することにより，$IOCU_G$，$IOCU_L$を作成する．

上記の条件を満たす入出力系列$IOCU_G$，$IOCU_L$は必ずしも存在するとは限らないが，存在する場合には図2の形のグラフに対して状態sから有向辺を逆向きに幅優先探索を行なうことにより最も系列長の短いものを見つける．

このような系列$IOCU_G$，$IOCU_L$を用いることにより，差分不等式d_iの定数cが正しく実装されていることを試験することができる．

3.1.2　レジスタが正しいことを試験する系列

条件式$d_i: x-y \leq c$の判定において，レジスタもしくは入力パラメータxが正しく参照されていることを試験するためには，xと異なるレジスタまたは入力パラメータ（以下ではzとする）について，3.1.1と同様の条件を満たすあるアクションaに対して，下記の2つの入出力系列のいずれかを用いる．

($IOC_{z,T}$)　初期状態から実行可能な入出力系列で最後のアクションが実行される前の有限制御部の状態はs．遷移条件が仕様通りに実装されている場合には条件式d_iをtrueと判定しaを実行するが，zを用いて$z-y \leq c$と誤って実装されている場合にはd_iをfalseと判定しa以外のアクションを実行する．

($IOC_{z,F}$)　初期状態から実行可能な入出力系列で最後のアクションが実行される前の有限制御部の状態はs．遷移条件が仕様通りに実装されている場合には条件式d_iをfalseと判定し，a以外のアクションを実行するが，zを用いて$z-y \leq c$と誤って実装されている場合にはd_iをtrueと判定しaを実行する入出力系列．

さらに，3.1.1と同様に必要に応じて$IOC_{z,T}$実行後の状態を同定する系列を連接することにより$IOCU_{z,T}$を作成する．$IOCU_{z,T}$と同様に$IOCU_{z,F}$も作成できる．

このような入出力系列$IOCU_{z,T}$，$IOCU_{z,F}$のどちらかを用いることにより，レジスタxが正しく参照されていることが試験できる．レジスタyについても同様にして試験することができる．

3.2　レジスタ代入式に対する試験系列

我々の手法では，アクションのレジスタ代入の正しさを試験するために，試験対象となるレジスタ代入で正しい値がレジスタに代入されていなければ実行できない系列を試験系列として用いる．各レジスタ代入操作に対して次のような二つの試験系列を作成する．ここでは，遷移tのレジスタ代入式$r \leftarrow x+c$を試験対象とする．

($IO_{TR} \cdot IO_{TG}$)　試験対象のアクションtでレジスタrに正しい値$(x+c)$より小さな値が代入された場合には実行できない入出力系列．

($IO_{TR} \cdot IO_{TL}$)　試験対象のアクションtでレジスタrに正しい値$(x+c)$より大きな値が代入された場合には実行できない入出力系列．

IO_{TR}は2つの系列に共通な系列である．また，各系列は，IO_{TR}の最後で実行される遷移t以外には遷移tを実行しないものとする．

さらに，3.1.1 と同様に必要に応じて$IO_{TR} \cdot IO_{TG}$，$IO_{TR} \cdot IO_{TL}$実行後の状態を同定する系列を連接することによりIOU_G,IOU_Lを作成する．IOU_G,IOU_Lを試験系列として用いることによって，レジスタrに仕様通りの値が代入されていることを試験できる．

しかし，この系列を一組適用するだけでは，アクションtの試験対象の代入操作$r \leftarrow x+c$が$r \leftarrow x'+c$（x'とxは異なるレジスタ）と実現されているような誤ったIUTであっても，試験系列を入力，実行した際に，IO_{TR}の最後の入力値を受けとった時点で$x=x'$となっていると，その誤りを検出できない．そこで，このような場合，最後の入力値を受けとった時点で$x \neq x'$となるような別のIO_{TR}を用いてIOU_G,IOU_Lを作成し，これらを併せて用いることによりこのような誤りを検出する．

FSM-C では出力される値は$x+c$の形で表される．この出力定義が正しく実装されていることも，ここで述べたのと同様の試験系列で試験できる．

4 ミューテーションシステムを用いた実験

通信プロトコルはハードウェア回路あるいはC言語などのソフトウェアで実装される．これらは，計算モデルとしてFSM-Cより一般的なものであり，実装の際に含まれうるフォールトも多様なものとなる．このため，仕様はFSM-Cモデルで記述されていても，実装の際にフォールトが含まれることにより，IUTがFSM-Cの枠を越えたり，実装モデルでの一つのフォールトがFSM-Cとしての複数のフォールトに相当したりする場合がある．我々の手法を含め多くの適合性試験系列生成手法では，そのフォールト検出能力を議論する際，実装されたプロトコルも仕様と同じモデルであるとの仮定をおいているが，上記の理由により妥当な仮定とはいいがたい．より実際的なフォールト検出能力を議論するには，実際のプロトコルの実装の際に起こるフォールトモデルを設定し，そのモデル上でのフォールト検出能力を調べる必要がある．

4.1 フォールトモデル

本研究ではプロトコルはC言語を用いて実装されるものとした．文献[6]ではC言語でのプログラミングの際に起こるすべてのバグについて詳細に分類している．本研究では，文献[6]の分類に基づき，コンパイラで検出可能なフォールトなど，試験系列を適用するまでもなく検出可能なフォールトを除外した以下のフォールトモデルを設定した．

