並列性を考慮した通信システムの相互接続試験系列生成法

朴 美娘[†] 岡崎直宣[††] 三上節子[††]
高橋薫[†] 白鳥則郎[†††] 野口正一[†]

本論文では，複数のプロセスが通信し合うシステムの相互接続試験における試験系列の生成法（PP 法）を提案する．本手法では，試験実施時に並列に制御／観測動作が可能であることを陽に表すための方法として並列試験系列記述法（PTSN）を提案する．そして，システム全体の動作を表すシステム状態グラフをnプロセスに拡張し，システム状態グラフ上において各状態の確認をし，さらに入出力に伴う状態遷移とその遷移先状態とを確認する系列を生成することで，エラーの検出能力の高い試験系列を生成する．特に相互接続試験の特徴である並列性を考慮し，PTSNを用いることにより効率的な相互接続試験を行うことを可能とするような試験系列の生成を行うことを特長とする．本手法を用いることにより，相互接続試験の信頼性の向上，試験の実施時間の短縮によるコストの低減等が期待される．

Test Sequence Generation Method for Interoperability Testing Considering the Parallel Processes

Mi Rang Park,[†] Naonobu Okazaki,[††] Setsuko Mikami,[††]
Kaoru Takahashi,[†] Norio Shiratori[†††] and Shoichi Noguchi[†]

This paper proposes a test sequence generation method (PP method) for interoperability testing applicable to an n-process system. And we suggest a PTSN (Parallel Test Sequence Notation) which is able to parallel control and observe the behaviour in testing. In the proposed method, firstly, we generate the system state graph from a protocol specification which shows the global behaviour of the system. Secondly, for all transitions in this graph we generate sequences which confirm that the process implementations exactly transit to the correct system state. In particular, considering the process independency of system, we can construct test sequences which consist of parallel test events. Consequently it is expected to shorten the time 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．試験に関する問題の中で主要な部分を占めるものの一つが，試験系列の生成に関する問題である．

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士を接続して行う“相互接続試験”との組合わせで実施されている．このうち，適合性試験[1]については，試験系列の記述法としてTTCN (Tree and Tabular Combined Notation)[1]や形式記述言語により表す方法[12]等が提案されている．また試験系列の生成法としてFSM (Finite State Machine) を対象とした手法[2)~6)]や形式記述技法 (FDT : Formal Description Technique) を対象とした手法[12]がある．一方，相互接続試験[10]については，我々は二つのプロセスの場合の相互接続試験系列生成法について提案した[14]．

本論文では，文献14) の拡張としてnプロセスの場合を対象とした相互接続試験の試験系列を生成する手法を提案する．ここでは，まず，多数のプロセスの相互接続試験における試験時間を短縮するために，並列試験系列記述法 PTSN (Parallel Test Sequence Notation) を提案する．これは，従来のTTCN等による記述法では表すことができない，複数のプロセスで並列に制御／観測動作が可能であることを陽に表現す

† 東北大学電気通信研究所
Research Institute of Electrical Communication, Tohoku University
†† (株)高度通信システム研究所
Advanced Intelligent Communication System Lab.
††† 東北大学工学部情報工学科
Department of Information Engineering, Faculty of Engineering, Tohoku University

ることができることを特徴とする．そして，この記述法に基づいた試験系列を生成する手法である PP 法を提案する．本手法では，システム全体の動作を表すシステム状態グラフを n プロセスに拡張し，システム状態グラフ上において状態の識別を行うことを基本とした試験系列を生成する．特に相互接続試験の特徴である複数プロセスの独立性を考慮し，PTSN に基づいた試験系列の生成を行うことによって，並列動作を含む試験の実施が可能になる．

以下では，まず2章で，準備として試験対象システムのモデル化について述べる．次に，3章で並列試験系列記述法 PTSN を提案する．この記述法に基づいた試験系列を生成する手法である PP 法については4章で述べる．5章では，本手法の評価を与え，本手法を用いることによって，n プロセスの相互接続試験の効率が向上することを示す．

2. 試験システムのモデル化

本手法では，**図1**のように被試験システムの環境をモデル化する．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-Entity-1～N-Entity-n を考え，N-Entity-k (k=1, …, n) 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 k と呼ぶ．さらに，N-Entity-k はそれぞれ ($N-1$) SAP を通して双方向の FIFO (First In First Out) チャネルに接続されている．ここでは，次のように定められる，通信し合う n 個の 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 $P=\langle P_1, P_2, \cdots, P_n\rangle$

ここで，

P_k：プロセス k (k=1, …, n)

$P_k=<Q_k, I_k, O_k, \omega_k, \delta_k, q_{k0}>$

Q_k：プロセス k の状態の集合

I_k：プロセス k の入力アクションの集合

$I_k=\{p?m_i \mid p\in \mathrm{SAP}_k, m_i\in(\bigcup_{j=1}^{n} M_{jk})\cup M_{ki}\}$

SAP_k：プロセス k の SAP の集合

M_{jk}：プロセス j からプロセス k へ送りうるメッセージの集合 (j=1, …, n)
ただし，$M_{kk}=\phi$

M_{ki}：ユーザ k からプロセス k への送りうるメッセージの集合

O_k：プロセス k の出力アクションの集合

$O_k=\{p!m_0 \mid p\in \mathrm{SAP}_k, m_0\in(\bigcup_{j=1}^{n} M_{kj})\cup M_{k_0}\}$

M_{kj}：プロセス k からプロセス j へ送りうるメッセージの集合 (j=1, …, n)
ただし，$M_{kk}=\phi$

M_{k_0}：プロセス k からユーザ k への送りうるメッセージの集合

ω_k：プロセス k の出力関数 $Q_k\times I_k\rightarrow O_k^*$

δ_k：プロセス k の遷移関数 $Q_k\times I_k\rightarrow Q_k$

q_{k_0}：プロセス k の初期状態 □

ここで，入出力アクションを構成する“?”および“!”はそれぞれそのアクションが入力アクション，出力アクションであることを表す．

また，本手法で対象とする相互接続試験の環境を**図2**のようにモデル化する．被試験システムである各プロトコルエンティティ N-Entity-k を IUT (Implementation Under Test) k 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

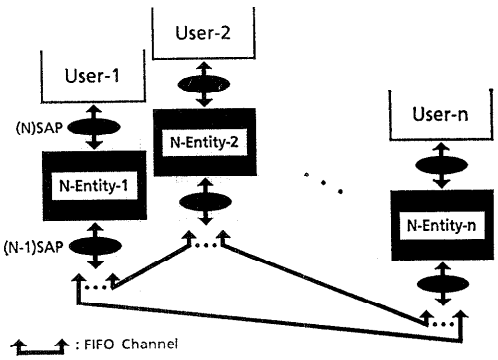


図 1 被試験システムの環境のモデル
Fig. 1 Model of environment for system under test.

