Vol. 36 No. 1 情報処理学会論文誌 Jan. 1995

リンクの故障を考慮に入れた分散システムの動作仕様の自動導出

岡野浩三† 今城広志†
東野輝夫† 谷口健一†

分散システム全体の外部から観測できる動作の内容や実行順を記述した要求仕様から要求仕様どおりに動作する各ノードの動作仕様（どのようなタイミングでどのノードとどのような同期用メッセージやレジスタ値を交換しながら自ノードの動作を実行したり，自ノードのレジスタ値を更新していくべきかを記述したもの）を自動生成できることが望ましい．一方，分散システムではノードやリンク故障を考慮して，これらの故障が生じても，システム全体として要求仕様どおり動くシステムの設計が望まれる．本論文ではレジスタを持つ拡張有限状態機械(EFSM)としてモデル化された分散システム全体の要求仕様から，要求仕様どおりに動作する各ノードの動作仕様 (EFSM) を自動生成するための一つのアルゴリズムを提案する．この動作仕様では，高々一つのリンク故障に対して，同一メッセージを異なる2経路で送受信することにより，エラー回避が行われる．その際，単純に二重化せず同一リンクで送信される複数のメッセージを単一のメッセージに統合して，その際のノード間のメッセージ交換の総数をできるだけ少なくしている．メッセージ総数の最小化のために0-1整数線形計画問題の解法を用いている．

Synthesis of Protocol Specifications from Service Specifications in Distributed Systems with Communication Link Errors

Kozo Okano,† Hiroshi Imajo,† Teruo Higashino† and Kenichi Taniguchi†

In a distributed system, the protocol entities must exchange some data values and synchronization messages in order to ensure the temporal ordering of the events described in a service specification for the distributed system. It is desirable that a correct protocol specification can be derived automatically from a given service specification. In this paper, we propose an algorithm which synthesizes automatically a correct protocol specification from a service specification described as an extended FSM (EFSM) model. In our model, message loss is considered as the communication link errors. We assume that at most one communication link may be down. Even if all the messages in the faulty link are lost, the derived protocol specification can simulate the service specification correctly. To avoid the influences of the communication errors, we use duplicated messages with the same information. In our method, however, in order to reduce the number of exchanged messages at each transition, all the messages used in the same link at each phase are merged into one message. To minimize the number of the exchanged messages, we use a procedure to calculate an optimal solution for 0-1 integer linear programming problems.

1. はじめに

近年，信頼性の高い分散システムを設計するための方法についての研究が盛んに行われている．抽象レベルでは，分散システムの外部から観測できるデータの入出力動作（処理）の実行順序の記述をそのシステムの要求仕様とみなすことができる[1]．与えられた要求仕様に対し，分散システム上の各ノードはお互いに同期用メッセージやデータを交換しながら要求仕様で指定された順に動作を実行しなければならない．各ノードがどのような順に同期用メッセージやデータを交換しながら動作を実行していくべきかを記述したものを各ノードの動作仕様と呼ぶ．分散システムの設計法として，設計者は要求仕様のみを記述し，その記述から

† 大阪大学基礎工学部情報工学科
Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University

各ノードの動作仕様を自動生成できることが望ましい．このようなプロトコル合成に関して多くの研究がなされている[1)~5),9)]．一方，通信ネットワークのリンク，ノード等に故障を仮定して，それに対しても全体としてシステムの動作が保証されるフォールトトレランスに関する研究も多くなされている．リンク故障に関しては，代替経路を保証するために，例えばルーティングを複数用意しておき，それらから動的に適応可能なルーティングを選択する方法などが知られている．また，プロトコル合成の際に，エラーリカバリ性を持たすような研究も報告されている[1),2),5)]．一般に，動的な経路制御は複雑であり，CPU 時間，通信路等の多くの資源を要する[7)]．また，下位層のプロトコルなどでは acknowledge を用いて経路の状態を調べる方法が一般的に用いられているが，この方法ではタイムアウト機構が必要である．一般にタイムアウトの時間は各動作時間に比べてきわめて大きな値が設定されるため，タイムアウト機構が働くと全体の処理効率が著しく低下する．このため，アプリケーションのレベルではこのような制御をせず，最初からメッセージを多重化して送信する方法が考えられる．

我々は，文献9)で有限個のレジスタを持つ拡張有限状態機械 (Extended Finite State Machine (EFSM)) でモデル化された要求仕様から，要求仕様どおりに動作する各ノードの動作仕様を自動生成する一つのアルゴリズムを考案した．このモデルでは，データベースやファイルなどを内部変数として抽象化したものを，一般にレジスタとして扱っている．そこではリンク故障のない理想的なネットワークを仮定している．しかし多くのノードからなる分散システムのすべてのリンクにまったく故障が生じないという仮定は非現実的であり，実用的な観点からは故障が発生しても正しく動作する動作仕様を生成することが重要である．

そこで本論文では，高々1個所のリンクの故障が発生しても要求仕様どおりに動くようにメッセージ交換を行う動作仕様を導出する．耐故障性のため，同一メッセージを異なる2経路で送受信するが，単純に二重化したのでは，メッセージ交換の回数が多くなる．そこで，メッセージ交換に用いるリンクの数をできるだけ少なくしたり，同一リンクを同じタイミングで送られる複数のメッセージを一つに統合すること等により，提案する模倣方針のもとで，各状態遷移でのノード間の送受信動作の総数が最小になるように各ノードの動作仕様を導出する．最小解を求めるために 0-1 整数線形計画問題の解法を用いている．

以下2章で EFSM モデルを形式的に定義し，3章でその動作を定義する．また対象とする故障のモデルについて述べる．4章で各ノードの動作仕様を自動生成するためのアルゴリズムなどについて述べる．

2. EFSM モデル

分散システム全体の要求仕様 SS（サービス仕様[1)]とも呼ばれる）および各ノードの動作仕様 PE を次のような EFSM モデルで記述する．

EFSM は FSM に有限個のレジスタを加えたものである．EFSM の状態遷移図は**図1**のように表す．この図で節点は状態を表し，枝は状態遷移を表す．状態遷移を表す枝には遷移条件，入出力動作，レジスタ更新式の3字組がラベルとして付けられている．入力動作 $a?x$ でゲート a からのデータ x の入力を表し，出力動作 $a!E(...)$ で式 $E(...)$ の値をゲート a に出力することを表す．E の引数は，レジスタが許される．入出力を何も行わないときは内部動作 i を行うものとする．便宜上，内部動作 i は要求仕様では用いない．初期状態では各レジスタの初期値が指定される．ある状態において，その状態から出る各状態遷移の実行可能性を判定する述語が遷移条件である．遷移条件の評価には現状態のレジスタ値と入力変数値が用いられる．これが真となる状態遷移から非決定的に一つの遷移が選択され，入出力動作が実行される．次にレジスタ値が更新され，次状態に遷移する．レジスタ値の更新を行う代入文の組がレジスタ更新式である．レジスタ値の更新は，各代入文の右辺（引数はレジスタと入力変数）の値を同時に評価した後，その評価値で行う．例えば，s_2 から s_3 の遷移において，遷移条件は，$r_2>0$ と与えられている．状態 s_2 でのレジスタ r_1, r_2, r_3 の値がそれぞれ，3, 5, 10 であった場合，この遷移

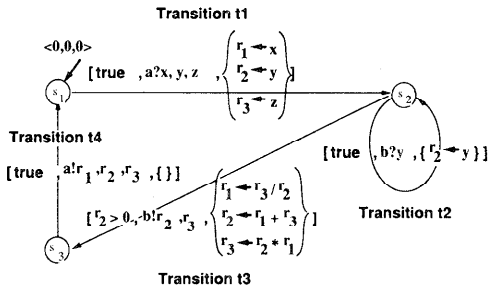


図 1　要求仕様
Fig. 1　Service specification.

