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†

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）各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 † and TATSUNORI HASHIMOTO †

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 each 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）[4]，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

† 日本電信電話株式会社
NTT Corporation

列(Unique Input/Output sequence)を用いる．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)各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，従来の二つの問題点を解決する．これにより，試験実施コストの少ない試験系列が得られる．

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く，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，提案した手法により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系列であることを示す．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．最後に第6章では，本稿の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（$=Q\times I\rightarrow Q$）
g：出力関数（$=Q\times I\rightarrow O$）
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，これらの三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

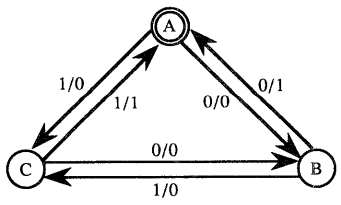


図1 UIO系列をもたないFSMの例（文献4）より）
（状態AはUIO系列をもたない）
Fig. 1 An example of FSM, in which state A doesn't have an UIO sequence. (extracted from Ref. 4))

いう理由から，U法で用いられるUIO系列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二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）．

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節のモデルの状態集合Qに対応し，遷移集合Eは，2.1節のモデルでは遷移$(q_i, q_j; a_k/o_l)$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．一つの遷移および二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．

本手法では，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）

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

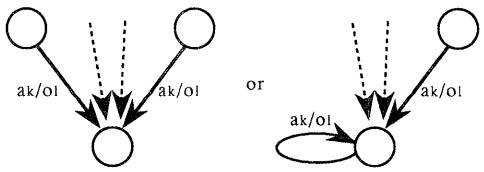


図2 分離条件に該当する遷移
Fig. 2 Example of transitions that satisfy separate condition.

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．また，状態v_iの最小コストのUIO系列を$SU(v_i)$で表現する．

（定義終了）

試験系列生成の基本方針を以下に示す．

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，その遷移系列の終点の状態確認系列を付加する．

3.1 手　　順

以下で扱う集合はすべて多重集合（要素の重複のありうる集合）である．集合の和や差の演算は，多重集合上の演算である．以下ではFSMのグラフ表現を$G=(V, E)$とする．

［1. UIO系列導出］

各状態$v_i \in V$に対する最小コストのUIO系列$SU(v_i)$を求める．求め方の基本方針を以下に示す（詳細は付録のAlgorithm 1参照）．

状態v_iから，順次遷移させ探索する過程で，遷移系列s′が以下の条件を満たす場合に，その先の探索を打ち切り，すべての探索可能な経路がなくなった時点での最小コストのUIO系列が求めるものである．

a) UIO系列である場合．
b) UIO系列になりえない場合．
c) 最小コストのUIO系列になりえない場合．すなわち，より少コストの遷移系列がs′と同等の状態能判別能力をもつ場合や，その時点の最小コストのUIO系列よりもs′のコストが大きい場合．

［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_{sc}にわける．

[3. UIO 経路導出]

$V_{och}=\{v|v=HEAD(e)$ and $e\in E_{oc}\}(V_{och}\subseteq V)$ とする．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．導出した経路の集合を E_{sup} とする．最小コストの UIO 経路 $SUP(v_i, v_j)$ の求め方の基本方針を以下に示す（詳細は付録の Algorithm 2 参照）．

状態 v_i から，順次遷移させ探索する過程で，遷移系列 s′ が以下の条件を満たす場合に，その先の探索を打ち切り，すべての探索可能な経路がなくなった時点での最小コストの UIO 経路が求めるものである．

a) UIO 系列である場合（その UIO 系列を利用して可能な最小コストの UIO 経路を求める）．
b) UIO 系列になりえない場合．
c) 最小コストの UIO 系列/経路になりえない場合．すなわち，より少コストの遷移系列が s′ と同等の状態判別能力をもつ場合や，その時点の最小コストの UIO 経路よりも s′ を利用した v_j への最小コスト経路の方がコストが大きい場合．

[4. 対称グラフ作成]

対称グラフ作成の前処理として，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，$V'\equiv V\cup U$, $E'\equiv E_{sc}\cup E_{oc}'\cup E_{sup}'$ であるような遷移グラフ $G'=(V', E')$ を作成する．ここで，E_{oc}', E_{sup}' は以下で定義される．

$$E_{oc}'=\{(u_p, v_j; a_k/o_l)|\ (v_i, v_j; a_k/o_l)\in E_{oc} \text{ and } u_p=CP(v_i)\}$$

$$E_{sup}'=\{SUP(v_l, u_p)|\ SUP(v_i, v_j)\in E_{sup} \text{ and } u_p=CP(v_j)\}$$

