通信プロトコルのエラーリカバリ性自動検証の一方式

4T-1

加藤良司[1]　東野輝夫[1]　谷口健一[1]　森将豪[2]　二宮清[3]
大阪大学 基礎工学部[1]　滋賀大学[2]　ダイキン工業[3*]

1 はじめに

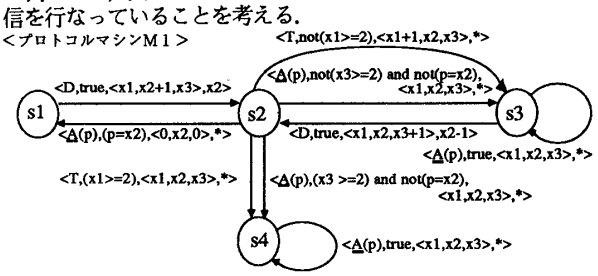
本稿では，プロトコルマシンのモデルを1つ定め，そのモデル上で与えられた通信プロトコルがエラーリカバリ性（異常な状況からいつかは正常な状況に復帰すること）を持つことを機械的に証明するための1つの方法を提案する．さらに，その方法をHDLC手順に適用した結果や証明手続きの高速化の方法についても述べる．

2 プロトコルマシンモデル

プロトコルマシンを「有限制御部」および整数値を保持する有限個の「レジスタ」を持つMealy型のオートマトンとしてモデル化する．オートマトンの「入力記号」は，「相手局からのフレーム受信」，「タイマーからの割り込み」等プロトコルマシンの動作に対応付ける．入力記号は「パラメタ」を持ってもよい．パラメタは1個の整数とし，通信相手局から受信するメッセージに対応付ける．「出力」は1個の整数とし，相手局へ送信するメッセージに対応付ける．「遷移条件」はレジスタ値や入力パラメタ値を表す変数の1次結合（以下，P項と呼ぶ），<，=，and，or，not から成る論理式（以下，P文と呼ぶ）で指定する．「入力記号」は同じでも「遷移条件」によって異なった遷移をしてもよい．遷移によるレジスタ値の変化はP項で指定する．

例として，次のようなプロトコルマシン M_1，M_2 が互いに通信を行なっていることを考える．

<プロトコルマシンM1>



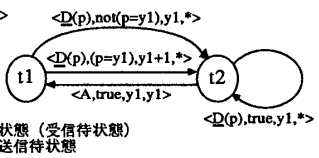
状態　s1:初期状態（送信待状態）
s2:Ack-受信待状態
s3:再送待状態
s4:オペレータによるリカバリを必要とする状態

レジスタ　x1:タイムアウトの回数（初期値＝0）
x2:新規データの送信回数（初期値＝0）
x3:再送回数（初期値＝0）

入力記号　D:送信
Δ:Ack-受信（p:Δの入力パラメタ）
T:タイムアウト

ラベル　<入力，遷移条件，次のレジスタ値，出力>

<プロトコルマシンM2>



状態　t1:初期状態（受信待状態）
t2:Ack-送信待状態

レジスタ　y1:新規データの受信回数（初期値＝0）

入力記号　D:受信（p:Dの入力パラメタ）
A:Ack-送信
T:タイムアウト

図1　プロトコルマシンM1，M2

D は M_1 の送信動作を，$\underline{D}$ は M_2 の受信動作を表す入力記号である．このとき $[D,\underline{D}]$ は「M_1 から送信されたメッセージを M_2 が受信した」という動作を表し，$[D,*]$ は「M_1 から送信されたメッセージは通信回線上で消失した」という動作を表すものとする．同様に，$[\underline{A},A]$，$[*,A]$，$[T,*]$ という動作が考えられる．

M_1 の状態が s_1，M_2 の状態が t_1 であるとき，$[D,\underline{D}]$ という動作が起これば M_1 の状態は s_2，M_2 の状態は t_2 に各々遷移し，レジスタ x_2 の値が1増加する．また，$p=y_1$（p は $\underline{D}$ のパラメタであり，それは M_1 の出力なので，つまり $x_2=y_1$）のときレジスタ y_1 の値は1増加され，$\mathbf{not}(p=y_1)$（つまり，$\mathbf{not}(x_2=y_1)$）のときレジスタ y_1 の値は変わらない．

また，$[D,*]$ という動作が起これば M_1 は状態 s_2 に遷移するが，M_2 は遷移しない．

3 エラーリカバリ性

プロトコルマシン M_1，M_2 の状態を s,t とし，M_1 のレジスタの内容を p_1,p_2,p_3，M_2 のレジスタの内容を q_1 とする．このとき，これらの組 $\langle s,t,\langle p_1,p_2,p_3\rangle,\langle q_1\rangle\rangle$ を M_1，M_2 の全状態と呼ぶ．

