マルチメディア通信と分散処理 58−12
（1992. 11. 19）

シミュレーション関係に基づくLOTOS仕様の検証システム

山野 敬一郎* 太田 正孝* 高橋 薫**

*高度通信システム研究所 **東北大学

形式的仕様記述言語の一つであるLOTOSを用いて、通信システム等の仕様を記述することの利点はその検証性にある。本報告では、LOTOS仕様の増補的な開発において、上位レベルの仕様と下位レベルの仕様の間の関係を検証するのに適当な、シミュレーション関係の概念をとりあげ、その定式化、および判定アルゴリズム等を示す。さらに、通信システムの高信頼化支援システムITECSを提案する。その中で仕様の段階的な詳細化過程を支援するために、このシミュレーション関係の概念を適用し、検証を行うことが有効であることを示す。

A Verification System for LOTOS Specifications based on the Simulation Relation

Keiichirou Yamano* Masataka Ohta* Kaoru Takahashi**

*AIC System Laboratories
6-6-3, Minami-Yoshinari, Aoba-ku, Sendai, Miyagi, 989-32, JAPAN

**Tohoku University
2-1-1, Katahira, Aoba-ku, Sendai, Miyagi, 980, JAPAN

An advantage of describing the specification of communication systems using LOTOS, one of the Formal Specification Techniques, is its ability of verification. In this paper, we discuss the formulation and verification algorithm of Simulation Relation, which is appropriate concept to verify the relation between another abstraction levels of the same specifications. Such levels appear in the process of incremental development of LOTOS specification. In addition, high-reliable development environment for communication systems, called ITECS is proposed. Also, we show the concept of simulation relation is effective in this development environment.

1 はじめに

近年、プロトコルなど通信システムの厳密な仕様化を行うために、各種の形式的記述技法 (FDT: Formal Description Techniques) が提案されている。

このFDTの1つとして開発されたLOTOS(Language Of Temporal Ordering Specification)[1] は、数学的モデルをベースとしており記述・検証能力が高いため、その活用が期待されている。例えば、仕様の段階的な開発工程において弱双模倣 (Weak-bisimulation) 関係 [3] の概念を適用することにより、LOTOSで記述された上位レベルと下位レベルの2つの仕様間の無矛盾性の検証を行うことが可能である。また、その判定手続きも提案されている [6]。

しかしながら、仕様の段階的な開発においては、複数の上位レベルの仕様を組み合わせて下位レベルの仕様を開発する場合や、下位レベルの仕様に上位レベルの仕様には含まれていない例外処理などの付加的な情報を加える場合が多い。このような増補的 (incremental) な仕様の詳細化を行う場合には、通常上記の弱双模倣等価関係は成立せず、むしろ片方向のシミュレーション関係を検証の概念として用いることが適当である。

本報告では、まず仕様の段階的な詳細化過程の概念と、それ支援するための通信ソフトウェアの高信頼化支援システムITECSについて紹介を行う。次に仕様記述言語LOTOSと、シミュレーション関係の概念を示し、その判定アルゴリズムを与える。さらに適用例として、この判定アルゴリズムに基づいたLOTOS仕様の検証システムを用いて、通信サービスの検証を行った例を挙げる。

2 高信頼化支援システム：ITECS

通信ソフトウェアの高信頼化のためには、通信ソフトウェアで何をするのかを明確に規定する必要がある。すなわち、仕様記述が重要な役割を果たす。通信システムにおける仕様は、抽象度の高い要求仕様を基に、それを実現するためのソフトウェア仕様に詳細化する必要がある。この詳細化を確実に行うために、図1に示す段階的な仕様の詳細化過程を用いる。ここでの特徴は、各詳細化の段階において、下位レベルの仕様が上位レベルの仕様を正しく詳細化したものであるかどうかの検証を行うことにある。このような過程を経て、抽象度の高い仕様を具体化してゆく。

