ソフトウエア工学 93−22
（1993．7．9）

外部イベント駆動型システムの試験系列生成手法

高木 浩則　　橋本 辰範
NTTソフトウェア研究所

〒108　東京都港区港南1-9-1 NTT品川TWINS

本稿では，決定性有限状態機械でモデル化される外部イベント駆動型システムの動作仕様から，各状態遷移の存在およびその遷移先状態の確認を含む，試験実施コストの少ない試験系列を生成する手法を提案する．試験実施コストとしては，試験実施時間，または系列長のいずれかを用いることができる．状態確認系列はUIO系列（Unique Input/Output sequence）を用いる．本手法は，i) 初期状態の最小コストのUIO系列，およびii) 各状態間の最小コストのUIO経路，を用いて試験系列を重ねることにより，試験実施コストを減少させることを特徴とする．いくつかの例題に適用して本手法およびUIO系列を用いた従来手法の評価を行った結果，本手法は，一般に従来手法よりもコストの少ない試験系列を生成することが明らかになった．

Test Sequence Generation for External Event Driven Systems

Hironori TAKAKI　　Tatsunori HASHIMOTO
NTT Software Laboratories

NTT Shinagawa TWINS Bldg., 1-9-1 Kohnan, Minato-ku, Tokyo 108, Japan

We have developed a method for generating minimum-cost test sequences. These test sequences verify the existence and the destination state of each transition in a control specification for an event driven system, modeled as a deterministic finite-state machine (FSM). Testing time as well as length of test sequences may be used to estimate test sequence cost. We confirm the destination state for each transition with the UIO sequence (Unique Input/Output sequence). Our method reduces the cost for test sequences by overlapping these using i) the minimum-cost UIO sequence for the initial state, and ii) the minimum-cost UIO path between each state. We compare our method to other methods that also use UIO sequences and show that the cost for test sequences generated by our method is less.

1．はじめに

外部イベント駆動型のシステムの例として，プロトコルにしたがって協調して動作する通信システムやオペレータからの入力にしたがって動作する会話型システムなどがある．このようなシステムをブラックボックスとしてとらえた場合の動作は，システムが入力イベント待ちである時点を状態とし，入力イベントの発生を起因として状態間を遷移する有限状態機械 (FSM : Finite State Machine) でモデル化できる．

当初，外部イベント駆動型システムの開発工程において，システムが仕様に記述された設計者の意図どおりに動作するか否かを確認するための試験系列生成は手作業で行っていた．そのため，システム規模の増大につれて，試験系列生成にかかる時間と費用が莫大になるという問題が生じていた．さらに，導出した試験系列の実施に時間がかかるという問題，また導出した試験系列の試験漏れといった信頼性や品質の面での問題も副次的に発生しており，より良い試験系列の自動生成が望まれている．

通信分野においては，有限状態機械でモデル化される通信システムの仕様から，コンフォーマンス試験で利用する試験系列の自動生成が盛んである．各状態遷移の存在確認を少なくとも一度は行う試験系列を生成するT法 (Transition-tour method)，各状態遷移の存在だけでなく遷移先状態の確認をも含む試験系列を生成するU法 (Unique input/output sequence method)，D法 (Distinguishing-sequence method)，およびW法 (Characterizing-sequence method) などが提案されている ([3], [6])．

U法，D法，W法は，各状態遷移の存在だけでなく，その遷移先状態の確認をも含む試験系列を生成するため，T法と比較して生成される試験系列の試験実施コスト (例えば，試験実施にかかる時間，あるいは試験系列長) が大きい．U法，D法，W法のなかで一番コストの少ない試験系列を生成するU法には，Rural Chinese Postman Tour を用いて試験系列のコストを改善する手法が文献 [1] で提案されている (以降 SUIO 法と呼ぶ)．U法では状態確認用の系列としてUIO系列を用いる ([4])．ある状態に対するUIO系列とは，システムがその状態にあった時以外は示されない入出力系列であり，一般に一つの状態につき複数個存在する．SUIO法では一つの状態に対して一つのUIO系列を用いる．一つの状態に対して複数のUIO系列を用いてコストの少ない試験系列を求める手法が文献 [7] で提案されている (以降，MUIO法と呼ぶ)．MUIO法に，系列を重ねてさらにコストを少なくするように改良を加えた手法が文献 [8] で提案されている (以降，MUIOO法と呼ぶ)．