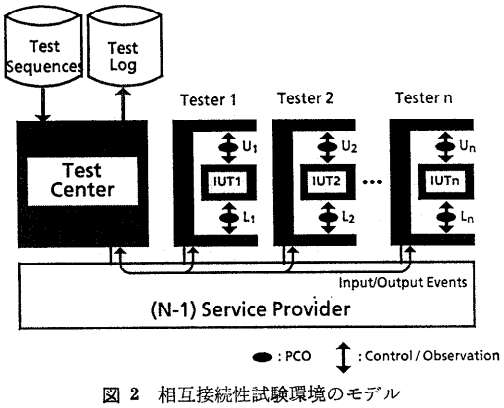


図 2 相互接続性試験環境のモデル
Fig. 2 Model of environment for interoperability testing.

する．また，各下位テスタは，$(N-1)$ 層以下のサービス提供者を通してテストセンタと通信する．以降，IUTk の上位テスタを UTk，下位テスタを LTk，また UTk と IUTk の間の PCO を U$_k$，LTk と IUTk の間の PCO を L$_k$ とそれぞれ呼ぶ．本手法で対象とする相互接続試験の試験系列とは，各 PCO，U$_k$，L$_k$ 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．なお，テストセンタと各テスタ間の通信に関しておよび同期の実現方法等に関しては，別にこれを定めるものとする．

3. 並列試験系列記述法（PTSN）

従来の TTCN[1]等の適合性試験系列の記述法は，基本的に試験系列を一つの時間系列として表すもので，並列性を意識していない．一方，相互接続試験においては，対象となるシステムは通信し合う複数の独立したプロセスである．ところが，試験系列を一つの系列として表す従来の記述方法によれば，このような試験系列に基づく試験下において，ある一つのプロセスで一つの試験イベントを制御／観測している時は，他のすべてのプロセスに対する制御／観測は行われない．このことは，試験の実施時間の増大を招き，試験の効率化に影響を及ぼすという問題があった．

そこで，本論文では，ネットワークを介して接続されている相互接続試験における多数のプロセスに着目し，複数のプロセスにおいて並列に制御／観測動作が可能な部分を陽に表すような試験系列の記述法である PTSN を提案する．以下では，まず並列動作を含む相互接続試験系列の記述に必要な定義を行い，その定義に基づいて，提案する試験系列記述法のシンタクスを与える．

【定義 2】 **試験イベント** e は，各テスタが，IUT を制御するために IUT に対して与える入力試験イベント $e?$ および IUT の動作として観測されるべき IUT から受け取る出力試験イベント $e!$ の組である． □

すなわち，$e=(e?, e!)$ で表す．ただし，出力試験イベントは入力試験イベントが決まればプロトコル仕様より一意に定まるため，これを省略することもある．以下では，特にことわりがない限り，試験イベントというのは入力試験イベントを表すことにする．ここで，テスタ i に対する j 番目の試験イベントを e_i^j と表す．また，空の試験イベントは “ε” で表す．

【定義 3】 **イベント系列** t は次のように帰納的に定義される．

（1） 試験イベント e はイベント系列である．

（2） e_1 を試験イベント，e をイベント系列とすると $e_1; e$ はイベント系列である． □

ここで，テスタ i に対するイベント系列 t_i は

$$t_i = e_i^1; e_i^2; \cdots; e_i^{ki}$$

と表される．

【定義 4】 **並列イベント系列** p は次のように帰納的に定義される．

（1） t_1，t_2 をイベント系列とすると，$\{t_1 \| t_2\}$ は並列イベント系列である．

（2） t_1 をイベント系列，$\{x\}$ を並列イベント系列とすると，$\{t_1 \| x\}$ は並列イベント系列である． □

並列イベント系列 $\{t_1 \| \cdots \| t_n\}$ はイベント系列 $t_1, \cdots, t_n$ をそれぞれに対応するテスタ 1, …, テスタ n で並列に試験することを示す．すなわち，テストセンタがすべての入力試験イベントをそれぞれのテスタにネットワークを介して並列に送ることを意味する．そして，各テスタはそれぞれ並列してこれらの入力試験イベントを各プロセスに対して与え，出力試験イベントを観測し，これをネットワークを介してテストセンタに送る．

【定義 5】 並列イベント系列またはイベント系列を**部分試験系列**と定義する． □

【定義 6】 **試験ケース**は部分試験系列の系列（直列結合）である． □

すなわち，試験ケースのうちで並列に動作可能な部分は並列イベント系列として表し，並列に動作不可能な部分は一つのイベント系列として記述する．

【定義 7】 **試験スイート**は試験ケースの集合である． □

すなわち，試験系列全体を PTSN では試験スイートと呼ぶ．

以上をまとめ，PTSN のシンタックスを BNF により表すと次のようになる．

【並列試験系列記述法 PTSN】

```
(a) test_suite : :=test_case {"+"test_case}
(b) test_case : :=test_subsequence
                    {";"test_subsequence}
(c) test_subsequence : :=parallel_event_sequence
                        |event_sequence
(d) parallel_event_sequence
    : :="{"event_sequence {"||"event_sequence}
      "}"
```

(e) event_sequence : :=event {";"event}
(f) event : :=text
|"ε" □

4. 相互接続試験系列生成手法（PP 法）

4.1 システム状態グラフの拡張

プロトコルが定義1のように n 個の 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，n 個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 n 個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，プロセス i および j に対する試験系列が $a_ib_ic_i$ および $d_je_jf_j$ であった時，b_i は a_i の後で c_i の前に起こることは分かるが，b_i と d_j，e_j，f_j との前後関係の情報が欠けているため，この二つの系列からは相互接続試験を行うことができない．このような問題をここでは相互接続試験におけるスケジューリングの問題と呼ぶ．

本論文では，このスケジューリングの問題を解決するために，システム全体の動作を表すシステム状態グラフ (SSG : System State Graph)[14]を n プロセスに拡張する．SSG は，各プロセスの状態およ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．以下に SSG の定義を与える．なお，各チャネルは FIFO であるものとする．

