社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.
COMP95−96, SS95−51 (1996−03)

レジスタ付き時間ペトリネットで記述された分散システムの時間制約付き全体仕様からその時間制約を満たす各ノードの動作記述の自動導出

山口 弘純　岡野 浩三　東野 輝夫　谷口 健一

大阪大学 基礎工学部 情報工学科
560 大阪府 豊中市 待兼山町 1-3
E-mail : { h-yamagu, okano, higashino, taniguchi }@ics.es.osaka-u.ac.jp

あらまし リアルタイム分散システムを設計する一つの手法として, 各動作の生起時刻に時間制約が与えられた分散システムの全体仕様から, ノード (計算機) 間の通信路の遅延時間等を考慮して, 与えられた時間制約を満たしながらメッセージ交換を行い, 全体仕様通りの動作をする各ノードの動作記述を自動導出する手法が提案されてきている. しかし, 従来の手法では, 並列動作の同期や, システムのリソースを取り扱えないクラスに制限されていた. 本稿では, 並列動作やその同期, 非決定的選択動作が記述でき, かつリソースを扱えるレジスタ付き時間ペトリネットモデル (TPNR モデル) を新たに提案し, そのモデルで記述された分散システムの全体仕様と各ノードへのリソースの配置指定, ノード間の通信路の最大遅延時間, ノード間のクロックの最大誤差が与えられたときに, (1) 全体仕様で与えられるシステムが分散システムとして実行可能であるか (分散実行可能性) を判定し, (2) 可能である場合には全体仕様の時間制約を満たす各ノードの動作記述を自動導出する手法を提案する. 提案する手法では, メッセージ交換の仕方などに一定の方針 (模倣方針) を定め, その模倣方針のもとでの分散実行可能性判定を線形計画問題に帰着し, 各ノードの動作を実行可能な時刻の幅の総和を最大にする解を得ることで, 動作の実行時刻に関して自由度の高い動作記述を得る工夫を行っている. 本手法により, 設計者自身が通信動作に要する遅延時間を詳細に検討せずに, 動作の実行可能時間幅の広い動作記述を自動生成できる, 通信動作を直接記述する場合の誤りをなくすことができる, などの効果が期待できる.

Deriving Protocol Specifications of Real-Time Distributed Systems from Service Specifications in a Time Petri Net Model with Registers

Hirozumi YAMAGUCHI　Kozo OKANO　Teruo HIGASHINO　Kenichi TANIGUCHI

Dept. of Information and Computer Sciences,　Osaka University
Machikaneyama 1-3, Toyonaka, Osaka 560, JAPAN

Abstract Some methods for deriving protocol specifications of real-time distributed systems from given service specifications with time constraints (deadlines of actions) have been proposed. However, these methods cannot treat the class of service specifications including parallel synchronization and parameters. In this paper, we propose a method for deriving a protocol specification automatically from a given service specification in an extended model of time Petri nets, called a *Time Petri Net model with Registers* (*TPNR model*). In our derivation algorithm, a resource allocation specifying the allocation of resources to the protocol entities, each maximum communication delay between two protocol entities, and a maximum error among the clocks of the protocol entities can be specified freely by the designers. The algorithm (1) decides whether the service specification can be implemented in the protocol entities satisfying the deadlines of actions using a method for solving linear programming problems, and if so, (2) derives a protocol specification, where the sum of the ranges of all deadlines can be maximized. Using our method, the designers will be free from considering the details of communication delays on the design of real-time distributed systems.

1 まえがき

分散システムの設計時におけるノード (計算機) 間の通信動作の記述の繁雑さや, それに伴う誤りをなくすことを目的とした設計法の一つとして, 分散システム全体を一つのシステムと見なしたときにそのシステムが実行すべき動作及びそれらの実行順序を形式的に記述したもの (全体仕様) から, 全体仕様で指定された通りの動作をするために分散システムの各ノードが実行すべき動作と他ノードとの通信動作及びそれらの実行順序を記述したもの (各ノードの動作記述) を自動導出する手法が LOTOS や拡張有限状態機械, ペトリネットの拡張モデルなど様々な計算モデルに対して提案されてきている [2, 3, 6, 10]. しかし, それらは処理の実行時刻のデッドライン (実時間制約) が指定されたリアルタイム分散システムを扱うことができないなどの制約があった. 一般に, ノード間の通信に要する時間 (通信路の遅延時間) を考慮して, デッドラインを守るために各ノードがどの動作をどの時刻に行えばよいかを設計者が直接動作記述として記述するのは容易ではない. このため, 実時間制約が与えられた全体仕様から各ノードの動作記述を自動導出できることが望ましい.

そのような目的のため, 動作の生起時刻に時間制約を与えた分散システムの全体仕様から, 通信に要する時間を考慮して, その時間制約を満たしながら全体仕様で指定された通りに動作する各ノードの動作記述を自動導出する研究が注目を集めている [8, 9]. 文献 [8], [9] の手法では全体仕様を記述するモデルとしてそれぞれ時間オートマトンの組及び時間制約付き LOTOS (LOTOS/T+) を用いている. しかし, 時間オートマトンの組ではシステムのリソースの変化が記述できず, 並列動作の同期も記述することができない. また, LOTOS/T+ では並列動作やその同期が記述できる能力を持つが, 文献 [9] では並列動作の同期が扱えない全体仕様のクラスに制限している. また基本的に Basic LOTOS を対象としているのでリソースの変化は記述できない.