現在、我々のグループでは以上の詳細化を支援するシステムとして、通信ソフトウェアの高信頼化支援システムITECS(InTegrated Environment for Constructing high-reliable Software) を開発している。図2に示すように、これは仕様定義部、仕様検証部、試験仕様生成部、Cコード生成部から成っている。ITECSの特徴は、プロセスを内包的に記述していくことによって詳細化を行うG-LOTOS[2] の仕様定義環境GLOER[4] や、従来より通信システムの仕様記述に広く活用されているSDL、MSC(Message Sequence Chart) を用いた、複数の図的インタフェースによる仕様定義を可能としていることにある。最終的に、これらにより定義した各仕様をT-LOTOSの仕様に統合し、それをラベル付き状態遷移システムに変換することによって検証を行う。この検証では、機械的な自動検証機能と、それでは検証しきれない部分を補うためのシミュレーションによる仕様確認機能を設けた。

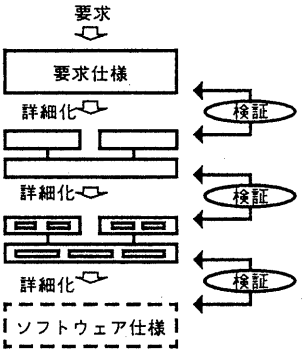

図 1: 仕様の詳細化過程

本報告では、これらの機能のうち図1の詳細化過程を支援する検証について、前述のシミュレーション関係を適用し、その考え方、アルゴリズムの適用例について示す。

3 LOTOSと遷移システム

3.1 LOTOS

LOTOSの概要について、簡単に説明する。

LOTOSにおいて、記述の対象となるシステムは互いに通信し合うプロセス、あるいはサブプロセスとみなされ、システムの仕様はプロセスの通信能力を記述することによって表現される。プロセス間、あるいはプロセスと環境との間の通信の基本単位はアクションもしくはイベントと呼ばれ、ゲートと呼ばれるインタラクション・ポイントにおいて観測可能である。ここで、各プロセスはブラック・ボックスとみなされ、各プロセス内部で発生するアクションは内部アクションiと呼ばれる。

3.2 遷移システム

本節では、LOTOS仕様の形式的なモデルである遷移システム (TS:Transition System) の定義を示す。

定義 1 遷移システム

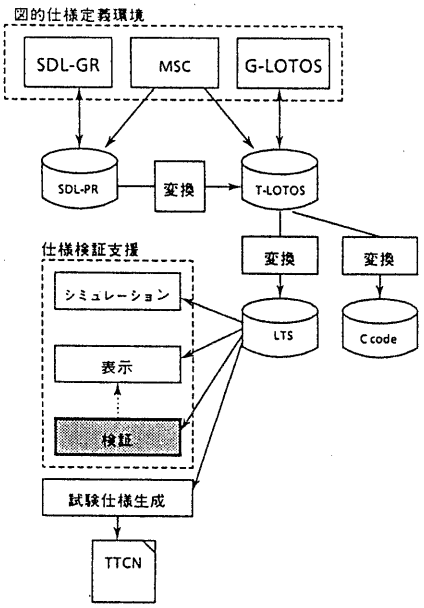

図 2: 高信頼化支援システム ITECS

TS Sys は 4 項組 $< S, Act, \longrightarrow, s_0 >$ である。ここで、

S : 状態の集合

Act : 外部から観測可能なアクションの集合と観測可能ではない内部アクション i から成るアクション集合

$\longrightarrow$: 遷移関係 ($\longrightarrow \subseteq S \times Act \times S$)

s_0 : Sys の初期状態 ($s_0 \in S$)

である。

$(s, \alpha, q) \in \longrightarrow$ のとき、アクション α による状態 s から状態 q への遷移を明確に表すのに、$s \xrightarrow{\alpha} q$ と書く。

TS Sys が有限のグラフとして表現可能であるとき、Sys は有限な TS であるという。 □

なお、上記の遷移システム (以下、TS と略記) は LOTOS のオペレーショナルセマンティクス (操作的意味) を与える公理と推論規則から定義される transition 導出体系に基づき、LOTOS の動作式から生成することが可能である。以下、本報告ではこの TS が有限の場合について議論する。

4 シミュレーション関係

LOTOS 仕様間の検証のためには、一般的に強・弱双模倣等価関係、あるいはトレース等価、試験等価などが知られている。しかし、本報告では仕様の増補的な詳細化過程に対応するため、特に片方向のシミュレーション関係を取りあげ、その性質、判定アルゴリズム、および適用について述べる。詳しくは、文献 [8] を参照されたい。

