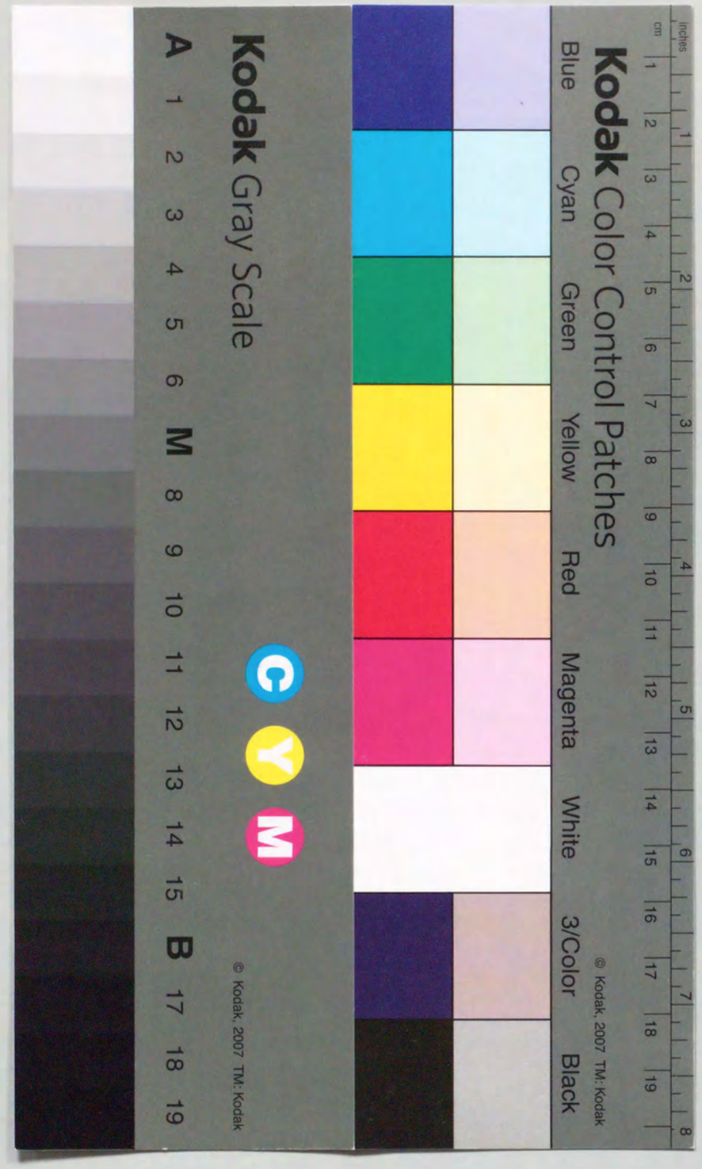


通信プロトコルの等価性及びエラーリカバリ性の検証とプログラムへの変換に関する研究

平成　2年　8月

二宮　清

通信プロトコルの等価性及び
エラーリカバリ性の検証と
プログラムへの変換に関する研究

平成 2年 8月

二宮 清

内容梗概

本論文は，筆者がダイキン工業株式会社，情報システム部に在職中に通信処理関係の研究の機会を得て，大阪大学基礎工学部情報工学科，嵩研究室と谷口研究室において行った通信処理に関する研究のうち，プロトコルマシンの等価性及びエラーリカバリー性の検証と通信プロトコルの代数的仕様からプログラムへの変換に関する研究を３章にまとめたものである．

緒論及び各章の第１節では，研究の現状，その工学上の意義，本研究の新しい成果について概説している．又，各章の最後の節及び全体の結論では，本研究で得られた主な結果と，今後に残された問題点について述べている．

第１章では，プロトコルマシンを，任意の整数値を保持する有限個のレジスタを持ち，演算として整数の加減算，等号・不等号判定，ＡＮＤ，ＯＲ，ＮＯＴを用いるオートマトンとしてモデル化し，その等価性を定義している．一般にこのモデルの等価性は判定不能である．本論文では二つのプロトコルマシンM1,M2の状態やレジスタ値の間の関係を表わす述語Ψを検証者が指定し，「M1,M2が常にΨをみたし，かつΨをみたす状態やレジスタ値に対しては次の入力に対する出力が等しいか」ということを調べる方法を提案する．第１章の前半では，述語Ψのクラスを定め，それらが成り立つための十分条件を与え，後半では，その判定手続きをプログラム化して，ＳＤＬＣ手順を実現する二つのマシン例に対して適用し，等価性の証明において本論文の手法が有効であることを確かめた．

第２章では，第１章でモデル化した２つのプロトコルマシンが，エラーリカバリ性を持つことを保証するための自動検証法について述べている．一般に，プロトコルマシン間で通信を行っている際に，マシンダウンや回線障害等が発生して異常な状況に陥ってもいつかは正常な状況に復帰することをエラーリカバリ性と呼ぶが，本論文ではプロトコルマシン対がエラーリカバリ性を持つ事を整数線形計画問題の解の非存在性の証明に帰着して機械的に証明する為の方法を提案した．又，その証明手続きをプログラム化し，それを用いてハイレベル手順を実現するプロトコルマシン対のエラーリカバリ性を検証することによりその手法の有効性を示した．

第３章では，通信プロトコルの代数的仕様から，手続き型プログラムへの変換について述べている．まず，変換法の正しさに関する議論が厳密に行えるよう，変換法を形式的に記述した．また，この変換法に従ってプロトコルの代数的仕様をＣ言語プログラムに変換するコンパイラを作成した．本コンパイラでは，レジスタ値の待避の個数が少なくなるような代入文の実行順を選択する等の最適化も行っている．OSIセションプロトコルの代数的仕様を入力例として生成されたプログラムを動作させた結果，人手で作成されたプログラムと同等の効率をもつプログラムを生成できることが確認された．

通信プロトコルの等価性及びエラーリカバリ性の
検証とプログラムへの変換に関する研究

目次

緒論

近年のデータ通信の発展に伴ない，情報の送受信の手順を規定する通信プロトコル（以下単にプロトコルと呼ぶ）の厳密かつ形式的な記述やその正しさの検証，さらには，プロトコルに従って動作する通信プログラムの開発を容易にし，その開発コストを減少させる為のプロトコル記述法やプログラム設計法等が重要な課題になってきており，様々な研究が行われている．(1)

一般に，プロトコルは選択性のあるいくつかの機能群から構成されている場合が多く，実際のプロトコルマシン（または単にマシンと呼ぶ）の開発に際しては，それぞれのマシンに必要な機能のみがインプリメントされる．又，タイマの設定値や再送回数の上限等はパラメータ化されていることが多く，各マシンの設計者がマシンの開発時に具体的にその値を設定する．このため，同じプロトコルに基づいて開発されたマシンの組であっても，等価性（同一の入力系列（マシンの遷移要因の系列）に対して，出力系列が同じであること）が保証されないことが多い．このため，一つのマシンを別のマシンに更新する（古いマシンを新しいマシンに置換えたり，あるメーカーのマシンを，別のメーカーのマシンに置換える）と，それまで通信できた相手と通信できなくなる場合がある．マシンの等価性が保証されれば更新が可能となる．しかし，従来，マシンの等価性を証明する有効な方法がなく，テスト入力による検査（幾つかのテスト系列を入力して，同一の出力系列が得られるかどうかをチェックする）が行われてきたが(1)，このようなテストでは必ずしも等価性が

保証されない，又，検査にはかなりの労力を要する．

第１章では２つのプロトコルマシンの等価性証明の一方法について述べる．まず，各マシンを，「有限制御部」と任意の非負整数値を保もする有限個の「レジスタ」を持ち、演算として整数の加減算，等号・不等号判定，ＡＮＤ，ＯＲ，ＮＯＴを用いるオートマトンとしてモデル化し，その等価性を定義している．一般的にデータリンク層のプロトコルマシンの多くが，本論文で提案するオートマトンとしてモデル化できる．また，このモデルを用いていわゆる２－計数機械の動作を模倣することができる．２－計数機械はチューリング機械と原理的な能力が同じであること，及びチューリング機械の等価性が判定不能であることが知られているので，このモデルにおける等価性も一般に判定不能である．このため，２つのマシンM1,M2 の状態やレジスタ値の間で動作途中において常に成り立つと思われる関係Ψを検証者が指定し，

(1) 任意の入力系列に対してM1とM2が到達する状態とその時のレジスタ値の組が関係Ψを満足すること，及び(2) 関係Ψを満たすどのような状態とレジスタ値についても，任意の一つの入力に対してM1とM2の出力が同一であることを，計算機を用いて証明するという方法を採用する．本論文では，Ψのクラスをプレスブルガー文の部分クラスに制限し，M1,M2,Ψが(1),(2) を満たすための一つの十分条件を導き出した．その十分条件が成り立つか否かの判定問題を整数線形計画問題の解の非存在性の証明に帰着して機械的に証明するための方法について述べる．又，十分条件の判定手続きをプログラム化して，SDLC手順の２次局に相当する２つのマシンの等価性を証明した結果についても述べる．

さて，一般にプロトコルは，正常な状況（正常に送受信を行っているような状況）における各プロトコルマシン（通信制御装置）の動作を記述した部分と，異常な状況（回線に障害等が発生したような状況）からどのように正常な状況に復帰するかを記述した部分，に分類されることが多い．この際，「異常な状況からいつかは正常な状況に復帰すること」（エラーリカバリ性をもつという）を機械的に証明できれば望ましい．しかし，従来，エラーリカバリ性を証明する有効な方法がなく，テスト手法による検査が行われてきたが，このようなテスト手法では必ずしもエラーリカバリ性を論理的に保証できない．

第２章では，第１章でモデル化した２つのプロトコルマシンがエラーリカバリ性をもつことを保証するための自動検証法について述べる．第１章と同様に各プロトコルマシンは，「有限制御部」と任意の整数値を保持する有限個の「レジスタ」を持ち，演算として整数の加減，等号・不等号判定，ＡＮＤ，ＯＲ，ＮＯＴを用いるオートマトンとして記述する．このモデルで記述された２つのプロトコルマシンM1，M2の状態とレジスタ値の組（以下，M1，M2対の全状態，または単に全状態と呼ぶ）を引数とする述語Φを線形不等式の論理結合の形で指定する．述語Φの値は，その値が真の時は「正常な状況」を表すように指定し，「M1，M2の初期状態と初期レジスタ値の組から，M1とM2が（通信回線の誤り等も考慮して）送受信等の動作を行うことによって遷移する任意の全状態に対して，たとえその全状態がΦを満足しない全状態であっても，それ以降可能などのような動作を続けてもいつかは再びΦを満足する全状態に遷移する」時，M1，M2はΦに対するエ

ラーリカバリ性をもつと定義する.

提案するモデルのマシン及びΦのクラスにおいては, M1, M2がΦに対するエラーリカバリ性をもつ場合でも, そのことを証明するアルゴリズムが存在しない. そこで, 必ずしも証明に成功する保証はないが, 「Φを満足する全状態からどのような動作を行っても再びΦを満足する全状態へ遷移すること」を保証するための一つの手続きを与え, その手続きを再帰的に用いることによって, M1, M2がΦに対するエラーリカバリ性をもつことを証明する. また, 検証の自動化のため, 与えられたプロトコルマシン対が与えられたΦに対するエラーリカバリ性をもつことを, 第1章と同様に整数線形計画問題の解の非存在性の証明に帰着して機械的に証明するための方法を提案する. 尚, 一般にこのような検証法では, HDLCにおけるウィンドサイズや最大再送回数等のようなプロトコルの「パラメータ値」が増大すると, 検証に要する時間が急激に増大する. 2章の後半では, 本検証法を改良して, このようなパラメータ値に依存しない計算時間で検証を行う手法を述べる. また, 提案する証明手続きを用いてHDLC手順に従うあるプロトコルマシン対のエラーリカバリ性を検証することにより, 本手法の有効性を示す.

さて, 与えられたプロトコル仕様から, それに従って動作するプログラムを導出する手法を検討することは, 通信ソフトウェアの設計・開発の際の最も基本的で重要な課題である. 与えられたプロトコルを通信プログラムに機械的に変換できるためには, プロトコル自身が, 形式的意味の定義された言語で記述されていることが必要である. ＩＳＯではプロトコル仕様の形

式的記述言語として, LOTOS, Estelleを採用しており[30][31], これらの言語で記述されたプロトコル仕様の解釈実行法や, 通信プログラムへの変換法に関する研究がなされている[32][33][34][35]. LOTOSでは, 仕様の厳密な意味は, 通信系の取り得る動作系列の集合として定義されている. そのため, ＯＳＩ規格のように, 状態遷移モデルに基づいて記述された仕様に対応するレベルで, 自然に記述するのは困難である. 同様に, LOTOS仕様と順序機械的に動作する通信プログラムとの間には, 記述レベルにかなりの隔たりがあり, プログラムへの変換法の正しさを厳密に証明するのは容易ではないと思われる. 一方, Estelleでは, 拡張状態遷移モデルに基づいて仕様の意味が定義されているので, 比較的容易に効率の良いプログラムに変換できるという利点をもつ. 反面, 上位層に提供する通信サービスを抽象的なレベルで記述することが困難である.

第3章では, 抽象的順序機械と呼ばれる形で記述された通信プロトコルの代数的仕様から, 手続き型プログラムへの変換法について述べる. 代数的仕様記述法は, 厳密な意味の定義が簡明であり, 検証の際に必要な仕様の無矛盾性や詳細化等の概念も, 合同関係という概念のみを用いて簡明に定義できる. 代数的仕様の部分クラスとして, 抽象的順序機械の形で記述された仕様[18][19]がある. この仕様では, 論理型, 整数型等の基本タイプを前提として, 状態の構造を陽にはもたない抽象的状態を表すタイプを定義する. 関数には状態遷移関数, 状態成分関数（抽象的状態からその成分の値を取り出す関数）, および補助関数があり, 各状態遷移後の状態成分関数の値を公理によって定義する. 抽象的状態は構造を陽にはもたないことから, 新

たに状態成分を追加する際，既に記述した部分を変更することなく，状態成分関数に関する構文と公理を局所的に追加するだけでよい，という特長がある．また，既にHDLCやOSIセション層等の通信プロトコルを抽象的順序機械の形で記述し，仕様の形式的検証が行われ，本記述法の有効性が実証されている．

仕様からプログラムへの変換法の正しさを厳密に議論するためには，変換法を形式的に記述しておくことが望ましい．第３章前半では，目的プログラムを解釈実行する機械を定義する代数的仕様，および，変換法自身を定義する代数的仕様を記述することにより，変換法を形式的に定義している．また，第３章後半では，この変換法に従って通信プロトコルの代数的仕様をＣ言語プログラムに変換するコンパイラについて述べる．本コンパイラでは，レジスタ値の待避の個数が少なくなるような代入文の実行順を選択する等の最適化も行っている．OSIセションプロトコルの代数的仕様を本コンパイラを用いてプログラムに変換し，いくつかのテスト系列についてもそのプログラムを動作させた結果について報告されている[29]．そこでは，人手で作成したプログラムと同程度の効率をもつプログラムが生成できることが確認されている．

（関連発表論文）

学術論文誌

(1) 二宮清，東野輝夫，谷口健一，木本智久:"プロトコルマシンの等価性証明の一方法",電子情報通信学会論文誌(D)，Vol.J71-D,No.12，pp.2630-2639 (昭63-12).
(2) 遠一光，粟屋英司，関浩之，藤井護，二宮清:"抽象的順序機械の形で記述された代数的仕様からプログラムへの変換について",電子情報通信学会論文誌(D-I)，Vol.J73-D-I，No.2，1-3,5,6節，pp.201-206,208-210 (平2-02).
(3) 木本智久，二宮清，東野輝夫，谷口健一,森将豪:"通信プロトコルにおけるエラーリカバリ性の自動検証の一方式"，電子情報通信学会論文誌(D-I)，Vol.J73-D-1，No.5，pp.500-509 (平2-05).

国際会議発表

(1) T. Higashino, K. Ninomiya, T. Kimoto, K.Taniguchi and M.Mori: "Automated Verification of Equivalence of Protocol Machines", Protocol Specification, Testing, and Verification, IX, pp.235-246, North-Holland (1990).

第 1 章

プロトコルマシンの等価性検証の一方法

1.1 序言

一般に，プロトコル（通信規約）は選択性のあるいくつかの機能群から構成されている場合が多く，実際のプロトコルマシン（または単にマシンと呼ぶ）の開発に際しては，それぞれのマシンに必要な機能のみがインプリメントされる．又，タイマの設定値や再送回数の上限等はパラメータ化されていることが多く，各マシンの設計者がマシンの開発時に具体的にその値を設定する．このため，同じプロトコルに基づいて開発されたマシンの組であっても，等価性（同一の入力系列（マシンの遷移要因の系列）に対して，出力系列が同じであること）が保証されないことが多い．このため，一つのマシンを別のマシンに更新する（古いマシンを新しいマシンに更新したり，あるメーカーのマシンを，別のメーカーのマシンに更新する）と，それまで通信できた相手と通信できなくなる場合がある．マシンの等価性が保証されれば更新が可能となる．しかし，従来，マシンの等価性を証明する有効な方法がなく，テスト入力による検査（幾つかのテスト系列を入力して，同一の出力系列が得られるかどうかをチェックする）が行われてきたが[1]，このようなテストでは必ずしも等価性が保証されない．又，検査にはかなりの労力を要する．

第１章では，マシンの等価性を証明するための一つの方法を提案する．まず，筆者らは，プロトコルマシンを，整数値を保持する有限個のレジスタ（送受信のためのデータやシーケンス番号等は整数化して取り扱う）を持つオートマトンとしてモデル化した．1.2 ではそのモデルについて述べ，1.3 ではこのモデルにお

けるマシンの等価性を定義している．このモデルは，特定の階層のプロトコルマシンを対象としたものではないが，例えば，データリンク層[1] のプロトコルマシンの多くが第１章で提案するオートマトンとしてモデル化できる．

このモデルを用いて，いわゆる２－計数機械（two-counter machine）の動作を模倣することができる．２－計数機械はチューリング機械と等価であること，及び，チューリング機械の等価性が判定不能であることが知られているので[2]，上述のモデルにおける等価性も一般には判定不能である．そこで，等価であるための次のような十分条件を考える．今，Ψを２つのマシンM1,M2 の状態やレジスタ値の間の関係を表す述語とする．この時，次の(1),(2) が共に成り立てば，M1とM2は等価である．

(1) 任意の入力系列に対して，M1とM2が到達する状態とその時のレジスタ値の組が関係Ψを満足する．
(2) 関係Ψを満たすどのような状態とレジスタ値についても，任意の一つの入力に対してM1とM2の出力が同一である．

第１章では，検証者（等価性の証明を行う者）が関係Ψを指定（予測）し，計算機を用いて(1)，(2)を証明することによって等価性を証明する方法を採用する．まず，関係Ψは1.2 で述べるＰ文（状態とレジスタ値を表す変数の線形不等式，またはそれらの不等式をand,or,notで結合した論理式．存在作用素∃を持たないプレスブルガー文に相当）で指定されるものとし，1.4ではM1,M2 及びＰ文Ψが(1),(2) を満たすための一つの十分条件を与え，M1,M2,Ψがその十分条件を満足する時，且つその時のみ，M1とM2はΨ等価であると定義す

る．定義よりM1とM2がΨ等価ならM1とM2は等価である．次に，M1とM2がΨ等価であるか否かの判定問題を整数線形計画問題の解の非存在性の判定問題に帰着して判定する方法について述べる．1.5では実際にワークステーション(IBM RTPC)上で作成したΨ等価性の判定手続き[3]を用いてSDLC手順の２次局[8]に相当する２つのマシンの等価性を証明した結果について述べる[4]．

提案する方法を用いて等価性を証明する場合，検証者自身が関係ΨをＰ文で指定（予測）しなければならない．しかし，Ψを指定すればΨ等価であるか否かはΨ等価性の判定手続きを用いて機械的に判定できるという特徴を持つ．尚，提案する方法は等価性を証明するための一つの十分条件であり，一般には２つのマシンM1,M2 が等価であってもΨ等価となるようなＰ文Ψが必ず存在するとは限らないし，そのようなΨが存在してもそのΨを検証者が指定できるとは限らない．また，Ψ等価性の判定手続きは多項式時間算法ではない．しかし，筆者らが記述した幾つかのマシンの組に対しては上述の方法で等価性の証明が実際に行えたので，第１章で提案する方法は実用のマシンの等価性の証明に幅広く適用出来ると考えられる．

１．２　対象とするプロトコルマシンのモデル化

１．２．１　プレスブルガー算術

第１章では，プロトコルマシンのモデル化や等価性の判定にプレスブルガー算術[5] を用いる．以下，第１章で用いる用語や記法について簡単に述べる．

Ｎを整数の集合とし，Ｎの変数(x,y,z,…)，Ｎの定数(0,±1,±2,…)及び加減算の演算子(＋,－）から成

る項（Ｎの変数の１次結合(線形結合)$^+$ ）を「Ｐ項」と呼ぶ．又，「Ｐ文」を次のように定義する．

(1) t_1, t_2がＰ項の時，$t_1 = t_2, t_1 < t_2, t_1 \leqq t_2, t_1 \geqq t_2, t_1 > t_2$はＰ文である．