本稿では, 我々が文献 [10] で提案した, 並列動作やその同期, 非決定的選択動作が記述できるレジスタ付きペトリネットモデルを, 動作の生起時刻間の実時間制約を指定できるように時間ペトリネット (Time Petri Net) [5] に拡張したレジスタ付き時間ペトリネットモデル (Time Petri Net model with Registers, 以下 TPNR モデル) を提案する. TPNR モデルではシステム外部との入出力が行われるゲートとシステムの内部リソースに相当するレジスタに対し, 時間ペトリネットにより入出力動作やレジスタ値の更新動作の実行制御を行う. また, 各トランジションの発火は, トークンの配置 (マーキング) による制御だけでなく, ゲートからの入力値やレジスタ値に関する条件式 (ガード) による制御及び各トランジションに与えられた発火時間の上限と下限による制御が可能である. このモデルで記述された分散システムの全体仕様と各ノードへのゲートとレジスタの配置指定及びノード間の通信路の最大遅延時間, 各ノードが持つクロック間の最大誤差が与えられたときに, 全体仕様通りの動作を行うために各ノードがデータの受渡しや動作の実行終了の通知のためのメッセージをどのように交換すべきかをあらかじめ定めた方針 (模倣方針) にしたがい, (1) 全体仕様として与えられたシステムがその模倣方針のもとで時間制約を満たしながら分散システムとして実行可能であるか否か (分散実行可能性) を判定し, (2) 可能であるときにはその判定で得られた解を用いて, その模倣方針に基づき動作する各ノードの動作記述を自動導出する手法を提案する. なお, 導出した各ノードの動作記述も TPNR モデルで記述されたものであるとする.

一般に, ある定められた模倣方針にしたがって動作する各ノードの動作記述を導出する場合, 分散実行可能性判定における解の存在範囲や導出した動作記述の各動作を実行可能な時刻の幅を広くするには, 模倣に要する時間がなるべく短く済むような模倣方針を定めることが望ましい. 本稿では, なるべくデータ及び実行順序の依存関係のある動作間にのみメッセージ交換を指定し, 依存関係のない動作はそれぞれ並列に実行できるようにすることで模倣に要する時間を節約する模倣方針を定める. また分散実行可能性判定では, 模倣方針にしたがって全体仕様通りの動作を行うのに各ノードはどの時刻までに入出力動作やレジスタ値の更新動作を行わなければならないかを, それらの実行時刻を表す変数上の線形不等式として表し, 線形計画問題の解法を用いてそれらを満たす解が存在するか否かを調べ, 導出時には, 同判定で得られた解を用いて各ノードの動作記述の各トランジションの発火時間の上限と下限を決定する. この際, 線形計画問題の目的関数として各トランジションの発火時間幅の総和を与え, それを最大にする解を求めることにより, 全体として動作の実行時刻幅の自由度が最も広くなる動作記述を得られるように工夫している.

本手法では, 以下のような利点がある. (1) 複数ノードが関係する非決定的選択動作 (分散選択動作) を含み, 並列動作によるリソースの同時アクセスが起こり得るような全体仕様からも自動導出が可能であるなど従来より広いクラスの全体仕様を対象に自動導出が行える. (2) 本アルゴリズムは, 全体仕様の他に分散システムのノード数, リソースの配置指定, 通信路の遅延時間, ノード間のクロックの誤差といった, 分散システムを規定するのに必要なパラメータを入力として与えることができる. したがって, 様々な分散環境に対しその環境上での分散実行可能性判定と各ノードの動作記述の自動導出とを容易に行える.

本手法により, 設計者自身が通信動作に要する遅延時間を詳細に検討せずに, 動作の実行可能時間幅の広い動作記述を生成できる, 通信動作を直接記述する場合の誤りをなくすことができる, などの効果が期待できる.

2 準備

2.1 記述モデル

本稿では, 文献 [10] で提案した, レジスタ付きペトリネットモデル (PNR モデル) の基本ペトリネットを時間ペトリネットに拡張したモデルを提案する.

提案するレジスタ付き時間ペトリネットモデル (Time Petri Net model with Registers, 以下 TPNR モデル) では, 入出力が実行される有限個のゲート及び大域変数を表す有限個のレジスタに対し, ゲートへの入出力イベント及びレジスタ値を表す変数 (レジスタ変数) の更新をトランジションに記述できる. 各トランジション t は, $[\mathrm{Eft}(t), \mathrm{Lft}(t)]$ で表される発火時間の下限 $\mathrm{Eft}(t)$ 及び上限 $\mathrm{Lft}(t)$ の他に, ⟨ ガード, 入出力イベント, レジスタ更新文 ⟩ のラベルを持つ. 入出力イベントは $a?x$, $a!Exp(...)$, i のいずれかの形 (a はゲート) である. $a?x$ は入力イベントを表し, ゲート a からの入力を入力変数 x に取り込むことを, $a!Exp(...)$ は出力イベントを表し, 表現 $Exp(...)$ の値をゲート a に出力することを, i は内部イベントを表し, いかなる入出力も行わないことをそれぞれ示す. なお, 入力変数はそのトランジションのみが値を参照できる局所変数として扱い, 表現 $Exp(...)$ はレジスタ変数を引数に持つことのできる関数として定義する. ガードは, 入力変数とレジスタ変数を引数に持つことのできる述語である. レジスタ更新文は $R_{h_1} \leftarrow f_1(...)$ ... $R_{h_i} \leftarrow f_i(...)$ の形かもしくは空文 (どの値も更新しない場合) で与えられ, 更新後のレジスタ値を表す各関数 f_j $(1 \leq j \leq i)$ はレジスタ変数及び入力変数を引数に持つことができる.

