マルチメディア通信と分散処理　91－19
（1999．1．22）

EFSM適合性試験系列自動生成における系列長短縮化について

別府 明　　樋口 昌宏　　藤井 護

大阪大学大学院基礎工学研究科

本論文では拡張有限状態機械（EFSM）モデル通信プロトコルのための適合性試験スイートの短縮化について議論する．我々の研究グループで提案しているEFSMのサブクラスであるカウンタ付き有限状態機械（FSM-C）に対する適合性試験系列生成手法では，すべての単一誤りを検出可能であるが，各試験項目で独立に試験系列を生成しているため試験スイートが長くなり試験の実施効率上で問題があった．そこで本論文では，複数の試験項目に共通して用いることができる試験系列を生成する手続きを提案する．また，提案手法に基づく試験スイート生成システムを作成し，生成された試験スイートの系列長が以前の手法で生成されたものの20%以下に短縮されたことが確認できた．

A Method of Generating Efficient Test Suites for Conformance Testing of EFSM Protocols

Akira Beppu　　Masahiro Higuchi　　Mamoru Fujii

Graduate School of Engineering Science,
Osaka University

In this paper, we propose the method of generating efficient test suites for conformance testing of communication protocols modeled as Finite-State Machine with Counters(FSM-C), a class of Extended Finite-State Machines (EFSM). Previously, we proposed a test suite generation method for FSM-C. The test suites can detect all single fault of the implementation, but those are not efficient, because every test case is generated for sigle test item, individually. We investigate a generating procedure of test cases which can be used for multiple test items. We also developed a test suite generating system based on the proposed method, and verified that the length of the test suites generated by the system is shorter than 20 % of those generated by the previous test suite generation method for some sample protocols.

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．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．

多くの通信プロトコルは一連番号やタイマを扱うためにFSMの有限制御部に整数値などを保持するレジスタを持たせた拡張有限状態機械（EFSM）モデルで定義される．EFSMに対する試験系列の生成手法として文献[1]では，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の生成手法を提案している．また，各遷移で実行するレジスタ操作を試験する系列の生成手法は，Wangらが文献[2]で提案している．しかし，Wangらの手法ではレジスタ操作のうち遷移条件の正しさを試験することができない．さらに，レジスタ代入に関し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．そこで，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付き有限状態機械（FSM-C）に対して，Wangらの手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いる．この手法に基づく試験系列は実装もFSM-Cであるとの仮定の下であらゆる単一誤りを検出できることが示されている[4]．

また，C言語のフォールトモデルに基づくミューテーションの誤り検出実験により実際的な誤り検出能力も他の手法に比べ高いことが実験的に評価されている [5]．しかし，この試験系列からなる試験スイートは他手法と比べ試験項目が多いため，系列長が長く試験効率がよくない．本論文では，各試験項目に関して試験系列を独立に生成するのではなく，各試験系列間の関係を考慮に入れて系列を生成することによって，系列長を短くする手法について述べる．以降，2. では，本論文で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. では各試験項目に対する試験系列の性質について，4. では，試験スイートの短縮化手法について，そして 5. では試験スイート生成システムを作成して実験を行った結果について述べる．

2. 準備

本論文ではプロトコル機械のモデルとして拡張有限状態機械の部分クラスであるカウンタ付き有限状態機械（FSM-C）をとり扱う．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付けたプロトコル機械のモデルである．FSM-C の状態遷移は，アクションの有限集合で定義される．アクションは始状態，遷移条件，入力コマンド，出力定義式，レジスタ代入式，終状態の組からなる．図 1は，FSM-C モデルの例である．

図 1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．それぞれの辺についているラベルは，そのアクションで実行する動作を表しており，[{ 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，遷移条件式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．

アクションは，入力を受けとると，入力コマンドがアクションの定義に一致し，その時のレジスタ及び入力パラメータが遷移条件を満たしている時に実行され，レジスタ代入，出力を行い，次の状態に遷移する．遷移条件は，$x-y \leq c$ という形をした差分不等式の論理積からなり，出力またはレジスタに代入される式は，$x+c$ という形の式に限る．ここで，c は整定数値を表し，x，y は入力パラメータまたはレジスタ名を表す．