(2) A,BがＰ文ならば，(A)or(B),(A)and(B),(A)⊃(B),not(A)は各々Ｐ文である（⊃は，含意を表す）．

(3) 上の(1),(2) を有限回適用して得られるもののみがＰ文である．

変数を含まないＰ文に対しては，演算子（＋,－,=,<,≦,≧,>,or,and,⊃,not）の通常の意味に従って，その真偽を定める（例えば，Ｐ文「(3＋5≦10)or(7=2)」の値は真）．

さらに，Ｐ文A に含まれる全ての変数x1,…,xnを全称記号∀で束縛した論理式を$\forall x_1, \cdots, x_n(A)$ で表す．論理式$\forall x_1, \cdots, x_n(A)$ において，Ｐ文A の変数$x_1, \cdots, x_n$にどのような整数を代入しても，代入した各Ｐ文が真である時，且つその時のみ，論理式$\forall x_1, \cdots, x_n(A)$は真であるという．尚，論理式$\forall x_1, \cdots, x_n(A)$ の真偽は，整数線形計画問題の解の非存在性の判定手続きを用いて判定できる[6]．

１．２．２　プロトコルマシンのモデル化

第１章では，プロトコルマシンを，「有限制御部」及び有限個の整数値を保持する「レジスタ」から成るオートマトンとしてモデル化する．「相手局からのフレームの受信」や「タイマーからの割り込み」，「相手局へのデータの送信」，あるいは，「相手局へのデータの再送」$^{++}$などを，各々，このオートマトンの「入力記号」に対応付ける．入力記号の種類は有限個とするが，各入力記号は有限個の整数値を「パラメータ」

として持ってもよいとする（以下では，入力記号とパラメータ値の組を「入力」と呼ぶ）．例えば，相手局からフレームを受信する場合，フレームの受信という事象が入力記号であり，そのフレームに含まれる受信データやシーケンス番号がパラメータ値に相当する．また，相手局への送信フレームや上位レベルへの受信通知，タイマーへのセット／リセット信号等をこのオートマトンからの「出力」に対応付ける．出力は，整数のｄ字組とし（ｄはマシンで決まる定数），ｄ字組の各要素は，通信相手局，上位レベル，タイマー等へ受け渡される内容である（入力パラメータ値同様，出力も整数化して取り扱う）．

オートマトンは，現在の状態（以下，状態は有限制御部の状態を表す），レジスタ値及び入力（入力記号とパラメータ値の組）の組に対して，次の状態，次のレジスタ値及び出力が定義されるいわゆる Mealy型のオートマトンとして定義する．また，すべての状態，レジスタ値，入力の組に対して，次の状態，次のレジスタ値，出力が「完全」に指定される必要はなく，特定の状態，レジスタ値，入力の組に対して，次の状態，次のレジスタ値，出力が指定されないいわゆる「不完全」に指定されたオートマトンでもよい．以下，プロトコルマシンＭを形式的に次の７字組として定義する．

+ 変数を含まない特別な場合も含む．

++第１章ではマシンの状態を変化させるような各要因を，各々「入力記号」と見なしている．従って，送信や再送のようなマシン内部からの自発的な動作も「入力記号」とみなす．

［プロトコルマシンＭ＝<Ｓ，Ｒ，S_{R1}，Ｉ，σ，δ，ρ>］

Ｓ＝｛s_1，…，s_k｝：有限個の状態集合．

但し，各状態s_1，…，s_kに適当な非負整数を割り当てる．

Ｒ＝＜Ｒ１，…，Ｒｎ＞：レジスタ名のｎ字組（ｎはマシンＭのレジスタの数）．

各レジスタＲｉは整数値を保持する．

S_{R1}＝＜s_1，r_{11}，…，r_{1n}＞∈Ｓ×N^n：

マシンＭの初期状態s_1とレジスタＲ１，…，Ｒｎの初期値（整数値）r_{11}，…，r_{1n}のｎ+１字組．

Ｉ＝｛α_1，…，α_m｝：有限個の入力記号の集合．

各入力記号α_iは（α_iで決まる）有限個のパラメータを保持してもよい．パラメータ値は整数とし，入力記号α_iのパラメータ値がq_{i1}，…，q_{ij}の時，入力を$\alpha_i(q_{i1},\cdots,q_{ij})$で表す．また，$\alpha_i$のパラメータ数を$|\alpha_i|$で表す．

Ｉｐ＝｛$\alpha_i(q_{i1},\cdots,q_{ij})$｜

α_i∈Ｉ且つq_{i1}，…，q_{ij}∈Ｎ且つ$|\alpha_i|=j$｝

とおく．Ｉｐは入力（入力記号とパラメータ値の組）の集合である．

σ：Ｓ×N^n×Ｉｐ→Ｓ，次の状態を定める関数．

現在の状態ｓ及びレジスタ値r_1，…，r_n及び入力$\alpha_i(q_{i1},\cdots,q_{ij})$から次の状態を定める．

δ：Ｓ×N^n×Ｉｐ→N^n，次のレジスタ値を定める関数．

現在の状態ｓ及びレジスタ値r_1，…，r_n及び入力$\alpha_i(q_{i1},\cdots,q_{ij})$からレジスタＲ１，…，Ｒｎの次の値（次のレジスタ値）のｎ字組を定める．

ρ：Ｓ×N^n×Ｉｐ→N^d，出力を定める関数（ｄはマシンＭの出力先の数）．

現在の状態ｓ及びレジスタ値r_1，…，r_n及び入力

$\alpha_i(q_{i1},\cdots,q_{ij})$ から出力（整数のd字組）を定める. ■

次に, σ, δ, ρの記述法, ならびに第1章で対象とするプロトコルマシンのクラスについて, 図1.1の例を用いて説明する.

[例1　プロトコルマシンM1]

図1.1のプロトコルマシンM1は, BSC手順[7]の送信局を簡単化したものである. M1の4つの状態s_1, s_2, s_3, s_4 は, 各々初期状態（ack受信状態）, 送信状態（ack/nak待ち状態）, 再送準備状態（nak受信状態）, 最終状態を表しており, R1は送信回数を, R2は送信データを保持するレジスタである（何れも整数値を保持する）. データdの送信をT(d)で, 再送をReで, 相手局からの受信確認をA(a)で表す（dは実際の送信データを整数化したもの. 又, aが1の時ackを, 0の時nakを受信したとみなす）. 出力先は相手局1か所のみとする. M1は初期状態から送信を開始し, 各送信データに対してackを受信すれば次のデータを送信し, nakを受信すれば元のデータを再送する. 10個の送信データすべてに対してackの受信確認を行えば動作を終了する. ■

図1.1において, グラフの頂点は, マシンの状態に対応し, グラフの有向辺によって, 各入力に対する次の状態, レジスタ値, 出力を指定する. グラフの各辺には, 4字組のラベル<①入力記号, ②条件式, ③次のレジスタ値, ④出力>が付加されている. 各状態（頂点）で①の入力記号が与えられ, その時点のレジスタ値や入力パラメータ値が②の条件式を満足する時, その辺の指す状態（頂点）に遷移し, ③の指定に従っ

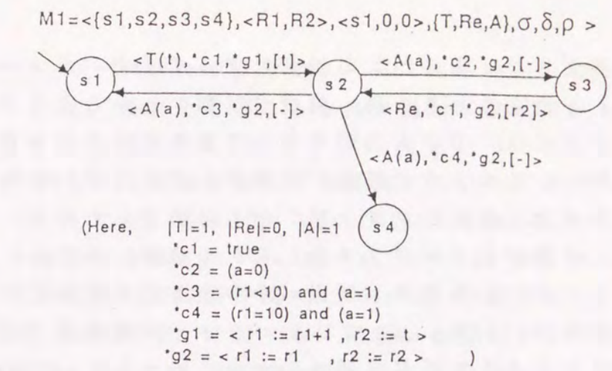


図1.1　プロトコルマシンM1

てレジスタ値が更新され, ④の指定に従って出力を出す. 与えられた入力記号を持つ辺が存在しなかったり, 存在する場合でも, その時点のレジスタ値と入力パラメータ値が②の条件式を満足するような辺が存在しない場合, 次の状態, レジスタ値, 出力は指定されない. 尚, ①の入力記号が同じで②の条件式が異なるような辺が一つの頂点から複数個出てもよいが, どのようなレジスタ値, 入力パラメータ値に対しても, 2個以上の条件式が共に真にならないように各条件式を指定しなければならない.

例えば, 現在の状態がs_1で, レジスタR1, R2の値が各々n_1, n_2 で, 入力がT(d)の場合, 辺$s_1 \rightarrow s_2$に付加されたラベルの①の入力記号がT(t)で（tはTの入力パラメータ値を表す変数）, ②の条件式*c1が恒真（true）であるので, 任意のパラメータ値dに対して次の状態はs_2となる. 又, この遷移において, レジスタR1, R2の値は, ③の*g1の指定に従って, 各々n_1+1,

d に変更され，④の[t] の指定に従って入力パラメータ値d が出力される．尚，頂点（状態）s_1から出る有向辺の中には，①の入力記号がReであるような辺が存在しない．このような場合，状態がs_1で入力がReの時の次の状態，次のレジスタ値，出力は指定されない．

又，状態がs_2でレジスタＲ1,Ｒ2 の値が，各々n_1，n_2で入力がA(k)の場合，頂点s2から出る辺のうち①の入力記号がA(a)（a はA の入力パラメータ値を表す変数）であるような辺が３個（$s_2 \rightarrow s_3$，$s_2 \rightarrow s_1$，$s_2 \rightarrow s_4$）存在する．この場合，②の条件式*c2,*c3,*c4 の変数r_1,r_2,a に値n_1,n_2,k を代入し，代入した論理式が真となるような辺の指す状態（s_3,s_1,s_4の何れか）に遷移する（どの条件式も満足しなければ，次の状態，レジスタ値，出力は指定されない）．

尚，辺$s_2 \rightarrow s_3$のように④の出力として[-] が指定されている場合，何も出力しないことを表し，辺$s_3 \rightarrow s_2$のように④の出力として[r_2]が指定されている場合，その時点のレジスタＲ2 の値を出力することを表す■

以下では，図１.１のグラフをM1の仕様と呼ぶ．図１.１のような仕様記述法（σ，δ，ρの指定法）は，状態遷移図と処理状態遷移図を用いた通常の仕様記述法に対応している．尚，第１章では仕様記述に際して，次のような制約を設けている（これらの制約は，1.4で等価性の十分条件を与える際に利用する）．

(1) 各頂点（状態）から出る辺の数は有限である．
(2) ②の条件式は，マシンの現在のレジスタ値を表す変数の組𝕏n と，入力記号α_i のパラメータ値を表す変数の組ℙijを変数とするＰ文で指定する．
(3) ③の次のレジスタ値及び④の出力は，現在のレジスタ値と入力パラメータ値を表す変数（𝕏n,ℙij）

の一次結合で指定する．又，何も出力しない場合も，便宜上そのことを表す適当な整数が出力されると考える．■

次に「定義域を指定する関数」について述べる．図１．１では，頂点s2から①の入力記号がA(a)であるような辺が３個（$s_2 \rightarrow s_3$,$s_2 \rightarrow s_1$,$s_2 \rightarrow s_4$）出ている．これらの辺の②の条件式*c2,*c3,*c4 の論理和を，状態s2，入力記号A に対する「定義域を指定する関数」と呼ぶ．但し，入力記号Reのように，状態s_2から①の入力記号がReであるような辺がない場合，Reに対する定義域を指定する関数は恒偽（false ）と定義する．以下では，状態s，入力記号α_iに対する定義域を指定する関数を

$\eta\langle s, \alpha_i \rangle$(𝕏n,ℙij)

で表す．上述の制約(1),(2) より，定義域を指定する関数はＰ文となる．

１.３　等価性の定義

等価性を定義するため，1.2.2 で述べた関数σ，δ，ρの定義を拡張し，次のような関数$\hat{\sigma}$,$\hat{\delta}$,$\hat{\rho}$を定める．

[$\hat{\sigma}$，$\hat{\delta}$，$\hat{\rho}$の定義]

(1) $\hat{\sigma}$：$(Ip)^* \rightarrow S$，初期状態と初期レジスタ値に有限長の入力系列ｗを与えた時に到達する状態を指定する関数で，次のように定義する．

Ⓐ初期状態の指定

$\hat{\sigma}(\varepsilon) \triangleq s_1$（ε：空入力系列）

Ⓑ各入力$\alpha_i(v_1,\cdots,v_j)$に対する次の状態の指定

$\hat{\sigma}(w\cdot\alpha_i(v_1,\cdots,v_j))$

≜ $\sigma(\hat{\sigma}(w),\hat{\delta}(w),\alpha_i(v_1,\cdots,v_j))$

（但し，wは任意の入力系列を，$v_1,\cdots,v_j$は，α_iの任意の入力パラメータ値を表す） ■

(2) $\hat{\delta}$：$(Ip)^*\rightarrow N^n$，初期状態と初期レジスタ値に有限長の入力系列wを与えた時に定まるレジスタR1,…,Rnの値のn字組を指定する関数．

(3) $\hat{\rho}$：$(Ip)^*\rightarrow(N^d)^*$，初期状態と初期レジスタ値に有限長の入力系列wを与えた時の出力系列を指定する関数． ■

さらに，2.2で定めた定義域を指定する関数$\eta\langle s,\alpha_i\rangle$の定義を拡張し，次のような補助関数$\hat{\gamma}$を導入する．

［$\hat{\gamma}$の定義］

$\hat{\gamma}$：$(Ip)^*\rightarrow$bool，初期状態と初期レジスタ値に有限長の入力系列wを与えた時に，出力系列$\hat{\rho}(w)$が指定されるか否かを定める関数．すなわち，

$\hat{\gamma}(\varepsilon)$ ≜ true

$\hat{\gamma}(w\cdot\alpha_i(v_1,\cdots,v_j))$

≜ $\hat{\gamma}(w)$ and

$\eta\langle\hat{\sigma}(w),\alpha_i\rangle(\hat{\delta}(w),\langle v_1,\cdots,v_j\rangle)$ ■

以下，整数のd字組b≜$\langle b_1,\cdots,b_d\rangle$，c≜$\langle c_1,\cdots,c_d\rangle$に対して，$b_1=c_1$且つ$b_2=c_2$…且つ$b_d=c_d$が成り立つ時，且つその時のみb＝cと定義する．また，x(≜b·x')，y(≜c·y')，x',y'$\in(N^d)^*$に対して，b＝c且つx'＝y'である時，且つその時のみx＝yと定義し，マシンの等価性を次のように定義する．

［プロトコルマシンの等価性］

入力記号の集合Iが同じで，出力が共に整数のd字組であるような2つのマシン

M1 ＝ ＜S,R,S_{R1},I,σ,δ,ρ＞

M2 ＝ ＜S',R',S_{R1}',I,σ',δ',ρ'＞

に同一の入力系列wを与えた時，wに対する出力系列が共に指定され同一，あるいは，共に指定されない時（すなわち，

$\forall w\in(Ip)^*$

[{$\hat{\gamma}(w)$ and $\hat{\gamma}'(w)$ and $\hat{\rho}(w)=\hat{\rho}'(w)$}

or {not($\hat{\gamma}(w)$) and not($\hat{\gamma}'(w)$)}]

が成り立つ時），且つその時のみ，M1とM2は等価であるという（但し，$\hat{\gamma}$，$\hat{\gamma}$'は，各々M1，M2の出力系列が指定されているか否かを表す関数）． ■

第1章では，互換性の立場より等価性を上述のように定義している．しかし，等価性を「wに対する出力系列が共に指定され同一，あるいは，少なくとも片方のマシンの出力系列が指定されない」と定義した場合等でも，以下の議論を若干変更することによって，それぞれの等価性を議論できる．

1.4 等価性の検証法

一般に，1.2.2で定義した2つのマシンが等価であるか否かは決定不能である(2)．そこで，第1章では検証者が2つのマシンM1,M2の状態とレジスタ値の間で動作途中において常に成り立つと思われる関係Ψを P文で指定し，M1,M2及びP文Ψが序言で述べた2つの条件(1),(2)を満足することを計算機を用いて証明するという方法を採用する．今，u,vを各々M1，M2の

状態を表す変数とし（1.2.2 の定義より，状態を整数に対応付けているので，u,v は整数値を取る変数），$\mathbb{B}n$,$\mathbb{C}h$ を各々２つのマシンM1，M2のレジスタ値を表す変数の組とする．又，$\Psi(u,v,\mathbb{B}n,\mathbb{C}h)$ をu,v,$\mathbb{B}n$,$\mathbb{C}h$ を変数とするＰ文とする．序言で述べた２つの条件(1),(2) に相当することを1.2.2 及び1.3で述べた表記法に従って記述すると，次のようになる．

M1,M2 及びＰ文Ψが次の(1),(2) を満足すれば，M1,M2は等価であることは容易に分かる．

(1) $\forall w \in (Ip)^*$

$[\{\hat{\varphi}(w)$ and $\hat{\varphi}'(w)$ and
$\Psi(\hat{\sigma}(w),\hat{\sigma}'(w),\hat{\rho}(w),\hat{\rho}'(w))\}$
or $\{not(\hat{\varphi}(w))$ and $not(\hat{\varphi}'(w))\}]$

(2) $\hat{\varphi}(w),\hat{\varphi}'(w)$の値が真で$s=\hat{\sigma}(w),s'=\hat{\sigma}'(w)$，であるような入力系列wが存在する任意の状態の組s,s'に対して，

$\forall \mathbb{B}n,\mathbb{C}h,\alpha_i,\mathbb{P}ij$
$[\Psi(s,s',\mathbb{B}n,\mathbb{C}h)$ ⊃
$\{\eta\langle s,\alpha_i\rangle(\mathbb{B}n,\mathbb{P}ij)$ and
$\eta'\langle s',\alpha_i\rangle(\mathbb{C}h,\mathbb{P}ij)$ and
$\rho(s,\mathbb{B}n,\alpha_i(\mathbb{P}ij))=\rho'(s',\mathbb{C}h,\alpha_i(\mathbb{P}ij))\}$
or $\{not(\eta\langle s,\alpha_i\rangle(\mathbb{B}n,\mathbb{P}ij))$ and
$not(\eta'\langle s',\alpha_i\rangle(\mathbb{C}h,\mathbb{P}ij))\}]$ ■

尚，一般には，初期状態の組から<s,s'>なる状態の組に到達する時には取り得ないレジスタ値の組が存在する．(2) はそのようなレジスタ値の組に対しても出

力が等しいことを要求しており，その意味で(1),(2) はマシンが等価であるための十分条件に相当する．

以下では，まずM1,M2 に相当する２つのグラフから同一の入力系列を与えたときに到達する状態の組を表す一つのグラフを構成する．次に，このグラフ上でM1,M2及びＰ文Ψが(1),(2) を満たすための一つの十分条件を与える．さらにその十分条件が成り立つか否かの判定問題が決定可能であることを示す．

最初に，準備として図１．１のようなグラフから次のようなグラフを作成する．

［死状態の導入と遷移の完全指定］

図１．１では，全ての状態，レジスタ値，入力の組に対して遷移が指定されているわけではない．そこで，特別の状態（以下，「死状態」と呼び，"se"で表す）を追加し，図１．１のグラフから図１．２のようなグラフ（以下，拡張グラフと呼ぶ）を作成する．図１．２では，図１．１の各状態s と入力記号α_i の組に対して，定義域を指定する関数$\eta\langle s,\alpha_i\rangle(\mathbb{X}n,\mathbb{P}ij)$ が恒真(true)でない限りs→se なる有向辺を加え，４字組

$\langle \alpha_i(\mathbb{P}ij),not(\eta\langle s,\alpha_i\rangle(\mathbb{X}n,\mathbb{P}ij)),-,-\rangle$

をその辺のラベルとする．但し，第３，４引数の"-"は値を指定しないことを表す．尚，1.2.2 の定義より，$\eta\langle s,\alpha_i\rangle(\mathbb{X}n,\mathbb{P}ij)$ はＰ文ゆえ恒真性は判定可能であり，任意のグラフから拡張グラフを構成できる．■

［例２　プロトコルマシンM2］

図１．３は，M1と等価と思われるマシンM2の拡張グラフである．M2の状態は，s_5,s_6,s_7の３個で，それぞれ，送信可能状態，受信待状態，最終状態を表し，死状態をseで表している．レジスタはＲ3,Ｒ4,Ｒ5 の３個で，Ｒ3 の値が１の時は通常の送信のみを実行し，０の時

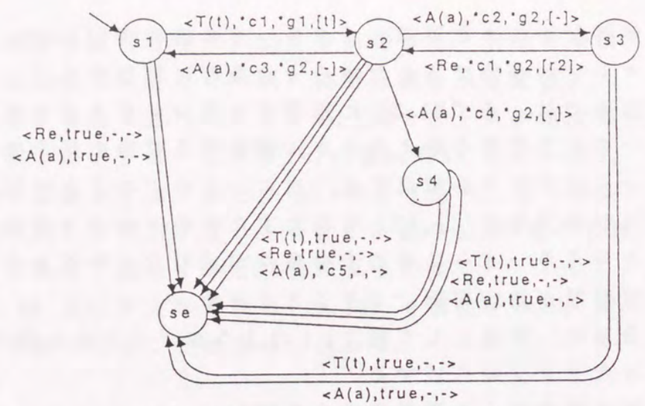


図１．２　プロトコルマシンＭ１の拡張グラフ

は再送のみを実行する．Ｒ４は，送信回数を記憶するレジスタであり，初期値が10で，１回送信するごとに１ずつその値を減らす．Ｒ５は，M1のＲ２同様，再送のためのデータを保持するために用いられる．■