各トランジション t は, 発火に必要なトークンが t の入力プレースに揃い, ガードの値が真となった時刻から時間 $\mathrm{Eft}(t)$ から $\mathrm{Lft}(t)$ の間に発火するか発火に必要なトークンを失う. トランジション t が発火すると, t の入出力イベントとレジスタ更新文がこの順に実行される. また, トランジションの実行時間は 0 とする.

図 1 は, 2 台のカメラを介して静止画を一定周期で取り込むことで動画として記録し, それらを同時再生する簡単な動画記録再生システムの TPNR モデルによる記述例である. このシステムは キーボード, ディスプレイ, カメラ 1, カメラ 2 にそれぞれ対応

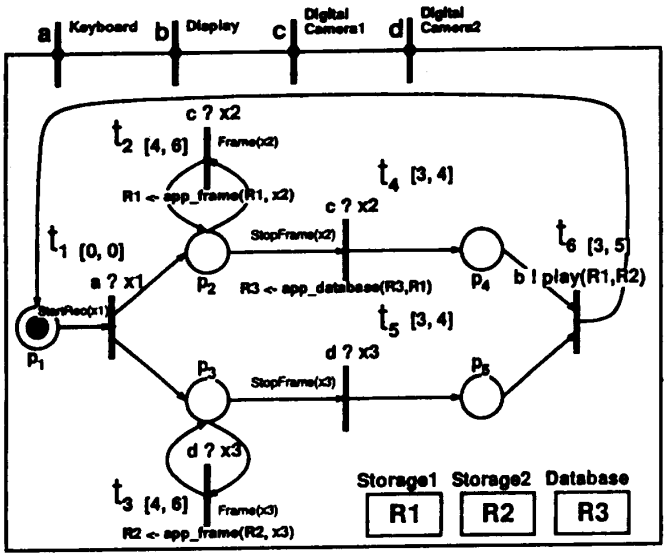

図 1: 動画記録再生システムの全体仕様記述例

する 4 つのゲート a, b, c, d と記憶装置 1, 記憶装置 2, 画像データベースにそれぞれ対応する 3 つのレジスタ R_1, R_2, R_3 を持つ. このシステムは初期マーキング $M_0 = (1,0,0,0,0)$ から以下のように動作する. まず, ゲート a (キーボード) からユーザからの録画命令が入力変数 x_1 に入力されると, ガード $\mathbf{StartRec}(x_1)$ が真となる. トランジション t_1 は発火時間 $[0,0]$ なので t_1 は即時に発火する. 発火時に実際にゲート a から入力が読み込まれる. t_1 の発火後 のマーキング $M_1 = (0,1,1,0,0)$ では, t_2 が発火時間 $[4,6]$ の間で発火を繰り返す. t_2 が発火すると, ゲート c (カメラ 1) からの静止画が入力変数 x_2 に取り込まれ, それがレジスタ R_1 (記憶装置 1) に格納される. ゲート c から静止画取り込みの停止命令が入力され, t_4 のガード $\mathbf{StopFrame}(x_2)$ が真となると t_4 が発火時間 $[3,4]$ の間に発火し, レジスタ R_1 (記憶装置 1) に格納された静止画列がレジスタ R_3 (画像データベース) に追加される. 同様に, マーキング M_1 において, t_3 の発火によりゲート d (カメラ 2) からの静止画の取り込みとレジスタ R_2 (記憶装置 2) への格納が上記の動作と並列に行われる. 最後に, t_4 及び t_5 が共に発火した後のマーキング $M_4 = (0,0,0,1,1)$ で, t_6 が発火時間 $[3,5]$ の間に発火し, レジスタ R_1 及び R_2 に格納された静止画列を動画としてゲート b (ディスプレイ) で同時再生し, 初期マーキング M_0 に戻る. これら一連の動作が繰り返し実行される.

TPNR モデルでは, ガードや出力イベント, レジスタ更新文に現れる関数 StartRec や app_database, play などを設計者が自由に記述できる (その内容は以下の導出アルゴリズムに影響しない). なお, 図 1 においては常に実行可能であることを表すガード "true" とどのレジスタ値も更新しない空のレジスタ更新文は省略している.

2.2 分散システムを実現する環境

本稿で対象とする分散システムは p 個のノード 1, 2, ..., p からなり, 任意のノード i とノード j の間には最大遅延時間 (本手法で対象とするシステムが送受信する任意長のメッセージに対してそれが発信されてから到達するまでに要する最大時間) が定数 $\mathrm{Delay}(i,j)$ で抑えられるような十分信頼できる全二重通信路が存在する (前提条件 1) とし, そのノード i (ノード j) 側の端をゲート g_{ij} (ゲート g_{ji}) で表す. ここで現実的なネットワークを考慮し, Delay は任意のノード i, j, k についての三角不等式 $\mathrm{Delay}(i,k) + \mathrm{Delay}(k,j) \geq \mathrm{Delay}(i,j)$ を満たすものとしておくが, これを満たさない場合も本手法は適用可能である. 表 1 は $p = 3$ である場合の最大遅延時間関数 Delay の例である.

表 1: 通信路の最大遅延時間関数 Delay の例

		受信ノード		
		ノード 1	ノード 2	ノード 3
送信ノード	ノード 1	0	2	2
	ノード 2	2	0	3
	ノード 3	5	4	0

各ノードはそれぞれ正確に時刻を更新するクロックを持ち, 任意の 2 ノードに対し, それらのクロックの誤差は常にある定数時間 δ で抑えられるものとする* (前提条件 2). すなわち, 最も早い時刻を示すクロックと最も遅い時刻を示すクロックの誤差は常に一定で, その最大は時間 δ で抑えられる.