制御フローの誤り

- if文の条件式の誤り．(例) if $(x < y)$ {} ⇒ if $(x > y)$ {}
- for文の初期値や終了条件などの誤り．(例) for $(i = 0; i <= 10; i++)$ ⇒ for $(i = 1; i <= 10; i++)$

領域の誤り

- 境界を定義する値の誤り．(例) $x < 0$ ⇒ $x < 1$
- 不等号の開閉条件の誤り．(例) $x <= 0$ ⇒ $x < 0$

場合分けの完全性

case文などにおいて場合分けの洩れがあるもの．

式の評価

- 演算記号の誤り．
- 符号の誤り．

データアクセスの誤り

変数の書き誤りなどのデータアクセスの誤り．

4.2 ミューテーションシステム

フォールト検出能力の評価にあたって，ミューテーションシステムを作成した[7]．ミューテーションシステムは仕様を正しく実装したCプログラムからフォールトを一つ含むCプログラム(ミューテーション)を順次生成する．ミューテーションに含ませるフォールトモデルは4.1で設定したものである．

4.3 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[4]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なう単純なデータ転送プロトコルで，状態数は8，レジスタ数は4，状態遷移の数は20である．送信するデータに対する一連番号などを保持するためにレジスタが用いられる．(2) OSIセションプロトコル[5]のうちカーネル機能単位，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，衝突解放機能単位を選択したものである．状態数は19，レジスタ数は12，状態遷移の数は126である．一連番号や同期点の設定にレジスタを用いる．

各手法に基づく試験系列の総系列長を表1に示す．

表1: 試験系列の総系列長

	E-UIO	ATSG	EFTG	我々の手法
(1)	274	114	295	757
(2)	1349	570	-	9783

EFTG法は試験系列を生成する際，事前に入力値を決定する方法を取っている．(2)のOSIセションプロトコルでは，入力値を頻繁に与え，その入力値でプロトコルの動作が決定するため，この手法では試験系列の生成ができなかった．

試験系列の総系列長を比較すると，ATSG 法では比較的短いのに対し，我々の手法では非常に長いものとなってしまった．

4.4 実験結果

これらのプロトコルを用いて，E-UIO 法，ATSG 法，EFTG 法，我々の手法のフォールト検出能力を調べた．実験では，生成された各ミューテーションにそれぞれの手法に基づく試験系列を適用し，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観察した．その結果を表 2，3に示す．表中の数字は検出できなかったフォールトの数を表しており，括弧内の数字は生成したミューテーションの数を表している．

表 2: (1) のプロトコルへの適用結果

	E-UIO	ATSG	EFTG	我々の手法
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

この例を用いると，我々の手法ではすべてのフォールトを検出できたのに対し，他の３つの手法ではいくつかのフォールトを検出できず，特に EFTG 法ではその数が多かった．

表 3: (2) のプロトコルへの適用結果

	E-UIO	ATSG	我々の手法
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

(2) のプロトコルを用いても，我々の手法はすべて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関わるフォールトの多くを検出できなかった．

ATSG 法は E-UIO 法で試験できる項目に加えて，レジスタ代入式に対しても，代入結果が出力として現れる場合にはその正しさを試験できる手法である．しかし，(2) のプロトコルではすべてのレジスタが状態遷移の制御に用いられるのみで出力には現れず，レジスタ代入式に対する試験系列は生成できなかったため，E-UIO 法と ATSG 法では検出できないフォールト数に差は生じなかった．

5 まとめ

本研究では我々の提案している EFSM に対する試験系列生成手法のフォールト検出能力を評価するために，ミューテーションシステムを用いた実験を行った．この研究では統計的分類に基づく C プログラム上のフォールトモデルを用いて評価を行った．この結果，我々の手法が従来の手法と比較してフォールト検出能力が高いことが確認できた．

しかし，我々の手法に基づく試験系列の総系列長は，従来の手法と比較して非常に長いものであった．そこで，フォールト検出能力を保ったまま総試験系列を短くすることが今後の課題である．

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] C.J.Wang and M.T.Liu: Axiomatic Test Sequence Generation for Extended Finite State Machines, *Proc 12th Int'l Conf. on Distributed Computing Systems*, pp. 252–259 (1992).

[3] 中石敬治，樋口昌宏，藤井護: あるクラスの拡張有限状態機械におけるレジスタ操作の試験系列の生成手法，情報処理学会マルチメディア通信と分散処理研究会 95-DPS-70-10 (1995).

[4] C.Bourfir, R.Dssouli, E.Aboulhamid, N,Rico: Automatic executable test case generation for extended finite state machine protocols, *Proc International Workshop on Testing Communication Systems*, pp. 75–90 (1997).

[5] ISO: “Information Processing System - Open Systems Interconnection - Basic Connection Oriented Session Protocol Specification”,IS 8327 ,(1987).

[6] Boris Beizer: 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

[7] Masaru Kohara , Masahiro Higuchi and Mamoru Fujii: “A Test Case Generation Method for FSM with Counters and its Fault Coverage Evaluation Using Mutant Generator”, in Proc. 1997 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, pp.555-559, Aug, 1997.