遷移グラフ $G'=(V', E')$ から，$V^*\equiv V', E^*$ は $E_{sc}\cup E_{oc}'$ に含ませる各遷移を 1 個以上，E_{sup}' に含まれる遷移を任意個含み，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）．

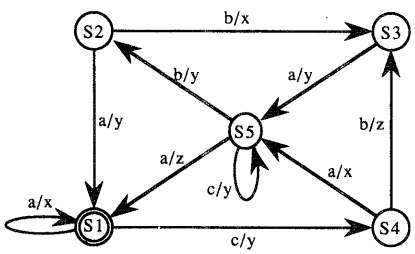
[5. 不要遷移系列削除]

E^* から $E_{sc}\cup E_{oc}'$ に含まれる各遷移を各々一つずつ選び，その遷移集合を E_{org} とし，それ以外の遷移集合を $E_{unorg}\equiv E^*-E_{org}$ で定義する．E_{org} は試験対象遷移の集合であり，E_{unorg} はそれ以外の対称グラフにするために付加した遷移の集合である．

E_{unorg} をもとに，$SU(v_i)(v_i\in V\cup U)$（ここで $v_i\in V$, $u_j\in U$, $u_j=CP(v_i)$ である場合，$SU(u_j)=SU(v_i)$ で定義される）を利用した付録の Algorithm 3 を用いて，状態 v_{del} から初期状態への不要な遷移系列 s_{del} を求める．対称グラフ G^* から不要な遷移系列 s_{del} を削除してできた遷移グラフを G_{opt} とする．

不要遷移系列を求めるための基本方針を以下に示す．

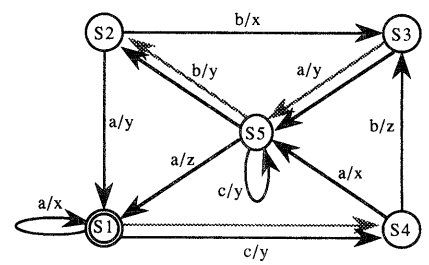
Symmetric Augmentation において対称グラフにするために付加した遷移を逆方向にたどり，初期状態からその遷移のみを用いて可能な最大コストの遷移系



(a) FSM のグラフ表現（文献 3) より）

State	UIO Sequence
S1	a/x・a/x
S2	b/x
S3	a/y・a/z
S4	b/z
S5	a/z

(b) 各状態に対する最小コストの UIO 系列



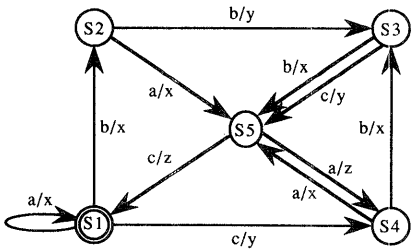
対称グラフにするために付加された遷移

(c) 対称グラフ G^*（$=G_{opt}$）

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）

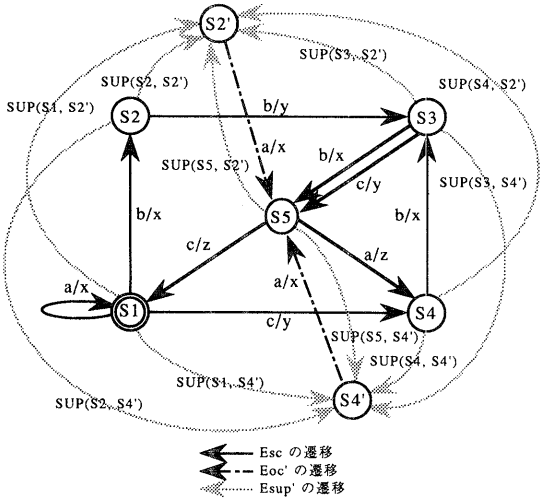
図 3 手順の適用例 その 1
（(a) は文献 3) より）
Fig. 3 Example 1: application of our method.
((a) is extracted from Ref. 3))