【定義8】 システム状態

$s=\langle(q_1,\cdots,q_n),\cdots,c\langle i,j\rangle,\cdots\rangle$

ただし，q_k：プロセス k の状態 $q_k\in Q_k$

$c\langle i,j\rangle$：プロセス i からプロセス j へのチャネルの内容　$c\langle i,j\rangle\in M_{ij}^*$　$(i\neq j)$ □

【定義9】 SSG $G=\langle S,I,O,\omega,\delta,s_0\rangle$

ここで，

S：システム状態の集合

I：入力アクションの集合

O：出力アクションの集合

ω：出力関数　$S\times I\rightarrow O^*$

δ：遷移関数　$S\times I\rightarrow S$

s_0：初期システム状態 □

次に，プロトコル P が与えられたときこの SSG を生成する規則を次のように与える．以下では，"s_j — $a_i/a_0\rightarrow s_k$" は "状態 s_j で a_i を入力すると a_0 を出力し状態 s_k に移る" 遷移を表す．

【SSG 生成規則】

プロトコル $P=\langle\cdots,\langle Q_k,I_k,O_k,\omega_k,\delta_k,q_{k0}\rangle,\cdots\rangle$ が与えられたとき，$I=\cup I_k$，$O=\cup O_k$ と**図3**の公理 A1 および推論規則 I1，I2 により推論される最小の集合 S および出力関数 ω，遷移関数 δ，初期システム

A1 (Initial System State)

$s_0=<(q_{10},\cdots,q_{n0}),\cdots,E,\cdots>\in S$

I1 (Inputs on U_k)

$s=<(q_1,\cdots,q_k,\cdots,q_n),\cdots,c<k,1>,\cdots,c<k,k-1>,c<k,k+1>,\cdots,c<k,n>,\cdots>\in S$

$q_k-U_k?m_{k_i}/U_k!m_{k_0}L_k!m_{k1}\cdots L_k!m_{kk-1}L_k!m_{kk+1}\cdots L_k!m_{kn}\rightarrow q_k'$

$s-U_k?m_{k_i}/U_k!m_{k_0}L_k!m_{k1}\cdots L_k!m_{kk-1}L_k!m_{kk+1}\cdots L_k!m_{kn}\rightarrow s'$

但し，

$s'=<(q_1,\cdots,q_k',\cdots,q_n),\cdots,c<k,1>',\cdots,c<k,k-1>',c<k,k+1>',\cdots,c<k,n>',\cdots>,$

$c<kj>'=append(c<k,j>,m_{kj})$

I2 (Inputs on L_k)

$s=<(q_1,\cdots,q_k,\cdots,q_n),\cdots,c<l,k>,\cdots,c<k,1>,\cdots,c<k,k-1>,c<k,k+1>,\cdots,c<k,n>,\cdots>\in S,$

$q_k-L_k?m_{lk}/U_k!m_{k_0}L_k!m_{k1}\cdots L_k!m_{kk-1}L_k!m_{kk+1}\cdots L_k!m_{kn}\rightarrow q_k',\quad top(c<l,k>)=m_{lk}$

$s-L_k?m_{lk}/U_k!m_{k_0}L_k!m_{k1}\cdots L_k!m_{kk-1}L_k!m_{kk+1}\cdots L_k!m_{kn}\rightarrow s'$

但し，

$s'=<(q_1,\cdots,q_k',\cdots,q_n),\cdots,c<l,k>',\cdots,c<k,1>',\cdots,c<k,k-1>',c<k,k+1>',\cdots,c<k,n>',\cdots>,$

$c<l,k>'=remain(c<l,k>),$

$c<kj>'=append(c<k,j>,m_{kj})$

図3 SSG 生成規則の公理と推論規則
Fig. 3 SSG generation rules.

状態 s_0 からなる SSG $G=\langle S, I, O, \omega, \delta, s_0\rangle$ を P に関する SSG と定める．以下の規則において，E はチャネルの内容が空であることを示す．また，$append(c_{kj}, m)$，$top(c_{kj})$，$remain(c_{kj})$ をそれぞれ次のように定める．

$append(c_{kj}, m)$：チャネル c_{kj} の最後に m を加えた状態のチャネル．

$top(c_{kj})$：チャネル c_{kj} の先頭の内容．チャネル c_{kj} が空の場合には定義されない．

$remain(c_{kj})$：チャネル c_{kj} 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_{kj} が空の場合には定義されない． □

ところで，プロトコルの仕様によっては，これらの規則の適用が有限回で終了しない場合がありうる．例えば，プロセス P_1 の中に，ある状態から U_1 で入力を受け 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，I1 が無限回適応できうる．ところが，この問題は本質的にはプロトコルの論理検証に含まれる一般には自動検証が不可能な停止性の問題であるため，ここでは議論の対象としない．そこで，以下本論文ではこれらの規則の適用が有限回で終了する場合のみを考える．したがって，以下では SSG を有限グラフとして扱う．このとき，明らかにある定数 σ が存在して

$C_S \leqq \sigma$ （C_S は SSG のチャネルの状態数）

となる．

SSG 上においては，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，スケジューリングの問題が解決される．

4.2 相互接続試験系列の導出

本論文で提案する相互接続試験系列生成法を次のように与える．本手法では，SSG 上の各遷移に対して，基本試験手順[14]の三つのステップを行うような系列を試験系列として生成する．ここでは，SSG 上の状態（すなわち各プロセスの合成状態）の確認において，SSG を単なる一つの FSM と見なしてその確認シーケンス[14]（UIO シーケンス[5]等）を適用するのではなく，相互接続試験の特質である複数のプロセスの独立性に着目し，その状態を構成する個々のプロセスの状態を確認する．そして，基本試験手順の三つのステップのうち，ステップ1およびステップ3に関しては，各プロセスで並列に動作可能な部分であるので，これを並列試験系列として記述することを特長としている．

【基本試験手順】

任意の遷移 $s_j - a_i/a_0 \rightarrow s_k$ について

ステップ1：状態 s_j に試験対象を遷移させる．

ステップ2：入力 a_i を試験対象に与え出力 a_0 を観測する．

ステップ3：試験対象が状態 s_k にあることを確認する． □

【相互接続試験系列生成法：PP 法】

フェーズ1：プロトコル $P=\langle P_1, P_2, \cdots, P_n\rangle$ より SSG 生成規則に従って SSG $G=\langle S, I, O, \omega, \delta, s_0\rangle$ を生成する．

