〔公　開〕

ＴＲ－Ｃ－０１２５

広域状態遷移に着目した通信サービス仕様の検証

佐藤　正和
Masakazu SATO

１９９５　１２．１

ＡＴＲ通信システム研究所

広域状態遷移に着目した通信サービス仕様の検証

佐藤 正和

1995 年 12 月 1 日

1 はじめに

通信サービスをソフトウェアで実現するにあたって，その信頼性を保つには，仕様記述の段階でその正当性が保証されることが望まれる．通信システムも他の計算機プログラムと同様，ある種の計算を実行する機械であるが，主として，1) 状態遷移を伴う複数の主体が，2) 信号の授受を通じて同期しながら，3) 全体として特定のサービスを実現する，点に特徴があると言える．従って，仕様記述法もこれらの特徴を反映したものが使われる．

本稿では，通信サービスの仕様記述法として 広域状態遷移ルール [1] と呼ばれる記法を用いた場合の検証法について述べる．

第 2 章では，通信サービス仕様の従来の記述法と検証技術全般について概観する．

第 3 章では，検証の対象として STR で書かれた仕様の可到達解析を取り上げ，そのアルゴリズムについて述べる．従来の仕様記述法においては，可到達解析は特定の少数プロセスを対象としたものであったが，プロダクションルールによって広域状態遷移を記述する場合には，一般に不特定多数個のプロセスの存在を考慮する必要がある．このとき，可到達状態を任意の複数のプロセスによって構成される到達可能な準広域状態として定義すると，可到達状態集合の大きさは初期状態として与えるプロセス数に依存する．このため可到達性を一般に保証するには，プロセス数を無限と仮定した解析を行なう必要がある．ここでは，システムを構成するプロセス数と可到達集合の大きさの関係を明らかにし，可到達状態集合の上限値の存在，およびその上限値を得るのに必要な最小のプロセス数を求める方法を示した．これによって，無限プロセスの存在を仮定するシステムの可到達性を有限プロセスのそれに置き替えて解析することが可能になった．

第 4 章では，第 3 章のアルゴリズムの高速化の手法を提案する．

2 通信サービス仕様の検証とは

2.1 検証項目

一般のプログラムで仕様の検証を行う場合，プログラムの停止性や入出力の関係が正しく計算されるかが主として問題とされるが，通信システムの場合にはこの他に，サービスの終了時に初期状態に戻ることや，デッドロックが起こらないこと，資源が公平に消費されることなどが問題になる．通信サービス仕様に関する検証項目としては，以下のようなものが考えられている．

- 可到達性

  ある状態 S から別の状態 S' へ遷移することが可能である．S として初期状態 S_0 が用いられることが多い．

- 活性

  弱い定義では，初期状態 S_0 から到達可能な全ての状態において，それ以降の遷移が存在すること．強い定義では，全ての状態に対して何度でも遷移が可能なこと．デッドロックにならないことを意味している．

- 有界性（安全性）

  特定の計算資源（データ等）が無限に使用されることがない．

- 可逆性

  初期状態 S_0 から到達可能な全ての状態 S から，その状態 S_0 へ到達可能である．必ず初期状態へ戻れることを保証するもの．

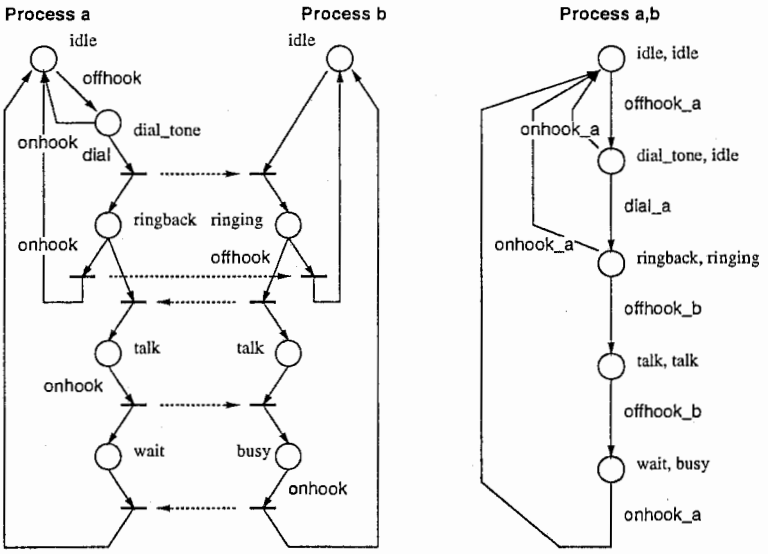
- その他

  その他の解析対象となる性質として，被覆性・パーシステンス・公平性，等がある．

検証の対象となる仕様は，各種の形式的な記述法で定義される．各々の記法によって記述可能な動作の範囲が異なり，利用者の目的や趣味によって使い分けられているのが現状である．従って，検証の方法は各記述法に応じて微妙に異なってくる．以下に，主な形式的仕様記述法の特徴を述べる．

- 有限状態オートマトン

  アルファベットとして入出力イベントを用いる有限状態オートマトン [2] は，通信システムを定義するうえで最も基本的な手法である．状態の表現法としては，個々の主体の状態遷移ごとに有限オートマトンを定義し，主体間の同期を信号の授受によって表す方法と，すべての主体の状態の直積に対して状態遷移を定義する方法が考えられる（図 1）．一般に，仕様定義には前者が用いられ，検証の際に後者（広域状態遷移）を生成することが多い．



a）主体ごとの状態遷移　　b）広域状態遷移

図 1: 有限状態オートマトン

有限状態オートマトンだけでは，遷移の条件やシステムのパラメータ等を表現するのが難しい．そこで，有限状態オートマトンに変数の記述を許した拡張有限状態オートマトンも用いられる．**SDL**[3] はこのような言語の一つで，現在，通信システムの仕様記述言語として最も良く用いられている．

有限状態オートマトンで記述できる範囲は，言語理論で言うところの正規表現のクラスである．汎用プログラムのモデルであるチューリングマシンよりは記述力が劣るので，タスク処理，システム制御等の複雑な計算は，有限状態オートマトンによる仕様とは別枠で記述される．

有限状態オートマトンで書かれた仕様の検証は，状態遷移グラフ上の生成・探索問題として定式化できる．しかし，個々の主体の状態遷移グラフから，広域状態遷移グラフを生成するには，主体の数 n に対して一般に $O(a^n)$ の計算量を必要とし，実際の検証の際に問題となることが多い．これに対して，状態を縮退させることで，広域状態の組合せを削減する方法等が提案されている．

- ペトリネット

  ペトリネットは，Petri によって提案され [4]，Peterson によって発展されせられた [5] 分散・並列システム記述のためのモデルである．ペトリネットは，有限状態オートマトンに主体を表すトークンと主体間の同期を表すトランジションを加えて，表現力を拡張したものと言える．この拡張により，有限状態オートマトンに比べて，複数主体間の同期を含む状態遷移を簡潔に記述することが可能である．

