オートマトンモデルに基づく OSIトランスポート·プロトコルの製品検証

加藤聰彦　　鈴木健二
国際電信電話株式会社研究所

1. はじめに

CCITT、ISOでは異機種計算機間通信を実現するために、開放型システム間相互接続(OSI : Open Systems Interconnection)の標準化をすすめており、現在OSI参照モデルをはじめ、トランスポート·レーヤ(TL)、セション·レーヤ(SL)のサービス定義とプロトコル仕様等各種の勧告·標準が作成されている。このようなOSIの進展に伴い、個々のOSI製品が標準プロトコルを正しくインプリメントしているかを検査する製品検証(Conformance Testing)が重要となっている。筆者らはオートマトンモデルに基づくプロトコルの製品検証方式について検討し、OSIトランスポート·プロトコル(TP)[1]に適用している[2,3]。本稿ではTPのクラス0、1、2の製品検証に関して、テスト系列の構成、製品検証実験について報告する。

2. 製品検証の実行形態

多くの異なった実現形態の製品を対象とするプロトコルの製品検証では、TPプログラムの詳細を調べて試験を行うことは困難であるために、TPプログラムの入出力応答を観測するという形で行われる。ここでは製品検証の方式として、図1のような、検証システム(プロトコル·テスタ)と被検証システムを接続し、プロトコル·テスタがテスト系列を発生し被検証システムに与え、それに対する応答を観測することにより検証を実行する方式を用いる。

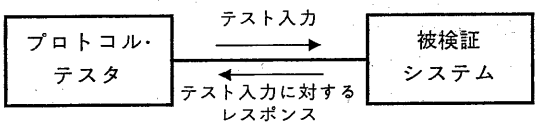

図1プロトコル製品検証の形態

OSI製品ではOSI参照モデルのレーヤ構造に従ったプログラム構造を実現しており、TPを実現する機能モジュールは、トランスポート·サービス(T)プリミティブ入力及びネットワーク·サービス(N)プリミティブ入力、相手機能モジュールからのTPDU(Transport Protocol Data Unit)入力並びにシステム内部のタイマ入力の三種類の入力を有する。その場合、被検証TPプログラムに対して、外部のテスタからのTPDU、Nプリミティブは容易に与えることができるが、Tプリミティブを外部のテスタからテスト入力として与えるには工夫を要する。

そこで外部テスタからのTプリミティブ入力の与え方によって、図2に示すような2種類の検証方式について検討する。

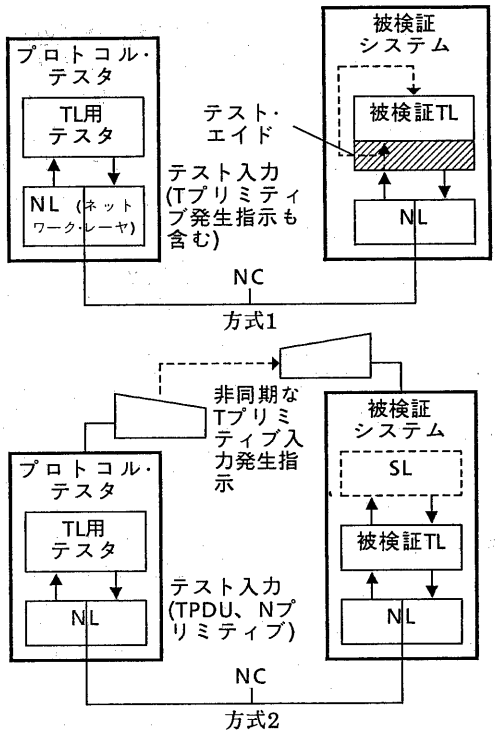

図2プロトコル製品検証の二方式

方式1: 検証のためのプログラム(検証用プログラムまたはテスト·エイドと呼ぶ)を被検証システム内に用意し、TPDU、Nプリミティブ及びタイマのタイムアウトに加えて、Tプリミティブも他の入力と同期的に用いて検証する方式。テスト·エイドは、被検証システム内において、外部のプロトコル·テスタから与えられる指示によりTプリミティブ入力を他の入力と同期的に発生する機能を有する必要がある。

方式2: プロトコル·テスタからネットワーク·コネクション(NC)を通して与えることのできるTPDUとNプリミティブ、及びタイムアウ

トのみを同期的に用い、Tプリミティブ入力は、他の入力と非同期に加えることのできるものを、例えば被検証システムのコンソールからマニュアルで入力する等の形で与えることにより検証を行う方式。

方式1は検証能力は高いが、追加のプログラミングの必要がある。一方、方式2は実現性は高いが、被検証プログラムの全ての動作を検査できないという弱点がある。

3.オートマトンモデルに基づくTPの製品検証の方法

TPはSLの要求するサービス品質にみあうデータ転送を提供する働きをする。TPでは、想定するNCの品質とトランスポート・コネクション(TC)をNCに多重化する機能に着目して0から4までのプロトコル・クラスが設けられている。0から2までのクラスが提供するプロトコル機構を表1に示す。

表1 TPのプロトコル機構

プロトコル機構	種別	0	1	2
NCへの割付け		*	*	*
TPDU転送		*	*	*
セグメンティング/アセンブリング		*	*	*
コンカティネーション/セパレーション			*	*
コネクション確立		*	*	*
コネクション拒否		*	*	*
普通解放	暗示 明示	*	*	*
エラー解放		*		*
TPDUのTCへの関連付け		*	*	*
DT TPDU番号付け	普通 拡張		*	m o
優先データ転送	NLの普通データ NLの優先データ		m ao	*
障害後の再割付け			*	
TPDUの保留	NLの送達確認 確認応答TPDU		ao m	
再同期			*	
マルチプレクシング/デマルチプレクシング				*
フロー制御 有り 無し		*	*	m o
リファレンス凍結			*	
プロトコル・エラーの処理		*	*	*

*:その手順を含む、 m:ネゴシエート可(実装は必須)、 o:ネゴシエート可(実装は任意)、 ao:ネゴシエート可(実装は任意、NLに依存)

