マルチメディア通信と分散処理　52－36
（1991．9．27）

通信システムにおける相互接続試験系列生成法

岡崎　直宣† 高橋　薫†† 白鳥　則郎†† 野口　正一††

†㈱高度通信システム研究所

††東北大学

あらまし　通信システムの相互接続試験における試験系列の生成の手法を提案する．本手法では，相互接続された2つのシステムの動作を規定するプロトコル仕様より，パータベーションの手法を応用して試験系列を生成する．さらに，ここではチャネルにエラーがある場合を想定したプロトコル仕様に対しても適用できるよう本手法を拡張する．そして，従来の適合性試験に対する試験系列生成法が適用できる形に導き，これらの手法を用いて試験系列を生成する．本手法を用いることによって，相互接続試験における試験系列生成のコストの削減，試験の信頼性の向上などが期待される．

Test Sequence Generation Method for Interoperability Testing

Naonobu OKAZAKI† Kaoru TAKAHASHI†† Norio SHIRATORI†† Shoichi NOGUCHI††

†Advanced Intelligent Communication System Lab.,
112-1, Gongenmoriyama, Imozawa, Aoba-ku, Sendai, 989-32 Japan

††TOHOKU UNIVERSITY,
Aoba-ku, Sendai, 980 Japan

Abstract　In this paper we propose a test sequence generation method for interoperability testing. In the proposed method, first, we apply the traditional perturbation method to a protocol specification that defines the behaviour of the interconnected two systems. Second, we expand the proposed method so that it can apply to a specification that can considers errors in channels. Third, we lead the result from the second stage to the form that is applicable to traditional conformance testing for a test sequence generation method. We propose a test sequence generation using these methods. By using the proposed method, it is expected to decrease the costs of test sequence generation and to enhance the reliability of testing.

1. はじめに

情報通信システムなどの分散処理システムの普及拡大に伴い，そのソフトウェアも大規模化，複雑化，多様化する傾向にある．このような状況において，開発された製品に対する試験の重要性がますます高まっている．

製品試験は一般にブラックボックス試験(試験対象である製品の内部構造については一切関知せず，その外部に現れる振る舞いのみを調べること)によって行なわれる．すなわち，まず元の仕様から予測されうる入出力の系列を取り出す．これは“試験ケース”または“試験項目”と呼ばれる．試験ケースの集合は“試験スイート”あるいは“試験系列”と呼ばれる．この試験系列を基に，製品にどのような入力を与えたとき，どのような出力が得られるかを調べることにより試験が行われる．

試験に関する問題の中で主要な部分を占めるものの1つが，試験系列の生成に関する問題である．これは，現状では人手で行われており，試験の信頼性やコストの面で問題があった．

現在，通信システムを対象とした製品試験は，個々の製品に対して個別に行う“適合性試験”と，実利用環境のもとで実際に製品同志を接続して行う“相互接続試験”との組合せで実施されている．

このうち，適合性試験については，その方法と枠組みの標準化がISOで進められており[1]，また，試験系列の生成に関する研究もいくつか行なわれている[2~5,9]．一方，相互接続試験については，統一的な方法論は依然として確立されておらず，特にその試験系列の生成に関する研究はほとんど行なわれていないのが現状である．

本論文では，相互接続試験における試験系列の生成の手法を提案する．本手法では，双方向のチャネルにより相互接続された2つのシステムの動作を規定する，有限状態機械(FSM:Finite State Machine)により記述されたプロトコル仕様より，パータベーションの手法[7]を応用して試験系列を生成する．さらに，ここではチャネルにエラーがある場合を想定したプロトコル仕様に対しても適用できるよう本手法を拡張する．そして，最後に，従来の適合性試験に対する試験系列生成法が適用できる形に導き，これらの手法を用いて試験系列を生成する．本手法を用いることによって，相互接続試験における試験系列生成のコストの削減，試験の信頼性の向上などが期待される．

以下，本論文では，2.で試験対象システムの環境のモデル及び相互接続試験のアーキテクチャについて述べる．そして，本論文で提案する相互接続試験系列生成手法については3.で述べる．4.では本手法の適用例を示す．5.はまとめである．

2. 試験対象システムの環境のモデル及び相互接続試験のアーキテクチャ

本手法では，図1のような相互接続試験における被試験システムの環境をモデルとする．すなわち，被試験システムとしては，第N層の通信し合う任意のプロトコルエンティティ，N-Entity1およびN-Entity2を考え，N-Entity1，N-Entity2とそれぞれ(N)SAP(Service Access Point)を通して接続される上位層をそれぞれユーザ1，ユーザ2と呼ぶ．さらに，N-Entity1およびN-Entity2はそれぞれ(N-1)SAPを通して双方向のFIFOチャネルに接続されている．但し，ここではこのチャネルが不完全な場合をも考慮する．具体的には，このチャネルは原則的には入力したデータを反対側に入力順に出力するが，任意のデータが紛失する可能性があるものとする．