  ペトリネットは，許される接続の形態やマーキングの種類等によって，いくつかのクラスに分けられる．代表的なものとして，状態機械・マークグラフ・自由選択ペトリネット・色付ペトリネットなどがある．例えば，状態機械は有限オートマトンと等価であり，（無限色の）色付ペトリネットはチューリングマシンと等価であることが知られている．

  ペトリネットの表現力と解析能力は相反する．すなわち，状態機械やマークグラフなどの表現力の制限されたペトリネットにおいては，その限定された性質を用いて検証に利用することができるが，（無限色の）色付ペトリネットでは，一般のプログラムを解析しているのと同等であり，検証はより困難なものとなる．

- 代数的仕様記述

  有限状態オートマトンやペトリネットが，状態の遷移によって主体の動作を表現するのに対し，プロセス代数は，発生するイベントの時系列によって主体の動作を表現する．プロセス代数における主たる構成要素は，プロセスとプロセス間の関係を定義する演算子である．プロセス代数におけるプロセスとは，それを外部から観測して得られるイベントの時系列によって規定される主体である．プロセス間の同期は，イベントが複数のプロセスに跨って同時に生起することにより表現される．

  プロセス代数として代表的なものに，C.A.R. Hoare の **CSP**[6] や R. Milner の **CCS**[7] がある．また，CCS を基礎とした通信システム用仕様記述言語として，ISO では **LOTOS**[8] が提案されている．

  2つのプロセスは，外部からの観測では区別がつかない時，等価であると呼ばれる．等価性は，プロセス代数を用いた仕様の検証における中心的な概念である．プロセス代数では，演算子の意味規則に従って式を機械的に書き換えて等価なプロセスを得ることが出来る．この性質によって，複数のプロセス間の等価性が証明できる場合がある．例えば，仕様 *Spec* とそのインプリメント *Imp* をそれぞれプロセス代数式で表す場合を考えよう．*Spec* が，システムを単一のプロセスと見なしたときの外部から観測されるプロセスの動作を記述したもの，*Imp* が内部の詳細な動作まで記述した複数プロセスの結合によって与えられるものとする（図 2）．このとき，*Spec* と *Imp* の外部観測点 Imp_{out} の動作が等しければ，*Imp* は *Spec* を正しくインプリメントしていることが保証できる．

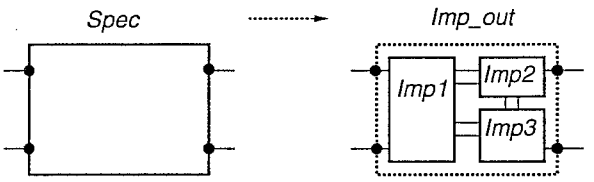


図 2: 等価性による検証

等価性は，その定義の方法によっていくつかに分類されており，強等価・弱等価・合同等価・観測等価・トレース等価などの種類がある．プロセス代数はその等価性の定義によって等式の意味が異なってくる．例えば，CCS は合同等価を採用しているので，CCS における等式は合同等価に関して健全（式の一部を，等式が成り立つ別のもので置き換えた式も，もとの式と合同等価となる）である．

3 広域状態遷移ルールを用いた仕様の可到達解析

通信サービス仕様の重要な検証項目の一つに，状態の可到達解析，即ち，特定の状態が，ある仕様によって規定されたシステムに出現するか否かの検査がある．従来の可到達解析では，解析の対象を相互に関連を持つ特定の少数プロセスに限定し，これらのプロセスの可能な遷移をすべてのイベントの発火を仮定して生成する方法によってなされていた [9]．しかし，STR においては，広域状態遷移ルールの適用対象となるプロセスは特定されないため，不特定多数のプロセスの存在を前提とした解析を行なう必要がある．

この際，1) 可到達解析の対象となる状態をどのように与えるか，2) システムに含まれるプロセス数をいくつに仮定するか，が問題となる．1) として，広域状態の一定数のプロセスからなる部分集合を**準広域状態**と定義し，解析の対象とする．この場合，可到達状態集合の大きさはシステムのプロセス数に依存するが，個々のプロセスがとり得る状態が有限であれば，可到達状態集合は上限値を持ち，解析が可能である．

Shibata[10] らは，上記の問題に関して，状態が個々のプロセスに独立して定義されるという条件の基で，可到達状態集合の上限値の存在と，その最大値を与えるプロセス数を求める方法を示した．しかし，STR では一般に複数のプロセスに跨った状態を定義することが可能であり，この方法の適用は限定された仕様に留まっていた．

ここでは，複数プロセスに跨った状態の定義を許す場合について，上記可到達状態集合の上限値の存在と，その上限値を与える最小のプロセス数を求める方法を示す．これによって，無限プロセスの存在を仮定するシステムの可到達性を有限プロセスのそれに置き替えて議論することが可能となる．

3.1 状態遷移ルール

この章では，本論文で取り扱う通信システム記述のためのプロダクションルール（以下，**状態遷移ルール**と呼ぶ）の構文および意味を定義する．

- **システム**

  システムは，任意の数のプロセスと仕様からなる．システムに含まれる全てのプロセスの初期状態は同一である．また，全てのプロセスは，状態遷移ルールの適用に関して公平である．n プロセスからなるシステムを T_n で表す．

- **プロセス**

  プロセスは，通信端末や交換機等の通信システムの構成要素に相当するエンティティである．プロセス識別子 $a, b, c, \ldots$ によって，表される．

- **プロセス変数**

  プロセス変数は，値として，プロセス識別子を採る変数である．$X, Y, Z, \ldots$ のように表す．

- **状態記述要素**

  状態記述要素は，プロセスの状態を表す基本要素で，

$$State\_id\ (\ Process\_id\ [, Process\_id\ ...]\ )$$

のように表す．ここで，$State\_id$ は状態識別子，$Process\_id$ はプロセス識別子またはプロセス変数である．$s1(a)$ は，プロセス a の状態が $s1$ であることを表す．また，$s2(a,b)$ のように，複数のプロセスに跨った状態を表すことができる．

- イベント

  イベントは，システム中に時系列的に発生し，同時刻に発生するイベントは唯一である．イベントは，以下のように表現される，

  $$Event\_id\ (\ Process\_id\ [,Process\_id\ ...]\ )$$

  ただし，$Event\_id$ はイベント識別子である．

- 広域状態

  広域状態は，システム T_n に含まれる全てのプロセスの状態で，g_n で表す．$g_n = \{P \mid P\text{は}T_n\text{内の状態記述要素}\}$ である．

- 状態遷移ルール

  状態遷移ルールは，イベントの発生に伴うシステムの状態変化を定義するプロダクションルールで，

  $$Current\_state\ \ Event:\ \ Next\_state\ .$$

  の形式で書かれる．$Current\_state$，$Next\_state$ は現状態および次状態を表し，それぞれ，状態記述要素の1つ以上の集合として与えられる．$Event$ はイベントである．ルール中の $Process\_id$ はすべてプロセス変数である．