4.1 シミュレーション関係の諸定義

本節では、上述の TS に基づき、文献 [3] の 9 章で述べられているプロセスのシミュレーション関係を基本として、LOTOS 仕様間のシミュレーション関係を定義する。

定義 2

$Sys =< S, Act, \longrightarrow, s_0 >$ を TS とする。$t \in Act^*$ のとき、$\hat{t} \in (Act - \{\mathrm{i}\})^*$ は t から i のすべてのオカレンスを削ることによって得られる系列である。$t = \alpha_1 \cdots \alpha_n \in Act^*$ のとき、$s \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} s'$ と表す。$t = \varepsilon$ (空系列) のとき、任意の $s \in S$ について、$s \xrightarrow{\varepsilon} s$ である。$t = \alpha_1 \cdots \alpha_n \in Act^*$ のとき、$s(\xrightarrow{i})^* \xrightarrow{\alpha_1} (\xrightarrow{i})^* \cdots (\xrightarrow{i})^* \xrightarrow{\alpha_n} (\xrightarrow{i})^* s'$ ならば、$s \stackrel{\hat{t}}{\Rightarrow} s'$ と表す。 □

定義 3 シミュレーション関係

Sys1 と Sys2 を任意の TS とする。

$Sys1 =< S_1, Act, \longrightarrow_1, q_1 >$

$Sys2 =< S_2, Act, \longrightarrow_2, q_2 >$

次の条件を満たす関係 $\mathrm{S} \subseteq S_1 \times S_2$ を Sys1 から Sys2 へのシミュレーション関係という :

$(s, q) \in \mathrm{S}$ ならばすべての $\alpha \in Act$ について、
$s \xrightarrow{\alpha} s'$ のとき、$q \stackrel{\hat{\alpha}}{\Rightarrow} q'$ なる q' が存在し、
$(s', q') \in \mathrm{S}$ □

以下、本論文をとおして、$Sys1 =< S_1, Act, \longrightarrow_1, q_1 >$、$Sys2 =< S_2, Act, \longrightarrow_2, q_2 >$ なる TS を前提とする。

定義 4 TS 間のシミュレーション関係

Sys1 と Sys2 を TS をする。$(q_1, q_2) \in \mathrm{S}$ であるような Sys1 から Sys2 へのシミュレーション関係 S が存在するとき、Sys1 は Sys2 を模倣する、あるいは、Sys1 は Sys2 によって模倣される、といい、$Sys1 \ll Sys2$ と書く。 □

これをもとに、LOTOS の仕様間のシミュレーション関係を次のように定義する。

定義 5 LOTOS 仕様間のシミュレーション関係

LOTOS 仕様 Spec1 と Spec2 に対応する TS を、それぞれ Sys1 と Sys2 とする。Sys1 から Sys2 へのシミュレーション関係が成り立つとき、かつそのときに限り Spec2 は Spec1 をシミュレートする、あるいは、Spec1 は Spec2 によってシミュレートされるといい、$Spec1 \ll Spec2$ と書く。 □

4.2 シミュレーション関係の判定アルゴリズム

本節では、TS 間の関係に基づくシミュレーション関係の判定アルゴリズムとその諸性質を示す。判定アルゴリズムは、まず関係 $\mathbf{R}^{(0)}$ を与えることを基本とし、$\mathbf{R}^{(k)}$ を前提として $\mathbf{R}^{(k+1)}$ を導出するという帰納的な定義によって与えられる。

『判定アルゴリズム』

有限な TS Sys1 と Sys2 が与えられたとする。

[基本ステップ] $\mathbf{R}^{(0)}$ を求める
$\mathbf{R}^{(0)}$ として、次のように、全称関係を構成する。

$$\mathbf{R}^{(0)} := S_1 \times S_2$$

[帰納的ステップ] $\mathbf{R}^{(k)}$ から $\mathbf{R}^{(k+1)}$ を求める
$\mathbf{R}^{(k)}$ $(k \geq 0)$ まで求められていたとする。