従来の手法には，コストの少ない試験系列を得るために，

i) どのUIO系列を用いればよいのか，あるいは，
ii) どの系列を重ねればよいのか，

が明確でない点に問題があった．

本稿では，決定性有限状態機械でモデル化される外部イベント駆動型システムの動作仕様から，各状態遷移の存在およびその遷移先状態の確認を含む，コストの少ない試験系列を生成する手法を提案する．また，いくつかの例題に本手法および従来手法を適用し，本手法の有効性を示す．本手法では，状態を確認する系列としてUIO系列を用いる．本手法は，

i) 初期状態の最小コストのUIO系列，および
ii) 各状態間の最小コストのUIO経路，

を用いて試験系列を重ねることにより，試験実施コストを減少させることを特徴とする．これにより，従来の2つの問題点が解決できる．

一般に各手法で生成した試験系列を比較する場合には，i) エラー検出率，およびii) 試験系列長，が評価尺度として用いられる．本来，試験実施コストとしては，試験実施にかかる時間に関して比較すべきである．しかし，すべての試験系列生成手法が試験実施にかかる時間を考慮しているわけではない．また，各状態遷移にかかる時間にバラつきがあるシステムを対象とした場合，各手法はこれに大きく影響を受けるため，試験実施にかかる時間による比較は困難である．本稿で提案する手法は，試験系列長，試験実施にかかる時間のいずれかを考慮した試験系列の生成が可能であるが，従来手法との比較は試験系列長で行う．ただし，エラー検出率の観点からの比較は本稿の対象外とする．

以下，第2章では，試験対象である外部イベント駆動型システム (以下，試験システムと呼ぶ) のモデル化，遷移の確認方法，および試験系列生成のための前提条件について述べる．第3章では，提案する手法およびその適用例について述べる．第4章では，生成された試験系列が各状態遷移の存在およびその遷移先状態を確認することを示す．また，提案した手法の停止性を示す．第5章では，いくつかの例題に対して，提案した手法と従来手法を適用して比較

検討を行い，その結果について述べる．第6章では，まとめと今後の検討課題について述べる．

2．試験システムのモデル化と前提条件

2．1　試験システムのモデル化

(1) 試験システムの基本動作を，外部からの入力イベントが発生すると，その時点でのシステムの内部状態にもとづき，出力イベントを出し，内部状態を変化させ，再び外部からの入力イベント待ちになると考え，以下の6項組で表される有限状態機械(FSM)でモデル化する．

FSM＝＜Q，I，O，f，g，q＞

ただし　Q：状態集合
　　I：入力イベント集合
　　O：出力イベント集合
　　f：遷移関数　(＝Q×I→Q)
　　g：出力関数　(＝Q×I→O)
　　q：初期状態

(2) 状態 q_i から状態 q_j へ，入力イベント a_k と出力イベント o_l で移動する遷移は $(q_i, q_j; a_k/o_l)$ で表現される．ここで，$q_i, q_j \in Q$，$a_k \in I$，$o_l \in O$，$q_j = f(q_i, a_k)$，$o_l = g(q_i, a_k)$である．

各遷移には遷移コストが付加されており，その遷移の実施にかかる時間を表す．この遷移コストは試験系列の試験実施時間の評価材料として用いる．

2．2　遷移の確認方法

以下では，状態遷移の存在および遷移先状態の確認方法について述べる．

システムをブラックボックスとしてとらえた場合，ある時点のシステムの状態は直接知ることができない．そこで各状態遷移の存在および遷移先状態の確認は，

i) 試験対象の遷移(試験対象遷移)を実施し，その後，
ii) 遷移先の状態を確認する系列(状態確認系列)を実施する，

ことにより行う．

2．3　試験系列生成のための前提条件

U法，D法，W法は，遷移関数および出力関数の誤りを検出する誤り検出能力に関しては同等である．しかし，3つの手法は，遷移先状態を確認するために用いる系列およびその生成方法が異なるため，適用可能なFSMの範囲，生成される試験系列数，試験系列長等に差がある([5], [6])．そこで本稿では．これらの3つの手法の中で最も適用可能なFSMの範囲が広く，また生成される状態確認用の系列が他の手法に比べて短いという理由から，U法で用いられるUIO系列 (Unique Input/Output sequence) を状態確認用の系列として用いる．