オートマトンモデルに基づく製品検証を行うにあたっては、

(1) TPのオートマトンモデルの作成

(2) オートマトンモデルによるテスト系列の構成

(3) テスト系列を用いた製品検証の実行

の3つの段階が考えられる。以下にこれらの概要を示す。

3.1製品検証のためのオートマトンモデルの作成

製品検証においては、被検証システム内のTPプログラムの動作を表現するオートマトンモデルを規定する必要がある。このモデルは被検証システムにおけるTPのインプリメントの形態には依らない、TP本来の動作のみを規定するものでなければならない。但し、TPにはプロトコル・エラーの扱い等でインプリメントに選択を任されている部分がある。このような製品に依存した部分をオートマトンモデルで表現するために、

①全ての可能性を表現する非決定的オートマトンモデルを用いる、

②検証対象に合わせてモデルをチューニングする

という2つの方法が考えられる。TPを表現するのに①の方法を用いると、状態数の異なるオートマトンを同時に扱う必要があるため、テスト系列の作成の方法等が複雑になる。そこで、ここでは②の方法を用いることにする。

またTPの製品検証では、1本のTCに着目してTPプログラムの検証を行うのが現実的であると考えられる。1つのTCに対してはTPプログラムはイニシエータ又はレスポンダとして動作するために、TPのオートマトンモデルはイニシエータとレスポンダを別個に記述することにする。

TPのCCITT勧告X.224に付された遷移表に基づいて、オートマトンモデルを作成する。しかし、

(i)この遷移表は

・データ転送フェーズの動作全てをを記述してはいない

・プロトコル・エラーの処理やタイマ等のインプリメントに依存した動作を規定していない

・NLから障害が通知された後の再割付け/再同期等に、全ての状況が尽くされていない場合がある

等、TPの全ての動作を記述していないため、データ転送フェーズの動作を表すためあらたな状態を設定し、またX.224の遷移表の状態にそれぞれの入力を加えたときの動作を考察し、異なる動作をとりうる場合は別々の状態を設定する、

(ii)テスト系列の作成を容易にするために、条件(Predicate)により表現されている場合は、状態で規定する

等を行うことにより、TPの独立な状態を系統的に抽出し、より厳密なオートマトンモデルを作成する必要がある。

3.2テスト系列の構成方法

製品検証に用いる検証用のテスト系列を、有限オートマトンの同定の方法に基づいて構成するが、その構成方法の概要は次のようになる[4]。

(1) 検証対象となるオートマトンの状態の数が正しいオートマトンの状態数と等しいという前提条件のもとで、判定系列を用いて各状態を判定す

る。判定系列とは各状態固有の出力列を送出させる入力列のことである。

(2) テスト系列では、まず各状態に判定系列を加えて、全ての状態が存在することを確認する。つぎに各状態にそれぞれの入力を加えた場合の遷移と出力を確認する。これはオートマトンの状態を検証対象となる状態に設定し、入力を加え出力を観測し、つぎに判定系列を加えて遷移先の状態を調べ、再び対象となる状態に戻すという手順で行われる。但し、オートマトンの状態を検証対象となる状態に設定するためには、一旦別の状態に遷移させ判定系列でこれを確認し、その後既に確認済みの系列を用いて検証対象となる状態に遷移させる。

判定系列とは、同一の入力列で各状態を区別できるものであるが、検証対象となるプロトコルによってはこのような系列が存在しない場合がある。そのときは、状態固有の出力列を送出させる入力列を各状態毎に用意する(状態毎の判定系列)。後述するように、TPのクラス0、2はこのような判定系列を用いて検証できる。この場合のテスト系列の構成法を付録1に示す。

また、TPのクラス1のように、状態毎に単一の判定系列も存在しないものもあり、その場合は1つの状態を区別するのに複数の入力列を用いる必要がある(状態毎の判定系列が複数の系列からなる)。この場合のテスト系列の構成法を付録2に示すが、非常に複雑になり、テスト系列長も長くなる。

以上のような構成法を用いて、方式1、2のテスト系列を作成する。方式1ではTPプログラムの全ての入力を検証に用いることができるために、TPプログラムを通常の有限オートマトンとみなし、この方法で検証用テスト系列を構成できる。

一方、方式2では

・判定系列はプロトコル・テスタからのTPDU、Nプリミティブまたはコンソールから非同期に加えるTプリミティブによって構成する必要がある。

・有効に検証できる範囲は、発生させることのできる入力によって互いに遷移できる状態が限られることがあるため、全ての範囲を検証できない場合がある。

・非同期に加えるTプリミティブは、プロトコル・テスタが次に入力を加えるタイミングを知るために、TPDUまたはNプリミティブを出力するものを用いる必要がある。

という条件の下で、検証用テスト系列を構成する必要がある。

3.3製品検証実験の形態

オートマトン・モデルによるTPの製品検証の実験を、筆者等が作成したTPプログラム[2]に対して行った。検証システムのTPプログラムはVAX11/780のOSであるVMSの下で複数のTCを一元的に管理するTマネジャプロセスと各クラスの手順を実行するクラスプロセスとして実現されている。またVMSの下でX.25の手順を実行するPSIと接するT/Nインタフェース、コンソールと接するS/Tインタフェースが実装され、Tマネジャがこれらとメールボックスを介してインタフェースする。現在は、TPプログラムの上にセション・プログラムが実装されているが、これを使用しない状態で製品検証の実験を行っている。

実験ではプロトコル・テスタもVAX11/780上に実現し、折り返し試験の形態をとった。方式1の製品検証実験の形態を図3に示す。TPプログラム、TPのプロトコル・テスタの処理を実行するテスタ・プロセス、テスト・エイドは全てC言語で実装されている。