(1) まず初期的に、$\mathbf{R}^{(k+1)} := \emptyset$ (空集合) とする。

(2) $\mathbf{R}^{(k)}$ の各要素 (s, q) について以下を行う：
各 $\alpha \in Act$ について、$s \xrightarrow{\alpha} s'$ のとき、$q \stackrel{\hat{\alpha}}{\Rightarrow} q'$ なる q' が存在し、$(s', q') \in \mathbf{R}^{(k)}$ が成立するとき、$\mathbf{R}^{(k+1)} := \mathbf{R}^{(k+1)} \cup \{(s, q)\}$ とする。

(3) $\mathbf{R}^{(k+1)} = \mathbf{R}$ ならば (以降、同じ関係しか生成しないから) 終了し、この関係を $\mathbf{R}$ とおく。
$\mathbf{R}^{(k+1)} \neq \mathbf{R}^{(k)}$ ならばこのステップを繰り返す。 □

ここで、Sys1 と Sys2 が有限な TS であれば、この判定アルゴリズムは有限のステップで停止し、帰納的ステップの (3) に示すように関係 R を得る。このようにして得られた関係 R は、Sys1 から Sys2 へのシミュレーション関係であり、R が初期状態対 (q_1, q_2) を含むならば、$Sys1 \ll Sys2$ が成り立つ。また、このとき、関係 R は Sys1 から Sys2 へのシミュレーション関係として最大のものであり、Sys1 から Sys2 への任意のシミュレーション関係 S をその部分集合として含む。逆に、関係 R に初期状態対を含まない場合には、$Sys1 \ll Sys2$ が成り立たない。

4.3 適用例

前節で与えた判定アルゴリズムの、簡単な適用例を示す。

この例は、上位レベルの LOTOS 仕様 Spec1 と、それを詳細化して例外処理を加えた下位レベルの仕様 Spec2 を与え、Spec2 が Spec1 をシミュレートしているかどうかを判定するものである。

```
Spec1 ≡P[a,b,c]
    P[a,b,c] := a;b;stop [] a;c;b;P[a,b,c]
Spec2 ≡Q[a,b,c]
    Q[a,b,c] := a; (b;stop [] c;Q1[a,b,c])
    Q1[a,b,c] := i;Q1[a,b,c] [] i;b;Q[a,b,c]
```

Spec1 と Spec2 に対応する遷移システム Sys1 と Sys2 は、図 3 に示す通りである。

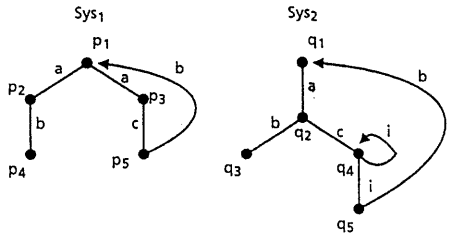

図 3: LOTOS 仕様に対応する遷移システム

これらに判定アルゴリズムを適用すると、次のような手順によってシミュレーション関係 R が得られる。

$$\begin{aligned}
\mathbf{R}^{(0)} &= S_1 \times S_2 \\
&= \{(p_1,q_1),(p_1,q_2),(p_1,q_3),\ldots,(p_5,q_5)\} \\
\mathbf{R}^{(1)} &= \{(p_1,q_1),(p_2,q_2),(p_2,q_4),(p_2,q_5),(p_4,q_1), \\
&\quad (p_4,q_2),(q_4,q_3),(p_4,q_4),(p_4,q_5),(p_3,q_2), \\
&\quad (p_5,q_4),(p_5,q_5)\} \\
\mathbf{R}^{(2)} &= \{(p_1,q_1),(p_2,q_2),(p_4,q_1),(p_4,q_2),(q_4,q_3), \\
&\quad (p_4,q_4),(p_4,q_5),(p_3,q_2),(p_5,q_4),(p_5,q_5)\} \\
\mathbf{R}^{(3)} &= \mathbf{R}^{(2)} \\
\therefore \mathbf{R} &= \{(p_1,q_1),(p_2,q_2),(p_4,q_1),(p_4,q_2),(q_4,q_3), \\
&\quad (p_4,q_4),(p_4,q_5),(p_3,q_2),(p_5,q_4),(p_5,q_5)\}
\end{aligned}$$

従って、関係 R には初期状態対 (p_1, q_1) が含まれているので、$Spec1 \ll Spec2$、つまり Spec2 は Spec1 をシミュレートしていることが結論づけられる。

