マルチメディア通信と分散処理 64−12
（1994. 3. 3）

ネットワークの一部に対する通信ソフトウェア機能追加手法

奥山 浩伸[1] 森保 健治 平川 豊

NTT ソフトウェア研究所 ソフトウェア開発技術研究部

概要

通信ネットワークは大規模化にともない、各ノードに対して同時に機能改良を行なうことは困難になってきている。本稿ではネットワークの一部のノードのみに新規の機能が追加された時の異常動作検出方法について扱う。一部のノードのみが新規機能を追加されたネットワークにおいては、一般に新規機能を追加されたノードと新規機能を追加されていないノードの間の通信によって異常動作が起きる。

本稿では、機能の異なったノード間の通信を原因として起きる異常動作の基本的な性質を明らかにし、異常動作を検出するアルゴリズムを提案する。

A Design Method for Communications Software Evolution

Hironobu OKUYAMA Kenji MORIYASU Yutaka HIRAKAWA

NTT Software Engineering Laboratories

3-9-11 Midori-cho, Musashino-city, Tokyo 180, JAPAN

Abstract

Communication network is becoming large and it is very difficult to update each node function simultaneously. This article discusses a verification method for communication services on a network where a new service function is added on a part of network nodes. In the network where a part of nodes have updated function, communications between functionally updated nodes and an unupdated nodes may result illegal actions. The article clarifies fundamental properties of illegal actions and proposes an algorithm to detect them.

[1] E-mail:okuyama@sdesun.ntt.jp

1 はじめに

ネットワークの高機能化に伴ない、アプリケーションへの多様な要求に対する柔軟かつ迅速な対応が必要になっている。大規模ネットワークにおいては、多様な要求を満たすためには、ネットワークの一部グループにおける機能改良する方法が有効であると考えられる。

ところが、一部グループにおける機能改良を実現する場合、一つのネットワーク上に、新規に機能を追加したプロセスと機能を追加していないプロセスが混在することになる。このため、異なった機能を持つプロセス同士で通信が行なわれ、機能の相違を原因とする異常動作が起きる場合がある。

一方、異常動作を検出する手法として、形式的な検証手法 が研究されている。[3), 4), 5), 6)]

しかし従来の検証手法では、一つのネットワーク上に、異なった機能を持つプロセスが混在する場合を考慮していないため、直接適用することは困難である。

本稿では、一部グループのプロセスに対し機能追加を行なったとき、機能追加を行なったプロセスと機能追加を行なっていないプロセスの間での通信に起因する異常動作に関し、基本的な性質を明らかにし、異常動作を検出するアルゴリズムを提案する。

2 問題例

具体例として、ファクシミリ送信サービスを以下に挙げる。

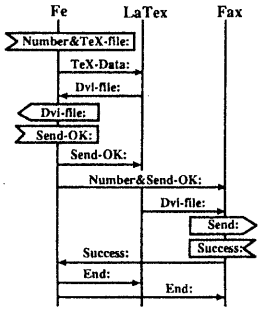

図 1: ファクシミリ送信サービス（既存機能 1）

ファクシミリ送信サービスとは、LaTeX 形式で作成されたファイルを端末からファクシミリで送信するサービスである。

ファクシミリ送信サービスの一部について、簡略化したものを図 1 および 2 に示す。図 1 の MSC が本サービスの正常系である。ここでは、最初にユーザがプロセス Fe に対し相手のファクスナンバー (Number) と LaTeX 形式のファイルを入力する。これをプロセス LaTeX が Dvi 形式のファイルにコンパイルし、ユーザに確認を求める。ユーザは、要求を満たす出力が得られることを確認し、信号 Send-OK: を入力することで送信を行なう。

また、図 2 の 2 枚の MSC は、一旦信号 Send-OK: を入力した後にユーザがキャンセルを要求する場合のシステムの動作を示す。このとき、ユーザからのキャンセル信号 Cancel: がプロセス Fax の送信処理中に間に合えば、送信の内容はキャンセル