通信システムの仕様は，状態遷移ルールの集合によって定義される．

状態遷移ルールの意味は，時刻 t においてあるイベント $Event$ が生起し，かつ $Current\_state$ がシステム中に存在しているならば，時刻 $(t+1)$ におけるシステムの状態は $Current\_state$ を $Next\_state$ に書き替えたものに遷移する，と解釈する．これは，プロダクションシステムにおいて，広域状態をワーキングメモリに，状態遷移ルールをワーキングメモリの書き換えルールとして解釈したものに相当する．ただし，一つのルール中に書かれた同じ名前の変数は同一のプロセスを，異なる変数は異なるプロセスを参照していなければならない．また，ルールは全てのプロセスに対して，等しく適用可能である．

図3に，ルールの実行の様子を示す．プロセス a に対し，イベント $ev(a,b)$ が生起した場合，時刻 t における広域状態 $\{s0(a), s0(b), s0(c), s2(d,e), s2(e,d)\}$ の部分集合および $ev(a,b)$ とユニフィケーション可能な，$Current\_state$ および $Event$ を持つルールが選択される．ここで，$Current\_state = \{s0(a), s0(b)\}$ 変数拘束 $A = a$，$B = b$ が得られるので，$Next\_state = \{s1(a,b), s1(b,a)\}$ が導かれ，結局，以下の広域状態遷移が実現される．

$$\{s0(a), s0(b), s0(c), s2(d,e), s2(e,d)\}$$
$$\rightarrow \{s1(a,b), s1(b,a), s0(c), s2(d,e), s2(e,d)\}$$

一つのプロセスが複数の状態記述要素に属すことも可能である．この場合，状態記述要素の書き替えは，ルールに一致した状態記述要素に対してのみ行なわれる．

ルールの適用時における非決定性を排除する等のため，ルール記述に以下の制約事項を設ける．

- 1つのルール中に，プロセス識別子が決定不能な変数が存在してはならない．
- すべてプロセス識別子が同一な状態記述要素が，広域状態中に複数存在してはならない．

通信サービスはこのような状態遷移ルールの集合によって定義される．図4の例は，2者間通話の仕様を表している．

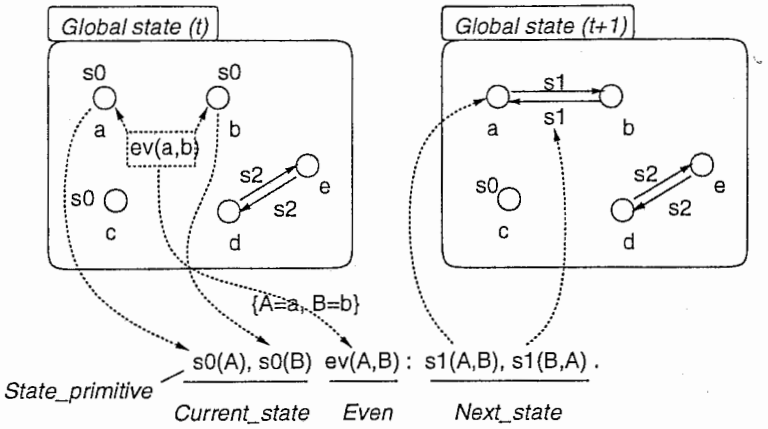


図 3: STR の操作的意味

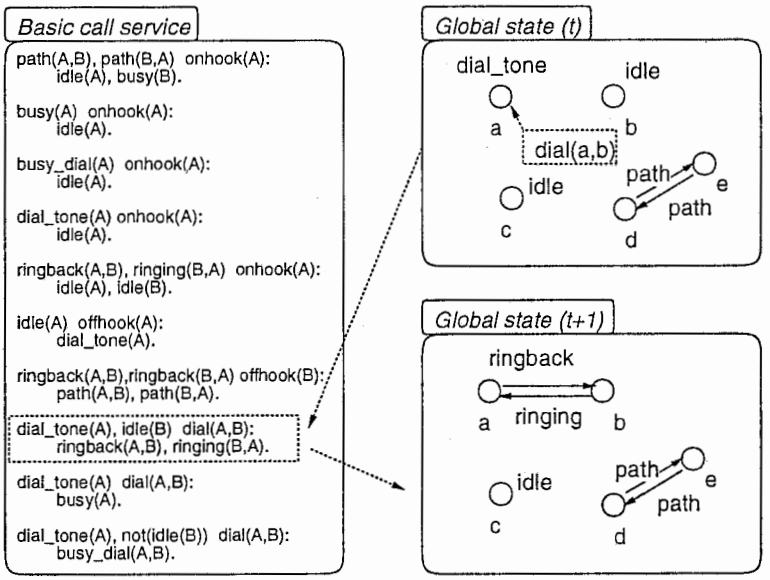


図 4: 2 者間通話の仕様

3.2 可到達解析

3.2.1 可到達判定問題

通信システムの仕様を検証する一つの代表的な方法として，単一ないしは複数のプロセスによって構成される特定の状態が，与えられたシステムに出現するか否かの検査がある．これを可到達解析と呼ぶ．状態遷移ルールの集合によって定義された仕様における可到達解析では，以下の2点が問題となる．

1. 判定の対象となる状態をどのように定義するか．

2. システムのプロセス数をいくつに仮定するか．

従来の可到達解析では，単一あるいは協調して動作する特定の複数プロセスに着目し，可能なすべてのイベントを生起することによってこれらのプロセスに関する状態遷移グラフを作成し解析を行なうのが一般的であった．

一方，状態遷移ルールによる仕様記述では，システム内の任意のプロセスがルールの適用対象となるため，システム全体を解析の対象とする必要が生じる．しかし，広域状態を対象とした可到達解析を行なうことは，一般の通信システムでは非現実的である．なぜなら，システム中に含まれるプロセス数は一般に不特定多数と考えざるを得ないからである（例：電話交換機における端末数）．そこで，広域状態のプロセス数が固定された部分集合として準広域状態を新たに定義し，これを可到達解析の対象として考えることにする．

定義 1 準広域状態 sg_m

特定の m 個のプロセスによって構成される状態．$sg_m = \{u_1, u_2, \ldots, u_k\}_m$ ．ただし，u_i は状態記述要素を値とする変数．

準広域状態を対象とした可到達判定問題は，以下のように定義される．

定義 2 可到達判定問題

システム T_n および準広域状態 sg_m が与えられた時，T_n の状態遷移木中のある広域状態 g_n が，sg_m を含むか否かを判定する問題．

一般に，通信サービスでは，全ての端末の初期状態は同一である．そこで，以下の議論では，システム T_n 内の全てのプロセスの初期状態を同一（$s0$）と仮定する．

3.2.2 可到達状態集合のプロセス数依存性

次に，前掲の問題 2. について検討する．

定義 3 可到達状態集合 $R_m(n)$

システム T_n において sg_m が可到達であることを $\underset{n}{\rightharpoonup} sg_m$ と表す．このとき，可到達状態集合 $R_m(n)$ を以下のように定義する．

$$R_m(n) = \{sg_m \mid \underset{n}{\rightharpoonup} sg_m\}$$

準広域状態 sg_m に含まれるプロセス数は固定であるが，可到達状態集合 $R_m(n)$ の大きさは n に依存する．例として，以下の仕様によって定義されるシステムにおいて，プロセス数 2 の準広域状態に関する可到達状態集合 $R_2(n)$ を考える．

$$s0(A), s0(B) \;\; ev0(A,B) : \;\; s0(A), s1(B).$$
$$s1(A), s1(B) \;\; ev1(A,B) : \;\; s0(A), s2(B).$$

