時間ペトリネットの拡張モデルを用いたプロトコル合成

山口弘純† 岡野浩三†
東野輝夫† 谷口健一†

動作の実行時刻に制約のある分散システムのサービス仕様（システム全体が1計算機で実行されるとしたときのその計算機の仕様）から計算機間の通信遅延を考慮してもその制約を満足するプロトコル仕様（実際の計算機ごとの仕様の組）を自動合成する手法を提案する．提案手法では，多くのリアルタイムシステムの仕様記述，検証に用いられている時間ペトリネットに基づくモデルを用い，従来手法において必要とされていた計算機間での同期時計を必要としないプロトコル仕様を合成できる．また，状態変数を含むサービス仕様からの合成もできる．サービス仕様を満足するプロトコル仕様が存在するかどうかを判定することや，存在する場合にある評価基準のもとで最適なプロトコル仕様であるかどうかを保証することは容易ではないが，本手法ではそれらの問題を線形計画問題に帰着し，それらを機械的に判定，求解する工夫をしている．本手法を用いることで，計算機間の通信遅延を直接計算することなく仕様記述が行えると考えられる．

Protocol Synthesis in a Time Petri Net Model with Registers

HIROZUMI YAMAGUCHI,† KOZO OKANO,† TERUO HIGASHINO† and KENICHI TANIGUCHI†

In this paper, we propose a method for deriving a protocol specification of a real-time distributed system from a given service specification with time constraints. In our method, a protocol specification can be derived without synchronous clocks among protocol entities, using a time Petri net based model. In addition, a service specification with state variables can be also treated. In general, it is not easy to check whether protocol specifications satisfying the required services exist or not, and to get the optimal one if they exist. We formulate it to linear programming problems and get the optimal protocol specification. Using our method, designers do not need to consider communication delays among protocol entities directly, on specifying real-time distributed systems.

1. まえがき

複数の計算機（エンティティ）が互いに通信しながら協調動作を行うような分散システムの仕様記述では，他のエンティティとの協調通信やデータ伝送を含んだ仕様を各エンティティごとに記述する必要がある．しかし，それらを記述することは繁雑で誤りも多いと思われるため，従来よりプロトコル合成法と呼ばれる手法が提案されている[2]．プロトコル合成法では，分散システム全体を1つのエンティティと見なしたときのシステムの各動作とそれらの実行順序が指定された仕様（以下，サービス仕様とよぶ）を与える．この仕様から，各エンティティが実行すべき各動作と他のエンティティとの通信動作，それらの実行順序が指定された各エンティティの仕様の組（以下，プロトコル仕様とよぶ）が機械的に合成される．

図1にサービス仕様とプロトコル仕様の例を示す．システムはSAP（Service Access Point）とよばれるインタフェース a, b, c を持つ．また時間ペトリネットに基づく記述モデルを用いており，各トランジションが発火すると指定されたSAPを介した入出力動作が実行される．サービス仕様（図1(a)）ではシステムを1つのエンティティと見なしたときにそれがSAP a, b, c を介して実行する入出力動作とそれらの実行順序（入力動作 $a?$, $b?$ を実行後に出力動作 $c!$ を実行）を指定する．一方，プロトコル仕様（図1(b)）ではシステムを3つのエンティティ（PE_1, PE_2, PE_3）の組からなる分散システムと見なし，a, b, c をいずれかのエンティティに配置する．各エンティティPE_iから$PE_j(i,j=1,2,3, i\neq j)$へはそれぞれ通信路が

† 大阪大学大学院基礎工学研究科情報数理系専攻
Graduate School of Engineering Science, Department of Informatics and Mathematical Science, Osaka University

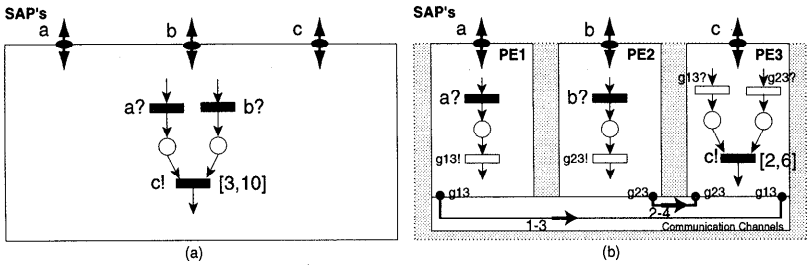


図1 サービス仕様とプロトコル仕様
Fig. 1 Service specification and protocol specification.

存在し，その両端をSAP g_{ij} で表す．PE_i は g_{ij} を介した出力動作 $g_{ij}!$ を，PE_j は g_{ij} を介した入力動作 $g_{ij}?$ をそれぞれ実行することで非同期通信を行う．プロトコル仕様では，各エンティティごとにそのエンティティに配置されたSAPを介した入出力動作と通信路のSAPを介した入出力動作（通信動作），それらの実行順序（PE_1，PE_2 は $a?$，$b?$ の実行後その実行終了を PE_3 に通知，PE_3 は それらの通知を受信後 $c!$ を実行）を指定する．

これまでにLOTOS[3]やペトリネット[4),7)]などさまざまな計算モデルに基づくプロトコル合成法が提案されている．しかし，それらの手法は動作の実行時刻に関する制約（動作の時間制約）を考慮していない．たとえば図1(a)のサービス仕様では，動作 $c!$ について“$a?$ と $b?$ の実行がともに終了してから3から10単位時間経過までに必ず実行されなければならない”といった時間制約（[3, 10] で表す）が与えられている．これに対し図1(b)のプロトコル仕様では，PE_3 は PE_1 と PE_2 からの通知にかかる遅延時間を考慮しても同様の時間制約を満たすように動作する必要がある．ここで，PE_1 からの $a?$ の実行終了通知にかかる遅延時間（1から3単位時間）を考慮した場合，PE_3 がその通知を受け取ってから $c!$ を実行するまでの時間制約を $[2(=3-1), 7(=10-3)]$ と決定できる．また，PE_2 からの $b?$ の実行終了通知にかかる遅延時間（2から4単位時間）を考慮した場合，$[1(=3-2), 6(=10-4)]$ と決定できる．PE_3 は $a?$ か $b?$ のうち後に実行を終了した動作に関する終了通知の遅延時間を考慮すればよいが，PE_1，PE_2 で実行されている動作 $a?$，$b?$ の終了時刻を正確に知ることができないため，どちらが後に実行された場合もサービス仕様の時間制約を満たすように [2, 6]（[2, 7] と [1, 6] の共通部分）と決定しなければならない．