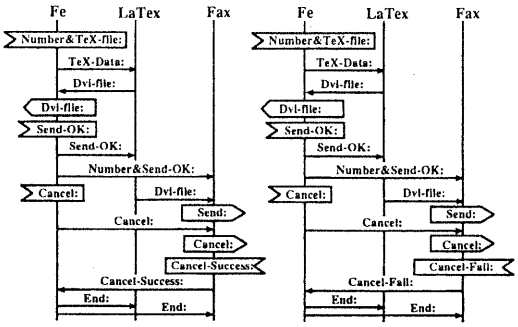

図 2: ファクシミリ送信サービス（既存機能 2）

され、信号 Cancel-Success: がユーザに返送される。しかし、既にプロセス Fax が送信処理を終えたあとに信号 Cancell: が着信した場合には、もはやサービスをキャンセルすることはできないため、信号 Cancel-Fail: がユーザに返送される。以上の仕様は異常動作を起こさない。

このサービスの運用中に、図 3 の仕様の新規機能を追加する場合を考える。

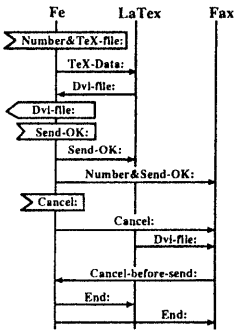

図 3: ファクシミリ送信サービス（新規機能）

この仕様はプロセス Fax において、信号 Cancel: が信号 Dvi-file: よりも早く着信する場合、信号 Send: を送信せずにキャンセルを行なう。このような仕様を加えることで、送信処理を開始せずにキャンセルを行なうことができる。

しかし、上記の機能追加がネットワークの一部グループのみに行なわれた場合、機能追加を行なっていないプロセスが、機能追加したグループ内にアクセスすることにより、機能の異なるプロセス間での通信となり（図 4 の太線）以下のような過程で異常動作が起きる。

Step1: 機能追加していないプロセス Fe で、 Cancel: を入力。

Step2: 機能追加しているプロセス Fax に Cancel: が Dvi-file: より早く着信。

Step3: Fax は Fe に対し Canceled-before-send: を送信。

Step4: 機能追加していないプロセス Fe は Canceled-before-

send: を受信できない（未定義受信）。

このように、ネットワークの一部グループのみに機能追加した場合、そのグループの外側のプロセスとの通信により、上の例で見たような異常動作を起こす可能性がある。以下本稿では、ネットワークの一部グループのみに機能追加を行なった場合に生ずる上記のような異常動作を検出する手法について述べる。

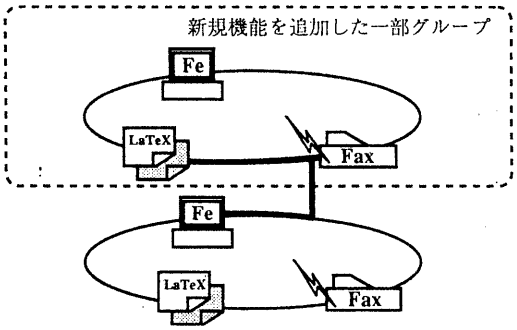

図 4: 一部グループのプロセスへの機能追加

3 システムモデル

3.1 システム

本稿におけるシステムは、有限状態機械でモデル化されたプロセスと、メッセージの送受信を行なう信頼性のある FIFO チャネルとで構成され、メッセージシーケンスチャート（以下 MSC）により与えられた仕様で動作定義がなされるものとする。

以下では、プロセス数を N、プロセス p_i $(1 \leq i \leq N)$ の状態の集合を Q_i、プロセス p_i の初期状態を o_i、プロセス p_i の最終状態の集合を $F_i = \{f_{i1}, ..., f_{i\tau(i)}\}$、プロセス p_i からプロセス p_j に送信するメッセージの集合を M_{ij} $(i \neq j)$、次状態関数を $succ$ とする。

定義 1（システム）

システムを $P = (Q, o, M, succ)$ の 4 項組で定義する。ここで $Q = Q_1 \cup ... \cup Q_N$, $o = (o_1, ..., o_N)$, $M = \bigcup_{i \neq j} M_{ij}$, $succ$ は各 i, j $(i \neq j)$ に対し、

$$Q_i \times (M_{ij} \cup M_{ji}) \rightarrow Q_i$$

の部分関数とする。ただし，各 i, j $(i \neq j)$ に対し $o_i \in Q_i, Q_i \cap Q_j = \phi$ である。

定義 2（グローバル状態）

グローバル状態を G=(S,C) とする。ここで，$S = (s_1, ...s_N), s_i \in Q_i \quad (1 \leq i \leq N), C = (c_{11}, ..., c_{NN}), c_{ij} \in M_{ij}^* \quad (1 \leq i, j \leq N, i \neq j)$ とする。