従って，対象とするFSMは，U法が適用可能な条件である以下の2つの条件を満たすものとする．

・最小……冗長な状態を含まない
・強連結……任意の二つの状態間を遷移する系列が存在する

また，システムの状態を最初に初期状態に設定するリセット系列rの存在を仮定する．本手法では特に，UIO系列をもたない状態を含むFSMは試験系列生成の対象外とする(図1参照)．

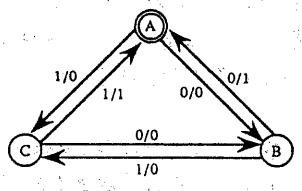


図1　UIO系列をもたないFSMの例(文献[4]より)
(状態AはUIO系列をもたない)

3．提案する手法

提案する手法について説明する前に，表記法について説明する．FSMをグラフ表現した状態遷移グラフ $G = (V, E)$ において，状態集合 $V = \{v_0, v_1, \cdots\}$，遷移集合 $E = \{e_0, e_1, \cdots\}$ とする．ここで，v_0 は初期状態である．また，$e_r = (v_i, v_j; a_k/o_l)$ である．状態集合Vは，2．1のモデルの状態集合Qに対応し，遷移集合Eは，2．1のモデルでは遷移 $(q_i, q_j; a_k/o_l)$ の集合に対応する．状態 v_i のUIO系列は $U(v_i)$ で表現する．1つの遷移および2つ以上の遷移の連結を遷移系列と呼び，遷移系列 X_i, X_j の連結を $X_i \cdot X_j$ で表現する．遷移系列Xの始点となる状態をHEAD(X)，終点となる状態をTAIL(X)で表す．また，遷移 $(v_i, v_j; a_k/o_l)$ の遷移コストを $COST(v_i, v_j; a_k/o_l)$ で表す．

本手法は，以下で定義する分離条件に該当する遷移を，該当しない遷移と分離して扱う．

［定義1］(分離条件)

ある状態を終点とする状態遷移のうちで，同じ入出力イベントをもち，かつ異なる状態を始点とする

状態遷移 (つまり図2のような状況にある遷移).

(定義終了)

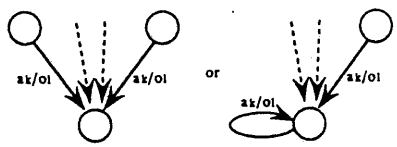


図2　分離条件に該当する遷移

また，以下のような用語を定義する．

［定義2］ (状態間の UIO 経路)

状態 v_i と状態 v_j 間の UIO 経路は，状態 v_i の任意の UIO 系列 $U(v_i)$ に，状態 $TAIL(U(v_i))$ から状態 v_j への任意の経路を付加した系列であり，$UP(v_i, v_j)$ で表現する．$COST(UP(v_i, v_j))$ はこの経路に含まれるすべての遷移のコストの和で与えられる．

特に，状態 v_i と状態 v_j 間の UIO 経路のなかで，最小コストの UIO 経路を $SUP(v_i, v_j)$ で表現する．

(定義終了)

試験系列生成の基本方針は以下のようになる．試験対象遷移として分離条件に該当する遷移を通過する前には，必ず UIO 経路を通過するようにする．また，それにより得られた遷移系列の一番最後に，状態確認系列を付加する．

分離条件に該当しない遷移からなる FSM に適用する手順を3．1の手順1に，該当する遷移を含む FSM に適用する手順を3．2の手順2に示す．以下では FSM のグラフ表現を $G = (V, E)$ とする．

3．1　手順1

分離条件に該当しない遷移からなる FSM に適用する手順を以下に述べる．

［1．UIO 系列導出］

初期状態 v_0 に対する最小コストの UIO 系列 $U(v_0)$ を求める (詳細は付録の Algorithm 1 参照).

［2．対称グラフ作成］

$G = (V, E)$ から，$V^* \equiv V$，E^* は E の各遷移を少なくとも1個以上含み，かつ E^* に含まれる遷移の総コストが最小であるような対称グラフ $G^* = (V^*, E^*)$ を作成する (詳細は文献 [1] の Symmetric Augmentation 参照).

［3．オイラー閉路探索］

対称グラフ G^* において，初期状態を始点・終点とするオイラー閉路 ET を求める．

［4．試験系列導出］

求めたオイラー閉路 ET に初期状態の UIO 系列を連結した遷移系列 $ET \cdot U(v_0)$ が求める試験系列である．

3．2　手順2

分離条件に該当する遷移を含む FSM に適用する手順を以下に述べる．

［1．UIO 系列導出］

初期状態 v_0 に対する最小コストの UIO 系列 $U(v_0)$ を求める (詳細は付録の Algorithm 1 参照).