また，本手法で対象とする相互接続試験の環境のモデルは図2のようになる．被試験システムである2つのプロトコルエンティティN-Entity1およびN-Entity2をIUT(Implementation Under Test)1，IUT2と呼び，それぞれ上位，下位のPCO(Point of Control and Observation)を通して上位テスタ，下位テスタとデータのやり取りをする．また，2つの下位テスタは，(N-1)層以下のサービス提供者を通して互いに通信する．ただし，ここで受ける(N-1)サービスは完全なものを仮定する．以降，IUTiの上位テスタをUTi，下位テスタをLTi，またUTiとIUTiの間のPCOをUi，LTiとIUTiの間のPCOをLiとそれぞれ呼ぶ(i=1,2)．

本手法で対象とする相互接続試験の試験系列とは，4つのPCO，U1，U2，L1，L2においてテスタがどの順番でどういったデータをやり取りするかを記述したものであるものとする．

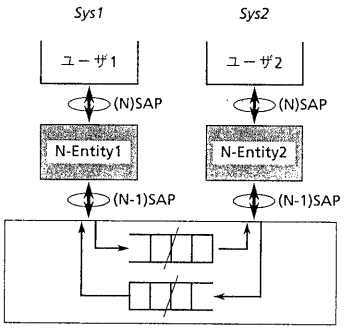

図1 被試験システムの環境のモデル

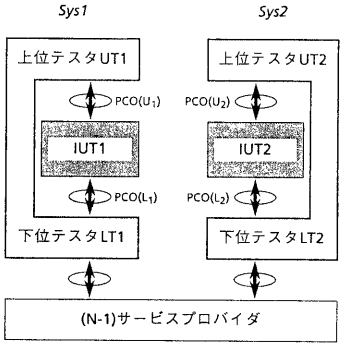

図2 相互接続性試験環境のモデル

3. 相互接続試験系列生成手法

3.1 システム状態グラフの導入

本論文では，次のように定められる，通信し合う2つのFSMでモデル化されたプロセスから成るプロトコルを対象とする．

【 定義1 】プロトコルP

$P=<P_1,P_2>$

ここで，

P_i：プロセスi (i=1,2)

$P_i=<Q_i,A_i,T_i,o_i,F_i>$

Q_i：プロセスiの状態の集合

A_i：プロセスiの送受信アクションの集合

$A_i=\{pd \mid p\in SAP_i, d\in D\}$

SAP_i：プロセスiのSAPの集合

$d\in D$ は $-m_o$ 又は $+m_i$ の形

但し，$m_o\in M_{ij}$, $m_i\in M_{ji}$,

M_{ij}：プロセスiからプロセスjへ送り得るメッセージの集合 (j=1,2, i≠j)

T_i：プロセスiの状態遷移を表す関係

$T_i\subseteq Q_i\times A_i\times Q_i$

o_i：プロセスiの初期状態

F_i: プロセスiの最終状態の集合　□

e=p−m∈A_iのとき，eを入力アクションと呼び，e=p+m∈A_iのとき，eを出力アクションと呼ぶ．

プロトコルが上の定義のように2つのFSMとして表されるプロセスから成るとき，各プロセスについて，それぞれに従来の適合性試験の試験系列生成法を適用することにより，2つの試験系列を得ることができる．ところが，相互接続試験においてはこの独立に得られた2つの試験系列中の各アクション間の前後関係が重要である．例えば，2つの試験系列中のあるアクション系列が$a_1b_1c_1$および$d_2e_2f_2$であった時，b_1はa_1の後でc_1の前に起こることは分かるが，b_1とd_2, e_2, f_2との前後関係の情報が欠けているため，この2つの系列からは相互接続試験を行うことができない．この様な問題を相互接続試験における2つのプロセスの同期の問題と呼ぶ．

そこで，本論文では，この同期の問題を解決するために，従来プロトコルの論理エラーの検出法などに用いられているパータベーションの手法[7]を応用し，システム状態グラフ(SSG)を生成する．SSGは，次に定められるような，2つのプロセスの状態及びその間のチャネルの状態を合成して得られるシステム状態をノードとするグラフである．尚，各チャネルはFIFO(First In First Out)であるものとする．

【 定義2 】システム状態s_k
$s_k=<q_1^k,q_2^k,c_{12}^k,c_{21}^k>$
但し，
q_i^k: プロセスiの状態　$q_i^k \in Q_i$
c_{12}^k: プロセス1からプロセス2へのチャネルの内容　$c_{12}^k \in M_{12}^*$
c_{21}^k: プロセス2からプロセス1へのチャネルの内容　$c_{21}^k \in M_{21}^*$
□

