ソフトウェア工学 101−12
（1994. 11. 18）

時制論理を用いたＬＯＴＯＳ仕様生成統合支援システムの作成

伊藤 光裕[1]、布施 和博[1]、安藤 敏彦[1]、加藤 靖[1]、高橋薫[2]

[1] 仙台電波高専
〒 989-31 仙台市青葉区上愛子字北原 1 番地
[2] (株) 高度通信システム研究所
〒 989-32 仙台市青葉区南吉成 6-6-3

概 要

形式記述技法（FDT）は、仕様を曖昧なく記述するために開発されたものである。FDT の一つである LOTOS は、仕様の検証・試験を行う際に大きな力を発揮する。本論文では LOTOS 仕様生成のための統合支援システムを紹介する。このシステムはワークステーション上に実現され、使いやすいユーザインターフェースを備えている。このシステムは時間的な性質から LOTOS 仕様を構成することや、トレース等価、模倣等価、弱双模倣等価といった等価性の検証、また時間的性質の検証を行うことができる。このシステムは LOTOS での段階的仕様化に有効であると思われる。

Development of Integrated Support System for Generating LOTOS Specifications using Temporal Logic

Mitsuhiro ITO[1],Kazahiro FUSE[1],Toshihiko ANDO[1],Yasushi KATO[1],Kaoru TAKAHASHI[2]

[1]Sendai National College of Technology.
1, Kitahara, Kamiayashi, Aoba-ku, SENDAI, 989-31, JAPAN,
Tel: +81-22-392-4761, Fax: +81-22-392-3359.
[2]AIC System Labs. Ltd.,
6-6-3, Minami-Yoshinari, Aoba-ku, SENDAI, 989-32, JAPAN,
Tel: +81-22-279-3310, Fax: +81-22-279-3640.

Abstruct

Formal Description Techniques(FDTs) have been developed to specify unambiguously. LOTOS is one of FDTs and have a strong power for verification and testing. In this paper, we report an integrated support system for generating LOTOS specifications. This system is implemented on a workstation and have a useful user interface. This system allows the composition of LOTOS specification from temporal properties, the verification of equivalence relation, such as trace equivalence, simulation equivalence and weak bisimulation equivalence, and the verification of temporal properties. We hope that this system can be effective for a step-wise specification in LOTOS.

1 はじめに

我々がある大きなシステムを作る際、通常分担作業という形式をとる。つまり一つのシステムの作成に複数の人間がかかわるわけだが、一人一人の担当部分を明確にするために担当内容を明文化する必要がある。また、そのシステムが完成し、運用・保守する場合もまた別の人間が担当することが多い。この時も、システムがどのような動作を行なうかを明確にしておく必要がある。このようにシステムの動作を明文化する場合、我々が普段、日常生活で使う自然言語を用いると、様々な問題が生じてくる。まず、自然言語を使うとどうしてもその内容に曖昧さが残ってしまう。様々な背景を持った人間がこれを読むわけであるから、その解釈に微妙なニュアンスの違いが出てきてしまう。また、特定の自然言語を使うと、その言葉を通常使わない人にとっては理解しにくく、間違った解釈をする原因ともなる。人間にとって微妙な違いでも、そのシステムを作るという作業においては、時に重大な障害をきたす場合がある。

そこで、曖昧な表現を避けるため、万国共通の言語ともいえる数学をベースにした記述法が提案されている。形式記述技法（Formal Description Technique:FDT）と呼ばれるもので、LOTOSはその一つとしてISOによって国際標準化されたものである[1]。LOTOSを用いることのメリットは形式記述ができるだけでなく、仕様の解析ができる点にある。LOTOSでは処理プロセスをブラックボックスとして扱い、それを外部から見たときに観測できる入力と出力のみに注目する。これにより、段階的な詳細化が可能であり、トップダウン的にシステムを設計する際、非常に便利である。

LOTOS関連のツールとしてはSAL[2]、LOLA[3]等があるが、これらは既に固まった仕様をLOTOSで記述することを支援する。一方、ユーザ要求から仕様を求めるといったより上流の行程は手作業で行なわれており、これを支援する必要があった。そこで、当研究ではトップダウン的な仕様記述方針に基づいた、LOTOS仕様の生成を支援するための統合支援システムを開発している。このシステムは次の3つのタイプの生成方法により、仕様導出を支援することを目標としている。