［2．遷移選別］

$G = (V, E)$ に含まれる各遷移を，分離条件に該当する遷移の集合 E_{oc} と該当しない遷移の集合 E_{uc} にわける．

［3．UIO 経路導出］

$V_{och} = \{ v \mid v = HEAD(e) \text{ and } e \in E_{oc} \}$ $(V_{och} \subseteq V)$ とし，すべての $v_i \in V$ からすべての $v_j \in V_{och}$ に対して最小コストの UIO 経路 $SUP(v_i, v_j)$ を導出する．導出した UIO 経路の集合を E_{sup} とする (詳細は付録の Algorithm 2 参照).

［4．対称グラフ作成］

対称グラフ作成手段の前処理として，すべての $v_i \in V_{och}$ に対応する状態を新たに別状態として作成し，作成した状態の集合を U とする．$v_i \in V_{och}$ に対応する $u_j \in U$ との関係を $u_j = CP(v_i)$ で表す．そして，$V' \equiv V \cup U$，$E' \equiv E_{uc} \cup E_{uc} \cup E_u$ であるような遷移グラフ $G' = (V', E')$ を作成する．ここで，E_{uc}，E_u は以下で定義される

$$E_{uc} = \{ (u_p, v_j; ak/ol) \mid (v_i, v_j; ak/ol) \in E_{oc} \text{ and } u_p = CP(v_i) \}$$

$$E_u = \{ SUP(v_i, u_p) \mid SUP(v_i, v_j) \in E_{sup} \text{ and } u_p = CP(v_j) \}$$

遷移グラフ $G' = (V', E')$ から，$V^* \equiv V'$，E^* は $E_{uc} \cup E_{uc}$ に含まれる遷移を1個以上，E_u に含まれる遷移を0個以上含み，かつ E^* に含まれる遷移の総コストが最小であるような対称グラフ $G^* = (V^*, E^*)$ を作成する (詳細は文献 [1] の Symmetric Augmentation 参照).

［5．オイラー閉路探索］

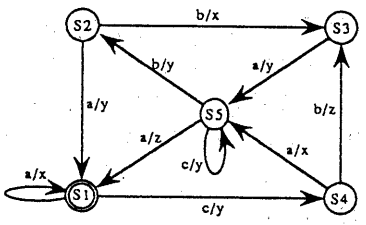
対称グラフ G^* において，初期状態を始点・終点とするオイラー閉路 ET を求める．

［6．試験系列導出］

求めたオイラー閉路 ET に初期状態の UIO 系列を連結した遷移系列 $ET \cdot U(v_0)$ が求める試験系列である．

3．3　適用例

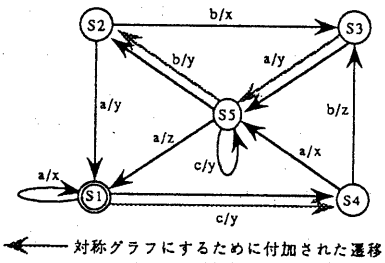
図3は分離条件に該当しない遷移からなるFSMに対する手順1の適用例である．図4は分離条件に該当する遷移をもつFSMに対する手順2の適用例である．以下の適用例においては，各遷移の遷移コストはすべて等しいと仮定して，試験系列を求める．



(a)　FSM のグラフ表現（文献 [3] より）

State	UIO Sequence
S1	a/x, a/x

(b)　初期状態に対する最小コストのUIO系列

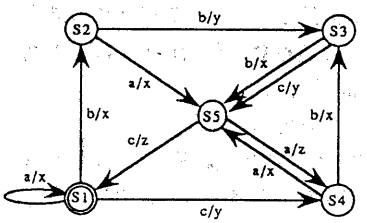


(c)　対称グラフ G*

r, c/y, a/x, c/y, b/y, b/x, a/y, a/z, a/x, c/y, b/z, a/y, b/y, a/y, a/x, a/x

(d)　生成された試験系列
（r はリセット系列）

図3　手順1の適用例
(図3(a)は文献 [3] より)