【 定義3 】SSG　$G_S=<S,A,Tr,s_0>$
ここで，
S: システム状態の集合
A: 送受信アクションの集合
Tr: 遷移関係
$Tr \subseteq S \times A \times S$
s_0: 初期システム状態　□

2つのプロセスからなるプロトコル$P=<<Q_1,A_1,T_1,o_1,F_1>,<Q_2,A_2,T_2,o_2,F_2>>$が与えられたとき，このSSGを生成するルールを次のruleA～ruleIとする．このうち，ruleA及びruleD～ruleGは従来のパータベーションの手法によるものと同じであり，ruleB，ruleCはそれぞれプロセス1，プロセス2とユーザとのやりとりによる遷移を拡張したものである．

【 定義4 】SSG生成規則
プロトコル$P=<<Q_1,A_1,T_1,o_1,F_1>,<Q_2,A_2,T_2,o_2,F_2>>$が与えられたとき，$A=A_1 \cup A_2$と次のrule A～rule Gにより推論される最小の集合S,Tr及び初期システム状態s_0からなるSSG $G_S=<S,A,Tr,s_0>$をPに関するSSGと定める．

以下のruleにおいて，Eはチャネルの内容が空であることを示す．また，append(c_{ij},m), top(c_{ij}), remain(c_{ij})をそれぞれ次のように定める．

append(c_{ij},m)　:
　チャネルc_{ij}の最後にmを加えた状態のチャネル．
top(c_{ij})　:
　チャネルc_{ij}の先頭の内容．チャネルc_{ij}が空の場合には定義されない．
remain(c_{ij}):
　チャネルc_{ij}の先頭の内容を取り去った残りの状態のチャネル．チャネルc_{ij}が空の場合には定義されない．

rule A (初期システム状態)
$\vdash s_0=<o_1,o_2,E,E> \in S$

rule B (U_1での入出力)
$s=<q_1,q_2,c_{12},c_{21}> \in S$, $(q_1,U_1d,q_1') \in T_1$, $U_1d \in A_1$
$\vdash$ $(s,U_1d,s') \in Tr$, $s' \in S$
　但し，$s'=<q_1',q_2,c_{12},c_{21}>$

rule C (U_2での入出力)
$s=<q_1,q_2,c_{12},c_{21}> \in S$, $(q_2,U_2d,q_2') \in T_2$, $U_2d \in A_2$
$\vdash$ $(s,U_2d,s') \in Tr$, $s' \in S$
　但し，$s'=<q_1,q_2',c_{12},c_{21}>$

rule D (L_1での出力)
$s=<q_1,q_2,c_{12},c_{21}> \in S$, $(q_1,L_1-m,q_1') \in T_1$, $L_1-m \in A_1$
$\vdash$ $(s,L_1-m,s') \in Tr$, $s' \in S$
　但し，$s'=<q_1',q_2,\mathrm{append}(c_{12},m),c_{21}>$

rule E (L_2での出力)
$s=<q_1,q_2,c_{12},c_{21}> \in S$, $(q_2,L_2-m,q_2') \in T_2$, $L_2-m \in A_2$
$\vdash$ $(s,L_2-m,s') \in Tr$, $s' \in S$
　但し，$s'=<q_1,q_2',c_{12},\mathrm{append}(c_{21},m)>$

rule F (L_1での入力)
$s=<q_1,q_2,c_{12},c_{21}> \in S$, $\mathrm{top}(c_{21})=m$,
$(q_1,L_1+m,q_1') \in T_1$, $L_1+m \in A_1$
$\vdash$ $(s,L_1+m,s') \in Tr$, $s' \in S$
　但し，$s'=<q_1',q_2,c_{12},\mathrm{remain}(c_{21})>$

rule G (L_2での入力)
$s=<q_1,q_2,c_{12},c_{21}> \in S$, $\mathrm{top}(c_{12})=m$,
$(q_2,L_2+m,q_2') \in T_2$, $L_2+m \in A_2$
$\vdash$ $(s,L_2+m,s') \in Tr$, $s' \in S$
　但し，$s'=<q_1,q_2',\mathrm{remain}(c_{12}),c_{21}>$　□

ところで，プロトコルの仕様によっては，これらのルールの適用が有限回で終了しない場合があり得る．例えば，プロセスP_1の中に，ある状態からL_1で出力を出して同じ状態に戻る遷移が定義されていると，rule Dが無限回適用できる．以下，本論文ではこれらのルールの適用が有限回で終了する場合のみを対象とする．従って，以下では，SSGは有限グラフとして扱う．