次に，図１．２，１．３のような２つのマシンM1,M2の拡張グラフの組に同じ入力系列を与えた時に到達する状態の組を表すグラフ（「遷移グラフ」と呼ぶ）を作成する．

［遷移グラフ］

まず，M1,M2 の拡張グラフから，次のようなグラフG(V,E)を作成する．頂点集合V はM1，M2の状態の２字組<s,s'>の集合とする．又，M1，M2に各々s→t，s'→t'なる辺が存在し，且つその辺で同じ入力記号に対して,次の状態,レジスタ値,出力が指定されている（２つの辺のラベルの第１引数が同じである）時，且つその

M2 = <{s5,s6,s7},<R3,R4,R5>,<s5,1,10,0>,{T,Re,A}, σ',δ',ρ' >

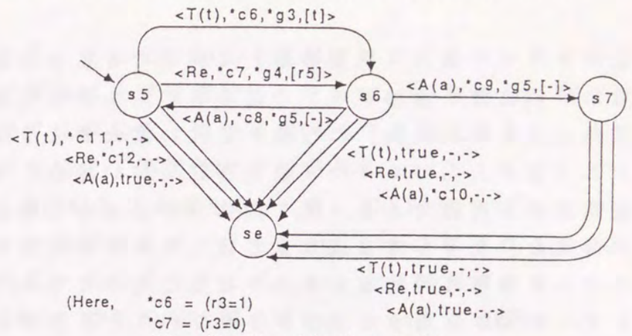


図１．３　プロトコルマシンＭ２の拡張グラフ

時のみ，辺<s,s'>→<t,t'>をE の要素とする．又，辺s→t及びs'→t'のラベルが，各々$<a_1,a_2,a_3,a_4>$，$<a_1,b_2,b_3,b_4>$ である時，$<a_1,\{a_2,b_2\},\{a_3,b_3\},\{a_4,b_4\}>$ を辺<s,s'>→<t,t'>のラベルとする．さらに，上述のような方法で作成したグラフG(V,E)において，初期状態の２字組$<s_1,s'_1>$に相当する頂点からの有向道(path)が存在しない頂点や辺を取り除いたグラフG'(V',E') を作成し，そのグラフをM1,M2 に対する「遷移グラフ」と呼ぶ．例えば，図１．４は，図１．２，１．３で定義したマシンM1，M2に対する遷移グラフである．簡単のため，図１．４では各辺のラベルの第３，４引数は省略している．■

遷移グラフの各頂点は，初期状態から到達する可能性のある状態の組を表している．尚，一般にはグラフ

上のすべての頂点（状態の組）に到達できるとは限らない．例えば，遷移グラフの辺のラベルの第２引数の条件式（上の$\{a_2,b_2\}$に相当）を共に満たすようなレジスタ値や入力パラメータ値が存在しない場合，その辺は遷移不可能である．又，動作途中において検証者の与えたＰ文Ψが常に成り立てば，その条件を利用してさらに遷移不可能な辺を見つけることができる．そこで，M1,M2 に対する遷移グラフG'(V',E') 及びＰ文Ψ(u,v,$\mathbb{B}$n,$\mathbb{C}$h) に対して，次のようなグラフ（「遷移可能グラフ」と呼ぶ）を作成する．

［遷移可能グラフ］

G'(V',E') の各辺<s,s'>→<t,t'>（但し，辺のラベルを$<\alpha_i(\mathbb{P}ij),\{A2(\mathbb{B}n,\mathbb{P}ij),B2(\mathbb{C}h,\mathbb{P}ij)\},\{a_3,b_3\},\{a_4,b_4\}>$ とする）に対して，

∀ $\mathbb{B}$n,$\mathbb{C}$h,$\mathbb{P}$ij

［Ψ(s,s',$\mathbb{B}$n,$\mathbb{C}$h)

⊃ not{A2($\mathbb{B}$n,$\mathbb{P}$ij) and B2($\mathbb{C}$h,$\mathbb{P}$ij)}］

--(*1)

が真（状態がs,s'の時，その時点のレジスタ値がΨを満足すれば，入力記号がα_i であるようなどのような入力パラメータ値に対しても<s,s'>から<t,t'>に遷移しない）ならば，その辺を遷移グラフから取り除く．このような辺すべてをE'から取り除いた後，初期状態の２字組$<s_1,s'_1>$に相当する頂点からの有向道(path)が存在しない頂点や辺を取り除いたグラフG"(V",E") を作成し，このグラフをM1,M2,Ψに対する「遷移可能グラフ」と呼ぶ．尚，1.2.2 の定義より状態s,s'を整数に対応付けているので，Ψ(s,s',$\mathbb{B}$n,$\mathbb{C}$h)は$\mathbb{B}$n,$\mathbb{C}$h を変数とするＰ文となる．また，1.2.2 で述べた制約(2) より，A2($\mathbb{B}$n,$\mathbb{P}$ij)，B2($\mathbb{C}$h,$\mathbb{P}$ij)もそれぞれ

Ｐ文．よって，(*1)全体はプレスブルガー文となりその真偽は決定可能[5]．よって，任意のM1，M2及びＰ文Ψに対して，遷移可能グラフを作成できる．■

［命題１（等価性の十分条件）］

M1,M2及びＰ文Ψ(u,v,$\mathbb{B}$n,$\mathbb{C}$h)に対して，上で定義した遷移可能グラフをG"(V",E") とする．この時，次の①～④が成り立てば，M1とM2は等価である．

①M1とM2の初期状態の組$<s_1,s'_1>$及び初期レジスタ値の組$<r_{11},\cdots,r_{1n}>$，$<r'_{11},\cdots,r'_{1h}>$に対して，

Ψ($s_1,s'_1,<r_{11},\cdots,r_{1n}>,<r'_{11},\cdots,r'_{1h}>$)

--(*2)

が真である．

②G"(V",E") には，一方が死状態でなく，他方が死状態であるような状態の組（頂点）は存在しない．

③G"(V",E") において，E"に含まれる任意の辺<s,s'>→<t,t'>（但し，t,t'は死状態でない．また，辺のラベルを$<\alpha_i(\mathbb{P}ij),\{a_2,b_2\},\{A3(\mathbb{B}n,\mathbb{P}ij),B3(\mathbb{C}h,\mathbb{P}ij)\},\{a_4,b_4\}>$ とする）に対して，

∀ $\mathbb{B}$n,$\mathbb{C}$h,$\mathbb{P}$ij

［Ψ(s,s',$\mathbb{B}$n,$\mathbb{C}$h) ⊃

Ψ(t,t',A3($\mathbb{B}$n,$\mathbb{P}$ij),B3($\mathbb{C}$h,$\mathbb{P}$ij))］ --(*3)

が成り立つ．

④G"(V",E") において，E"に含まれる任意の辺<s,s'>→<t,t'>（但し，t,t'は死状態でない．また，辺のラベルを$<\alpha_i(\mathbb{P}ij),\{a_2,b_2\},\{a_3,a_4\},\{A4(\mathbb{B}n,\mathbb{P}ij),B4(\mathbb{C}h,\mathbb{P}ij)\}>$ とする）に対して，

∀ $\mathbb{B}$n,$\mathbb{C}$h,$\mathbb{P}$ij

［Ψ(s,s',$\mathbb{B}$n,$\mathbb{C}$h) ⊃

A4($\mathbb{B}$n,$\mathbb{P}$ij)＝B4($\mathbb{C}$h,$\mathbb{P}$ij)］ --(*4)

が成り立つ．■

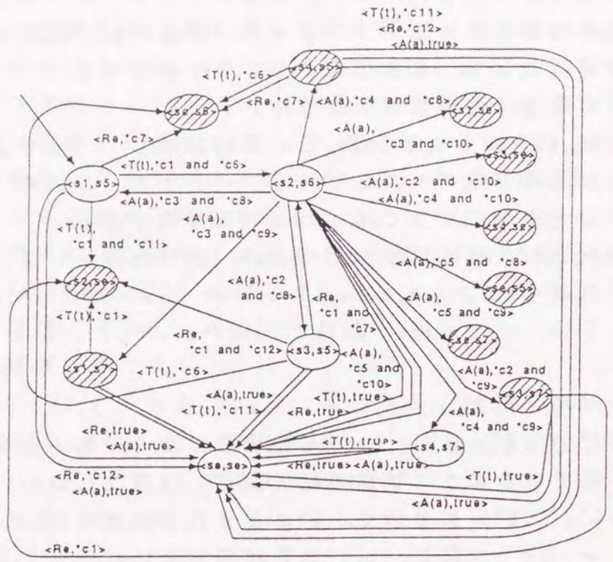


図１．４　Ｍ１，Ｍ２の遷移グラフ

（証明）1.4のはじめに述べた(1),(2) が成り立てば，M1,M2 は等価．そこで，①から③を満たせば(1) 及び(3) 分(ｗ),分'(ｗ)の値が共に真で，且つ，　δ̂(ｗ)=s,
　　δ̂'(ｗ)=s'の時，状態の組<s,s'>はV"-{<se,se>}
　　（但し，seは死状態を表す）の要素である．
が成り立つことを入力系列ｗの長さに関する並行帰納法を用いて証明し，次に(1),(3) 及び②，④を用いて(2) が成り立つことを証明する．
(A) 初期段階：

分，分'の定義よりｗが空系列の時，分(ｗ)，分'(ｗ)の値は共に真．また，①が成り立つので，空系列に対する(1) が成り立つ．さらに，初期状態の組<s₀I,s'I>はV"-{<se,se>}の要素．よって，(3) が成り立つ．
(B) 帰納段階：
長さがｋの任意の入力系列ｗに対して，(1),(3) が成り立つと仮定する．帰納法の仮定(1)より，分(ｗ)，分'(ｗ)の値は共に真か，共に偽かのいずれか．
(B-1) 分(ｗ)，分'(ｗ)の値が共に偽の場合：
分，分'の定義より，長さがｋ+1の任意の入力系列ｗ'=ｗ・α_i(Ｑij)に対しても，分(ｗ')，分'(ｗ')の値が偽となり，長さがｋ+1のｗ'に対する(1),(3) が共に成り立つ（但し，Ｑijはα_iに対する入力パラメータ値（整数）の組を表す）．
(B-2) 分(ｗ)，分'(ｗ)の値が共に真の場合：
帰納法の仮定(3) より，δ̂(ｗ)=s，δ̂'(ｗ)=s'とおいた状態の組<s,s'>はV"-{<se,se>}の要素となる．長さがｋ+1の任意の入力系列ｗ'=ｗ・α_i(Ｑij)を考える．δ̂(ｗ')=t，δ̂'(ｗ')=t'とおく．遷移グラフG'の定義より<s,s'>→<t,t'>はG'の辺である．この辺に対する(*1)の値は偽である（(*1)の［　］中の変数Ｂn，Ｃh，Ｐijにδ̂(ｗ)，δ̂'(ｗ)，Ｑij を代入すると，帰納法の仮定(1)よりΨ(s,s'，δ̂(ｗ)，δ̂'(ｗ))の値は真．さらに入力α_i(Ｑij)によって<t,t'> に遷移することから，A2(δ̂(ｗ)，Ｑij)，B2(δ̂'(ｗ)，Ｑij)の値は共に真となり，［　］全体が偽となる）ので，遷移可能グラフG"を作成する際に辺<s,s'>→<t,t'>が取り除かれることはない．よって<t,t'>はG"の頂点である．②を満たすことより，このt,t'は，(ｲ) t,t'共に死状態か，又は(ﾛ) 共に死状態でない．

(ｲ)の場合：　$\hat{\sigma}(w')$,$\hat{\sigma}'(w')$の値が共に偽となり，(1),(3)の定義より長さがｋ+1のｗ’に対する(1),(3)が成り立つ．

(ﾛ)の場合：　$\hat{\sigma}(w')$,$\hat{\sigma}'(w')$の値が共に真となる．状態の組<t,t'>は上述のように$V''-\{<se,se>\}$の要素であり，長さがｋ+1のｗ’に対する(3)が成り立つ．帰納法の仮定(1)より$\Psi(s,s',\hat{\delta}(w),\hat{\delta}'(w))$の値が真であること，及び③が成り立つことより状態の組<t,t'>においても$\Psi(t,t',\hat{\delta}(w'),\hat{\delta}'(w'))$の値が真である．よって，長さがｋ+1のｗ’に対する(1)が成り立つ．

次に(1),(3)及び②,④から(2)が成り立つことを示す．状態の組<s,s'>に対して$\hat{\sigma}(w)=s$,$\hat{\sigma}'(w)=s'$なる入力系列ｗが存在しない場合は考える必要がない．そのような入力系列ｗが存在する場合，(3)より<s,s'>は$V''-\{<se,se>\}$の要素．上述の議論と同じように(1)及び②を用いて，任意のレジスタ値と入力$\alpha_i(\in Qij)$の組に対して，次の状態は共に死状態か共に死状態でないかのどちらかである．共に死状態の場合，上述の(ｲ)の場合同様 (2)の$\eta_{<s,\alpha_i>}(\cdots)$,$\eta'_{<s',\alpha_i>}(\cdots)$の値は共に偽となり(2)が成り立つ．共に死状態でない場合，上述の(ﾛ)の場合同様 (2)の$\eta_{<s,\alpha_i>}(\cdots)$,$\eta'_{<s',\alpha_i>}(\cdots)$の値は共に真となる．④が成り立つので，そのようなレジスタ値と入力に対して出力は等しく，従って(2)が成り立つ．　（証明終）

［命題２］

M1,M2 及びＰ文Ψが与えられた時，命題１の①～④が成り立つかどうかは決定可能である．　■

（証明）前述のように，任意のマシンの組M1, M2及びＰ文Ψに対して，遷移可能グラフが構成できる．又，$\Psi(u,v,\mathbb{B}n,\mathbb{C}h)$はＰ文ゆえ(*2)は変数を含まないＰ文．よって①の真偽は決定可能．②は遷移可能グラフを構成すれば判定可能．③については1.2.2の定義よりs,s'は整数であり，(*3)の$\Psi(s,s',\mathbb{B}n,\mathbb{C}h)$は，$\mathbb{B}n$,$\mathbb{C}h$を変数とするＰ文．又，次のレジスタ値を指定する関数A3($\mathbb{B}n,\mathbb{P}ij$),B3($\mathbb{C}h,\mathbb{P}ij$)も1.2.2で述べた制約(3)より，各々$\mathbb{B}n$と$\mathbb{P}ij$（又は$\mathbb{C}h$と$\mathbb{P}ij$）の１次結合で表される．よって，$\Psi(t,t',A3(\mathbb{B}n,\mathbb{P}ij),B3(\mathbb{C}h,\mathbb{P}ij))$も$\mathbb{B}n,\mathbb{C}h,\mathbb{P}ij$を変数とするＰ文．ゆえに(*3)もプレスブルガー文となり，その真偽は決定可能[5]．　④についても③同様出力を指定する関数A4($\mathbb{B}n,\mathbb{P}ij$), B4($\mathbb{C}h,\mathbb{P}ij$)が$\mathbb{B}n$と$\mathbb{P}ij$（または$\mathbb{C}h$と$\mathbb{P}ij$）の１次結合で表される．よって(*4)もプレスブルガー文となり，その真偽は決定可能[5]．　（証明終）

第１章では，M1,M2及びＰ文$\Psi(u,v,\mathbb{B}n,\mathbb{C}h)$に対して命題１の①～④が共に成り立つ時，且つその時のみM1とM2はΨ等価であると定義する．尚，$\forall x_1,..,x_n(A)$の形のプレスブルガー文の真偽はＰ文A の否定が整数解を持たないかどうかを判定すればよく，整数線形計画問題の解の非存在性の判定手続きを用いて判定できる[6]．

次に，図１．２，１．３で与えた２つのマシンM1,M2に対して，Ψ等価となるようなＰ文Ψを与える．

$$\Psi(u,v,<r_1,r_2>,<r_3,r_4,r_5>)$$
$$=\{(r_1+r_4=10)\ \text{and}\ (r_2=r_5)\ \text{and}\ (u=s_1 \supset r_3=1)\ \text{and}\ (u=s_3 \supset r_3=0)\}\quad --(*5)$$

（但し，u,v はそれぞれM1,M2 の状態を表す変数，r_1,r_2はM1のレジスタＲ1,Ｒ2 の値を表す変数，r_3,r_4,

r_5はM2のレジスタＲ3,Ｒ4,Ｒ5 の値を表す変数）.

図１.４において，斜線で塗りつぶされた頂点はM1, M2 の遷移グラフからM1,M2,Ψに対する遷移可能グラフを作成する際に取り除かれる（遷移不可能な）頂点であり，斜線で塗りつぶされていない頂点とその間を結ぶ辺が，M1,M2,Ψに対する遷移可能グラフを表している．さらに，図１.２，１.３で与えたM1，M2及び(*5)で指定したＰ文Ψに対して，命題１の①～④が成り立つ．よって，M1とM2はΨ等価となり，従ってM1とM2は等価である．

マシンM1，M2及びＰ文Ψを与えて，M1,M2 がΨ等価かどうかの判定は計算機を用いて機械的に行えるので，上の方法は，等価性を証明する際の有効な方法に成り得ると考えられる．

１.５　検証例

筆者らは，マシンM1，M2及びＰ文Ψを入力として，マシンM1，M2のΨ等価性を判定するプログラムを作成し[3][4]，データリンクレベルにおけるSDLC手順の２次局[8] に相当する２つのマシンについて等価性の証明を行った．証明に使用したマシンの１つは，あるメーカーの作成した装置[9] をそのまま今のモデルで記述したもの（マシンＡと呼ぶ）であり，もう１つのマシンは，マシンＡと全く独立に筆者らが文献(8) の仕様書を基に作成したもの（マシンＢと呼ぶ）である．

入力記号は12個で，各入力記号は０～４個のパラメータを持つ．出力は整数の８字組である．２つのマシンA，Ｂの状態数は各々11個と７個，レジスタの数は各々４個と６個である．又，各マシンの仕様を図１.