2.3 取り扱う問題の定義

以下, 全体仕様を $Sspec$, ノード k の動作記述を $Pspec_k$, p 個のノードからなる分散システムの動作記述を $Pspec^{(1,p)}$, $Sspec$ のゲート及びレジスタを各ノードへどのように割り当てるか (ゲートとレジスタの配置指定) を $\mathrm{Alloc}(Sspec)$ で表す. 図 1 の $Sspec$ に対し $p = 3$ である場合の $\mathrm{Alloc}(Sspec)$ の例を表 2 に示す.

表 2: ゲートとレジスタの配置指定 $\mathrm{Alloc}(Sspec)$ の例

	ノード 1	ノード 2	ノード 3
ゲート	a, b	c	d
レジスタ	R_3	R_1, R_3	R_2

本節では, 本稿で扱う問題を導出問題として定義する.

定義 1 (導出問題) TPNR モデルで記述された全体仕様 $Sspec$ とゲートとレジスタの配置指定 $\mathrm{Alloc}(Sspec)$, ノード間の通信路の最大遅延時間関数 Delay, 及びノード間のクロックの最大誤差 δ が与えられたとき, TPNR モデルで記述された正しい動作記述 $Pspec^{(1,p)}$ を導出する問題を扱う. ただし, 対象とする分散システムは前節の前提条件 1, 前提条件 2 を満たすものとする. ここで, 各ノード間のメッセージの送受信に用いられる入出力イベント ($g_{ij}!\ldots$, $g_{ij}?\ldots$) と内部イベント i を観測不能なイベントとし, その他のイベントを観測可能なイベントとしたとき, 全体仕様 $Sspec$ と動作記述 $Pspec^{(1,p)}$ が観測合同 [4] であれば両者は等価であるといい, さらにイベントの生起時刻を無視したときに導出した動作記述が全体仕様と等価であり, かつ動作記述で実行可能かつ観測可能なイベント系列について, そのイベント系列の各イベントが動作記述で指定された生起時刻制約を満たす範囲の任意の時刻で生起するようなイベント系列を全体記述でも実行可能であるとき, 導出した動作記述は正しいと定義する. □

ここでアルゴリズムの簡単化のため, 全体仕様 $Sspec$ は以下の制約を満たすものとする.

[制約 1] $Sspec$ を記述する TPNR モデルのペトリネットは活性安全な自由選択ネット [1] でなければならない.

[制約 2] $Sspec$ は一方があるレジスタ R の値を更新し他方がそのレジスタ R の値を更新または参照するような同時発火可能な 2 つのトランジション t_i, t_j (レジスタ値競合と呼ぶ) を含んでもよいが, そのような t_i, t_j でレジスタ R を出力イベントやガードに用いない.

[制約 3] $Sspec$ は内部イベント i を含まない.

[制約 4] $Sspec$ はレジスタ値競合と複数ノードが関係する非決定的選択動作を含まない. □

自由選択ネットは, 複数の出力トランジションをもつプレース p に対し, それらのトランジションは p 以外に入力プレースを持たないペトリネットである. 制約 1 で全体仕様を記述する TPNR モデルのペトリネットのクラスを自由選択ネットに制限することにより, 選択構造を持つ (出力トランジションを複数持つ) プレース p は, 他のプレースに依存することなく出力トランジションの発火可能性を判定できるため, 複数ノードが関係する非決定的選択の実現を簡単化できる. さらにその自由選択ネットが活性安全

*分散システムにおける同期時計の実現法として, 文献 [7] を始めとした多くの手法が提案されている.

であるとすることで, 活性安全な自由選択ネットが満たす複数の状態機械への分解可能性を導出アルゴリズムに利用できる. これらの詳細は文献 [11] 参照.

制約 2 において, t_i があるレジスタ R の更新を, t_j がガードや出力イベントに R を用いるとする. 一般に分散システム上では t_j のガードの真偽値判定と出力イベントは t_i のレジスタ R の更新とは別のノードで実行される. このため, R の値が更新されたとしても, t_j の真偽値判定や出力イベントは更新前の値か更新後の値かのどちらを用いればよいかを知ることができないため, この制約を与える.

制約 3 は導出アルゴリズムの簡単化のため与えるものであり, 本質的制約ではない.

以下では紙面の都合上, 制約 4 を満たす全体仕様に対する導出アルゴリズムの概要を述べる. 制約 4 を満たさない全体仕様から導出を行う場合の問題点とその解決法, 及びアルゴリズムの詳細は文献 [11] 参照.

3 導出アルゴリズムの概要

3.1 分散実行の基本方針

一般にある定められた模倣方針にしたがって動作する各ノードの動作記述を導出する場合, 分散実行可能性判定における解の存在範囲や導出した動作記述の各動作を実行可能な時刻幅を広くするには, 模倣に要する時間がなるべく短く済むような模倣方針を定めることが望ましい.