SSG上において，初期状態からグラフ上のラベルを追うことによって，Pを構成する2つのプロセスを合成したシステム全体の各アクションの系列が得られる．このアクション系列にはシステム全体の各アクション間の前後関係が明確に表されている．これによって，同期の問題が解決される．

例えば，プロトコルPを構成するプロセスP_1, P_2より得られるアクションの系列がそれぞれ$a_1b_1c_1$および$d_2e_2f_2$であり，さらに上のSSGより得られるアクション系列が$a_1d_2e_2b_1f_2c_1$であるものとすると，アクションb_1はd_2，e_2の後に起こり，またf_2の前に起こることがわかる．

3.2 システム状態グラフの拡張

前節の手法によって，2つのプロセス間の通信路が完全な場合(つまり送ったデータは必ず正しく届く場合)を想定したプロトコルでは，各プロセスの全てのアクションがSSGに現れる(完全性)為，全てのアクション間の同期の問題が解決する．

ところが，2つのプロセス間の通信路が完全でない場合を想定したプロトコル，つまり，誤り制御機能をもったプロトコル等においては，その全てのアクションがSSG上に現れない．例えば，後述の図7のようなプロトコルでは，同図(a)における状態4から状態5への"T+timeout"でラベル付けされた遷移は，チャネル内でデータの消失があった場合を想定しており，前節の手法だけではこの遷移を含む正常な試験系列を求めることができない．

そこで，ここでは通信路の誤りのうちデータ損失を対象とし，この誤りがある場合においても完全性のあるSSGを生成することができるようにSSG生成ルールを拡張する．

すなわち，ここで前述のruleA～ruleIに次のruleJ，ruleKを加えたものを新たに拡張SSG(ESG)生成規則とする．

【 定義5 】ESG生成規則

プロトコルP=<<Q_1,A_1,T_1,o_1,F_1>, <Q_2,A_2,T_2,o_2,F_2>>が与えられたとき，A=$A_1 \cup A_2 \cup \{\tau\}$とrule A～rule G及び次のruleH，ruleIにより得られる最小の集合S,Tr及び初期システム状態s_0からなるSSG G_E=<S,A,Tr,s_0>をPに関するESGと定める．

rule H(チャネル12中のデータ消失)

$s=<q_1,q_2,c_{12},c_{21}> \in S$, $c_{12} \neq \emptyset$

$\vdash$ $(s,\tau,s') \in Tr$, $s' \in S$

但し，$s'=<q_1,q_2,remain(c_{12}),c_{21}>$

rule I(チャネル21中のデータ消失)

$s=<q_1,q_2,c_{12},c_{21}> \in S$, $c_{21} \neq \emptyset$

$\vdash$ $(s,\tau,s') \in Tr$, $s' \in S$

但し，$s'=<q_1,q_2,c_{12},remain(c_{21})>$ □

ここで，τはチャネル中のデータが消失したことを示す空アクションである．試験実施時には，テスタはこのアクションを起こすか他の入力を与えるかを決定できるものとする．

3.3 相互接続試験系列の導出

本節では，**3.2**で得られたESGより，相互接続試験系列を導出する方法について述べる．

適合性試験に関する試験系列の生成法に関して，従来行われてきた研究として，TT法[2]，PW法[3]，DS法[4]，UI法[5]等がある．これらはいづれもMealy型FSMをモデルとし，入力に一定の制限をおいた上で，Mealy型FSMの重要な性質である出力関数および遷移関数の一方又は両方を調べることができる手法である[6]．

本論文で対象とする相互接続試験においても，これらの手法を応用して有用な試験系列を得ることができると考えられる．ところが，**3.2**で得られたESGは，Mealy型FSMとは異なるため，これに"直接"上記の手法を適用することはできない．そこで，ここではESGから，上記の手法の制限のうち共通的なものである「完全定義」「最小」「強連結」の3つの条件を満たしているようなMealy型FSMを導出し，このMealy型FSMに上記の手法の一つを適用して試験系列を導出する方法を提案する．

以下に，ESGからMealy型FSMを導出する方法を示し，Mealy型FSMの最小化の方法を適用し，さらに簡単な仮定をおくことによりこのMealy型FSMが最小，完全定義及び強連結の各条件を満たすことを述べる．

ESGとMealy型FSMの異なる主要な点は，入出力の区別に関してである．すなわち，前者が入力と出力を形の上で区別せず，どちらも一つのアクションとしてとらえているのに対して，後者は入力と出力を区別し，ある状態からの遷移は入力によってはじめて起こり，その遷移の過程で出力を行うという形で表されることである．つまり，Mealy型FSMの上では現れない状態(例えば，入力を行った後でかつ出力をする前の状態など)が，ESG上に存在することになる．そこで，ESGからMealy型FSMを導出するためには，基本的には，前者の状態のうち後者に存在するもののみを取り出して(これを，ここでは状態の抽出と呼ぶ)，グラフを再構成すればよい．