１のようなグラフで表現した場合，それぞれの状態から最大５個（平均２個）程度の有向辺が出ている．

このような２つのマシンＡ，Ｂに対して，マシンＡ，Ｂの状態とレジスタ間で常に成り立つと思われる関係を12個抽出し，それら12個の関係が共に成り立つとしてΨを指定した．これらのＡ,Ｂ,Ψに対して，約2000個のプレスブルガー文の真偽を判定し，およそ３０分のcpu 時間(SUN3/60)でＡ，ＢがΨ等価であるという結果を得た．

尚，作成したプログラムでは判定時間を短縮するため，整数解の非存在性を調べる代わりに，実数解の非存在性を調べている．この方法では，整数解は存在しないけれども実数解が存在する場合は証明に失敗するが，ｘ＜ｙをｘ≦ｙ－１に変形する等の処置を講じて，できるだけ作成したプログラムでも証明が失敗しないよう配慮した．このため，筆者らが行った幾つかの検証例では，すべて上述の方法で証明が成功した．また，作成したプログラムでは，検証者が与えたΨに対してΨ等価であると判定出来なかった場合でも，遷移可能グラフを出力し，どの頂点（状態の２字組）と入力記号の組が命題１の①～④を満足しなかったかを表示する．これらの表示からどのようにΨを変更すれば，あるいは，どのようにM1，M2の仕様を変更すればΨ等価となるか，についての有益な情報が得られる．

1.6 結言

第１章では，検証者が与えたＰ文Ψに対して，２つのプロトコルマシン間のΨ等価性を定義し，それらがΨ等価であるかどうかを判定する手続きを与えた．またその手続きを実際の装置例（SDLC手順の２次局）に適用し，「等価性」（従って「互換性」）の証明において，第１章の方法が有効であることを確かめた．尚，現在第１章で述べたモデルや検証法が等価性以外の検証の問題に適用出来るかどうかを検討している．

第２章
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法

2.1 序言

近年のデータ通信の発展に伴い，通信の手順を規定する通信プロトコル（以下，単にプロトコルと呼ぶ）の厳密かつ形式的な記述やその正しさの検証が重要な課題になっており，様々な研究が行われている[10]．

通常，プロトコルは，正常な状況（正常に送受信を行っているような状況）における各プロトコルマシン（通信制御装置）の動作を記述した部分と，異常な状況（回線に障害等が発生したような状況）からどのように正常な状況に復帰するかを記述した部分，に分類されることが多い．この際，「異常な状況からいつかは正常な状況に復帰すること」（エラーリカバリ性を持つという）を機械的に証明できれば望ましい．しかし，従来，エラーリカバリ性を証明する有効な方法がなく，テスト手法による検査が行われてきたが，このようなテスト手法では必ずしもエラーリカバリ性を論理的に保証できない．本論文では，プロトコルマシンの動作を記述するためのモデルを一つ定め，そのモデルで記述されたプロトコルマシン対がエラーリカバリ性を持つことを機械的に証明するための方法を提案する[12]．

第１章と同様に，各プロトコルマシンは，「有限制御部」と任意の整数値を保持する有限個の「レジスタ」を持つオートマトンとして記述し，このオートマトン上での演算を+,-,=,>,and,or,notに限定する．このモデルで記述した２つのプロトコルマシンM1，M2の状態とレジスタ値の組（以下，M1，M2対の全状態，または単に全状態と呼ぶ）を引数とする述語Φを線形不等式の論理結合の形で指定する．述語Φの値は，その値が真の時は「正常な状況」を表すように指定し，「M1，

M2の初期状態と初期レジスタ値の組から，M1とM2が（通信回線の誤り等も考慮して）送受信等の動作を行うことによって遷移する任意の全状態に対して，たとえその全状態がΦを満足しない全状態であっても，それ以降可能などのような動作を続けてもいつかは再びΦを満足する全状態に遷移する」時，M1，M2はΦに対するエラーリカバリ性を持つと定義する．

提案するモデルのマシン及びΦのクラスにおいては，M1，M2がΦに対するエラーリカバリ性を持つ場合でも，そのことを証明するアルゴリズムが存在しない．そこで，必ずしも証明に成功する保証はないが，「Φを満足する全状態からどのような動作を行っても再びΦを満足する全状態へ遷移すること」を保証するための一つの手続きを与え，その手続きを再帰的に用いることによって，M1，M2がΦに対するエラーリカバリ性を持つことを証明する．本章では，第１章と同様に，M1,M2がΦに対するエラーリカバリ性を持つことを整数線形計画問題の解の非存在性の証明に帰着して機械的に証明している．またその証明手続きを用いて，HDLC手順[14]に従うあるプロトコルマシン対のエラーリカバリ性を検証することにより，提案する方法の有効性を示した．

2.2 プロトコルのモデル化

本章では，第１章と同様に，プロトコルマシンを「有限制御部」及び任意の整数値を保持する有限個の「レジスタ」を持つオートマトンとして記述する．また，「相手局からのフレームの受信」や「タイムアウトの発生」，「相手局へのフレームの送信」等を各々オー

トマトンの「入力記号」とみなす．入力記号はその動作の種類により，

(1) 相手局へのフレームの送信動作に相当する入力記号の有限集合（送信動作集合）I_s．

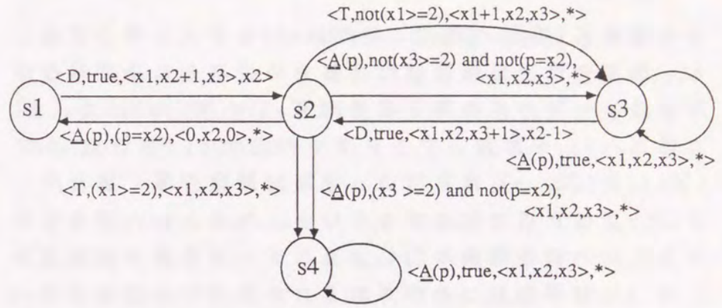
(2) 相手局からのフレームの受信動作に相当する入力記号の有限集合（受信動作集合）I_r．

(3) その局内で単独に行われる動作に相当する入力記号の有限集合(単独動作集合)I_o．

に分類する．例えば，フレームの送信や再送はI_sに，受信はI_rに，タイムアウトの発生はI_oに属する．送信動作集合I_sに属する入力記号に対してのみ「出力」が定義される．出力は整数とし，相手局へ送信されるフレームに相当する．また，受信動作集合I_rに属する入力記号のみ「パラメータ」を持ってもよい．パラメータ値は整数とし，通信相手局から受信したフレームに相当する．

プロトコルマシンは，現在の状態（以下，状態は有限制御部の状態を表す），レジスタ値及び入力（入力記号とその入力記号のパラメータ値の組）の組に対して，次の状態，次のレジスタ値，出力が定義されるいわゆるMealy 型の決定性オートマトンとしてモデル化する．モデル化したプロトコルマシンの仕様は，図２.１，２.２（例１，２参照）のような有向グラフで記述する．

図において，グラフの頂点はプロトコルマシンの状態を表し，有向辺及びそのラベルによって各入力に対する次の状態，次のレジスタ値，出力を指定する．グラフの各辺s → s'には，４字組のラベル＜入力，条件式，次のレジスタ値，出力＞が付加されており，その第１成分（入力）に書かれた入力記号αがI_s, I_r,



States s1: Initial state (Transmission state)
s2: Ack-waiting state
s3: Retransmission state
s4: Operetor-notification state

Registers x1: The number of Timeout (Initial value=0)
x2: The number of Transmission (Initial value=0)
x3: The number of Retransmission (Initial value=0)

Input symbols D: Transmission
A: Reception of acknowledgement (p: parameter of "A")
T: Timeout
Is={D}, Ir={A}, Io={T}

図２.１　プロトコルマシンＭ１

I_oの何れに属するかに応じて各々①～③のようなラベルが与えられる．

① $<\alpha, c(X_n), \delta(X_n), \rho(X_n)>$　　($\alpha \in I_s$)

② $<\alpha(p), c(X_n, p), \delta(X_n, p), *>$　　($\alpha \in I_r$)

③ $<\alpha, c(X_n), \delta(X_n), *>$　　($\alpha \in I_o$)

ここで，nはレジスタの数，X_nは現在のレジスタ値を表す変数$x_1, x_2, \ldots, x_n$の組を表す．pは入力記号αのパラメータ値を表す変数である．"2x+3y+10"のように整数と変数の線形結合をＰ項と呼び，"not(y=w+1) and (x+y>z-1)"のように，Ｐ項,=,>,and,or,not から

なる論理式(線形不等式の論理結合)をＰ文と呼ぶ（但し，以下では表現を簡単にするため便宜上≧やコも使用する）．ラベルの第２成分c($\mathbb{X}_n$),c($\mathbb{X}_n$,p)は$\mathbb{X}_n$(または$\mathbb{X}_n$,p）を変数とするＰ文で指定する．第３成分δ($\mathbb{X}_n$),δ($\mathbb{X}_n$,p）はＰ項のｎ字組で指定する．また，ρ($\mathbb{X}_n$）はＰ項で指定する．いま，与えられたプロトコルマシンの仕様中に①のようなラベルを持つ有向辺 s → s' が存在し，そのプロトコルマシンの現在の状態が s，その時のレジスタ値の組が $\mathbb{R}_n$，入力がα(∈I_s)であるとする．この時，c($\mathbb{R}_n$）の値が真ならば，次の状態はs'，次のレジスタ値の組はδ($\mathbb{R}_n$）となり，ρ($\mathbb{R}_n$）が相手局に出力される．また，②，③のようなラベルを持つ有向辺の場合，出力は指定されないので，ラベルの第４成分を＊としている．尚，一つの頂点からラベルの第１成分の入力が同じであるような有向辺が２本以上出てもよいが，それらの辺のラベルの第２成分の条件式は，オートマトンが決定性に動作するように指定されなければならない．

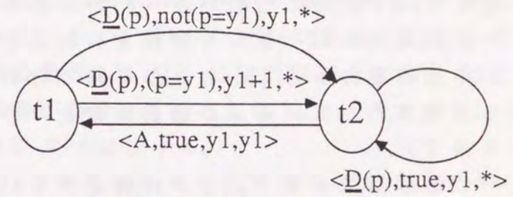
[例１　プロトコルマシンM1]

図２．１のプロトコルマシンM1はデータリンクレベルにおけるHDLC手順の一次局を一部変更し，かつ簡単化したものである．このマシンは，HDLC手順におけるウィンドウサイズを１に制限したモデルであり，フレームを１個送信（動作Ｄ）し，それに対する応答フレームを受信（動作A）するという動作を繰り返す．もし，フレームの送信後にタイムアウト（動作Ｔ）が発生した場合，フレームが正しく受信されなかったと判断し，そのフレームの再送（動作Ｄ）を行う．タイムアウトの発生回数（レジスタX_1の値）が2 以上，またはフレームの再送回数（レジスタX_3の値）が2 以上となった場合には，オペレータによる回復を必要とする特別の状態s_4（オペレータ通告状態と呼ぶ）へ遷移する．尚，レジスタX_2は，新規フレームの送信回数を保持するために用いられる．■

例えば，現在の状態がs_2，レジスタX_1,X_2,X_3の値が各々r_1,r_2,r_3，入力がＴの場合，r_1が2 未満ならば状態s_3に遷移し，X_1,X_2,X_3の値は各々r_1+1,r_2,r_3となる．r_1が2 以上ならば状態s_4に遷移するが，レジスタX_1,X_2,X_3の値は変化しない．尚，入力記号ＴはＩ。に属するので出力は指定されない．

[例２　プロトコルマシンM2]

図２．２のプロトコルマシンM2はデータリンクレベルにおけるHDLC手順の二次局を簡単化したものであり，図１のM1と送受信を行う．M2には相手局への応答フレームの送信(動作Ａ)と相手局からのフレームの受信（動作D）という２つの動作があり，相手局から受信したフレームの個数を保持するレジスタY_1がある．■



States　t1: Initial state (Reception state)
　　t2: Ack-transmission state

Register　y1: The number of Reception (Initial value=0)

Input symbols　D: Reception (p: parameter of "D")
　　A: Transmission of acknowledgement
　　Is={A}, Ir={D}, Io={}

図２．２　プロトコルマシンＭ２

2.3 通信系の動作

本論文では互いに通信を行う2つのプロトコルマシン対から成る通信系を考え，次のような仮定の下でエラーリカバリ性を議論する．

(A) フレームを送信する場合，そのフレームが送信と同時に相手局に正しく受信されるか，あるいは，送信と同時に通信回線上で消失するかの何れかである．

(B) 受信が正しく行われた場合，送信局の送信動作によって出力される整数値が受信局の受信動作に相当する入力記号のパラメータ値として受け渡される．

以上の仮定より，プロトコルマシン対の動作として次の6つの動作を考え，これらを通信系の動作と呼ぶ．尚，M1,M2の送信動作集合をI_{s1}, I_{s2}，受信動作集合をI_{r1}, I_{r2}，単独動作集合をI_{o1}, I_{o2}，で各々表す．

[通信系の動作]

(1) M1の送信動作$\alpha \in I_{s1}$とαに対応するM2の受信動作$\underline{\alpha} \in I_{r2}$が同時に行われる（M1から送信されたフレームが同時にM2へ正しく受信されたことを表す）．

(2) M1の送信動作$\alpha \in I_{s1}$のみが単独で行われる（M1から送信されたフレームが通信回線上で消失したことを表す）．

(3) M2の送信動作$\beta \in I_{s2}$とβに対応するM1の受信動作$\underline{\beta} \in I_{r1}$が同時に行われる．

(4) M2の送信動作$\beta \in I_{s2}$のみが単独で行われる．

(5) M1の単独動作集合I_{o1}に属する動作γが単独で行われる．

(6) M2の単独動作集合I_{o2}に属する動作γが単独で行われる． ■

本論文では(1)～(6)の通信系の動作を各々[α, $\underline{\alpha}$], [α, ＊], [$\underline{\beta}$, β], [＊, β], [γ, ＊], [＊, γ] で表す．又，[α, $\underline{\alpha}$]([$\underline{\beta}$, β])については，M1(M2)からα(β)によって出力される整数値が，M2(M1)の$\underline{\alpha}$($\underline{\beta}$)の入力パラメータ値として割り当てられるとする．

例えば，図2.1，2.2のプロトコルマシンM1，M2において，M1，M2の状態が各々s_1, t_1 で，レジスタX_1, X_2, X_3 の値が各々0,3,0 で，レジスタY_1 の値が3であったとする．M1で $<D, true, <x_1, x_2+1, x_3>, x_2>$ なるラベルを持つ有向辺$s_1 \rightarrow s_2$が存在するので，通信系の動作[D, ＊]（M1がフレームの送信を行ったがそのフレームが回線上で消失したことを表す）が実行可能となり，[D, ＊]の実行後，M1の状態のみがs_1からs_2に変化し，レジスタX_2 の値が1 増加する．

さらに，M2では$<\underline{D}(p), (p=y_1), y_1+1, *>$なるラベルの有向辺$t_1 \rightarrow t_2$が存在する．M1の出力（レジスタ$X_2$の値）3 をM2の入力記号$\underline{D}$のパラメータ（変数p ）に割り当ててM2の$t_1 \rightarrow t_2$のラベルの条件$(p=y_1)$を評価すると真となり，かつ，M1の$s_1 \rightarrow s_2$の条件も真であるので，通信系の動作[D, $\underline{D}$] は実行可能となり，実行後，M1，M2の状態が各々s_2, t_2に変化し，レジスタX_2, Y_1の値が各々1 ずつ増加する．

尚，M1,M2の状態対を一つの状態とみなし,(1)～(6)のような通信系の動作を入力記号とみなすと，プロトコルマシン対全体を一つのオートマトン（有限制御部と有限個のレジスタから成るMealy 型のオートマトン）とみなすことができる．以下では,このオートマトンをM1，M2の対オートマトンと呼ぶ．また，$<s_1, t_1, <0,3,0>, <0>>$ のように，M1，M2の状態とレジスタ値の組をM1，M2対の全状態（あるいは単に全状態）と呼ぶ．

2.4 エラーリカバリ性の定義

まず，２つのプロトコルマシンM1，M2対の全状態を引数とする述語Φを Ｐ文で指定する．述語Φの値は，その値が真の時は「正常な状況」を表すように検証者が指定する．例えば，図２．１，２．２のプロトコルマシンM1，M2に対して，M1，M2対が次の(1)または(2)を満足する時，正常な状況であるとみなすことにする．

(1) M1から送信された新規フレームの個数（レジスタ X_2 の値）とM2が正しく受信したフレームの個数（レジスタ Y_1 の値）が等しく，且つ最後に送信したフレームに対するタイムアウトの発生回数（レジスタ X_1 の値）や再送回数（レジスタ X_3 の値）が共に0 である．

(2) M1がオペレータによるリカバリを必要とする特別な状態（オペレータ通告状態 s_4）にいる+．

この場合，次のような述語Φを指定すればよい++．

脚注 +: 通常，状態 s_4 に遷移した場合，オペレータが相手局のオペレータと協調してM1，M2の状態やレジスタ値を初期化するという回復動作が行われ，それにより，エラーから「リカバリ」したと考えるのが妥当である．しかし，本論文では，簡単のため，s_4 に遷移することによって「リカバリ性」を持つというような表現を用いる．

脚注 ++: 本論文では，状態 $s_0, s_1, s_2, ...$ を各々整数（例えば0,1,2,..）とみなすことにする．その場合，Φ中の $(u_1 = s_4)$ はＰ文となり，ΦもＰ文となる．

$\Phi(u_1, u_2, x_1, x_2, x_3, y_1) =$
$\{(x_2 = y_1)$ and $(x_1 = 0)$ and $(x_3 = 0)\}$
or $(u_1 = s_4)$ …(i)

（u_1, u_2 は各々M1,M2の状態を表す変数．x_1, x_2, x_3 は各々M1のレジスタ X_1, X_2, X_3 の値を表す変数．y_1 はM2のレジスタ Y_1 の値を表す変数．）

[エラーリカバリ性の定義]: M1，M2の初期状態と初期レジスタ値の組から2.3で述べた通信系の動作を行うことによって遷移する任意の全状態に対して，たとえその全状態がΦを満足しない全状態であっても，それ以降可能などのような通信系の動作を続けても，いつかは再びΦを満足する全状態に遷移する時，M1，M2はΦに対するエラーリカバリ性を持つという． ■

一般に，異なるΦを指定することによって，異なるエラーリカバリ性を議論できる．

エラーリカバリ性の定義より，M1，M2がΦに対するエラーリカバリ性を持てば，M1，M2は時々Φを満足しながら無限に遷移し続けるか，あるいは，遷移を終了する時は必ずΦを満足する全状態で終了する．

[命題１][15]

2.2で述べたプロトコルマシン及びΦのクラスに対しては，M1，M2がΦに対するエラーリカバリ性を持つ場合でも，そのことを証明するアルゴリズムは存在しない． ■

2.5 エラーリカバリ性の検証法

命題１が成り立つので，以下では，必ずしも証明に成功するとは限らないが，もし，成功すればエラーリカバリ性を保証する一つの証明手続きを与える．

まず幾つかの定義を行う．図２．１，２．２の$<x_1, x_2+1, x_3>$や$<y_1+1>$のようにM1,M2 のレジスタ値を計算するためのＰ項の組を各々M1,M2 のレジスタ計算式と呼び，

$<s_2, t_2, <x_1, x_2+1, x_3>, <y_1+1>>$

のように，M1,M2の状態対とM1,M2のレジスタ計算式の４字組をΣ式と呼ぶ．また，特別な場合として，

$<s_1, t_1, <x_1, x_2, x_3>, <y_1>>$

のように，M1,M2 のレジスタ計算式が変数のみからなるΣ式を特にΣ項と呼ぶ．

図２．１，２．２のような２つのプロトコルマシンM1,M2及び$<s_1, t_1, <x_1, x_2, x_3>, <y_1>>$のようなM1,M2の一つのΣ項が与えられると，図２．３のように，与えられたΣ項からM1,M2 の状態やレジスタ値がどのように変化していくかを表す（無限の大きさの）グラフ（木）を考えることが出来る．すなわち，図２．３の根ノードにはM1,M2 の一つのΣ項$<s_1, t_1, <x_1, x_2, x_3>, <y_1>>$がラベルとして付加されており，各枝には通信系の動作と条件式の対がラベルとして付加されている．各ノードにはΣ式がラベルとして付加されており，根ノードからそのノードに至るまでの枝に付加された通信系の動作を順次実行した時に到達する状態対とその時点のレジスタ値が根ノードのレジスタ値のどのような関数で表されるかが記述されている．例えば，図２．３の

$<s_1, t_1, <x_1, x_2, x_3>, <y_1>>$
$\rightarrow <s_2, t_1, <x_1, x_2+1, x_3>, <y_1>>$
$\rightarrow <s_3, t_1, <x_1+1, x_2+1, x_3>, <y_1>>$

という道を辿ることによって，[Ｄ，＊]，[Ｔ，＊]なる動作を実行した時に状態対とレジスタ値がどのように変化するかがわかる．

以下では，M1,M2 及び一つのΣ項$<s_0, t_0, \mathbb{X}_n, \mathbb{Y}_m>$（但し，$\mathbb{X}_n, \mathbb{Y}_m$ は各々M1，M2のレジスタ値を表す変数の組）が与えられた時，$<s_0, t_0, \mathbb{X}_n, \mathbb{Y}_m>$ を根ノードのラベルとする（一般に無限の大きさの）木が次のような条件を満足する時，その木を$<s_0, t_0, \mathbb{X}_n, \mathbb{Y}_m>$に対する遷移グラフと呼ぶ．

［遷移グラフの条件］

与えられた木のすべてのノード$<s, t, \mathbb{B}_n, \mathbb{C}_m>$（$\mathbb{B}_n$, $\mathbb{C}_m$は各々$\mathbb{X}_n$,$\mathbb{Y}_m$を変数とするM1,M2 のレジスタ計算式）に対して，次の①～④の条件が成り立つ．

① αがM1の送信動作又は単独動作でM1に$<\alpha, c_1(\mathbb{X}_n), \delta_1(\mathbb{X}_n), \rho_1(\mathbb{X}_n)>$なるラベルの有向辺$s \rightarrow s'$が存在する時，且つその時のみ$<s', t, \delta_1(\mathbb{B}_n), \mathbb{C}_m>$なるラベルのノードが存在し，$<[\alpha, *], c_1(\mathbb{B}_n)>$なるラベルの枝

$<s, t, \mathbb{B}_n, \mathbb{C}_m> \rightarrow <s', t, \delta_1(\mathbb{B}_n), \mathbb{C}_m>$

が存在する．

② αがM1の送信動作で$\underline{\alpha}$がM2の受信動作で，M1に$<\alpha, c_1(\mathbb{X}_n), \delta_1(\mathbb{X}_n), \rho_1(\mathbb{X}_n)>$なるラベルの有向辺$s \rightarrow s'$が存在し，M2に$<\underline{\alpha}(p), c_2(\mathbb{Y}_m, p), \delta_2(\mathbb{Y}_m, p), *>$なるラベルの有向辺$t \rightarrow t'$が存在する時，且つその時のみ$<s', t', \delta_1(\mathbb{B}_n), \delta_2(\mathbb{C}_m, \rho_1(\mathbb{B}_n))>$なるラベルのノードが存在し，$<[\alpha, \underline{\alpha}], \{c_1(\mathbb{B}_n)$ and $c_2(\mathbb{C}_m, \rho_1(\mathbb{B}_n))\}>$なるラベルの枝

$<s, t, \mathbb{B}_n, \mathbb{C}_m>$
$\rightarrow <s', t', \delta_1(\mathbb{B}_n), \delta_2(\mathbb{C}_m, \rho_1(\mathbb{B}_n))>$

が存在する．

③ βがM2の送信動作又は単独動作でM2に$<\beta, c_2(\mathbb{Y}_m), \delta_2(\mathbb{Y}_m), \rho_2(\mathbb{Y}_m)>$なるラベルの有向辺$t \rightarrow t'$が存在する時，且つその時のみ$<s, t', \mathbb{B}_n, \delta_2(\mathbb{C}_m)>$な

るラベルのノードが存在し，<[＊，β]，$c_2(\mathbb{C}_m)$>なるラベルの枝

<s，t，$\mathbb{B}_n$，$\mathbb{C}_m$>→<s，t'，$\mathbb{B}_n$，$\delta_2(\mathbb{C}_m)$>