は実行可能となり，もしこれが選択されると，ゲート b から 5, 10 が出力され，状態 s_3 でのレジスタ r_1, r_2, r_3 の値はそれぞれ，2(=10/5), 13(=3+10), 15(=5×3) となる．

3. 分散環境での動作仕様の動作

各動作仕様の動作環境をここで述べる．各ノードの動作仕様も上述の EFSM モデルで記述する．各ノード k の動作仕様を PE_k で表す．p 個のノードの動作仕様の組を $\langle PE_1, \cdots, PE_p \rangle$（または，$PE^{1-p}$ と略記）で表し（p ノードの）分散システムの動作仕様（あるいは，プロトコル仕様[1]）と呼ぶ．

分散システム全体の要求仕様 SS に対し，m 個のレジスタおよび入出力ゲートがそれぞれどのノードに属するかをユーザが指定し，それらの割当を Alloc(SS) で表す．各入出力ゲートは必ず一つのノードに属すると仮定するが，同一レジスタが複数ノードに属してもよい（**図 2**）．

3.1 通信環境の仮定

PE_i から PE_j への通信路（リンク）は無限容量を持つ信頼性の保証されていないリンク ($buff_{ij}$) で結ばれているとし，両端のゲート名を g_{ij} とする*．

リンクの故障はメッセージ消失のみとし，以下の性質を仮定する．

- もし，リンクが故障していなければ，そのリンクは FIFO キューとして動作する．
- リンクが故障状態になれば，そのリンク内のすべてのメッセージは消失する．また，故障状態のリンクに入ったメッセージも消失する．

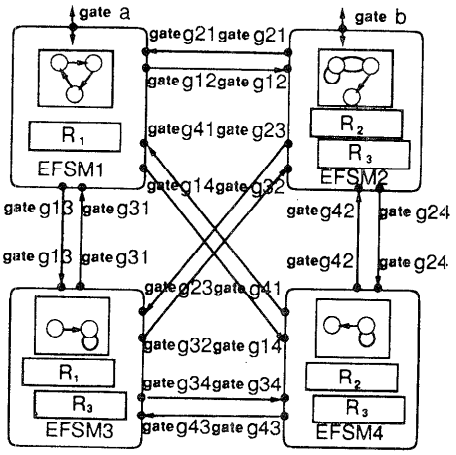


図 2 PE と通信モデル
Fig. 2 Communication model.

- 任意の時点で高々 1 個所のリンクしか故障状態にならない．
- あるリンクが故障状態から復帰すれば，そこからある単位期間内はいずれのリンクも故障しない．

ここでは，メッセージの消失のみを取り扱う．伝送路の雑音によりメッセージの一部が欠落，変質することも，実際問題として起こりうる．この場合，メッセージにチェックサムをつけることにより，データの一部変質や欠落は検出可能である．もし，異なるメッセージを受信した場合，どちらのメッセージが間違ったものなのかはチェックサムを見ることにより判定できる．これにより，間違ったチェックサムの方を破棄してメッセージの消失として取り扱えばよい．もちろん，両方のチェックサムが間違っていた場合は，両方のメッセージが消失したものとみなすので，本稿の仮定である高々一つのリンクエラーという仮定を満たさなくなる．

なお，リンク中のメッセージの最大伝送処理時間，レジスタ更新に要する最大時間をそれぞれ，Δ, δ とし，その他動作に要する時間を無視できるものと仮定し，$4\Delta+\delta$ 時間を上述の単位時間（すなわち，リンク故障から次のリンク故障までの最低時間）とする．この値は要求仕様の一つの状態遷移を（p 個の）全ノードで実行するのに必要な時間の最大値に相当する．詳細については，4 章で述べる．

3.2 分散 EFSM モデルの等価性

分散システムの動作仕様 PE^{1-p} において，各 PE_k $(1 \leq k \leq p)$ が初期状態でかつ，通信路の各リンクがすべて空であるときを，PE^{1-p} の初期状態とする．分散システム全体の要求仕様 SS と，分散システムの動作仕様 PE^{1-p} が等価であることを以下のように定義する．

［等価性］

分散システムの動作仕様 PE^{1-p} において，各 PE_i, PE_j 間の通信に用いられる送受信動作 $g_{ij}?x, g_{ij}!E(\cdots)$ と，内部動作 i を，観測不可能な動作とし，その他の動作を観測可能な動作とする．このとき SS と PE^{1-p} が観測合同[8]であれば，両者は等価とする． □

q と w が観測合同というのは，q で実行できる任意の観測可能な動作の系列が w でも実行でき，かつ任意の実行可能な観測可能動作系列を行った時点での実行可能かつ観測可能な動作が互いに等しいこと，か

* i から j へのリンクと，j から i へのリンクは区別する．

つ，それが q と w を入れ替えても成り立つことである．

3.3 分散 EFSM 動作仕様の導出問題

[導出問題]

要求仕様 SS と分散システムの p 個のノードおよび，各ゲートやレジスタの割当 Alloc(SS) が与えられたとき，上記の通信環境で，SS と等価な p ノードの分散システムの動作仕様 PE^{1-p} を導出する問題． □

ただし，要求仕様として与えられる SS および，Alloc(SS) は次のような制約条件を満足するものとする．

1. 与えられた SS の各状態 s について，次が成り立つこと．状態 s において二つの動作 "a…" と "b…" (a, b はゲート名) が記述されていたら，ゲート "a" と "b" の属するノードは同一でなければならない．
2. 初期状態を始点とする各状態遷移 t について以下が成り立つこと．t の遷移条件 C(…) は，t の動作 a… に使われるゲート "a" が属するノードに属するレジスタのみを用いた述語でなければならない．また，その動作 a… が出力動作なら，引数とするレジスタもゲート "a" が属するノードに属していなければならない．
3. ノードの数 p は3以上であること．
4. Alloc (SS) において各レジスタは2個以上のノードに分散配置されていること． □

制約条件1を外すと，各状態について動作を実行するノードが複数存在することになる．これを解決する方法として，これらのノード間で二重にトークンメッセージをまわす方法が考えられる[10]．ここでは簡単のためこの制約条件をつけておく．制約条件2がない場合，新たに初期状態を設け，その初期状態からもとの初期状態への状態遷移を設けることにより，本手法では解決できるので本質的な制約条件ではない．制約条件 3, 4 は次章で述べる本手法に必要である．これらの条件が導出の際にどのように用いられるかは 4.1, 4.6 節で述べる．

3.4 表記法

上の制約条件1より，各状態 s_i で動作の実行可能判定と実行は一つのノードで行われる．このノードを Snode(s_i) で表し，(状態 s_i の) 責任ノードと呼ぶ．

また各レジスタ r_h の属するノードの集合を Rnode (r_h) で表す．さらに，状態 s_i から始まる状態遷移の集合について，(イ)動作の実行可能性を判定する遷移条件の引数に指定されたレジスタ名，(ロ)出力で用いられるレジスタ名，および (ハ)Snode(s_i) に属するレジスタ名の集合和を Cset(s_i) で表す．Cset(s_i) は状態 s_i において Snode(s_i) が知っている (べき)，レジスタ名の集合である．

例えば，要求仕様 SS (図1) と以下の割当 Alloc (SS) (図2) に対して以下のようになる．

	ノード1	ノード2	ノード3	ノード4
レジスタ	r_1	r_2, r_3	r_1, r_3	r_2, r_3
ゲート	a	b		

Snode　Snode(s_1)≡ノード 1, Snode(s_2)≡ノード 2, Snode(s_3)≡ノード 1,

Rnode　Rnode(r_1)≡{ノード 1, ノード 3},
Rnode(r_2)≡{ノード 2, ノード 4},
Rnode(r_3)≡{ノード 2, ノード 3, ノード 4}

Cset　Cset(s_1)≡{r_1}, Cset(s_2)≡{r_2, r_3},
Cset(s_3)≡{r_1, r_2, r_3} □

3.5 動作例

要求仕様 SS (図1) と，割当 Alloc(SS) (図2) が与えられたとき，一つの状態遷移 $s_2-[r_2>0, b!r_2, r_3, \{r_1 \leftarrow r_3/r_2, r_2 \leftarrow r_1+r_3, r_3 \leftarrow r_2 \times r_1\}]\rightarrow s_3$ に対する各動作仕様 $PE_1, \cdots, PE_4$ の一つの可能なタイミングチャートを**図3**に表す．図3において，まず PE_2 が出力動作 $b!r_2, r_3$ を選択し，実行する．ステージ(I)では，レジスタ更新のために必要なメッセージの送受信が各 PE_k 間で行われ，ステージ(II)では，レジスタ更新を行ったすべての PE_k が PE_1 に更新終了を表すメッセージや以降 PE_1 が必要とするレジスタ値 (r_2, r_3) などを含むメッセージを送信する．PE_1 はメッセージを適宜，受信した後次の遷移に対応する動作を実行する(各メッセージには，状態遷移名の情報があるとする)．

