

通信制御手順の検証手法に関する一検討

川井 清幸, 周東 晃四郎, 浅野 正一郎, 野村 民也
(東京大学 宇宙航空研究所)

1. まえがき

データ通信システムに, 階層構造をもたせたネットワーク・アーキテクチャの思想が導入されて, これまでに数多くのアーキテクチャが発表されている。ネットワーク・アーキテクチャが, システムとして実現されるためには, アーキテクチャを構成する各階層レベルにおけるプロトコルが厳密に規定されることがまず要求される。これに関してプロトコルの規定内容を明確に表現する手段として種々のプロトコル記述法が報告されている。[1]

一方, プロトコルで規定する論理の正当性の検証手法に関して, 幾つかの報告がある。[2][3] これらはプロトコルで規定する論理の記述法により手法的に異なるものとなっているが, 一般にプロトコル検証手法として次の機能を具備することが望ましいと考える。

- (A) 広汎なプロトコルに適用可能であること。
- (B) 自動化を前提とした系統的な手法であること。
- (C) 広汎な検証項目を取扱い得ること。

筆者らは, これらの機能を備えた検証手法を, 状態遷移によりプロトコル論理構造を記述した上で検討し, あわせてこの手法に基づいたプロトコル検証プログラムを試作したので御報告する。

2. 状態遷移記述法

プロトコルを図1. に示すように, 相互交信する2つの有限オートマトンでモデル化する。さらにオートマトン間の相互交信を合成オートマトンにより表現するが, その際にオートマトン間の相互交信信号として次の3者を想定している。

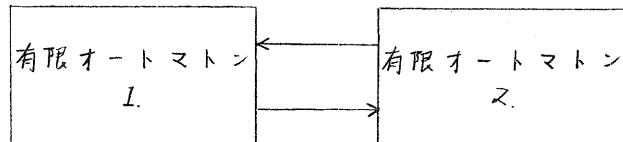


図1. プロトコルのモデル

- (a) 一方の有限オートマトンから出力され, 他方に伝送されてそこで状態遷移を起こす交信信号。
- (b) 一方の有限オートマトンから出力され, 他方に伝送されるが, 受信側では状態遷移を起こさない交信信号。
- (c) 伝送中に紛失する交信信号。

以上の場合の想定は, 自明のことではあるが, 送信側オートマトンから交信信号が送出される際には, 信号の正常受信がなされかつ受信側オートマトンと状態の同期が保たれているか((a)), あるいは同期が保たれていないか((b)), さらに信号が紛失するか((c))のいずれかの可能性が存在する。またこのことは, 送信側オートマトンで交信信号の出力を生じさせる要因とは独

立に，交信信号出力を行うときには必ずこの可能性がある．これから交信信号出力を行うときには，常に上記の場合を組合わせて考慮することが，以降の処理を系統的に進める上で役立つ．

この点に関しては Bochkann [5] も指摘している通りであるが，これにあわせて，以下に述べるように交信信号の出力と，それを生じさせる要因とを区別した状態遷移を考慮することが，以降の処理に役立つ．[4]

ス. 1 ミーリー・モデルの変換

プロトコルを状態遷移図(表)で記述する場合，ミーリー・モデルで表わされるのが通例であるので，本稿ではミーリー・モデルで与えられたプロトコルを検証することを前提として以降の議論を進める．ミーリー・モデルにより，2つのオートマトンの合成状態遷移を記述することを想定すると図2. に示すような記述が行なわれることになろう．この場合，各アークには，各々のオートマトンの