が存在する．

④ $\underline{\beta}$がM1の受信動作でβがM2の送信動作で，M1に<$\underline{\beta}$(p)，$c_1(\mathbb{X}_n,p)$，$\delta_1(\mathbb{X}_n,p)$，＊>なるラベルの有向辺s→s'が存在し，M2に<β，$c_2(\mathbb{Y}_m)$，$\delta_2(\mathbb{Y}_m)$，$\rho_2(\mathbb{Y}_m)$>なるラベルの有向辺t→t'が存在する時，且つその時のみ<s'，t'，$\delta_1(\mathbb{B}_n,\rho_2(\mathbb{C}_m))$，$\delta_2(\mathbb{C}_m)$>なるラベルのノードが存在し，<[$\underline{\beta}$，$\beta$]，{$c_1(\mathbb{B}_n,\rho_2(\mathbb{C}_m))$ and $c_2(\mathbb{C}_m)$}>なるラベルの枝

<s，t，$\mathbb{B}_n$，$\mathbb{C}_m$>

→<s'，t'，$\delta_1(\mathbb{B}_n,\rho_2(\mathbb{C}_m))$，$\delta_2(\mathbb{C}_m)$>

が存在する． ■

いま，Σ項<s_θ，t_θ，$\mathbb{X}_n$，$\mathbb{Y}_m$>に対する遷移グラフの中に<[a，b]，c($\mathbb{B}_n$，$\mathbb{C}_m$)>なるラベルを持つ枝

<s，t，$\mathbb{B}_n$，$\mathbb{C}_m$>→<s'，t'，$\mathbb{B}'_n$，$\mathbb{C}'_m$>

が存在したとする．この時，条件(a)

$\forall\ \mathbb{X}_n,\mathbb{Y}_m$

[$\Phi(s_\theta,t_\theta,\mathbb{X}_n,\mathbb{Y}_m) \supset \mathrm{not}(c(\mathbb{B}_n,\mathbb{C}_m))$] …(a)

の値が真ならばこの枝を「遷移する可能性のない枝」と呼び，(a)の値が偽ならばこの枝を「遷移する可能性のある枝」と呼ぶ．

また，<s_θ，t_θ，$\mathbb{X}_n$，$\mathbb{Y}_m$>に対する遷移グラフの各ノード<s，t，$\mathbb{B}_n$，$\mathbb{C}_m$>に対して，条件(b)

$\forall\ \mathbb{X}_n,\mathbb{Y}_m$

[$\Phi(s_\theta,t_\theta,\mathbb{X}_n,\mathbb{Y}_m) \supset \Phi(s,t,\mathbb{B}_n,\mathbb{C}_m)$] …(b)

の値が真ならば，ノード<s，t，$\mathbb{B}_n$，$\mathbb{C}_m$>をΦ復帰ノードと呼ぶ．<s，t，$\mathbb{B}_n$，$\mathbb{C}_m$>がΦ復帰ノードならば，根

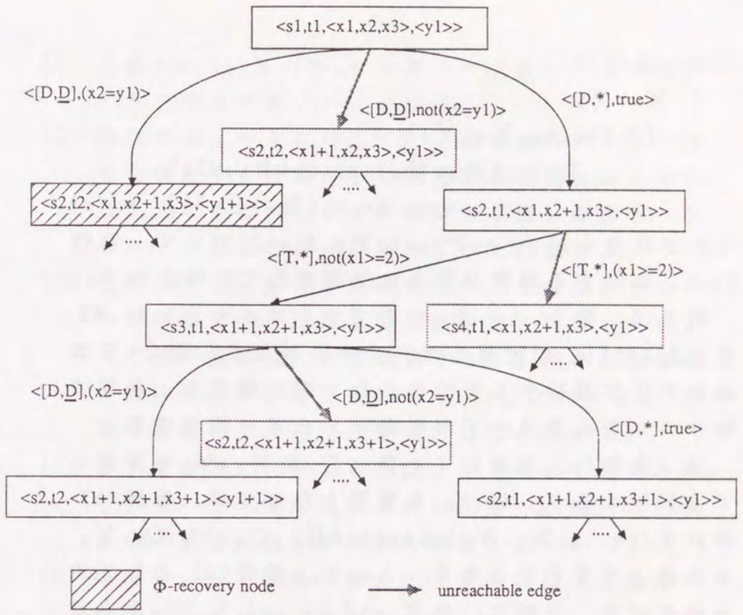


図２．３　状態とレジスタ値の変化を表わすグラフ

ノードからこのノードに至る枝に付加された通信系の動作を順次実行することによって，全状態<s_θ，t_θ，$\mathbb{R}_n$，$\mathbb{V}_m$>から全状態<s，t，$\mathbb{R}'_n$，$\mathbb{V}'_m$>に遷移した時，もし前者の全状態がΦを満足しているならば，後者の全状態もまたΦを満足する．

さらに<s_θ，t_θ，$\mathbb{X}_n$，$\mathbb{Y}_m$>に対する遷移グラフの各ノード<s，t，$\mathbb{B}_n$，$\mathbb{C}_m$>から<[a，b]，$c_1(\mathbb{B}_n,\mathbb{C}_m)$>，<[a，b]，$c_2(\mathbb{B}_n,\mathbb{C}_m)$>，……，<[a，b]，$c_p(\mathbb{B}_n,\mathbb{C}_m)$>のようにラベルの第１成分の動作が同じであるような枝（動作[a，b]に対する枝と呼ぶ）がｐ本出ていたとする．こ

の時，条件(c)

$\forall\ \mathbb{X}_n, \mathbb{Y}_m$

$[\Phi(s_\alpha, t_\alpha, \mathbb{X}_n, \mathbb{Y}_m)$

$\supset \{c_1(\mathbb{B}_n, \mathbb{C}_m)$ or $c_2(\mathbb{B}_n, \mathbb{C}_m)$

or …… or $c_p(\mathbb{B}_n, \mathbb{C}_m)\}]$ …(c)

の値が真ならば，ノード<s,t,$\mathbb{B}_n$,$\mathbb{C}_m$>において，動作[a,b]に対する遷移が完全に定義されているという．

例えば，図２．１，２．２のプロトコルマシンM1,M2及び2.4の(i)で定義したＰ文Φに対して，図２．３の破線の枝が遷移する可能性のない枝に相当し，斜線で塗りつぶされたノードがΦ復帰ノードに相当する．

尚，条件(a)において，$\mathbb{B}_n$,$\mathbb{C}_m$は$\mathbb{X}_n$,$\mathbb{Y}_m$を変数とするＰ項であり，s_α,t_αを整数とみなせば，条件(a)中の$\Phi(s_\alpha, t_\alpha, \mathbb{X}_n, \mathbb{Y}_m)$もnot(c($\mathbb{B}_n$,$\mathbb{C}_m$))も$\mathbb{X}_n$,$\mathbb{Y}_m$を変数とするＰ文となる．よって，条件(a)の[ ]内全体もＰ文．一般に，Ｐ文,and,or,not,∀,∃からなる論理式の内，その論理式に現れるすべての変数が∀,∃で束縛されているような論理式をプレスブルガー文と呼び，その真偽は決定可能であることが知られている[11]．上述の条件(a)全体は，∃を含まない

$\forall x1, x2, \ldots, xn[\Psi(x1, x2, \ldots, xn)]$

の形のプレスブルガー文である．その真偽判定については2.6参照．同様に条件(b),(c)の真偽も判定可能である．

[Φ-到達可能性グラフ]

Ｐ文Φ，M1，M2及びM1,M2の状態対<s_α,t_α>が与えられて，次のような手続きを実行し，(4)で手続きが終了した時，かつその時のみ，手続き終了時に作成されたグラフ（木）を<s_α,t_α>に対するΦ-到達可能性グラフと定義する．

(1) Σ項<s_α,t_α,$\mathbb{X}_n$,$\mathbb{Y}_m$>を根ノードとし，まず根ノードを注目するノードとする．

(2) 注目するノードに対して，上述の遷移グラフの条件を満足する枝の中から遷移する可能性のある枝（上述の条件(a)の値が偽になる枝）を選び，新しいノードとそれらの枝をすべて付加する．

(3) 上の(2)で付加された枝を調べ，その注目するノードにおいて，遷移が完全に定義されていない（上述の条件(c)の値が偽である）動作が存在する時は手続きを終了する（<s_α,t_α>に対するΦ-到達可能性グラフの作成に失敗したと考える）．

(4) すべての葉ノードがΦ復帰ノードなら手続きを終了する．その時に生成された木を<s_α,t_α>に対するΦ-到達可能性グラフと定義する．

(5) Φ復帰ノード以外の葉ノードがあれば，任意の一つを注目するノードとし，(2)へ． ■

尚，この手続きでは，手続きが永久に終了しないこともあり，その場合も<s_α,t_α>に対するΦ-到達可能性グラフの作成に失敗したと考える．定義より，上述の手続きでΦ-到達可能性グラフの作成に成功した場合，(5)で選ぶ葉ノードの順に依存せず，同じ到達可能性グラフが得られる．逆に，(3)で終了したとき，葉ノードの選ぶ順にかかわらず，Φ-到達可能性グラフの作成は成功しない．

[補題１]

上の手続きでM1,M2の状態対<s_α,t_α>に対するΦ-到達可能性グラフが作成できれば，Φを満足する任意の全状態<s_α,t_α,$\mathbb{R}_n$,$\mathbb{V}_m$>からどのような通信系の動作を行っても，いつかはΦを満足する全状態に遷移する．

（証明） Φ-到達可能性グラフTの各ノードのレジス

タ計算式の変数X_n,Y_mにR_n,V_mを代入した木をT'とする．Tでは，各動作に対する遷移が完全に定義されているので，全状態<s_0,t_0,R_n,V_m>から順次通信系の動作を実行した時に遷移する全状態の列は，必ずT'の根から一つの葉への道上に書かれている．T中のすべての葉ノードがΦ復帰ノードであるので，<s_0,t_0, $_n$,V_m>からどのような通信系の動作を行っても，いつかはΦを満足する全状態に遷移する． ■

一般に，M1，M2及びその状態対<s_0,t_0>，P文Φ，自然数kが与えられた時，状態対<s_0,t_0>に対するΦ-到達可能性グラフでサイズがk以下のものがあるか否かは決定可能である．以下，状態対<s_0,t_0>に対するΦ-到達可能性グラフの葉ノード(Φ復帰ノード)<s,t,B_n, $_m$>の状態対<s,t>の集合をF(<s_0,t_0>,Φ)で表す．本論文では，次のようなエラーリカバリ性の検証手続きを提案する．

[命題2（エラーリカバリ性の検証手続き）]

M1，M2の初期状態と初期レジスタ値の組がΦを満足する時，M1,M2,P文Φ及び自然数kを与えて，次の手続きを行い，(3.2)で手続きを終了すれば，M1，M2はΦに対するエラーリカバリ性を持つ．

(1) S←{<s_1,t_1>}
　（s_1,t_1はM1，M2の初期状態．<s_1,t_1>はマークされていない）

(2) S中でマークされていない状態対<s,t>を見つけ，上述のΦ-到達可能性グラフの作成手続きを用いてk以下のサイズで<s,t>に対するΦ-到達可能性グラフが作成できるか否かを調べる．
　(2.1) k以下で作成できなければ終了(証明に失敗)．
　(2.2) k以下で作成できれば，<s,t>をマークし，

S←S∪{<s',t'>| <s',t'>∈F(<s,t>,Φ)，
　　且つ<s',t'>∉S}
　（但し，<s',t'>にはマークしない）

(3) S中でマークされていない状態対が存在するか否かを調べる．
　(3.1) 存在すれば，(2)へ．
　(3.2) 存在しなければ終了（証明に成功）．

（証明）(3.2)で手続きを終了した時の変数Sに初期状態の対が含まれること及び補題1よりあきらか． ■

例えば，図2.1，2.2のプロトコルマシンM1，M2において，Φとして2.4で述べた(i)の式を，kの値として8以上の自然数を与えて命題2の手続きを実行すると，(3.2)で手続きが終了し，エラーリカバリ性の証明に成功する．また，4つの状態対<s_1,t_1>,<s_2,t_2>,<s_4,t_1>,<s_4,t_2>が手続き終了時の変数Sに含まれる．尚，Φ-到達可能性グラフを生成する場合，最悪深さkの木が生成されるため，命題2のステップ(2)につき，一般にはO(C^k)回（k以外は固定と考える．Cは定数）条件(a)～(c)のようなプレスブルガー文の真偽判定を行わねばならない．

また，たとえM1，M2がΦに対するエラーリカバリ性を持つ場合でも命題2の方法で証明に失敗することもある．しかし，このような場合でも，M1，M2の全状態を引数とするような適当なP文Qを見つけて，M1，M2が（Φ and Q）に対するエラーリカバリ性を持つことを証明することが出来る場合がある．その時は，もちろん，M1，M2がΦに対するエラーリカバリ性を持つ．例えば，図2.1，2.2のプロトコルマシンM1，M2において，Φとして，

$\Phi(u_1, u_2, X_1, X_2, X_3, y_1)$

$= (X_2 = y_1)$ or $(u_1 = s_4)$ …(ii)

を指定し，kの値を十分大きく取って命題２の手続きを実行しても，k以下のサイズで初期状態対$<s_1, t_1>$に対するΦ-到達可能性グラフを作成できないので，証明に失敗する．しかし，QとしてM1のレジスタX_1とX_3の値は０以上であるという性質，

$Q(u_1, u_2, X_1, X_2, X_3, y_1)$

$= (X_1 \geqq 0)$ and $(X_3 \geqq 0)$ …(iii)

を指定し，命題２の手続きを用いて（Φ and Q）に対するエラーリカバリ性の証明を行うと証明に成功する．このように，QはΦに対するエラーリカバリ性をいわゆる並行帰納法を用いて証明する時の補題に相当する．尚，初期状態と初期レジスタ値の組がΦを満足しない場合でも，初期状態と初期レジスタ値の組から遷移可能な全状態を順次生成しながら初めてΦを満足する全状態の集合を求め，それらの全状態に含まれる状態対の集合を命題２の(1)の変数Sの初期値として与えれば，同様の方法で扱える．

２．６　検証例

2.5の命題２の手続きに相当するプログラムをワークステーションSun-3/260(メモリ 16MB)上で言語Ｃを用いて作成した．作成したプログラムでは，各状態対に対する到達可能性グラフを生成する際に同一のΣ式をノード名として持つノードを一つのノードにまとめてダグ（有向無閉路グラフ）の形で表現した．これにより，同一のプレスブルガー文の真偽判定の重複を防ぐことが出来る．尚，提案する方法でエラーリカバリ性

を検証する場合，$\forall z_1 \ldots z_t$(Ａ）の形のプレスブルガー文が真であることを証明しなければならない．

$\forall z_1 \ldots z_t$(Ａ）は，not$\exists z_1 \ldots z_t$(notＡ）（notＡが整数解を持たないこと）と等価である．notＡ は積和形展開することによって”A_1 or A_2 or … or A_k”（但し，各A_iは幾つかの線形不等式の論理積）に展開できる．すべてのA_iが整数解を持たなければnotＡも整数解を持たない．各A_iが整数解を持たないことの証明は，A_iの各線形不等式を整数線形計画問題の制約条件とみなして，その整数線形計画問題が整数解を持たないことを証明すればよい．この方法を用いて，$\forall z_1 \ldots z_t$(Ａ)が真であることを証明する．但し，A_iが整数解を持たないことを証明するのは一般には時間がかかると思われるので，作成したプログラムでは，シンプレックス法を用いてA_iが実数解を持たないことを証明している．この方法では，整数解の非存在性を直接証明する代わりに，十分条件として，実数解の非存在性を証明しているが，実際に行ったいくつかの検証例ではすべてこの方法で証明が成功した．

また，検証者が与えたΦに対して，M1，M2がΦに対するエラーリカバリ性を持つと証明できなかった場合でも，どの状態対に対するΦ-到達可能性グラフが構成できなかったかや，その状態対に対する深さｋの到達可能性グラフの内容を表示する．これらの情報から，仕様記述の誤りや，Φに対するエラーリカバリ性の証明を行う際に検証者が補題としてどのようなＰ文Ｑを指定すればよいか等に関する有益な情報が得られる．

検証例として，データリンクレベルにおけるHDLC手順（正規応答モード）の一次局と二次局に相当する２つのプロトコルマシンM1,M2 に対して，「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している，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」とき真であるようなＰ文Φを定め，作成したプログラムを用いて，M1,M2 がΦに対するエラーリカバリ性を持つかどうかを調べた．これらのプロトコルマシンはあるメーカーの作成した装置を今のモデルで記述したものであり，入力記号は一次局が15個，二次局が12個である．また，プロトコルマシンの状態数は各々12個ずつ，レジスタの数は一次局６個，二次局５個である．

一般に，HDLC手順には幾つかのパラメータ（同一シーケンス番号のＩ(情報)フレームの最大再送回数，相手局が受信したことの確認を受けずに送信できる最大Ｉフレーム数（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，これらの値は各製造者が独自に指定できるようになっている．これらのパラメータ値を大きくすると証明に要する時間も増大する．表２．１に幾つかのパラメータ値の組に対する検証結果を付す．尚，２．７では，以上述べた検証法を改良し，プロトコルが上述のようなパラメータ値をもつ場合に，パラメータ値に依存しない計算時間でエラーリカバリ性を機械的に検証する方法を述べる．

表２．１　命題２を用いた検証結果

	パラメータ値			
Iフレームの最大再送回数	1	2	3	4
ウインドウサイズ	1	2	3	4
タイムアウトの発生回数の上限	1	2	3	4
証明に要したＣＰＵ時間（分）	9	53	242	1372
ﾌﾞﾚｽﾌﾞﾙｶﾞｰ文判定回数（回）	8446	47952	254109	1280056
Φ-到達可能性ｸﾞﾗﾌの頂点の個数(個)	6172	34907	184749	930407
Φ-到達可能性ｸﾞﾗﾌのｻｲｽﾞの最大値	18	23	28	33

２．７　パラメータ値に依存しない検証法

図２．１のM1の初期状態S_1と図２．２のM2の初期状態t_1とからなる全状態において，通信系の動作[Ｄ，＊]が実行されたとする（図２．４参照）．このとき，M_1,M_2の状態は，それぞれ，S_2,t_1となり，M1のレジスタX_2の値には１が加えられる．この後，[Ｔ，＊]と[Ｄ，＊]がこの順に２回ずつ実行された後，さらに[Ｔ，＊]が実行されると，通信系はΦを満たす全状態に遷移する(図２．４)．さて，図２．１において，条件式"not($x_1 \geqq 2$)"をすべて"not($x_1 \geqq 100$)"で置き換えた通信系ＣＳ$_{100}$を考える．ＣＳ$_{100}$では，[Ｔ，＊]と[Ｄ，＊]がこの順に100回ずつ実行された後，さらに[Ｔ，＊]が実行されて，初めてΦを満たす全状態に遷移する．このように，パラメータ値"2"を"100"に変更したことにより，Φを満たす全状態に遷移するのに必要な[Ｔ，＊]と[Ｄ，＊]の実

行回数が増加するので，Φ-到達可能性グラフのサイズも図２．４に比較して非常に大きくなり，従って，検証に要する計算時間も膨大となる．

以下では，２．６で述べた検証手続きをこのようなパラメータ値に依存しない計算時間で実行できるように改良する．まず，Φ-到達可能性グラフに関する若干の定義を与える．Φ-到達可能性グラフにおいて，あるノードＮとその子孫であるノードＮ'について，ＮとＮ'の第１成分と第２成分同志が等しい（すなわち，ＮとＮ'の表す二つの全状態において，対応するプロトコルマシンの有限制御部の状態がそれぞれ等しい）と仮定する．このとき，ＮからＮ'へ至る道を，「疑似閉路」と呼び，ノードＮをその疑似閉路の始点，Ｎ'を終点と呼ぶ．もし疑似閉路Ｃが他の疑似閉路を含まないとき，Ｃを「単純疑似閉路」と呼ぶ．例えば，CS_{100}のΦ-到達可能性グラフにおいて，

$$\langle s_2, t_1, \langle x_1, x_2+1, x_3\rangle, \langle y_1\rangle\rangle$$
$$\rightarrow \langle s_3, t_1, \langle x_1+1, x_2+1, x_3\rangle, \langle y_1\rangle\rangle$$
$$\rightarrow \langle s_2, t_1, \langle x_1+1, x_2+1, x_3+1\rangle, \langle y_1\rangle\rangle \quad (7\text{-}1)$$

は単純疑似閉路である．Φ-到達可能性グラフに単純疑似閉路が存在するとき，Ｃ上の有向辺のラベルによって表される通信系の動作の系列が繰り返し実行される可能性がある．そのような動作系列を，「単純疑似閉路Ｃに対応する（通信系の）動作系列」と呼ぶ．例えば，動作系列[Ｔ，＊]，[Ｄ，＊]は，上述の単純疑似閉路(7-1)に対応する動作系列である．

ｅをΦ-到達可能性グラフの有向辺とする．ｅのラベルの表す通信系の動作が，各レジスタの値の内容を定数（０を含む）分だけ増加または減少させる動作であるとき，ｅを「定数更新辺」と呼ぶ．ある単純疑似閉路Ｃ