(a)　FSM のグラフ表現（文献 [1] より）

State	UIO Sequence
S1	a/x, a/x

(b)　初期状態に対する最小コストのUIO系列

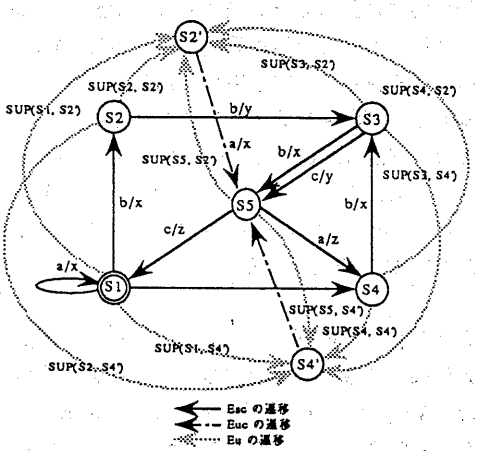
図4　手順2の適用例 (part 1)
(図4(a)は文献 [1] より)

V = { S1, S2, S3, S4, S5 }
Voch = {S2, S4}
Eoc = { (S2, S5; a/x), (S4, S5; a/x) }
Esc = { (S1, S1; a/x), (S1, S2; b/x), (S1, S4; c/y), (S2, S3; b/y), (S3, S5; b/x), (S3, S5; c/y), (S4, S3; b/x), (S5, S1; c/z), (S5, S4; a/z) }

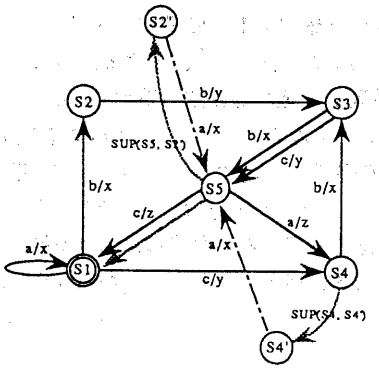
SUP(S1, S2) = SUP(S1, S2') = a/x b/x
SUP(S2, S2) = SUP(S2, S2') = b/y b/x c/z b/x
SUP(S3, S2) = SUP(S3, S2') = b/x c/z b/x
SUP(S4, S2) = SUP(S4, S2') = b/x b/x c/z b/x
SUP(S5, S2) = SUP(S5, S2') = c/z b/x
SUP(S1, S4) = SUP(S1, S4') = a/x c/y
SUP(S2, S4) = SUP(S2, S4') = b/y b/x a/z
SUP(S3, S4) = SUP(S3, S4') = b/x a/z
SUP(S4, S4) = SUP(S4, S4') = b/x b/x a/z
SUP(S5, S4) = SUP(S5, S4') = a/z

Esup = { SUP(S1, S2), SUP(S2, S2), SUP(S3, S2), SUP(S4, S2), SUP(S5, S2), SUP(S1, S4), SUP(S2, S4), SUP(S3, S4), SUP(S4, S4), SUP(S5, S4) }
U = { S2', S4' }
Euc = { (S2', S5; a/x), (S4', S5; a/x) }
Eu = SUP(S1, S2'), SUP(S2, S2'), SUP(S3, S2'), SUP(S4, S2'), SUP(S5, S2'), SUP(S1, S4'), SUP(S2, S4'), SUP(S3, S4'), SUP(S4, S4'), SUP(S5, S4') }

(c)　各集合，遷移系列の意味



(d)　遷移グラフ G'



(e)　対称グラフ G*

r, a/x, b/x, b/y, b/x, c/z, c/y, b/x, c/y, a/z, b/x, b/x, a/z, a/x, c/z, b/x, a/x, c/z, a/x, a/x

(f)　生成された試験系列
（r はリセット系列）

図4　手順2の適用例 (part 2)

4．理論的裏付け

以下では，提案した手順により生成された試験系列は，仕様上に記述されたすべての遷移の存在およびその遷移先状態を確認するという基準を満たすことを4．1で示す．また，提案した手順の停止性を4．2で示す．

まず基本となる定理を述べる．本手法で分離条件を設定する理由は，以下の定理1による．

［定理1］ (UIO系列にならない条件)

分離条件に該当する遷移をeとする．状態TAIL(e)を始点とするいかなる遷移系列sに対しても，遷移系列e・sはUIO系列にはならない．

［証明1］図2より明らかである．

(証明終)

［定理2］ (ユニークな遷移系列)