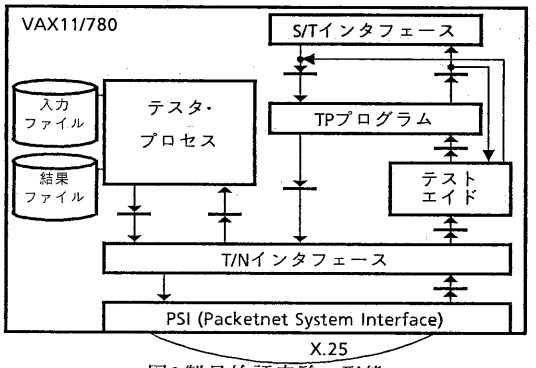

図3製品検証実験の形態

テスタ・プロセスは、T/Nインタフェース上に実現され、次の動作を繰り返す。

①入力ファイルからテスト入力と期待する応答を読む。

②入力がTプリミティブである場合はその発生指示をDT TPDUのユーザ・データにセットして送信する。

③入力がTPDUまたはNプリミティブの場合はNCを通じてその入力を与える。

④入力がタイムアウトの場合はタイムアウトの時間だけ待つ。

⑤その後、被検証プログラムでの処理と応答の伝送遅延をみこして、ある一定時間応答を待ち、応答を解析し、結果ファイルに出力する。

またテスト・エイドは、被検証プログラムの変更が少なくなるようにT/NインタフェースとTPプログラムの間にプロセスの形で実現した。その場合の変更はTPプログラム(Tマネジャ)のメールボックス名を変えるだけとなる。テスト・エイドの動作は以下のようになる。

①受信するTPDUを監視し、Tプリミティブの発生を指示したDT TPDUならば指定の上位プリミティブを指定のタイミングでS/TインタフェースからTPへのメールボックスにいれ、DT TPDUを破棄する。それ以外はそのままTPへ渡す。

②TPプログラムから上位に通知されるTプリミティブをメールボックスを介して読み込み、レスポンダに対する検証を行う場合はTCONindにより通知されるTCの識別子(Transport Service Connection Identifier[2])を得、またそれ以外のTプリミティブを破棄する。

方式2では、テスト・エイドが不要の他は方式1と同様な形態で製品検証実験が行われる。検証においてTプリミティブが必要な場合は、テスタ・プロセスはその発生をコンソールにより指示し、そのTプリミティブは他の入力とは非同期にS/Tインタフェースから加えられる。

次に筆者等が作成したTPプログラムのクラス0、1、2の製品検証について述べる。

4.TPクラス0の製品検証

4.1 クラス0のオートマトンモデル

クラス0は、最も単純なクラスでCCITT勧告T.70と同等である。クラス0においてプロトコル・エラーに対してER TPDUを使用する場合は(これが標準的な動作と考えられる)、ER TPDUを送信しタイマを起動した状態が想定される。この状態はインプリメントに依存するためX.224の遷移表には規定されていない。しかし、実際の製品の検証のためのオートマトンモデルではこの状態を区別する必要がある。また、レスポンダでは、TCが解放された状態として、TCに割付けられるNCが切断されている状態と、NCは確立されて次にCR TPDUを待っている状態とが想定され、これらを区別して扱うことにする。

クラス0イニシエータのオートマトンモデルは文献[2]で述べているため、ここではクラス0レスポンダについて述べる。X.224ではクラス0レスポンダの状態としてCLOSED(TCが解放されている状態)、WFTRESP(上位層にTCONindを発行した後TCONrespを待っている状態)、OPEN(データ転送が可能な状態)が設定されている。製品検証のためのオートマトンモデルでは、状態CLOSEDを、NCが割付けられていない状態と割付けられている状態に分け、さらにプロトコル・エラーの状態ERRORを設ける必要がある。表2にクラス0レスポンダのオートマトンモデルの一部を示す。

表2 クラス0レスポンダのオートマトンモデル

STATE / EVENT	CLOSED (NC closed) N1	CLOSED (NC open) N2	WFTRESP A	OPEN B	ERROR C
TCON resp	N1	N2	B CC	B	C
TDT req	N1	N2	A	B DT	C
NCON ind	N2 NCONrsp	//*	//	//	//
CR	//	A	C ER	C ER	C
DT	//	N2	C ER	B	C
ER	//	N2	A	N1 NDISreq	N1 NDISreq

注 *: //は有り得ない遷移を示す。

4.2クラス0レスポンダの製品検証実験

TPのクラス0レスポンダを例にとり、その製品検証実験について述べる。

クラス0レスポンダの方式1による検証のために、表2に示すオートマトンモデルの各状態に対して

N1: NCONind (出力:NCONconf)
N2: TDTreq, CR, TCONresp (出力:無し, 無し, CC)
A: TDTreq, CR, TCONresp (出力:無し, ER, 無し)
B: TDTreq, CR, TCONresp (出力:DT, ER, 無し)
C: TDTreq, CR, TCONresp (出力:無し, 無し, 無し)

という判定系列を用いて、各々の状態を他の状態から区別することができる。

クラス0レスポンダのオートマトンモデルでは各状態毎に用意された判定系列を用いて各状態を識別できるため、付録1に示した方法によってテスト系列を構成する。クラス0レスポンダの製品検証のためのテスト系列とそれを用いた実験結果の一部を図4に示す。

この実験で用いたテスト系列は734入力から成り、遷移表のうち検査対象となる70遷移の全てを検証することができる。またテスト・エイドのプログラム規模はC言語のソース・プログラムで、コメント行と空白行を除いて(実効行数と呼ぶ)277行である。

また方式2による製品検証では、TPDUとNプリミティブから次のような判定系列を用いて各状態を識別できる。

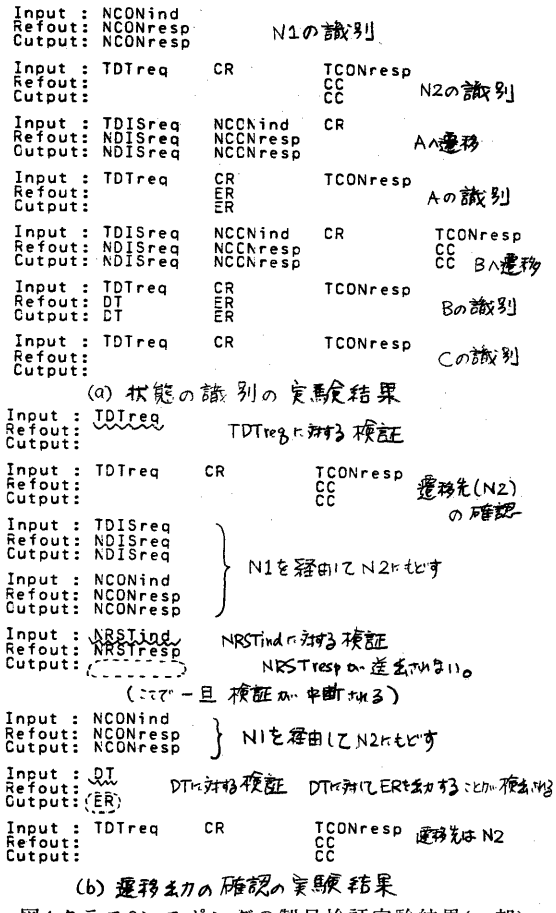

図4クラス0レスポンダの製品検証実験結果(一部)