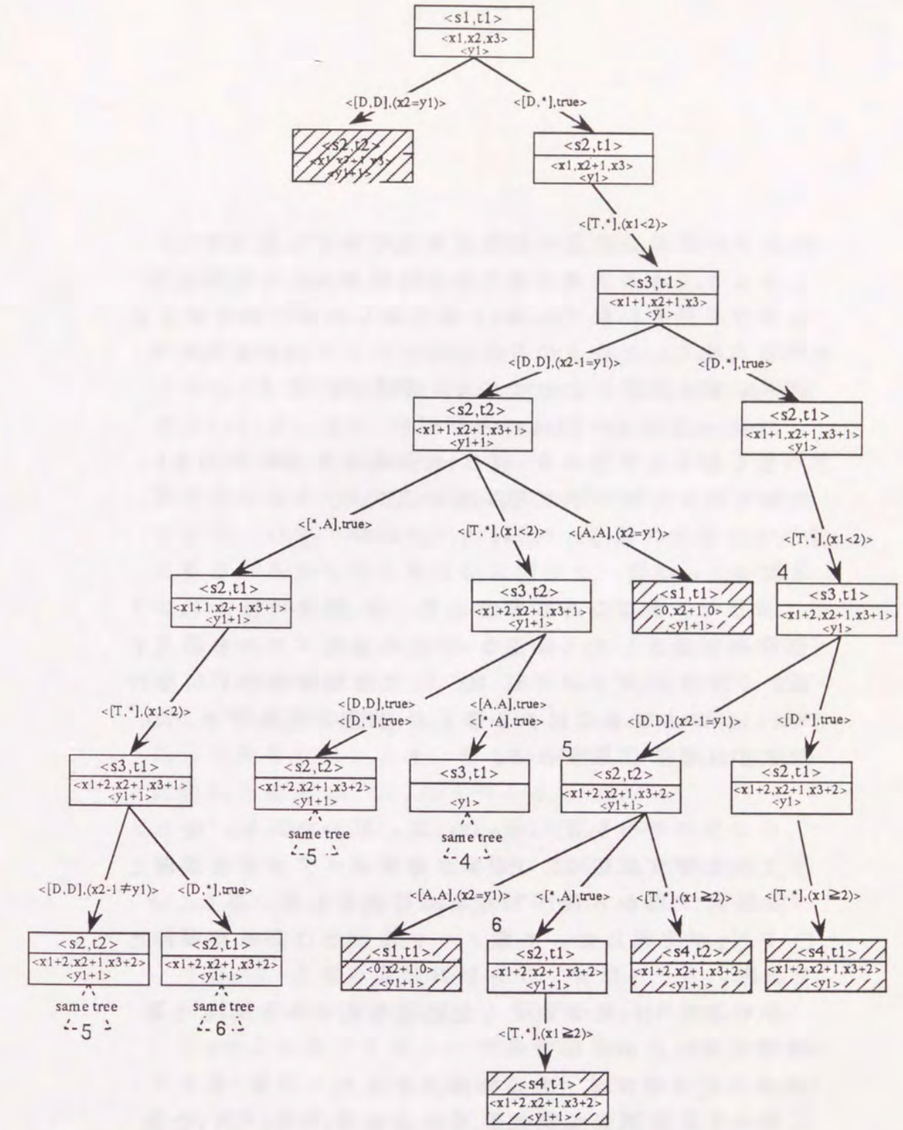


図2.4 M1, M2のΦ-到達可能性グラフ

上のすべての有向辺が定数更新辺であると仮定する．このとき，あるΣ式で表される全状態からCに対応する動作系列をk_1回（$k_1 \geqq 0$）繰り返した後の全状態を表すΣ式は，k_1とC上の有向辺のラベルの内容を用いて，容易に書き表すことができる．例えば，Σ式

$<s_2, t_1, <x_1, x_2+1, x_3>, <y_1>>$

で表される全状態から，(7-1)に対応する動作系列をk_1回繰り返した後の全状態を表すΣ式は，

$<s_2, t_1, <x_1+k_1, x_2+1, x_3+k_1>, <y_1>>$

となる．

以上のようなことを考慮して，Φ-到達可能性グラフの定義を変更して，修正Φ-到達可能性グラフを導入する．プロトコルマシンM1,M2,（有限制御部の）状態対$<s_0, t_0>$およびΦに対する修正Φ-到達可能性グラフGは次のように定義される．

Gの根のラベルは，$<s_0, t_0, \mathbb{X}_n, \mathbb{Y}_m>$である．他のノードおよび有向辺は，２．５の遷移グラフの定義を満たすように“根から葉へ”と順に定義される．但し，ノードN_2が定義に従って葉ノードとしてGに導入されたときに，

条件SCC“N_2を終点とし定数更新辺のみからなる単純疑似閉路Cが存在する”

が成り立つならば，次の操作Pを行う．

［操作P］単純疑似閉路Cの始点をN_1とし，N_1と追加しようとするノードN_2のラベルを，それぞれ，

$<s', t', \mathbb{B}_n, \mathbb{C}_m>$

$<s', t', \mathbb{B}_n+\mathbb{D}_n, \mathbb{C}_m+\mathbb{E}_m>$

とする．ここで，$\mathbb{B}_n, \mathbb{C}_m$は，一般に，$\mathbb{X}_n, \mathbb{Y}_m$を変数とするレジスタ計算式であり，$\mathbb{D}_n, \mathbb{E}_m$は整数の定数の

n字組，m字組である．このとき，新しい変数k_1と適当な整数の定数h_1を導入し，N_2のラベルを，

$<s', t', \mathbb{B}_n+k_1\mathbb{D}_n, \mathbb{C}_m+k_1\mathbb{E}_m>[h_1]$

に置き換える．このラベルは，ノードN_1が表す全状態において，単純疑似閉路Cに対応する通信系の動作系列が$k_1(1 \leqq k_1 \leqq h_1)$回実行された後の全状態を表す．

一般には，根からノードN_1への道の上には，既に定数更新辺のみからなる単純疑似閉路が存在する可能性がある．いま，根からノードN_1への道に含まれる定数更新辺のみからなる単純疑似閉路を，根から近い順に$C_1, \ldots, C_{q-1}$とし，それらに対応して導入した新しい変数を$k_1, \ldots, k_{q-1}$とする．このとき，上述の条件がノードN_2について成り立ち，しかも，N_2を終点とする新しい単純疑似閉路が，$C_1, \ldots, C_{q-1}$のいずれとも有向辺を共有しないとき，新しい変数k_qと適当な整数の定数h_qを導入し，N_2のラベルを，

$<s', t', \mathbb{B}_n+k_q\mathbb{D}_n, \mathbb{C}_m+k_q\mathbb{E}_m>[h_q]$

に置き換える．尚，以上の操作で導入した整数の定数$h_1, \ldots, h_q$を，それぞれ，閉路$C_1, \ldots, C_q$の「繰返し回数の上限」と呼ぶ．□

さらに，上述の変更に従って，復帰ノード，遷移する可能性のない枝の定義を，以下のように拡張する．

N_2をGにおける葉ノードとする（但し，以下で定義するΦ-復帰ノードでも繰返しノードでもないとする）．Eを，N_2を始点とし，２．５で述べた［遷移グラフの条件］を満たすすべての有向辺からなる集合とする．$e \in E$とし，eのラベルを

$<[\alpha 1, \beta 1], c(K_q, \mathbb{X}_n, \mathbb{Y}_m)>$

（但し，$K_q = <k_1, \ldots, k_q>$とする）

とする．有向辺eが次の条件(7-2)を満たすとき，eは（

繰返し回数の上限h_qについて）「遷移する可能性のない枝」と呼ばれ，そうでないとき，「遷移する可能性のある枝」と呼ばれる．

$$\forall K_q, \mathbb{X}_n, \mathbb{Y}_m$$
$$[\Phi(<s_0, t_0, \mathbb{X}_n, \mathbb{Y}_m>) \text{and} (1 \leqq k_1 + \ldots + k_{q-1} + k_q \leqq h_q) \supset \text{not}(c(K_q, \mathbb{X}_n, \mathbb{Y}_m))] \quad (7\text{-}2)$$

Eに属する遷移する可能性のある枝は，すべてグラフに付け加えられる．ここで，頂点N_2のラベルに記述されている繰返し回数の上限h_qは，それらの有向辺の終点にも書き込む．

このようにして付け加えられた有向辺の任意の終点をN_3とする．但し，N_3のラベルを，

$$<s, t, \mathbb{B}_n, \mathbb{C}_m>[h_q]$$

（$\mathbb{B}_n, \mathbb{C}_m$は，$K_q, \mathbb{X}_n, \mathbb{Y}_m$を変数とするレジスタ計算式）

とする．このとき，次の条件(7-3)が成り立つならば，N_3をΦ-復帰ノードと呼ぶ．

$$\forall K_q, \mathbb{X}_n, \mathbb{Y}_m$$
$$[\Phi(<s_0, t_0, \mathbb{X}_n, \mathbb{Y}_m>) \text{and} (1 \leqq k_1 + \ldots + k_{q-1} + k_q \leqq h_q) \supset \Phi(<s, t, \mathbb{B}_n, \mathbb{C}_m>)] \quad (7\text{-}3)$$

Φ-復帰ノードには，以降それを始点とする有向辺は付け加えない．

Gの根からノードN_2への道に存在する単純疑似閉路を，根に近い順に，$C_1, \ldots, C_q$とする．Gに有向辺e：$N_2 \rightarrow N_3$付け加えることにより，GにN_3を終点とする単純疑似閉路C_{q+1}が作られたと仮定する．次の(1)-(3)に場合分けして考える．

(1) C_{q+1}に定数更新辺でない有向辺が含まれているとき，この手続きは停止する．このとき，与えられたM1,M2,$<s_0, t_0>$とΦに対して，修正Φ-到達可能性グラフ

は定義されない．

(2) C_{q+1}に含まれる有向辺がすべて定数更新辺であり，C_{q+1}に対応する通信系の動作系列が，あるC_i($1 \leqq i \leqq q$)に対応する通信系の動作系列と一致するとき，ノードN_3を，「繰返しノード」と呼ぶ．Φ-復帰ノードの場合と同様，繰返しノードには，それを始点とする有向辺は付け加えない．

(3) C_{q+1}に含まれる有向辺がすべて定数更新辺であり，C_{q+1}に対応する通信系の動作系列が，どのC_i($1 \leqq i \leqq q$)に対応する通信系の動作系列とも一致しないとき，N_3のラベルを，前述の［操作P］で述べたラベルに置き換える．

このようにしてGに付け加えたノードN_3が，Φ-復帰ノードでも繰返しノードでもない場合，以上の手続きを繰り返す．

上述の手続きを実行して，もうそれ以上付け加えるノードがなくなったならば，この手続きは停止する．このときに得られたGが，次の条件(A),(B)をともに満たすならば，Gを，与えられたM1,M2,$<s_0, t_0>$とΦに対する修正Φ-到達可能性グラフであると定義し，そうでなければ，修正Φ-到達可能性グラフは定義されない．

(A) Gのすべての葉ノードは，繰返しノードかまたはΦ-復帰ノードである．

(B) Gにおける任意の単純疑似閉路Cについて，Cに次の条件(7-4)を満たす有向辺u→vが含まれる．

$$\forall K_q, \mathbb{X}_n, \mathbb{Y}_m$$
$$[\Phi(<s_0, t_0, \mathbb{X}_n, \mathbb{Y}_m>) \text{and} (h_q \leqq k_1 + \ldots + k_{q-1} + k_q) \supset \text{not}(c(K_q, \mathbb{X}_n, \mathbb{Y}_m))] \quad (7\text{-}4)$$

但し，有向辺u→vのラベルを，

$<[\alpha 1, \beta 1], c(K_q, X_n, Y_m)>$

とし，h_qを，ノードuに記述されている繰返し回数の上限とする．

いま，M1とM2からなる通信系が全状態$<s_\alpha, t_\alpha, R_n, V_m>$にあるとし，$<s_\alpha, t_\alpha, R_n, V_m>$がΦを満たすと仮定する．もし，$<s_\alpha, t_\alpha>$に対して，修正Φ-到達可能性グラフが存在するならば，通信系は，いつかはΦを満足する全状態に遷移する．従って，２．５の命題２の手続きにおいてΦ-到達可能性グラフを修正Φ-到達可能性グラフに置き換えて得られる手続きは，Φに対するエラーリカバリ性の検証に用いることができる．この手続きの計算時間は，本節冒頭で述べたパラメータ値に依存しない．さらに，修正Φ-到達可能性グラフの定義からこの手続きはいつかは必ず停止するので，グラフのサイズの上限kを指定する必要はない．但し，定数更新辺のみからなる各単純疑似閉路C_i(i=1,2,...)について，繰返し回数の上限$h_1, h_2, ...$を検証者が与える必要がある．

図２．４に対応する修正Φ-到達可能性グラフを図２．５に示す．前述の単純疑似閉路(7-1)は，定数更新辺のみからなるので，その終点のラベル

$<s_2, t_1, <x_1+1, x_2+1, x_3+1>, <y_1>>$　　は，

$<s_2, t_1, <x_1+k_1, x_2+1, x_3+k_1>, <y_1>>$[2]

に置き換えられる．但し，検証者によって，繰返し回数の上限は２と指定されている．さらに，このノードを始点とする次のような単純疑似閉路が構成される．

$<s_2, t_1, <x_1+k_1, x_2+1, x_3+k_1>, <y_1>>$

$\rightarrow <s_3, t_1, <x_1+k_1+1, x_2+1, x_3+k_1>, <y_1>>$

$\rightarrow <s_2, t_1, <x_1+(k_1+1), x_2+1, x_3+(k_1+1)>, <y_1>>$　　(7-5)

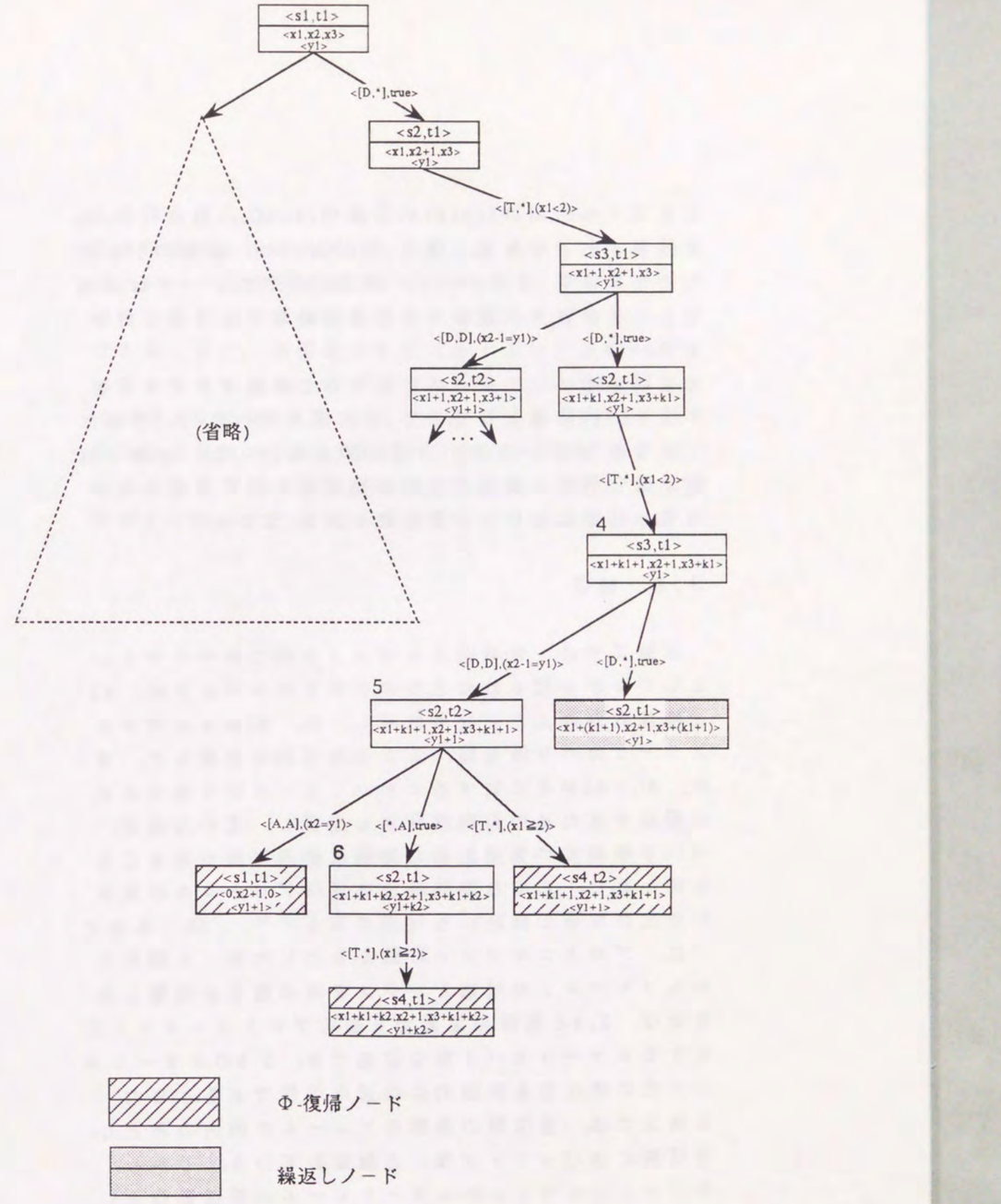


図2.5 修正Φ-到達可能性グラフ

ここで，$<s_2, t_1, <x_1+(k_1+1), x_2+1, x_3+(k_1+1)>, <y_1>>$は，繰返しノードである．図２．５において，斜線の引かれたノードはΦ-復帰ノード，網掛けされたノードは繰返しノードを表す．遷移する可能性のない枝は書いていない．

本節で述べたアイデアに基づいて検証プログラムを作成し，検証例として，２．６末尾で述べたHDLC手順についてΦ-エラーリカバリ性の検証を行った．命題２の検証法を用いた場合と本節の検証法を用いた場合について，検証に要した計算時間を表２．２に示す．

２．８　結言

本論文では，有限個のレジスタを持つオートマトンとしてモデル化された２つのプロトコルマシンM1，M2と検証者が与えたＰ文Φに対し，M1，M2がΦに対するエラーリカバリ性を持つことを形式的に定義した．また，M1，M2がΦに対するエラーリカバリ性を持つことを保証するための自動検証法を提案し，その方法が，HDLC手順程度の実用規模の問題に適用可能であることを確かめた．この自動検証法は通信プロトコルの安全性や生存性等の検証にも利用できる[13]．尚，本論文では，プロトコルマシンの数を２としたが，ｎ個のプロトコルマシンの状態とレジスタ値の組を全状態とみなせば，2.4と同様の定義でｎ個のプロトコルマシンに対するエラーリカバリ性を定義でき，2.5のエラーリカバリ性の検証法も原理的には適用可能である．また，本論文では，通信路の異常をフレームの消失のみとし，通信路にはバッファがないと仮定している．しかし，各プロトコルマシンにエラーフレームの受信動作という動作を導入し，M1の送信動作とM2のエラーフレームの受信動作を同時に実行することによって通信路上でのフレームの変形を表すとみなせば，フレームの変形が起こるような通信路に対しても提案する方法が適用できる．また，通信路が有界（サイズｋ）のFIFOキューとみなせる場合，各プロトコルマシンに新たにｋ個のレジスタを導入し，それらのレジスタでFIFOキューの内容を保持すれば，提案する検証法が適用できる．

尚、本章で述べた検証法が実用規模の問題にどの程度適用可能かどうかや，その評価が今後の課題である．

表２．２　検証に要した計算時間の比較

(a) 命題２を用いた検証結果

	パラメータ値			
Iフレームの最大再送回数	1	2	3	4
ウインドウサイズ	1	2	3	4
タイムアウトの発生回数の上限	1	2	3	4
証明に要したＣＰＵ時間（分）	9	53	242	1372
ﾌﾞﾚｽﾌﾞﾙｶﾞｰ文判定回数（回）	8446	47952	254109	1280056
Φ-到達可能性ｸﾞﾗﾌの頂点の個数(個)	6172	34907	184749	930407
Φ-到達可能性ｸﾞﾗﾌのｻｲｽﾞの最大値	18	23	28	33

(b) ２．７の方法を用いた検証結果

	ﾊﾟﾗﾒｰﾀ値
Iフレームの最大再送回数	8
ウインドウサイズ	7
タイムアウトの発生回数の上限	8
証明に要したＣＰＵ時間（分）	270
ﾌﾞﾚｽﾌﾞﾙｶﾞｰ文判定回数（回）	96590
Φ-到達可能性ｸﾞﾗﾌの頂点の個数(個)	73776
Φ-到達可能性ｸﾞﾗﾌのｻｲｽﾞの最大値	29

第３章

通信プロトコルの代数的仕様からプログラムへの変換

3.1 序言

代数的仕様記述法[17]は，厳密な意味の定義が簡明であり，検証の際に必要な仕様の無矛盾性や詳細化等の概念も，合同関係という概念のみを用いて簡明に定義できる．代数的仕様の部分クラスとして，抽象的順序機械の形で記述された仕様[18],[19]がある．この仕様では，論理型，整数型等の基本タイプを前提として，状態の構造を陽にはもたない抽象的状態を表すタイプを定義する．関数には状態遷移関数，状態成分関数（抽象的状態からその成分の値を取り出す関数），および補助関数があり，各状態遷移後の状態成分関数の値を公理によって定義する．抽象的状態は構造を陽にはもたないことから，新たに状態成分を追加する際，既に記述した部分を変更することなく，状態成分関数に関する構文と公理を局所的に追加するだけでよい，という特長がある．また，既にHDLC，ＯＳＩセション層等の通信プロトコルやスクリーンエディタの仕様を抽象的順序機械の形で記述し，仕様の形式的検証が行われ，本記述法の有効性が実証されている[19]-[21]．さらに，自然語による要求定義から抽象的順序機械の形で記述された仕様を導出する方法論も検討されている[22]．