分離条件に該当しない遷移eに，状態TAIL(e)のUIO系列を付加した遷移系列e・U(TAIL(e))は状態HEAD(e)を確認するUIO系列である．

［証明2］遷移eは分離条件に該当しない遷移であるため，状態TAIL(e)を終点とする遷移のなかでユニークな入出力イベントをもつ．従って，遷移系列e・U(TAIL(e))はユニークであり，状態HEAD(e)を確認するUIO系列となる．

(証明終)

［定理3］ (分離条件に該当しない遷移の確認)

分離条件に該当しない遷移からなる遷移系列sに，状態TAIL(s)のUIO系列を付加した遷移系列s・U(TAIL(s))は，sに含まれるすべての遷移の存在および遷移先状態を確認する遷移系列であり，かつ状態HEAD(e)を確認するUIO系列である．

［証明3］$s = e_1 \cdot e_2 \cdot \cdots \cdot e_n$とする．これにTAIL(s)のUIO系列uを連結した遷移系列$e_1 \cdot e_2 \cdot \cdots \cdot e_n \cdot u$を考える．定理2より，$e_n \cdot u$は$e_n$の存在およびその遷移先状態を確認すると同時に，UIO系列となる．同様に，$e_{n-1} \cdot (e_n \cdot u)$は，$e_{n-1}$の存在およびその遷移先状態を確認すると同時に，UIO系列となる．これを再帰的に適用すると定理3が成り立つ．

(証明終)

4．1　基準を満たすことの証明

［手順1の証明］

手順1において，対称グラフにするために付加した遷移はすべて分離条件に該当しない．従って，求めたオイラー閉路ETは分離条件に該当しない遷移からなる遷移系列である．得られる試験系列ET・U(vo)はET・U(TAIL(ET))とも表せる．従って，定理3より手順1で生成される試験系列は仕様上に記述されたすべての遷移の存在およびその遷移先状態を確認する系列である．

(証明終)

［手順2の証明］

分離条件に該当する遷移をe_{uc}，該当しない遷移からなる遷移系列をs_{ec}，状態確認系列として用いる初期状態の最小コストのUIO系列および各状態間の最小コストのUIO経路をs_uとすると，生成される試験系列の各部分は以下の3種類に分類できる．

1）$s_{ec} \cdot s_u$

2）$e_{uc} \cdot s_u$

3）$e_{uc} \cdot s_{ec} \cdot s_u$

1の系列は，定理3よりs_{ec}に含まれるすべての遷移の存在およびその遷移先状態を確認する系列であり，かつUIO系列となる．2の系列は，遷移e_{uc}の存在およびその遷移先状態を確認する系列であることは明らかである．3の系列は，$s_{ec} \cdot s_u$の部分が1の系列と同じであることから，s_{ec}に含まれる遷移と遷移e_{uc}の存在およびその遷移先状態を確認する系列となる．本来，$E = E_{cc} \cup E_{uc}$であるが，対称グラフG*では，E_{cc}はE_{uc}で置き換えられている．従って，E_{uc}およびE_{ec}に含まれるすべての遷移の存在およびその遷移先状態を確認することにより，手順2で生成される試験系列は仕様上に記述されたすべての遷移の存在およびその遷移先状態を確認する系列となる．

(証明終)

4．2　停止性

以下では，FSMに含まれる状態数をn，遷移数をm，任意の状態を始点とする遷移数の最大値あるいは終点とする遷移数の最大値をd_{max}，対称グラフの遷移数をm_0とする．

［手順1の停止性の証明］

1のUIO系列導出の処理は，初期状態に対するUIO系列が存在すれば$O(n^2 d_{max} 2^{n-1})$の計算時間内に必ず停止する．本手法では，各状態に対してUIO系列が必ず存在するFSMを考慮対象とするため，1の処理の停止性は保証される．文献[1]により，2の対称グラフ作成の処理は線形計画法，あるいはネットワークフローアルゴリズムを用いて$O(m_0 m \log n)$の計算時間内に停止し，また3のオイラー閉

路探索の処理は$O(m_0)$の計算時間内に停止する．4の試験系列導出の処理の停止性は明らかである．従って，手順1の処理は必ず停止する．

(証明終)

[手順2の停止性の証明]