本論文で扱うプロトコル機械の仕様は以上のような FSM-C として記述されているものとし，その状態遷移は完全かつ決定性で実行時間は有限であるものとする．そして，ある一つの状態を始状態とする状態遷移 $t_1, \cdots, t_n$ がある時，任意の二つの状態遷移 t_i, t_j $(i \neq j, 1 \leq i, j \leq n)$ について，その終状態もしくは出力のいずれかが異なるものとする．以上のように限定したモデルでも一連番号やタイマを扱うプロトコルをモデル化できる．

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われる．

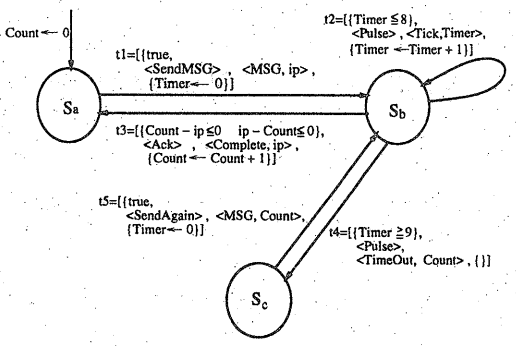

図 1　FSM-C モデルのプロトコル機械の例

3. 試験系列生成手法

FSM-C では，有限制御部の状態の存在，各アクションの始状態と終状態の正しさ，レジスタ代入式及び遷移条件式の定数とレジスタ参照の正しさを試験する必要がある．これらを試験項目と呼ぶ．文献 [4] で提案した試験系列生成手法では FSM-C でモデル化されたプロトコルの各アクションの個々の試験項目に関して，他のアクションの実装に誤りがないという仮定の下で，正しく実装されているかどうかを判定する試験系列を生成する．各試験項目を試験する系列は初期状態から実行可能な入出力系列として生成される．そして，プロトコル機械を初期状態に遷移させる入出力（リセット）によって，生成された入出力系列を連接したものを試験スイートとしている．この試験スイートは，あらゆる単一誤りを検出できる．以下では各試験項目を試験する系列が満たすべき性質について説明する．

3.1 状態の正しさの試験

状態の正しさとは，仕様に記述された状態の存在及び，各アクションの始状態と終状態の正しさを指す．これらは，状態を同定する E-UIO 系列 [1] によって試験できる．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 においては一つの状態を同定するために，その状態の 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．各アクションの実行前後の有限制御部の状態の正しさを，この E-UIO 系列を用いて確認する．

3.2 レジスタ代入式の正しさの試験

ここでは試験対象をアクション t のレジスタ代入式 $R \leftarrow X+C$ とする．このレジスタ代入式が正しく実装されていることを試験するために試験対象のレジスタ代入式で仕様より大きな値が代入された場合実行できない系列 IO_L と，同様に仕様より小さな値が代入された場合実行できない系列 IO_G の対 $\langle IO_L, IO_G \rangle$ を用いる．具体的には

$IO_G = (IO_{TR} \cdot IO_{TG})$

$IO_L = (IO_{TR} \cdot IO_{TL})$

であり，IO_{TR}，IO_{TG}，IO_{TL}は，以下のような入出力系列である．

IO_{TR}：初期状態から実行可能な入出力系列で，その最後の入出力で実行されるアクションはt．最後のアクション以外はt以外のアクション．

IO_{TG}：IO_{TR}実行後に実行可能で，アクションtは含まない．試験対象のアクションtでレジスタRに$X+C$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_{TL}：IO_{TR}実行後に実行可能で，アクションtは含まない．試験対象のアクションtでレジスタRに$X+C$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．

試験系列IO_G，IO_Lによってレジスタ代入式の単一誤りを検出できる[4]．

3.3 遷移条件式の正しさの試験

ここでは試験対象をアクションtの遷移条件式$d_i : x - y \leq c$とする．

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は，次のような条件を満たす系列である．

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tによるものであり，それ以外の遷移はt以外のアクション．また，最後の遷移前の状態において，$x - y = c$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，t以外のアクションのみによるものである．また，最後の遷移前の状態はアクションtの始状態であり，t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．

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，下記の条件を満たす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アクションtの始状態．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t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t以外のアクションを実行する．最後以外はt以外のアクションからなる．

$IOC_{z,F}$：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アクションtの始状態．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，t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tを実行する入出力系列．t以外のアクションのみからなる．

入出力系列$IOC_{z,T}$，$IOC_{z,F}$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．