5 通信サービス仕様への適用

5.1 仕様検証システム

以上に述べたような判定アルゴリズムを利用することにより、図 1 に示した段階的な詳細化過程を支援するため、現在我々は通信システムの高信頼化支援システム ITECS の一部となる LOTOS 仕様の検証システムを開発中である。(図 4)

この仕様検証システムでは、入力として仕様の開発工程における上位レベルの LOTOS 仕様 Spec1 と下位レベルの仕様 Spec2 を与える。システムの内部では、これらの仕様をそれぞれ対応する遷移システム Sys1 と Sys2 に変換する。次に、これらの遷移システムに判定アルゴリズムを適用し、関係 R を得る。これにより、仕様間の対応関係がわかる。もし、R に Sys1 と Sys2 の初期状態対が含まれていない場合には、下位レベルの仕様 Spec1

が上位レベルの仕様 Spec2 を正しく詳細化していないことがわかる。最終的に、この検証結果を画面上に表示する。

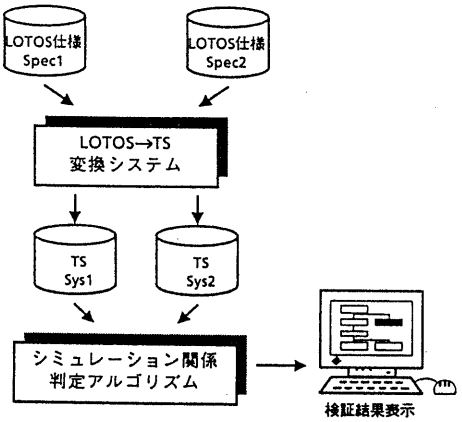

図 4: 仕様検証システム

以下、本検証システムを利用した実際の通信サービスの仕様記述と検証の適用例を示す。具体的には、まず高信頼化支援システム ITECS の一部である、LOTOS 仕様の図的支援システム GLOER を用いてサービス仕様の記述を行い、さらにそれを詳細化した仕様を記述する。次に、本検証システムを利用して、詳細化が正しく行われているかどうかの検証を行う。

5.2 通信サービスのモデル

記述対象として、図 5に示すような通信サービスを考える。これは ISDN の回線交換サービスの、接続から切断までを簡単にモデル化したものである。

まず、通信サービスの要求仕様をレベル 1(上位仕様のレベル) と考える。レベル 1 では、ネットワーク全体と端末 A および B とのインタラクションを仕様化するレベルであり、Setup 等の信号が規定される。ネットワーク内部の動作に関しては規定しない。

次に、レベル 1 から一段階の詳細化を行ったものがレベル 2(下位仕様のレベル) である。レベル 2 では、ネットワークを発信局、中継局、着信局に分け、共通線信号 (CCS) 網によって局間でやりとりされる、IAM 等の信号を規定している。ただし、図 5の信号シーケンスでは代表的な正常ルートとして、端末 A 側から発信される Disc のシーケンスのみを示している。実際には端末 B からの Disc もあり、シーケンスは複雑になるが、ここでは省略している。