2つのプロトコルマシン間の通信において，正常な送受信を行なっているときには真となり，回線に障害等が発生しているときには偽となるような関係Φを検証者が指定する．Φは全状態を表す変数を引数とするP文で指定する．

M_1,M_2 の初期状態と初期レジスタ値に相当する全状態（初期全状態）から，動作を行なうことによって，到達可能な任意の全状態に対し，それ以降どのような動作を行なっても，いつかはΦを真とする全状態に到達できるとき，M_1,M_2 はΦに対するエラーリカバリ性を持つという．

4 エラーリカバ性の検証法

M_1,M_2 の初期全状態に対して，Φの値が真であるとする．このとき，Φを真とする任意の全状態からどのような動作を行なっても，いつかは再びΦを真とする全状態に遷移することが保証されれば，M_1,M_2 はΦに対するエラーリカバリ性を持つといえる．そこで図2の様なグラフ（有向木）を考える．

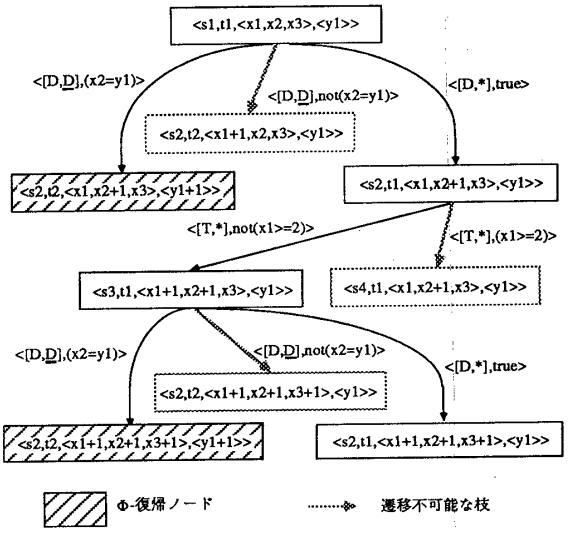


図2　状態とレジスタ値の変化を表すグラフ（有向木）

図2の有向木の根ノードには M_1,M_2 の状態と，M_1,M_2 のレジスタ値を表す変数の組が記されている．各枝には動作とその動作が実行されるための遷移条件が記されている．根ノードから各ノードに至る枝に付加された動作を実行したときに，M_1,M_2 の状態とレジスタ値がどの様に変化するかが各ノードに記されている．

以下では次の1～6の様な手続きによって図2の様な有向木を作成し，その有向木を用いてエラーリカバリ性の検証を行う．

Mechanical Proof of Error Recoverability for Communication Protocols
R.Kato[1], T.Higashino[1], K.Taniguchi[1], M.Mori[2] and K.Ninomiya[3]
Osaka Univ.[1], Shiga Univ.[2], Daikin Ltd.[3]

1. Φを指定する．例えば，
$$\Phi(u,v,x_1,x_2,x_3,y_1) \equiv \{(x_2=y_1)\mathbf{or}(u=s_4)\}\mathbf{and}(x_1\geq 0)\mathbf{and}(x_3\geq 0)$$
と指定することにする．M_1,M_2の初期状態（s,tとする），およびレジスタ値を表す変数（$\mathbf{x},\mathbf{y}$とする）の組$\langle s,t,\mathbf{x},\mathbf{y}\rangle$を根ノードとする．

2. 根ノードを「現在のノード」とする．

3. 「現在のノード」$\langle s',t',\mathbf{x}',\mathbf{y}'\rangle$（$\mathbf{x}',\mathbf{y}'$は$\mathbf{x},\mathbf{y}$を変数とするP項の組）より可能なすべての遷移についてその子を作る．
但し，「現在のノード」から出る枝に付加された遷移条件を$C(\mathbf{x}',\mathbf{y}')$とする．
$$\forall\mathbf{x},\mathbf{y}[\Phi(s,t,\mathbf{x},\mathbf{y})\supset \mathbf{not}C(\mathbf{x}',\mathbf{y}')] \quad --(1)$$
が真のとき，この枝を「遷移不可能な枝」と呼び，その遷移による子は作らない．
例えば，根ノードから出る枝の中で$([D,\underline{D}],\mathbf{not}(x_2=y_1))$なるラベルを持つ枝を考える．この枝に対して(1)の値は真となるので，この枝は有向木に付加しない．