N1: NCONind	(出力:NCONconf)
N2: DT,CR,DT	(出力:無し,無し,ER)
A: DT,CR,DT	(出力:ER,無し,無し)
B: DT,CR,DT	(出力:無し,ER,無し)
C: DT,CR,DT	(出力:無し,無し,無し)

本方式では、TCONindに対してコンソールからマニュアルでTCONrespを加える必要がある。方式2による検証では、遷移表のうち61の遷移を検証でき、かなりの範囲が検証可能である。

検証実験の結果、筆者等が作成したTPプログラムにおいて、状態N2(TCは解放されていて対応するNCが接続された状態)で、NRSTind及びDTの処理に、また状態C(ERROR)でTDISreq、CC、ER及びNRSTindの処理に誤りが発見された。X.224の遷移表ではN2の状態とCの状態が明確に記述されていないために、これらの状態での処理に誤りが多く生じたと考えられる。

5.TPクラス2の製品検証

5.1. クラス2のオートマトンモデル[3]

TPのクラス2では、フロー制御及び、TCをNCに多重化する機能が特徴であり、フロー制御についてはその使用をネゴシエートできる。オートマトンモデルではこれらを表現する必要がある。

(1) 筆者等が作成したクラス2プログラムでは、TCの解放においては、そのTCが割付けられているNCに他のTCが割付けられていないならば、イニシエータがNCを解放するようになっている。従ってイニシエータのオートマトンモデルではNCに他のTCが多重化されているか否かを区別するために別個の状態を設定する必要がある。

(2) レスポンダでは、NCの解放は行わないため、TCが解放された状態のみでNCが張られているかどうかを区別するだけでよい。

(3) データ転送フェーズでは、ED TPDUを連続して送信できないため、ED TPDUを送信したか否かを区別する必要がある。またフロー制御を表現するために、DT TPDU転送のためのウィンドウが開いているかどうかを区別する状態を設定する。

TPクラス2のイニシエータに対して、状態数が19のオートマトンモデルを規定できる。その一部を表3に示す。このオートマトンモデルでは、入力として同一NCに割当てられた他のTCに対するTCONreq、TDISreqを加え、またTDTreq、AK TPDUについてウィンドウを開くものと閉じるものの区別を設ける必要がある。但し、クラス2でフロー制御を用いない場合はED TPDUも用いられないのでOPENの状態はB01とB02に限定される。

一方、クラス2レスポンダのオートマトンモデルでも同様にして8つの状態が設定される。

5.2クラス2イニシエータの製品検証実験

次にTPクラス2のイニシエータに対して、製品検証実験の結果を示す。

クラス2イニシエータの方式1による検証のために、各状態に対して次のような判定系列を用いることができる。

①データ転送中の状態(B01からB32)に対して
TDTreq, TEXreq, AK-OPENW, DR
②N3, A21, A22, A31, A32, C1, C2に対して
TCONreq, DC, CC, TDTreq, DR
③N1に対して TCONreq
④N2, A11, A12に対して NCONconf, DR

①の判定系列では、TDTreq, TEXreq, AK-OPENWによって、状態B01からB32はDTまたEDを出力するために、それ以外の状態から区別される。

表3 TPクラス2イニシエータのオートマトンモデル(一部)

STATE	CLOSED *1			Wait for NC (WFNC *1)		Wait for CC (WFCC *1)		Wait before releasing (WBCL *1)	
INPUT	NC not assigned	NC assign-ing	NC assigned	not mpx	mpxed	not mpx	mpxed	not mpx	mpxed
	N1	N2	N3	A11	A12	A21	A22	A31	A32
TCONreq	A11 NCONreq	A12	A22 CR	A11	A12	A21	A22	A31	A32
TDISreq	N1	N2	N3	N1 NDISreq	N2	A31	A32	A31	A32
TCreq -other*2	N2 NCONreq	//	//	A12	//	A22	//	A32	//
NCON conf	//	N3	//	A21 CR	A22 CR	//	//	//	//
CC	//	//	N3 DR	//	//	B01	B02	C1 DR	C2 DR
DR	//	//	N3 DC	//	//	N1 NDISreq	N3	N1 NDISreq	N3

STATE	OPEN *1								CLOSING *1	
	window open				window closed					
	ED not sent		ED sent		ED not sent		ED sent			
INPUT	not mpx	mpxed	not mpx	mpxed	not mpx	mpxed	not mpx	mpxed	not mpx	mpxed
	B01	B02	B11	B12	B21	B22	B31	B32	C1	C2
TDTreq	B01 DT	B02 DT	B11 DT	B12 DT	B21 保留	B22 保留	B31 保留	B32 保留	C1	C2
TDTreq -last*3	B21 DT	B22 DT	B31 DT	B32 DT	//	//	//	//	//	//
TEXreq	B11 EX	B12 EX	B11 保留	B12 保留	B31 EX	B32 EX	B31 保留	B32 保留	C1	C2
TDISreq	C1 DR	C2 DR	C1 DR	C2 DR	C1 DR	C2 DR	C1 DR	C2 DR	C1	C2
TCreq -other	B02	B02	B12	B12	B22	B22	B32	B32	C2	C2
DR	N1 DC NDISreq	N3 DC	N1 DC NDISreq	N3 DC	N1 DC NDISreq	N3 DC	N1 DC NDISreq	N3 DC	N1 NDISreq	N3
AK -open*4	B01	B02	B11	B12	B01	B02	B11	B12	C1	C2