近年，動作の時間制約が指定されたリアルタイム分散システムのサービス仕様から，エンティティ間の通信遅延時間を考慮してもサービス仕様の時間制約を満たすようなプロトコル仕様を自動合成する手法が提案されてきている[5),6)]．文献5)，6)ではそれぞれ計算モデルとして時間制約付きFSM，時間制約付きLOTOSを用いている．これらは非隣接動作間の時間制約も指定できる一方で，すべてのエンティティが正確に同期した時計（大域時計）を持つと仮定し，システムの状態変数を扱えないクラスにサービス仕様を制限するなどいくつかの仮定を置いている．

本論文では，(1) 時間ペトリネット（time Petri net）[1)]の拡張モデルで記述された分散システムのサービス仕様，(2) 各入出力ゲート（SAP）とレジスタ（状態変数）のエンティティへの配置指定，(3) エンティティ間の各通信路の最小および最大通信遅延時間（定数），が与えられたとき，一定の模倣方針のもとで，サービス仕様の時間制約を満たしながら模倣するような同モデルで記述されたプロトコル仕様を自動合成する手法を提案する．

時間ペトリネットはこれまで多くのリアルタイムシステムの仕様記述，検証に用いられてきた有用なモデルである．このモデルでは並列同期動作や選択動作などを含む複雑な制御構造が指定できる．また各動作の時間制約はすべての前動作の実行が終了してからの経過時間の上限と下限で指定する．本手法では時間ペトリネットの状態変数付き拡張モデルを用い，大域時計を必要としないプロトコル仕様を合成できる．また，従来手法では動作の実行順序の制御のみに着目していたが，本手法では入力値による状態変数値の更新動作を含むサービス仕様を効率良く模倣するプロトコル仕様を合成できる．

本手法では，(1) ある一定の模倣方針に基づいてまず時間制約なしのプロトコル仕様を構成し，(2) 次にその仕様の時間制約を決定する．(1) では入出力動作は図1(b)のようにその実行順に終了通知を次のエンティティに送信していくが，状態変数値はそれを保持しているエンティティが更新した直後の値を将来必要とする可能性のあるすべてのエンティティに前もって

先送りしておく模倣方針を採用する．一般に模倣方針はいくつか考えられるが，この模倣方針ではメッセージ数が増える可能性がある一方，状態変数値を必要とするエンティティになるべく早く渡すことができる．また (2) では，プロトコル仕様がサービス仕様を正しく実現できるように (1) で得られた仕様の時間制約を決定するが，正しく実現する時間制約が存在するか否か，あるいはそれが存在した場合に（ある評価基準で）最適な時間制約か否かは単純には判定できない．本手法ではプロトコル仕様の時間制約を決定する問題を線形計画問題に帰着し，その解法を用いて上述の実現方針のもとでそのような時間制約が存在するかどうかを機械的に判断し，存在する場合にはその幅の総和が最大である（なるべく緩い）時間制約を機械的に求める工夫もしている．

本論文の構成は以下のとおりである．2 章では記述モデルを定義する．3 章ではサービス仕様の例と合成したプロトコル仕様について述べる．4 章では本論文で扱う合成問題を定義し，与えられたサービス仕様に対するプロトコル仕様の正しさを定義する．5 章では正しいプロトコル仕様を合成するアルゴリズムについて述べ，6 章では結論と今後の課題について述べる．

2. 記述モデル

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 モデルとよぶ）はシステム外部とのやりとりが行われる入出力ゲートでの入出力動作とシステム内部の状態を保持する変数であるレジスタの値の更新動作をトランジション発火時の動作として指定できる．

TPNR モデルの各トランジション t はラベル $\langle$ ガード式，入出力動作式，レジスタ値更新動作式 $\rangle$ を持つ．また，時間制約 $[\mathrm{Eft}(t), \mathrm{Lft}(t)]$ を持つ．

入出力動作式は (a) “$a!Exp_1(...), Exp_2(...), ...?x, y, ...$”：入出力ゲート a への式 $Exp_1(...), Exp_2(...), ...$ の値の出力および a からの入力の入力変数 $x, y, ...$ への割当てを表す動作式（入出力動作式），(b) “i”：何も行わないことを表す内部動作式，のいずれかである．Exp_i はレジスタ変数を引数に持つことができる．入力変数はこのトランジション t のみが参照できる変数であり，レジスタ変数はすべてのトランジションで参照，更新できる変数である．ガード式はレジスタ変数とそのトランジションで有効な入力変数を引数に持つことのできる述語である．レジスタ値更新動作式は “$R_{h_1} \leftarrow f_{h_1}(...), ..., R_{h_m} \leftarrow f_{h_m}(...)$” または空文で与えられる．各関数 $f_{h_j}(1 \leq j \leq m)$ はレジスタ変数または t で有効な入力変数を引数に持つことができる．更新動作が実行されると，関数 $f_{h_1}(...), ..., f_{h_m}(...)$ の値が計算され，レジスタ $R_{h_1}, ..., R_{h_m}$ にそれぞれ同時代入される．空文はどのレジスタ値も更新しないことを表す．$\mathrm{Eft}(t)$，$\mathrm{Lft}(t)$（ともに非負有理数，$\mathrm{Eft}(t) \leq \mathrm{Lft}(t)$）はそれぞれ最小発火時間，最大発火時間を表す．$t$ は (a) t の各入力プレースに少なくとも 1 つのトークンがあり，(b) t のガードの値が真であるとき，発火可能であるという．t は発火可能となってから $\mathrm{Lft}(t)$ 単位時間経過までに発火に必要なトークンを失わない限り $\mathrm{Eft}(t)$ 単位時間経過後 $\mathrm{Lft}(t)$ 単位時間経過までに発火しなければならない．t が発火すると，t の入出力動作が実行され，次にレジスタ値更新動作が実行される．その後，マーキングがペトリネットの発火規則に従って更新される．発火に必要な時間はつねに 0 であるとする．

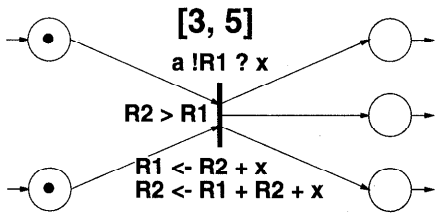


図 2 TPNR モデル
Fig. 2 TPNR model.