（1）直接LOTOSを記述して、いくつかの等価性を検証しながらの段階的詳細化。

（2）ユーザの要求を時制論理で記述し、それを満足するLOTOS仕様の合成。

（3）LOTOSの意味であるラベル付き遷移システム（Labelled Transition System:LTS）を使ったLOTOS仕様の合成。

このシステムは仕様生成の様々なレベルで、設計者を支援するので仕様設計の効率を高めるものと期待される。

2 時間的性質からLOTOS仕様を求めるアルゴリズム

我々はユーザの要求をLOTOS記述に反映させるための方法として、要求する時間的性質を記述し、これを満足するLOTOS記述を合成する方法を提案している[4]。ここではその合成方法について述べる。

要求される性質を表現するためには、時制論理を用いる。時制論理とは様相論理の一種であり、イベント系列上の性質を直接表現する事ができる。ここで使用している時制論理の演算子は、表1に示されている。アルファベットの大文字で表される演算子は、記述対象のシステムの全アクション系列でその性質を満足することを表し、小文字で表されたものは1つ以上のアクション系列でその性質を満足することを表す。

表1. 時制論理の演算子

P ⊕ Q	P または Q
ALWAYS P,always P	常に P
SOMETIME P,sometime P	いつか P
NEXT Q ,next Q	次が Q
P UNTIL Q,P until Q	Q まで P
P UNLESS Q ,P unless Q	Q でなければ P
P BEFORE Q,P before Q	P の前が Q
FREQ P,freq P	しばしば P
P ⇒ Q	P ならば Q

例として、これらを用いて簡単な自動販売機が満たすべき性質を記述すると以下のようになる。

ALWAYS(Coin ⇒ (sometime Coffee) ⊕ (sometime Tea))

これは「お金を入れたならば、いつかお茶が出る、またはいつかコーヒーが出る、ということが常におこる」という性質を意味している。あるシステムの仕様を設計する場合、大局的な性質が与えられる事が多く、そのような性質は、FDTよりも時制論理の方が記述しやすい。

そこで次のようなステップで、与えられた時間的性質からそれを満足するようなLOTOS仕様を合成する。

＜1＞時間的性質を与える。

＜2＞各時間的性質に対応する標準的なLTSを求める。

＜3＞＜2＞で求めたLTSを直列に接続し、修正する。

＜4＞＜3＞で得られたLTSをLOTOS記述に変換する。

一般に一つの時間的性質に対して、それを満足するLTSは無数にある。そこで時制論理の各演算子や特定の時制論理式を満足する極小のLTSを、テンプレートとして定義し、一つの時間的性質から一つのLTSが定まるようにした。これにより、与えられた時間的性質を満足する極小のLTS、従って極小のLOTOS仕様が得られる。用意したテンプレートを図1に示す。このテンプレートのLTSに見られるhはダミーアクションである。ダミーアクションは空であるか、または他のアクション系列で置き換える事ができる。これはLTSの合成を容易にするために導入されたものである。