分散環境上では, 全体仕様の各トランジションの入出力イベントやレジスタ更新文は, その入出力イベントが実行されるゲートや更新されるレジスタを保持するノードがそれぞれ実行する. 例えば, 図 1 の全体仕様のトランジション t_4 と表 2 の配置指定に対し $c?x_2$ はノード 2 が, $R_3 \leftarrow \mathrm{app\_database}(R_3, R_1)$ はノード 1 とノード 2 がそれぞれ実行する. 本稿で定める模倣方針では, 基本的に, トランジション t_i の入出力イベントを実行したノードが次のトランジション t_j のイベントを実行するノードと t_i のレジスタ更新文を実行するノードへ入出力イベントの実行終了通知を送信する. これにより, t_j 以降のトランジションの入出力イベントやレジスタ更新文と t_i のレジスタ更新文とは独立に実行でき, 文献 [6, 10] などで用いている t_i の入出力イベントとレジスタ更新文を順に実行していく方針に比べ模倣に要する時間を節約することができる. また, それらの実行に必要なレジスタ値を知らないノードにはそのレジスタ値を更新したノードが更新後に直接送信する.

[模倣方針]

(a) (t_i のガードの真偽値判定後) t_i の入出力イベントを実行したノード (これを t_i の責任ノードと呼び $\mathrm{Rnode}(t_i)$ で表す) は次トランジション t_j の責任ノード ($\mathrm{Rnode}(t_j)$) に t_i の入出力イベントの実行終了の通知 (type-1 メッセージ) を送信する. t_j の入出力イベントは各 t_i の type-1 メッセージを受信後に実行する.

(b) $\mathrm{Rnode}(t_i)$ は t_i の入出力イベントの実行後, t_i のレジスタ更新文を実行するノードに t_i の入出力イベントの実行終了の通知 (type-2 メッセージ) を送信する. レジスタ更新文は t_i の type-2 メッセージ受信後に実行する. ただし, 更新に入力変数の値が必要な場合はそれをメッセージに含めておく.

(c) 各トランジション t_j の出力イベントやガードの真偽値判定, レジスタ更新文を実行するノードがその実行に必要なレジスタ R_q の値を持っていない場合があり得る. それらの値は, t_j から見て最後に R_q を更新したトランジション t_i から t_j までの複数トランジションの模倣中に R_q を保持しているノードが送信する (type-3 メッセージ). そのようなノードが複数ある場合には送信先のノードへの通信路の最大遅延時間が最も短いノードが送信する. □

3.2 分散実行可能性の自動判定法

本節では, 前節で与えた模倣方針に基づき, 全体仕様の時間制約を満たしながら全体仕様と同様の動作を各ノードで協調実行が可能かどうかを判定する方法を述べる. ここで, メッセージの送信及び受信は, (それが可能となった時刻に) 即時に行うものとする.

ここで, トランジション t_i の責任ノード $\mathrm{Rnode}(t_i)$ において, (type-1 メッセージを受信してから) t_i の入出力イベントを実行するまでの時間の上限及び下限を表す非負変数 $ETmax(t_i)$, $ETmin(t_i)$ をそれぞれ導入する. また, t_i で値を更新される各レジスタ R_q を保持する各ノード z において, (type-2 メッセージを受信してから) レジスタ R_q を更新するまでの時間の上限及び下限を表す非負変数 $RTmax(R_q, t_i, z)$, $RTmin(R_q, t_i, z)$ をそれぞれ導入する. 基本的には,

(a) (メッセージ交換に必要な遅延時間を考慮した場合), 各ノードは入出力イベントやレジスタ更新文をいつまでに実行しなければならないかを 上記の非負変数上の線形不等式として与える.

(b) それらの線形不等式をすべて満たす解が存在するかどうかを, 線形計画問題の解法を用いて調べる. 解があれば分散実行可能であると判断する.

ここで, 以下のアルゴリズムでは, 各メッセージはそれが送受信される通信路の最大遅延時間が経過した時点で受信されるものとする. 一般に各メッセージが送受信されるのに必要な遅延時間は一定ではないが, 各メッセージにそのメッセージが送信された時刻の時刻印を与えることによりそのメッセージを受信したノードはメッセージに含まれた時刻印と自ノードでそれを受け取った時刻とを比較して通信に要した遅延時間を知ることができる. したがって, 最大遅延時間より短い遅延時間で到着したメッセージは, 送信されてから最大遅延時間経過後に受信可能となるものとして扱う. ただし, 前提条件 2 よりノード間のクロックのずれが存在するため, 実際には最大遅延時間経過 $\pm\delta$ の時間内に受信される.

なお, 表記の簡単のため, 全体仕様の各トランジション t_j に対し以下で表される変数 $Tmin(t_j)$, $Tmax(t_j)$ を導入する.

$$\begin{aligned} Tmin(t_j) &= \mathrm{Delay}(w, v) + ETmin(t_j) \\ Tmax(t_j) &= \mathrm{Delay}(w, v) + ETmax(t_j) \end{aligned}$$

(ただし $v = \mathrm{Rnode}(t_j)$, w は t_j の前トランジションの責任ノードのうち, v への最大遅延時間が最大であるノード)

以下は各非負変数 $ETmax(t_j)$, $ETmin(t_j)$ ($Tmax(t_j)$, $Tmin(t_j)$), $RTmax(R_q, t_i, z)$, $RTmin(R_q, t_i, z)$ に対する線形不等式の与え方である.