たとえばトランジション t のラベルが $\langle R_2 > R_1, a!R_1?x, “R_1 \leftarrow R_2 + x, R_2 \leftarrow R_1 + R_2 + x”\rangle$，時間制約が [3, 5] であるとする（図 2）．また現在のマーキングでのレジスタ R_1，R_2 の値はそれぞれ 1，2 であるとする．ある時刻 T に t の各入力プレースにトークンが存在すれば，ガードの真偽値が判定される．この場合その値は真であるため，t は時刻 T において発火可能となる．t は時刻 $T+5$ までに発火に必要なトークンを失わない限り時刻 $T+3$ から $T+5$ の間に必ず発火する．t が発火すると，入出力動作 “$a!R_1?x$” が実行され，その後レジスタ値更新動作 “$R_1 \leftarrow R_2 + x$” と “$R_2 \leftarrow R_1 + R_2 + x$” とが同時実行される．ここで入力値が 3 であるとすると，R_1，R_2 の値はそれぞれ 5（$= 2 + 3$），6（$= 1 + 2 + 3$）に更新される．

3. サービス仕様とプロトコル仕様

以下では，あるネットワークの異なるノード上に分散配置されているモジュール A，B，C の下位の共通モジュールとして位置し，ネットワーク上の経路への帯域幅割当てを行うモジュール X を例として用いる．ここで A はある 2 つのノードのアドレスの組を X に

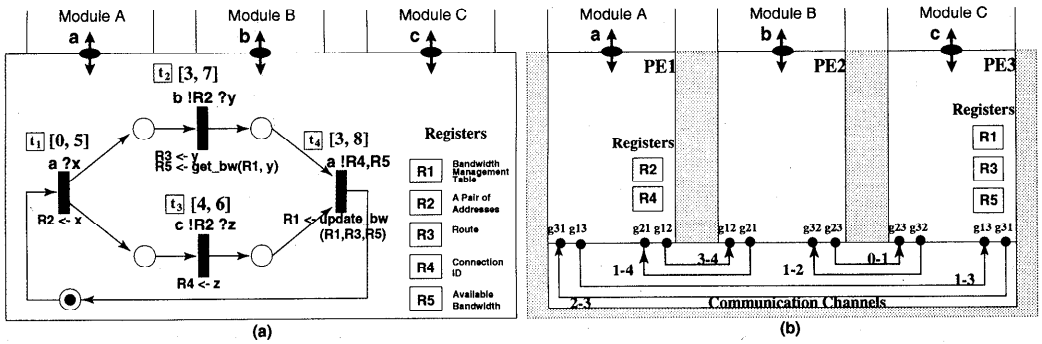


図 3 モジュール X のサービス仕様 $Sspec$ と Alloc($Sspec$), 各 $Dmin_{ij}/Dmax_{ij}$
Fig. 3 $Sspec$ for module X, Alloc($Sspec$) and each $Dmin_{ij}/Dmax_{ij}$.

渡し，その間に設定された経路の識別番号と経路に割り当てられた帯域幅を受け取るモジュール，B，C は X から渡されたアドレスの組に対しそれぞれその間の経路設定および経路の識別番号を決定するモジュールである．モジュール X はモジュール A から受け取ったアドレスの組をモジュール B，C にそれぞれ渡し，B からはそれらのノード間の経路を，C からはその経路に対する識別番号を受け取る．X は B から受け取った経路に対し，その経路に割当て可能な帯域幅を自身が保持する帯域幅管理テーブルを用いて計算し，その値を C から受け取った識別番号とともにモジュール A に渡す．その後，帯域幅管理テーブルを更新して初期状態に戻る．

3.1 サービス仕様

図 3 (a) に TPNR モデルで記述されたモジュール X のサービス仕様の例を示す．X と A，B，C とのインタフェースは入出力ゲート a，b，c で表される．また，X の帯域幅管理テーブルはレジスタ R_1，A，B，C からの入力（入力変数 x，y，z の値）を保持する変数はそれぞれレジスタ R_2，R_3，R_4，計算した帯域幅の値を保持する変数はレジスタ R_5 で表される．また，トランジション t_1，t_2，t_3，t_4 は A からの入力，B への入出力と帯域幅計算，C への入出力，A への出力と帯域幅管理テーブル更新をそれぞれ行う．なお帯域幅を計算する関数 “get_bw” および帯域幅管理テーブルの更新を行う関数 “update_bw” はその引数のみが分かっていればよく，具体的な動作はアルゴリズムに影響しない．なお，以降の図ではガード式 “true” と空のレジスタ値更新文は省略する．

本論文では $Sspec$ に対し以下の制約を仮定する．

(1) $Sspec$ 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☆．

(2) 同時発火可能である $Sspec$ のトランジション t_i と t_j の各組に対し，(a) t_i，t_j 上で同一レジスタ R の値が更新されるか，または (b) t_i 上でレジスタ R の値が更新され t_j 上で R の値が参照されるかのいずれかが成り立つとき，これをレジスタ競合とよぶ．$Sspec$ はレジスタ競合を含んではならない．

(3) $Sspec$ は内部動作 “i” を含んではならない．

ペトリネットは初期マーキング M_0 から可達な任意のマーキング M と各トランジション t に対し，t を発火可能とする M からの有限長の発火系列が存在するとき活性であるといい，M において各プレースのトークン数がたかだか 1 であるとき安全であるという．また，自由選択ネットは入力プレース（p とする）を共有する任意のトランジションの組に対し，それらが p 以外に入力プレースを持たないようなペトリネットである．自由選択ネットでは，選択構造があるときにどちらのトランジションが発火するかを p のトークンのみで決定できる．制約 (1) によりペトリネットの制御構造を簡単化でき，アルゴリズムを簡単化できる．制約 (2) は複数トランジションの同時実行による同一レジスタへの書き込みを禁止している[7]．制約 (3) は本質的ではないが，簡単のため仮定する．

3.2 プロトコル仕様

プロトコル仕様では，サービス仕様の動作は p 個のエンティティの協調動作として実現される．各エンティティ PE_i $(1 \le i \le p)$ にはサービス仕様のレジスタおよび入出力ゲートのいくつかが配置され，その配置指定を Alloc($Sspec$) で表す．また，各 PE_i から各 PE_j へは十分信頼できる通信路 $comm_{ij}$ が存在しその両端を入出力ゲート g_{ij} で表すとする．また，各 $comm_{ij}$ の最小および最大通信遅延時間はそれぞれ定数 $Dmin_{ij}$，$Dmax_{ij}$ で抑えられるとする．