システムのプロセス数が 2 の場合の可到達状態集合 $R_2(2)$ は，広域状態遷移が，

$$\{s0(a), s0(b)\} \rightarrow \{s0(a), s1(b)\}$$

となるので，

$$R_2(2) = \{\{s0, s0\}, \{s0, s1\}\}$$

（プロセス識別子は省略）．一方，プロセス数 4 では，

$$\begin{aligned}
&\{s0(a), s0(b), s0(c), s0(d)\} \\
&\rightarrow \{s0(a), s1(b), s0(c), s0(d)\} \\
&\rightarrow \{s0(a), s1(b), s0(c), s1(d)\} \\
&\rightarrow \{s0(a), s0(b), s0(c), s2(d)\} \\
&\rightarrow \{s0(a), s1(b), s0(c), s2(d)\} \\
&\rightarrow \{s0(a), s1(b), s1(c), s2(d)\} \\
&\rightarrow \{s0(a), s0(b), s2(c), s2(d)\}
\end{aligned}$$

なので，

$$\begin{aligned}
R_2(4) \quad = \quad & \{\{s0, s0\}, \{s0, s1\}, \{s1, s1\}, \\
& \{s0, s2\}, \{s1, s2\}, \{s2, s2\}\}
\end{aligned}$$

このように，準広域状態を対象にした可到達解析であっても，システムのプロセス数をいくつに仮定するかが問題となる．もし，可到達状態集合 $R_m(n)$ が n に関して有界であるならば，無限プロセスを含むシステム T_∞ を仮定しても有限な集合 $R_m(\infty)$ が存在する．ある準広域状態が $R_m(\infty)$ に含まれないならば，どのような規模のシステムにおいてもその状態は発生し得ないことが保証できる．また，ある準広域状態が $R_m(\infty)$ に含まれるならば，十分大きなプロセスを用意すればその状態が実現されることが保証できる．

以下では，このような $R_m(\infty)$ の存在の証明，および，これを求める手続きについて述べる．

3.2.3 無限のプロセスを含むシステムにおける可到達問題

可到達判定問題の解決に先立ち，可到達性に関するいくつかの性質を示す．

広域状態 $g_n = \{u_1, u_2, \ldots, u_l\}_n$ について，各 u_i に含まれるプロセス識別子を，プロセス間の関係を保存しつつ書き換えることによって得られる任意の広域状態（準広域状態）を**同型の広域状態（同型の準広域状態）**と呼ぶ．図 5は，同型の広域状態の例である．

広域状態 g_n に含まれるプロセス記述子を，プロセスの一致・不一致の関係を保存するように，同一のプロセスは同一の変数に異なるプロセスは異なる変数に書き換えたものを G_n とすると，G_n は g_n と同型の全ての広域状態を含んでいる考えられる．これを抽象化広域状態と呼ぶ．同様に，準広域状態 sg_m に対する抽象化準広域状態を SG_m と表す．

定理 1 $\underset{n}{\rightarrow} sg_m$ ならば $\underset{n}{\rightarrow} SG_m$ である．

証明　システム T_n 内の全てのプロセスの初期状態は同一なので，T_n の状態遷移グラフに含まれる任意の広域状態 g_n を考えると，g_n と同型な広域状態もこの遷移グラフに含まれる．よって，上記は明らか．　□

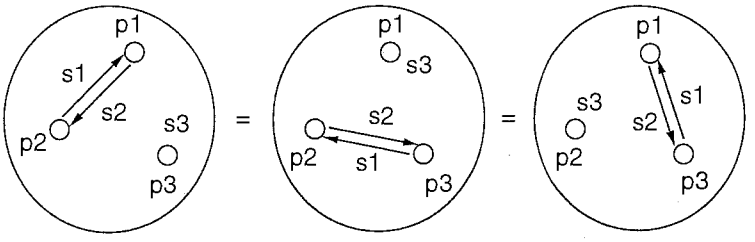


図 5: 同型の広域状態

このように，可到達性に関しては g_n と G_n，sg_m と SG_m は等価である．よって，以後の議論では，g_n，sg_m の代りに G_n，SG_m を広域状態，準広域状態として用いる．

定理 2 可到達状態集合 $R_m(n)$ は n に関して単調増加である．

証明 $R_m(n)$ が単調増加でないと仮定すると，

$$R_m(N) \supset R_m(N+1)$$

なる N が存在する．ここで，T_{N+1} のうち N 個のプロセスについてのみルールを適用して得られる可到達状態集合を $R'_m(N+1)$ とすれば，明らかに，

$$R_m(N) \subseteq R'_m(N+1) \subseteq R_m(N+1)$$

であるから，$R_m(N) \subseteq R_m(N+1)$ となり仮定に反する． □

定理 3 可到達状態集合 $R_m(n)$ は n に関して有界である．

証明 T_n における任意の準広域状態 SG_m について考えると，SG_m に含まれる状態記述要素の種類は有限であり，ルール記述に関する制限から，同一プロセスに関する状態記述要素は高々 1 つしか存在しない．よって，SG_m の種類は有限である．従って，$R_m(n)$ は有界となる． □

以上の定理から，$R_m(n)$ の上限値 $(R_m)_{max}$ を与え，かつそれ以上の数では $R_m(n) = (R_m)_{max}$ となるような $n = N_{max}$ の存在が保証される．

次に，N_{max} を求めるために，$R_m(n)$ と n の関係について考察する．

定義 4 最小プロセス数 $Min(G_n, m)$

広域状態 G_n に含まれる m 個のプロセスからなる任意の準広域状態 $SG_m \subseteq G_n$ 全てについて $\underset{i}{\rightarrow} SG_m$ を満たす最小の i.