3.4 系列の生成法

入出力系列IOに対して，IOを入力した時に実行されるアクション系列TRは一意に定まる．各試験項目に関して，上記の条件を満たす入出力系列は必ずしも存在するとは限らない．存在する場合は，幅優先探索によりアクション系列TRの候補を順次生成し，試験系列としての条件を満たすかを調べればよい．生成されたアクション系列TRから入出力パラメータを決定するために，連立不等式Φ_{TR}を生成し，その整数解を求める．不等式Φ_{TR}は，試験対象をFSM-Cに限定したことにより，遷移条件，レジスタ代入式の右辺の形に制限をおいているのでBellman-Fordのアルゴリズム[3]を用いることにより，$O(lm)$（ただし，lは不等式の数，mは変数の数）で解くことができる．

不等式Φ_{TR}の例として，遷移条件式の定数の正しさを試験する入出力系列IOC_Gの不等式Φ_{TRC_G}を形式的に示す．

$IOC_G = i_{G,-m}/o_{G,-m} \cdot \ldots \cdot i_{G,0}/o_{G,0}$とおく．ただし，$i_{G,k}/o_{G,k}$は入力と出力の組を表しており，それぞれ$i_{G,k} = \langle CMD_{G,k}, ip_{G,k} \rangle$，$o_{G,k} = \langle RSP_{G,k}, op_{G,k} \rangle$とする．また，入出力系列$IOC_G$に対応するアクション系列を$TRC_G = t_{G,-m} \cdot \ldots \cdot t_{G,0}$とおく．$r_{G,k}$ $(-m \leq k \leq 0)$は$t_{G,k}$実行前のレジスタrの保持している値を，$r_{G,1}$はIOC_G実行後のレジスタrの保持している値を表すものとする．

連立不等式Φ_{TRC_G}は以下の連立不等式からなる．

1. 各レジスタrに対して
   $r_{G,-m} = r_{init}$ （r_{init}はレジスタrの初期値）
2. $-m \leq k \leq -1$について：
   (a) $t_{G,k}$の遷移条件式中の各不等式 “$x - y \leq c$” に対して
   $x_{G,k} - y_{G,k} \leq c$
   (b) $t_{G,k}$中の各レジスタ代入式 “$r \leftarrow x + c$” に対して
   $r_{G,k+1} = x_{G,k} + c$

(c) $t_{G,k}$中のいずれのレジスタ代入式の左辺にも現れないレジスタrに対して

$r_{G,k+1} = r_{G,k}$

(d) 出力定義式 "$\langle \mathrm{RSP}_{t_{G,k}}, x + c \rangle$" に対して

$op_{G,k} = x_{G,k} + c$

3. $k = 0$について：

(a) $t_{G,0}$の遷移条件式中のd_i以外の各不等式 "$x - y \leq c$" に対して

$x_{G,0} - y_{G,0} \leq c$

(b) $t_{G,0}$中の各レジスタ代入式 "$r \leftarrow x + c$" に対して

$r_{G,1} = x_{G,0} + c$

(c) $t_{G,0}$中のいずれのレジスタ代入式の左辺にも現れないレジスタrに対して

$r_{G,1} = r_{G,0}$

(d) 出力定義式 "$\langle \mathrm{RSP}_{t_{G,0}}, x + c \rangle$" に対して

$op_{G,0} = x_{G,0} + c$

4. 試験対象の式$d_i : x - y \leq c$に対して

$x_{G,0} - y_{G,0} = c$

他の試験項目に対する不等式Φ_{TR}も同様に，初期条件，アクション系列TR中の遷移条件式，レジスタ代入式及び試験項目を試験するための条件式からなる．

4. 試験系列短縮化手法

文献 [4] の提案手法は 3. で示した性質を持つ試験系列を各試験項目に対して独立に生成しているために試験スイートが冗長となり，総系列長が長くなる．ここでは，単一誤りを検出できる性質を保ちながら，試験スイートの短縮化手法について議論する．

4.1 冗長な試験スイート

2つの系列$C = c_1 \cdot \ldots \cdot c_k$と，$D = d_1 \cdot \ldots \cdot d_m$において，$k \leq m$かつ$1 \leq i \leq k$について$c_i = d_i$ならば，$C$は$D$の接頭語であると言い，$C \preceq D$と表すとする．

試験スイート中に$IO_A \preceq IO_B$なる入出力系列IO_A（試験項目A）とIO_B（試験項目B）が存在する時，この試験スイートは冗長である．なぜならば，入出力系列IO_Bによって，試験項目Aも試験できるからである．この場合，試験スイートからIO_Aを除去しても，誤り検出能力を低下させることない．