手順1の証明同様，1のUIO系列導出処理の停止性は保証される．2の遷移選別の処理は，$O(d_{max}^2 n)$の計算時間内に停止する．3のUIO経路導出の処理は，$O(n^4 d_{max} 2^{n-1})$の計算時間内に停止する．4の対称グラフ作成，5のオイラー閉路探索，6の試験系列導出の処理の停止性は，手順1の証明同様，明らかである．従って，手順2の処理は必ず停止する．

(証明終)

5．評価

5．1　実験的評価

ここでは本手法，U法，SUIO法，MUIO法，およびMUIOO法をいくつかの例題に対して適用して比較し評価を行う．

本来従来手法との比較は，生成される試験系列の試験実施にかかる時間に関して行うべきであるが，各遷移に付加される遷移コストはすべて等しく1であると仮定することにより，生成される試験系列の試験実施にかかる時間のかわりに，試験系列長を評価尺度として用いる．

各手法を文献[3]，[1]，[6]から抜き出した遷移グラフ図3(a)，図4(a)，図5に対して適用した．生成された試験系列のコストを表1に示す．

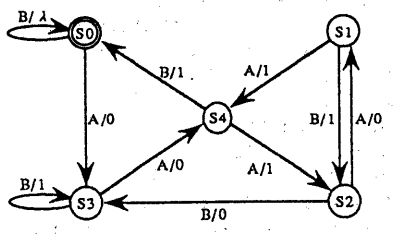


図5　分離条件に該当しない遷移からなるFSM
(文献[6]より)

表1　各生成手法の比較

適用対象 / 生成手法	図3(a)のFSM	図4(a)のFSM	図5のFSM
本手法	15 + R	19 + R	15 + R
U法	28 + 6R	31 + 8R	29 + 6R
SUIO法	38 + R	34 + R	30 + R
MUIO法	32 + R	30 + R	30 + R
MUIOO法	17 + R	17 + R	16 + R

(Rはリセット系列rのコスト)

5．2　考察

実験的な評価から本稿で提案した手法は，分離条件に該当しない遷移からなるFSMに対して適用した場合に効果が大きく，他の手法と比較してコストの小さい試験系列を生成することがわかる．分離条件に該当する遷移をもつ図4(a)のFSMに対して適用した場合，MUIOO法が最もコストの少ない試験系列を生成した．これは，このような単純な例であると，MUIOO法でもコストの少ない試験系列がヒューリスティックに生成可能であるためである．しかし，FSMの規模が大きくなるとMUIOO法が必ずしもコストの少ない試験系列を生成するとは限らない．

6．まとめ

過去から現在までの外部からのイベント入力により現時点の動作および出力が決まるような外部イベント駆動型システムの動作試験を対象に，システムの動作をモデル化したFSMから試験実施コストの少ない試験系列を導出する手法を提案した．また，提案した手法と従来の手法を比較検討し，その有効性を述べた．

従来の手法には，コストの少ない試験系列を得るために，

i) どのUIO系列を用いればよいのか，あるいは，

ii) どの系列を重ねればよいのか，

が明確でない点に問題があった．本手法は，

i) 初期状態の最小コストのUIO系列，および

ii) 各状態間の最小コストのUIO経路，

を用いて試験系列を重ねることにより，試験実施コストを減少させることを特徴とする．これにより，従来の2つの問題点が解決できる．

文献[3]，[1]，[6]から抜き出した遷移グラフ図3(a)，図4(a)，図5に対して本手法を適用した結果，分離条件に該当しない遷移からなるFSMに対して生成される試験系列は従来の手法より短くなることがわかった．今後の課題として，

1)本手法が適している問題領域の特定，

2)エラー検出率からの試験品質評価，および

3)各手法のアルゴリズムの計算量の比較，

があげられる．

参考文献

[1] Aho, A.V., Dahbura, A. T., Lee, D. and Uyar, M. U. : An Optimization Technique for Protocol Conformance Test Generation Based on UIO Sequences and Rural Chinese Postman Tours, Protocol Specification, Testing, and Verification VIII, pp.75-86, North-Holland, 1988.

[2] Amer, P. D. : Improvements on UIO Sequence Generation and Partial UIO Sequences, 12th International Symposium on Protocol Specification, Testing, and, Verification, Lake Buena Vista, Florida, USA, Jun., 1992.

[3] Bosik, B. S. and Uyar, M. U. : Finite State Machine based Formal Methods in Protocol Conformance Testing: from Theory to Implementaion, Computer Networks and ISDN Systems, Vol.22, pp.7-33, North-Holland, 1991.