本手法では，次のようにしてESGから状態を抽出する．このようにして導出されたグラフをここでは状態遷移グラフ(STG)と呼ぶ．

【 定義6 】STG

ESG G_Eが

G_E=<S,A$\cup\{\tau\}$,Tr,s_0>

のとき，STG G_Fを次のように定める．

G_F = < $S_{exp} \cup S_{imp}$, A$\cup\{\tau\}$, Tr', s_0 >

ここで，

S_{exp} = {s | $\exists$s' (s, p+m, s')$\in$Tr, s$\in$S}

S_{imp} = {imp(s) | s$\in$S', S'=S$-S_{exp}$}

ただし，imp(s)はsに対応するimplicitな状態．

Tr' $\subseteq$ ($S_{exp} \cup S_{imp}$)$\times$(A$\cup\{\tau\}$)$\times$($S_{exp} \cup S_{imp}$)

Tr' = {(s,e,s') | (s,e,s')$\in$Tr, s,s'$\in S_{exp}$}

$\cup$ {(s,e,imp(s')) | (s,e,s')$\in$Tr, s$\in S_{exp}$, s'$\in$S'}

$\cup$ {(imp(s),e,s') | (s,e,s')$\in$Tr, s$\in$S', s'$\in S_{exp}$}

$\cup$ {(imp(s),e,imp(s')) | (s,e,s')$\in$Tr, s,s'$\in$S'} □

なお，implicitでない状態をexplicitな状態と呼ぶ．

STG G_F = < $S_{exp} \cup S_{imp}$, A$\cup\{\tau\}$, Tr', s_0 > において，(s_1,e,s_2)$\in$Tr'であるとき，これを"$s_1 \text{—} e \rightarrow s_2$"と表す．

一般に，このSTGにはimplicitな状態における合流及び分岐が存在する．一方，Mealy型FSMは入力の前を状態と見なすので，implicitな状態における合流及び分岐が存在しないSTG(このようなSTGを標準形STG(NSG)と呼ぶ)であればこれをMealy型FSMに変換しやすい．そこで，一般のSTGを標準形に変形する．以下にその方法を示す．変形は，STG上に存在するimplicitな状態における合流及び分岐に対して，それぞれ図3及び図4に示す方式に従って行う．すなわち，図3(a)のようなimplicitな状態における合流が存在した場合，新たな(implicitな)状態を設け，合流状態を複数に分ける(同図(b))．このような変形をimplicitな状態における合流が存在しなくなるまで繰り返す．明らかに，上記の一回の変形においてimplicitな状態の数は同じかまたは一つ減少し，またSTG上のimplicitな状態の数は有限であるので，この変形の繰り返しは有限回で終了する．同様に，図4(a)のようなimplicitな状態における分岐が存在した場合，新たな(implicitな)状態を設け，分岐状態を複数に分ける(同図(b))．このような変形をimplicitな状態における分岐が存在しなくなるまで繰り返す．上と同様に，この場合も変形の繰り返しは有限回で終了する．

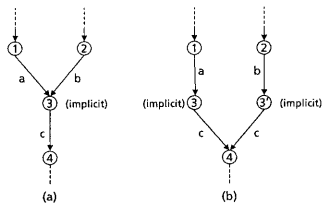

図3 implicitな状態における合流の変形

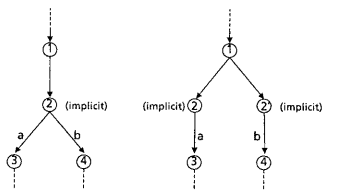

図4 implicitな状態における分岐の変形

以上の様にして得られるNSGにおいて，次の記法を定める．

すなわち，NSG G_N = < $S_{exp} \cup S_{imp}$', A$\cup\{\tau\}$, Tr", s_0 >において任意の2つのexplicitな状態s_0, $s_n \in S_{exp}$を結ぶ遷移の系列

$s_0 \text{—} a_1 \rightarrow s_1$, $s_1 \text{—} a_2 \rightarrow s_2$, …, $s_{n-1} \text{—} a_n \rightarrow s_n$

(ただし，s_1, …, $s_{n-1} \in S_{imp}$, n$\geqq$1)