定義から明らかなように，$m \leq i \leq n$ である．

定義 5 最大連結数 C_{max}

広域状態のグラフ表現において，状態記述要素によって連結されているプロセスの数を連結数と呼ぶことにする．このとき，システム T_n において可到達な全ての広域状態 G_n における連結数の最大値．

定理 4 時刻 t における広域状態 $G_n(t)$ がルールの適用によって時刻 $(t+1)$ に $G_n(t+1)$ となったとする．ルールで参照されるプロセス数の最大値を K_{max}，最大連結数を C_{max} とすると，$m \geq K_{max}$ かつ $m \geq C_{max}$ なる m に関して以下の性質が成り立つ．

$$Min(G_n(t+1), m) \leq 3 \cdot Min(G_n(t), m)$$

証明　ある状態遷移ルールの $Current\_state$ および $Event$ で参照されるプロセス数を kc, $Next\_state$ で参照されるプロセス数を kn とすると，$kc \geq kn$ である．ここで，$kc > kn$ である場合は，$Next\_state$ で参照されていないプロセス識別子を含む状態プリミティブが $Current\_state$ に存在することを意味するが，この状態プリミティブを $Next\_state$ に加えても状態遷移ルールの意味は保存される．従って，以後の証明では，全ての状態遷移ルールは $kc = kn = k$ となるように正規化されていると仮定する．

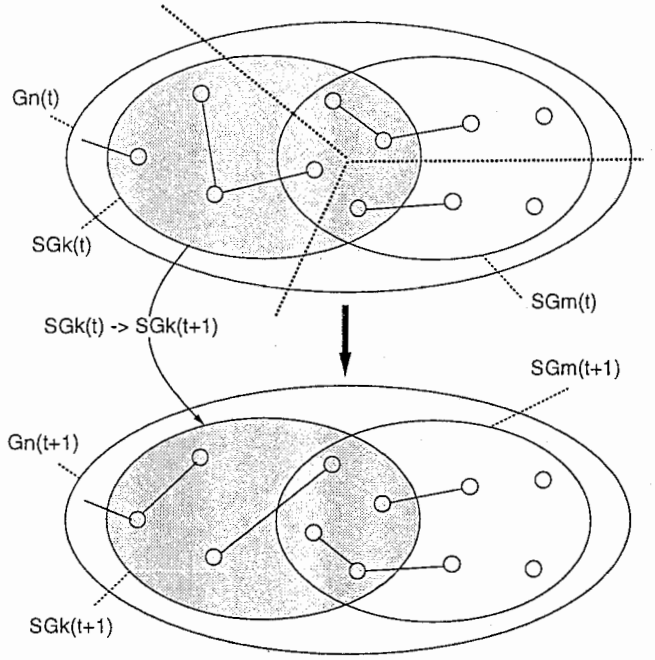


図 6: 状態遷移ルールの適用による広域状態遷移

$G_n(t)$ が状態遷移ルール $SG_k(t)$ $Event:$ $SG_k(t+1)$ の適用によって $G_n(t+1)$ に遷移する様子を図示すると，図 6 のようになる．ここで，$G_n(t+1)$ に含まれる m プロセスの準広域状態 $SG_m(t+1)$ が可到達となる十分条件は，$G_n(t)$ において，$SG_m(t+1)$ のルール適用前の状態 $SG_m(t)$ と ルールによって参照される状態 $SG_k(t)$ が存在することである．即ち，

$$\underset{n}{\rightharpoonup}(SG_k(t) \cup SG_m(t)) \text{ ならば } \underset{n}{\rightharpoonup} SG_m(t+1)$$

次に，$SG_k(t) \cup SG_m(t)$ が m 以下のプロセスを含む，他の部分と連結を持たない高々 3 つの部分に分割可能であることを示す．

$SG_k(t) \cup SG_m(t)$ 内で，連結されているプロセスのみで構成される準広域状態 SGC を連結数の多い順にソートしたものを SGC_i とする．

$$(SG_k(t) \cup SG_m(t)) = SGC_1 \cup SGC_2 \cup \ldots \cup SGC_n$$

各 SGC_i に含まれるプロセス数を $pn(SGC_i)$ としたとき，最大連結数 $C_{max} \leq m$ なので，

$$\sum_{i=1}^{l} pn(SGC_i) \leq m$$

を満たす最大の l が存在する．これを L_{max} とする．$SG_k(t) \cup SG_m(t)$ を

$$\begin{aligned} SG1 &= \{SGC_1, \ldots, SGC_{L_{max}}\}, \\ SG2 &= \{SGC_{L_{max}+1}\}, \\ SG3 &= \{SGC_{L_{max}+2}, \ldots, SGC_n\} \end{aligned}$$

の 3 つの部分に分割すれば，$SG_k(t) \cup SG_m(t)$ に含まれるプロセス数は $\leq 2m$ なので，$SG1, SG2, SG3$ の各々に含まれるプロセス数は $\leq m$ である．

以上のように分割された $SG1, SG2, SG3$ は，それぞれ $Min(G_n(t), m)$ 個の初期プロセスから可到達なので，

$$\underset{3 \cdot Min(G_n(t),m)}{\rightarrow} SG_k(t) \cup SG_m(t)$$

従って，

$$\underset{3 \cdot Min(G_n(t),m)}{\rightarrow} SG_m(t+1)$$

よって，

$$Min(G_n(t+1), m) \leq 3 \cdot Min(G_n(t), m)$$

□

定理 5 任意のシステム T_n について，以下の性質を満たす N が存在する．

$$\begin{aligned} R_m(3^i) &\subset R_m(3^{i+1}) \qquad i < N \\ R_m(3^i) &= R_m(3^{i+1}) \qquad i \geq N \end{aligned}$$

証明 $R_m(3^N) = R_m(3^{N+1})$ のとき $R_m(3^M) \supset R_m(3^{M+1})$ なる $M > N$ が存在すると仮定する．この時，以下の条件を満足する準広域状態 SG_m が存在する．

$$SG_m \notin R_m(3^M),\ SG_m \in R_m(3^{M+1})$$

従って，3^{M+1} プロセスからなる初期広域状態 $G_{3^{M+1}}(t_0)$ から $SG_m \in G_{3^{M+1}}(t_0+n)$ なるある広域状態 $G_{3^{M+1}}(t_0+n)$ への以下のような遷移系列が存在する．

$$G_{3^{M+1}}(t_0), G_{3^{M+1}}(t_0+1), \ldots, G_{3^{M+1}}(t_0+n)$$

この遷移系列の各広域状態に Min を適用した以下の系列を考える．

$$\begin{gathered} Min(G_{3^{M+1}}(t_0), m), \\ Min(G_{3^{M+1}}(t_0+1), m), \\ \vdots, \\ Min(G_{3^{M+1}}(t_0+n), m) \end{gathered}$$

ここで，明らかに $Min(G_{3^{M+1}}(t_0), m) = m$,
$Min(G_{3^{M+1}}(t_0+n), m) = 3^{M+1}$ なので，以下の関係を満たす $L < M$ が存在する．

$$\begin{aligned} Min(G_{3^{k+1}}(t_0+L), m) &\leq 3^N \\ Min(G_{3^{k+1}}(t_0+L+1), m) &> 3^N \end{aligned}$$

一方，定理 4 より，

$$Min(G_{3^{k+1}}(t_0+L+1), m) \leq 3 \cdot Min(G_{3^{k+1}}(t_0+L), m)$$

従って，

$$\begin{aligned} Min(G_{3^{k+1}}(t_0+L+1),m) &\leq 3 \cdot Min(G_{3^{k+1}}(t_0+L),m) \\ &\leq 3 \cdot 3^N \\ &= 3^{N+1} \end{aligned}$$

以上から，

$$3^N < Min(G_{3^{k+1}}(t_0+L+1),m) \leq 3^{N+1}$$

これは，$SG'_m \notin R(3^N)$ かつ $SG'_m \in R(3^{N+1})$ なる準広域状態 SG'_m が存在することを意味するが，$R_m(3^N) = R_m(3^{N+1})$ の仮定に反している．よって，そのような M は存在しない．よって，$R_m(3^N) = R_m(3^{N+1})$ ならば，$i > N$ について $R_m(3^i) = R_m(3^{i+1})$ が成り立つ．