ここで，ステージ(I)では，いずれのレジスタ値も異なった経路で二重化されて送受信されるので，受信側で二度目に受信したメッセージは，もし，すべてのリンクが信頼できるのであれば，無視できる．図3では，破線で表されるメッセージは，二度目の受信となるので無視されるメッセージを表している．

ところが，リンク故障がある場合は振る舞いは変わってくる．例として，PE_3 から PE_4 のリンク $buff_{34}$ が故障していると仮定する．このときレジスタ r_1 の情報は PE_3 から PE_4 には送られない．しかし，レジスタ r_1 の情報は PE_1 からは伝えられる．よって，リンク $buff_{34}$ が故障していても，PE_4 は自ノードのレ

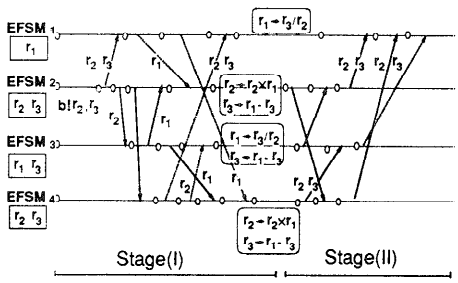


図 3 遷移 $s_2 \rightarrow s_3$ のタイミングチャート
Fig. 3 A timing chart of transition $s_2 \rightarrow s_3$.

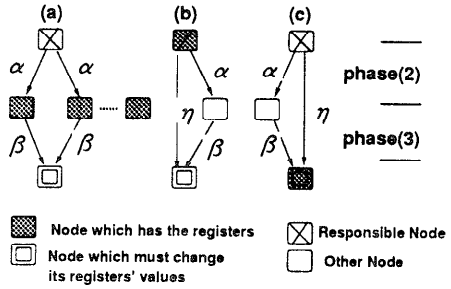


図 4 更新のためのレジスタ転送
Fig. 4 Exchange of registers' values.

ジスタ値を更新することができる．図3では，どのリンクがそのように故障しても，すべての PE_k が自ノードのレジスタ値を更新することが可能である．また同様に，PE_1 は，すべての PE_k がレジスタ更新を終了し，ステージ(II)が終了したことが分かる．よって，図3は，高々一つのリンクが故障する限りにおいて，正しく動作することがわかる．同様の動作系列を他の状態遷移でも導出することが可能である．これらの系列を接続することにより，要求仕様どおりに動く動作仕様を導出することができる．その動作仕様 $\langle PE_1, \cdots, PE_4 \rangle$ の例を付録に付す．

4. 各ノードの動作仕様の導出

4.1 基本方針

人力でメッセージ数や状態遷移を少なくする回避方法を考えるのは繁雑であり，得られた動作仕様が正しく動くことも保証するのは難しい．ここでは，メッセージの二重化を用いて故障に対応することとし，また，一状態遷移ごとにそれを模倣する各ノードの状態遷移系列を求めることにする．そこで，以下の部分問題 [SIM] を考える．そこで用いている模倣方針の妥当性については，5.2 節で述べる．

部分問題 [SIM]

入力：Alloc(SS)，リンク故障を考慮していない SS 中の1状態遷移 t

出力：各 PE_k における，t を模倣するための状態遷移系列とメッセージ内容

条件：後述の模倣方針のもとで（t を模倣する間の）総メッセージ数を最小にすること． □

[模倣方針] 全体仕様の1状態遷移を次の2ステージで実現する．

ステージ(I) レジスタ更新までのメッセージ転送ステージ(I)を以下のフェーズで構成する．

(1)責任ノードが実行可能な遷移を決め，動作を実行する．

(2)責任ノードがメッセージを送信し，中継すべきノードがこれを受信する．

(3)中継ノードがメッセージを中継し，最終的な受信ノードがこれを受け取る．

(4)レジスタを更新すべきノードがレジスタを更新する．

図4はフェーズ(2)，(3)におけるメッセージ転送を表したものであり，制約条件 3，4 より，三つの場合を考えれば十分である．(a)は更新を行うべきノード(二重丸)が更新に必要とするレジスタの一つに着目したとき，そのレジスタが自ノードと責任ノード（×丸）以外の2個所以上のノード（網がけ丸）にある場合である．責任ノードがレジスタを持っている二つのノードにきっかけを与えるメッセージ α を送り，これを受け取ったノードが，そのレジスタ値を用いてレジスタ更新すべきノードに，(レジスタの値の情報を持つ)メッセージ β を送ることを表している．次にレジスタを持つノードの一つが責任ノードである場合，(a)のように，二つのノード中継ノードとしてレジスタ値を送信してもよいし，(b)のように，一方のメッセージ η を直接送ることも考えられる．(a)，(b)のうち，メッセージ数の少ない方を用いるが，この詳細は 4.6 節で述べる．(c)は，自ノードでレジスタ値を更新できる場合である．この場合，更新のきっかけを与えるメッセージを二重化して送る．この場合も α, β メッセージを用いるが，レジスタ値の情報は持たなくてよい．いずれの場合でも，いずれか一つのメッセージが消失してもレジスタ更新すべきノードがレジスタ更新を行うことができる．また，責任ノードがフェーズ(1)で自ノードのレジスタ値をあらかじめ更新することも考えられる．これが可能な場合はこれを行う．

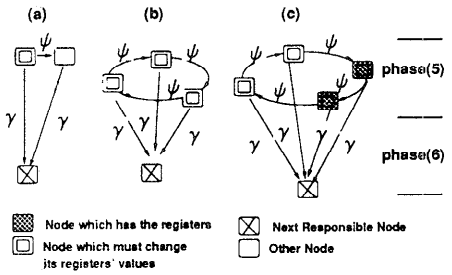


図 5 更新終了の通知
Fig. 5 Message exchange in stage (II).

ステージ(II) 更新終了通知のメッセージ転送

図5(a)は,更新を行うノードが一つの場合である.この場合,更新終了したノードが更新終了メッセージを次の責任ノードに送信するときメッセージの二重化のため,三つのメッセージが必要である.一般に更新を行うノードは複数存在すると考えられる.このとき(b)のようなメッセージの送受信パターンとする.まず,更新を行ったすべてのノード間でサイクルを形成するように更新終了メッセージ ψ をとなりへ一斉に送信する.次に各ノードは,ψを受信したなら,他ノードから受け取った更新終了情報と自ノードの更新終了情報の二つの情報をまとめて次の責任ノードに送る(メッセージ γ).このようにすれば,高々一つのリンク故障が生じても,次の責任ノードはすべての更新ノードが更新終了したことを知ることができる.図5のように,ステージ(II)の二つの部分をそれぞれフェーズ(5),(6)とする.

この際,次の責任ノードが必要とするレジスタ値を送信できるノードが,レジスタ更新をしたノード集合中にあれば,(c)のように,その一つがレジスタ値を,次の責任ノードに送信する.次責任ノードが必要とするレジスタの中には,この遷移で更新されないものがあるかも知れない.その場合は,フェーズ(I)において,図4(a),(b)の送受信パターンを用いて,あらかじめ,現在の責任ノードから次責任ノードへ必要なレジスタ値を送信しておくことにする.これは図4(a),(b)の二重丸で表しているノードを次責任ノードと読み変えれば良い.

なお,3.1節の単位時間 $4\Delta+\delta$ については,上述の方法では,一状態遷移を模倣する系列中に最大4回メッセージの送受信が行われる個所しかないこと,レジスタ更新も高々1回であるため,$4\Delta+\delta$ 時間経過すれば必ず次の状態遷移に移行しており,要求仕様での状態遷移を実行中に高々一つのリンク故障しか発生しないという条件が満たされる.