ここで M_{ij}^* は ϕ あるいは $M_{ij} \times M_{ij} \times ... \times M_{ij}$ を表す。

また $C = (\varepsilon, ..., \varepsilon)$ のとき G は安定であるという。ここで ε は空系列を表す。

$S^0 = (o_1, ..., o_N)$ かつ $C^0 = (\varepsilon, ..., \varepsilon)$ が成立するとき、$G = (S^0, C^0)$ を初期グローバル状態とよぶ。同様に

$S' = (f_{1\lambda(1)}, ..., f_{N\lambda(N)}) \quad (1 \leq \lambda(i) \leq \tau(i), i = 1, ..., N)$ かつ $C^0 = (\varepsilon, ..., \varepsilon)$ が成立するとき、$G = (S', C^0)$ を最終グローバル状態とよぶ。

定義 3（グローバル遷移）

グローバル状態 $G = (S, C)$ からグローバル状態 $G' = (S', C')$ への遷移を $G \vdash G'$ で表す。$G \vdash G'$ であるための必要十分条件は次の条件（1）あるいは（2）を満たすことである。ただしこれらの条件において、"." は系列の連結を表す。

（1）送信遷移 (s_i, x) が起こる、つまり $s_i' = succ(s_i, x)$, $c_{ij}' = c_{ij} \cdot x$, $x \in M_{ij}$ を満たす i, j, x が存在する。ここで $s_i, s_i', c_{ij}, c_{ij}'$ 以外の G と G' の要素は全て同じである。このような送信遷移によるグローバル状態の遷移を $G \vdash G'(s_i, x)$ と書く。

（2）受信遷移 (s_j, x) が起こる、つまり、$s_j' = succ(s_j, x)$, $c_{ji} = x \cdot c_{ji}', x \in M_{ji}$ を満たす i, j, x が存在する。ここで $s_j, s_j', c_{ji}, c_{ji}'$ 以外の G と G' の要素は全て同じである。

このような送信遷移によるグローバル状態の遷移を $G \vdash G'(s_j, x)$ と書く。

定義 4（到達可能性）

グローバル状態 G が到達可能であるための必要十分条件は、グローバル遷移系列 $G^0 \vdash G^1 \vdash G^2 \vdash \cdots G^{m-1} \vdash G^m = G$（$G^0$ は初期グローバル状態）を満たす $G^1, G^2 \cdots G^{m-1}$ が存在することである。このとき $G^0 \vdash^* G$ と書く。

定義 5（異常動作）

あるグローバル遷移系列 $G^0 \vdash^* G$ において、$G \vdash G'$ なるグローバル状態 G' が存在せず、G が最終グローバル状態でないとき、$G^0 \vdash^* G$ を異常動作と呼ぶ。また、異常動作 $G^0 \vdash^* G$ における G を異常停止状態と呼ぶ。

異常停止状態 $G = (S, C)$ において、C の $(i, j)-$ 成分 c_{ij} について $c_{ij} \neq \varepsilon$ であり、S の $j-$ 成分が s_j のとき、プロセス p_j は状態 s_j で未定義受信を起こしているという。

3.2 因果関係

一枚の MSC は一つの実行動作を表す。ここにおいて、送信・受信の各イベントは因果関係によって半順序をつけることができる。

ここでいう因果関係とは、Lamport のいう "happening before" の関係 $(\rightarrow)$ である[1]。これは以下のように定義される。

1. a と b が同一プロセス内のイベントで a が b より先に生起するならば a→b。
2. a があるプロセスにおける送信イベントであり、b が別のプロセスの 対応する受信イベントならば、a→b。
3. a→c かつ c→b ならば a→b。

a→b のとき、a は b の前に起こる、あるいは b は a の後に起こる、という。また、ある二つのイベント a、b が因果関係にないとは、$a \not\rightarrow b$ かつ $b \not\rightarrow a$ のときである。