4. 「Φ復帰ノード」とは，次式が真となるノード$\langle s',t',\mathbf{x}',\mathbf{y}'\rangle$である．
$$\forall\mathbf{x},\mathbf{y}[\Phi(s,t,\mathbf{x},\mathbf{y})\supset\Phi(s',t',\mathbf{x}',\mathbf{y}')] \quad --(2)$$
例えば，図2の$\langle s_2,\langle t_2,x_1,x_2+1,x_3\rangle,\langle y_1+1\rangle\rangle$のノードに対して，
$$(2)\equiv\forall x_1,x_2,x_3,y_1[(x_2=y_1)\mathbf{and}(x_1\geq 0)\mathbf{and}(x_3\geq 0)\supset(x_2+1=y_1+1)\mathbf{and}(x_1\geq 0)\mathbf{and}(x_3\geq 0)]$$
この式は真なので，$\langle s_2,t_2,\langle x_1,x_2+1,x_3\rangle,\langle y_1+1\rangle\rangle$のノードは$\Phi$復帰ノードである．
有向木の葉にΦ復帰ノードでないノードがあるとき，そのノードを「現在のノード」として3へ行く．有向木の葉がすべてΦ復帰ノードであるとき5へ行く．

5. Φ復帰ノードに現われる状態対のうち，その状態対とレジスタ値を表す変数の組を根ノードとする有向木を作っていないときは，その組を根ノードとして2へ行く．そうでないときは6へ行く．

6. Φに対するエラーリカバリ性を持つと判断する．

(1)や(2)の[]内の式はP文であるから，(1)や(2)の真偽は整数線形計画問題の解の非存在性の判定に帰着して判定できる．よって，上述の検証手続きは計算機を用いて機械的に実行できる．

5 HDLC手順への適用

我々は上述の検証法に基づく証明手続きを作成し，HDLC手順のあるエラーリカバリ性を検証した．我々が記述した一次局，二次局のプロトコルマシンの状態数はともに12個，レジスタ数は各々6個，5個，入力記号は15個，12個であった．また，「一次局の送信状態変数と二次局の受信状態変数の値が一致するとき，あるいは一次局が回線の永久障害を検出したことを表す特別な状態にいるとき」真となるようなP文Φを指定した．HDLC手順には幾つかのパラメタ（同一シーケンス番号のI（情報）フレームの最大再送回数，ウインドウサイズ，タイムアウトの発生回数の上限値）があり，これらの値の差異によって検証時間は大幅に異なる．

Iフレームの最大再送回数	1	2	2	2
ウインドウサイズ	1	1	2	2
タイムアウトの発生回数の上限	1	2	1	2
証明に要したCPU時間（時間）	2	9	55	122
プレスブルガー文判定回数（万回）	4	20	71	104

表1　検証結果

6 検証作業における繰り返し部分の除去

図1のプロトコルマシンM_1の状態s_2でタイムアウト（動作T）が起こると，再送（動作D）が実行される．但し，タイムアウトの回数（レジスタx_1の値）が2回を越えると，状態s_4に遷移する．図1中の遷移条件$(x_1\geq 2)$を$(x_1\geq 100)$に変更すると，状態s_4に遷移するまでに必要な動作の回数が増加する．表1の検証例でパラメタ値を大きくすることによって検証時間が増大するのはこのためである．そこで次の様な方法を考える．

図2のノード$\langle s_2,t_1,\langle x_1,x_2+1,x_3\rangle,\langle y_1\rangle\rangle$とノード$\langle s_2,t_1,\langle x_1+1,x_2+1,x_3+1\rangle,\langle y_1\rangle\rangle$の様に木の祖先，子孫の関係にあるノードで同じ状態対を持つノードがあるとする．このときレジスタ値の違いこそあれ$\langle s_2,t_1\rangle\rightarrow\langle s_3,t_1\rangle\rightarrow\langle s_2,t_1\rangle$なる遷移（ループと呼ぶ）が繰り返し行なわれることが予想されるので，ループを回る回数をパラメタ化する．