このメッセージ交換法は,要求仕様の一状態遷移を各動作仕様が模倣する際に,なるべく少ないフェーズでメッセージ交換することを意図している.

4.2 古いメッセージの処理について

前節で述べた方法では,二重化されたメッセージの一方のみを受信することにより,次の動作に進むため,古い(後で到着した)メッセージを矛盾の生じないようにうまく破棄する必要がある.本論文では,"タイムスタンプ"と呼ばれる識別子を用いてこのことを実現する.

タイムスタンプはノード間でやり取りされるすべてのメッセージに付加する.その値は,初期状態では0で,各責任ノードがステージ(I)の最初の動作を実行するごとに1ずつ増加させる.また,レジスタを更新したノードがステージ(II)の最初のメッセージを送信する際にもその値を1ずつ増加させる.このようにタイムスタンプは,要求仕様 SS の1状態遷移の1ステージが終了するごとに1ずつ増えるようにしておく(すなわち1状態遷移で2ずつ増える).これを行うために,各 PE_k にはカウンタレジスタ r_c を持たせておく.また,メッセージ受信時には,r_c を(その時点で分かりうる)最新のタイムスタンプにセットする.これにより,二重化したメッセージの一方のみを受信して,次のステージに進んだ場合や,一方のメッセージがステージの終了後に到着した場合でも,メッセージのタイムスタンプが r_c より小さければ破棄して良いことが分かり(一世代)古いメッセージによる誤動作を防ぐことができる.このようなメッセージ破棄のための状態遷移があっても,観測合同のもとでは,等価性は保証できる.

4.3 メッセージの型と合成

本論文では,ノード間でやり取りされるメッセージには,すべて状態遷移の識別子(要求仕様の各遷移に固有の名前)を付加する.状態遷移の識別子を持つことより,メッセージを受け取ったノードは要求仕様のどの状態遷移に相当する動作系列を実行すればよいかが判断できる.また,更新終了のメッセージ(レジスタ値を含む)を受け取った次責任ノードは,今要求仕様のどの状態にいてどの遷移が選択可能であるかを知ることができる.

次にメッセージの型について述べる.図4,図5のように,ノード間でやり取りされるメッセージにはそ

れぞれ型が付加されているものとする．このうち，α 型メッセージと η 型メッセージが同一のノードに送信される場合がある（例えば，他ノードへのレジスタ値の転送依頼のための α 型メッセージとそのノードがレジスタ値更新に必要とするレジスタ値を含む η 型メッセージの両方が責任ノードから送られてくる場合）．この場合，二つのメッセージをまとめて一つのメッセージとして送信すればメッセージ総数を少なくできる．そこで，α 型メッセージと η 型メッセージをまとめて μ 型メッセージと呼ぶことにする（二つの型の合成が生じたか生じなかったかにかかわらず μ 型と呼ぶ）．

4.4 模倣遷移系列の生成

ここでは，各 PE_k の導出方法について述べる．一般に，一つのレジスタは二つ以上のノードに分散配置されていると仮定しているので，レジスタ値を送受信するノードの決定には幾つかの自由度がある．これらの自由度の中から通信コストをできるだけ小さくするようなノードの選択法を 4.6 節で述べる．ここでは，それらのノードがすでに決定しているものと仮定して，各 PE_k の導出法を説明する．後で状態遷移図の縮約を行うが，基本的には各 PE_k はもとの要求仕様 SS と同じような形の状態遷移図を持ち，もとの状態遷移図の一つの状態遷移を通信のための動作を含む幾つかの状態遷移の系列に置き換えることにより構成する．

各ノードには Alloc(SS) で割り当てられたレジスタに加え，作業用のレジスタ r_0 を割り当てる．基本関数として，メッセージ m に含まれる入力変数やレジスタの名前と値の組を r_0 に追加する関数 put(r_0, m) と，レジスタ r_0 に最後に追加されたレジスタ r_h（または，入力変数 x）の値を取り出す関数 get(r_0, r_h)（または，get(r_0, x)）を用いる．

今，要求仕様の状態遷移 t ($=s-\langle G, a\cdots, CR\rangle-s'$) に対して，各 PE_k の状態遷移系列を求めることにする．以下の各フェーズで各 PE_k は指定された状態遷移系列を実行する．各 PE_k ごとに，これらの状態遷移系列の順次結合を得ると各 PE_k の状態遷移 t に対する模倣系列が得られる．

説明のための例として，例題の $s_2 \rightarrow s_3$ 間の状態遷移 ($s_2-\langle r_2>0, b!r_2, r_3, \{\cdots\}\rangle-s_3$) を用いる．**図 6** は，この遷移に対応する各 PE_k の状態遷移系列である．

（1）責任ノード（この例では，Snode(s_2)$=PE_2$）は次の状態遷移 $\langle G, a\cdots, CR'\rangle$ が与えられる．ここで，$G, a\cdots$ は t と同じ，CR' は以下のようなレジスタ更新式である．責任ノードに割り当てられたレジスタのうち，Cset 中のレジスタ値で更新できるレジスタのレジスタ更新式と，更新の対象となったレジスタのうち，今後送信メッセージ中に使うレジスタ値の旧値を r_0 に待避させるレジスタ更新式．その他のノードは，ϵ 遷移を与える．

（2）責任ノードは $gki!\mu(\,);\cdots;gkj!\mu(\,)$ が与えられる．ここで，送信動作は送るべき μ 型メッセージの数だけ繰り返される．順序は適当でよい．図 6 の例では，$g21!\mu(r_1, r_2)$; $g23!\mu(r_2)$; $g24!\mu()$ となる．ここでは，$\mu(r_1, r_2)$ でレジスタ r_1, r_2 の値を持つ μ 型メッセージを表している．

（3）μ 型メッセージの受信，β 型メッセージの送信，受信に関する動作は少々複雑である．今，PE_k を μ 型メッセージと β 型メッセージを共に受信するようなノードとする．一般に μ 型メッセージと β 型メッセージの到着順序は一意ではない．また，このノードが受信する可能性のある μ 型，β 型メッセージ数を n とするとき，リンク故障により，$n-1$ 個のメッセージしか受信できない場合もある．これらの点をふまえて，上述のようなノードは，以下のような遷移系列が与えられる．n 個の (μ or β) 型メッセージを $m_1, \cdots, m_n$ とし，これらが $PE_{i_1}, \cdots, PE_{i_n}$ から送信されるとする．

$$R \gg gki!\beta(\,);\cdots;gkj!\beta(\,) \gg \underbrace{R \gg \cdots \gg R}_{n-2}$$

where $R=\langle G_1, gki_1?m_1\rangle[\,]\cdots$

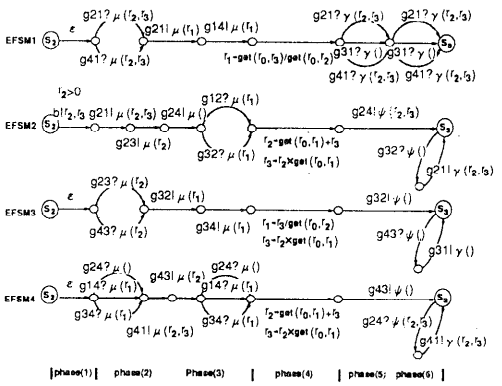


図 6 各 PE_k の遷移 s_2-s_3 の状態遷移系列
Fig. 6 Implementation of transition s_2-s_3.