☆ 自由選択ネットが活性安全であるかどうかを判定する多項式時間アルゴリズムが存在する[8]．

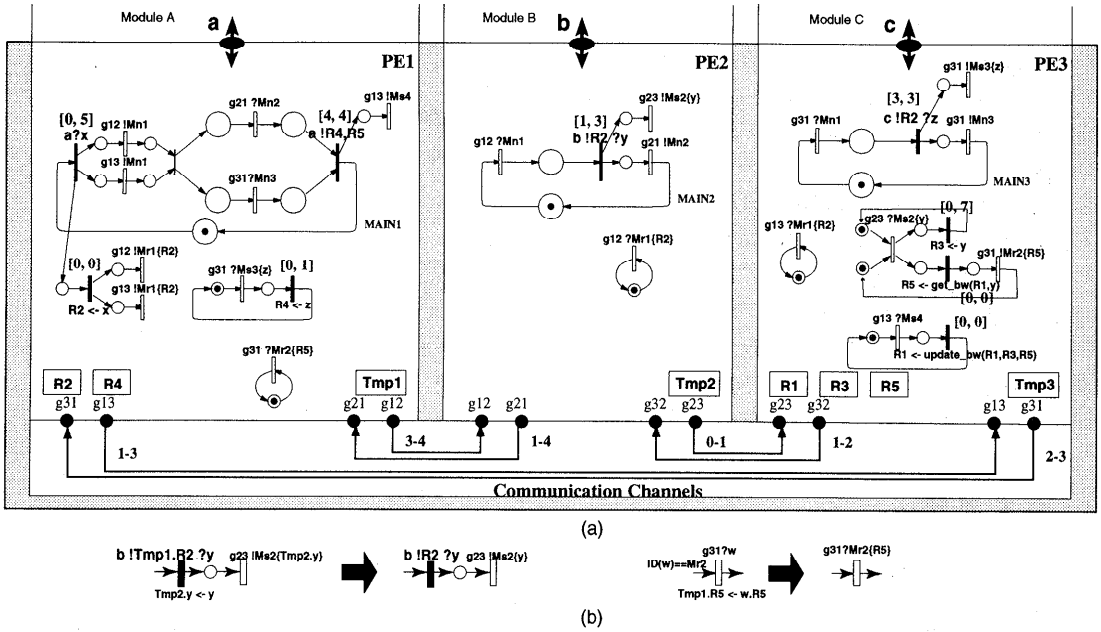


図4 合成したプロトコル仕様 $Pspec^{(1,3)}$
Fig. 4 $Pspec^{(1,3)}$.

ここで，モジュール X を3つのサブモジュール（エンティティPE_1, PE_2, PE_3）で実現することを考える．図3(b) に Alloc($Sspec$) および各 $Dmin_{ij}/Dmax_{ij}$ を示す．Alloc($Sspec$) では，入出力ゲート a, b, c をそれぞれ PE_1, PE_2, PE_3 に配置することで，PE_1, PE_2, PE_3 がそれぞれ A, B, C の下位モジュールとして実現されることを表している．

$Sspec$ の各トランジション t_i に対し，t_i の入出力動作が実行される入出力ゲートを保持するエンティティを t_i の責任エンティティとよび，RPE(t_i) で表す．本論文では前節の制約に加え，$Sspec$ と Alloc($Sspec$) に対し，以下の制約を仮定する．

(4) 入力プレース p を共有する $Sspec$ の t_i と t_j の各組に対し，Alloc($Sspec$) では RPE(t_i) = RPE(t_j) でなければならない．

本手法では各トランジションの責任エンティティがその発火に関する決定権を持つため，選択構造の場合には関係する責任エンティティの間でどのトランジションを発火させるかの合意をとる必要がある．制約(4)により，それを1つのエンティティに決定できる．

以下，PE_k の仕様を $Pspec_k$ で表す．また，$Pspec_1$, ..., $Pspec_p$ の組を $Pspec^{(1,p)}$ で表し，これをサービス仕様に対しプロトコル仕様とよぶ．図4(a)は図3(a)の $Sspec$ と図3(b)の Alloc($Sspec$) と各 $Dmin_{ij}/Dmax_{ij}$ から合成したモジュール X のプロトコル仕様 $Pspec^{(1,3)}$ である．本手法では送受信動作に要する時間は通信遅延時間に比べて十分小さいと仮定しそれを0としている☆．したがって，図4(a)では送受信動作を行うトランジション（白色のトランジション）の時間制約はすべて [0, 0] であり，それらは省略している．また，入力変数値や受信したレジスタ値は作業用レジスタを用いた値の保存が必要であり，メッセージを受信するトランジションは受信した値が受信すべきメッセージか否かの判定と，それに含まれる値を作業用レジスタに保存する動作が必要であるが，図4(a)では可読性を考慮し，それらに関して図4(b)のような略記表現を用いている．

各エンティティは互いにメッセージを交換しながらサービス仕様の動作を模倣する．たとえばトランジション t_1, t_2 の動作は以下のように模倣される．まず，PE_1 は，ゲート a を介してモジュール A から2つのノードのアドレスの組（入力変数 x の値）を受け取り，その入力動作の実行終了を PE_2 に通知する（メッセージ "Mn_1"☆☆）．また，受け取ったアドレスの組はレジスタ R_2 に格納し，その値も同様に PE_2 に通知する（メッセージ "$Mr_1\{R_2\}$"）．PE_2 は "$Mr_1\{R_2\}$" を受信し，受信した値を作業用レジスタ Tmp_2 に格

☆ 0と見なせない場合の解決は6章で述べる．

☆☆ メッセージ ID Mw_i は，$Sspec$ のトランジション t_i に関するタイプ w のメッセージ（w は n（入出力動作終了通知），s（入力値転送），r（レジスタ値転送）のいずれか）であることを表す．

納する☆．また，“Mn_1” を受信後，作業用レジスタ Tmp_2 に格納されているレジスタ R_2 の値をゲート b を介してモジュール B に出力し，2つのノード間の経路（入力変数 y の値）を受け取る．PE_2 は受け取った値を PE_3 に通知し（メッセージ “$\mathrm{Ms}_2\{y\}$”），PE_3 は受け取った y の値と帯域幅管理テーブル（レジスタ R_1）の値から割当てる帯域幅を計算し，その値をレジスタ R_5 に格納する．R_5 の値はモジュール A への出力動作（t_4 の出力動作）に用いられるため，PE_1 に通知される（メッセージ “$\mathrm{Mr}_2\{R_5\}$”）．

図4の $Pspec^{\langle 1,3\rangle}$ は図3(a)の $Sspec$ を正しく実現している．プロトコル仕様の正しさの定義は次章で述べる．