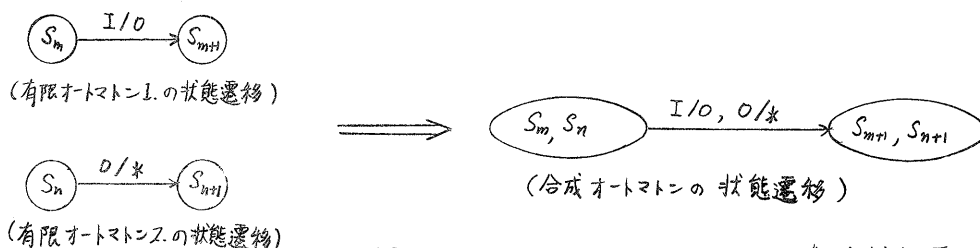


図2. ミーリー・モデルによる合成状態遷移図

遷移が併記されることが必要となる．

一方，Zaffatopulo [4] も指摘しているように，ある状態への入力要因と出力信号とを分離して記述することを考えると，図3. に示す記述が行ない得る．



図3. ミーリー・モデルからの変換

この場合，送信側オートマトンからの交信信号は前述の3通りの場合に従って受信側オートマトンの入力要因となるという事実から，各アーク上の記述が簡易なものとなり得る．図2. の場合は図4. に示す合成状態図の作成が可能になる．

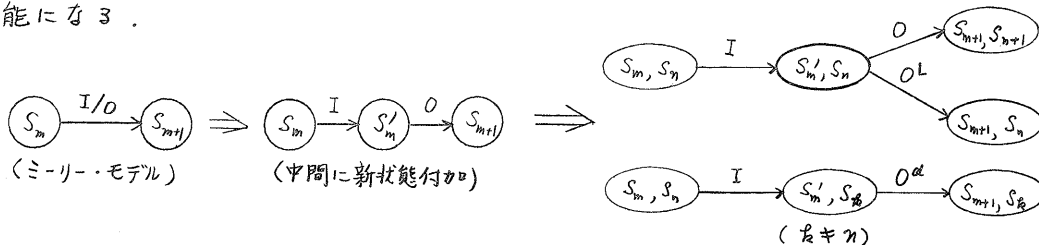


図4. 相互交信信号で起こり得る状態遷移

この判点から，本稿では入力要因と出力信号を分離し，中間状態を設けることによる状態遷移記述法を採用することにしている．

ミーリー・モデルから本記述モデルへの変換は次のように一義的になされる．

ミーリー・モデルが形式的に6項組 $\langle S, s_0, I, O, f, \varphi \rangle$ で表されているとする。

ここに

$$\left\{ \begin{array}{l} S: \text{状態の集合} \\ s_0: \text{初期状態} \\ I: \text{入力要素の集合} \\ O: \text{出力信号の集合} \\ f: I \times S \rightarrow S, \text{ 状態遷移関数} \\ \varphi: I \times S \rightarrow O, \text{ 出力関数} \end{array} \right.$$

変換を図3に従って説明する。ミーリー・モデル上で状態 s_m から s_{m+1} への遷移には交信信号出力が存在するため、入力要素と出力とを分離するため中間に新状態 s'_m を設ける。 s'_m は出力 o_m に対応して付加され、この新状態 s'_m は入力アーク(I_e)と出力アーク(O_m)の間にノードとして入れられる。

ここで $S' = \{s'_m\}$ とすれば、 $S \cap S' = \emptyset$ である。また、 $S'' = S \cup S'$ 、 $I' = I \cup O$ 、 $I \cap O = \emptyset$ とすると、変換された状態遷移モデルは形式的に5項組 $\langle S'', s_0, I', f, \varphi \rangle$ で表わされる。これは付加した新状態の集合 S' だけ冗長であるが、ミーリー・モデルと全く同一の機能を表している。

3. 合成オートマトン

2つの有限オートマトンを1つのオートマトンに合成することで、交信信号の伝送を含め2者の動作を一括して記述する。この合成オートマトンは、前節で述べた状態遷移記述による2つの有限オートマトンの状態数の積の状態数をもつことになる。

3.1 内部要因による状態遷移

2節で得た状態遷移モデルは相互交信に着目したモデルであり、前述の3種の交信信号伝送の場合、すなわち、(a)、(c)、(c')の状態遷移の検討を容易にしている。

また、各々のオートマトンの内部要因による状態遷移は、各オートマトンで独立に行なわれる。

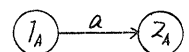
2節のミーリー・モデルの変換で新状態が付加されていることは、オートマトン合成の操作を系統的で簡易なアルゴリズムで表現することを可能としている。この新状態は出力発生を意味する過渡的な状態であると思ふことができ、以後付加された新状態を出力発生状態と呼ぶことにする。この出力発生状態は、現実の機械の動作に於いて時間零で通過する過渡的な状態であると解釈する。したがって、一方のオートマトンがこの出力発生状態にとどまっている時、他方のオートマトンに内部要因の生起はないと仮定して以降の処理をおこなう。