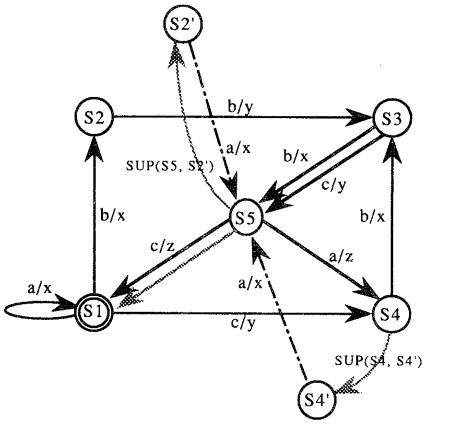
(a) FSM のグラフ表現（文献 1) より）

State	UIO Sequence
S1	a/x・a/x
S2	b/y
S3	b/x・c/z
S4	b/x・c/y
S5	c/z

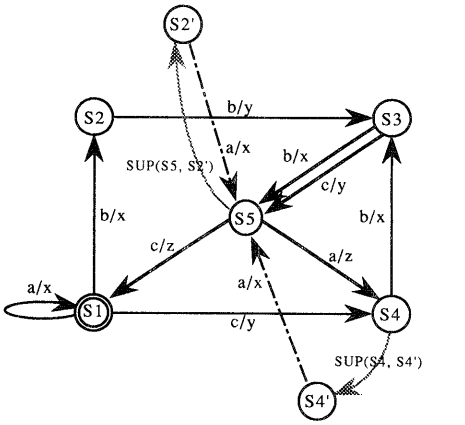
(b) 初期状態に対する最小コストのUIO系列



(c) 遷移グラフ G'



(d) 対称グラフ G*



(e) グラフ Gopt

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

(f) 生成された試験系列
（r はリセット系列）

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

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' }
Eoc' = { (S2', S5; a/x), (S4', S5; a/x) }
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') }

(g) 各集合，遷移系列の意味

図4 手順の適用例その2 （(a)は文献1)より）
Fig. 4 Example 2: application of our method.
((a) is extracted from Ref. 1))

列を求める．求められた遷移系列が不要遷移系列 s_{del} であり，状態 HEAD(s_{del})が状態 v_{del} である．

［6．オイラー小道探索］

遷移グラフ G_{opt} において，初期状態を始点とし，G_{opt} の各遷移をただ一度だけ通過し，状態 v_{del} を終点とするオイラー小道(Euler trail)EP を求める．

［7．試験系列導出］

求めたオイラー小道 EP に状態 v_{del} の最小コストの UIO 系列を連結した遷移系列 EP・SU(v_{del})が求める試験系列である．

3.2 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 に対する手順の適用例である．適用例では，各遷移の遷移コストはすべて等しいと仮定して，試験系列を求めた．

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 節で示す．また，提案した手順の停止性を 4.2 節で示す．

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 $e=(v_i, v_j; a_k/o_l)$ とする．遷移 e は分離条件に該当しない遷移であるため，状態 v_j を終点とし，a_k/o_l を入出力イベントとしてもつ遷移の始点はユニークに状態 v_i に決まる．U(TAIL(e)) はユニークな系列であるため，遷移系列 e・U(TAIL(e))もユニークである．従って，e・U(TAIL(e))は状態 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］ $s=e_1・e_2・\cdots・e_n$ とし，s に TAIL(s) の UIO 系列 u を連結した遷移系列 $e_1・e_2・\cdots・e_n・u$ を考える．定理2より，$e_n・u$ 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，$e_{n-1}・(e_n・u)$ 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

不要遷移系列削除処理は，グラフ G* のオイラー閉路(始点と終点が一致するオイラー小道)を考えた場合に，i)最後尾に位置し，ii)E_{unorg} に含まれ，かつiii)コストの極力大きい，遷移系列を取り除くために行う．E_{org} に含まれる遷移は試験対象遷移であり，E_{unorg} に含まれる遷移は，状態確認系列あるいは各部分試験系列を連結するための遷移である．そのため，E_{unorg} に含まれる遷移を取り除いても試験対象遷移が削除されることはない．