本章では，抽象的順序機械の形で記述された通信プロトコルの代数的仕様からプログラムへの変換について述べる．まず3.2で，関連する諸定義と仕様の例を示す．3.3では，文献(2),(3)の方針に従って，与えられた仕様から効率の良い手続き型プログラムへの変換法を述べ，目的プログラムを解釈実行する機械を定義する代数的仕様，および変換を定義する代数的仕様を示す．さらに3.4では，この変換法に従って，与えられた仕様をＣプログラムに変換するコンパイラについて述べる．

3.2 抽象的順序機械

3.2.1 代数的仕様

仕様記述には，言語ASL/*[23]を用いる．ASL/*では，始記号を指定しない文脈自由文法Gと公理の集合AXの組 t=(G,AX) を仕様と呼ぶ．Gの非終端記号Aから生成される終端記号列全体からなる集合を$L_G(A)$，Gのいずれかの非終端記号から生成される終端記号列全体の集合をE_tと書き，E_tの元をtの表現式と呼ぶ．タイプ名と，そのタイプの元を表す表現式を生成する非終端記号を同一視して，$L_G(A)$に属する表現式を，タイプAの表現式と呼ぶ．また，生成規則$A \rightarrow f(A_1,A_2,...,A_n)$ （A, $A_1,A_2,...,A_n$は非終端記号，f，括弧"("，")"およびコンマ","は終端記号）が記述されているとき，終端記号fを関数名とみなし，関数fの引数のタイプは$A_1,A_2,...,A_n$であり，関数値のタイプはAであるという．

論理型のtrue,false，任意の非負整数の10進表現0,1,2,...等のように，それ自身が値を表すと考えられる表現式を構成子表現式と呼ぶ．構成子表現式の集合は，それを生成する非終端記号の部分集合Vcを指定することにより，$\bigcup_{A \in Vc} L_G(A)$と定義する．

公理とは変数を含む表現式の対 Ω ==r である．但し，公理に現れる各変数xにはGの非終端記号Mxが一つ割り当てられているとし，変数xのタイプはMxであるという．公理 Ω ==rは，Ωまたはrに現れる各変数に，そのタイ

プである非終端記号から生成される任意の表現式を代入して得られる終端記号列の対が，仕様tにおいて合同となることを表す．形式的には，AXのすべての公理を満たすE_t上の最小の合同関係[23]を仕様tの指定する合同関係といい，$\equiv_t$で表す．$e \equiv_t e'$が成り立つとき，表現式eとe'は仕様tにおいて合同であるという．

3.2.2 抽象的順序機械の定義

仕様tが次の条件(1)～(3)をすべて満たすとき，tを抽象的順序機械の形で記述された仕様と呼ぶ．

(1) tの文法は，抽象的状態を表す表現式を生成する非終端記号（stateとする）をもつ．

(2) tの関数は，(2a)状態遷移関数（関数値のタイプはstate），
(2b)状態成分関数（第1引数のタイプはstate，関数値のタイプは基本タイプ），および(2c)補助関数（関数値のタイプは基本タイプ）に分類される．これらはいずれも，タイプがstateである引数を二つ以上もってはならない．簡単のため，タイプがstateの引数をもつとき，それは第1引数であるとする．状態成分関数Oは，タイプがstateの引数が表す抽象的状態において，Oの表す特定の状態成分(そのタイプは，例えば整数や配列等，状態成分ごとに定められた任意の基本タイプ)の値を表す．以下(2a),(2b)をそれぞれ単に遷移関数，成分関数と呼ぶ．

(3)tの公理は次の条件(3a)～(3c)をすべて満たす．
(3a)公理の左辺は線形で，かつ重なりをもたない[24]．
(3b)公理の右辺に現れる変数は，その左辺にも現れる．
(3c)公理は次の(i)～(iii)のいずれかの形をしている．

これらを順にi～iii型公理と呼ぶ．

(i) $O(T(x_1,x_2,...,x_n),u_2,...,u_m)$ == ...
(ii) $T'(x_1,x_2,...,x_n)$ == ...
(iii) $A(x_1,x_2,...,x_n)$ == ...

但し，T,T'は遷移関数，Oは成分関数，Aは補助関数，$x_1,x_2,...,x_n$は変数，$u_2,...,u_m$は変数か構成子表現式である．i,iii型公理の右辺に遷移関数が現れてはならない．i,ii型公理の左辺に現れる遷移関数（条件(3a)よりi,ii型両方の公理の左辺に現れることはない）を，それぞれI,II型の（状態）遷移関数と呼ぶ． □

条件(3c)より，i,iii型公理の右辺に遷移関数は現れないので，状態遷移後の成分関数の値は，遷移前の状態における成分関数と補助関数の値と，遷移関数のstate以外のタイプの引数のみに依存して定まる．

抽象的順序機械の形で記述された仕様の例として，ＯＳＩセションプロトコル[25]を取り上げる．セション層の通信主体SPM（セションプロトコルマシン）の仕様t_{SPM}[20],[29]の一部を表3.1に示す．但し，表現式の構文を定める文法は省略している（以下でも同様）．MAP(s,spdu)は，SPMが状態sにおいて相手SPMの送信したMAPという種類のデータ(spduはその内容を表す)を受信したときの動作を表すII型の遷移関数であり，[1]は，MAP(s,spdu)の値が，そのときのSPMのフェーズ(成分関数phase(s)で表される)に従って定まるI型の遷移関数の値と合同となることを表すii型公理である．[2]～[6]は，I型の遷移関数MAP1による遷移後の成分関数の値を定義したi型公理である．例えば[3]は，遷移MAP1後のV[M](SPMの一つの内部変数の内容を表す成分関数)の値が遷移前の値に1加えたものに等しいことを表す．[7],[8]は補助関数p12,p19を定義するiii型公理である．

表３．１　ＳＰＭの仕様 t_{SPM}(部分)

```
MAP(s, spdu) ==
   if phase(s) = idle&TCcon then err1(s)
   else if phase(s) = data_trans then
      if ¬ SSYNM(s) ∧ p12(s) ∧ p19(s, spdu)
      then MAP1(s, spdu) else err0(s)
   else if phase(s) = abort then s
   else err0(s)                                    [1]
V[A](MAP1(s, spdu))
   == if Vsc(s) then V[A](s) else V[M](s)          [2]
V[M](MAP1(s, spdu)) == V[M](s) + 1                 [3]
SSYNM(MAP1(s, spdu))     == true                   [4]
phase(MAP1(s, spdu))     == phase(s)               [5]
OWNED(MAP1(s, spdu), x) == OWNED(s, x)             [6]
p12(s) == Ac(s, dk) ∧ Ac(s, mi) ∧ AAc(s, ma)       [7]
p19(s, spdu)   == sn(spdu)=V[M](s)                 [8]
                  ...
```

3.3　プログラムへの変換法

ここでは，抽象的順序機械の形で記述された仕様からプログラムへの変換法を述べる．変換法を形式的に記述するため，以下の(a)～(d)の仕様を考える．
(a)抽象的順序機械の形で記述された入力仕様t_{ASM}．
(b)目的プログラムを解釈実行する機械OMを定義する仕様t_{OM}（3.3.1参照）．
(c)入力仕様t_{ASM}に対する目的プログラムを定義する仕様t_{COMP}（3.3.2.1参照）．t_{COMP}は，t_{ASM}を引数としてそれを実現する目的プログラムを値とする関数を定義するため，仕様t_{ASM}やその中の関数名等を表現式として扱う．以下では表現式としての仕様は[t_{ASM}]と書き，仕様そのものと区別する．関数名についても同様である．従って厳密にはt_{COMP}において入力仕様[t_{ASM}]の構

文を定義しなければならないが，以下では省略している．
(d)t_{ASM}の各関数をt_{OM}とt_{COMP}の各関数によってどのように実現するかを指定する仕様CRP(t_{ASM})(3.3.2.1参照)．CRP(t_{ASM})は，t_{ASM}の関数名等に依存して定まる．
t_{ASM}に対して対応関係を指定する仕様CRP(t_{ASM})を与える写像CRPを，対応関係の指定と呼ぶ．

3.3.1　目的機械OM

目的プログラムの構文を定義する生成規則（これらは次に述べる仕様t_{OM}=(G_{OM}, AX_{OM})の文法G_{OM}の生成規則の集合に含まれる）を表3.2に示す．プログラム変数は$V_1, V_2, \ldots$であり，プログラムは変数への代入文とIF文の有限系列である．厳密には，各プログラム変数はタイプをもち，それに従って正しい構文をもつプログラムが定義されるが，簡単のため，その詳細は省略している．また，表3.2の非終端記号programから生成される表現式の部分系列に対して，それらの連接を表す関数が定義されており，表3.2の生成規則の右辺において，隣接する各終端記号や非終端記号の間には，連接を表す関数が挿入されていると考えるが，表では省略している．t_{COMP}で扱う入力仕様の構文についても同様である．

OMは，抽象的順序機械の形で記述された仕様t_{OM}によって定義される．t_{OM}は表3.3に示すような関数をもつ．OMの状態s_{OM}におけるプログラム変数V_j（$j \geqq 1$）の値は，同じ名前の成分関数$V_j(s_{OM})$で表される．t_{OM}を表3.4に示す．公理[11],[12]はそれぞれ，代入文，IF文から始

表３．２　目的プログラムの構文

program → ε | statement ';' program
statement → 'IF' expression 'THEN' program 'ELSE' program | variable ':=' expression
expression → variable | constant | '¬' expression | expression '+' expression | . . .
variable → 'V_1' | 'V_2' | . . .
constant → 'TRUE' | 'FALSE' | '0' | '1' | . . .

まるプログラムの実行を定義する．関数EXECは，本来は，

```
EXEC(sOM,prog')
  == if HEAD(prog') = ε  then sOM
     else HEAD(prog') = 'Vj:='exp then
          EXEC(Assign(sOM,Vj,exp),TAIL(prog'))
               ...
```

のように，ii型公理で定義されている．ここで，HEAD(prog')およびTAIL(prog')は，それぞれ，プログラムprog'の先頭の文（但し，prog'に含まれる文が０個ならε）および，prog'からHEAD(prog')を取り除いたプログラムを表す関数である．仕様の読みやすさのため，表3.4では，[10]～[12]のように左辺で場合分けをしている．他の関数も同様である．[13]～[14]は代入文の実行を定義している．[15]～[18]は，プログラム中の式の値を評価する関数EVALを，式の構造に基づいて再帰的に定義している．@は基本タイプの演算子を表す．但し，OMにおける基本演算はすべて全域関数とする．

表３．３　仕様tOMの関数

状態成分関数	
$V_1(s_{OM}), V_2(s_{OM}), \ldots$	それぞれ，OMの状態s_{OM}におけるプログラム変数$V_1, V_2, \ldots$の内容を表す．
状態遷移関数	
INIT_OM	OMの初期状態を表す．
EXEC(s_{OM}, prog)	状態s_{OM}においてプログラムprogを実行した後の状態を表す．
Assign(s_{OM}, V_j, exp)	状態s_{OM}において式expを評価した値を変数V_jに代入した後の状態を表す．
補助関数	
EVAL(s_{OM}, exp)	状態s_{OM}において式expを評価した値を表す．

表３．４　目的機械ＯＭの仕様 t_{OM}

V_i(INIT_OM) == 0 [9]
EXEC(s_{OM}, ε) == s_{OM} [10]
EXEC(s_{OM}, 'V_j :=' exp ';' prog)
  == EXEC(Assign(s_{OM}, j, exp), prog) [11]
EXEC(s_{OM},
  'IF' exp 'THEN' $prog_1$ 'ELSE' $prog_2$ ';' prog)
  == if EVAL(s_{OM}, exp)
    then EXEC(EXEC(s_{OM}, $prog_1$), prog)
    else EXEC(EXEC(s_{OM}, $prog_2$), prog) [12]
V_j(Assign(s_{OM}, V_j, exp)) == EVAL(s_{OM}, exp) [13]
V_i(Assign(s_{OM}, V_j, exp)) == $V_i(s_{OM})$ (i≠j) [14]
EVAL(s_{OM}, '@' exp) == @ EVAL(s_{OM}, exp) [15]
EVAL(s_{OM}, exp_1 '@' exp_2)
  == EVAL(s_{OM}, exp_1) @ EVAL(s_{OM}, exp_2) [16]
EVAL(s_{OM}, 'V_i') == $V_i(s_{OM})$ [17]
EVAL(s_{OM}, 'c') == c [18]

3.3.2 変換法の概要

3.3.2.1 変換の基本方針

簡単のため，入力仕様t_{ASM}は，以下の条件(1)～(3)をすべて満たす抽象的順序機械の形で記述された任意の代数的仕様とする．一般の場合については**3.3.2.2**で述べる．t_{ASM}で定義される抽象的順序機械（ASMと呼ぶ）の状態を表すタイプをstateASMとする．

(1)t_{ASM}の成分関数と遷移関数は，タイプがstateASM以外の引数をもたない．

(2)公理はすべてi型である．

(3)公理の右辺において，if関数は，右辺全体または他のthen部またはelse部全体にしか現れない．

文献(2),(3)の方針に従い，以下のように変換を行う．O_j（$1 \leqq j \leqq p$）をt_{ASM}の成分関数，T_i（$1 \leqq i \leqq q$）を遷移関数とする．目的プログラムは，各遷移関数T_iを実現する以下のようなプログラムOBJ([t_{ASM}],[T_i])の集合である．OBJ([t_{ASM}],[T_i])は，プログラム変数として，成分関数O_jの値を保持する変数W_jと，退避のための変数R_jを用いる．但し，OMの変数名は$V_1, V_2, \ldots$なので（**3.3.1**参照），各j（$1 \leqq j \leqq p$）について，W_jはV_{2j-1}，R_jはV_{2j}で表すと考え（厳密には，成分関数の関数値と同じタイプをもつプログラム変数を用いなければならない），以降，W_j，R_jを用いて説明する．

以上のように，t_{ASM}の関数O_jをOMのプログラム変数W_jで，また，関数T_iをOBJ([t_{ASM}],[T_i])で実現することを指定するのが，仕様CRP(t_{ASM})である（表3.5）．表では省略しているが，CRP(t_{ASM})の生成規則は，t_{ASM}の生成規則すべてと，表3.5で参照しているt_{OM},t_{COM}

表3.5　対応関係を指定する仕様ＣＲＰ(t_{ASM})

$$O_j(map(s_{OM})) == W_j(s_{OM}) \quad [19]$$
$$T_k == map(EXEC(INIT\_OM, OBJ([t_{ASM}], [T_k]))) \quad [20]$$
$$T_i(map(s_{OM})) == map(EXEC(s_{OM}, OBJ([t_{ASM}], [T_i]))) \quad [21]$$

pの関数に関する生成規則とからなる．CRP(t_{ASM})では，この対応関係を指定するために，関数mapを導入する．map(s_{OM})は，OMの状態s_{OM}が実現しているASMの状態を表す．公理[19]は，“OMの状態s_{OM}の実現しているASMの状態map(s_{OM})における関数O_jの値は，s_{OM}における関数W_jの値（変数W_jの値）と合同である”ことを表す．[21]は，１引数の（タイプがstateASMの引数のみをもつ）遷移関数T_iについて，“状態map(s_{OM})において状態遷移T_iが起こった後のASMの状態は，s_{OM}においてOBJ([t_{ASM}], [T_i])を実行した後のOMの状態の実現しているASMの状態と合同である”ことを表している．引数をもたない遷移関数（初期状態を表す関数）についても同様である（[20]）．これを図示すると，図3.1のようになる．破線の矢印は，t_{ASM}の関数がOMや目的プログラムによってどのように実現されるかを表す．

変換の正しさは，t_{COMP}とt_{OM}の関数を用いてt_{ASM}の関数を実現できることを示すことにより保証されるが，CRP(t_{ASM})は，このような関数間の対応関係を仕様で陽に記述したものである（この対応関係のもとでt_{ASM}の関数が正しく実現されることの形式的な証明については，文献(29)の**4**.を参照されたい）．

各プログラムOBJ([t_{ASM}],[T_i])（$1 \leqq i \leqq q$）の内容は，

状態遷移T_i後の各成分関数O_j(1≦j≦p)の値の変数W_jへの代入文の系列である．例えば，表3.6の仕様は表3.7に変換される．i型公理の右辺に遷移関数は現れないので（3.2.2の定義参照），右辺に現れる成分関数は，状態s_{ASM}での（遷移前の）値を表している．従って，公理の右辺からこのような代入文を容易に生成できる．但し，OBJ([t_{ASM}],[T_i])では，まずW_jの値を局所変数R_jに退避し，代入文右辺では退避変数の方を参照するようにする．なぜならば，もし退避を行わないとすると，状態遷移前後の成分関数の値の区別がプログラムでは行えず，正しく計算が行えないからである．

以上の方法でt_{ASM}を実現する目的プログラムを定義する仕様t_{COMP}を表3.8に示す．t_{ASM}の遷移関数T_iを実現するOBJ([t_{ASM}],[T_i])は，公理[22]より，comp(S_of_ax([t_{ASM}],[T_i]),#0([t_{ASM}]))である．ここで，S_of_ax([t_{ASM}],[T_i])は，t_{ASM}においてT_iが左辺に現れるi型公理の系列を表す表現式であり，また，#0([t_{ASM}])は，t_{ASM}の成分関数の個数を表す．これらの関数を定義する公理は表3.8では省略している．comp(s_of_ax,p)は，reserve(p)と$comp_{s_of_ax}$(s_of_ax)の連接からなるプログラムである（[23]）．ここで，reserve(p)は，各j（1≦j≦p）について順に，W_jの値をR_jに退避する代入文の系列であり（[24],[25]），$comp_{s\_of\_ax}$(s_of_ax)は，i型公理の系列s_of_axに属する各公理'$O_j(T_i(s_{ASM}))==r_{ij}$'あるいは'$O_j(T_i)==r_{ij}$'に対して，状態遷移$T_i$後の$O_j$の値$r_{ij}$を退避変数$R_k$（1≦k≦p）を用いて計算して$W_j$に代入する文の系列である([26]～[35]）．[28]～[35]において，c_exp_ax,c_exp_1_ax,c_exp_2_axは，t_{ASM}の公理右辺に現れる（一般にifを含む）表現式を表す変数，exp_ax,exp_1_ax,exp_2_axは，ifを含まない表現式を表す変数である．$comp_{c_exp}$や$comp_{exp}$も，表3.4と同様，仕様の読みやすさのため，引数の場合分けを左辺で行っている．$comp_{ax}$(ax)は公理axに対応する一つの文を，$comp_{c\_exp}$(j,c_exp_ax)はc_exp_axに対応するIF文を，$comp_{exp}$(exp_ax)はexp_axに対応するプログラムの式を表す．

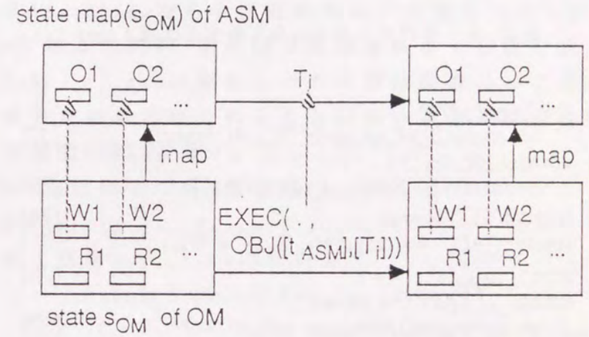


図３．１　ＯＭとＯＢＪ([t_{ASM}],[T_i])によるＡＳＭの実現

表３．６　入力仕様t_{ASM}の例

$O_1(T_1(s))$ == $O_1(s) + O_2(s)$;
$O_2(T_1(s))$ == $O_3(s)$;
$O_3(T_1(s))$ == $O_2(s) + O_3(s)$;

表3.7　表3.6を実現する目的プログラム

R_1 := W_1;
R_2 := W_2;
R_3 := W_3;

W_1 := $R_1 + R_2$;
W_2 := R_3;
W_3 := $R_2 + R_3$;