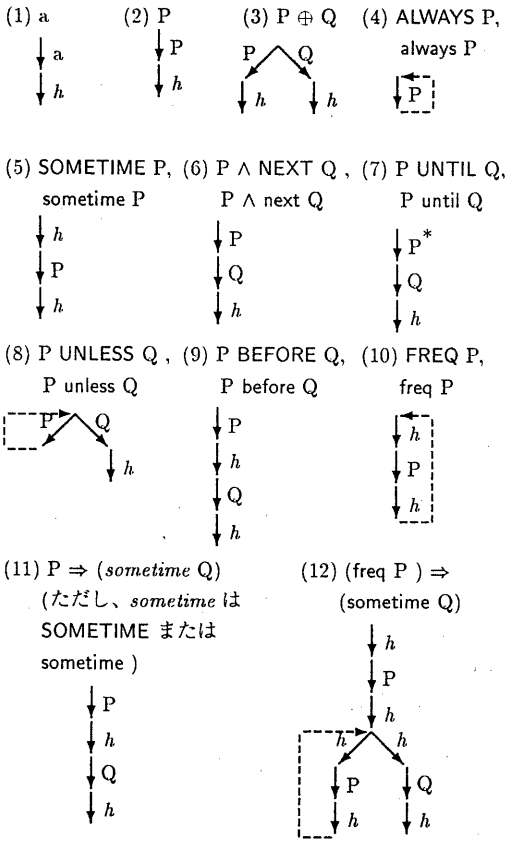


図1 時制論理の演算子などに対応するテンプレート

次に合成の例を示す。

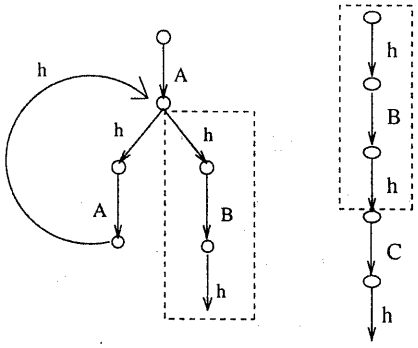
＜1＞ まず、以下のような時間的性質を与える。

1. **ALWAYS(freq A ⇒ (sometime B))**
2. **ALWAYS(B ⇒ sometime C)**

性質1は、「しばしばAならば、いつかBであり、この事象が常に起こる」という意味である。性質2は、「Bならばいつかcである、ということが常に起こる」という意味を表している。

＜2＞ 次にこれらに対するLTSを導出する。

テンプレートを用いて、時間的性質1,2に対応するLTSがそれぞれ導出される（図2）。



(a) 性質1に対応するLTS (b) 性質2に対応するLTS

図2 テンプレートを用いて導出したLTS

＜3＞ それぞれのLTSを接続し修正する。

接続できるのは、一方のLTSの先端を含む部分と、もう一方のLTSの初期状態を含む部分が一致する場合である。図2では破線で囲まれた部分がそれに当てはまるのでそこを重ね合わせる。また、ループに分岐するダミーアクションhを内部アクションiに置き換える。必要ならば他のhアクションをiで置き換えてもよい。最後に不要なhアクションを消去する。こうして合成されたLTSが図3である。

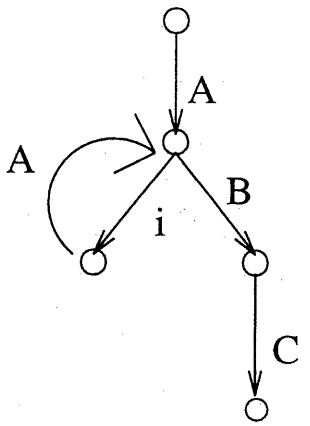


図3 接続して導出されたLTS

＜4＞最後にLOTOS仕様に変換する。

```
process EX[A,B,C]:=
        A;proc1[B,C,D]
  where
  process proc1[A,B,C]:=
     hide int in
        int;A;proc1[A,B,C]
      [ ]B;C;exit
  endproc
endproc
```

3 本システムの構成

設計者はまず、設計対象システムの大まかな仕様をLTSを用いて作成する。作成に当たっては自分ではじめから構成しても、時間的性質から自動生成してもよい。次に作成したLTSのうち、処理プロセスの部分を詳細化してゆく。そして、詳細化したLTSが詳細化前のLTSと等価であるかどうかを調べる。この際、弱双模倣等価、模倣等価、トレース等価の3つの種類についてのチェックが行なえる。詳細化されたLTSは最終的にLOTOS記述へと変換される。

このようにして、トップダウン的に仕様化対象となるシステムを詳細化してゆくわけだが、詳細化の際に等価性や時間的性質のチェックを行ないながら進めてゆくことで、詳細化前の仕様にフィードバックしながら設計を行なうことができる。

LOTOSによる仕様の設計法として、以下の方法が考えられる。

（1）システムをブラックボックスと見なしたときに、その内部を顕在化しながら段階的に詳細化してゆく。

（2）システムの大局的な性質を満足するような仕様を合成する。

（3）システムの動作を具体的にイメージしながら仕様を作成する。

この場合、（1）においては詳細化の前後で、ある等価性、例えば弱双模倣等価性が保存されているかどうかを検証しながら詳細化を進める必要がある。また、（2）では得られたLOTOS仕様が、元の時間的性質を満足しているかどうか検証したり、より適正なスタイルとなるよう編集できる必要がある。（3）においてはシステムの動作が適切に表現され、かつその表現からLOTOS記述が導出できるようになっていなければならない。

本システムはこのような要求を実現するものとして構想された。

本システムは図4のように構成された統合環境を与えている。この中には以下のようなモジュールが統合されている。

（1）テキストエディタ (Text Editor)