5.3 LOTOS 記述例

上記のモデルをもとに、G-LOTOS による仕様記述環境 GLOER を用いて、仕様記述を行う。まず、レベル 1

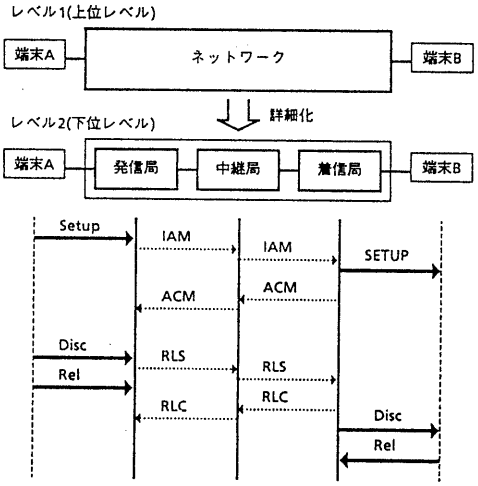

図 5: 通信サービスのモデル

を G-LOTOS によって記述し、T-LOTOS への変換を行う。GLOER の実行画面イメージを図 6に、変換された T-LOTOS を図 7に示す。

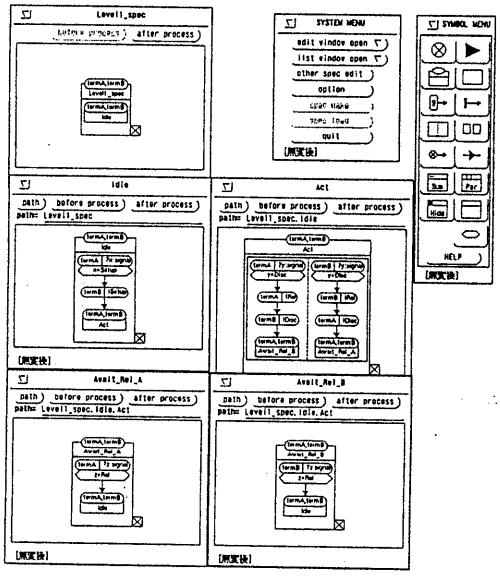

図 6: GLOER 実行画面

次に、G-LOTOS のもつプロセスの内包的な記述機能によって、仕様の詳細化を行い、レベル 2 を記述する。図 10が、T-LOTOS によるレベル 2 の仕様である。

5.4 シミュレーション関係

以上で得られた、レベル 1 およびレベル 2 の LOTOS 仕様から、それぞれの遷移システムを生成し、それをら

```
specification Level1[termA,termB]:noexit:=

type Signal is
    sotrs  signal
    opns   Setup    :   →signal
           Disc     :   →signal
           Rel      :   →signal
endtype

behaviour
    Idle[termA, termB]

    where
    process Idle[termA, termB]:noexit:=
        termA?x:signal[x=Setup];
        termB!Setup;
        Act[termA, termB]

        where
        process Act[termA, termB]:noexit:=
            termA?y:signal[y=Disc];
            termB!Disc;
            termA!Rel;
            (* await Rel *)
            termB:x:signal[z=rel];
            Idle[termA, termB]
            []
            termB?y:signal[y=Disc];
            termA!Disc;
            termB!Rel;
            (* await Rel *)
            termB?z:signal[z=Rel];
            Idle[termA, termB]
        endproc
    endproc
endspec
```

図 7: レベル 1(上位レベル) の LOTOS 仕様

仕様検証システムによって検証する。図 8は、検証システムの実行画面である。

検証結果として、最終的に以下のようなシミュレーション関係 R が得られる。

$$R = \{ (1,1),(1,33),(1,35),(2,2),(2,3),(2,4), (3,5),(3,6),(3,8),(4,11),(6,14),(6,18), (6,23),(6,27),(6,32),(8,31),(8,36),(5,7), (5,10),(7,9),(7,12),(7,15),(7,19),(7,24), (7,29),(9,28),(9,34) \}$$

関係 R が示しているのは、それぞれがシミュレーション関係となっている状態の組である。ここで示される状態の番号は、LOTOS から遷移システムの変換システムによって生成された遷移システムにおける、状態の番号である。図 9に、レベル 1 とレベル 2 の遷移システムの一部と、システムによって得られたシミュレーション関係のイメージを示す。

図 9では、スペースの都合により各アクションは省略した表現となっている。例えば、`a!Rel` は `termA!Rel` のことであり、`a?Setup` は `termA?x:signal[x=Setup]` という表現を省略したものである。また、矢印とその先の番号は、その状態番号に遷移することを示し、遷移システムを簡略化している。

得られたシミュレーション関係 R には、遷移システムの初期状態対が含まれているので、レベル 2 の LOTOS 仕様はレベル 1 の LOTOS 仕様をシミュレートしている、つまり正しく詳細化していることが結論づけられる。ここで、レベル 2 の LOTOS 仕様がレベル 1 の LOTOS 仕様をシミュレートしているとは、レベル 1 の仕様において規定された動作は、レベル 2 においても正しく実行可能である、ということである。ここで、要求仕様としては代表的な信号のシーケンスしか与えていないため、通常の詳細化を行い、例えばこの場合には内部をいくつかの交換機に分割すると、信号の逆転等が起る場合がある。従って、あらかじめ詳細なシーケンスまで要求仕様として与えられていないと、一般的な等価関係は成り立たないことがわかる。ここにシミュレーション関係を用いた意味がある。