対称グラフ作成の過程において，E_{oc} に含まれる遷移は E_{oc}' に含まれる遷移で置換される．同様に，E_{sup} に含まれる遷移は E_{sup}' に含まれる遷移で置換される．これを考慮して，遷移グラフ G_{opt} から求められるオイラー小道に含まれる各遷移を，元のグラフ G=(V, E) に含まれる，

i) 分離条件に該当する遷移 $e_{oc} \in E_{oc}$

ii) 分離条件に該当しない遷移 $e_{sc} \in E_{sc}$

および，手順の途中で導出した，

iii) 最小コストの UIO 経路 $e_{sup} \in E_{sup}$

にマッピングして，正規表現で表すと，

$$(e_{sc}|e_{sup}・e_{oc})^+$$

で表される．この遷移系列の終点の UIO 系列を u とすると，最終的に求められる試験系列は，

$$(e_{sc}|e_{sup}・e_{oc})^+・u$$

で表される．グラフ G の各遷移 $e_i \in E$ は，上記の e_{sc} および e_{oc} のいずれかの部分に入る．求められた試験系列には以下の3種類の系列が部分試験系列として含まれる．

1) $e_{sc}^+・e_{sup}$ あるいは $e_{sc}^+・u$

2) $e_{oc}・e_{sc}^+・e_{sup}$ あるいは $e_{oc}・e_{sc}^+・u$

3) $e_{oc}・e_{sup}$ あるいは $e_{oc}・u$

1の系列は，定理3より e_{sc}^+ 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，$e_{sc}^+・e_{sup}$ ある

いは e_{sc}^{+}・u の部分が1の系列と同じであることから，e_{sc}^{+} に含まれる遷移と遷移 e_{oc} の存在およびその遷移先状態を確認する系列となる．3の系列が e_{oc} 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．従って，提案した手順で生成される試験系列は，E 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，任意のある状態を始点（あるいは終点）とする遷移数の最大値を d_{max}，対称グラフ G* における試験対象遷移以外の遷移数を m_0，遷移グラフ G_{opt} の遷移数を m' とする．また，付録の各アルゴリズムにおいて，最小コストの系列 s をもつ vertex が生成されるまでに vertex が置換される最大数を p_{max} とする．

1 の UIO 系列導出の処理は，$O(n^3 p_{max} d_{max} 2^{n-1})$ の計算時間内に必ず停止する．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，$O(n^4 p_{max} d_{max} 2^{n-1})$ の計算時間内に停止する．文献1)により，4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多項式オーダーの計算時間内に停止する．5の不要遷移系列削除の処理は，$O(np_{max} d_{max}(\log_2 n + m_0)2^{m_0})$ の計算時間内に停止する．6のオイラー小道探索の処理は $O(m')$ の計算時間内に停止する．7の試験系列導出の処理の停止性は明らかである．従って，提案した手順の処理は必ず停止する．

(証明終)

5. 評価

5.1 実験的評価

各手法（本手法，U 法，SUIO 法，MUIO 法，および MUIOO 法）を文献1)，3)，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．

ただしここでは，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，試験系列長をコストとして用いる．

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 法は本手法と同じコストの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．

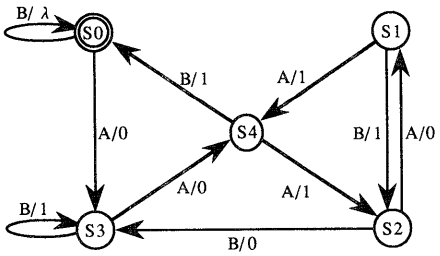


図5 分離条件に該当しない遷移からなる FSM（文献6)より）
Fig. 5 An example of FSM, in which no transition satisfy separate condition. (extracted from Ref. 6))

表1 各生成手法の比較
Table 1 Comparison of each method in cost.

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	17+R	14+R
U 法	28+6 R	31+8 R	29+6 R
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)

従来手法と比較した本手法の優位性は以下のように説明できる．初めに，系列長をコストとし，求められた試験系列の中で試験対象遷移と UIO 系列を除いた，純粋に連結するための遷移の数が，本手法と従来手法でほぼ等しいと仮定する．

本手法で生成した試験系列には，状態確認系列は分離条件に該当しない遷移個分の UIO 系列と最後に付加する UIO 系列しか必要ない．これが必要最低限度であるのに対して，従来手法はそれ以上の個数（最大，試験対象遷移個分）が必要である．従って，その分だけ本手法で生成した試験系列の方がコストが少ないことになる．さらに，最初に，純粋に連結するための遷移の数が等しいと仮定して話を進めたが，本手法では，この連結する遷移を極力減らすための工夫（最小コストの UIO 経路の利用，および不要遷移系列の削除処理）をしており，従来手法と比較して，連結するための遷移の数も少なくなる．そのため，生成された試験系列のコスト差はさらに広がる．