4 異常動作の性質

4.1 用語の説明

以下では次のような記号、用語を使用する。

既存機能の仕様は MSC の集合 $\{MSC_1, ..., MSC_l\}$ で与えられるとし、これにより規定されるシステムを E と書く。

同様に、新規機能の仕様はMSCの集合$\{MSC_{l+1},...,MSC_m\}$で与えられるとし、既存機能の仕様にこれを加えた仕様$\{MSC_1,...,MSC_l,MSC_{l+1},...,MSC_m\}$により規定されるシステムをE+Aと書く。また一部のプロセスは既存機能を持ちその他のプロセスは新規機能を持つシステムの一つをE+(A)と書く。

既存イベント： 既存仕様に定義されているイベント

新規イベント： 既存仕様に定義されておらず、新規仕様にのみ定義されているイベント

新規トリガ： 既存イベントのみで到達できる状態から定義された新規受信イベント

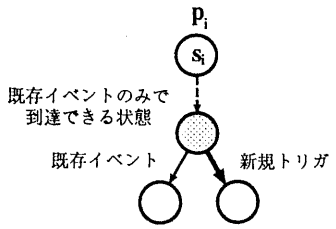

図5: 新規トリガ

4.2 仮定

与えられる仕様は以下の仮定を満たすものとする。

(仮定1) Eは異常動作を起こさない。

(仮定2) E+Aは異常動作を起こさない。

(仮定3) Eおよび E+Aにおける任意の動作に対応するMSC記述が存在。

(仮定4) プロセスは受信イベントでのみ分岐する。

(仮定5) ループは含まない。

4.3 諸性質

【性質1】

前節の仮定のもとで、E+(A)において異常動作が起こるならば、その停止状態では少なくとも一つの未定義受信が起きている。

[証明]

E+(A)において異常動作が起こり、その異常停止状態では未定義受信が起きていないと仮定し矛盾を導く。いまE+(A)の異常停止状態を$G=(S,C),S=(s_1,...,s_N),C=(\varepsilon,...,\varepsilon)$とする。ただし、$s_1,...,s_N$の少なくとも一つは最終状態でない。このとき、E+(A)において$G_0 \vdash G$であれば、E+Aにおいても$G_0 \vdash G$であるから、グローバル状態GはE+Aでも起こり得る。ところがチャネルは$C=(\varepsilon,...,\varepsilon)$であるから、$G$から他のグローバル状態への遷移は不可能。これは前節の仮定(2)に矛盾。□

【性質2】前節の仮定のもとで、E+(A)において未定義受信が起こるならば、その未定義受信の起こる状態で、E+Aにおいては新規トリガが定義されているものが少なくとも一つ存在する。

[証明] 異常停止状態$G=(S,C)$においてシステムが停止し、Gのなかで未定義受信を起こした全ての状態で、E+Aにおいて新規トリガが定義されていない、と仮定して矛盾を導く。

定義より、E+(A)においてGから遷移できるグローバル状態は存在しない。このとき一般に、E+AにおいてGから遷移できるグローバル状態が存在するとすれば、それは新規トリガによる遷移である。なぜなら、E+AとE+(A)の到達可能木を比較すれば、その差分はE+(A)に定義されていない新規イベントによる遷移であり、またその新規イベントの定義されているプロセスは、E+(A)においては新規機能を追加されていなかったプロセスであるから、Gからの遷移を引き起こす新規イベントは新規トリガである。

ところがここで仮定より、Gにおける各状態には新規トリガは定義されていない。よってE+AにおいてもGから遷移できるグローバル状態は存在しない。しかしGは異常停止状態であるから、これは前節の仮定(2)に矛盾。□

性質1と2より、異常動作の存在を判定するには、E+Aにおける新規トリガの定義されている状態に対して、未定義受信が起こるかどうかを調べれば十分であることが分かる。

プロセスp_jにおける、新規トリガeを定義されている状態sについて、次を定義する。

ET(e)：p_jのFSMにおいて、新規イベントeの定義された状態sから定義された既存イベントにおける受信信号全体からなる集合。

QM(e,i)：QM(e,i)は以下の条件(1)、(2)を満たす信号messの集合。

(1) プロセスp_jがプロセスp_kからの信号messを受信するイベントe_rが存在し、イベントe_rはMSC_i上で$e \rightarrow e_r$となる最初の受信イベントである。