4. 合成問題

プロトコル仕様において，サービス仕様で用いられる入出力ゲート上の入出力動作（“$a?x$” など）のみを観測可能である動作とし，エンティティ間の送受信動作（“$g_{12}!\mathrm{Mn}_1$” など）や内部動作（“i”）を観測不可能な動作と見なす．今，サービス仕様とプロトコル仕様に対し，一方で実行可能かつ観測可能な入出力動作系列が（任意の連続する入出力動作間の時間間隔も含めて）他方においてもそうであれば両者は等価であるという．

一般には，与えられた $Sspec$ に対し，合成した $Pspec^{\langle 1,p\rangle}$ が $Sspec$ と等価であることが望ましい．しかし，たとえば**図5**(a)のように $Sspec$ で b の時間制約が [3, 10] であっても，図5(b)の $Pspec^{\langle 1,p\rangle}$ では通信遅延を考慮しなければならないため b は（a が終了してから）8から10単位時間経過後でないと実行できない（すなわち b は実質的に時間制約 [8, 10] で実行されていることに等しい）ように，エンティティ間の通信遅延を考慮した場合ほとんどの場合 $Sspec$ と等価であることは難しい．

ここで時間ペトリネットのトランジションの時間制約はすべての前トランジションが終了してからの経過時間の幅の指定であるため，先のように $Pspec^{\langle 1,p\rangle}$ での実質的な幅が狭くなっていてもそのトランジションが実行できなくなることはない．ただし，例外として選択構造の場合は実行できないトランジションが生じる場合もある．たとえば**図6**(a)のように互いに重なりのある時間制約を持つ2つのトランジションによる選択構造と図6(b)のように実質的な幅が狭くなり，それぞれ重なりを持たなくなったトランジションによる選択構造に対し，前者はいずれのトランジションも実行されるが後者は a が5単位時間経過までに必ず実行される必要があるため b は実行されない．また，並列に実行可能な2つのトランジションの実行順序の組合わせ数が減少することもある．たとえば “$a;b$”，“$b;a$” のいずれの順序でも観測される2つの並列動作 a，b の実質的な幅が狭くなった場合，“$a;b$” の順でのみ観測されるような場合もありうる．ここで，一般に選択動作は仕様記述者が明示的に指定するものであるため $Pspec^{\langle 1,p\rangle}$ での選択可能性も保証すべきであるのに対し，並列動作の時間制約により生じる実行順序関係は明示的な指定ではないため，特にこれに関する保証はする必要はないと考えられる．

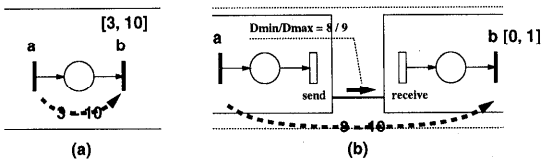


図5 時間制約幅の減少
Fig. 5 Decrease of time range intervals.

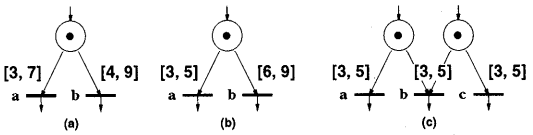


図6 選択構造
Fig. 6 Choice.

これらをふまえ，本論文では $Pspec^{\langle 1,p\rangle}$ の正しさを以下のように定義する．まず，$Sspec$ の時間制約幅を狭くして得られる仕様を $Sspec'$ で表す．$Sspec$ に対しある $Sspec'$ が存在し，(A) $Sspec'$ と $Pspec^{\langle 1,p\rangle}$ が等価であり，(B) $Sspec$ の選択構造で選択可能なトランジションは $Sspec'$ でも選択可能であるとき，$Pspec^{\langle 1,p\rangle}$ は $Sspec$ を正しく実現しているという．(A) は $Pspec^{\langle 1,p\rangle}$ における実質的な幅が通信遅延によって $Sspec$ の時間制約幅より狭くなっても正しい実現であることを意味する．(B) は選択構造において $Sspec$ で実行でき，$Pspec^{\langle 1,p\rangle}$ で実行できないトランジションが生じないようにしている．なお，図6(c)のように $Sspec$ が自由選択ネットでない場合，選択構造でのトランジションの実行可能性は互いに依存しない複数のトークンの入力プレースへの到着時間に依存するため，上述のように単純には判定できない．本手法では制約(1)で $Sspec$ のペトリネットを自由選択

☆ 各 PE_k は作業用レジスタを1つ保持するとし，これを Tmp_k とする．Tmp_k は他のエンティティから受信したレジスタや入力変数ごとその最新値を保持できるとする．Tmp_k に含まれる複数の値を区別するため，$Tmp_k.R_q$（$Tmp_k.x$）で Tmp_k が保持するレジスタ R_q（入力変数 x）の最新値を表すとする．

ネットに制限しているため，(B) を保証するための条件を簡単化できる．これについては 5.2 節で述べる．

本論文では制約 (1)，(2)，(3)，(4) を満たす $Sspec$ と Alloc($Sspec$)，さらに各 $Dmin_{ij}/Dmax_{ij}$ が与えられたとき，一定の模倣方針に従って $Sspec$ を正しく実現可能であれば $Sspec$ を正しく実現する $Pspec^{\langle 1,p\rangle}$ を合成し，そうでない場合は停止するアルゴリズムを提案する．なお，正しさの定義に従った場合，実質的な幅が不必要に狭いような $Pspec^{\langle 1,p\rangle}$ も認められるが，本論文では各動作の時間制約幅の総和が最大であるような $Pspec^{\langle 1,p\rangle}$ を合成する．

5. プロトコル仕様の合成アルゴリズム

合成アルゴリズムは，(手順 1) 時間制約を持たないプロトコル仕様（各エンティティの仕様群）をある一定の実現方針に基づき構成し（5.1 節），(手順 2) プロトコル仕様の各トランジションの時間制約を決定する (5.2 節)．

5.1 時間制約を持たないプロトコル仕様の合成

手順 1 では，与えられた $Sspec$ と Alloc($Sspec$) に対し以下に定める模倣方針に基づいて各エンティティがどの動作をどの順序で実行すべきかを一意に決定し，その決定から TPNR モデルで記述された時間制約なしのプロトコル仕様（$Pspec^{\langle 1,p\rangle}/ut$ で表す）を構成する．

[$Sspec$ の模倣方針]