6. おわりに

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）各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，従来の二つの問題点を解決する．これにより，試験実施コストの少ない試験系列が得られる．

複数のFSMを例に本手法を適用した結果，従来の手法より少ないコストの試験系列を生成することがわかった．その効果は，特に分離条件に該当しない遷移からなるFSMに適用した場合に顕著である．

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) Chun, W. and Amer, P. D.: Improvements on UIO Sequence Generation and Partial UIO Sequences, *12th Int. Symp. Protocol Specification, Testing, and Verification*, Lake Buena Vista, Florida, USA (June 1992).

3) Bosik, B. S. and Uyar, M. U.: Finite State Machine Based Formal Methods in Protocol Conformance Testing: from Theory to Implementation, *Comput. Networks ISDN Systems*, Vol. 22, pp. 7-33, North-Holland (1991).

4) Sabnani, K. K. and Dahbura, A. T.: A Protocol Test Generation Procedure, *Comput. Networks ISDN Systems*, Vol. 15, pp. 285-297 (1988).

5) Sidhu, D. P. and Leung, T.: Fault Coverage of Protocol Test Methods, *Proc. 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, pp. 413-426 (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*, Vol. 20, No. 4, pp. 118-125 (Sep. 1990).

付　録

Algorithm 1は，各状態の最小コストのUIO系列を求めるアルゴリズムであり，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．Algorithm 2は，各状態から状態vに対する最小コストのUIO経路を求めるアルゴリズムである．Algorithm 3は，削除可能な不要遷移系列を求めるアルゴリズムである．各アルゴリズムにおいて，OPENはさらに探索すべきvertexのリストであり，VISITEDは探索範囲を限定するためのvertexのリストである．OPENは系列sのコストで，VISITEDはv_sとV_sあるいはv_sとE_sで順序付けされる2分探索木を用いて表現される．

アルゴリズムの計算量において，FSMの状態数をn，遷移数をm，ある状態を始点(あるいは終点)とする遷移の最大数をd_{max}，対称グラフにするために付加された遷移数をm_0，最小コストのsを含むvertexが生成されるまでにvertexが置換される最大回数をp_{max}とする．

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 a sequence, is cost function.

SP(v_i, v_j) is the minimun-cost path from state v_i to state v_j.

TAIL(s) is the final entry state after firing sequence s.

SU(v_i) is the minimum-cost UIO sequence for state v_i.

MAX is the sequence that have the cost more than (maxmum-cost of one transition)$\times 2n^2+1$.

null is empty sequence.

v_0 is initial state in FSM.

let a vertex g_s be a tuple <v_s, V_s, s> where

v_s is the state that results from state v after firing sequence s.

V_s is the set of states that are reachable by s from any state other than v.

s is a sequence.

let a vertex h_s be a tuple <v_s, E_s, s> where

v_s is the state that results from state v after firing sequence s.

E_s is the set of transitions that are not traversed.

s is a sequence.

E_{unorg} is the set of transitions which replicated from the corresponding symmetric augmentation.

The Sentence between /* and */ is comment.