（2）時制論理からLTSへの変換モジュール (TP to LTS)

（3）LTS編集モジュール (LTS Editor)

（4）LTSからLOTOSへの変換モジュール (LTS to LOTOS)

（5）LOTOSからLTSへの変換モジュール (LOTOS to LTS)

（6）LTSの等価性検証モジュール (Equiv. Check)

（7）時間的性質の検証モジュール (TP Check)

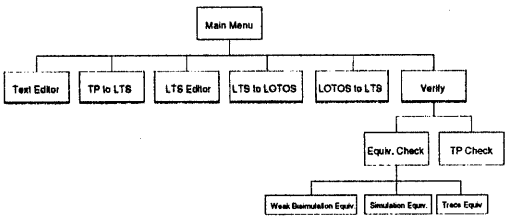


図4 システム構成図

このうち（1）、（5）、（7）には、それぞれ既存のEmacs、SAL[2]、堰合[5]を使っているが、その他のモジュールは当研究で開発している。本システム内では、LOTOS記述を直接扱わずLTSを用いて様々な処理を行なっている。LTSはLOTOSプロセスの動作を有向グラフで表したものであり、LOTOS記述よりもその振舞いを直感的に理解しやすく、合成・検証などの処理に使いやすいためである。また、LTSからほかのFDT言語に変換することもできるという利点もある。上記の各モジュールはLTSファイルを媒介としてデータの受け渡しを行なっている。LTSは、通常以下の4項組で表される。ここで示すLTSをSysとすると、

Sys ＝ ＜S, Act, →, σ＞

となる。ここで、Sは状態の集合、Actはアクションの集合、→は遷移関係（→$\subseteq$S × Act × S）、σはSysの初期状態（$\sigma \in$S）である。

このシステムで扱うLTSファイルは、この集合の要素をテキスト形式で記録したものであり、以下のような構成となっている。

LTS名 ＝ ＜{状態名, ‥‥}, {アクション名, ‥‥}, {(遷移関係), ‥‥}, 初期状態名＞

例）

Sys1 = ＜{1, 2, 3, 4, 5}, {a, b, c, i},
{(1, a, 2), (2, b, 3), (2, i, 4), (4, c, 5)}, 1＞

3.1 時間的性質からのLTSの導出

時間的な性質を時制論理を用いて与えてやることにより、この性質を満足したLTSが得られる機能である。時制論理の演算子を用いて、簡単な自動販売機の時間的性質とそこから導かれたLTSの例を示す。この自動販売機には図5に示す時間的性質を与える。この性質の意味は「常に、Coinを入れれば、いつかコーヒーもしくは、いつか紅茶が出てくる」である。その時間的性質に対する出力として、図6に示されるLTSが得られる。

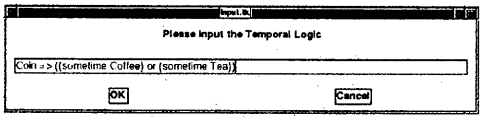


図5 自動販売機の時間的性質

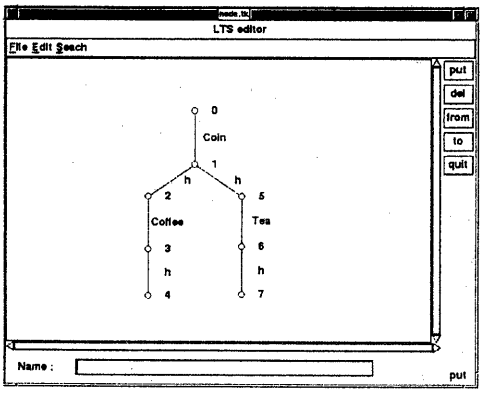


図6 導出されたLTS

3.2 LTSの編集

この機能を用いることにより、LTSを画面上に表示することはもとより、ユーザがはじめからLTSを構成したり、時間的性質から導出されたLTSを編集することができる。
この機能を提供するウインドウには、

- 中心部にLTSを表示・編集するフィールド
- フィールドの右端及び下端に、フィールドをスクロールさせるためのスクロールバー
- フィールドの上部に、ファイル関係の機能や編集の機能を備えたメニューバー
- フィールドの右側に、状態やアクションの生成・削除に関連したモード選択ボタン
- フィールドの下には状態・アクションの名前を入力するエントリーフィールド