フェーズ2：各プロセス k に関して，$q_{ki} \in Q_k$ をユニークに識別するシーケンス $unique(q_{ki})$ を求める．

フェーズ3：SSG 上の $\omega(s_p, a_{in})=a_{out}$，$\delta(s_p, a_{in})=s_q$ である任意の遷移 $s_p - a_{in}/a_{out} \rightarrow s_q$（ここで，$s_p=\langle(q_1, \cdots, q_n), \cdots, c\langle i, j\rangle, \cdots\rangle$，$s_q=\langle(q_1', \cdots, q_n'), \cdots, c\langle i, j\rangle', \cdots\rangle$ とする）について基本試験手順のステップ1～3に対応する系列を以下の①～③のように構成する．

① $r\,;\{hide_1(x(s_0, s_p))\|\cdots\|hide_n(x(s_0, s_p))\}$；

② a_{in}；

③ $\{unique(q_1')\|\cdots\|unique(q_n')\}$

ここで，r はリセット系列，$x(s_0, s_i)$ は初期システム状態 s_0 からシステム状態 s_i へ遷移させる入力の系列，$hide_k(t)$ は系列 t のうちプロセス k 以外の入力を除いたものを表す．また，";" は系列の連結を表す． □

ここで，①～③の各々が PTSN の部分試験系列であり，これらの直列結合が SSG 上の一つの遷移に対応する試験ケースになる．SSG 上のすべての遷移に対応する試験ケースが，試験スイートである．以下，上記の①の部分を試験ケースのプリシーケンス，③の部分をポストシーケンスと呼ぶ．なお，$unique(q_{ki})$ を求める方法については，文献 3)～5)，13) を参照されたい．

ところで，本手法では，フェーズ3において SSG 上の各遷移について基本試験手順のステップ1～3に対応する系列を構成する．ここで，ステップ3の部分の系列を構成する各プロセスの状態を確認するシーケンスは各プロセスごとに独立に求められる．そのため，この部分の試験を実行する際に，プロセス間のデータ送受信の整合性がとれなくなり，以降の試験を正常に続けられなくなることが考えられる．これに対し，本論文で提案する手法においては，この場合の送受信の整合性を次のように確保している．

まず，フェーズ3において，SSG 上の各遷移について基本試験手順のステップ1〜3に対応する系列を構成する際に，ステップ1に対応する部分の最初にリセット系列を加えている．これにより，次の遷移に対する系列の実行時の初めには常に整合性がとれた状態になる．すなわち，送受信の不整合性は，ステップ3の部分の実行時にのみ起こりうる一時的なものである．

また，本手法においては，対象とする相互接続試験の環境を図2のようにモデル化している．すなわち，各 IUT 同士は直接データのやり取りをせず，それぞれ上位，下位の PCO 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．つまり，試験実施時には，テスタがすべてのデータ(試験イベント)を制御／観測する．したがって，上記のように一時的に送受信の整合性がとれなくなった場合でも，テスタが不整合なデータを棄却する等の方法により，容易にシステム全体におけるデータの整合性を確保することができる．

以上のように，相互接続試験の本質的な性質である並列性を考慮して，複数のプロセスにおいて並列に制御／観測動作が可能な部分を陽に表すような試験系列を生成することができる．これにより，試験下において，ある一つのプロセスで一つの試験イベントを制御／観測している時にも，他のプロセスに対する制御／観測を同時に並行して行うことができ，相互接続試験の実施時間の短縮による試験の効率化が期待される．

4.3 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4**のような ISDN 基本サービスを実現するための三つのプロセス A，T，B からなるシステムを考える．同図でAは発信局，Bは着信局，Tは中継局を想定している．ここで，回線の接続と切断を行うような簡単なプロトコル P_{ex} を考える（**図5**）．これに対して，SSG 生成規則を適用することにより**図6**の SSG G_{ex} が得られる（フェーズ1）．また，図5の各プロセスの FSM に関する状態確認シーケンス[14]を**図7**のように求めることができる（フェーズ2）．さらに，G_{ex} において，例えば遷移

“$s_7 - L_T?REL_{AT}/L_T!RLC_{TA}, L_T!REL_{TB} \rightarrow s_8$”

（ただし，$s_7 = \langle (3, 8, 12), REL_{AT}, E, E, E, E, E \rangle$，

$s_8 = \langle (3, 9, 12), E, E, RLC_{TA}, REL_{TB}, E, E \rangle$）

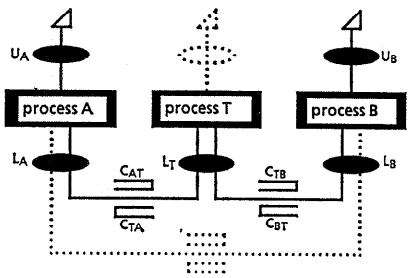


図 4 三つのプロセスからなるシステム
Fig. 4 An example system.

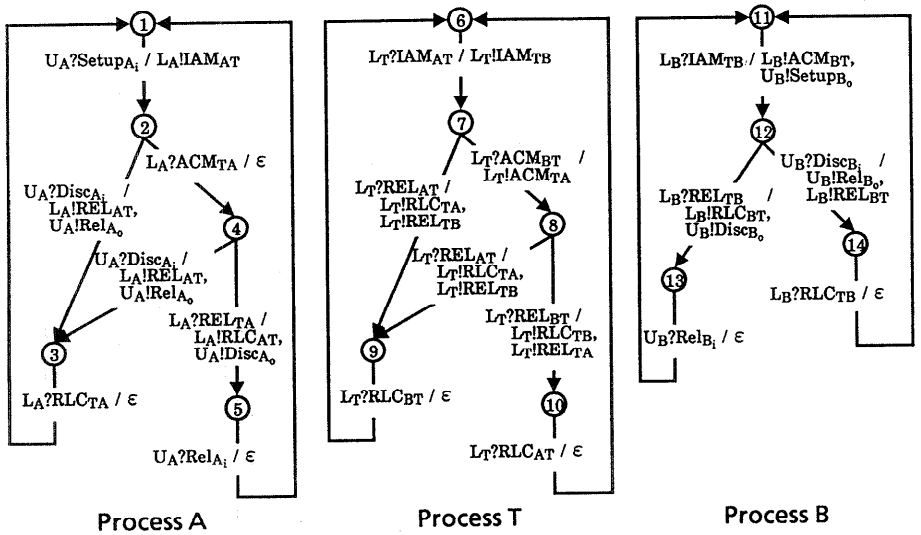


図 5 プロトコル P_{ex}
Fig. 5 An example protocol P_{ex}.