(1) $Sspec$ のトランジション t_j に対し，RPE(t_j) は (a) すべての RPE(t_i)（ただし各 t_i は t_j の前トランジション）から t_i の入出力動作の実行終了を通知するメッセージ（入出力終了通知メッセージとよぶ）を受信後 t_j のガードの真偽値を判定し，(b)（真であれば）t_j の入出力動作を実行し，(c) 各 RPE(t_k)（ただし各 t_k は t_j の次トランジション）に入出力終了通知メッセージを送信するとともに，t_j の実行時に値が更新されるレジスタを保持する各エンティティに入力変数値を含むメッセージ（入力値転送メッセージとよぶ）をそれぞれ送信する．入出力終了通知メッセージにより入出力動作を $Sspec$ の実行順序通りに実行できる．なお，入力値転送メッセージはレジスタ値更新動作のきっかけも与えている．

(2) t_j で値が更新されるレジスタを保持する各エンティティは (a) RPE(t_j) からの入力値転送メッセージを受信すると，(b) t_j のレジスタ値更新動作を実行し，(c) 更新後のレジスタ値を含むメッセージ（レジスタ値転送メッセージとよぶ）をそれを将来のトランジションのガードの真偽値判定，入出力動作やレジスタ値更新動作の実行に必要とする可能性のあるすべてのエンティティに送信する．各エンティティは必要なレジスタ値のうちそのエンティティが保持しないものは，そのエンティティがレジスタ値転送メッセージにより受け取った最新の値を利用する．なお，必要なレジスタ値がそれらの動作の実行までに必ず到着していることは 5.2 節での時間制約の決定の際に保証する．

図 4 (a) の各エンティティにおけるネット MAIN-1, MAIN-2, MAIN-3（ただしこの段階では時間制約は決定されていない）は (1) を実現するネットである．たとえば，PE_2 (= RPE(t_2))，PE_3 (= RPE(t_3)) ではそれぞれ t_2，t_3 の入出力動作を実行後，メッセージ "Mn_2"，"Mn_3" を PE_1 (= RPE(t_4)) に送信する．PE_1 はそれらを受信後に t_4 の入出力動作を実行する．また，PE_2 は "Mn_2" の送信と同時に入力変数 y の値を含んだメッセージ "$Ms_2\{y\}$" をレジスタ R_3 と R_5 の値を更新する PE_3 に送信する．これらのネットはそれぞれ $Sspec$ のネットを利用して構成することができる．また，その他のネットは (2) を実現するネットである．たとえば PE_3 は "$Ms_2\{y\}$" を受信後 R_3 と R_5 の値を更新し，R_5 の値を含むメッセージ "$Mr_2\{R_5\}$" を PE_1 に送信する．PE_1 は "$Mr_2\{R_5\}$" を受信し，含まれている R_5 の値を作業用レジスタ Tmp_1 に格納する．

5.2 プロトコル仕様の時間制約の決定

手順 2 では，手順 1 で得られた時間制約なしのプロトコル仕様 $Pspec^{\langle 1,p\rangle}/ut$ のトランジションの時間制約を，$Sspec$ を正しく実現するように決定する．

$Pspec^{\langle 1,p\rangle}/ut$ は入出力終了通知メッセージにより $Sspec$ と同様の実行順序で入出力動作を実行できることのみが保証される．したがって，$Pspec^{\langle 1,p\rangle}/ut$ から正しい $Pspec^{\langle 1,p\rangle}$ を実現するには，

(a) 入出力動作系列が $Sspec$ で指定された時間間隔内で実行されること，

(b) それらの入出力値が $Sspec$ 通りであること，

(c) $Sspec$ において選択可能なトランジションが $Pspec^{\langle 1,p\rangle}$ でも選択可能であること（4 章参照），

を保証すればよい．しかし，メッセージの通信遅延時間を考慮した場合，条件 (a)，(b)，(c) を保証するような $Pspec^{\langle 1,p\rangle}$ の時間制約は存在しない可能性もある．たとえば，メッセージ "$Mr_2\{R_5\}$" は PE_1 が t_4 の入出力動作 "$a!R_4, R_5$" の実行に必要なレジスタ R_5 の値を含むが，このメッセージが t_2 のレジスタ値更新式 "$R_5 \leftarrow \mathrm{get\_bw}(R_1, y)$" の実行後すぐに送信

[条件 (a) を保証する不等式]

- $Sspec$ のトランジション t_j とその前トランジション t_i の各組に対し，

$$ETmin(t_j) \leq ETmax(t_j) \tag{1}$$

$$\mathrm{Eft}(t_j) \leq Dmin_{uv} + ETmin(t_j) \leq Dmax_{uv} + ETmax(t_j) \leq \mathrm{Lft}(t_j) \tag{2}$$

ただし $\mathrm{RPE}(t_i) = \mathrm{PE}_u$，$\mathrm{RPE}(t_j) = PE_v$ とする．

[条件 (b) を保証する不等式]

- $Sspec$ の各トランジション t_j と t_j のレジスタ値更新式で値を更新される各レジスタ R_q に対し，

$$RTmin(R_q, t_j) \leq RTmax(R_q, t_j) \tag{3}$$

- $Sspec$ の各トランジション t_i と t_i のレジスタ値更新式で値を更新される各レジスタ R_q に対し，R_q の値は将来実行されるトランジション t_j の入出力動作に必要であるとする．ここで $\mathrm{RPE}(t_i) = PE_u$，$\mathrm{RPE}(t_j) = PE_w$，レジスタ R_q は PE_v が保持するとする．このとき，

$$min\_sequence(t_i, t_j)_h \geq Dmax_{uv} + RTmax(R_q, t_i) + Dmax_{vw} \tag{4}$$

ここで，各 $min\_sequence(t_i, t_j)_h$ $(h = 1, 2, ..., r)$ は $Pspec^{\langle 1,p \rangle}$ における t_i の入出力動作実行から t_j の入出力動作実行までの入出力動作系列の（入出力終了通知メッセージの遅延を含めた）最小実行時間である．一般に t_i から t_j への入出力動作系列は 1 つではないため，それらの各々についてこの条件が成り立つ必要がある．なお，R_q の値が t_j の（R'_q の値を更新する）レジスタ値更新式に用いられる場合，式 (4) の左辺を $min\_sequence(t_i, t_j)_h + Dmin_{wz} + RTmin(R_q, t_j)$（$\mathrm{PE}_z$ は R'_q を保持するエンティティ）に，t_j のガード式に用いられる場合，$min\_sequence(t_i, t_j)_h - ETmin(t_j)$ のように変更する．

[条件 (c) を保証する不等式]

- 入力プレース p を共有する $Sspec$ のトランジション t_{j_a}，t_{j_b} と，p を出力プレースに持つトランジション t_i の組に対し，$\mathrm{RPE}(t_i) = \mathrm{PE}_u$，制約 (4) より $\mathrm{RPE}(t_{j_a}) = \mathrm{RPE}(t_{j_b}) = PE_v$ とする．もし $\mathrm{Eft}(t_{j_a}) \leq \mathrm{Lft}(t_{j_b})$ かつ $\mathrm{Eft}(t_{j_b}) \leq \mathrm{Lft}(t_{j_a})$ であるなら，

$$\begin{aligned} &ETmin(t_{j_a}) \leq ETmax(t_{j_b}) \\ &ETmin(t_{j_b}) \leq ETmax(t_{j_a}) \end{aligned} \tag{5}$$

[目的関数]

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

図 7 線形不等式と目的関数

Fig. 7 Linear inequalities and objective function.