3.2 オートマトン合成

2つの有限オートマトンの状態遷移を表わす合成オートマトンの状態遷移は、2節で述べた相互交信信号の伝送上の場合、すなわち、(a)、(c)、(c')の各場合の出力遷移と3.1項で述べたように出力発生状態以外で生起する内部要因による遷移との4種類の状態遷移を考えればよい事になる。これらの遷移について、以下に示す操作でオートマトンの合成をおこなう。

(1) 相互交信に関する状態遷移に着目したオートマトンの合成アルゴリズム

i) 一方のオートマトンの出力遷移を選ぶ出す。



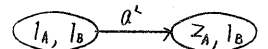
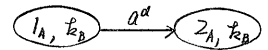
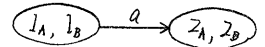
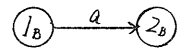
ii) 他方側オートマトンから信号の一致する

入カ遷移を選ぶ.

iii) ステップ (a) の状態遷移 (通信成立イベント) として i), ii) の対を作る.

iv) ステップ (b) の状態遷移 (無視イベント) を作る.
($l_B \neq l_B$)

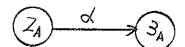
v) ステップ (c) の状態遷移 (紛失イベント) を作る.



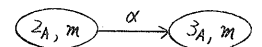
vi) 双方のオートマトンの出力遷移の処理が終了するまで, i) ~ v) を繰り返す.
終了したら (2) へ移る.

(2) 内部要因による状態遷移に着目したオートマトン合成アルゴリズム

i) 一方のオートマトンの内部要因による状態遷移を選ぶ.



ii) 他方オートマトンの出力発生状態を除く全ての状態 m に対して状態遷移を作る.



iii) 双方のオートマトンの内部要因の処理が終了するまで i) ~ ii) を繰り返す.
本処理の終了でオートマトン合成は完了する.

以上の処理により, 合成オートマトンの起こり得る状態遷移は全て規定され, ステップの相互通信オートマトン間における信号の一致性, 同期条件をも同時に考慮されている.

4. 検証アルゴリズム

Zoffatopulo [4] は, 自動化された検証法として, ステップの有限オートマトンのイベント系列の対を作り, 相互通信系としての実行可能性やデッドロックのチェックをするアルゴリズムを提案している. しかし, このアルゴリズムはオートマトンにループが含まれない場合にだけ適用可能である. これはオートマトンにループが含まれる場合, イベント系列が無数に存在し, 無限長のイベント系列も定義されるからである. 一般にプロトコルには繰返し動作が規定内容に含まれるのでプロトコルをモデル化した有限オートマトンにはループが含まれる.

プロトコル検証としてイベントや状態の系列に着目するアルゴリズムは, ループを含むオートマトンへの適用が困難であり適用範囲が著しく限定される. しかしステップの有限オートマトンを合成したオートマトンは有限状態であるので, これに着目すると状態遷移はループを含む場合にも有限系列で検定が可能である.

合成オートマトンは, ステップの相互通信オートマトンの動作を完全に記述している. この状態数が少なければ, 状態遷移図から目視によって動作の確認が可能である. しかし, 状態遷移図が複雑になれば目視では動作の詳細まで全て網羅することが困難になる. したがって合成オートマトンの動作を網羅し, 検証項目を全て検討できる目視によらない検証アルゴリズムが必要であり, 以下にこれについて述べる.

4. 1 状態木

本稿で述べる検証アルゴリズムは, 基本的には初期状態から出発する遷移可能な状態遷移を順次実行して, 遷移した状態の木を作成することである. ここで, 木の長さを有限とするために, 木を構成する状態の系列からなる枝は, ループが

存在する場合には、ループを1回通った時点で先へ枝を伸ばさない事とする。しかし、木は有限オートマトンの初期状態からの到達可能な状態遷移を全て含むので、このような木によりオートマトンの動作は全て記述できるという。図5. にこの簡単な例を示す。