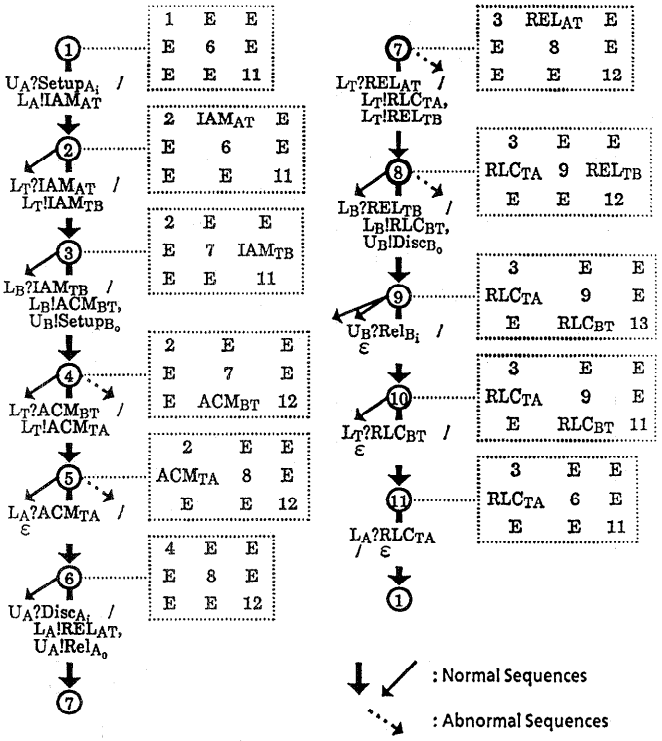


図 6 SSG G_{ex}
Fig. 6 SSG G_{ex}.

States	UIO
1	$U_A?Setup_{A_i}$ / $L_A!IAM_{AT}$
2	$L_A?ACM_{TA}$ / ε
3	$L_A?RLC_{TA}$ / ε
4	$L_A?REL_{TA}$ / $L_A!RLC_{AT}$,$U_A!Disc_{A_o}$
5	$U_A?Rel_{A_i}$ / ε

Process A

States	UIO
6	$L_T?IAM_{AT}$ / $L_T!IAM_{TB}$
7	$L_T?ACM_{BT}$ / $L_T!ACM_{TA}$
8	$L_T?REL_{BT}$ / $L_T!RLC_{TB}$,$L_T!REL_{TA}$
9	$L_T?RLC_{BT}$ / ε
10	$L_T?RLC_{AT}$ / ε

Process T

States	UIO
11	$L_B?IAM_{TB}$ / $L_B!ACM_{BT}$,$U_B!Setup_{B_o}$
12	$L_B?REL_{TB}$ / $L_B!RLC_{BT}$,$U_B!Disc_{B_o}$
13	$U_B?Rel_{B_i}$ / ε
14	$L_B?RLC_{TB}$ / ε

Process B

図 7 UIO シーケンス
Fig. 7 UIO sequences.

に関する部分試験系列は，

$x(s_1, s_7) = U_A?Setup_{A_i}$;
$L_T?IAM_{AT}$; $L_B?IAM_{TB}$;
$L_T?ACM_{BT}$; $L_A?ACM_{TA}$;
$U_A?Disc_{A_i}$ より
①r ; {$U_A?Setup_{A_i}$; $L_A?ACM_{TA}$;
$U_A?Disc_{A_i}$
‖$L_T?IAM_{AT}$; $L_T?ACM_{BT}$‖
$L_B?IAM_{TB}$} ;
②$L_T?REL_{AT}$;
③{$L_A?RLC_{TA}$‖$L_T?RLC_{BT}$‖
$L_B?REL_{TB}$}

のように求めることができる．ここで，$L_T?IAM_{AT}$ 等は一つの入力試験イベントを表し，例えば L_T はプロセスTの PCO，IAM_{AT} はプロセスAからTへのメッセージ，$Setup_{A_i}$ はユーザAからプロセスAへのメッセージである．また，例えば③の“$L_A?RLC_{TA}$”の部分はプロセスAの状態3を確認する UIO シーケンスである．このようにして，図6の G_{ex} の各遷移について**図8**のような試験ケースが求められる（フェーズ3）．

このようにして得られる試験スイートに基づき試験を実施する際には，次のように行う．すなわち，テストセンタは，試験スイートの中から一つの試験ケースを選択し，その中で，並列に動作可能な部分を表す“‖”で連結されている部分に関してはそのすべての入力試験イベントをそれぞれのプロセスに対応するテスタに送る．また並列に動作可能な部分でなければ一つの入力試験イベントをそのプロセスに対応するテスタに送る．そして各テスタはそれぞれ並列してこれらの入力試験イベントを各プロセスに対して与え，出力イベントを観測し，これを（$N-1$）サービス提供者を介してテストセンタに送る．テストセンタは試験ケースの対応する出力試験イベントと比較を行い，これらの観測された出力イベントが“正しい”ものであるかどうかを判断し，結果を試験ログに記録する（図2）．