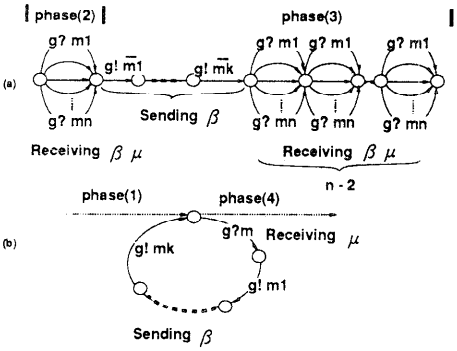


図 7　フェーズ (2) (3) における PE_i の構成
Fig. 7　Phase (2) and (3) in PE_i.

$$[]\langle G_n, gki'_n!m_n\rangle$$

すなわち，一つの（μ or β）型メッセージを受信した段階で，そのノードが送信すべきすべての β 型メッセージを順次送信する．その後，μ，β 型メッセージの受信動作を再び行うが，リンク故障を仮定しているので，残りの $n-2$ 個のメッセージを受信した段階（全体で $n-1$ 個）で次のフェーズに進むように定める．ここで，G_h は各メッセージの受信条件である（以降では簡単のため省略）．**図 7**（a）はそのための状態遷移系列の構成図である．なお，PE_k が μ 型メッセージを受信しないような場合も同様の状態遷移系列で実現する．この場合，β 型メッセージの送信部分はない．

次に PE_k が μ 型メッセージのみを受信し，β 型メッセージを受信しないような場合を考える．この場合，この遷移で PE_k にはただ一つの μ 型メッセージしか送信されない．また，PE_k はレジスタ更新を行わず，他のノードのレジスタ更新に必要なレジスタ値の送信（β 型メッセージの送信）のみがその仕事となる．ところで，リンク故障のために，このようなノードに μ 型メッセージが到着しないこともありうる．そこで，状態遷移系列を図 7（b）のように定める．図 7（b）では，μ 型メッセージが消失し，受信できない場合を考慮し，このフェーズの状態遷移の開始状態（フェーズ（1）の終了状態）と終了状態（フェーズ（4）の開始状態）は同一状態にしている．

図 6 の例では，すべての PE_k がレジスタ更新を行うので，前者の場合に相当する．このフェーズで各 PE_k へ送られるメッセージ数は順に 2, 2, 2, 3 である．各 PE_k はそれぞれ 1, 1, 1, 2 個のメッセージを受信した段階で次のフェーズに進む．メッセージの到着順は不定であるので，例えば，PE_4 では，$g24?\mu()[]g14?\mu(r_1)[]g34?\mu(r_1)$ を二つ直列に並べている（それらの間に β 型メッセージの送信系列を挿入している）．これにより，二つのメッセージの受信が可能である．一般に各メッセージ m の受信の際，そのメッセージにレジスタ値や入力変数の値が入っている．これらの値は後のレジスタ値の更新フェーズ（フェーズ（4））で用いるので，それらの値をレジスタ r_0 に格納する．すなわち，各受信動作でレジスタ更新式 $[r_0 \leftarrow \mathrm{put}(r_0, m)]$ を実行する（簡単のため図 6 では省略している）．

（4）ノード k がレジスタ更新を行うノードであるとき，内部動作 i（とレジスタ更新式）で自ノードのレジスタ更新を与える．ただし，更新式の右辺に現れるレジスタで，自ノードが保持しないレジスタ r' については，その値がレジスタ r_0 に保持されているので，その出現を $\mathrm{get}(r_0, r')$ に置き換える．例えば，PE_4 では，レジスタ r_1 が割り当てられていないので，$\{r_2 \leftarrow \mathrm{get}(r_0, r_1)+r_3,\ r_3 \leftarrow r_2 \times \mathrm{get}(r_0, r_1)\}$ というレジスタ更新式が得られる．

（5）ノード k が次の責任ノード（この例では，PE_1）以外で，かつこの状態遷移でレジスタ更新を行う必要があるノードである場合（この例では PE_1 以外のすべての PE_k が相当する），図 5 のように，リング状に ψ 型メッセージを送り，ψ 型メッセージの受信に成功した PE_k が次の責任ノード（PE_1）へ γ 型メッセージを送信するような遷移を得る．三つの γ 型メッセージのうちの二つを PE_1 が受信すれば，すべてのノードでのレジスタ更新が完了したことも分かる．もし，次責任ノードにメッセージを送る必要のあるノードの数が 1 であれば，普通にメッセージを二重化して送る（図 5（a））．

4.5 状態簡約について

一般に，SS の一つの状態遷移に対して，対応する状態遷移系列がすべて，ϵ 遷移である PE_k が存在する．そのような遷移系列の両端の状態はすべて非責任ノードになる状態である（一方が責任ノードであれば何らかの遷移系列が得られる）．そこで，これらの非責任ノードに相当する状態をすべて一つにまとめ ϵ 遷移を取り除く．非責任ノードである状態から出る遷移

系列はすべて受信動作から始まっていること，および各メッセージには遷移の識別子が付いていることより，非責任ノードを一つの状態にまとめても，識別子の内容に従って遷移を選択できるので，全体としては矛盾なく要求仕様どおりに動く．

4.6 メッセージの送信方法

以下では，前述の［模倣方針］とメッセージ統合の方法のもとで，1状態遷移の模倣に必要な総メッセージ数を最小にするメッセージの送受信内容（どの PE からどの PE へどの内容のメッセージを送るか）を決定する方法について述べる．例えば，図4の(a)の場合，中継ノードにどの二つを用いればメッセージ総数が少なくなるかは，他のレジスタ値の送信パターンに依存するので自明でない．同様に，責任ノードがレジスタ R_h を保持しており，そのレジスタ値をあるノードが必要とするときに，図4の(a)，(b)のいずれのパターンを用いるべきかも，同様の理由により自明でない．そこで，要求仕様 SS の各状態遷移について独立に，以下に述べる 0-1 整数線形不等式を導出し，これに対してメッセージ総数を表す目的関数の最小解を求めるという方法を用いる．最小解を与える変数の値から，どのノードからどのノードへどのタイミングでどのような内容のメッセージを送信すべきかが分かる．

4.6.1 メッセージの送信方法の決定に用いる命題変数

ある状態遷移 $s_i-\langle a\cdots, C(\cdots), CR\rangle\rightarrow s_j$ に対して，レジスタ更新するノードの集合 $\boldsymbol{\tau}$，自力でレジスタ更新できるノードの集合 $\boldsymbol{\lambda}$，レジスタ更新の際入力変数 x を必要とするノードの集合 $\boldsymbol{\theta}(x)$ は，SS と，Alloc(SS)からただちに求めることができる．以下，状態 s_i の責任ノード Snode(s_i) を u とする．また Snode(s_j) を z とする．βwv をノード w からノード v へ β 型メッセージを送信すべきときに 1，それ以外のとき 0 をとる命題変数とする．また，βwv_R_h (βwv_x) をノード w からノード v へ β 型メッセージでレジスタ R_h（入力変数 x）の値を送信すべきときに 1，それ以外のとき 0 をとる命題変数とする．他の変数も同様とする．

4.6.2 各命題変数の決定に用いる制約条件

以下の式は(1)〜(5)がステージ(I)に相当し，(6)〜(9)がステージ(II)に相当する．

(1)もし，ノード v が，更新時にレジスタ R_h を必要とするなら，

(1-1) $R_h\notin$Cset(s_i) のとき

$$\sum_{w\in \mathrm{Rnode}(R_h)} \beta wv\_R_h \geq 2$$

(1-2) $R_h\in$Cset(s_i) のとき

$$\sum_{w\neq u\wedge 1\leq w\leq p} \beta wv\_R_h + \eta u\_vR_h \geq 2$$

これは，ノード v がレジスタ R_h の値を2個所から受信することを表している．それぞれ，図4(a)，(b)に対応する．制約条件 3, 4 より，この不等式を満たす解領域は存在する．

(2)入力変数 x と各ノード $v\in\boldsymbol{\theta}(x)$ の組について，

$$\sum_{w\neq u\wedge 1\leq w\leq p} \beta wv\_x + \eta uv\_x \geq 2$$

これは，ノード v が2個所から入力変数の値を受信することを表している．

(3)各ノード $v\in\boldsymbol{\lambda}$ に対して，

$$\eta uv=1,\quad \sum_{1\leq w\leq p\wedge w\neq u\wedge w\neq v} \beta wv\geq 1$$