表3.8　目的プログラムを定義する仕様 t COMP

OBJ(spec, 'Ti')
  == comp(S_of_ax(spec, 'Ti'), #O(spec))　[22]
comp(s_of_ax, j)
  == reserve(j) $comp_{s\_of\_ax}$(s_of_ax)　[23]
reserve(0)　== ε　[24]
reserve(j+1)　== reserve(j) 'Rj+1 := Wj+1 ;'　[25]
$comp_{s\_of\_ax}$(ε)　== ε　[26]
$comp_{s\_of\_ax}$(ax ';' s_of_ax)
  == $comp_{ax}$(ax) $comp_{s\_of\_ax}$(s_of_ax)　[27]
$comp_{ax}$('Oj(Ti(sASM)) ==' c_exp_ax)
  == $comp_{c\_exp}$(j, c_exp_ax)　[28]
$comp_{ax}$('Oj(Ti) ==' c_exp_ax)
  == $comp_{c\_exp}$(j, c_exp_ax)　[29]
$comp_{c\_exp}$(j,
  'if' exp_ax 'then' c_exp_1_ax 'else' c_exp_2_ax) ==
'IF' $comp_{exp}$(exp_ax) 'THEN' $comp_{c\_exp}$(j, c_exp_1_ax)
'ELSE' $comp_{c\_exp}$(j, c_exp_2_ax) ';'　[30]
$comp_{c\_exp}$(j, exp_ax)
  == 'Wj :=' $comp_{exp}$(exp_ax) ';'　[31]
$comp_{exp}$('@' exp_ax)　== '@' $comp_{exp}$(exp_ax)　[32]
$comp_{exp}$(exp_1_ax '@' exp_2_ax)
  == $comp_{exp}$(exp_1_ax) '@' $comp_{exp}$(exp_2_ax)　[33]
$comp_{exp}$('Oj(sASM)') == 'Rj'　[34]
$comp_{exp}$('c')　== 'c'　[35]

3.3.2.2　一般の場合の変換法

(a)i型公理O(T(x_1,x_2,...,x_n),u_2,...,u_m) == ...において，遷移関数，成分関数の引数のうち，タイプがstateASM以外のものの扱いを考える．

成分関数Oの第1引数以外の引数のタイプを順にA_2,...,A_m（m≧2）とし，これらのタイプの値（構成子表現式）が有限個しか存在しないと仮定する．Oは，タイプがA_2,...,A_mである任意の構成子表現式の組<c_2,...,c_m>に対して次のi型公理で定義される1引数の成分関数O<c_2,...,c_m>の組を，一つの成分関数としてまとめて表すために用いられることが多い（例えば，表3.1の成分関数OWNED参照）．

“Oを左辺にもつ各i型公理
O(T(x_1,x_2,...,x_n),u_2,...,u_m)== r,
および各<c_2,...,c_m>に対して，
O<c_2,...,c_m>(T(x_1,x_2,...,x_n))
== r{u_2/c_2,...,u_m/c_m}”

（$\gamma\{\alpha_1/\beta_1,...,\alpha_n/\beta_n\}$は，系列$\gamma$の互いに重なり合わない部分系列$\alpha_1$,...,$\alpha_n$を，系列$\beta_1$,...,$\beta_n$に置き換えて$\gamma$から得られる系列を表す）

いま，第i引数（2≦i≦m）がk_i個の値を取り得ると仮定する．プログラムでは，成分関数Oの値を保持する変数として，i-1番目の添字の範囲が1～k_iであるm-1次元の配列を用い，各構成子表現式の組<c_2,...,c_m>に対して配列の一つの要素を割り当て，O(s,c_2,...,c_m)の値をその要素内に保持する．

もし第1引数以外の引数のタイプが，値を無限個取り得るとすると，上のように添字の範囲が有限の配列を用いて実現することはできない．例えばリスト構造

を用いて公理による式の書換えを直接行うことにより実現が可能ではあるが，実行効率は著しく低下する．

遷移関数Tの引数$x_1,\ldots,x_n$については，OBJ([t_{ASM}],[T_i])を，$x_1,\ldots,x_n$に対応する引数をもつ手続きとして実現すればよい．

(b)ii，iii型公理は，左辺の関数の引数がすべて変数であり，いわゆる関数型プログラム[26]の形をしているので，各公理に対応する手続きを導入し，公理右辺に現れる基本タイプの関数を対応する基本演算に，遷移関数，補助関数を対応する手続き呼出しに，成分関数を変数の参照にそれぞれ変換すればよい．

これらの拡張を可能にするため，3.3.1で述べた仕様t_{OM}に，配列に関する基本演算と，手続き呼出しの機能を付け加え，手続きの引数によってt_{ASM}の遷移関数や補助関数のstateASM以外の引数を実現することをCRPで指定する必要があるが，容易であるので省略する．

なお，変数R_jへの値の退避はすべての成分関数や補助関数の値に対して必要なわけではない．例えば，表3.7では変数W_1,W_3の値の退避は不要である．退避すべき値の個数が最小となる更新順を決める問題は，レジスタ割付けの問題として知られている[27]．3.4で述べるコンパイラでは，待避する値の個数の“少ない”更新順を選ぶ（特に，値の待避が全く不要な更新順が存在する場合は，そのうちの一つが選ばれる）．

3.4 コンパイラの実現と変換例

3.4.1 プロトコルの代数的仕様のコンパイラ

3.3.2.2で述べた一般の場合の変換法に従い，C言語のプログラムを生成するコンパイラをUNIX上で実現した．以下では，その概要について述べる．

3.4.1.1 コンパイラへの入力

本コンパイラは，通信プロトコルの記述を行うことを考慮して，基本データタイプに①整数型②論理型③スカラ型④構造体型の４つを用意している．特に，スカラ型はトークン等の名前の集合，構造体型は送受信データ等を定義するのに有用である．

構造体型には，定数として，そのフィールドの値がすべて未定義の構造体を表すnull，基本演算として，構造体rのフィールドfの値をvに変更した構造体を表すput(r,f,v)，構造体rのフィールドfの値を表すget(r,f)がある．

3.4.1.2 基本データタイプの実現

整数型，論理型，スカラ型はC言語のint型で実現する．define文により論理型の定数trueおよびfalseにそれぞれ1と0，スカラ型を構成する各定数に異なる一つの整数を割り当てる．

構造体型はC言語の構造体を用いて実現する．また，構造体型の定数nullをdefine文により0と定義する．

3.4.1.3 公理から代入文への変換

基本的には，3.3.2.2で述べた変換法に従って変換を行う．Cプログラムの代入文右辺では，公理右辺の状態成分関数が変数の値の参照に，整数型・論理型の関数がC言語の整数演算・論理演算に，関数ifがif文に相当する．また，関数putは構造体のフィールドへの代入文に，関数getは構造体のフィールドの参照に相当する．但し，putの引数がnullの場合，もし必要ならば，値の代入の前に関数mallocにより構造体の領域を確保する．

3.4.1.4 状態成分関数の値の更新順序

状態遷移に伴う状態成分関数の値の更新の際に，どの状態成分関数から値を更新するかによって，待避すべき値の個数が異なる場合がある．生成されるプログラムの効率を考えると待避する値の個数が少なくなるように，更新の順番を決めることが望ましい．ここでは，状態成分関数の値の更新順を決定するために，本コンパイラが用いている方法の概要を述べる．以下では，コンパイラの入力となる仕様に補助関数は存在しないとする．以下の議論はそれが存在する場合に容易に拡張できる．

現有コンパイラでは，各状態遷移関数T_iに対して，次の条件(1),(2)をともに満たす有向グラフG_iを作成する．G_iは状態遷移T_iの際の状態成分関数間の依存関係を表している．

(1)状態成分関数の集合と有向グラフG_iの頂点の集合の間には1対1の対応がついており，有向グラフG_iの各

頂点は対応する状態成分関数名をラベルとしてもつ．

(2)状態遷移T_i後の状態成分関数0_jの値を定義するi型の公理の右辺に状態成分関数$0_{j'}$（但し$j \neq j'$）が現れているときかつそのときのみ，0_jをラベルにもつ頂点から$0_{j'}$をラベルにもつ頂点への有向枝が存在する．

有向グラフG_iにおいて，入射枝をもつ頂点に対応する状態成分関数$0_{j'}$はその有向枝を出射枝としてもつ頂点に対応する状態成分関数0_jの値の定義に用いられているので，$0_{j'}$の更新は0_jの更新より後に行うかまたは$0_{j'}$の値を待避するかしなければならない．コンパイラでは，上記の性質を利用して，以下のように更新順を決める．

(a) まず，入射枝が一つもない頂点を任意に1つ選び，対応する関数の値を更新する代入文を生成する．また，その頂点とその頂点から出ている枝をすべて取り除く．以降，全ての頂点が取り除かれるか，入射枝を持たない頂点がなくなるまでこれを繰り返す．もし，全ての頂点が取り除かれれば終了．さもなければ次の(b)を行う．

(b) どの頂点も少なくとも1つ入射枝を持つならば，適当に頂点を選び（本コンパイラでは出射枝の数が最も多い頂点を選ぶことにしている），その関数について待避変数を作る．以後，この関数の値の参照は待避変数の方で行う．そして，その関数の値の更新を行う代入文を生成し，その頂点とその出射枝，入射枝をすべて取り除く．以上を行った後，(a)を実行する．

3.4.2 変換例

3.2.2で述べた仕様t_{SPM}を入力例として，本コンパイラの生成する目的プログラムの効率等について述べる．

図3.2に，コンパイラへの実際の入力仕様のうち，I1型の遷移関数MAPとI型の遷移関数MAP1に関する部分を示す（表3.1に対応）．図において，for each "α" in {"β_1",…,"β_m"} "γ"は，γに現れるαをβ_i（$1 \leqq i \leqq m$）に置き換えて得られる系列$\gamma\{\alpha/\beta_1\}\cdots\gamma\{\alpha/\beta_m\}$を表すマクロ記法である．コンパイラによって生成されたCプログラムのうち，MAPおよびMAP1に対応する部分を，それぞれ図3.3の(a),(b)に示す．送受信データを表すタイプは，構造体型として定義されており，これらを表す成分関数（SSP1～SSP4，TSP，SPDU1～SPDU5）の値は，構造体型の変数に代入される．

図3.3(b)で，(1)状態遷移の前後で値が変化しない成分関数（phaseやOWNED）については不要な代入文は生成されておらず，また，(2)成分関数の値の待避が全く不要な値の更新順（3.3の末尾参照）のうちの一つが選択され，対応するプログラムが生成されている．このような最適化により，本例題では，生成された目的プログラムは，人手で作成した手続き型プログラムとほぼ同程度の効率で実行が可能である．


```
MAP(s,pa_MAP) ==
   if phase(s) = idle_and_TCcon then err1(s)
   else if phase(s) = data_trans then
      if not(SSYNM(s)) and p12(s) and
               p19(s,get(pa_MAP,sn))
                  then MAP1(s,pa_MAP)
      else err0(s)
   else if phase(s) = abort then s
   else err0(s);

for each "X" in {"phase" "Vsc" "Vtca" "lc"}
"X(MAP1(s,pa_MAP)) == X(s);"
for each "X" in {"SSYNM" "awaitSSYNMrsp"}
"X(MAP1(s,pa_MAP)) == true;"
Vm(MAP1(s,pa_MAP)) == Vm(s)+1;
Va(MAP1(s,pa_MAP)) ==
   if Vsc(s) then Va(s) else Vm(s);
for each "x" in {"mi" "ma" "dk"}
"OWNED(MAP1(s,pa_MAP),x) == OWNED(s,x);"
for each "x" in {"FD" "HD" "SY" "MA"}
"FU(MAP1(s,pa_MAP),x) == FU(s,x);"
SSP4(MAP1(s,pa_MAP)) ==
  put(put(put(null,name,_SSYNMind),
         sn,get(pa_MAP,sn)),
      data,get(pa_MAP,data));
for each "X" in {"SSP1" "SSP2" "SSP3" "TSP"
   "SPDU1" "SPDU2" "SPDU3" "SPDU4" "SPDU5"}
"X(MAP1(s,pa_MAP)) == null;"
```

図3.2　コンパイラへの入力仕様(一部)

```
MAP(pa_MAP)
pdu5 *pa_MAP;
{
  if (phase == idle_and_TCcon)
  {
    err1();
  }
  else
  {
    if (phase == data_trans)
    {
      if (!(SSYNM) && p12() &&
                    p19(pa_MAP->sn))
      {
        MAP1(pa_MAP);
      }
      else
      {
        err0();
      }
    }
    else
    {
      if (phase == abort)
      {
      }
      else
      {
        err0();
      }
    }
  }
}
```

(a)

```
MAP1(pa_MAP)
spdu5 *pa_MAP;
{
  SSYNM = TRUE;
  awaitSSYNMrsp = TRUE;
  if (Vsc)
  {
  }
  else
  {
    Va = Vm;
  }
  if (SSP4 == NULL)
    SSP4 = (ssp4 *)malloc(sizeof(ssp4));
  SSP4->name = _SSYNMind;
  SSP4->sn = pa_MAP->sn;
  SSP4->data = pa_MAP->data;
  SSP1 = NULL;
  SSP2 = NULL;
  SSP3 = NULL;
  TSP = NULL;
  SPDU1 = NULL;
  SPDU2 = NULL;
  SPDU3 = NULL;
  SPDU4 = NULL;
  SPDU5 = NULL;
  Vm = Vm + 1;
}
```

(b)

図3.3　生成されたCプログラム

3.5　結言

抽象的順序機械の形で記述された通信プロトコルの代数的仕様から手続き型プログラムへの変換法の形式的記述と，その方法に従って作成されたコンパイラについて述べた．今後，通信プロトコル以外の具体的な例題について本章で示した方法を適用すると共に，コンパイラの機能の拡張と改善を図る予定である．

結 論

本論文は，通信プロトコルの等価性及びエラーリカバリー性の検証とプログラムへの変換に関する研究をまとめたものである．

(1) 等価性の検証に関しては，２つのプロトコルマシンM_1，M_2間のΨ等価性を定義し，それらがΨ等価であるかどうかを判定する手続き与えた．
また，その手続きを実際の装置例に適用し，等価性（従ってM_1，M_2の「互換性」）の証明において本手続きが有効であることを確かめた．

(2) エラーリカバリー性の検証に関しては，等価性の検証に用いたモデルを適用して，２つのプロトコルマシンM_1，M_2が検証者が与えたＰ文Φに対して，エラーリカバリー性を持つ事を保証するための自動検証法を提案した．
また，その方法がＨＤＬＣ手順程度の実用規程の問題に適用可能であることを確かめた．
なお，この自動検証法は通信プロトコルの安全性や生存性の検証にも利用できる．本論文で述べた検証法がどの程度まで実用規程の問題に適用可能かどうかやその評価が今後の課題である．

(3) 通信プロトコルのプログラムへの変換に関しては，抽象的順序機械の形で記述されたプロトコルの代数的仕様から手続き型プログラムへの変換法とその方法に従って作成されたコンパイラについて述べた．
今後，プロトコル以外の具体的問題についてここで

示した方法を適用すると共に，コンパイラの機能の拡張と改善を図る予定である．

謝辞

本研究に関して，直接理解ある御指導を賜り，さらに論文提出の機会を与えて頂いた大阪大学基礎工学部情報工学科の谷口健一教授に心から感謝申し上げます．

また，情報工学科の嵩忠雄教授には筆者が大阪大学基礎工学部大学院に在学中より御指導を賜り，本研究の全過程を通じてつねに励まして頂きました．ここに心より感謝の意を表します．

本研究に関して，終始，適切な御指導と御助言を頂いた東野輝夫講師並びに関浩之助手に心から感謝します．

本論文の作成に際し、有益な御助言を頂いた大阪大学情報工学科 都倉信樹教授，宮原秀夫教授，大阪大学教養部の藤井護教授，並びに滋賀大学情報管理学科の森将豪助教授に心から感謝します．

また，通信プロトコルの実現法とテスト系列に関して種々御教示頂いた慶応大学 松下温教授並びに沖電気工業㈱の新田哲二氏に心から感謝します．

さらに，等価性，エラーリカバリー性の検証に際し検証手続きに相当するプログラム作成を担当頂いた大学院生（現在 住友電気工業㈱ 勤務）の木本智久氏に感謝します．

文　献

(1) 野口, 斉藤, 白鳥: ”分散処理とコミュニケーション”, ソフトウェア工学ハンドブック（榎本編）オーム社, pp177-221(1986).
(2) J.E.Hopcroft and J.D.Ullman:"Introduction to Automata Theory, Language, and Computation", Addison-Wesley, (1979).
2-計数機械の定義, 及び, 2-計数機械が ﾁｭｰﾘﾝｸﾞ機械と等価であることについてはpp.171-173を, ﾁｭｰﾘﾝｸﾞ・機械の等価性が判定不能であることについてはp.150及びp.281を参照. 訳書（野崎他訳）”オートマトン 言語理論 計算論I,II”, サイエンス社(1984) では上記ページは各々巻Iのpp.224-226, 及び巻1のp.196, 巻IIのp.82に相当.
(3) 木本, 東野, 谷口, 二宮: ”あるクラスの通信制御装置のモデル化と等価性判定について”, 言語理論とオートマトンシンポジウム(1987-07).
(4) 木本智久: ”あるクラスの通信制御装置のモデル化と等価性判定”, 大阪大学基礎工学部情報工学科特別研究報告(1987-03).
(5) 竹内外史: ”数学基礎論の世界”, 日本評論社 (1972).
(6) 東野, 工藤, 縄田, 杉山, 谷口: ”代数的仕様検証支援系及びそれを用いた検証例”,信学論(D), **J67**-D,4,pp.472-479(昭59-04).
(7) "IBM Binary Synchronous Communications", IBM N:GA27-3004-1(1969).
(8) ”IBM 同期データ・リンク制御概説書”, IBM N:GA27-3093-0(1975).
(9) ”SDLC（２次局）プロトコル仕様書”, 沖電気工業㈱ (1979).

(10)野口, 斉藤, 白鳥: ”分散処理とコミュニケーション”, ソフトウェア工学ハンドブック（榎本編）オーム社, pp177-221(1986).
(11)J.E.Hopcroft and J.D.Ullman:"Introduction to Automata Theory, Language, and Computation", Addison-Wesley,(1979).
(12)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”, 信学技報IN88-80(昭63-09).
(13)東野,谷口:"整数線形計画問題の解非存在性判定を利用した通信プロトコルの自動検証について”, 京都大学数理解析研究所講究録695, pp.243-252 (平01-06).
(14)”JIS ハイレベルデータリンク制御手順”, JIS C 6363-6365(昭53-11).
(15)木本 智久: ”通信プロトコルにおけるエラーリカバリ性の自動検証”, 大阪大学大学院基礎工学研究科修士学位論文(平01-02).
(16)加藤, 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自動検証の一方式”, 第39情処全大,4T-1, pp.1961-1962 (平01-10).
(17)稲垣康善, 坂部俊樹:"抽象データタイプの代数的仕様記述法の基礎", 情報処理, **25** (昭59).
(18)杉山裕二, 谷口健一, 嵩忠雄:"基底代数を前提とする代数的仕様", 信学論(D), **J64-D**, 4, pp.324-331 (昭56-04).

(19)T. Higashino, M. Mori, Y. Sugiyama, K. Taniguchi and T.Kasami: "An algebraic specification of HDLC procedures and its verification", IEEE Trans. Software Eng., SE-10, 6, pp.825-836 (1984).
(20)粟屋英司，遠一光，関浩之，藤井護，嵩忠雄:"ＯＳＩセション層の代数的記述とその検証"，信学論(D-I)，J72-D-I，4，pp.272-283（平1-04).
(21)田中哲雄，谷口健一，奥井順:"スクリーンエディタの代数的仕様記述とその実現"，信学論(D)，J71-D，7，pp.1207-1217(昭63-07).
(22)K.Torii, Y.Morisawa, Y.Sugiyama and T.Kasami :"Functional programming and logical programming for the telegram analysis problem", Proc. of the 7th Intl.Conf. on Software Eng., pp.463-472 (1984).
(23)嵩忠雄，谷口健一，杉山裕二，関浩之:"代数的言語ASL/* －意味定義を中心に－"，信学論(D)，J69-D，7，pp.1066-1074 (昭61-07).
(24)二木厚吉，外山芳人:"項書き換え型計算モデルとその応用"，情報処理，24，2，pp.133-146(昭58-02).
(25)ISO: "Basic connection oriented session protocol specification", ISO 8327.
(26)井上克郎，関浩之，谷口健一，嵩忠雄:"関数型言語ASL/Fとその最適化コンパイラ"，信学論(D)，J67-D，4，pp.458-465（昭59-04).
(27)A.V.Aho, R.Sethi and J.D.Ullman: "Compilers, Principles,Techniques, and Tools", Addison Wesley (1986).

(28)犬伏健二:"代数的に記述された通信プロトコルからプログラムへの変換システムの試作"，大阪大学基礎工学部特別研究報告 (1988-03).
(29)遠一光，粟屋英司，関浩之，藤井護，二宮清:"抽象的順序機械の形で記述された代数的仕様からプログラムへの変換について"，信学論(D-I)，J73-D-I，2，pp.201-213（平2-02).
(30)ISO: "LOTOS, a formal description technique based on the temporal ordering of observational behaviour", DP 8807.
(31)ISO : "Estelle, a formal description technique based on an extended state transition model", DP 9074.
(32)G.V. Bochmann, G.W. Gerber and J.M. Serre: "Semiautomatic implementation of communication protocols", IEEE Trans. Software Eng., SE-13, 9, pp.989-1000 (1987).
(33)J.P.Briand, M.C.Fehri, L.Logrippo and A.Obaid:"Executing LOTOS specifications", Protocol Specification, Testing, and Verification, VI, pp.73-84, North-Holland (1987).
(34)加藤聰彦，長谷川亨，堀内浩規，鈴木健二:"EstelleからAdaへのトランスレータ"，信学技報，IN86-97 (1986-11).
(35)高橋薫，白鳥則郎，野口正一:"LOTOS解釈機構の構成"，信学技報，IN87-107 (1988-01).