が配置されている。(図7)

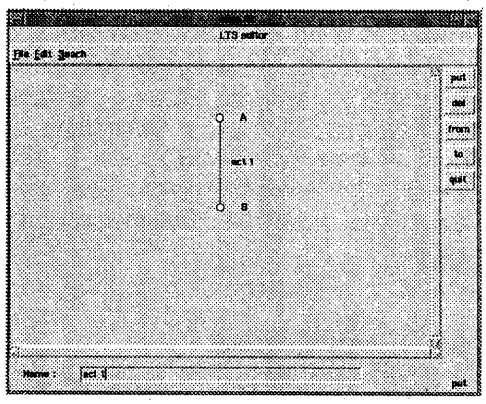


図7 LTS Editor

状態・アクションの生成に関しては、以下のモードがあり、ウインドウ右側のモード選択ボタンにより切り替えることができ、ウインドウ右下には、現在のモードが表示される。なお、起動直後は状態生成のモードとなっている。

(1) 状態の生成

状態の生成は、ウインドウ右側のボタンでモード選択し、そのあとフィールドの任意の位置でマウスの左ボタンをクリックすることで行なわれる。状態に名前を付ける場合は、フィールド下のエントリーフィールドで名前を入力してから状態を生成することで可能となる。

(2) アクションの生成

アクションは、遷移元と遷移先の2つの状態をフィールド内で指し、マウスの中ボタンをクリックする、またはモード選択後、遷移元状態をマウスの右ボタンで選択し、同様に遷移先状態も選択することで生成される。これもまた生成時にエントリーフィールドに書かれた名前をつけることができる。

(3) 状態・アクションの削除

状態・アクションの削除は、削除モード選択をした後、削除したい状態・アクションをフィールド上で指し、マウスの左ボタンをクリックすることで行なわれる。

LTSの編集は基本的にメニューバーに配置された機能から選択する。メニューバー上のEditをマウスの右ボタンでクリックするとメニューが現れ、カット、コピー、ペースト、全クリアの各機能を選択できる。

カット・コピーはメニューから機能選択後、カットする範囲をマウス左ボタンで指定し、ペーストする際はメニューから機能選択後、貼り付けたい位置をマウスの左ボタンで指定する。また、全クリアはフィールド上の全状態・アクションを消去する機能で、機能選択後、確認ウインドウのボタンをクリックすることで実行される。また、フィールド上でどのような場合でも状態をマウスの右ボタンでドラッグすると、状態を移動することができる。状態の移動に連動し、アクションも同時に移動する。なお、ファイルのセーブやロードはメニューバー上のFileから選択し、ロードの場合はファイルを候補のなかから選ぶ。またセーブの場合は、ファイル名を指定してから書き込む。

3.3 LTSの性質の評価

メインメニューからVerifyを選択すると、
(1) 時間的性質のチェック
(2) 2つのLTS間の等価性のチェック
を選ぶことができる。

3.3.1 時間的な性質の検証

時間的性質の検証は、キーボードからチェックしたい性質を記述することで、システムがその性質を満たしているかどうかを判断し、その結果を得ることができる[5]。設計者がはじめからLTSを作った場合や、時間的性質から導出されたLTSを編集した場合、時間的な性質をチェックすると、そのLTSの大局的な振舞いを知ることができる。つまり、段階的な仕様の詳細化の際、詳細化後に得られた仕様が、詳細化前の仕様が満たしていた時間的な性質を満足しているかどうか、などの検証に用いることができる。
メニューから時間的性質の検証を選択し、チェックしたい性質を入力する。
その性質を満たしているかどうかのチェックをシステムが自動的に行ない、その結果が出力される。

3.3.2 2つのLTS間の等価性の検証

これは、2つのLTSが等価な振舞いを行なうかどうかのチェックである[6),7)]。
この等価性チェックには以下の3つのレベルがある。このうちの一つを選び、またチェックしたい2つのLTSのファイルを選択すると、選ばれた等価性のチェックをシステムが自動的に行ない、等価性があるかどうかを判定する。
(a) 弱双模倣等価 (Weak Bisimulation Equivalence)
(b) 模倣等価 (Simulation Equivalence)
(c) トレース等価 (Trace Equivalence)