(2) イベントe_rに対応する送信イベントとプロセスp_jのイベントeとはMSC_i上で因果関係がない。

【性質3】

前節の仮定のもとで次が成り立つ。システムがMSC_iで動作しているとき、プロセスp_jがイベントeを実行する直前の状態で、受信チャネルの先頭に存在し得る信号の集合と、QM(e,i)の集合は一致する。

[証明]

MSC_iに従ってシステムが動作しているときに、p_jでイベントeが実行されない時、システムは、eより後に因果関係のつくイベントは実行することができない。よってプロセスp_jが受信できる信号は、プロセスp_jに信号messを送信するイベントをe_sとすると、$e \rightarrow e_s$なる因果関係がない場合である。

また、e_sに対応するプロセスp_jの受信イベントをe_rとすると、$e_r \rightarrow e$のとき、e_rの受信信号messは、p_jがeに到達するまでに消費されチャネルに残らない。よって$e \rightarrow e_r$のときのみ、信号はチャネルに残る。

このような条件を満たす信号の中で最も早くp_jが受信する信号がチャネルの先頭に来る信号であるから、その集合はQM(e,i)に一致する。□

【性質4】

前節の仮定のもとで、MSC_iの動作において、E+Aでプロ

セスpの新規トリガeの定義された状態sについて、次が成り立つ。

あるE+(A)においてsで未定義受信が起こる

$\Longleftrightarrow \exists i$ s.t. ET(e) $\cap$ QM(e,i) $= \phi$

［証明］

（$\Longleftarrow$）性質3より、

ET(e) $\cap$ QM(e,i) $= \phi$

とは、QM(e)は、状態sから定義された既存パスへの遷移時の全ての受信信号（既存トリガ）と、pのチャネルの先頭に来る信号とにおいて、共通の信号がないことを意味する。

すなわち、pが状態sで新規トリガの信号を受けてから、他の状態に遷移することができない。よって、新規トリガの信号により、状態sで未定義受信となる。

（$\Longrightarrow$）対偶として

ET(e) $\cap$ QM(e,i) $\neq \phi$

$\Longrightarrow$ E+(A)においてsで未定義受信が起きない

をしめす。

ET(e) $\cap$ QM(e,i) が空でなければ、この共通部分に属する信号により、pは状態sから次の状態に遷移できるので、状態sでは未定義受信は起きない。

性質1～4より、次が成り立つ。

【定理】

前節の仮定のもとで次が成り立つ。

あるE+(A)が異常動作を起こす

$\Longleftrightarrow$

MSC_i 上に新規トリガeの実行が記述されており

ET(e) $\cap$ QM(e,i) $= \phi$

が成立する。

［証明］

（$\Longrightarrow$）あるE+(A)が異常動作を起こすならば、その異常停止状態は性質1より少なくとも1つの未定義受信を含む。このときE+Aでは未定義受信を起こした状態をsとすると、性質2よりsには新規トリガが定義されている。この新規トリガをeとすると性質4より定理の右辺が成り立つ。

（$\Longleftarrow$）定理の右辺が成り立つとする。このとき性質4より、あるE+(A)において未定義受信が起こる。□

与えられた仕様に対し、上の定理に関する判定を行なえば、一部グループに機能追加を行なったとき異常動作が起きるかどうかが検出できる。

5 異常動作検出アルゴリズム

5.1 提案アルゴリズム

前章で明らかにした性質を利用した異常検出アルゴリズムの概要を示す。

Step1: 与えられたMSC仕様から各プロセスの動作仕様を生成する。

Step2: 生成された各プロセスの動作仕様とMSCを比較し、既存イベントと新規イベントの分類を行なう。このとき、新規トリガの集合Σが得られる。また、新規トリガe$\in \Sigma$の実行が記述されているMSCの集合をSM(e)とかく。