[入出力イベントの実行時刻に関する制約]

- t_j の前トランジションの入出力イベント実行終了を通知する type-1 メッセージの送受信に必要な最大遅延時間 $\mathrm{Delay}(w, v)$ やノード間のクロックの最大誤差 δ を考慮しても, t_j の入出力イベントは $\mathrm{Eft}(t_j)$ から $\mathrm{Lft}(t_j)$ の間で実行されなければならない. したがって, 各トランジション t_j と t_j の責任ノード v ($= \mathrm{Rnode}(t_j)$), t_j の前トランジションの責任ノードのうち, v への最大遅延時間が最大であるノード w に対し,

$$\begin{aligned} \mathrm{Eft}(t_j) &\le ETmin(t_j) + \mathrm{Delay}(w, v) - \delta \\ &\le ETmax(t_j) + \mathrm{Delay}(w, v) + \delta \\ &\le \mathrm{Lft}(t_j) \end{aligned}$$

ただし $w = v$ なら

$$\mathrm{Eft}(t_j) \le ETmin(t_j) \le ETmax(t_j) \le \mathrm{Lft}(t_j) \qquad (1)$$

[レジスタ値更新の実行時刻に関する制約]

全体仕様のペトリネット上のトランジション t_i から t_j への有向路上のトランジションの並び (系列) のうち, 先頭と後尾以外に同じトランジションを含まない系列を単純系列と呼び, σ_k で表す. また, σ_k に含まれるトランジションの数を $|\sigma_k|$ で, σ_k の x 番目に出現するトランジションを $\sigma_k(x)$ でそれぞれ表す.

- 各トランジション t_j と t_i のレジスタ更新文で値を更新される各レジスタ R_q, R_q を保持する各ノード z に対し, 定義より

$$RTmin(R_q, t_j, z) \leq RTmax(R_q, t_j, z) \quad (2)$$

- t_j のレジスタ R_q の更新文は, t_j のイベントの実行終了を通知する type-2 メッセージを受信後に実行し, 更新後の値をそれを必要とするノードに type-3 メッセージとして送信する. このとき, type-2 及び type-3 メッセージの送受信に必要な最大遅延時間とノード間のクロックの誤差を考慮してもあるノード y において R_q の値が用いられる最も早い時刻に間に合うように実行されなければならない. したがって, そのような各 t_j と R_q, R_q を用いるトランジションのうち, t_j からの単純系列 σ_k ($r = |\sigma_k|$ とおく) が存在するようなトランジション t_h, (t_j における) R_q の更新文実行後の値を送信するノード z, $v = \mathrm{Rnode}(t_j)$, $y = \mathrm{Rnode}(t_h)$ であるノード v, y に対し,
  - t_h が R_q をガードに用いるなら, $y' = \mathrm{Rnode}(\sigma_k(r-1))$ として,

$$\sum_{2 \leq x \leq r-1} Tmin(\sigma_k(x)) + \mathrm{Delay}(y', y) - \delta > \mathrm{Delay}(v, z) + RTmax(R_q, t_j, z) + \mathrm{Delay}(z, y) + \delta \quad (3)$$

    左辺は (t_j の入出力イベントがノード v で実行された時刻から) t_h のガードの真偽値がノード y で判定される時刻までの最小時間を, 右辺は (t_j の入出力イベントがノード v で実行された時刻から) ノード z で t_j の更新文における R_q の値更新を実行し, 更新後の値をノード y に送信するまでの最大時間をそれぞれ表す.
  - 同様に t_h が R_q を出力イベントに用いるなら,

$$\sum_{2 \leq x \leq r} Tmin(\sigma_k(x)) - \delta > \mathrm{Delay}(v, z) + RTmax(R_q, t_j, z) + \mathrm{Delay}(z, y) + \delta \quad (4)$$

  - 同様に t_h のレジスタ $R_{q'}$ の更新文にレジスタ R_q を用いるなら, ノード y' をレジスタ $R_{q'}$ を保持する各ノードとして

$$\sum_{2 \leq x \leq r} Tmin(\sigma_k(x)) + \mathrm{Delay}(y, y') + RTmin(R_{q'}, t_h, y') - \delta > \mathrm{Delay}(v, z) + RTmax(R_q, t_j, z) + \mathrm{Delay}(z, y') + \delta \quad (5)$$

- 各レジスタに R_q と R_q の更新文を持つトランジション t_j, t_j からの単純系列 σ_k が存在しかつ R_q の更新文を持つトランジション t_h, R_q を保持する各ノード z において, t_j のレジスタ R_q の更新文は, t_h のレジスタ R_q の更新文より前に実行されなければならない. したがって, そのような各レジスタ R_q, トランジション t_j と t_h, ノード z, 単純系列 σ_k ($r = |\sigma_k|$ とおく), ノード $v = \mathrm{Rnode}(t_j)$ に対し,

$$\sum_{2 \leq x \leq r} Tmin(\sigma_k(x)) + RTmin(R_q, t_h, z) - \delta > \mathrm{Delay}(v, z) + RTmax(R_q, t_j, z) + \delta \quad (6)$$

- t_j のレジスタ更新文で値を更新され, かつ同一ノード z に配置されているレジスタ R_q, $R_{q'}$ に対し, $R_{q'}$ が その更新に R_q の値を用いるのなら, $R_{q'}$ の値更新に用いる R_q の値は更新前のものでなければならない. したがって,

$$RTmin(R_q, t_j, z) \geq RTmax(R_{q'}, t_j, z) \quad (7)$$

(1) - (7) から得られる線形不等式をすべて満たす解を線形計画問題の解法を用いて求め, 解が存在するのであれば, 上述の模倣方針のもとで分散実行可能である. なお, 動作記述全体として動作 (入出力イベント及びレジスタ値更新) の時間制約をなるべく全体仕様の時間制約に近付け, 全体としてそれらの実行時刻の自由度の高い動作記述を得るために, 以下で与える目的関数 OBJ を最大とするような変数の解を求めることにする. OBJ の各項に適当な係数を与えることで, 特定の動作に関する時間幅を大きくすることもできる.