これは，図4(c)に対応する．

(4)任意のノード s, t ($1\leq s\leq p, 1\leq t\leq p$) とレジスタ R_h の組について，

(4-1) $s\in$Rnode(R_h) のとき

$$\alpha us\geq \beta st\_R_h$$

(4-2) $s\notin$Rnode(R_h) のとき

$$\alpha us\_R_h\geq \beta st\_R_h$$

(5)任意のノード s, t ($1\leq s\leq p, 1\leq t\leq p$) と入力変数 x の組について，

$$\alpha us\_x\geq \beta st\_x,\quad \alpha us\geq \beta st$$

(4)，(5)は，β 型メッセージを送信するノード s に対して必ず，ノード u から α 型メッセージを送信すべきことを表している．

次に，更新終了の通知，ノード z への Cset(s_j) のレジスタ値の通知について考える．

(6)各ノード $v\in\boldsymbol{\tau}-\{z\}$ に対して，

$$\gamma vz=1$$

(7)ノード z がレジスタ R_h を必要とするとき ($R_h\in$Cset(s_j)$\wedge z\notin$Rnode(R_h))，R_h と z の各組に対し，

(7-1) レジスタ R_h をもつノード w がレジスタ更新をする場合 ($w\in$Rnode(R_h)$\wedge w\in\boldsymbol{\tau}$)，そのような，$w$ を一つ選び，

$$\gamma wz\_R_h=1$$

(7-2) レジスタ R_h をもつノード w がレジスタ更新をしない場合 ($w\in$Rnode(R_h)$\wedge w\notin\boldsymbol{\tau}$)，

(7-2-a) $R_h\notin$Cset(s_i) の場合

$$\sum_{w\in \mathrm{Rnode}(R_h)\wedge w\neq z} \beta wz\_R_h \geq 2$$

(7-2-b)　$R_h \in \mathrm{Cset}(s_i)$ の場合

$$\sum_{w\neq u\wedge 1\leq w\leq p} \beta wz\_R_h + \eta uz\_R_h \geq 2$$

(8)もし $\gamma=\emptyset$ かつ，ノード z がすべてのレジスタ R_h を必要としないなら，

$\gamma uz=1$

これは，ノード間のメッセージのやりとりが全くなく，しかも責任ノードが替わる場合，ノード u から直接ノード z へメッセージを送る必要があることを表している．

以上で，フェーズ(6)の γ 型メッセージの制約が得られた．フェーズ(5)の ψ 型メッセージに関しては，γ 型メッセージの送信ノードが得られれば機械的に求めることができるので，これについては後述する．

メッセージの統合に関して，以下の制約式を与える．

(9)任意のノード w，s とレジスタ R_h あるいは，入力変数 x の組について

$\mu ws \geq \alpha ws$，$\mu ws \geq \alpha ws\_R_h$，
$\mu ws \geq \alpha ws\_x$，$\mu ws \geq \eta ws\_R_h$，
$\mu ws \geq \eta ws\_x$，$\beta ws \geq \beta ws\_R_h$，
$\beta ws \geq \beta ws\_x$

4.6.3 各命題変数の決定

上述の(1)～(9)の制約条件を満足するような変数の組の中で，

$$N=\left\{\sum_{1\leq y\leq p\wedge y\neq u}\mu uy\right\}+\left\{\sum_{1\leq w\leq p\wedge w\neq u}\sum_{1\leq v\leq n}\beta wv\right\}+\left\{\sum_{1\leq y\leq p\wedge y\neq z}\gamma yz\right\}$$

の値を最小にするように各変数の値を定める．

以上で，N を目的関数する 0-1 整数線形計画問題に帰着できた．この問題に対するアルゴリズムを用いて N の値を最小にするような解を求めればよい．求めた解から各 PE_k がどのタイミングでどのレジスタ値をどのノードと送受信しなければならないかがわかる．なお，0-1 整数線形計画問題は一般には NP 完全であるが，実用的には様々な高速化の手法が考案されている．

最後に，フェーズ(5)の ψ 型メッセージを定めるために，次のことを行う．

(10)上述の解で $\gamma wz=1$ なるノード w の集合を ξ と置く．

(10-1)　$|\xi|=1$ のとき $w\in\xi$ なる唯一の w と z について，$k\neq w\wedge k\neq z$ なる k を，一つ任意に選び，

$\psi wk=1$，$\gamma kz=1$

とする．

(10-2)　$|\xi|\geq 2$ のとき $w\in\xi$ なる各ノード w に対して，適当な順列を与える．この順列のもとでノード p_i の直後のノード p_{i+1} とするとき $\mathrm{next}(p_i)=p_{i+1}$ なる関数 next を導入する．順列の最後の要素 p_t と最初の要素 p_0 に対して $\mathrm{next}(p_t)=p_0$ とする．このとき，各ノード w($w\in\xi$)とノード $k=\mathrm{next}(w)$ に対して，

$\psi wk=1$

$\gamma wz\_R_h=1$ なら $\psi wk\_R_h=1$，$\gamma kz\_R_h=1$ とする．

4.7 導出アルゴリズムの全体構成

導出アルゴリズムの全体構成は以下のとおりである．

begin
　for each transition t^i in SS
　　$\langle ts_1{}^i, \cdots, ts_p{}^i\rangle$ = make_subsequence(Alloc(SS), t^i)
　$\langle PE_1, \cdots, PE_p\rangle$ = **replace each** t^i **in** $\langle SS, \cdots, SS\rangle$
　　to $\langle ts_1{}^i, \cdots, ts_p{}^i\rangle$
　$\langle PE_1, \cdots, PE_p\rangle$ = remove_redundant_transitions
　　($\langle PE_1, \cdots, PE_p\rangle$)
　return $\langle PE_1, \cdots, PE_p\rangle$
end

ここで，make_subsequence はリソースの配置情報と要求仕様の一状態遷移を入力として，各動作仕様の模倣系列 $\langle ts_1{}^i, \cdots, ts_p{}^i\rangle$ を得る手続きであり，4.4，4.6 節に相当する．3行目は，全体仕様の各状態遷移を上述の手続きで得られた模倣系列に置き換える操作を表し，remove_redundant_transitions は状態簡約の手続きである（4.5 節）．

5. 議　　論

5.1 評　　価

本手法で得られたメッセージの総数は，図1の例題の場合 33 メッセージであった．単純に二重化した場合（45 メッセージ）と比較すると 73% 程度にメッセージ削減できた．

要求仕様 SS と割当 Alloc (SS) を入力として，各動作仕様 PE_k を導出する導出系をワークステーション SONY NEWS 5000 (100 MIPS) 上で作成している．この導出系に対して以下のデータで導出時間を計測した．要求仕様 SS は 10 のレジスタを持ち，これらは平均して五つのノードに割り当てられているとする．各状態遷移で平均五つのレジスタを更新するレジスタ更新式が与えられているとする．このような各状態遷移をランダムに与えたとき，これらの状態遷移に対して，対応する各 PE_k の状態遷移系列を以下の時間で導出した．状態遷移のうち 67% について 20 秒以内，90% について 5 分以内で導出した．なお必ずしも最適解を出すわけではないが，数秒で近似解を求めるグリーディーアルゴリズムを用いた別ルーチンも用意している．

5.2 動作仕様 PE^{1-p} の構成法の妥当性について

我々の導出方法に関して，以下の定理が成り立つ．

定理 導出された動作仕様は，仮定している分散環境のもとで，要求仕様と等価である． □

証明 以下が成り立つことを示す．(i) ある動作系列の実行によってノード Snode(S_i) に制御が回ってきた時，次に行うべき動作の実行可能性がそのノードで判定可能であり，かつすべてのリンクは Snode(S_i) がそれまでに受信したタイムスタンプの最大値以下の（古い）タイムスタンプを持つメッセージしかたまっていない，(ii) その時点までの動作系列の実行によって変化した各レジスタの値は同じ動作系列を要求仕様 SS で実行した時の各レジスタの値に等しい，(iii) Snode (S_i) から始まる一連の動作を実行したとき，高々 1 個所リンクが故障しても全体として動作が進行し，次の責任ノード Snode(S_j) に制御が渡される．