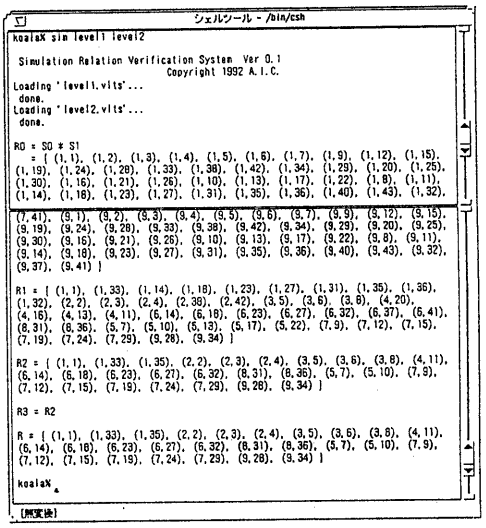

図 8: 検証システムの実行画面

ただし、単純な詳細化を行うだけでは、シミュレーション関係が成立したとしても仕様中に不具合が残る場合がある。例えば上記のレベル 2 の仕様では、A、B 双方の端末から Disc 信号が発せられるとデッドロックになることがある。しかしながら、このような仕様の誤りは、LOTOS から遷移システムへの変換システムの出力結果より、容易に検出が可能である。従って、単純な詳細化を行い、シミュレーション関係が成り立っていることを確認しながら、このようなデッドロック等の回避のための処理を組み込むことによって、誤りのない高信頼な仕様の詳細化が可能であると考えられる。

6 おわりに

本報告では、LOTOS 仕様に対するシミュレーション関係を定義し、その判定アルゴリズムを示した。さらに、通信ソフトウェアの高信頼化支援システム ITECS

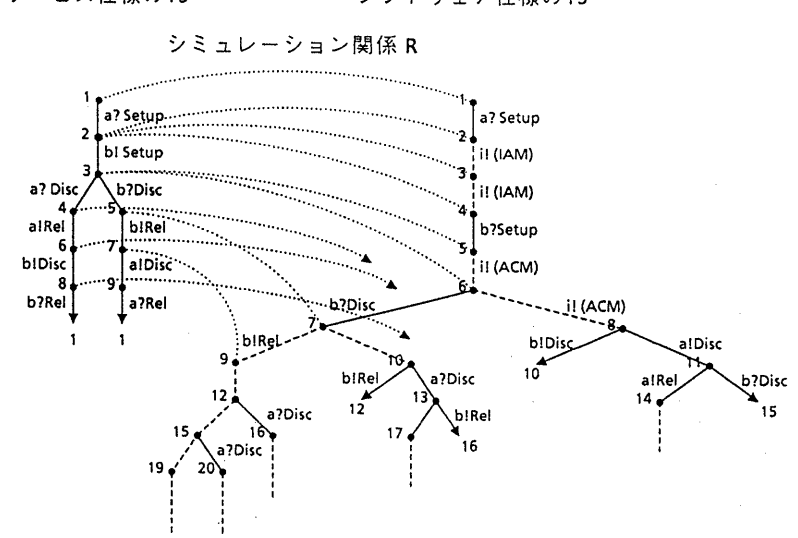

図 9: 遷移システムとシミュレーション関係

を提案し、その中で段階的な仕様の詳細化を行うために、シミュレーション関係の概念を適用し、検証を行うことが有効であることを示した。

今後の課題としては、より実用的な検証システムの開発と、実際の通信システム等の仕様記述への適用が考えられる。また、シミュレーション関係による単一の検証法のみでは、正しい詳細化を行う上で足りない部分もあるので、他の関係との組み合わせによる検証についても検討中である。

謝辞

最後に、本研究を行うにあたって、数々の有益な助言をして頂いた東北大学の野口正一教授、白鳥則郎教授に深く感謝致します。また、本研究の機会を与えてくださった当研究所の緒方常務、ならびに日頃有益な検討をして頂いた当研究所の研究員の皆様に感謝致します。

参考文献

[1] ISO : "Information processing systems - Open Systems Interconnection - LOTOS - A formal description technique based on the temporal ordering of observational behaviour," ISO8807 (1989).