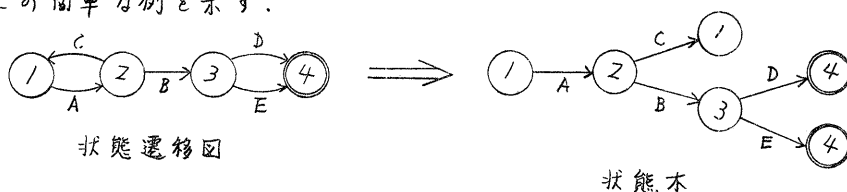


図5. 状態遷移図と状態木の対応例

4. 2 状態木の作成アルゴリズム

前項で述べたように、木の長さを有限にとどめるために、2種のフラグを処理の過程に用いる。1つは初期状態から状態遷移を開始する状態木の作成過程で、ある状態に達した事を示すためのもので、状態フラグと呼ぶことにする。もう一方のフラグは、ある状態から複数の状態遷移が定義されていることを示すもので分岐フラグと呼ぶことにする。

(i) 状態フラグの役割

状態フラグの付された状態は初期状態から始まる状態系列を構成している。したがって、ある状態からの遷移先状態に状態フラグがすでに付されているとすると、その遷移はすでに通過してきた状態に再び戻ることになる。すなわち、その遷移をとるとループを形成することになるので、遷移を進める事を中止する。遷移先の状態にはループが形成される状態であることを示すために状態フラグを負にする。

(ii) 分岐フラグの役割

状態木の作成では、状態遷移を順次進めて行く。したがって、ある状態からの遷移先が複数ある場合は、1つの状態遷移を実行した後に、他の状態遷移の検定が必要である事を示すために分岐フラグを用いる。処理の過程で分岐フラグの付された状態に通時戻り、以前に検定していない遷移を実行し状態木の枝を伸ばす。分岐フラグは、その状態からの遷移が全て検定終了した時点でクリアされる。図6. に状態木作成の過程を示す。

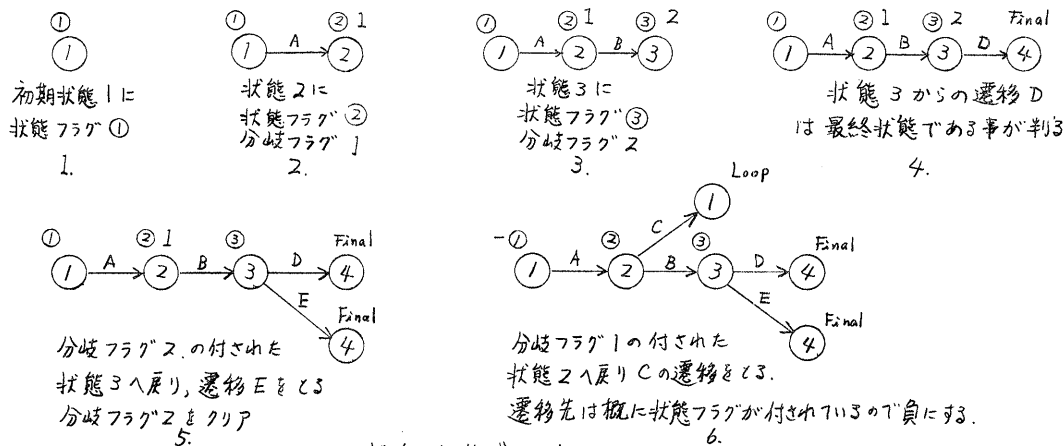


図6. 状態木作成過程

4. 3 検証項目

本稿の検証アルゴリズムは次のものを、その対象としている。

(1) 状態の到達可能性

合成オートマトンの全ての状態について、初期状態からの到達可能性を検定する。状態遷移の木が得られれば自明となる。

(2) 最終状態へ達する状態遷移系列

初期状態から所定の最終状態への到達系列の存在を検定する。最終状態へ到る枝を逆に初期状態へ戻ることによって遷移系列が示される。

(3) デッドロック

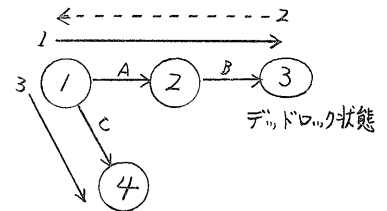
デッドロックとは2つのオートマトンが互いに待つの状態になる、いわゆるすくみの状態であると定義する。合成オートマトンでは遷移先が定義されていない状態として検定される。またデッドロックへ到る遷移系列を示す。