されたとしても PE_1 における "$a!R_4, R_5$" の実行時刻に到着していない可能性がある（その場合 (b) は保証されず，正しい $Pspec^{\langle 1,p \rangle}$ は実現されない）．また，"$a!R_4, R_5$" が実行可能となる最も早い時刻を，条件 (a)，(c) を満たす範囲内で遅らせることが可能であれば "$a!R_4, R_5$" の実行までには必ず "$\mathrm{Mr}_2\{R_5\}$" が PE_1 に到着していることを保証できる可能性もある．

手順 2 では，手順 1 で得られた $Pspec^{\langle 1,p \rangle}/ut$ の各トランジションの時間制約を表す非負変数を導入し，(a)，(b)，(c) それぞれを保証するための十分条件をそれらの変数の線形不等式で表す．それらをすべて満たす解が存在すれば，得られた解をその時間制約として持つ $Pspec^{\langle 1,p \rangle}$ は $Sspec$ を正しく実現できる．存在しない場合は $Sspec$ を正しく実現する $Pspec^{\langle 1,p \rangle}$ は存在しないと判定する．以下，導入する変数と (a)，(b)，(c) を保証するための線形不等式について述べる．

[導入する変数]

- $Sspec$ の各トランジション t_j に対し，非負変数 $ETmin(t_j)$, $ETmax(t_j)$ を導入する．これらはそれぞれ $\mathrm{RPE}(t_j)$ がすべての前トランジションに関する入出力終了通知メッセージを受信してから t_j の入出力動作を実行するまでの最小および最大時間を表す．したがって，$\mathrm{RPE}(t_j)$ において t_j の入出力動作を実行するトランジションの時間制約は $[ETmin(t_j), ETmax(t_j)]$ で表される．
- $Sspec$ の各トランジション t_j とそのレジスタ値更新動作で値を更新されるレジスタ R_q の各組に対し，非負変数 $RTmin(R_q, t_j)$, $RTmax(R_q, t_j)$ を導入する．これらはそれぞれ R_q の値の更新動作を実行するエンティティが入力値転送メッセージを受信してから R_q の値の更新動作を実行するまでの最小および最大時間を表す．したがって，そのエンティティで t_j のレジスタ R_q の値更新動作を実行するトランジションの時間制約は $[RTmin(R_q, t_j), RTmax(R_q, t_j)]$ で表される．
- 3.2 節で述べたようにメッセージ送受信を実行するトランジションの時間制約はすべて $[0, 0]$ とする．

[線形不等式]

図 7 が (a)，(b)，(c) を保証する線形不等式である．

条件 (a) に対しては不等式 (1)，(2) が必要となる．

不等式 (1) は変数の定義より必要である．不等式 (2) は入出力終了通知メッセージの遅延時間を考慮しても入出力動作間の時間間隔は $Sspec$ で指定された時間制約内であることを保証する．

条件 (b) に対しては不等式 (3), (4) が必要となる．不等式 (3) は変数の定義より必要である．不等式 (4) はトランジション t_j の入出力動作の実行に必要なレジスタ値 R_q は PE_w がそれを実行する前に PE_w に到着していることを保証する．ここで，t_i の入出力動作が実行された時刻を T とする．t_j の入出力動作が実行される可能性のある最も早い時刻は $min\_sequence(t_i,t_j)_h$ $(h=1,2,...,r)$ で表される．これに対し，R_q の値が PE_w の到着する最も遅い時刻は $T+Dmax_{uv}+RTmax(R_q,t_i)+Dmax_{vw}$ で表される．したがって，もし不等式 (4) が成り立つなら R_q の値は必ず t_j の入出力動作の実行前には PE_w に到着していることが保証される．これがデータ転送実現のキーポイントとなる．

条件 (c) に対しては不等式 (5) が必要となる．$\mathrm{Eft}(t_{ja}) \le \mathrm{Lft}(t_{jb})$ かつ $\mathrm{Eft}(t_{jb}) \le \mathrm{Lft}(t_{ja})$ はトランジション t_{ja}, t_{jb} が $Sspec$ においてともに選択可能であることを意味する．不等式 (5) はトランジション t_{ja}, t_{jb} が $Pspec^{\langle 1,p\rangle}$ においてともに選択可能であることを保証する．

最後に，不等式 (1)～(5) をすべて満たす解を線形計画問題の解法を用いて求める．ここで，$Pspec^{\langle 1,p\rangle}$ の時間制約はなるべく緩い（時間制約幅が広い）ことが望ましい．したがって，$Pspec^{\langle 1,p\rangle}$ の時間制約幅の総和を表す関数 OBJ を目的関数とする．ここで OBJ の第1項は入出力動作の時間制約幅の総和を，第2項はレジスタ値更新動作の時間制約幅の総和をそれぞれ表す．そしてこの OBJ を最大とする解を得ることで全体としてその幅がなるべく広くなるような時間制約を求めることができる．

5.2.1 線形不等式の例

以下はモジュール X に対する線形不等式の一部である．

たとえば，$Sspec$ の t_4 とその前トランジションの1つである t_2 の組について，以下が成り立つ必要がある．

$$ETmin(t_4) \le ETmax(t_4)$$
$$\begin{aligned}\mathrm{Eft}(t_4) &\le Dmin_{21}+ETmin(t_4)\\ &\le Dmax_{21}+ETmax(t_4)\\ &\le \mathrm{Lft}(t_4)\end{aligned}$$