注 *1:これらの状態がX.224の遷移表の状態に対応 *2:検証対象以外のTCに対するTCONreq
*3:ウインドウを閉じるTDTreq *4:ウインドウを開くAK

またB01からB32の中で、この系列により、順にウィンドウが開いているか否か、EDを送信したか否か、他のTCが同じNCに多重化されているかが区別され、状態B01からB32が識別される。

また②の判定系列では、N3はTCONreqに対して送信されるCR TPDUにより他の状態から区別され、DC, CCでC1とC2が識別される。またA21とA22、A31とA32を区別するためには、これらの状態にCC TPDUを加えてデータ転送フェーズに移るか解放フェーズに移るかを調べる必要があるため、②の判定系列の内のCC,TDTreq,DRを用いる。

このように、クラス2イニシエータのオートマトンモデルでは各状態毎に用意された判定系列を用いて各状態を識別できるため、付録1に示した方法によってテスト系列を構成できる。

クラス2イニシエータの製品検証のためのテスト系列とそれを用いた実験結果の一部を図5に示す。

このテスト系列は5471入力からなり、遷移表のうち検証対象となる336の遷移全てを検証することができる。またクラス2イニシエータを検証するテスト・エイドの実効行数は335行であり、クラス0の検証に用いたものに、多重化に関するチェックを行うために検証対象以外のTCに対するTCONreq、TDISreqを発生する機能を加えている。

一方、方式2による製品検証ではデータ転送フェーズの状態B01からB32において、TDTreq、TEXreq、TDISreqを、また状態N1、N3においてTCONreqをプロトコル・テスタから非同期に与えられる発生指示によってコンソールから加える必要がある(これらのTプリミティブをそれぞれ、(tdt-req), (tex-req), (tdis-req), (tcon-req)で表す)。これにより、WBCL に対応する状態A31、A32以外の状態に対して検証を行うことができる。状態A31、A32に対する検証ができないのは、これらの状態に遷移させるためにはTS1タイマが動作中にコンソールからTDISreqを加えなければならず、これは困難であるからである。方式2による製品検証のために、次のような判定系列を用いる。

①状態B01, B02, B11, B12に対して
(tex-req), (tdt-req), DR
②状態B21, B22に対して
(tdt-req), (tex-req), DR

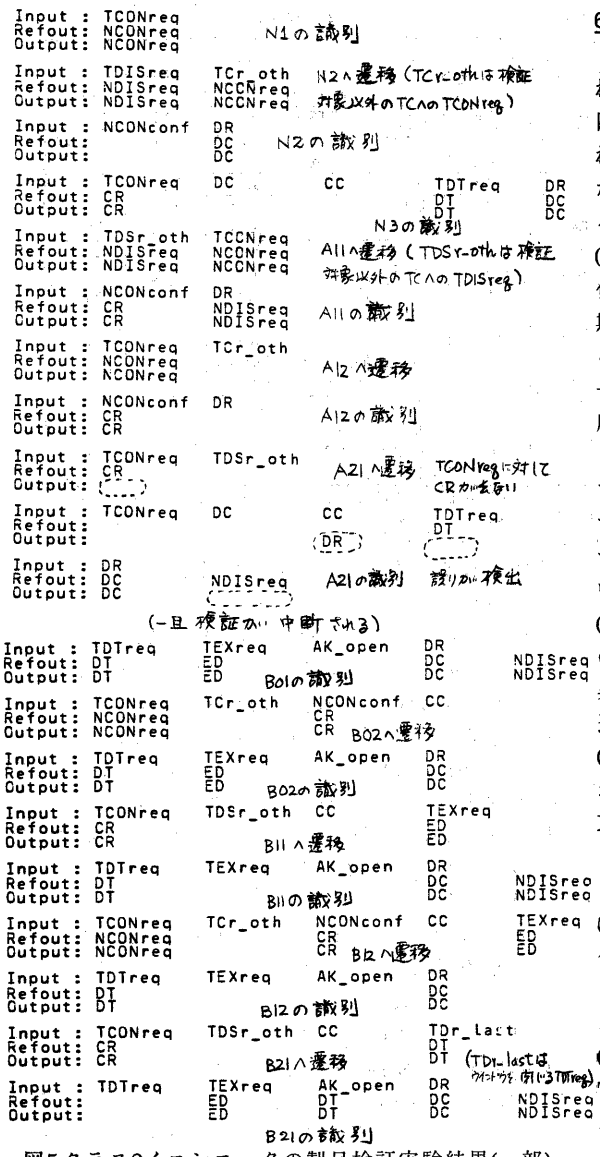

図5クラス2イニシエータの製品検証実験結果(一部)

③状態B31, B32に対して
(tdt-req), (tex-req), (tdis-req), DR

④状態N1, N3に対して (tcon-req)

⑤状態N2, A11, A12に対して NCONconf, DR

⑥状態A21, A22に対して CC, (tdt-req)

⑦状態C1, C2に対して DC, CC

方式2による製品検証ではオートマトンモデルの定める336の内236の遷移が検証可能である。

検証実験の結果、状態N2、N3でのTCONreqの処理等、多重化の処理に誤りが検出された。

6. TPクラス1の製品検証

6.1 クラス1のオートマトンモデル[3]

TPのクラス1ではネットワークレーヤから通知される誤りから回復するために障害後の再割付けと再同期の機能が特徴であり、オートマトンモデルはこれらの手順を正確に表現しなければならない。このためにX.224のクラス1の遷移表を以下の点を表現するよう詳細化する必要がある。

(1)再同期の手順はNRSTind受信および障害後の再割付に引き続いて行われるが、NRSTind受信後の再同期ではネットワーク内に滞留したTPDUを受信することがあるが、これらを無視しなければならない。一方、障害後の再割付に引き続く再同期ではTPの手順に従わないTPDUはプロトコル・エラーとして処理されなければならない。この動作はX.224の遷移表ではレスポンダの一部についてしか記述されておらず、イニシエータ、レスポンダのオートマトンモデルはコネクション確立、データ転送フェーズにおいて、別個の状態を設定する必要がある。