が存在するとき，これを

$s_1 = a_1 a_2 \cdots a_n \Rightarrow s_2$

と表す．

NSGは，STGの作り方より，各explicitな状態から入力のラベルを持った遷移が存在することが，また上記の変形により，分岐

はexplicitな状態のみで起こることが，それぞれ保証されている．そこで，NSG $G_N = < S_{exp} \cup S_{imp}', A \cup \{\tau\}, Tr'', s_0 >$ 上の各explicitな状態をMealy型FSMの状態に対応させることにより，次のようにしてMealy型FSMを導出する．

【 定義7 】Mealy型FSM

$M_M = <Q, I \cup \{\tau\}, O, \omega, \delta>$

ここで，

$Q = S_{exp}$

$I = \{e | e \in A, e$は入力アクション$\}$

$O = \{e | e \in A, e$は出力アクション$\}$

ω: 出力関数　　$Q \times I \to O^*$

ここで，　O^*は出力系列の集合

$\omega(s_1, a_1) = a_2 a_3 \cdots a_n$

$(\exists s_2\ s_1 = a_1 a_2 \cdots a_n \Rightarrow s_2,\ a_1 \in I,\ a_2 \cdots a_n \in O,\ n \geqq 2)$

$\omega(s_1, a_1) = \varepsilon \quad (\exists s_2\ s_1 \text{—} a_1 \to s_2,\ a_1 \in I)$

$\omega(s_1, \varepsilon) = a_1 a_2 \cdots a_n\ (\exists s_2\ s_1 = a_1 a_2 \cdots a_n \Rightarrow s_2,\ a_1 \cdots a_n \in O,\ n \geqq 1)$

δ: 遷移関数　　$Q \times I \to Q$

$\delta(s_1, a_1) = s_2 \quad (\exists s_2\ s_1 = a_1 a_2 \cdots a_n \Rightarrow s_2,\ a_1 \in I,\ a_2 \cdots a_n \in O,\ n \geqq 1)$

$\delta(s_1, \varepsilon) = s_2 \quad (\exists s_2\ s_1 = a_1 a_2 \cdots a_n \Rightarrow s_2,\ a_1 \cdots a_n \in O,\ n \geqq 1)$ □

すなわち，NSG $G_N = < S_{exp} \cup S_{imp}', A \cup \{\tau\}, Tr'', s_0 >$ 上の各explicitな状態をMealy型FSMの状態に，入力アクションの集合をIに，出力アクションの集合をOにそれぞれ対応させる．さらに，隣り合う2つのexplicitな状態を結ぶアクションの系列より，Mealy型FSMの遷移が得られる(図5)．ただし，出力のラベルを持った遷移の存在する状態(混合状態)においては，そのような遷移の先頭に空の入力εを加える(図6)．

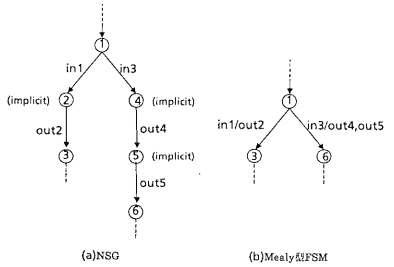

図5　NSGからのMealy型FSMの導出

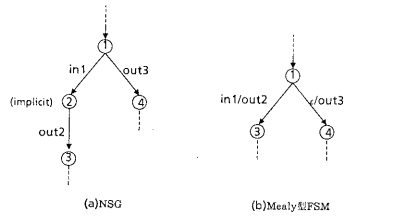

図6　混合分岐の場合のMealy型FSMの導出

以上のようにして得られたMealy型FSMは，FSMの最小化アルゴリズムを適用することによって，最小のものが得られる[8]．また，後に図9に示すように，定義されない入力に対しては，空出力を出して同じ状態に戻る遷移を仮定することによって，このMealy型FSMは完全定義であると見なすことができる．また，各状態について，初期状態に戻るためのリセット系列を仮定することにより，強連結の条件を満たすと考えることができる．

これらのことから，上で導出されたMealy型FSMが，従来の試験系列の生成法であるTT法，PW法，DS法，UI法に共通する制限条件を満たすことが示された(ただし，DS法に関してはさらに判定系列が存在することが必要)．

そこで，このMealy型FSMについて，これらの手法の一つを適用して試験系列を得ることができる．本論文では，**4.**でUI法を適用した例を示す．UI法は，UIO(Unique Input and Output)シーケンスと呼ばれる各状態をユニークに識別する系列をもとに，各遷移についてその出力と遷移先状態を調べる手法である．

4. 適用例

以上のことを実際の仕様例に適用した例を示す．図7のような仕様を考える．

この仕様例において，例えば，プロセスP2の“L2+mess”でラベル付けされた状態1から状態2への遷移に注目する．これに対して，**3.2**で示したESG生成規則のrule Gを適用することにより，システム状態<4,1,mess,E>から<4,2,E,E>への“L2+mess”でラベル付けされた遷移が推論される．このようにして図7の仕様から，ESG生成規則を適用することにより図8のSSGが得られる．同図において，破線で示される遷移は，**3.2**で拡張したルールによって推論される遷移である．