[4] Sabnani, K. K. and Dahbura, A. T. : A Protocol Test Generation Procedure, Computer Networks and ISDN Systems, Vol.15, pp.285-297, 1988.

[5] Sidhu, D. P. and Leung, T. : Fault Coverage of Protocol Test Methods, Proceeding IEEE INFOCOM '88, pp.80-85, Mar., 1988.

[6] Sidhu, D. P. and Leung, T. : Formal Methods for Protocol Testing: A Detailed Study, IEEE Trans. Softw. Eng., Vol.15, No.4, Apr., 1989.

[7] Shen, Y. -N., Lombardi, F. and Dahbura, A. T. : Protocol Conformance Testing Using Multiple UIO Sequences, Protocol Specification, Testing, and Verification, IX, pp.131-143, North-Holland, 1990.

[8] Yang,B. and Ural, H. : Protocol Conformance Test Generation using Multiple UIO Sequences with Overlapping, SIGCOMM '90 Symposium : Communication Architecture and Protocols in Computer Comm. Review, 20(4), pp.118-125, Sep,. 1990.

付録

Algorithm 1 は，最小コストの UIO系列を求めるアルゴリズムである．Algorithm 2 は，最小コストの UIO経路を求めるアルゴリズムである．Algorithm 1 は，文献[2] で述べられているアルゴリズムを，コストを扱えるように拡張したものである．

Definition:

V is the set of states in FSM.
E is the set of transitions in FSM.
n is the number of states in FSM.
COST(s), where s is sequence, is cost function.
SP(vi, vj) is the minimun-cost path from state vi to state vj.
TAIL(s) is the final entry state after firing sequence s.
U(vi) is the minimum-cost UIO sequence for state vi.
MAX is the sequence that have the cost more than
(maxmum-cost of one transition)$\times 2n^2+1$.

let a vertex g_s be a tuple $<v_s, P_s>$ where
v_s is the state that results from state v after firing sequence s.
P_s is the set of states that are reachable by s from any state other than v.

Algorithm 1

Minimum-cost UIO sequence generation algorithm for state v.

```
put a vertex g λ = <v, V - {v}> into a queue OPEN
                                    and a list VISITED;
U(v) ← MAX;
while (OPEN is not empty) do {
  remove a vertex gs = <vs, Ps> from the head of OPEN;
  for each transition of the form (vs, vd; ak/ol) do {
    s' ← s · ak/ol;
    Ps' ← {vq | vp∈Ps and (vp, vq; ak/ol)∈E};
    gs' ← <vd, Ps'>;
    if (Ps' = φ) then {
      if (COST(s') < COST(U(v))) then
        U(v) ← s';
    } else if (gs' ∈ VISITED) then {
      continue;
    } else if (vd ∈ Ps') then {
      insert gs' into VISITED;
    } else {
      if (COST(s') < COST(U(v))) then
        insert gs' onto the tail of OPEN and into VISITED;
      else
        insert gs' into VISITED;
    }
  }
}
if (U(v) = MAX) then
  return "no UIO sequence for the state v";
else
  return U(v);
```

Algorithm 2

Minimun-cost UIO path generation algorithm from each state to state v.

```
for each state vstart ∈ V do
  SUP(vstart, v) ← U(vstart) · SP(TAIL(U(vstart)), v);
for each state vstart ∈ V do {
  put a vertex g λ = <vstart, V - {vstart}> into
                                    a queue OPEN and a list VISITED;
  while (OPEN is not empty) do {
    remove a vertex gs = <vs, Ps> from the head of OPEN;
    for each transition of the form (vs, vd; ak/ol) do {
      s' ← s · ak/ol;
      Ps' ← {vq | vp∈Ps and (vp, vq; ak/ol)∈E};
      gs' ← <vd, Ps'>;
      if (Ps' = φ) then {
        if (COST(s') + COST(SP(vd, v))
                                  < COST(SUP(vstart, v))) then
          SUP(vstart, v) ← s' · SP(vd, v) to ;
      } else if (gs' ∈ VISITED) then {
        continue;
      } else if (vd ∈ Ps') then {
        insert gs' into VISITED;
      } else {
        if (COST(s') < COST(SUP(vstart, v))) then
          insert gs' onto the tail of OPEN and into VISITED;
        else
          insert gs' into VISITED;
      }
    }
  }
}
return ∀vstart ∈ V, SUP(vstart, v);
```