Algorithm 1

Minimum-cost UIO sequence generation algorithm for each state.

```
(1)   for each state vstart ∈ V do {
(2)     put a vertex gstart = <vstart, V−{vstart}, null> into a list OPEN and a list VISITED;
(3)     SU(vstart) ← MAX;
(4)     while (OPEN is not empty) do {
(5)       remove the vertex gs = <vs, Vs, s>, which have minimum-cost sequence s, from OPEN;
(6)       for each transition of the form (vs, vd; ak/ol) ∈ E do {
(7)         s' ← s・ak/ol;
(8)         Vs' ← {vq | vp ∈ Vs and (vp, vq; ak/ol) ∈ E};
(9)         gs' ← <vd, Vs', s'>;
(10)        if (Vs' = φ) then {    /* 真なら s' はUIO系列である */
(11)          if (COST(s') < COST(SU(vstart))) then
(12)            SU(vstart) ← s';
(13)        } else if (vd ∈ Vs') then {    /* 真なら s' はUIO系列になりえない */
(14)          continue;
(15)        } else if (∃gt = <vt, Vt, t> ∈ VISITED such that vt = vd and Vt = Vs') then { /* 真
                    なら s' と同じ状態判別能力をもつ遷移系列がみつかっている */
(16)          if (COST(s') < COST(t)) then {    /* 偽なら s' の先を探索しても最小コスト
                                                   のUIO系列になりえない */
(17)            replace gt with gs' in VISITED;
(18)            if (gt exist in OPEN) then
(19)              remove gt from OPEN;
(20)            insert gs' into OPEN;
(21)          }
(22)        } else {
(23)          if (COST(s') < COST(SU(vstart))) then    /* 偽なら s' の先を探索しても最小
                                                         コストのUIO系列になりえない */
(24)            insert gs' into OPEN;
(25)          insert gs' into VISITED;
(26)        }
(27)      }
(28)    }
(29)    if (SU(vstart) = MAX) then
```

```
(30)      SU(vstart) ← "no UIO sequence that have cost less than COST(MAX) for this state";
(31)  }
(32)  return ∀vstart ∈ V, SU(vstart);
```

Algorithm 1の計算量

(1)-(31)は最大n回繰り返される．(4)-(28)はOPENがemptyになるまで要素が繰り返し取り除かれる．OPENへ新たな要素が追加されるのは最大$p_{max}n2^{n-1}$回である（状態数×$n-1$個の状態の部分集合数の組合せが$n2^{n-1}$，一つの組み合わせにつきp_{max}回の置換で最小コストのvertexが求められると仮定した）．(5)は最大$\log_2(n2^{n-1})$ステップかかる．(6)-(27)は最大d_{max}回繰り返される．(8)のV_s'の生成はnステップかかる．(13)は最大nステップかかる．(15)は最大$\log_2(n2^{n-1})$ステップかかり，(24)，(25)は最大$\log_2(n2^{n-1})$ステップかかる．従って，Algorithm 1全体の計算量は以下のようになる．

$$O(np_{max}n2^{n-1}(\log_2(n2^{n-1})+d_{max}(n+\log_2(n2^{n-1})+n+\log_2(n2^{n-1})+\log_2(n2^{n-1}))))$$
$$=O(n^3p_{max}d_{max}2^{n-1})$$

Algorithm 2

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

```
(1)   for each state vstart ∈ V do {
(2)     put a vertex gstart = <vstart, V−{vstart}, null> into a list OPEN and a list VISITED;
(3)     SUP(vstart, v) ← SU(vstart) · SP(TAIL(SU(vstart)), v);
(4)     while (OPEN is not empty) do {
(5)       remove the vertex gs = <vs, Vs, s>, which have minimum-cost sequence s, from OPEN;
(6)       for each transition of the form (vs, vd; ak/ol) ∈ E do {
(7)         s' ← s · ak/ol;
(8)         Vs' ← {vq | vp ∈ Vs and (vp, vq; ak/ol) ∈ E};
(9)         gs' ← <vd, Vs', s'>;
(10)        if (Vs' = φ) then {    /* 真なら s' はUIO系列である */
(11)          if (COST(s') + COST(SP(vd, v)) < COST(SUP(vstart, v))) then
(12)            SUP(vstart, v) ← s' · SP(vd, v);
(13)        } else if (vd ∈ Vs') then {    /* 真なら s' はUIO系列になりえない */
(14)          continue;
(15)        } else if (∃gt = <vt, Vt, t> ∈ VISITED such that vt = vd and Vt = Vs') then { /* 真
                    なら s' と同じ状態判別能力をもつ遷移系列がみつかっている */
(16)          if (COST(s') < COST(t)) then {    /* 偽なら s' の先を探索しても最小コスト
                                                   のUIO系列になりえない */
(17)            replace gt with gs' in VISITED;
(18)            if (gt exist in OPEN) then
(19)              remove gt from OPEN;
(20)            insert gs' into OPEN;
(21)          }
(22)        } else {
(23)          if (COST(s') + COST(SP(vd, v)) < COST(SUP(vstart, v))) then /* 偽なら s' の
                                先を探索しても最小コストのUIO経路になりえない */
(24)            insert gs' into OPEN;
(25)          insert gs' into VISITED;
(26)        }
(27)      }
(28)    }
(29)  }
(30)  return ∀vstart∈V, SUP(vstart, v);
```