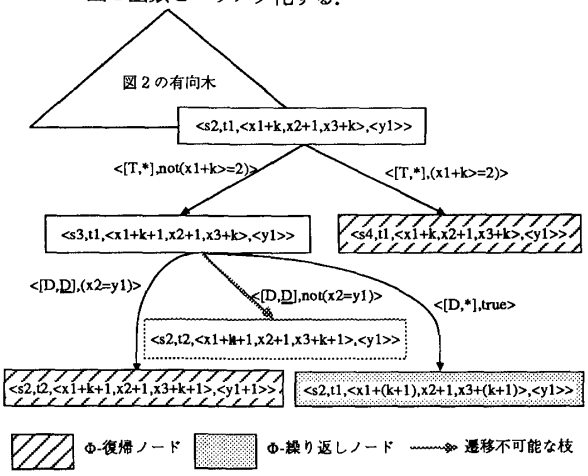


図3　パラメタ付有向木

すなわち，ノード$\langle s_2,t_1,\langle x_1+1,x_2+1,x_3+1\rangle,\langle y_1\rangle\rangle$を，図3の一番上のノードの様にループを回る回数$k$を導入して$\langle s_2,t_1,\langle x_1+k,x_2+1,x_3+k\rangle,\langle y_1\rangle\rangle$の様に表す．このノードの子孫のノードや遷移条件は根ノードのレジスタ値を表す変数とパラメタkを用いて記述する．本稿で述べるパラメタ化の方法は1回の遷移で各レジスタの値が定数値しか変化しない場合にのみ適応し，各ノードのレジスタ値がP項で記述できない場合（例えば，レジスタ値が$k\times x_1$である場合）は，適応しない．このとき，図3の葉ノード$\langle s_2,t_1,\langle x_1+(k+1),x_2+1,x_3+(k+1)\rangle,\langle y_1\rangle\rangle$の様にループを一巡したことに相当するノードに到達したとき，このノードを「Φ繰り返しノード」と呼ぶ．Φ繰り返しノードに対しては子孫を作らない．この方法を用いて検証を行なう場合，ループの遷移を無限に繰り返してΦに対するエラーリカバリ性を持たない場合が起こり得る．そこで図3の$\langle s_2,t_1,\langle x_1+k,x_2+1,x_3+k\rangle,\langle y_1\rangle\rangle$から$\langle s_2,t_1,\langle x_1+(k+1),x_2+1,x_3+(k+1)\rangle,\langle y_1\rangle\rangle$へ至る枝のいずれか，例えば$([T,*],\mathbf{not}(x_1+k\geq 2))$に対して，ループを何回か回ればその枝の遷移が不可能になること，すなわち，
$$\exists k\forall\mathbf{x},\mathbf{y}[\Phi(s_1,t_1,\mathbf{x},\mathbf{y})\supset\mathbf{not}\{\mathbf{not}(x_1+k\geq 2)\}] \quad --(3)$$
が真であることを証明する．(3)はプレスブルガー文であるのでその真偽は判定可能であるが，判定にはコストを費やすので，十分条件として検証者がループを回る回数の上限m（例えば$m=2$）を与え，
$$\forall\mathbf{x},\mathbf{y},k[\Phi(s_1,t_1,\mathbf{x},\mathbf{y})\wedge(k\geq m)\supset\mathbf{not}\{\mathbf{not}(x_1+k\geq 2)\}] \quad --(4)$$
が真であることを証明する．(4)が真ならばループを永久に回ることはない．

また，ノード$\langle s_2,t_1,\langle x_1+k,x_2+1,x_3+k\rangle,\langle y_1\rangle\rangle$の子孫はループを高々$m$回しか回らないので，4の検証手続き3の条件(1)の判定は，
$$\forall\mathbf{x},\mathbf{y},k[\Phi(s,t,\mathbf{x},\mathbf{y})\wedge(1\leq k\leq m)\supset\mathbf{not}C(\mathbf{x}',\mathbf{y}',k)]$$
で判定する．同様に，検証手続き4の条件(2)の判定は，
$$\forall\mathbf{x},\mathbf{y},k[\Phi(s,t,\mathbf{x},\mathbf{y})\wedge(1\leq k\leq m)\supset\Phi(s',t',\mathbf{x}',\mathbf{y}',k)]$$
で判定する．

一般にループが複数個存在する場合も，各ループを回る回数をパラメタ化することによって，同様に検証時間の短縮が計れる（詳細については紙面の都合上省略する）．

7 あとがき

現在，6で述べたパラメタ化の方法に基づく検証手続きを作成中である．この手続きを用いて大きなパラメタ値に対する検証時間がどの程度短縮できるか等の評価が今後の課題である．

参考文献

1. 二宮，東野，谷口，木本："プロトコルマシンの等価性証明の一方法"，信学論(D)，j71-D,12,pp.2630-2639(昭63-12).

2. 木本，東野，谷口，森，二宮："通信プロトコルにおけるエラーリカバリ性の検証の一方式"，信学技法 IN88-80(昭63-09).