$R_m(n)$ は n に関して単調増加かつ有界であるから，このような N は必ず存在する． □

すなわち，図 7 に示すように，システムを構成するプロセス数を 3 倍づつ増加させていくとある時点で可到達状態集合の大きさが変化しなくなり，その時の値が可到達集合の最大値 $(R_m)_{N_{max}}$ となる．また，N_{max} は 3^{N-1} と 3^N の間に存在する．これによって，T_∞ における可到達問題は $T_{N_{max}}$ における可到達問題に還元できる．

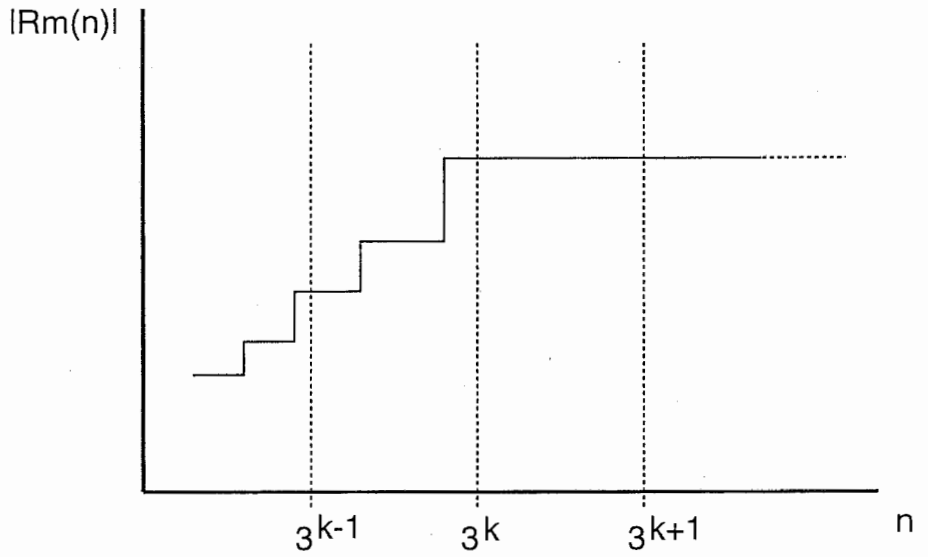


図 7: 可到達状態集合とプロセス数

3.2.4 可到達判定アルゴリズムへの応用

上記の結果から，状態遷移ルールの集合と準広域状態 SG_m $(m \geq C_{max}, m \geq K_{max})$ を入力して可到達性を判定する以下のアルゴリズムが導ける．

1. $i \leftarrow 1$, $n \leftarrow m$, $R_0 = \emptyset$.

2. $R_i \leftarrow R_m(n)$.

3. $R_i = R_{i-1}$ なら 6. へ.

4. $i \leftarrow i+1$, $n \leftarrow 3 * n$

5. 2. へ.

6. $SG_m \subset R_i$ なら SG_m は可到達，そうでなければ非可到達.

4 BDD を用いた可到達検証の高速化

前章では，あたえられた状態遷移ルールセットについて，特定の条件の元で準広域状態の可到達集合に上限値が存在すること，および，それを求めるアルゴリズムを示した．前記アルゴリズムにおいては，上限値が求まるまで，各プロセス数に対する可到達状態集合を求めるステップを繰り返すものであった．しかし，特定のプロセス数に対する全ての可到達状態集合を求めるだけでも，プロセス数が大きくなれば，非常に大きな計算量が必要となる．本章では，プロセス数と計算量の関係を検討し，これを削減するための方法として，BDD の適用を検討する．

4.1 可到達状態数のプロセス数依存性

一般に，n 個のプロセスに関する広域状態遷移グラフを計算するには，$O(a^n)$ のオーダを必要とし，多数の n に関しては，非現実的な計算量とならざるを得ない．

状態遷移ルールは，信号の授受に関する記述を省略し，直接，広域状態の遷移を記述する方法である．この場合も，広域状態遷移グラフの状態数はプロセス数 n に依存することが予想される．

図 8 は，状態遷移ルールで書かれた 2 者間通話の仕様について，プロセス数 n と，初期状態より可到達な状態数 R との関係を計算した結果である．実線は，全てのプロセスを区別して計算したもので，グラフの傾きから，

$$|R(n)| = 0.44 \times 5^n$$

であることが読み取れる．因みに，

$$\begin{aligned} |R(6)| &\approx 6875 \\ |R(10)| &\approx 4,297,000 \end{aligned}$$

であり，SUN SparcStation2 上の Prolog を用いて計算したところ，6 プロセスが限度であった．

全てのプロセスを区別しないと仮定すると，対象性を利用して状態数を削減できる．ある広域状態において，プロセスの識別子を別のプロセスの識別子と入れ換えることによって得られる状態を同型の広域状態と呼ぶ．破線のグラフは，同型の広域状態を 1 つ広域状態としてカウントした時の，プロセス数と広域状態数の関係を示している．この場合は，状態数が 1/100 程度に削減されるが，オーダはほぼ $O(a^n)$ であり，やはり，$n = 7$ 程度が計算の限界であった．

4.2 BDD による高速検証方式

4.2.1 BDD

BDD (Binary decision diagram; 2 分決定木) は，論理関数を効率良く表現する方法として 1978 年 Akers[16] によって考案され，その後 1985 年に Bryant[17] によって効率的な演算法が開発されて以来，主として論理回路設計の分野で広く用いられるようになった．

図 9 に BDD の例を示す．(a) は表現対象である論理関数 $x_1\overline{x_2} + x_3$ の真理値表で，(b) はその BDD 表現，(c) は既約な BDD である．BDD は図のように変数でラベル付けされた変数接点と，'0' または '1' でラベル付けされた定数接点からなる．変数接点の左の枝は，その変数の値として '0' を選択したことを，右の枝は '1' を選択したことを意味する．この選択を繰り返し，定数接点に到達した時の値が，その変数への値の割り当てに対する論理関数値を表している．

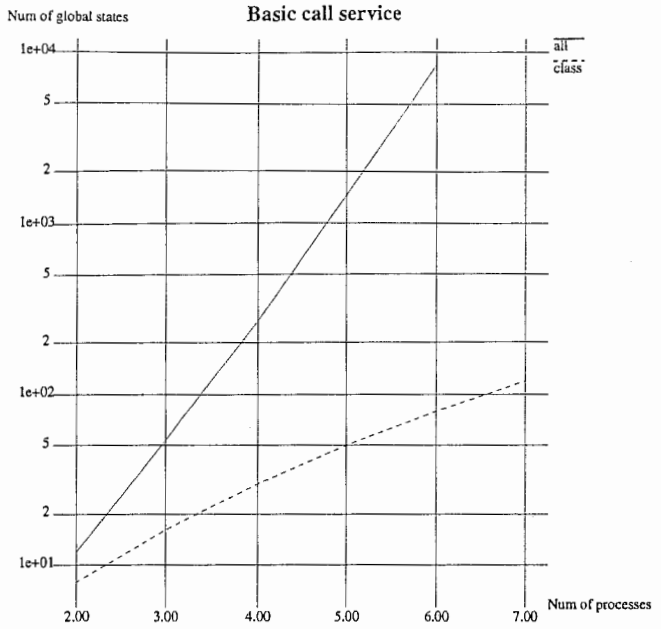
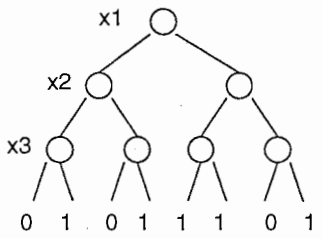


