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

3V−7

岡崎直宣† 朴美娘†† 太田正孝† 高橋薫†† 白鳥則郎†† 野口正一††

†AIC ††東北大学

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1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(1)”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”との組合せで実施されている．

本稿では，相互接続試験に関して，nプロセスの場合を対象とした試験系列の生成法を提案する．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導入し，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．ここでは，特に相互接続試験の特徴であるプロセスの並列性を考慮し，試験実施時に並列動作可能な部分を陽に表すような試験系列の生成を行う．このことにより，nプロセスの相互接続試験の効率化が可能になると考えられる(4)．

2. 試験システムのモデル化

本手法では，次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$P=<P_1,P_2,\cdots,P_n>$ 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

O_k：プロセスkの出力アクションの集合

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_{k0}：プロセスkの初期状態　□

本手法で対象とする相互接続試験の環境を図1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互いに通信する．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Uk，LTkとIUTkの間のPCOをLkとそれぞれ呼ぶ．

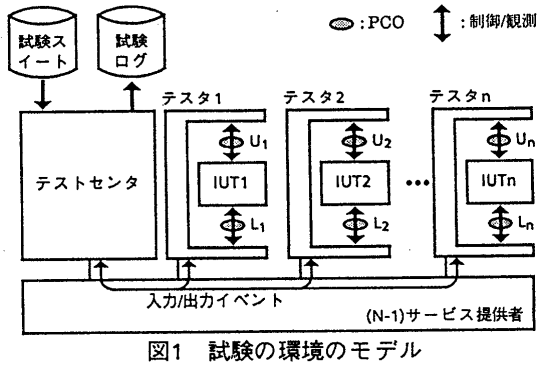

図1　試験の環境のモデル

本手法で対象とする相互接続試験の試験系列とは，各PCO, Uk, Lk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，各下位テスタ間の通信に関して，および各テスタ間の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. 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

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．

本稿では，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 States Graph)を導入する．SSG

Test sequence generation method for interoperability testing considering parallel processes.

Naonobu OKAZAKI†, Mi Rang PARK††, Masataka OHTA†, Kaoru TAKAHASHI††,
Norio SHIRATORI††, Shoichi NOGUCHI††
†Advanced Intelligent Comm. Sys. Lab.
††TOHOKU University.

は，各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．以下にSSGの定義を与える．尚，各チャネルはFIFOであるものとする．

【 定義2 】システム状態

$s=<(q_1,\cdots,q_n),\cdots,c<i,j>,\cdots>$ 但し，

q_k：プロセスkの状態 $q_k \in Q_k$

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

【 定義3 】SSG $G=<S, I, O, \omega, \delta, s_0>$ 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を生成するルール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$P=<\cdots, <Q_k, I_k, O_k, \omega_k, \delta_k, q_{k0}>, \cdots>$が与えられたとき，$I=\cup I_k$，$O=\cup O_k$と図2の公理A1及び推論規則I1，I2により推論される最小の集合S及び出力関数ω，遷移関数δ，初期システム状態s_0からなるSSG $G=<S,I,O,\omega,\delta,s_0>$を$P$に関するSSGと定める．(Eはチャネルの内容が空であることを示す) □

A1 (初期システム状態)

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

I1 (U_kでの入力)

$$q_k - U_k?m_k / U_k|m_{k_2} L_k|m_{k1} \cdots L_k|m_{kk-1} L_k|m_{kk+1} \cdots L_k|m_{kn} \rightarrow q_k',$$
$$s=<(q_1,\cdots,q_k\cdots,q_n),\cdots,c<k,1>,\cdots,c<k,k-1>,c<k,k+1>,\cdots,c<k,n>,\cdots> \in S$$

$$s - U_k?m_k / U_k|m_{k_2} L_k|m_{k1} \cdots L_k|m_{kk-1} L_k|m_{kk+1} \cdots L_k|m_{kn} \rightarrow s'$$

但し，

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

$c<k,j>'=append(c<k,j>,m_k)$

I2 (L_kでの入力)

$$s=<(q_1,\cdots,q_k\cdots,q_n),\cdots,c<l,k>,\cdots,c<k,1>,\cdots,c<k,k-1>,c<k,k+1>,\cdots,c<k,n>,\cdots> \in S,$$
$$q_k - L_k?m_{lk} / U_k|m_{k_2} L_k|m_{k1} \cdots L_k|m_{kk-1} L_k|m_{kk+1} \cdots L_k|m_{kn} \rightarrow q_k', \quad top(c<l,k>)=m_{lk}$$

$$s - L_k?m_{lk} / U_k|m_{k_2} L_k|m_{k1} \cdots L_k|m_{kk-1} L_k|m_{kk+1} \cdots L_k|m_{kn} \rightarrow s'$$

但し，

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

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

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

図2　SSG生成規則の公理と推論規則

SSG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おり，これによってスケジューリングの問題が解決される．

3.2　相互接続試験系列の導出

本稿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[3]の3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1つのFSMと見なしてその確認シーケンス[3](UIOシーケンス[2]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並列性に着目し，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3つのステップのうち，ステップ1及びステップ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，これを試験系列として陽に表すことを特長としている．

【 相互接続試験系列生成法 】

フェーズ1：プロトコル$P=<P_1,P_2,\cdots,P_n>$よりSSG生成規則に従ってSSG $G=<S,I,O,\omega,\delta,s_0>$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$u(q_{ki})$を求める．

フェーズ3：SSG上の各遷移$s_p - a_{in}/a_{out} \rightarrow s_q$(ここで，$s_p=<(q_1,\cdots,q_n),\cdots,c<i,j>,\cdots>$，$s_q=<(q_1',\cdots,q_n'),\cdots,c<i,j>',\cdots>$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$r;\{hide_1(x(s_0,s_p))\|\cdots\|hide_n(x(s_0,s_p))\}$;

②a_{in};

③$\{u(q_1')\|\cdots\|u(q_n')\}$

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． □

以上のように，相互接続試験の本質的な性質である各プロセスの並列性を考慮して，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1つのプロセスで1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]．

4.　まとめ

本稿では，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を提案した．ここでは，システム全体の動作を表すシステム状態グラフを導入し，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，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した．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実施時間の短縮による効率化が期待される．

今後の課題としては，効率に関する検討の他，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) K. Sabnami and A. Dahbura：“A protocol test generation procedure”, Computer Networks ISDN System, 15, pp. 285-297 (1988).

(3) 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における相互接続試験系列生成法”，信学技報，IN92-20, pp. 7-12 (1992).

(4) 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”，情処第45回全国大会発表予定．