(4) ループ

ループは、プロトコルの規定内容による繰り返し動作を表わす正常なものと、規定ミスによって正常な動作へ復帰できずに無限回同じ動作を繰り返す異常なものがある。したがって完全なループの判定は意味論的にも検討を加えなければならぬ。意味論的判定は計算機では困難であるので、本稿では、構造的に明らかな異常ループだけを検定の対象とする。詳細は以降の項で述べる。

4. 4 デッドロック検定アルゴリズム

図7. の例によりデッドロック検定アルゴリズムの概要を述べる。状態3がデッドロック状態であるとする。



(1) 枝を伸ばして状態3に到達すると、この状態がデッドロック状態であることが判明する。

(1の実線)

(2) 分岐のある状態1にまで逆戻りをとする。(

2の点線)ただし状態2, 3にはデッドロックへ到る状態であることを状態フラグに記入する。

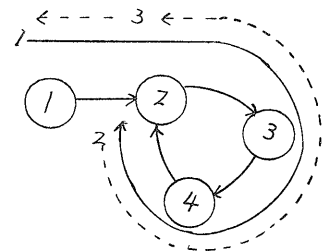
図7. デッドロック検定

(3) 状態1へ戻ると、以前実行した遷移(A)と異なる遷移(C)をとる、(3の実線)ただし(2)における遷移の逆戻り操作で分岐フラグに戻った以前に最終状態に到達可能な状態に戻った場合は、その時点で逆戻りを中止することができ。

4. 5 ループ検定アルゴリズム

4. 5. 1 構造的異常ループ検定

図8. に示すように、ループ内の状態から外部への遷移が不可能な場合、このループを構造的異常ループとよび検定の対象とする。図8. を例にとり構造的異常ループ検定アルゴリズムを簡単に述べる。



(1) 実線1の遷移系列をとるとループである事が判明し、状態2の状態フラグを負にする。

(2) この遷移系列がループなので遷移の逆戻りをする。(点線2)ただし状態3, 4の状態フラグはクリアする。

図8. 構造的異常ループ

そして状態 2 に異常ループを示すフラグを付する.

4. 5. 2 ループ検定アルゴリズム

(2) 遷移の逆戻りを開始する(点線2)

☑ 9. 12-7°

(4) ただし、状態3→状態5の遷移(E)が信号紛失イベントの場合には無視して遷移の逆戻りを続行する。

本稿2節から4節までで述べたアルゴリズムをFACOM 230-38 を使用しプロトコル検証プログラムの試作を行った. このプログラムの概要を述べる.

- この検証プログラムで検証できるプロトコルは次に述べるものに限定される。
- 個別に規定された2つのミーリー・モデルで表わされるプロトコルである。
- ミーリー・モデルの状態数，信号数の最大は40。