[2] ISO : "Revised Text of ISO 8807/PDAM 1, Information processing systems - Open Systems Interconnection - LOTOS - A formal description technique based on temporal ordering of observational behaviour - Draft Amendment 1: G-LOTOS," ISO/IEC JTC 1/SC 21 N 6751 (1992).

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

[4] 更科 他 : "Graphical-LOTOS 仕様記述支援 - GLOER : Graphical-LOTOS Editor -," 第1回「通信ソフトウェアのための新しい方法論」ワークショップ, A-2 (1992).

[5] E.Brinksma, Giuseppe Scollo and Chris Steenbergen : "LOTOS Specifications, Their Implementations and Their Tests," Protocol Specification, Testing, and Verification VI, pp.349-360 (1987).

[6] N.Shiratori, H.Kaminaga, K.Takahashi and S.Noguchi : "A Verification Method for LOTOS Specifications and Its Application," Proc. of 9th IFIP WG 6.1 International Symposium on Protocol Specification, Testing and Verificaton (1989).

[7] K.Yamano, D.Jokanovic, T.Ando, M.Ohta and K.Takahashi : "Formal specification and verification of ISDN services in LOTOS," IEICE Transactions on Communications, E75-B, No.8 (1992).

[8] 高橋 薫, 山野 敬一郎, 太田 正孝 : "プロセス仕様の検証のための模倣性判定法," 電子情報通信学会論文誌 D-I 掲載予定.

```
specification Level2[termA,termB] :noexit :=

type Signal is
    sotrs  signal
    opns   Setup     :   →signal
           Disc      :   →signal
           Rel       :   →signal
endtype

type ISUPsignal is
    sorts  isup
    opns   IAM       :   →signal
           ACM       :   →signal
           RLS       :   →signal
           RLC       :   →signal
endtype

behaviour
    Idle[termA, termB]

    where
    process Idle[termA, termB] :noexit :=
    hide ccsAs, ccsAr, ccsBs, ccsBr in
        Idle__A[termA,ccsAs,ccsAr]
        |[ccsAs, ccsAr]|
        NodeT[ccsAs,ccsAr,ccsBs,ccsBr]
        |[ccsBs,ccsBr]|
        Idle__B[ccsBs,ccsBr,termB]

        where
        process Idle__A[termA,ccsAs,ccsAr] :noexit :=
            termA?x:signal[x=Setup];
            ccsAs!IAM;
            ccsAr?y:isup[y=ACM];
            Act__A[termA,ccsAs,ccsAr]
        endproc

        process Act__A[termA,ccsAs,ccsAr] :noexit :=
            ccsAr?x:isup[x=RLS];
            ccsAs!RLC;
            termA!Disc;
            (* await Rel *)
            termA?y:signal[y=Rel];
            Idle__A[termA,ccsAs,ccsAr]
            []
            termA?x:signal[x=Disc];
            termA!Rel;
            ccsAs!RLS;
            (* await RLC *)
            ccsAr?y:isup[y=RLC];
            Idle__A[temA,ccsAs,ccsAr]
        endproc

        process NodeT[ccsAs,ccsAr,ccsBs,ccsBr] :noexit :=
            Link__T[ccsAs, ccsBr]
            |||
            Link__T[ccsBs, ccsAr]
            where
            process Link__T[send, receive] :noexit :=
                send?sig:isup;
                receive!sig;
                Link__T[send, receive]
            endproc
        endproc

        process Idle__B[ccsBs,ccsBr,termB] :noexit :=
            ccsBr?x:isup[x=RLS];
            termB!Setup;
            ccsBs!ACM;
            Act__B[ccsBs,ccsBr,termB]
        endproc

        process Act__B[ccsBs,ccsBr,termB] :noexit :=
            ccsBr?x:isup[x=RLS];
            ccsBs!RLC;
            termB!Disc;
            (* await Rel *)
            termB?y:signal[y=Rel];
            Idle__B[ccsBs,ccsBr,termB]
            []
            termB?x:signal[x=Disc];
            termB!Rel;
            ccsBs!RLS;
            (* await RLC *)
            ccsBr?y:isup[y=RLC];
            Idle__B[ccsBs,ccsBr,termB]
        endproc
    endproc
endspec
```

図 10: レベル 2(下位レベル) の LOTOS 仕様