次に，図8のSSGにおいて，下線を引いた状態がexplicitな状態であり，他の状態がimplicitな状態であると考えることにより，これは**3.3**で示した方法により状態の抽出をしたSTGであると見なすことができる．ここで，例えば同図の状態<4,1,mess,E>は，入力ラベル“L2+mess”を持った遷移が存在するため，explicitな状態となる．一方，状態<4,2,E,E>は，出力ラベル“U2−mess”を持った遷移のみ存在し，入力ラベルを持った遷移が存在しないため，implicitな状態となる．

ここで，図8のimplicitな状態<5,1,E,E>を見ると，状態<4,1,E,E>からの遷移と状態<4,1,E,nak>からの遷移の，2つの遷移の合流が存在する．そこで，図3の方法によりこれを変形する．このようにして**3.3**で示した方法により図8のSTGを変形し，図9の標準形を得ることができる．ただし，同図においてimplicitな状態については省略している．

さらに，図9中の

<4,1,mess,E>—L2+mess→<4,2,E,E>,

<4,2,E,E>—U2−mess→<4,3,E,E>

(ただし，図中で状態<4,2,E,E>は省略)

の遷移の系列から，Mealy型FSMにおける

<4,1,mess,E> —L2+mess / U2−mess→ <4,3,E,E>

という遷移が導かれる．ここで，“$s_1 \text{—} e_i / e_o \to s_2$”は，入力$e_i$によって状態$s_1$から出力$e_o$を出して状態$s_2$へ移る遷移を表す．このようにして図9のNSGから導出されるMealy型FSMを最小化したものは図10のようになる．ここで，図中の“-”は，空出力を出して同じ状態に戻る遷移を表す．

最後に，このMealy型FSMについて，従来の試験系列の生成法の一つを適用する．ここでは，UI法を適用した例を示す．UIOシーケンスは図11のようになる．ここで，例えば同図の5列目より，入力“L1+ack”を与えたときの出力が“U1−success”であれば，この入力を与える前の状態は“5”であることがわかる．このUIOシーケンスをもとに，図12のような試験系列が得られる．例えば同図の3列目は，図12の状態2(<4,3,E,E>)から入力“U2+nak”による状態4(<4,1,E,nak>)への遷移を調べる系列を表す．ここで，“U1+mess,L2+mess”の部分は初期状態から状態2への遷移を，“U2+nak”の部分は目的の遷移をそれぞれ表す．また，“L1+nak”の部分は目的の遷移の遷移先状態を調べるUIOシーケンスである．

5. まとめ

本論文では，相互接続試験における試験系列の生成の手法を提案した．ここでは，パータベーションの手法を応用し，システム状態グラフを導入して，2つのFSMとして記述されたプロトコル仕様より試験系列を生成した．そして，このことにより，相互接続