5. 評　価

5.1 評価項目

試験系列生成手法の評価項目としては，一般に次の点が重要である[6),7)]．

Transitions			Test Sequences
1	$-U_A?Setup_{A_i}$ / $L_A!IAM_{AT}\rightarrow$	2	r; $U_A?Setup_{A_i}$; {$L_A?ACM_{TA} \| L_T?IAM_{AT} \| L_B?IAM_{TB}$}
2	$-L_T?IAM_{AT}$ / $L_T!IAM_{TB}\rightarrow$	3	r; {$U_A?Setup_{A_i}$}; $L_T?IAM_{AT}$; {$L_A?ACM_{TA} \| L_T?ACM_{BT} \| L_B?IAM_{TB}$}
3	$-L_B?IAM_{TB}$ / $L_B!ACM_{BT}, U_B!Setup_{B_0}\rightarrow$	4	r; {$U_A?Setup_{A_i} \| L_T?IAM_{AT}$}; $L_B?IAM_{TB}$; {$L_A?ACM_{TA} \| L_T?ACM_{BT} \| L_B?REL_{TB}$}
4	$-L_T?ACM_{BT}$ / $L_T!ACM_{TA}\rightarrow$	5	r; {$U_A?Setup_{A_i} \| L_T?IAM_{AT} \| L_B?IAM_{TB}$}; $L_T?ACM_{BT}$; {$L_A?ACM_{TA} \| L_T?REL_{BT} \| L_B?REL_{TB}$}
5	$-L_A?ACM_{TA}$ / $\varepsilon\rightarrow$	6	r; {$U_A?Setup_{A_i} \| L_T?IAM_{AT}$; $L_T?ACM_{BT} \| L_B?IAM_{TB}$}; $L_A?ACM_{TA}$; {$L_A?REL_{TA} \| L_T?REL_{BT} \| L_B?REL_{TB}$}
6	$-U_A?Disc_{A_i}$ / $L_A!REL_{AT}, U_A!Rel_{A_0}\rightarrow$	7	r; {$U_A?Setup_{A_i}$; $L_A?ACM_{TA} \| L_T?IAM_{AT}$; $L_T?ACM_{BT} \| L_B?IAM_{TB}$}; $U_A?Disc_{A_i}$; {$L_A?RLC_{TA} \| L_T?REL_{BT} \| L_B?REL_{TB}$}
7	$-L_T?REL_{AT}$ / $L_T!RLC_{TA}, L_T!REL_{TB}\rightarrow$	8	r; {$U_A?Setup_{A_i}$; $L_A?ACM_{TA}$; $U_A?Disc_{A_i} \| L_T?IAM_{AT}$; $L_T?ACM_{BT} \| L_B?IAM_{TB}$}; $L_T?REL_{AT}$; {$L_A?RLC_{TA} \| L_T?RLC_{BT} \| L_B?REL_{TB}$}
8	$-L_B?REL_{TB}$ / $L_B!RLC_{BT}, U_B!Disc_{B_0}\rightarrow$	9	r; {$U_A?Setup_{A_i}$; $L_A?ACM_{TA}$; $U_A?Disc_{A_i} \| L_T?IAM_{AT}$; $L_T?ACM_{BT}$; $L_T?REL_{AT} \| L_B?IAM_{TB}$}; $L_B?REL_{TB}$; {$L_A?RLC_{TA} \| L_T?RLC_{BT} \| U_B?Rel_{B_i}$}
9	$-U_B?Rel_{B_i}$ / $\varepsilon\rightarrow$	10	r; {$U_A?Setup_{A_i}$; $L_A?ACM_{TA}$; $U_A?Disc_{A_i} \| L_T?IAM_{AT}$; $L_T?ACM_{BT}$; $L_T?REL_{AT} \| L_B?IAM_{TB}$; $L_B?REL_{TB}$}; $U_B?Rel_{B_i}$; {$L_A?RLC_{TA} \| L_T?RLC_{BT} \| L_B?IAM_{TB}$}
10	$-L_T?RLC_{BT}$ / $\varepsilon\rightarrow$	11	r; {$U_A?Setup_{A_i}$; $L_A?ACM_{TA}$; $U_A?Disc_{A_i} \| L_T?IAM_{AT}$; $L_T?ACM_{BT}$; $L_T?REL_{AT} \| L_B?IAM_{TB}$; $L_B?REL_{TB}$; $U_B?Rel_{B_i}$}; $L_T?RLC_{BT}$; {$L_A?RLC_{TA} \| L_T?IAM_{AT} \| L_B?IAM_{TB}$}
11	$-L_A?RLC_{TA}$ / $\varepsilon\rightarrow$	1	r; {$U_A?Setup_{A_i}$; $L_A?ACM_{TA}$; $U_A?Disc_{A_i} \| L_T?IAM_{AT}$; $L_T?ACM_{BT}$; $L_T?REL_{AT}$; $L_T?RLC_{BT} \| L_B?IAM_{TB}$; $L_B?REL_{TB}$; $U_B?Rel_{B_i}$}; $L_A?RLC_{TA}$; {$U_A?Setup_{A_i} \| L_T?IAM_{AT} \| L_B?IAM_{TB}$}

図 8 試験系列
Fig. 8 Test sequences.

(1) 系列生成条件：PP 法では，対象とする各プロセスの状態確認シーケンスを用いている．そのため，各種適合性試験の試験系列生成手法[3)~6)]と同様に，各プロセスを定義する FSM に関して，最小，強連結等の条件が必要である．

(2) 誤り検出能力：PP 法では SSG 上において状態の識別を行うことを基本とした試験系列を生成する．そのため，SSG の出力関数，遷移関数はすべて検出可能である[7)]．

(3) 試験時間：得られた試験系列を用いた試験時間は試験の費用に直接反映されるため，非常に重要である．従来，並列性を考慮しない適合性試験の試験系列生成手法においては，試験時間は試験系列の系列長に比例するため，この評価として系列長について議論した．一方，並列性を考慮した場合，試験時間は試験系列の系列長には必ずしも比例しない．そのため，ここではトータルな試験時間について検討する．

(4) 系列生成計算量：これも試験の費用に反映される．試験系列生成のための計算量のオーダについて検討する．

以下では，本論文で提案した PP 法の評価を与えるために，(3)および(4)に関して検討する．まず 5.2 節において，試験時間を調べるための準備として試験系列の系列長を求め，さらに試験系列生成のための計算量についても検討する．次に，5.3 節で試験系列を用いた試験時間について考察し，本手法を用いることによって n プロセスの相互接続試験の効率が向上することを示す．

5.2 試験系列長および計算量

まず，試験系列の系列長および試験系列生成のための計算量について検討するための準備として，SSG の大きさを表す状態数と入力数，遷移数について考察する．与えられたプロトコル $P=\langle\cdots,\langle Q_k, I_k, O_k, \omega_k, \delta_k, q_{k0}\rangle,\cdots\rangle$ を構成するプロセス k ($k=1,\cdots,n$)の状態数および入力数をそれぞれ $|Q_k|$, $|I_k|$ と記述し，また P に SSG 生成規則を適用して得られる $G=\langle S, I, O, \omega, \delta, s_0\rangle$ の状態数および入力数，遷移数をそれぞれ M, U, Tr と表す．ここで，簡単化のために

$$|Q_1|=\cdots=|Q_n|=m,$$
$$|I_1|=\cdots=|I_n|=u$$

とする．SSG の構成の仕方より，$I=\cup I_k$, $S\subseteq Q_1\times\cdots\times Q_n$ であり，さらに P の定義より $I_j\cap I_k=\phi$ $(j\neq k)$ であるから，チャネルの状態の組合わせの数は高々 σ であるものとすると

$$U=nu \quad (1)$$
$$m\leqq M\leqq\sigma m^n \quad (2)$$

となる．ここで，$M=m$ であるのは P の各プロセスが完全に同期しながら動作する場合であり，$M=\sigma m^n$ であるのは P の各プロセスが互いに全く同期せず独立に動作する場合である．ところで，現実のプロトコル