これらの性質が成り立つことを証明するため，ある状態 S_i の責任ノード Snode(S_i) が一つの動作を選択・実行するとき，上述の性質 (i)，(ii) が共に成り立つことを仮定し，一つの動作を実行したとき，上述の性質 (iii) が成り立ち，かつ次の責任ノード Snode(S_j) に制御が移ったとき，性質 (i)，(ii) が再び成り立つことを帰納法を用いて証明する．なお，動作の開始時点（初期状態）で性質 (i)，(ii) が成り立つことは 3.4 節の制約条件 2 を仮定しているので明らか．

今ノード q を Snode(S_i) とする．ノード q が状態 S_i からの遷移を一つ選んだとき，q 以外のノードが選ぶべき遷移はノード q からのメッセージにより一意に定まる．この後，次の（要求仕様 SS に対応する）状態 S_j に進む際にリンク故障があっても各ノードで必要なレジスタ値が受信できることや，古いタイムスタンプのメッセージに基づいた誤った処理をしないこと，は 4 章の動作仕様 PE^{1-p} の構成法より明らか．よって，性質 (iii) が成り立つ．

これら一連の動作の実行により，次の責任ノード Snode(S_j) に制御が渡される．その際，要求仕様 SS に対応するレジスタ値の更新が行われている．よって，Snode(S_j) に制御が渡された時点でのレジスタ値は性質 (ii) を満たす．また，動作仕様 PE^{1-p} の構成法より Snode(S_j) に制御が渡された時点でノード Snode (S_j) は遷移条件の判定に必要なすべてのレジスタ値を保持しており，自ノードで次に行うべき動作の実行可能性が判定できる．また，一連の送受信動作によってやり取りされるメッセージのタイムスタンプはすべて Snode(S_j) が受信したタイムスタンプの最大値以下である．よって，性質 (ii) が成り立つ． □

提案する模倣方針の妥当性を示す性質として，以下の性質が挙げられる．

提案する模倣方針は，以下の条件を満たす模倣方針で導出される動作仕様のクラスでは，模倣フェーズ数を最小にする模倣方針の一つである．条件：要求仕様の一状態遷移ごとそれを模倣する状態遷移系列を導出すること，リンク故障回避のメッセージの送受信方法として，メッセージの二重化を用いること．（要求仕様の）各状態遷移に対して唯一の責任ノードが存在し，このノードがその状態遷移の入出力動作を行うこと．

□

上記のクラスに属する模倣方針として，以下の模倣方針も考えられる．その時点での責任ノードが，現時点でのすべてのレジスタ値を知っているとする．この責任ノードが，（本模倣方針と同様に）入出力動作を行い，さらに，各レジスタの更新後の値をまとめて計算し，これを必要とするノードに二重化して送信し，次責任ノードに各レジスタの更新後の値を二重化して送信するという集中型の方法である．この方法も，模倣フェーズ数を最小にする模倣方針の一つである．この模倣方針と本論文で述べる模倣方針の違いは，本論文で述べる模倣方針の方が分散性が高いことである．これにより，本模倣方針は，一般に，模倣効率の増加が期待できる．

また，本模倣方針を用いる理由として，以下の点が挙げられる．より実行効率のよい動作仕様を得るために，例えば，どのノードがどのタイミングでレジスタ

の更新後の値の計算を行い，また実際に更新をするかについて自由度を与えると，模倣フェーズ数やメッセージ数を最小にする解を得るときに考えるべき状態空間が大きくなり，動作仕様の導出が困難になること．

実行効率を考慮に入れた場合，模倣フェーズ数や，使用メッセージ数がコスト基準として大事であるという立場から，本導出法では，内部状態遷移数はコスト基準に入れていない．もちろん，これをコスト基準に入れた導出問題も考えられるであろう．

6. おわりに

本論文では，リンクでのメッセージ消失を仮定し，その上で分散システムの要求仕様から各ノードの動作仕様を自動生成する方法について述べた．本方式を現在構築中のグループウェアシステム[10]に発展させることが今後の課題である．

本論文で述べた導出方法では，一状態遷移ごとにそれを模倣する動作系列を導出する．複数の状態遷移に着目するとさらに最適化できる可能性がある．これについて，さらなる最適化を考えるのも今後の課題の一つとして考えられる．

謝辞 有益なコメントをいただきました査読者の方々に感謝いたします．

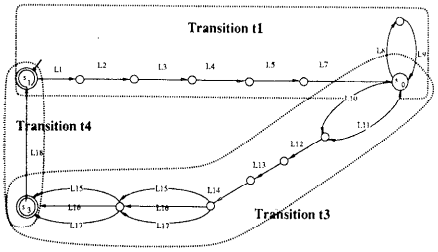
参考文献

1) Probert, R. and Saleh, K.: Synthesis of Communication Protocols: Survery and Assessment, *IEEE Trans. Comp.*, Vol. 40, No. 4, pp. 468-476 (1991).
2) Ramamorthy, C. V., Dong, S. T. and Usuda, Y.: Synthesis and Performance of Two-party Error Recoverable Protocols, *Proc. COMPSAC '86*, pp. 214-220 (1986).
3) Khendek, F., Bochmann, G. v. and Kant, C.: New Results on Deriving Protocol Specifications from Service Specifications, *Proc. of ACM SIGCOMM '89*, pp. 136-145 (1989).
4) Bochmann, G. v. and Gotzhein, R.: Deriving Protocol Specifications from Service Specifications, *Proc. of ACM SIGCOMM '86*, pp. 148-156 (1986).
5) Chu, P.-Y. M. and Liu, M. T.: Synthesizing Protocol Specifications from Service Specifications in FSM Model, *Proc. Computer Networking Symp. '88*, pp. 173-182 (Apr. 1988).
6) Park, D.: Concurrency and Automata on Infinite Sequences, *Theoretical Computer Science*, Lecture Notes in Computer Science, Vol. 104, pp. 167-183, Springer-Verlag (1981).
7) Wang, Z.: Shortest Path with Emergency Exits, *Sigcomm '90 Symp. Communications Architectures & Protocol*, pp. 166-176 (1990).
8) Milner, R.: *Communication and Concurrency*, Prentice-Hall (1989).
9) 岡野浩三，今城広志，東野輝夫，谷口健一：拡張有限状態機械モデルを用いた分散処理システムの要求仕様から各ノードの動作仕様の自動生成，情報処理学会論文誌，Vol. 34, No. 6, pp. 1290-1301 (1993).
10) 今城広志，岡野浩三，東野輝夫，谷口健一：拡張有限状態機械を用いた協調計算向きの計算システム，情報処理研究報告，93-DPS-7, 61, pp. 147-154 (1993).

付録

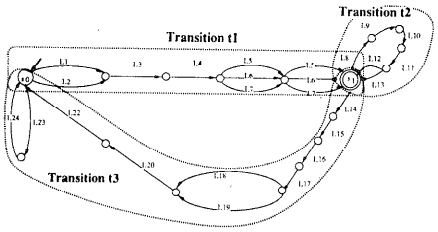
例題から得られた各動作仕様 PE_k $(1 \leq k \leq 4)$ を**図8～11**にあげる．PE_k ごとの各ラベルの内容の一覧においては，カウンタレジスタに関する操作，遷移条件等は省略している．

(平成6年3月18日受付)
(平成6年10月13日採録)



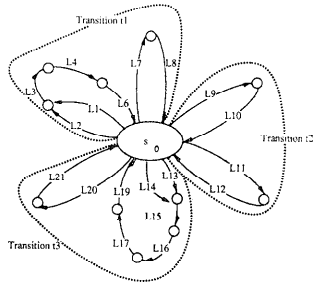
L_1	$a?x,y,z,$	$r_0 \leftarrow \mathrm{put}(r_0,\{x,y,z\})$
L_2	$g12!\mu(x,y,z)$	
L_3	$g13!\mu(x,y,z)$	
L_4	$g14!\mu(x,y,z)$	
L_5	i	$r_1 \leftarrow \mathrm{get}(r_0,x)$
L_7	$g14!\psi$	
L_8	$g31?\psi$	
L_9	$g12?\gamma$	
L_{10}	$g21?\mu(r_2,r_3),$	$r_0 \leftarrow \mathrm{put}(r_0,\mu(r_2,r_3))$
L_{11}	$g41?\beta(r_2,r_3),$	$r_0 \leftarrow \mathrm{put}(r_0,\beta(r_2,r_3))$
L_{12}	$g12!\mu(r_1)$	
L_{13}	$g14!\mu(r_1)$	
L_{14}	i	$r_1 \leftarrow \mathrm{get}(r_0,r_3)\ /\ \mathrm{get}(r_0,r_2)$
L_{15}	$g21?\gamma(r_2,r_3),$	$r_0 \leftarrow \mathrm{put}(r_0,\gamma(r_2,r_3))$
L_{16}	$g41?\gamma(r_2,r_3),$	$r_0 \leftarrow \mathrm{put}(r_0,\gamma(r_2,r_3))$
L_{17}	$g31?\gamma$	
L_{18}	$a!\ r_1, \mathrm{get}(r_0,r_2), \mathrm{get}(r_0,r_3)$	

図 8 PE_1
Fig. 8 PE_1.