(2) イニシエータの解放フェーズにおいてTDISreqの受信と再同期開始のどちらが先かによって動作が異なる場合がある。オートマトンモデルはこれを記述できるような状態を有する必要がある。

(3) イニシエータの再同期の手順では、TTRタイマが作動中とタイムアウト後では異なる動作を行う。X.224の遷移表ではこれを条件(Predicate)を用いて表しているが、オートマトンモデルでは別個の状態で表現する。

(4) レスポンダではTCが解放された状態または再割付け中において、TCに割付けられるNCが切断されている場合と確立されている場合とを区別する必要がある。

(5) データ転送フェーズではクラス2と同様に、ED TPDUを送信済みかどうかを別個の状態を用いて表現する。

TPクラス1のイニシエータに対して、そのオートマトンモデルを規定すると、状態数34の状態遷移表で表現できる。そのデータ転送フェーズの部分を表4に示す。データ転送フェーズに対しては、X.224の遷移表はOPENとNCの再割付け中の状態OPEN-Rの2つを設定している。しかし製品検証のためのオートマトンモデルでは、表4に示すようにOPENに対して正常の場合と再同期が行われている場合に区別され、OPENの正常の場合とOPEN-Rは(5)に述べたようにED TPDUを送信したかどうかの場合分けが必要である。またOPENの再同期中の場合は、(1)に述べたリセット通知後か再割付け後か、(3)に述べたTTRタイマが動作中かタイムアウト後か、(5)に述べ

表4 TPクラス1イニシエータのオートマトンモデル (一部)

STATE	OPEN*1										OPEN and reassignment in progress (OPEN-R*1)	
	normal		resynchronization in progress									
			after NRSTind				during reassignment					
			TTR running		TTR run out		TTR running		TTR run out			
INPUT	ED not sent	ED sent	ED not sent	ED sent	ED not sent	ED sent	ED not sent	ED sent	ED not sent	ED sent	ED not sent	ED sent
	B01	B02	B11	B12	B13	B14	B21	B22	B23	B24	B31	B32
TDTreq	B01 DT	B02 DT	B11 保留	B12 保留	B13 保留	B14 保留	B21 保留	B22 保留	B23 保留	B24 保留	B31 保留	B32 保留
TEXreq	B02 ED	B02 保留	B11 保留	B12 保留	B13 保留	B14 保留	B21 保留	B22 保留	B23 保留	B24 保留	B31 保留	B32 保留
NCON conf	//	//	//	//	//	//	//	//	//	//	B21 RJ	B22 RJ
NRSTind	B11 NRSTrsp RJ	B12 NRSTrsp RJ	B11 NRSTrsp RJ	B12 NRSTrsp RJ	N1*2 NRSTrsp NDISreq	N1 NRSTrsp NDISreq	B11 NRSTrsp RJ	B12 NRSTrsp RJ	N1 NRSTrsp NDISreq	N1 NRSTrsp NDISreq	//	//
DT= *3	B01	B02	B11	B12	B13	B14	C21*2 DR	C21 DR	C22*2 DR	C22 DR	//	//
RJ	B01 DT	B02 ED*4 DT	B01 DT	B02 ED DT	B01 DT	B02 ED DT	B01 DT	B02 ED DT	B01 DT	B02 ED DT	//	//

注 *1 : OPEN、OPEN-RがX.224の遷移表の状態に対応 *2 : N1 (CLOSED且つNC割付無し)、C21 (CLOSING且つ再同期が先に開始且つTTR作動中)、C22 (CLOSING且つ再同期が先に開始且つTTRタイムアウト) *3 : 正しい順序で受信されるDT TPDU *4 : 最新の標準ではED TPDUはNRSTindを受信した時に送信される

たようにED TPDUを送信したかどうかの場合分けが行われる。このようにして、データ転送フェーズには12個の状態が設定される。

レスポンダでも同様にして27個の状態が設定される。

6.2クラス1イニシエータの製品検証実験

クラス1イニシエータは上述のように複雑な構造を持つため、これまで述べてきたクラス0、2と異なり、状態によってはそれを識別するために複数の系列を必要とするものがある。そこで、

(1)判定系列が複数の系列からなるような状態を識別するためには、その状態に判定系列を構成する全ての系列を加える。

(2)オートマトンの状態を検証対象の状態に設定するためには、別の状態を経由する必要があるが、この状態は単一の判定系列で識別される必要がある。

(3)従って、状態を設定する過程で経由する状態を、各状態毎に明確に定義する必要がある(クラス0、2では遷移表で隣の状態を経由していた)。

という方法を用いてテスト系列を構成する必要がある。この場合のテスト系列の構成法の詳細を付録2に示す。

クラス1イニシエータの方式1による製品検証にために、データ転送フェーズの各状態に対して以下の判定系列を用いることができる。

①B01, B02に対して TDTreq, TEXreq
②B11, B12に対して NRSTind, DT=, RJ
③B13, B14に対して NRSTindとDT=, RJ
④B21, B22に対して NRSTind, RJとDT=
⑤B23, B24に対して NRSTindとRJとDT=
⑥B31, B32に対して NCONconf, RJ

このうち④を例にあげると、状態B21をTTRタイマがタイムアウトした状態(B13, B14, B23, B24)と区別するためには、NRSTindが必要で、ED TPDU送信済みの状態(B02, B12, B22)と区別するためにはRJが必要である。しかしNRSTind, RJをB21に加えると状態B01、B11と区別できなくなる。B01、B11から識別するためにはDT=が必要であるが、これを加えるとB22と区別できなくなる。従って状態B21を識別するためには④の系列を別々にB21に加える必要がある。

図6にクラス1イニシエータの製品検証実験の結果の一部を示す。この実験に用いたテスト系列は8248入力から成る。

一方、クラス1イニシエータに対する方式2による製品検証も、状態B01、B02においてコンソールからTDTreq、TEXreqを加えることにより、クラス2イニシエータと同様に、CR TPDU送信直後にTDISreqを受けた場合以外の状態に対し実行可能である。方式2による製品検証のために