3.4で述べたように，各試験項目Iの試験系列IO_Iはアクション系列TR_Iと，そのアクション系列から得られる連立不等式Φ_{TR_I}の整数解によって入出力パラメータを決定し生成される．試験項目A，Bに対してアクション系列間に$TR_A \preceq TR_B$が成立し，それぞれの連立不等式Φ_{TR_A}，Φ_{TR_B}を合わせた連立不等式が整数解を持つ時，$IO_A \preceq IO_B$なる入出力系列IO_A，IO_Bが存在し，IO_Bのみで試験項目A，Bを試験できる．そのため，2つの試験項目A，Bを共通の試験系列で試験できるかを考慮する際，試験系列IO_A，IO_Bを生成後にそれらを比較するのではなく，TR_A，TR_B，Φ_{TR_A}，Φ_{TR_B}が得られた段階でそれらが上記の条件を満たすか調べることで，よりよい共通化がはかれる．

4.2 短縮化の問題点

一般には，ある試験項目Aを試験するアクション系列の候補は複数存在する．それらを$TR_{(A,1)}, TR_{(A,2)}, \ldots, TR_{(A,m)}$とする．同様に試験項目$B$を試験するアクション系列の候補を$TR_{(B,1)}, TR_{(B,2)}, \ldots, TR_{(B,n)}$とする．この時，ある$i$，$j$について次の条件が成り立つ時，$IO_A \preceq IO_B$なる入出力系列が存在する．

1. $TR_{(A,i)} \preceq TR_{(B,j)}$ $(1 \leq i \leq m,\ 1 \leq j \leq n)$
2. アクション系列$TR_{(A,i)}$，$TR_{(B,j)}$それぞれから得られる連立不等式$\Phi_{TR_{(A,i)}}$，$\Phi_{TR_{(B,j)}}$を合わせた連立不等式が整数解を持つ．

原理的には各試験項目に対して，試験のためのアクション系列の候補を生成し，生成されたそれぞれのアクション系列が，上記の条件を満たすか網羅的に確かめることによって，試験系列の共通化がはかれる．しかし，アクション系列の候補は一般には無数に存在し，たとえ系列長の短いものから順に一定数以下のもののみ考慮の対象としたとしても，全試験項目を同時に考慮して共通化をはかると，計算時間，必要メモリ量の問題が生じることが予想できる．そこで，以下では実際的な計算時間，メモリ量で系列を短くする手法を議論する．

4.3 試験系列生成手続き

まず、初期状態から出るアクション中の試験項目の試験アクション系列を生成し，以下それ以外の試験項目に関して順に試験アクションを生成する．各試験項目に関して，アクション系列を生成する前に，既に生成された他の試験項目を試験するアクション系列を利用できないか調べる．生成されたアクション系列がない場合や生成されているアクション系列を利用できない場合は，その試験項目を試験するアクション系列の候補の中で，系列長が最短であるアクション系列を生成し，その連立不等式と共に保持する．以下では，既に生成されたアクション系列を利用する手法を述べる．

4.3.1 複数の試験項目を試験できる系列

ある試験項目を試験するための試験系列として保持されているアクション系列が，他の試験項目を試験する条件を満たすかを調べることにより，一つの試験系列によって複数の試験項目を試験できる系列が生成できる．試験項目Aを試験するための試験系列として，アクション系列$TR_{A,m} = t_{A,-m} \cdot \ldots \cdot t_{A,0}$及びその連立不等式$\Phi_{TR_{A,m}}$が保持されているとする．調べる試験項目を$B$，試験対

象のアクションをt_Bとする．この時，以下の条件を満たすかを調べる．

1. $t_B = t_{A,k}$を満たすk $(-m \leq k \leq 0)$が存在する．
2. $k \neq -m$の時，$-m \leq l \leq k-1$について，$t_B \neq t_{A,l}$
3. 連立不等式$\Phi_{TR_{A,m}}$に試験項目Bを試験するための条件式（例：3.4のΦ_{TRC_G}における4.の式）を加えた連立不等式$\Phi'_{TR_{A,m}}$が整数解を持つ．
4. 試験項目Bがアクションt_Bを実行後にアクションを実行しなければならない試験項目である場合（例：レジスタ代入式の正しさを試験する系列），$k \neq 0$かつ$TR_{A,m}$中に存在するアクション系列$t_{A,k+1} \cdot \ldots \cdot t_{A,j}$ $(k+1 \leq j \leq 0)$が試験項目Bを試験できるアクション系列であり，連立不等式$\Phi_{TR_{A,m}}$に試験項目Bを試験するための条件式を加えた連立不等式$\Phi'_{TR_{A,m}}$が整数解を持つ．