5. 2 検証プログラムの入力形式

5. 3 検証プログラムの結果出力

= 7 =

5. 4. 1 被験プロトコルモデル

検証プログラムの入力として、図11. に示す PRIMARY, SECONDARY の接続切断を行うモデルを用いる。

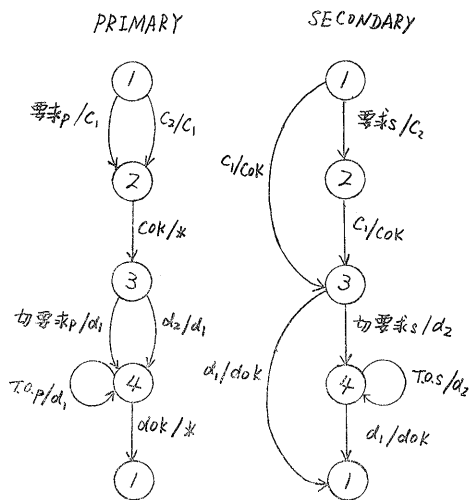


図11. 被験モデル

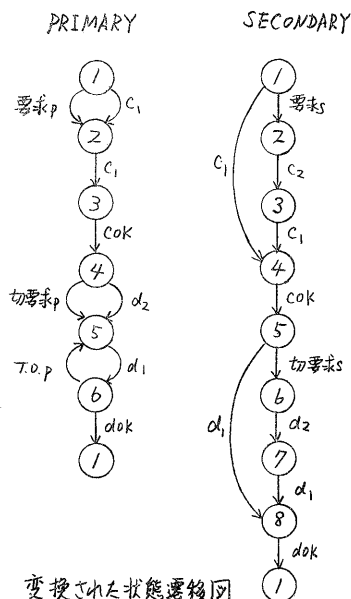
新状態名	PRIMARY 状態名	SECONDARY 状態名
1	1	1
2	1 0 0 1	2 0 0 1
3	2	2
4	3	2 0 0 2
5	1 0 0 3	3
6	4	2 0 0 4
7		4
8		2 0 0 6

(表中の1000番台, 2000番台の状態はミラー・モデルに付加した出力発生状態である。)

表1. 状態名変換表

5. 4. 2 合成オートマトン

入力として与えられたミラー・モデルは図12. に示す状態遷移図に変換される。そして、この2つの状態遷移図を合成したオートマトンが表2. に示すような状態遷移表として出力される。ただし、状態名は表1. に従って変換されている。PRIMARY, SECONDARYの状態数は各々6, 8であるから合成オートマトンの状態数は48である。参考のために合成状態遷移図を図13. に示す。



変換された状態遷移図

図12.

遷移 始状態	遷移先状態		遷移イベント		初期状態 からの到達 可能性	最終状態 への到達 可能性
	1	2	1	2		
1, 1	2, 1	1, 2	要求p	要求s	○	○
1, 2	2, 3	1, 3	C ₂ 受信	C ₂ 紛失	○	○
1, 8	1, 1		dok無視		X	X
6, 2	6, 3		C ₂ 無視		○	X

合成オートマトンの状態遷移表の一部

(表中の到達可能性は検証結果として記入される。)

表2.

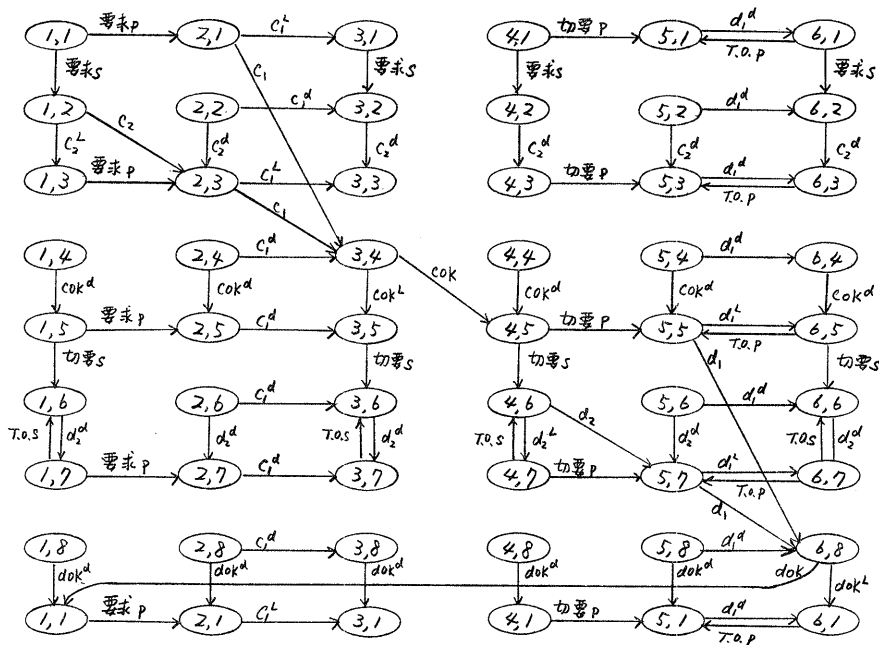


図13. 被験モデルの合成状態遷移図

5. 4. 3 動作例の結果

(1) 最終状態へ到達可能な状態遷移系列.