は両者の場合の中間であり，ある程度同期しながら互いに独立して動作する．そこで，以下に与える評価結果の導出過程の理解性を向上するために，ここでは同期の程度を θ を用いて表し，対象とするプロトコルに対して次のような仮定を置く．

【プロトコルの同期性に関する仮定】

プロトコル $P=\langle\cdots,\langle Q_k, I_k, O_k, \omega_k, \delta_k, q_{k0}\rangle,\cdots\rangle$ において，プロセス k の状態 q_k と同時に存在しうる他のプロセスの状態の組合わせの数は高々 θ_n である(θ は任意の定数)． □

例えば，図4のプロトコルにおいてはプロセスBの状態14と同時に存在しうるプロセスAおよびプロセスTの状態の組合わせの数が14であり，これが三つのプロセスの中の任意の状態と同時に存在しうる他のプロセスの状態の組合わせの最大値である．したがって，仮定よりこの場合の θ は

$$\theta=14/3$$

である．この仮定のもとでは，M の上限 M_{MAX} は $\lambda=\sigma\theta$ とおくと

$$M_{MAX}=\lambda nm \qquad (3)$$

となり，またSSGの遷移数 Tr は U と M の積で与えられるので，その上限値 Tr_{MAX} は

$$Tr_{MAX}=U\cdot M_{MAX}=\lambda n^2 um \qquad (4)$$

となる．

以上の準備のもとに，PP 法における試験系列の系列長および計算量について以下に示す．

PP 法では，SSG の各遷移についてプリシーケンスの長さの上限が $M_{MAX}-1$，ポストシーケンスの長さの上限が $n(m-1)$ であることから，系列長の上限 Ts_{MAX} は

$$\begin{aligned} Ts_{MAX}&=Tr_{MAX}\{(M_{MAX}-1)+1+n(m-1)\}\\ &=\lambda n^3 um\{(\lambda+1)m-1\} \end{aligned} \qquad (5)$$

である．

また，試験系列の生成に要する計算量としては，SSG を生成するのに要する計算量と，SSG から試験系列を生成するために要する計算量について考えればよい．SSG を生成する際は SSG の遷移数 Tr_{MAX} の回数だけ SSG 生成規則の適用を行うので，n^2um のオーダの計算を要する．また，SSG から試験系列を生成するためにやはり SSG の遷移数のオーダ n^2um の計算が必要である．結局全体では n^2um のオーダの計算量となる．

5.3 試験時間

ここでは，図2のようにモデル化された相互接続試験環境のもとで，PP 法により得られた試験系列に基づいた試験を行う場合の試験時間について考察する．

いま，テストセンタと各テスタ間の1回の試験データのやりとりに必要な時間を τ，また各テスタが一つの試験イベントを IUT に与えてから次の試験イベントを IUT に与えるまでに必要な時間を μ とする．ここで，試験データとは，各テスタが制御／観測すべき試験イベントまたはその系列で，一度にテストセンタと各テスタとの間でやりとりされる単位を表す．

従来の並列性を考慮しない試験系列生成手法による試験系列を用いた試験では，一つのイベントごとに，テストセンタと各テスタ間の試験イベントの往復のやりとりと，テスタと IUT との試験イベントのやりとりが1回必要である．そこで，この場合の試験時間の上限値 T_{MAX} を求めると

$$\begin{aligned} T_{MAX}&=2Ts_{MAX}\cdot\tau+Ts_{MAX}\cdot\mu\\ &=\lambda n^3 um\{(\lambda+1)m-1\}(2\tau+\mu) \end{aligned} \qquad (6)$$

となる．ここで，μ は τ に比べて無視できるものとすると，次のように近似できる．

$$T_{MAX}\simeq 2\lambda n^3 um\tau\{(\lambda+1)m-1\} \qquad (7)$$

一方，並列性を考慮した PP 法では，SSG の各遷移について，基本試験手順の三つのステップに対応した部分試験系列①～③からなる試験ケースを構成する．この試験ケースに基づいた試験では，各部分試験系列ごとに，テストセンタと各テスタ間の試験データのやりとりが往復で2回，およびテスタと IUT との試験イベントのやりとりが試験データを構成するイベント数の回数だけ必要である．この場合の各ステップに関する試験時間を求めると，

① $2\tau+(M_{MAX}-1)\mu$

② $2\tau+\mu$

③ $2\tau+(m-1)\mu$

となる．これより，全体の試験時間の上限値 T_{MAX}' はこれらの和に SSG の遷移数 Tr_{MAX} を乗じて，

$$\begin{aligned} T_{MAX}'&=\{6\tau+(M_{MAX}+m-1)\mu\}\,Tr_{MAX}\\ &=\lambda n^2 um\{6\tau+(\lambda nm+m-1)\mu\} \end{aligned} \qquad (8)$$

ここでも，μ は τ に比べて無視できるものとすると，これらはどちらも次のように近似できる．

$$T_{MAX}'\simeq 6\lambda n^2 um\tau \qquad (9)$$

ここで，式(7)と式(9)の比を取ると

$$T_{MAX}'/T_{MAX}=3/n\{(\lambda+1)m-1\} \qquad (10)$$

となる．すなわち，PP 法による試験スイートに基づ

いた試験の試験時間は，PTSN を用いて並列に試験を実行することによって nm のオーダの改善がみられることがわかる．このことは，プロセス数 n の多いより大規模なシステムのプロトコルや，各プロセスの状態数 m が大きいより複雑なプロトコルの場合において，効率的な試験を行うために PP 法がより有効であることを示している．

なお，式(10)は式(7)と式(9)の比であるため，5.2節におけるプロトコルの同期性に関する仮定は結果的に式(10)に影響を与えない．すなわち，この仮定が成り立たない場合においても，上記と同様の結果が得られる．