①B01, B02に対して (tex-req), (tdt-req)
②それ以外のデータ転送フェーズの各状態に対しては方式1による製品検証と同じ系列

という判定系列を使用できる。

実験の結果、再同期中にTTRタイマがタイムアウトした時点でNCを切断してしまう、再割付けに続く再同期においてDT等のTPDUに対してプロトコル・エラーの処理をしない等の誤りが発見された。これは、筆者等が作成したTPプログラムは再同期中の状態としてオートマトンモデルのB11とB12の2つの状態しか想定していないことを意味する。このような

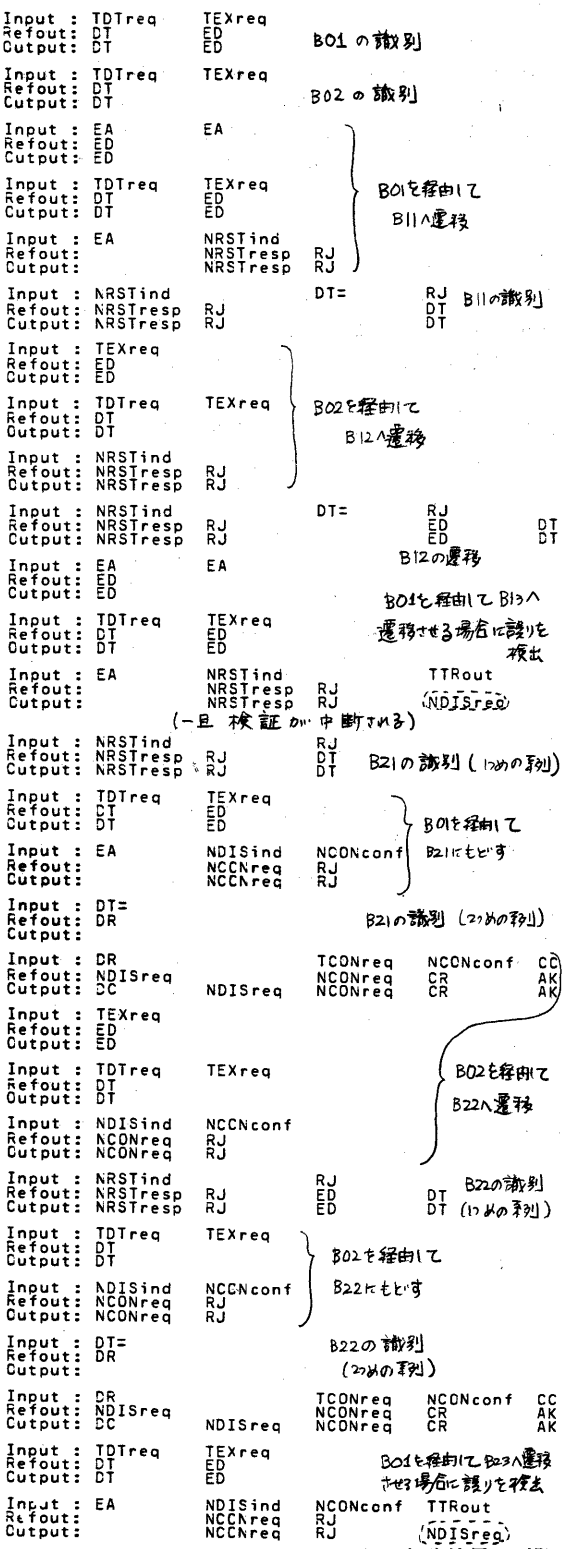

図6クラス1イニシエータの製品検証実験結果(一部)

誤りはX.224において再同期の手順を明確に規定していないために生じたものであると考えられる。

7.結論

以上のように、オートマトンモデルに基づく製品検証をTPに対して適用し、その有用性を示した。

一般に、標準プロトコルとの整合性を検証するプロトコル製品検証には、

(i) プロトコル要素の送出の順序等のプロトコル手順に関する検査

(ii) データを運ぶプロトコル要素に付加されたシーケンス番号やタイムアウト回数の管理のような同様の手順が複数回繰り返されるような動作に関する検査

(iii)大量データ転送等を行う過負荷検査

等の検査項目が考えられる。表1に述べたTPのプロトコル機構では、コネクション確立、解放、障害後の再割付け、再同期などが(i)の検査、DTTPDU番号付け、フロー制御等が(ii)の検査の対象となり、過負荷検査は総合的な検査を目的とすると考えられる。

本稿で述べたオートマトンモデルに基づく製品検証では、(i)の検査を有効に実行することができるが、(ii)の検査に対しては、シーケンス番号個々の値に対する動作等を有効に検査するのは困難である。しかしオートマトンモデルを規定する場合に、シーケンス番号を持つプロトコル要素に対しては正しい順序のものとそうでないものを別の入力として扱い、またフロー制御ではウインドウが開いているか閉じているかを、多重化では多重化されているかいないかを別の状態で表現することにより、現実的には有効な製品検証が行われていると考えられる。また(iii)については別途検討・実験しており、本方式と過負荷検査とを組み合わせてより有効な検証方式とする必要があろう。

また本稿で述べた方式1、2の2つの検証方式は、TPの製品検証に対しては共に有効であることが判明した。但し、方式2では現在TLのみが単独で存在する状況での製品検証を行っているが、今後はTLの上位にSL等が実装されている状況での製品検証を検討してゆく必要がある。

最後に、日頃御指導頂くKDD研究所鍛治所長、野坂副所長、深田次長、小野情報処理研究室長に感謝します。

参考文献 [1]: CCITT, Rec. X.214, X.224, March, 1984
[2]: 鈴木,加藤,"OSIトランスポート・プロトコルのインプリメントと製品検証", 情処学会 分散処理システム研究会, 22-9, May 1984

[3]: 加藤,鈴木,"OSIトランスポート・プロトコル・クラス1、2の製品検証のためのオートマトンモデル", 第29回情処全大, 4H-4, Sept., 1984
[4]: 斉藤,加藤,猪瀬,"オートマトン・モデルによるHDLCプロトコル製品検証の一方式",信学論(D),J63-D,8, Aug., 1980

付録1 状態毎の判定系列を用いたテスト系列の構成

オートマトンM

$M=(Q, I, O, \delta, \omega)$ (1)