上記の条件を満たせば，試験項目A，Bはアクション系列$TR_{A,m}$によって試験可能となる．

このようなアクション系列$TR_{A,m}$が保持されていれば，このアクション系列の連立不等式$\Phi_{TR_{A,m}}$の代わりに連立不等式$\Phi'_{TR_{A,m}}$を保持する．

4.3.2 他試験項目を試験可能とするための系列拡張

4.3.1手法により試験可能にならなかった試験項目に関して，既に保持されているアクション系列を拡張することにより，試験系列が得られるか調べる．4.3.1の1.と2.の条件を満たすアクション系列$TR_{A,m}$が存在する時，$t_{A,0}$の終状態から有向辺と同方向に幅優先探索を行いアクション系列を拡張することによって，他の試験項目が試験可能になるか調べる．$t_{A,0}$の終状態から始まるアクション系列$TR_{B,n} = t_{B,1} \cdot \ldots \cdot t_{B,n}$を連接して，$TR_{A,m}$を拡張することにより試験可能になったとする．この時，拡張したアクション系列$TR_{C,m+n} = t_{A,-m} \cdot \ldots \cdot t_{A,0} \cdot t_{B,1} \cdot \ldots \cdot t_{B,n}$を$TR_{A,m}$の代わりにアクション系列として保持し，その連立不等式として$\Phi_{TR_{A,m}}$にアクション系列$TR_{B,n}$中の遷移条件式，レジスタ代入式及び試験項目を試験するための条件式を加えた連立不等式$\Phi'_{TR_{C,m+n}}$を保持する．

この拡張を行う時に，幅優先探索で行うので大量のメモリを消費する．そのため探索する深さを一定の深さで打ち切る必要がある．

4.3.3 先行系列の重ね合わせ

4.3.1，4.3.2の手法により試験可能にならなかった試験項目は初期状態から始まるアクション系列を生成することになる．この時に，初期状態から試験対象となるアクションの始状態まで実行可能でかつ試験項目を試験するための条件を満たすアクション系列が必要となる．この系列を先行系列と呼ぶ．先行系列は複数の候補があり，その中の一つを採用することになる．そこで，既に保持されているアクション系列を先行系列の一部とすることができれば，総系列長を短くできる．具体的な手法は，試験対象の遷移から有向辺と逆向きに幅優先探索を行い，初期状態もしくは保持されているアクション系列の終状態を探し，先行系列の候補を順次作成し，先行系列として利用できるか確かめる．調べる試験項目をZ，試験対象のアクションをt_Zとする．探索によりアクション系列$TR_{A,m}$の終状態を見つけ，試験対象の遷移の始状態からアクション系列$TR_{A,m}$の終状態までの探索されたアクション系列を$TR_{B,n}$とする．この時，$TR_{A,m}$及び$TR_{B,n}$が次の条件を満たせば，2つのアクション系列を連接した系列$TR_{C,m+n} = t_{A,-m} \cdot \ldots \cdot t_{A,0} \cdot t_{B,-n} \cdot \ldots \cdot t_{B,0}$が先行系列となる．

1. $-m \leq k \leq 0$について$t_{A,k} \neq t_Z$
2. $-n \leq j \leq 0$について$t_{B,j} \neq t_Z$
3. 連立不等式$\Phi_{TR_{A,m}}$に$TR_{B,n}$中に現れる遷移条件式，レジスタ代入式及び試験項目Zを試験するための条件式を加えた連立不等式が整数解を持つ．

幅優先探索で探索しているため，$TR_{B,n}$の系列長よりも短い初期状態から始まるアクション系列$TR_{D,l}$が存在すれば，先に先行系列としての条件を満たすか確かめられる．もし先行系列としての条件を満たせば初期状態から始まる新たなアクション系列の先行系列となる．幅優先探索で探索することにより，総系列長をより短くする先行系列が採用されることになる．

5. システム作成と実験

FSM-Cの各試験項目が3.で示した性質を満たすような試験系列を4.の手法により生成する試験スイート生成システムを作成し，実験を行った．作成したシステムは入力としてFSM-Cとして記述されたプロトコル仕様をとる．生成された試験系列をリセットによって連接することによって、仕様中の各試験項目に対して試験できる試験スイートを出力する．