すなわち，

$$2 \le ETmin(t_4) \le ETmax(t_4) \le 4$$

さらに，トランジション t_2 とレジスタ R_5 に対し，更新後の R_5 の値は t_4 の入出力動作に用いられる．したがって，以下の不等式が成り立つ必要がある．

$$\begin{aligned}&Dmin_{21}+ETmin(t_4)\\ &\ge Dmax_{23}+RTmax(R_5,t_2)+Dmax_{31}\end{aligned}$$

この場合 $min\_sequence(t_2,t_4)$ はただ1つであり，$min\_sequence(t_2,t_4)_1 = Dmin_{21}+ETmin(t_4)$ である．t_2 の入出力動作が実行された時刻を T とすると，左辺は時刻 T から t_4 の入出力動作が実行される時刻までの最小時間，右辺は時刻 T から PE_1 に更新後の R_5 の値が到着する時刻までの最大時間である．この式により，t_4 の入出力動作は R_5 の値が到着した後に実行されることが保証される．なお，この式を簡単にすると

$$ETmin(t_4)-RTmax(R_5,t_2) \ge 3$$

となる．

6. あとがき

本論文では，TPNR モデルで記述された分散システムのサービス仕様 $Sspec$ とゲートとレジスタの配置指定 $\mathrm{Alloc}(Sspec)$，エンティティ間の各通信路の最小/最大通信遅延時間 $Dmin_{ij}/Dmax_{ij}$ が与えられたときに，同モデルで記述されたプロトコル仕様 $Pspec^{\langle 1,p\rangle}$ を自動合成する手法について述べた．

本手法では $\mathrm{Alloc}(Sspec)$ をアルゴリズムの入力としているが，プロトコル仕様がある評価基準に基づいて最適となるような $\mathrm{Alloc}(Sspec)$ をアルゴリズム中で決定することも考えられる．しかし，このような問題はネットワークのデータベース配置最適化問題などとして従来より多くの研究がなされている．そこで本手法では，それらの問題の解法を用いて得られる $\mathrm{Alloc}(Sspec)$ や設計者が指定した $\mathrm{Alloc}(Sspec)$ など任意の $\mathrm{Alloc}(Sspec)$ に対してプロトコル仕様を自動合成できるよう，$\mathrm{Alloc}(Sspec)$ を入力とするアルゴリズムとしている．

また，本手法では記述モデルとして大域変数（レジスタ）が扱えるモデルを用いているが，大域変数を用いることなく仕様記述を行いたい場合もある．その場合も各レジスタごとにその値を更新するトランジションはただ1つとし，更新された値はその次トランジションのみが利用できるとすることにより，擬似的に大域変数を用いないような仕様記述も行える．また，そのように記述されたサービス仕様に対してもプロトコル仕様を合成することができる．

最後に本手法では動作の実行にかかる時間を0としているが，現実には大容量のデータベースなどを検索

するのに必要な時間が無視できないなどの問題もある．本手法ではそういった場合も不等式 (4) を

$$min\_sequence(t_i, t_j)_h \geq Dmax_{uv} + RTmax(R_q, t_i) + max\_exec(R_q, t_i) + Dmax_{vw}$$

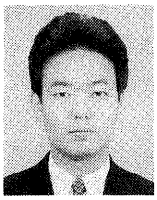
（ただし $max\_exec(R_q, t_i)$ は R_q のレジスタ値更新動作（データベース検索）実行の最大時間）のように少し変更するだけで対応できる．

今後の課題は，本手法を現実的なシステム設計例に適用し，手法の有効性を確認することなどである．

参考文献

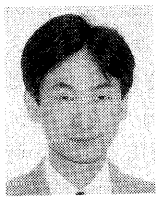
1) Murata, T.: Petri Nets: Properties, Analysis and Applications, *Proc. IEEE*, Vol.77, No.4, pp.541–580 (1989).
2) Saleh, K.: Synthesis of Communication Protocols: an Annotated Bibliography, *ACM SIGCOMM Computer Communication Review*, Vol.26, No.5, pp.40–59 (1996).
3) Kant, C., Higashino, T. and Bochmann, G.v.: Deriving Protocol Specifications from Service Specifications Written in LOTOS, *Distributed Computing*, Vol.10, No.1, pp.29–47 (1996).
4) Chao, D.Y. and Wang, D.T.: A Synthesis Technique of General Petri Nets, *Journal of System Integration*, Vol.4, pp.67–102 (1994).
5) Khoumsi, A., Bochmann, G.v. and Dssouli, R.: On Specifying Services and Synthesizing Protocols for Real-time Applications, *Proc. 14th IFIP WG6.1 Symp. on Protocol Specification, Testing and Verification (PSTV-XIV)*, pp.185–200 (1994).
6) 中田明夫，東野輝夫，谷口健一：時間制約の記述された LOTOS 仕様からのプロトコル合成，情報処理学会論文誌，Vol.37, No.5, pp.672–685 (1996).
7) Yamaguchi, H., Okano, K., Higashino, T. and Taniguchi, K.: Synthesis of Protocol Entities' Specifications from Service Specifications in a Petri Net Model with Registers, *Proc. 15th Int. Conf. on Distributed Computing Systems (ICDCS-15)*, pp.510–517 (1995).
8) Kemper, P. and Bause, F.: An Efficient Polynomial-Time Algorithm to Decide Liveness and Boundedness of Free-Choice Nets, *Proc. Int. Conf. on Application and Theory of Petri Nets 1992*, LNCS 616, pp.263–278 (1992).

(平成 9 年 6 月 30 日受付)
(平成 10 年 1 月 16 日採録)



山口 弘純（学生会員）

平成 6 年大阪大学基礎工学部情報工学科卒業．平成 8 年同大大学院博士前期課程修了．現在，同大大学院博士後期課程在学中．分散システムの設計法等の研究に従事．平成 8 年より日本学術振興会特別研究員．



岡野 浩三（正会員）

平成 2 年大阪大学基礎工学部情報工学科卒業．平成 5 年同大大学院博士後期課程中退．同年同大学情報工学科助手，現在に至る．工学博士．代数的手法によるプログラム開発，分散システムなどの研究に従事．電子情報通信学会会員．

東野 輝夫（正会員）

昭和 54 年大阪大学基礎工学部情報工学科卒業．昭和 59 年同大大学院博士課程修了．同年同大助手．平成 2，6 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．工学博士．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, ACM 各会員．

谷口 健一（正会員）

昭和 40 年大阪大学工学部電子学科卒業．昭和 45 年同大大学院博士課程修了．同年同大学基礎工学部助手，現在，同大大学院基礎工学研究科教授．工学博士．この間，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・検証法等に関する研究に従事．