Step3: Σから元（新規トリガ）eを取り出す。eは、プロセスP_h上の状態sで定義されているイベントであるとする。

Step4: プロセスP_hの動作仕様から、状態sに定義されている既存トリガの信号を全て求める。（集合ET(e)の計算）

Step5: 新規トリガeの実行が記述されているMSCの集合SM(e)からMSC_iを取り出す。「QM獲得手順」（後に記載）を用いて、状態sでプロセスp_hの受信チャネルの先頭に存在し得る信号の集合を求める。（集合QM(e,i)の計算）

Step6: ET(e)とQM(e,i)の共通部分を求める。共通部分が空の場合には、「異常動作が起こる」と判定して停止。共通部分が存在し、SM(e)が空でなければ、Step5へ進む。SM(e)が空であれば、Step7へ進む。

Step7: Σが空のとき、「異常動作は起こらない」と判定して停止。Σが空でない時には、Step3へ進む。

全ての新規イベントeに対し、eが記述されている全てのメッセージシーケンスチャートにおいて未定義受信が存在しない時、そしてその時に限りこの仕様は異常動作を起こさない。

[QM獲得手順]

準備：MSC_iにおける、プロセスp_hの状態sの新規トリガeについて調べるとする。イベントeの実行時に、各プロセスからの受信チャネルの先頭に存在し得る信号を保存するために、初期値がnullであるサイズNの配列Hを用いる。また、探索用トークンの生成を制御するために、サイズがN × Nで初期値がnullの配列Iを用いる。また、生成された探索用トークンは、順番にリストTを作って管理するとする。

Step1: プロセスp_hについて、状態sから最終状態に向かって、以下の操作を行なう。操作は、最終状態に到達するまで行なう。

(Case1.1) プロセスp_fからの信号messの受信イベントに遭遇した場合：

H(f)=nullのとき、H(f)=messとし、そのイベントに○印を付ける。それ以外の場合、そのイベントに△印を付ける。

(Case1.2) プロセスp_fへの送信イベントに遭遇した場合：

I(h,f)=nullのとき、I(h,f)=genとし、プロセスf上の対応する受信イベントに×印を付け、そこに新たな探索用トークンを生成する。新たに生成したトークンはリストTの最後に登録する。

Step2: トークンのリストTが空の時にはアルゴリズムを終了する。空でない時には、先頭のトークンをリストから抜き取り、現在トークンが存在するイベント（プロセスP_d上とする）からその最終状態に向かって以下の操作を行なう。操作は最終状態に到達するか、あるいは×印のついたイベントに到達するまで行ない、到達した場合には共にトークンは消滅する。トークンの消滅後は、step2を行なう。

(Case2.1) 受信イベントに遭遇した場合：

そのイベントに×印を付ける。

(Case2.2) プロセス$P_f(f \neq h)$への送信イベントに遭遇した場合：

I(d,f)=nullのときI(d,f)=genとする。プロセスp_f上の対応する受信イベントに×印を付け、そこに新たな探索用トー

クンを生成する。新たに生成したトークンはリストTの最後に連結する。

(Case2.3) プロセス p_h への送信イベントに遭遇した場合：
対応する受信イベントが○印であれば、これを×印に置き換え、H(d)=none とする。また、対応する受信イベントが△印であれば、これを×印に置き換える。

以上の手順が完了した時点で、配列H(f)の各成分において、○印のついたイベントの信号全体の集合がQM(e,i)である。

この手順により、MSC_i 上において因果関係がイベントeより後につく全てのイベントに、もれなく×印を付けることができる。△の付くイベントの信号は、チャネルにはたまるが先頭に来ないことを意味する。

またこの手順は一つのイベントについてチェックの重複は最大でも2回におさえられているという性質を持つ。

5.2 アルゴリズムの評価

本問題の検証について、従来手法を利用した検出法と提案アルゴリズムによる検出法とについて、計算量のオーダを評価する。パラメータとしては、プロセス数 N、MSCの数 m、FSMの最大長 u を用いる。

本稿で議論している異常動作検出のためには、前節に示した提案アルゴリズムだけでなく、全てのプロセスが機能追加された場合（システムE+Aの場合）の動作を検証する必要がある。このために、文献3)の手法を用いるとすると、その手法の部分の計算量は $O(um^3N^3)$ となる。