Q:状態集合, I:入力集合, O:出力集合

$\delta: Q \times I \rightarrow Q$ 状態遷移関数

$\omega: Q \times I \rightarrow O$ 出力関数

において

(i)状態数をn、入力の数をmとする。

(ii) Mの状態q_iに対する判定系列をx_{Di}とする。x_{Di}は、Mの任意の状態q_i、q_jに対して

$q_i \neq q_j$ ならば $\omega(q_i, x_{Di}) \neq \omega(q_j, x_{Di})$

を満たす。以下では状態q_iに判定系列x_{Di}を加えた場合の、q_iの固有の出力系列をz_i、遷移先の状態を$q_i^{(l)}$と略記する。

(iii) Mの状態をq_iからq_jに遷移させる遷移系列を$x_T(q_i, q_j)$で表す。

以下に製品検証のためのテスト系列の入出力応答を示す。オートマトンは状態q_1から出発するものとし、まず次のような入出力応答によりn個の状態が存在することを確認する。

$$x: x_{D1}\ x_T(q_1^{(l)}, q_2)\ x_{D2}\ \cdots\cdots\ x_T(q_{n-1}^{(l)}, q_n)\ x_{Dn} \quad (2)$$
$$z: z_1 \text{ ——} z_2 \cdots\cdots \text{—— } z_n$$

次に状態をq_1に戻す。

$$x: x_T(q_n^{(l)}, q_1)\ x_{D1}\ x_T(q_1^{(l)}, q_n)\ x_{Dn}\ x_T(q_n^{(l)}, q_1) \quad (3)$$
$$z: \text{——} z_1 \text{——} z_n \text{——}$$

更に以下の入出力応答により状態q_1の各入力に対する出力と遷移先の状態を調べる。

$$x: x_1\ x_{D1}{}^{(1)}(1)\ x_T(q_1^{(1)(l)}(1), q_n)\ x_{Dn}\ x_T(q_n^{(l)}, q_1)\ x_2\ x_{D1}{}^{(1)}(2)$$
$$z: z_1(1)\ z_1{}^{(1)}(1) \text{——} z_n \text{——} z_1(2)\ z_1{}^{(1)}(2)$$
$$x_T(q_1^{(1)(l)}(2), q_n)\ x_{Dn}\ x_T(q_n^{(l)}, q_1) \cdots x_m\ x_{D1}{}^{(1)}(m) \quad (4)$$
$$\text{——} z_n \text{——} \cdots z_1(m)\ z_1{}^{(1)}(m)$$

次に状態をq_2に移す。式(2)において$x_T(q_1^{(l)}, q_2)$による$q_1^{(l)} \rightarrow q_2$の遷移が確認されているから

$$x: x_T(q_1^{(1)(l)}(m), q_1)\ x_{D1}\ x_T(q_1^{(l)}, q_2) \quad (5)$$
$$z: \text{——} z_1 \text{——}$$

という系列を加えればよい。状態s_2以下の検証は式(4)に示した状態s_1と同様である。

付録2 状態毎の判定系列が複数の系列からなる場合のテスト系列の構成

オートマトンMに対して

(i) Mの状態q_iに対する判定系列として用意された系列の個数を$d(q_i)$、k番目の系列を$x_D(q_i,k)$とする。Mの任意の状態q_i、q_jに対して

$q_i \neq q_j$ ならば $\omega(q_i, x_D(q_i,k)) \neq \omega(q_j, x_D(q_i,k))$

となるk($k \leqq d(q_i)$)が存在する。

(ii) 状態q_iに対して状態を設定する過程で経由する状態$prec(q_i)$が定められている。$prec(q_i)$は1つの系列で識別できる。即ち$d(prec(q_i))=1$を満たす。

以下にテスト系列の構成方法を示す。ここで、

$$\left\{x(j)\right\}\Big|_{j=1}^{l}$$

はjを1からlまで変化させて系列x(j)を連結することを示す($l=0$ならば空列を示す)。オートマトンは状態$prec(q_1)$から出発するものとする。まず状態をq_1に遷移させるためにつぎの入力列を用いる。

$$x_D(prec(q_1), 1)\ \ x_T(\delta(prec(q_1), x_D(prec(q_1), 1)), q_1) \quad (6)$$

次に状態q_1が存在することを確認するために、q_1の判定系列に属する系列を順に加える。

$$\left\{x_D(q_1, j)\ \ x_T(\delta(q_1, x_D(q_1, j)), prec(q_1))\ \ x_D(prec(q_1), 1)\right.$$
$$\left.x_T(\delta(prec(q_1), x_D(prec(q_1), 1)), q_1)\right\}\Big|_{j=1}^{d(q_1)-1}\ x_D(q_1, d(q_1)) \quad (7)$$

Mの状態を$prec(q_2)$を経由してq_2に移す。

$$x_T(\delta(q_1, x_D(q_1, d(q_1))), prec(q_2))\ \ x_D(prec(q_2), 1)$$
$$x_T(\delta(prec(q_2), x_D(prec(q_2), 1)), q_2) \quad (8)$$

状態q_2からq_nが存在することの確認は式(7)と同様に行う。その後次の入力列を用いて状態をq_1に戻す。

$$x_T(\delta(q_n, x_D(q_n, d(q_n))), prec(q_1))\ \ x_D(prec(q_1), 1)$$
$$x_T(\delta(prec(q_1), x_D(prec(q_1), 1)), q_1) \quad (9)$$

次に状態q_1の各入力 x_k (k=1 … m)に対する出力と遷移先の状態を調べる。

$$\left\{\left\{x_k\ \ x_D(\delta(q_1, x_k), j)\ \ x_T(\delta(\delta(q_1, x_k), x_D(\delta(q_1, x_k), j)), prec(q_1))\right.\right.$$
$$\left.\left.x_D(prec(q_1), 1)\ \ x_T(\delta(prec(q_1), x_D(prec(q_1), 1)), q_1)\right\}\Big|_{i=1}^{d(\delta(q_1, x_k))}\right\}\Big|_{k=1}^{m} \quad (10)$$

状態q_2からq_n対する出力と遷移先の確認は式(10)と同様に行われる。