$$OBJ = \sum_{t_i} (ETmax(t_i) - ETmin(t_i)) + \sum_{t_j} \sum_{R_q} \sum_{z} (RTmax(R_q, t_j, z) - RTmin(R_q, t_j, z))$$

(ただし t_j はその更新文でレジスタ R_q の値を更新する各トランジション, z はレジスタ R_q を保持するノードとする)

3.3 各ノードの動作記述の導出法

各ノード v は全体仕様のネット N をそれぞれ持つとし, これを N_v で表す. 基本的には (a) N_v の各トランジションのうち v がその責任ノードであるトランジション t_j に対し, t_j を, t_j のガードと入出力イベントのみを持つトランジションで置き換え, t_j の次トランジションの各責任ノードに type-1 メッセージを送信するトランジションをそれに付加する. また, t_j の前トランジション t_i を t_i の責任ノードからの type-1 メッセージを受信するトランジションに置き換える. これにより, 入出力イベントを順次実行するネットを得る. (b) v の保持する各レジスタ R_q と R_q の値をその更新文で更新する各トランジション t_k ごとに, type-2 メッセージの受信, R_q のレジスタ値更新, type-3 メッセージ送信, をこの順に実行するサブネットを追加する. これにより, type-2 メッセージを受信して必要なレジスタ値更新を行うネットを得る. (c) v へ送信される type-3 メッセージごとに, それを受信する単体のトランジションからなるサブネットを追加する.

得られたトランジションの発火時間として t_j の入出力イベントを実行するトランジションには $[ETmin(t_j), ETmax(t_j)]$, t_j の レジスタ R_q の更新文を実行するノード z のトランジションには, $[RTmin(R_q, t_j, z), RTmax(R_q, t_j, z)]$, メッセージを送受信するトランジションには $[0, 0]$ をそれぞれ与える. ただし, 複数の type-1 メッセージを受信後に t_j の入出力イベントを実行する場合はそれらの type-1 メッセージ間の最大遅延時間の差を受信時に調節できるように受信トランジションの発火時間を与える必要があるが, 詳細は文献 [11] 参照.

また, 各 N_v において, 手順 1 では置き換えられないトランジション (無置換トランジション) が存在する場合があるが, N_v がそのようなトランジションを含む場合は, N_v から無置換トランジションをあらかじめ除去したネット N'_v に対し上記の手順を適用する. 基本的には (a) ネット N_v は活性安全な自由選択ネットであるので, それを覆う活性安全な SM 成分と呼ばれる有限状態機械の組に分割し [1], (b) 分割した各状態機械に含まれる (無置換トランジション以外の) 同一トランジションを再合成し, (c) 合成して得たネットに含まれる無置換トランジションの入出力プレースを 1 つのプレースに合成することで除去する, ことによりネット N'_v を得ることができる. 詳細は文献 [11] 参照.

3.4 導出例

図 2 は図 1 の全体仕様 $Sspec$ をノード 1, 2, 3 からなる分散システムとして実現した場合の動作記述 ($Pspec_1$, $Pspec_2$, $Pspec_3$) である. 通信路の最大遅延時間関数とゲートとレジスタの配置指定には表 1 の Delay, 表 2 の Alloc($Sspec$) をそれぞれ用いる. また, 簡単のためノード間のクロックの最大誤差 δ は 0 とする.

各ノードは作業用レジスタ R_0 を持つものとし, レジスタまたは入力変数 u の値をレジスタ R_0 に格納しその格納後のレジスタ R_0 の値を返す関数 $put(R_0, u)$, レジスタ R_0 に格納されているレジスタ R_h (入力変数 x) の最新の値を返す関数 $get(R_0, R_h)$ ($get(R_0, x)$) をプリミティブ関数として用いる. また, 各メッセージは他のメッセージと混同されないようメッセージ ID を持つも

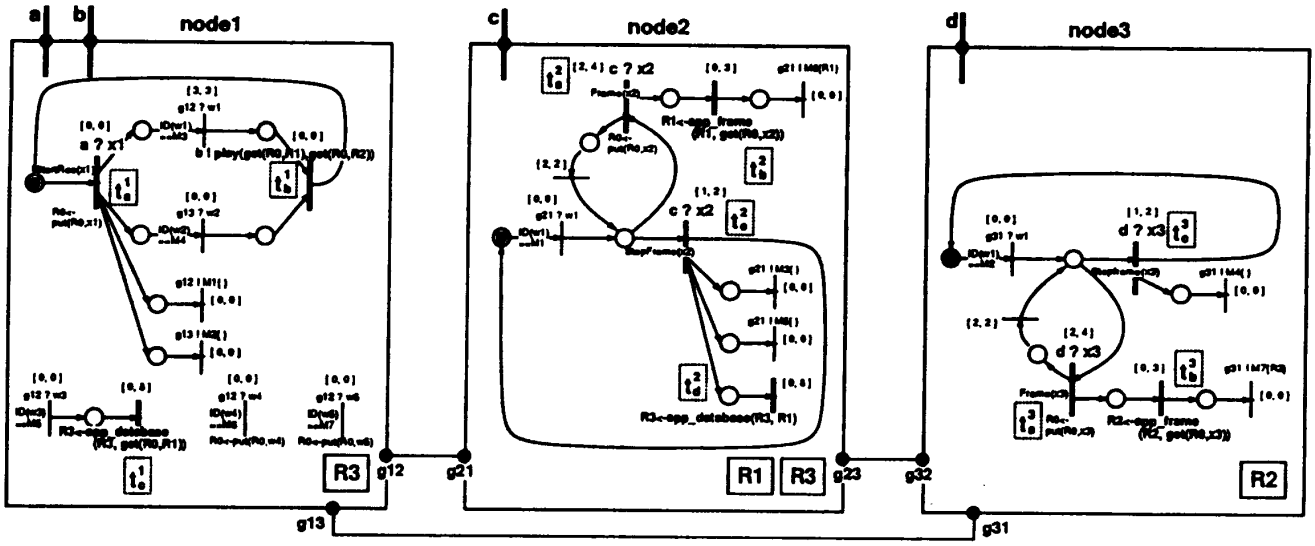

図 2: 導出した動作記述 $\langle Pspec_1, Pspec_2, Pspec_3 \rangle$ の例

のとし, 入力 w に含まれているメッセージ ID を返す関数 ID(w) もプリミティブ関数として用いる.