次に、前節の提案アルゴリズムの部分の計算量を評価する。

Step1、Step2は、文献3)の手法により処理が終了する。

Step3、Step4の処理は、全ての状態に対する各分岐のチェックである。この計算量は（一つのプロセスのFSMの状態数）×（プロセス数）×（一つの状態からの分岐数）$= O(um \times N \times N)$。

Step5（QM獲得手順）計算量のオーダは、各イベントのチェックを行なうときの重複する回数は、最大でも2回であるから、一枚のMSC上のイベント数のオーダとなり $O(uN)$。これをStep6で最大m（MSCの数）回、Step7で最大N（プロセス数）回繰り返すから、$O(umN^2)$ である。

Step6におけるET(e)とQM(e,i)の比較は、QM(e,i)は配列Hの形で与えられ、送信したプロセスが p_h の信号は、Hの $h-$ 成分に格納されているとして計算する。このときET(e)の最大数がm(MSCの数)であるから計算量は $O(m)$。

これをQM獲得手順と同様に、Step6で最大m（MSCの数）回、Step7で最大N（プロセス数）回繰り返すから、$O(m^2N)$ である。

この結果から提案アルゴリズムの部分の計算量を求めると、$O(umN^2)+O(umN^2)+O(m^2N)$ より $O(um^2N^2)$。

異常動作検出のための処理全体の計算量は、文献3)の手法の部分と提案アルゴリズムの部分の合計であるから $O(um^3N^3)$ となる。

比較のために、文献3)の手法のみを基本として異常動作検出場合の計算量を算出する。もし、システムを構成しているプロセスのうち特定のもの（K個のプロセス）だけが、機能追加されていないことが想定される場合には、考えられるパターンは 2^K 個存在し、このそれぞれについて文献3)の検証を適用する必要がある。そのため計算量は $O(um^3N^32^K)$ となる（この場合、2^K は定数）。

しかし本稿で提案の手法では、システムを構成するN個のプロセスの全てについて機能追加がされているか否か確定しない状況においても動作を保証するものであり、これと同等の保証をするためには、$O(um^3N^32^N)$ の計算量が必要となる。

従来手法を利用すると一般に、プロセスへの新規機能の追加の組合せを全て検証しなくてはならないため、その計算量を考えると現実的ではない。一方、提案アルゴリズムにおける計算量は多項式オーダであり、また従来手法のオーダを越えないため、実用システムへの適用は可能であると考えられる。

6 おわりに

本稿では、ネットワーク上の一部グループで自由な機能改良を行なうために、機能の異なるプロセス間で通信が行なわれるときの異常動作の基本的性質及び仕様に関する異常動作の検出方法について述べ、その計算量に関する評価を行なった。

今後は、ネットワークの不特定の一部グループへの機能追加をn回繰り返すときの異常動作の性質、あるいはループを含む仕様における異常動作の性質等について検討していく。

参考文献

1) L.Lamport,"Time,Clocks,and the Ordering of Evvents in a Distributed System",Communications of ACM,21,7,pp.558-565 (1978).
2) CCITT:"Draft Recommendation Z.120 Message Sequence Chart (MSC)" (1992).
3) M.Itoh and H.Ichikawa, "Protocol Verification Algorithm Using Reduced Reachiability Analysis" Trans. of IECE of Japan, vol.E66,No.2 pp.88-93 (1983).
4) D.Brand and P.Zafiropulo, "On Communicating Finite-State Machines", Journal of the ACM,Vol.30,No.2,April, pp.323-342 (1983).
5) F.J.Lin,P.M.Chu,and M.T.Liu, "Protocol Verification Using Reachability Analysis: The State Space Explosion Priblem and Relief Strategies", ACM SIGCOMM (1987).
6) Y.Kakuda,Y.Wakahara and M.Norigoe, "An Acyclic Expansion Algorithm for Fast Protocol Validation", IEEE Trans. on Software Engineering ,Vol.14 ,No.8 ,pp.1059-1070 (1988).
7) 伊藤 市川 ," 並行プロセスを基本とした交換プログラム仕様の階層的検証法", 電子情報通信学会論文誌 Vol.J69-B No.5 pp.449-459 (1990).
8) 奥山 森保 平川 , " 通信ソフトウェアの機能改良手法 " 情報処理学会第47回全国大会,6E-1,(1993).