(a) 弱双模倣等価
弱双模倣等価は、2つのLTSの間に、内部アクションを無視した同一のアクション系列で遷移する対応した状態が、全ての状態について存在する場合等価であるとするものである。つまり、2つのLTSがどの状態からも、全く同一のアクション系列しかとらず、外部観測上全く区別が付かない場合を等価としている。

この等価性のチェックは、例えばOSIアーキテクチャにおけるサービス仕様を元の仕様とし、プロトコル仕様をその詳細化と考えた場合、プロトコル仕様がサービス仕様と矛盾しないかどうかを判断する場合などに利用できると思われる。
(b) 模倣等価
LTS Sys1のある状態から、同一のアクション系列をとる状態がもう一方のLTS Sys2に存在し、これがSys1の全ての状態について存在する場合、模倣関係があるという。この場合、Sys2はSys1を模倣する関係にあるが、この関係が双方向で成り立つとき模倣等価であるという。各状態から同一のアクション系列がとれれば、模倣関係は成立するので、模倣する側のLTSに、模倣される側はとれないアクション系列が存在してもよい。

この等価性のチェックは、システムの設計を、例外機能など機能拡張しながら行なう際、元の仕様を満足しているかどうかを判断する場合などに利用できると思われる。
(c) トレース等価
トレース等価は、2つのLTSにおいて初期状態から同一のアクション系列で遷移することができる場合に等価であるとするものである。これは、模倣関係において初期状態からの遷移だけに注目した場合と見ることができる。

この等価性のチェックは、外部観測的な振舞いの等価性よりは、2つのシステムが同じことをできるかどうかのチェックに用いられ、元の仕様と後の仕様の間の緩い等価性を表すものである。
この3つの等価性の間には以下のような関係が成り立つ。

双模倣等価 $\Longrightarrow$ 模倣等価 $\Longrightarrow$ トレース等価

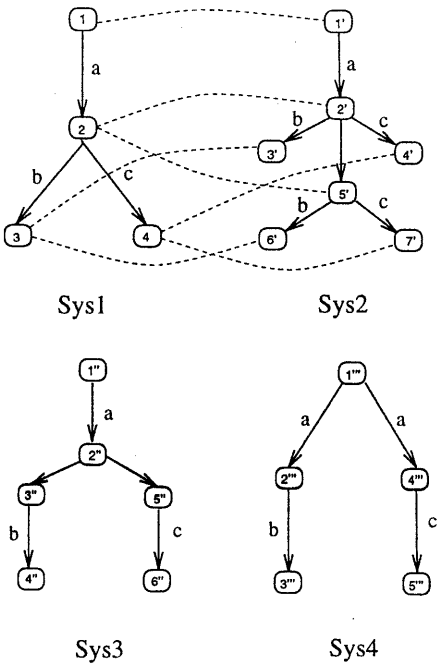


図8 各等価性の関係を示す例

図8に、それぞれの等価性の例を示す。ここで、Sys2はSys1と弱双模倣等価である。これは、Sys1の各状態に対応する状態がSys2に存在しているからである。またSys3はSys1の模倣等価であるが、弱双模倣等価ではない。これは、Sys1およびSys3の各状態から同じアクションの遷移を取ることはできるが、すべての状態が対応していないためである。そして、Sys4はSys1のトレース等価であるが、模倣等価および弱双模倣等価ではない。これは、Sys1とSys4が、初期状態からのみ同じアクションの遷移を取ることができるからである。

3.4 LTSからのLOTOSへの変換

先に述べたように，LTSはLOTOSの意味を表しているので、根から順に記述していくことによってLOTOS記述に変換することが可能である。LOTOSではそれぞれの事象の起こり得る順番に;(プレフィクス) を用いて連結してゆく。また分岐がある場合 [] (チョイス) でそれらを連結してゆく。ループがある場合は、合流点をサブプロセスとして表現する。LOTOSではサブプロセスもアクションと同様に扱う事ができ、構造化して記述することが可能である。