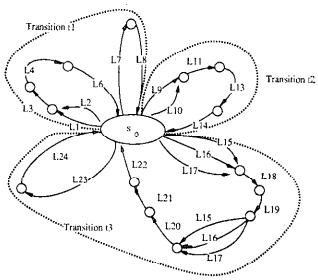
L_1	$g12?\mu(x,y,z),$	$r_0 \leftarrow \mathrm{put}(r_0,\mu(x,y,z))$
L_2	$g32?\beta(x,y,z),$	$r_0 \leftarrow \mathrm{put}(r_0,\beta(x,y,z))$
L_3	$g24!\beta(x,y,z)$	
L_4	i	$r_2 \leftarrow \mathrm{get}(r_0,y),\ r_3 \leftarrow \mathrm{get}(r_0,z)$
L_5	$g12?\gamma$	
L_6	$g42?\gamma$	
L_7	$g32?\gamma$	
L_8	$b?y,$	$r_2 \leftarrow y$
L_9	$g24!\mu(y)$	
L_{10}	$g23!\mu(y)$	
L_{11}	i	$r_2 \leftarrow \mathrm{get}(r_0,y)$
L_{12}	$g42?\gamma$	
L_{13}	$g32?\gamma$	
L_{14}	$r_2 > 0,\ b!r_2,r_3$	
L_{15}	$g21!\mu(r_2,r_3)$	
L_{16}	$g23!\mu(r_2)$	
L_{17}	$g24!\mu$	
L_{18}	$g12?\beta(r_1),$	$r_0 \leftarrow \mathrm{put}(r_0,\beta(r_1))$
L_{19}	$g32?\beta(r_1),$	$r_0 \leftarrow \mathrm{put}(r_0,\beta(r_1))$
L_{20}	i	$r_2 \leftarrow \mathrm{get}(r_0,r_1)+r_3,\ r_3 \leftarrow r_2 \times \mathrm{get}(r_0,r_1)$
L_{22}	$g24!\psi(r_2,r_3)$	
L_{23}	$g32!\psi$	
L_{24}	$g21!\gamma$	

図 9 PE_2
Fig. 9 PE_2.



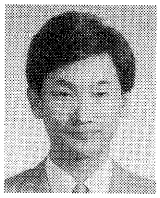
L_1	$g13?\mu(x,y,z),$	$r_0 \leftarrow \mathrm{put}(r_0,\mu(x,y,z))$
L_2	$g43?\beta(x,y,z),$	$r_0 \leftarrow \mathrm{put}(r_0,\beta(x,y,z))$
L_3	$g32!\beta(x,y,z)$	
L_4	i	$r_1 \leftarrow \mathrm{get}(r_0,x),\ r_3 \leftarrow \mathrm{get}(r_0,z)$
L_6	$g31!\psi$	
L_7	$g43?\psi$	
L_8	$g32!\gamma$	
L_9	$g23?\mu(y)$	
L_{10}	$g34!\beta(y)$	
L_{11}	$g43?\psi$	
L_{12}	$g32!\gamma$	
L_{13}	$g23?\mu(r_2),$	$r_0 \leftarrow \mathrm{put}(r_0,\mu(r_2))$
L_{14}	$g43?\beta(r_2),$	$r_0 \leftarrow \mathrm{put}(r_0,\beta(r_2))$
L_{15}	$g32!\beta(r_1)$	
L_{16}	$g34!\beta(r_1)$	
L_{17}	i	$r_1 \leftarrow r_3\ /\ \mathrm{get}(r_0,r_2),\ r_3 \leftarrow \mathrm{get}(r_0,r_2) \times r_1$
L_{19}	$g32!\psi$	
L_{20}	$g43?\psi$	
L_{21}	$g31!\gamma$	

図 10 PE_3
Fig. 10 PE_3.



L_1	$g14?\mu(x,y,z)$	$r_0 \leftarrow \mathrm{put}(r_0,\mu(x,y,z))$
L_2	$g24?\beta(x,y,z)$	$r_0 \leftarrow \mathrm{put}(r_0,\beta(x,y,z))$
L_3	$g43!\beta(x,y,z)$	
L_4	i	$r_2 \leftarrow \mathrm{get}(r_0,y),\ r_3 \leftarrow \mathrm{get}(r_0,z)$
L_6	$g43!\psi$	
L_7	$g14?\psi$	
L_8	$g42!\gamma$	
L_9	$g24?\mu(y),$	$r_0 \leftarrow \mathrm{put}(r_0,\mu(y))$
L_{10}	$g34?\beta(y),$	$r_0 \leftarrow \mathrm{put}(r_0,\beta(y))$
L_{11}	i	$r_2 \leftarrow \mathrm{get}(r_0,y)$
L_{13}	$g42!\gamma$	
L_{14}	$g43!\psi$	
L_{15}	$g24?\mu$	
L_{16}	$g14?\beta(r_1),$	$r_0 \leftarrow \mathrm{put}(r_0,\beta(r_1))$
L_{17}	$g34?\beta(r_1),$	$r_0 \leftarrow \mathrm{put}(r_0,\beta(r_1))$
L_{18}	$g41!\beta(r_2)$	
L_{19}	$g43!\beta(r_2)$	
L_{20}	i	$r_2 \leftarrow \mathrm{get}(r_0,r_1)+r_3,\ r_3 \leftarrow r_2 \times \mathrm{get}(r_0,r_1)$
L_{22}	$g43!\psi$	
L_{23}	$g24?\psi(r_2,r_3)$	
L_{24}	$g41!\gamma(r_2,r_3)$	

図 11 PE_4
Fig. 11 PE_4.



岡野 浩三（正会員）

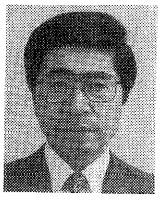
昭和 42 年生．平成 2 年大阪大学基礎工学部情報工学科卒業．平成 5 年同大学大学院博士後期課程中退．同年同大学基礎工学部情報工学科助手，現在に至る．代数的手法によるソフトウェア設計開発法，分散システム等の研究に従事．電子情報通信学会会員．

今城 広志（正会員）

昭和 44 年生．平成 4 年大阪大学基礎工学部情報工学科を中退し，同大学院に進学．平成 6 年同大学院博士前期課程修了．在学中は，分散システム，グループウェア等の研究に従事．現在，日本電気(株)パーソナルコンピュータ事業部製品技術部所属．

東野 輝夫（正会員）

昭和 31 年生．昭和 54 年大阪大学基礎工学部情報工学科卒業．昭和 59 年同大学院博士課程修了．工学博士．同年同大学助手．平成 2 年，6 年モントリオール大学客員研究員．平成 3 年大阪大学基礎工学部情報工学科助教授，現在に至る．分散システム，通信プロトコル等の研究に従事．電子情報通信学会，IEEE-CS, ACM 各会員．



谷口 健一（正会員）

昭和 17 年生．昭和 40 年大阪大学工学部電子工学科卒業．昭和 45 年同大学院基礎工学研究科博士課程修了．工学博士．同年同大学基礎工学部助手．現在，同情報工学科教授．計算理論，ソフトウェアやハードウェアの仕様記述・実現・検証の代数的手法および支援システム，関数型言語の処理系，分散システムや通信プロトコルの設計・検証法などに関する研究に従事．