図 2 において, 太線で表されるトランジションは, 全体仕様において指定された入出力イベント及びレジスタ更新を行うトランジション, 細線で表されるトランジションは他ノードへのメッセージ送信あるいは他ノードからのメッセージ受信に用いられるトランジションである. ここで, 例えばノード 1 のトランジション t_b^1 はもとのトランジションが発火時間 [3, 5] であることと, 前トランジションの責任ノード (ノード 2 とノード 3) からの type-1 メッセージ M3 と M4 に要する最大遅延時間が 2 と 5 であることを考慮して, M3, M4 の受信後から時間 [0, 0] で発火する. また, ノード 3 の t_b^3 は type-3 メッセージ M7 に要する遅延時間 5 を考慮しても (t_a^3 が発火した時刻から) ノード 1 が t_b^1 でレジスタ R_2 の更新後の値を必要とする時刻に間に合うように時間 [0, 3] で発火する.

4 あとがき

本稿では, レジスタを持つ時間ペトリネットモデル (TPNR モデル) を提案し, そのモデルで記述された分散システムの全体仕様と各ノードへのリソースの配置指定及びノード間の通信路の最大遅延時間, ノードが持つクロック間の最大誤差が与えられたときに, 与えられた時間制約を満たしながらその分散システム上で分散実行可能であるかどうかを判定し, 可能である場合は分散システムの各ノードの動作記述を導出する手法を提案した.

提案した手法は, 並列動作の同期や非決定的選択動作を含みかつ並列動作によるリソースの同時アクセスが起こり得る全体仕様からの自動導出が可能であり, リソースが各ノードに分散配置されたような分散システムを対象に各ノードの動作記述を自動導出できる. このため, 従来より広いクラスのリアルタイム分散システムの全体仕様を対象に自動導出が行える. また, 導出した各ノードの動作記述は, 入出力動作を実行するネットと, 各レジスタの値の更新を実行するネットの間に依存関係が少ない (並列度が高い) ため, マルチスレッドなどの機構を用いた実装なども容易である.

今後の課題は, 実用的な例題に提案した手法を適用することで本手法の有効性を確認することなどである.

参考文献

[1] Murata, T. : "Petri Nets: Properties, Analysis and Applications," *Proc. of the IEEE*, Vol. 77, No. 4, pp. 541-580 (1989).

[2] Probert, R. and Saleh, K. : "Synthesis of Communication Protocols: Survey and Assessment," *IEEE Trans. on Computers*, Vol. 40, No. 4, pp. 468-476 (1991).

[3] Kant, C., Higashino, T. and Bochmann, G. v. : "Deriving Protocol Specifications from Service Specifications Written in LOTOS," *Proc. of 12th Int. IEEE Phoenix Conf. on Computers and Communications (IPCCC'93)*, pp. 310-318 (1993) (an extended version will appear in *Distributed Computing*).

[4] Milner, R. : "Communication and Concurrency," *Prentice-Hall* (1989)

[5] Merlin, P. M. and Farber D. J.: "Recoverability of Communication Protocols Implications of a Theoretical Study," *IEEE Trans. on Communications*, Vol. COM-24, pp.1036-1043 (1976).

[6] Higashino, T., Okano, K., Imajo, H. and Taniguchi, K. : "Deriving Protocol Specifications from Service Specifications in Extended FSM Models," *Proc. of the 13th Int. Conf. on Distributed Computing Systems (ICDCS-13)*, pp. 141-148 (1993).

[7] Kopetz, H. : "Clock Synchronization in Distributed Real-Time Systems," *IEEE Trans. on Computers*, Vol. C-36, No. 8, pp.933-940 (1987).

[8] Khoumsi, A., Bochmann, G.v. and Dssouli, R. : "On specifying services and synthesizing protocols for real-time applications," *Proc. of the 14th IFIP WG6.1 Symp. on Protocol Specification, Testing and Verification (PSTV-XIV)*, pp.185-200 (1994).

[9] Nakata, A., Higashino, T. and Kenichi, T. : "Protocol Synthesis from Timed and Structured Specifications," *Proc. of the Int. Conf. on Network Protocols (ICNP'95)*, pp. 74-81 (1995).

[10] Yamaguchi, H., Okano, K., Higashino, T. and Taniguchi, K. : "Synthesis of Protocol Entities' Specifications from Service Specifications in a Perti Net Model with Registers," *Proc. of the 15th Int. Conf. on Distributed Computing Systems (ICDCS-15)*, pp. 510-517 (1995).

[11] 山口弘純 : "レジスタ付き時間ペトリネットで記述された分散システムの全体仕様から時間制約を満たす各ノードの動作記述の自動導出," 大阪大学大学院基礎工学研究科修士学位論文 (1996).