変換されたLOTOS記述は図9のように無味乾燥なプロセス名が付けられるが、Emacs上で意味のある名前に書きかえることができる。

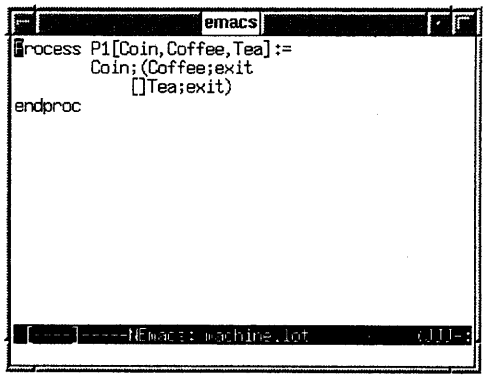


図9 導出されたLOTOS記述

3.5 LOTOSからのLTSの導出

LOTOS記述を変更した後改めて検証を行なうために、SALを用いてLOTOSからLTSを導出する。本システムではSALを子プロセスとして利用している。SALの詳細は文献2)を参照のこと。

この機能があることによって設計過程を現段階からより上流の段階へフィードバックさせることが可能である。

3.6 他のLOTOS関連ツールとの比較

前に述べた通りLOTOS関連ツールの代表的なものにSAL,LOLAがある。SALは与えられたLOTOS仕様の動作のシミュレーションを行うツールであり、LOLAはシミュレーションの他、編集も行うことができる。これらはすでに設計が固まった仕様をLOTOSで記述するための支援ツールである。

前に述べた通りLOTOS関連ツールの代表的なものにSAL,LOLAがある。SALは与えられたLOTOS仕様の動作のシミュレーションを行うツールであり、LOLAはシミュレーションの他、編集も行うことができる。これらはすでに設計が固まった仕様をLOTOSで記述するための支援ツールである。

一方より上流の工程を支援する本システムは、今のところ編集機能こそLOLAには及ばないものの、GUIの採用や等価性の検証機能、時間的性質からの導出機能など、ユーザ要求から仕様を設計してゆく上で有効な機能を備

えている。
以下にLOLAと本システムを比較した表を示す。

表2. 本システムとLOLAの比較

	本システム	LOLA
シミュレーション	○	○
GUI	◎	×
編集	△	◎
性質の検証	○	×
時間的性質からの導出	○	×

4 まとめ

本システムを用いることにより、設計者は多角的な観点からLOTOSによる仕様記述ができると思われる。仕様記述は、厳密さを要求されるものであるが、それを行なうためにはトップダウン的な方法だけでは不十分で、様々なレベルでのフィードバックが必要になる。本システムはそれを可能とし、早い段階で仕様の間違いの検出・検証ができるということは、仕様設計の効率を向上させるであろう。

現在このシステムの、時間的性質からLTSを導出する機能、LTSからLOTOS記述を導出する部分、LTSの時間的性質をチェックする部分については完成しているが、LTSを編集する機能の一部と、等価性を判定する機能については開発中である。それらの点を完成させるとともに、LOTOSからLTSを導きリアルタイムに表示する機能や、他のFDT言語への対応、並列システムの記述を可能にするなどの改良を重ねていきたいと思う。

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) 佐藤, 他："SAL:Semantics Analyzer for LOTOS Specifications", 情報処理学会マルチメディア通信と分散処理研究会資料, 1990.

3) S.Pavón, D.Larrabeiti, J.Quemada : LOLA User Manual(version 3.4). E.T.S.I. de Telecomunicacion Univ. Politécnica de Madrid, 1993.

4) 安藤敏彦, 加藤　靖, 高橋　薫, 野口正一 : 時制論理に基づくプロトコルのLOTOS仕様の合成, 情報処理学会論文誌, Vol 34, No.6, pp.1268-1280, (1993).

5) 堰合宏史 : LOTOSと時制論理に基づくプロトコル検証, 仙台電波工業高等専門学校平成4年度卒業論文, 1993.

6) Kanellakis P.C. ,Smolka S.A. : CCS expression, finite state processes and three problems of equivalence, Information and Computation, vol.86, pp.43-63(1990).

7) Bolognesi T., Smolka S.A. : Fundamental results for the verification of observational equivalence:a survey, Protocol Specification, Testing and Verification VII, pp.165-179, North-Holland(1987).

8) 布施和博, 伊藤光裕, 安藤敏彦, 加藤　靖 : LOTOS仕様生成統合支援システム, 仙台電波工業高等専門学校研究紀要, 1994.

9) 伊藤光裕, 布施和博, 安藤敏彦, 加藤　靖 : 時制論理に基づくLOTOS仕様導出アルゴリズムの実現, 仙台電波工業高等専門学校研究紀要, 1994.