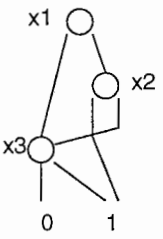
図 8: プロセス数 対 状態数

x1	x2	x3	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

(a) 真理値表



(b) BDD



(c) 既約なBDD

図 9: 論理関数 の BDD 表現

真理値表表現では，n 変数に対して $O(2^n)$ の記憶域を必要とするが，BDD では多くの論理関数が，既約化によって 2^n 以下の接点数（即ち記憶域）に縮退される．これによって，従来，不可能とされていた論理関数の組合せ的な計算のいくつかが可能になっている．

BDD の特徴をまとめると以下のようになる．

1. 既約な BDD は，論理関数の標準形を与える．
2. 多くの論理関数が真理値表表現に比べ少ないサイズで表現できる．
3. 論理関数に対する演算が BDD のサイズに比例した時間で実行できる．

4.2.2 可到達解析アルゴリズムの BDD 表現

可到達解析には，初期状態から到達可能な全ての状態を網羅的に生成する必要があるが，状態遷移グラフに表れる状態を陽にリストで表現する方法をとると，プロセス数が大きくなると状態集合の記憶域があふれ解析不可能となる．

この対策として，BDD を用いて状態集合を陰に表現する方法がある [18, 19]．即ち，図 9 において，状態が x_1, x_2, x_3 の 3 bit で表されていると考え，$f = 1$ がその状態の存在を $f = 0$ が非存在を表すものとすると，この論理関数は

集合演算	BDD
$A \cap B$	$\chi_A(x) \wedge \chi_B(x)$
$A \cup B$	$\chi_A(x) \vee \chi_B(x)$
$A \subset B$	$\chi_A(x) \Rightarrow \chi_B(x)$
$\overline{A}$	$\neg\chi_A(x)$
$A \times B$	$\chi_A(x) \Leftrightarrow \chi_B(x)$

表 1: 集合演算と論理演算

状態集合 $\{s_1, s_3, s_4, s_5, s_7\}$ を陰に表しているものと見なすことが出来る．この論理関数は，集合を表す以下の特徴関数と等価である．

$$\chi_S(x) = 1 \qquad \text{iff } x \in S$$

可到達状態集合を求める計算は，可到達状態集合の初期値 R_0 を初期広域状態 S_0 とし，これにルールを適用し新たな広域状態が発生すればこれを新たな可到達状態集合とする．この操作を繰り返し，R_i の最小不動点 R_∞ を求める（図 10）．

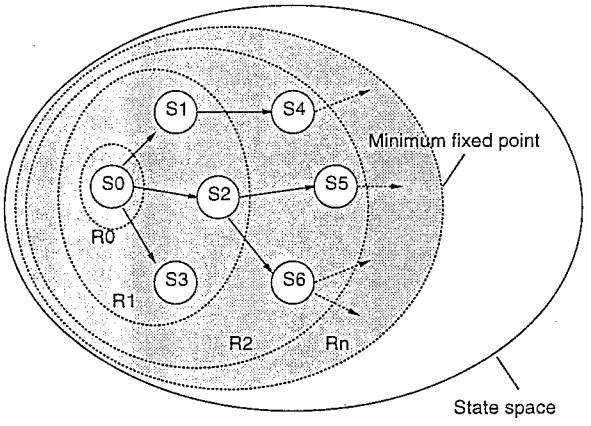


図 10: 可到達集合の求め方

この計算は集合に対する適当な演算の組合せにより実現される．状態集合を特徴関数 (BDD) で表現した場合は，集合演算は特徴関数間の論理演算に置き換えられる．表 1 にこの対応関係を示す．

以上から，可到達状態集合は $R_0 = \chi_{S_0}(S)$ として，論理関数，

$$R_{i+1}(S') = R_i(S') + \exists e, S\ (R_i(S) \cdot N(S, e, S'))$$

における R_i の最小不動点として求められる．これは，以下のアルゴリズムによって計算される．

```
procedure reach(χ_init(S), N(S,S')) {
    R(S) := χ_init(S) ;
    loop {
        R'(S') := ∃ S. ( R(S) · N(S,S') ) ;
        if ( R'(S) ⇒ R(S) ) then end
        R(S) := R'(S) + R(S) ;
```

```
    }
}
```

状態変数 S は，広域状態を値とする．$N(S,e,S')$ は，広域状態 S' においてイベント e が発生すると広域状態 S に遷移する場合に値として '1' をとる遷移関数である．これは，図 10 の接点間の遷移関係を集合要素としたときの特性関数と見なすことが出来る．

上記の各演算は，BDD の接点数に比例する時間で実行可能である．従って，計算途上の可到達状態集合 R_i が BDD によって効率的に表現できるならば，最小不動点を効率的に求めることが出来る．

5 状態の表現

BDD を用いて状態遷移ルールを計算するためには，与えられたルールセットから上述の状態と遷移関数を構成する方法を明らかにする必要がある．

広域状態は状態記述要素の集合である．そこで，状態識別子とプロセス識別子（プロセス定数）の任意の組合せが，それぞれ，広域状態を表す変数 S の特定のビットに写像するように構成すれば，全ての種類の広域状態を表現することが出来る．

広域状態空間 G の大きさは，仕様中に現れるパラメータとして 1 つのプロセスを採る状態記述要素の種類を m_1，2 つのプロセスを採る状態記述要素の種類を m_2 とした場合，以下の式で得られる．

$$|G(m_1,m_2,n)| = 2^{m_1 \cdot n + m_2 \cdot n^2}$$

（2 者間通話の仕様では，$n=8$ とした場合 $|G| = 2^{216}$ である．）

従って，任意の広域状態は，長さ $log_2|G|$ のベクトル S で表すことができる．広域状態 $g \in G$ に含まれる 1 つの状態記述要素は，S の i 番目のビット $s_i =' 1'$ として表現する．状態記述要素から i への変換は，パラメータ数 1 の状態記述要素に関しては，これを $Sid_1(p_1)$ とした場合，

$$i = n \cdot (Sid_1)_{index} + (p_1)_{index}$$

パラメータ数 2 の状態記述要素に関しては，これを $Sid_2(p_{21},p_{22})$ とした場合，

$$i = n \cdot m_1 + n^2 \cdot (Sid_2)_{index} + n \cdot (p_{21})_{index} + (p_{22})_{index}$$