6. むすび

本論文では，n プロセスの相互接続試験における試験系列の生成の手法である PP 法を提案した．ここでは，複数のプロセスに対する試験時間を短縮するために，並列に制御／観測動作が可能であることを陽に表す方法として並列試験系列記述法 (PTSN) を導入した．そして，システム全体の動作を表すシステム状態グラフを n プロセスに拡張し，システム状態グラフ上において状態の識別を行うことを基本としたエラー検出能力の高い試験系列を生成した．特に相互接続試験における各プロセスの並列性を考慮し，PTSN に基づいた試験系列を生成することにより，相互接続試験における並列動作を含む試験の実施が容易になり，試験実施時間の短縮による効率化が可能であることを示した．

今後の課題としては，本手法に基づいた相互接続試験系列生成システムおよび本手法により得られた試験系列を用いた相互接続試験の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くだ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 (1991).
2) Naito, S. and Tsunoyama, M.: Fault Detection for Sequential Machines by Transition Tours, *Proc. IEEE Fault Tolerant Comput. Conf.*, pp. 238-243 (1981).
3) Chow, T.: Testing Software Design Modeled by Finite-state Machines, *IEEE Trans. Software Eng.*, Vol. SE-4, pp. 178-187 (1978).
4) Gonenc, G.: A Method for the Design of Fault Detection Experiment, *IEEE Trans. Cmput.*, Vol. C-19, pp. 551-558 (1970).
5) Sabnami, K. and Dahbura, A.: A Protocol Test Generation Procedure, *Computer Networks ISDN System*, Vol. 15, pp. 285-297 (1988).
6) 佐藤文明，宗森 純，井手口哲夫，水野忠則：有限オートマトンに基づくシステムの試験系列自動生成手法の提案，信学論，Vol. J 72-B-I，No. 3, pp. 183-192 (1991).
7) Sidhu, D. P. and Leung, T.: Formal Method for Protocol Testing: A Detailed Study, *IEEE Trans. Software Eng.*, Vol. SE-15, No. 4, pp. 413-426 (1989).
8) West, C. H.: General Technique for Communications Protocol Validation, *IBM J. Res. & Devel.*, Vol. 22, No. 4, pp. 394-404 (1978).
9) Hogrefe, D.: Conformance Testing Based on Formal Methods, *Proc. FORTE '90*, pp. 213-245 (1990).
10) Rafiq, O. and Castanet, R.: From Conformance Testing to Interoperability Testing, *Proc. the 3rd International Workshop on Protocol Test Systems*, pp. 371-385 (1990).
11) Arakawa, N. and Soneoka, T.: A Test Case Generation Method for Concurrent Programs, *Proc. the 4th International Workshop on Protocol Test Systems*, pp. 95-106 (1991).
12) 岡崎直宣，高橋 薫，白鳥則郎，野口正一：LOTOS 仕様からの効率的な試験系列の生成法，信学論，Vol. J 74-B-I，No. 10，pp. 733-747 (1991).
13) 当麻喜弘，内藤祥雄，南谷 崇：順序機械，岩波書店 (1983).
14) 朴 美娘，岡崎直宣，太田正孝，高橋 薫，白鳥則郎，野口正一：プロセスの独立性を考慮した通信システムの相互接続試験系列生成法，信学論，Vol. J 76-B-I，No. 3, pp. 264-273 (1993).
15) 朴 美娘，岡崎直宣，高橋 薫，白鳥則郎，野口正一：並列動作を含む相互接続試験系列記述法の提案，第45回情報処理学会全国大会論文集，Vol. 1, pp. 155-156 (1992).
16) 岡崎直宣，朴 美娘，太田正孝，高橋 薫，白鳥則郎，野口正一：並列性を考慮した通信システムにおける相互接続試験系列生成法，第45回情報処理学会全国大会論文集，Vol. 1, pp. 157-158 (1992).
17) 朴 美娘，岡崎直宣，高橋 薫，白鳥則郎，野口正一：並列性を考慮した相互接続試験系列生成法，信学技報，IN 92-40, pp. 75-80 (1992).

(平成4年10月30日受付)
(平成5年 4月 8日採録)

朴　美娘（正会員）

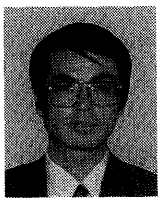
1959 年生．1983 年韓国漢陽大学工学部電子工学科卒業．1985 年同大学院修士課程修了．1987 年東北大学大学院工学研究科情報工学科専攻博士課程入学．並列処理，通信システムにおけるプロトコルの検証・試験に関する研究に従事．現在，東北大学電気通信研究所勤務．工学博士．

岡崎　直宣（正会員）

1962 年生．1986 年東北大学工学部通信工学科卒業．1991 年同大学院博士課程修了．工学博士．現在，(株)高度通信システム研究所研究員．通信システムにおけるプロトコルの仕様記述・検証・試験に関する研究に従事．電子情報通信学会会員．

三上　節子（正会員）

1965 年生．1988 年山形大学工学部情報工学科卒業．現在，富士通東北通信システム(株)より(株)高度通信システム研究所に出向中．通信ソフトウェア開発環境に関する研究に従事．



高橋　薫（正会員）

1954 年生．東北大学電気通信研究所・助手．工学博士．分散システム・通信プロトコルの設計法，仕様記述技法，検証・論理検証法，合成法，試験法，通信ソフトウェアの自動インプリメンテーション法などの研究に従事．電子情報通信学会，IEEE Computer Society 各会員．現在，(株)高度通信システム研究所・客員研究員．

白鳥　則郎（正会員）

昭和 21 年生．昭和 52 年東北大学大学院博士課程修了．同年，東北大学電気通信研究所勤務．昭和 59 年，同大学助教授（電気通信研究所）．平成 2 年，同大教授（工学部情報工学科）．情報通信システムの構成論，ソフトウェア開発法，ヒューマンインタフェースの研究に従事．情報処理学会 25 周年記念論文賞受賞．IEEE，電子情報通信学会，人工知能学会各会員．

野口　正一（正会員）

昭和 5 年生．昭和 29 年東北大学工学部電気工学科卒業．昭和 35 年同大学院博士課程修了．工学博士．昭和 46 年東北大学電気通信研究所教授．昭和 59 年東北大学大型計算機センター長．平成 2 年東北大学応用情報学研究センター長．主として情報システム構成論，知識処理に関する研究に従事．著書「情報ネットワーク理論」（岩波書店），「知識工学基礎論」（オーム社）など．現在，日本大学工学部教授．