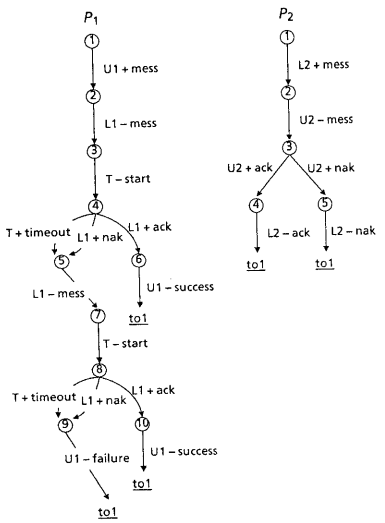

図7　プロトコル例

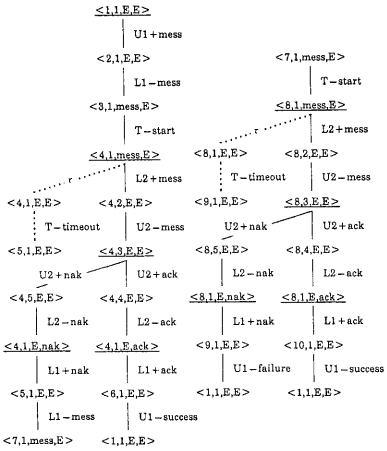

図8　SSG, STG例

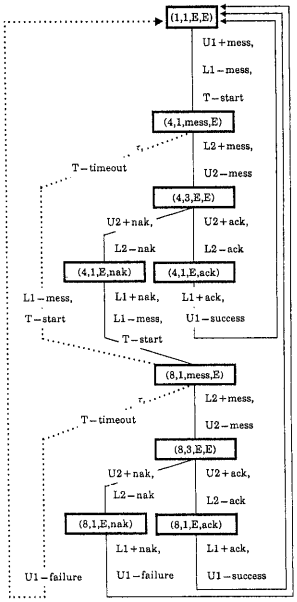

図9　NSG

状態\入力	U1+mess	L2+mess	U2+ack	U2+nak	L1+ack	L1+nak	r
1 <1,1,E,E>	L1−mess, T−start / 7	-	-	-	-	-	-
2 <4,3,E,E>	-	-	L2−ack / 5	L2−nak / 4	-	-	-
3 <8,3,E,E>	-	-	L2−ack / 5	L2−nak / 6	-	-	-
4 <4,1,E,nak>	-	-	-	-	-	L1−mess, T−start / 8	-
5 <4,1,E,ack>, <8,1,E,ack>	-	-	-	-	U1−success /1	-	-
6 <8,1,E,nak>	-	-	-	-	-	U1−failure /1	-
7 <4,1,mess,E>	-	U2−mess / 2	-	-	-	-	T-timeout, L1-mess, T-start / 8
8 <8,1,mess,E>	-	U2−mess / 3	-	-	-	-	T-timeout, U1-failure / 1

図10　Mealy型FSMの最小形

state	UIO
1	U1+mess / L1−mess,T−start
2	U2+nak / L2−nak L1+nak / L1−mess,T−start
3	U2+nak / L2−nak L1+nak / U1−failure
4	L1+nak / L1−mess,T−start
5	L1+ack / U1−success
6	L1+nak / U1−failure
7	r / T-timeout,L1-mess,T-start
8	r / T-timeout,U1-failure

図11　UIOシーケンス

r,U1 + mess,r
r,U1 + mess,L2 + mess,U2 + ack,L1 + ack
r,U1 + mess,L2 + mess,U2 + nak,L1 + nak
r,U1 + mess,r,L2 + mess,U2 + ack,L1 + ack
r,U1 + mess,r,L2 + mess,U2 + nak,L1 + nak
r,U1 + mess,L2 + mess,U2 + nak,L1 + nak,r
r,U1 + mess,L2 + mess,U2 + ack,L1 + ack,U1 + mess
r,U1 + mess,r,L2 + mess,U2 + nak,L1 + nak,U1 + mess
r,U1 + mess,L2 + mess,U2 + nak,L1 + nak
r,U1 + mess,r,r
r,U1 + mess,r,L2 + mess,U2 + nak
r,U1 + mess,r,r,U1 + mess

図12　試験系列

試験における同期の問題が解決することを示した．また，チャネルにエラーがある場合を想定したプロトコル仕様に対しても適用できるようパータベーションの手法を拡張し，その有効性を示した．さらに，従来の適合性試験に対する試験系列生成法が適用できる形に導き，これらの手法を用いて試験系列を生成した．これによって，従来の技術を有効に利用し，相互接続試験における有効な試験系列の生成が行えると考えられる．本手法を用いることによって，相互接続試験における試験系列生成のコストの削減，試験の信頼性の向上などが期待される．

今後の課題としては，長さ，計算量などの効率に関することを含め，具体的に本手法が有効に適用可能なプロトコルの種別，規模，複雑さなどに関する研究の他，支援システムの構築等がある．

謝辞　本研究の機会を与えて下さいました高度通信システム研究所緒方常務に感謝いたします．

参考文献

(1) ISO : "OSI conformance testing methodology and framework", ISO 9646 (1989).

(2) S. Naito and M. Tsunoyama : "Fault detection for sequential machines by transition tours," in *Proc. IEEE Fault Tolerant Comput. Conf.*, 1981.

(3) T. Chow : "Testing software design modeled by finite-state machines," *IEEE Trans. Software Eng.*, vol. SE-4, pp. 178-187, 1978.

(4) G. Gonenc : "A method for the design of fault detection experiment," *IEEE Trans. Comput.*, vol. C-19, pp. 551-558, 1970.

(5) K. Sabnami and A. Dahbura : "A protocol test generation procedure", *Computer Networks ISDN System*, 15, pp. 285-297 (1988).

(6) D. P. Sidhu and T. Leung : "Formal method for protocol testing : a detailed study," *IEEE Trans. Software Eng.*, SE-15, 4, pp. 413-426 (1989).

(7) C. H. West : "General technique for communications protocol validation," *IBM J. Res. & Devel.*, vol. 22-4, pp. 394-404 (1978).

(8) 当麻喜弘："順序回路論"，昭晃堂(昭51).

(9) 岡崎，高橋，白鳥，野口："LOTOS仕様からの効率的な試験系列の生成法"，信学論(B-1)掲載予定.