で与える．ただし，$(x)_{index}$ は，全ての x を全順序で並べた場合のインデックスを表す．

6 状態遷移ルールから論理関数への変換

ルールセットから上記の遷移関数 $N(S,e,S')$ を生成するには，遷移関数を以下のように各々のルールに関して展開する．

$$\begin{aligned} N(S,e,S') &= N_{r_0}(S,S') + N_{r_1}(S,S') + \cdots \\ &\quad + N_{r_{(n-1)}}(S,S') \end{aligned}$$

r_i は i 番目のルールをあらわす．$N_{r_i}(S,S')$ は，状態 S に対してルール r_i が適用可能で，かつ r_i の適用によって状態 S から状態 S' へと遷移可能の場合に値 '1' を採る論理関数である．ルール r_i は広域状態との単一化の際に複数の単一化代入を持ちうる．これらを α_j とし，単一化代入を施された r_i を r_i/α_j とすると，$N_{r_i}(S,S')$ は各単一化代入に関してさらに以下のように展開される．

$$N_{r_i}(S,S') = N_{r_i/\alpha_0}(S,S') + N_{r_i/\alpha_1}(S,S') + \cdots + N_{r_i/\alpha_{m-1}}(S,S')$$

さらに，ベクトル S' を各ビットごとの積形に展開する．

$$N_{r_i/\alpha_j}(S,S') = N'_{r_i/\alpha_j}(S,s'_0)\cdot N'_{r_i/\alpha_j}(S,s'_1)\cdots \cdot N'_{r_i/\alpha_j}(S,s'_{l-1})$$

ただし，$N_{r_i/\alpha_j}(S,s'_k)$ は，単一化されたルール r_i/α_j によって，状態 S から，状態 S' に遷移するときの k 番目のビットが s'_k である，ことを表す．

S および S' の k ビット目を s_k および s'_k ，単一化代入を施されたルールの *Current_State* 部を集合 C，*Next_State* 部を集合 N で表す．また，集合 C および 集合 N に含まれる各状態記述要素を，それぞれ対応する S のビット s_k に置き換えたものを C_s および N_s で表す．

広域状態がルール r_i の適用を満たしている場合，各 s'_k の値は，それが r_i の *Current_State* 部のみに含まれるなら '0' に，*Next_State* 部のみ含まれるなら '1' に，それ以外であれば無変化 (s_k) になるため，各々の $N_{r_i/\alpha_j}(S,s'_k)$ は結局以下のように計算される．

$$N_{r_i/\alpha_j}(S,s'_k) =$$
$$\begin{array}{ll} (s'_k \equiv s_k) & \text{if} \quad (s_k \in C_s)\cdot(s_k \in N_s) \\ (s'_k \equiv s_k \cdot \overline{(\Pi(C_s))}) & \text{if} \quad (s_k \in C_s)\cdot(s_k \notin N_s) \\ (s'_k \equiv s_k + \Pi(C_s)) & \text{if} \quad (s_k \notin C_s)\cdot(s_k \in N_s) \\ (s'_k \equiv s_k) & \text{if} \quad (s_k \notin C_s)\cdot(s_k \notin N_s) \end{array}$$

ただし，$\Pi(X)$ は，集合 X の全ての要素の積結合を表す．

7 まとめ

本稿では，通信サービスの形式的な要求仕様における検証の概念と，従来の主要な手法（有限状態オートマトン・ペトリネット・プロセス代数）を紹介し，これにたいする新たな手法として状態遷移ルールに基づく記述法を取り上げ，可到達検証の手法について検討した．

状態遷移ルールは，記述内容が直感的に理解しやすい，ルール集合であるため複数の仕様の合成が比較的容易である，などの従来の形式的記述法にはない利点がある．一方，複雑な仕様を記述する場合にルール間の矛盾が生じないように機械的な検証の必要性が高い．

本稿では，特定の条件のもとで，与えられた状態遷移ルールの集合に対する可到達状態集合の上限値をもとめるアルゴリズムを示した．また，このアルゴリズムの高速化のために，BDD を用いる手法について検討した．

今後の課題としては，上記の条件を緩和すること，および，BDD の適用については，実際に効果を発揮させるためのさらにアルゴリズムの改良が必要である．

参考文献

[1] Y. Hirakawa and T. Takenaka: "The telecommunication service description using state transition rules", Proc. of Sixth International Workshop on Software Specification and Design (1991).

[2] J. E. Hopcroft and J. D. Ullman: "Introduction to automata theory, languages and computation", Addison-Wesley (1979).

[3] CCITT: "Functional Specification and Description Language (SDL)" (1984).

[4] C. Petri and J. C. F. Greene: "Communication with automata", Supplement 1 to Technical Report RADC-TR-65-377, **1**, (1966).

[5] J. Peterson: "Petri nets", Computing Surveys, **9**, 3, pp. 223–252 (1977).

[6] C. Hoare: "Communicating Sequential Processes", Prentice-Hall International (1985).

[7] R. Milner: "Communication and concurrency", Prentice-Hall International (1989).

[8] T. . ISO/IEC/TR 9751: "Information technology — Open Systems Interconnection — LOTOS description of the session service (protocol)", ISO (1989).

[9] B. Sarikaya, V. Koukoulidis and G. V. Bochmann: "Method of analyzing extended finite-state machine specifications", Computer Communication, **13**, 2, pp. 83–92 (1990).

[10] K. Shibata, Y. Hirakawa, A. Takura and T. Ohta: "Reachability analysis for specified processes in a behavior description", IEICE Trans, Commun, **E76-B**, 11 (1993).

[11] 平川： "STR (state transition rule) 記述仕様書", ATR Technical Report, TRC-0073 (1992).

[12] Y. Inoue, T. Ohta and et al: "Method for supporting detection and elimination of feature interaction in a telecommunication system", Proc. of Int. Workshop on Feature Interaction, pp. 61–81 (1992).

[13] K. Takami, T. Ohta and et al: "A visual design support system for telecommunications services", Proc. of IEEE Phoenix Conference on Computers and Communications, pp. 593–599 (1993).

[14] K. Kawata, T. Ohta and et al: "On a communication software generation method from communication service specifications described by a declarative language", Proc. of 5th International Conference on Computing and Information (1993).

[15] 村田： "ペトリネットの解析と応用", 近代科学社 (1989).

[16] S. B. Akers: "Binary decision diagrams", IEEE Trans. Comput., **C-35**, 12, pp. 1035–1044 (1978).

[17] R. E. Bryant: "Graph-based algorithms for boolean function manipulation", IEEE Trans. Comput., **C-35**, 8, pp. 677–691 (1986).

[18] O. Coudert and J. Madre: "A unified framework for the formal verification of sequential circuits", IEEE International Conference on Computer-Aided Design, pp. 126–129 (1990).

[19] B. Lin, H. Touati and A. R. Newton: "Don't care minimization of multi-level sequential logic networks", IEEE International Conference on Computer-Aided Design, pp. 414–417 (1990).