Algorithm 2の計算量

Algorithm 1と同様に考えると，Algorithm 2全体の計算量は以下のようになる．

$$O(np_{max}n2^{n-1}(\log_2(n2^{n-1})+d_{max}(n+\log_2(n2^{n-1})+n+\log_2(n2^{n-1})+\log_2(n2^{n-1}))))$$
$$=O(n^3p_{max}d_{max}2^{n-1})$$

Algorithm 3

Maximum-cost useless sequence generation algorithm.

```
(1)   put a vertex hstart = <v0, Eunorg, null> into a list OPEN and a list VISITED;
(2)   max_cost ← −COST(SU(v0));
(3)   hdel ← <v0, Eunorg, null>;
(4)   while (OPEN is not empty) do {
(5)     remove a vertex hs = <vs, Es, s>, which have maxmum-cost sequence s, from OPEN;
(6)     if (∃ (vd, vs; ak/ol) ∈Es) then { /* 真なら状態 vs から逆方向にたどれる遷移がある
                                             偽なら逆方向にたどれる遷移がない */
(7)       for each transition of the form (vd, vs; ak/ol) ∈ Es do {
(8)         s' ← ak/ol · s;
(9)         Es' ← Es − {(vd, vs; ak/ol)};
(10)        hs' ← <vd, Es', s'>;
(11)        if (∃ht = <vt, Et, t> ∈ VISITED such that vt = vd and Et = Es') then {   /* 真なら
                  s' に含まれる全遷移と同じ遷移を含む遷移系列 t がすでに探索済み */
(12)          continue;
(13)        } else {        /* 状態 vd の先はまだ(逆方向に)探索する必要あり */
(14)          insert hs' into OPEN;
(15)          insert hs' into VISITED;
(16)        }
(17)      }
(18)    } else {
(19)      if (max_cost < COST(s)−COST(SU(vs))) then {     /* 真ならこれまでの中で最大
                                                            コストの遷移系列である */
(20)        max_cost ← COST(s)−COST(SU(vs));
(21)        hdel ← hs;
(22)      }
(23)    }
(24)  }
(25)  return vdel and sdel in hdel = <vdel, Edel, sdel>;
```

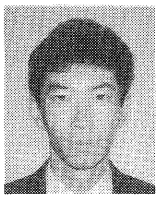
Algorithm 3 の計算量

(4)-(24)は OPEN が empty になるまで要素が繰り返し取り除かれる．OPEN へ新たな要素が追加されるのは最大 $p_{\max} n 2^{m_0}$ 回である（状態数×m_0 個の遷移の部分集合数の組合せが $n2^{m_0}$，一つの組合せにつき $p_{\max}$ 回の置換で最大コストの vertex が求められると仮定した）．(5)は最大 $\log_2(n2^{m_0})$ ステップかかる．(6)は最大 m_0 ステップかかる．(7)-(17)は最大 $d_{\max}$ 回繰り返される．(9)の E_s' の生成は最大 m_0 ステップかかる．(11)，(14)，(15)は最大 $\log_2(n2^{m_0})$ ステップかかる．

従って，Algorithm 3 全体の計算量は以下のようになる．

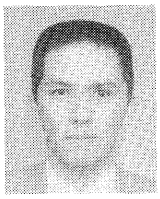
$$O(p_{\max} n 2^{m_0}(\log_2(n2^{m_0}) + m_0 + d_{\max}(m_0 + \log_2(n2^{m_0}) + \log_2(n2^{m_0}) + \log_2(n2^{m_0}))) = O(np_{\max} d_{\max}(\log_2 n + m_0)2^{m_0})$$

(平成5年7月 7 日受付)
(平成7年3月13日採録)



高木 浩則（正会員）

1991年茨城大学大学院工学研究科修士課程情報工学専攻修了．同年日本電信電話(株)に入社．ソフトウェア研究所に配属．主にソフトウェアのテスト支援技術の研究開発に従事．現在，情報システム本部において社内情報システムの開発に従事．



橋本 辰範（正会員）

1988年九州工業大学大学院修士課程修了．同年日本電信電話(株)に入社．ソフトウェア研究所に配属．現在はグループ事業推進本部事業企画部に所属．主にソフトウェア開発支援技術とインターネットアプリケーションの研究開発に従事．