5.1 システム作成における問題点と対処法

実用規模のプロトコルを適用した時に，実際的なメモリ量で系列生成するためには，幅優先探索を打ち切る深さを選択する必要がある．そこで，作成したシステムでは，幅優先探索を打ち切る深さの範囲を指定することができるようにした．まず，指定された範囲の最小値の深さまで探索することによって生成された試験系列を出力する．次に残りの試験項目に関して，探索を打ち切る深さを1増やして再び試験系列を生成する．この方法を繰り返し，指定された範囲の最大値の深さで終了し，そこまでで試験系列が存在しない試験項目に関してはその旨のメッセージを出力することにした．

5.2 実験方法と結果

このシステムに適用するプロトコルとして，OSIセションプロトコルの主要な6機能単位を選択したものを用いた．その規模は下記の通りである．

表 1　試験スイートの短縮化による系列長と系列数の比較

	短縮化あり	短縮化なし
系列長	1615	9783
系列数	276	1409

- 状態数： 19
- レジスタ数: 12
- アクション数： 126
- 遷移条件式に現れる不等式の数：251
- レジスタ代入式の数：58
- 総試験項目数：2580

このプロトコルに対する試験スイートを，作成した試験スイート生成システムにより生成した．適用実験はSun SPARC STATION 4（メモリ 32MB）上で行った．また，幅優先探索を打ち切る深さの範囲は5から9までとした．生成された試験スイートの系列長と系列数を，各試験項目に関して独立に試験系列を生成し，それらを連接した試験スイートの系列長，系列数と比較したものを表1に示す．なお，系列長は入出力が1組で1としている．結果より，系列長において 1/5 以下に短縮されたことが確認できた．

さらに，生成された試験系列の現実的なフォールト検出能力を調べるために，ミューテーションシステムを使用した．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んだプログラム（ミューテーション）を順次生成するシステムである．このシステムによって生成されたミューテーションのフォールトを，生成された試験スイートによって検出できるか調べることにより，試験スイートのフォールト検出能力を評価することができる [5]．上記のプロトコルに対して 869 個のミューテーションを生成した．作成した試験スイート生成システムによって生成された試験スイートは，このすべてのミューテーションのフォールトを検出することができた．よって，この例プロトコルに関してはフォールト検出能力を落すことなく系列長を短くできることが確認できた．

また短縮化における各段階の系列長の比較として，4. で示した 4.3.1の方法のみを使った時，4.3.1と 4.3.2の方法を使った時の系列長と総 CPU 実行時間について示したものが表2である．結果より，この例プロトコルでは最も単純な 4.3.1の手法で，かなり系列長が短くなっている．そしてさらに，4.3.2，4.3.3の手法を用いることにより，系列長の短縮に関して一層の効果が得られていることがわかる．

表 2　短縮化の段階による系列長と総 CPU 実行時間

	系列長	総 CPU 実行時間（秒）
4.3.1	1938	11653
4.3.1, 4.3.2	1741	12476
4.3.1, 4.3.2, 4.3.3	1615	21720

6. まとめ

本論文では EFSM のサブクラスである FSM-C としてモデル化されたプロトコルに対する適合性試験系列生成手法として，従来我々の研究グループで提案しているすべての単一誤りを検出できる試験系列生成手法を基に，複数の試験項目を試験できる試験系列を生成することによって，試験スイートの系列長を短くする手法を提案した．この手法を用いることによって，すべての単一誤りを検出できる性質を保ちながら，より短い試験スイートを現実的な計算時間，記憶量で生成でき，試験の実施効率が向上する．また，提案手法に基づく試験スイート生成システムを作成し，実用規模のプロトコルを用いて実験を行った．その結果，実際に提案手法に基づいた試験系列が生成でき，実際のフォールトに関する検出能力を落すことなく系列長の短い試験スイートが生成できた．

References

[1] X.Li, T.Higashino,M.Higuchi and K.Taniguchi: Automatic Generation of Extended UIO Sequences for Communication Protocols in an EFSM Model, *Proc 7th Int'l Workshop on Protocol Test Systems*, pp. 213–228 (1994).

[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] T.H.Cormen, C. and R.L.Rivest: *Introduction to Algorithms*, The MIT Press, pp. 539–543 (1990).

[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] 小原勝: 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, 大阪大学基礎工学研究科修士学位論文 (1998).