遷移順	1	2	3	4	5	6	7	8	9
1	1, 1	2, 1	3, 4	4, 5	5, 5	6, 8			
2	1, 1	2, 1	3, 4	4, 5	5, 5	6, 5			
3	1, 1	2, 1	3, 4	4, 5	5, 5	6, 6	6, 7	5, 7	
4	1, 1	2, 1	3, 4	4, 5	5, 5	6, 5	6, 6	6, 7	5, 7
5	1, 1	2, 1	3, 4	4, 5	5, 5	6, 5	6, 6	6, 7	
6	1, 1	2, 1	3, 4	4, 5	4, 6				
7	1, 1	2, 1	3, 4	4, 5	4, 6	4, 7			
8	1, 1	1, 2	2, 3						
9	1, 1	1, 2	1, 3						

(2) 構造的異常ループへ達する系列

遷移順	1	2	3	4	5	6	7	8	9
1	1, 1	2, 1	3, 4	4, 5	5, 5	6, 8	6, 1	6, 2	6, 3
2	1, 1	2, 1	3, 4	4, 5	5, 5	6, 8	6, 1	6, 2	
3	1, 1	2, 1	3, 4	4, 5	5, 5	6, 8	6, 1		
4	1, 1	2, 1	3, 4	3, 6	3, 7				
5	1, 1	2, 1	3, 4	3, 6					

この系列から $(6, 8) \rightarrow (6, 1)$ の状態遷移をすると異常ループになることが判る。このときの遷移イベントは状態遷移表から dok 紛失である。すなわち、このモデルには dok 紛失に対する回復規定がなされていないことが明らかになる。

また、同様に $(3, 4) \rightarrow (3, 6)$ の状態遷移から、このときの遷移イベントが cok 紛失であり、やはり cok 紛失に対する回復規定がなされていないことが明らかになる。

(3) デッドロックへ達する系列

状態 種類	1	2	3	4	5
1	1, 1	2, 1	3, 1	3, 2	3, 3
2	1, 1	2, 1	3, 1	3, 2	
3	1, 1	2, 1	3, 1		

前項のループの場合と同様に(2, 1)→(3, 1)の状態遷移をするとデッドロックに陥いることが示されている。このときの遷移イベントはC紛失である。すなわちCの紛失に対する回復規定がなされていない。

6. おわりに

本稿では、一般的な手段である状態遷移法を用いるプロトコル検証の一手法を述べた。2つのオートマトンを合成したオートマトンに基づいて目視によらずに系統的にプロトコル検証を行ない得るアルゴリズムを提案し、これをプログラム化して手法の確認を行なっている。その結果、初期の目的である一括した検証項目に対して自動化が可能であることが確認された。しかし、この試作プログラムは使用した計算機(FA COM 230-38)の記憶容量(64kWord)の関係から、被験プロトコルの適用範囲が制限されている。この点は現在検討を進めている。

概に指摘されているように(3)状態遷移的検証法は状態数の増加に伴う実現性が問題となるが、状態数が少ないコントロール・フェイズのプロトコル検証には適している。データ転送フェイズに対しても原理的には適用可能であるが、一般には状態数の増加が伴う。したがって、被験プロトコルの適用範囲を広くするためには状態数の増加に対する適用限界の検討が必要とされる。これは今後の課題である。

謝辞 本研究を進めるにあたり、日頃御指導をいただいている本学宇宙航空研究所 二宮敬虔助教授、ならびに二宮研究室の皆様には謝意を表します。

参考文献

- [1] 友永, 阿部, 宮沢, 河岡 "プロトコル記述法の一検討"
コンピュータネットワーク研資 CN-12-2 (1977)
- [2] Sunshine S.C. "Survey of Protocol Definition and Verification Techniques"
Computer network protocols, Université de Liège (1978)
- [3] 河岡, 友永, 高橋 "プロトコル記述方法と検証方法について"
ソフトウェア工学研資 (1978)
- [4] Zaffiropoulos P. "A New Approach to Protocol Validation"
IEEE ICC conference (1977)
- [5] Bochmann G.V. "Finite State Description of Communication Protocols"
Computer network protocols, Université de Liège (1978)
- [6] IBM "System Network Architecture Format and Protocol Reference Manual: Architecture" IBM (1976)