# 修 士 論 文 概 要 書

CD

2008 年 2 月提出　　学籍番号 3606U021 – 5

| | | | | | |
|---|---|---|---|---|---|
| 専攻名 | 情報・ネットワーク | 氏 名 | 岡部 亮 | 指 導 教 員 | 上田 和紀 印 |
| 研究指導名 | 並列知識情報処理 | | | | |
| 研 究 題 目 | 階層グラフ書換え言語 LMNtal によるモデル検査 | | | | |

## 1 はじめに

階層グラフ書換え言語 LMNtal は，その記述力と書換え系という特徴からモデル検査への応用が期待できる．また実用言語としての機能を備えた処理系が存在するため，LMNtal によるモデル検査器が実現すれば，開発とモデル検査を同一言語で行うことができる．

そこで LMNtal 実行時処理系 SLIM 上に LMNtal をモデル記述言語とする LTL モデル検査器 LMNtalMC を実装した．これにより階層グラフで記述される状態遷移システムについて，階層グラフに対するマッチング条件であるルール左辺で記述できるような性質を検証できるようになった．

本発表では LMNtalMC および LMNtalMC を用いたモデル検査例を紹介する．

## 2 LMNtalMC 概要

モデル検査とは，状態遷移系でモデル化されたシステムが，時相論理式で記述された性質を仕様として満たすかどうかを探索により検証する手法のことである．仕様の記述を時相論理式で行うことにより，状態遷移に関する性質を検査することができる．

LMNtalMC では時相論理として LTL（線形時相論理）を採用した．これは複数ある実行の可能性から処理系が 1 つを選択するという LMNtal プログラムの特性により，全ての実行経路に関してある性質が成り立つことを検証したい場合が多いためである．LTL によるモデル検査は，システムを表す Büchi オートマトン（**システムオートマトン**）と仕様の否定を表す Büchi オートマトン（**性質オートマトン**）との同期積（**同期積オートマトン**）が受理実行を持つかどうかの探索問題に帰着される．Büchi オートマトンの受理実行とは最終状態を無限にしばしば訪れる実行のことである．

### 2.1 LMNtal によるモデル検査

LMNtalMC では，LMNtal プログラムによって記述された状態遷移システムに対して，階層グラフのマッチング条件を指定する部分であるルール左辺で記述できるような性質に関して検証を行う．

例として図 1 はリストの連結を非決定的に行うプログラムについて，最終的に整列済みとなることを検証するための LMNtal プログラムであり，これが LMNtalMC に対する入力となる．2 行目から 3 行目は検証対象となるシステムを LMNtal で記述したものである．システムの状態遷移を表すルールを**システムルール**と呼び，初期プロセスとともに**システム膜**内に記述される．6 行目から 10 行目は，仕様として与えられた LTL 式と等価なオートマトンを表現するルールであり，これを**性質ルール**と呼ぶ．性質ルールは膜階層最上位に記述され，性質ルール左辺で表される性質が成り立つかどうかについてシステム膜内を監視する役割を持つ．

```
% システム膜
init{ l=([3,2,5,1]). //初期プロセス
sort@@ R=[X,Y|T] :- X>Y | R=[Y,X|T]. //システムルール
}.
% 性質ルール
ltl1@@ init{$p,@p} :- init{$p,@p}.
ltl2@@ init{R=[X,Y|T],$p[R,T],@p} :- X>Y |
       accept{R=[X,Y|T],$p[R,T],@p}.
ltl3@@ accept{R=[X,Y|T],$p[R,T],@p} :- X>Y |
       accept{R=[X,Y|T],$p[R,T],@p}.
```

図 1: LMNtalMC に対する入力プログラム（リストの連結）

この LMNtal プログラムを LMNtalMC が 3 節で述べる方法で処理することによりモデル検査が行われる．

### 2.2 オートマトンの LMNtal による表現

前述したように LTL モデル検査はオートマトンベースで行うため，本節では各オートマトンと LMNtal との対応付けを定義する．

#### 2.2.1 システムオートマトン

システムオートマトン $A$ を，初期状態 $A.S_0$ を初期プロセス，状態集合 $A.S$ を初期プロセスからルールによって到達可能であるプロセス，そして遷移ラベル集合 $A.L$ をシステムルールとして定義する．その上で状態遷移 $A.T$ を，遷移ラベルが表すルールの実行によるプロセス書換えとして解釈する．これにより $A$ はシステムの状態遷移グラフとして解釈することができる．

#### 2.2.2 性質オートマトン

図 2 は LTL 式 <>[]$p$ と等価な性質オートマトン $S$ である．このような LTL 式からそれと等価なオートマトンへの変換は，例えば Ltl2Ba などのツールによって行うことができる．

原子命題 $p$ が「隣り合う要素が降順であるリスト構造が存在する」という意味であるとき，上記 LTL 式は仕様の否定である「いずれ常に隣り合う要素が降順であるリスト構造が存在するようになる」という意味になる．ここで原子命題 $p$ の意味定義に，階層グラフに対するマッチング条件であるルール左辺の記法を用いると，$p$ の意味定義は図 3 のように記述できる．そし

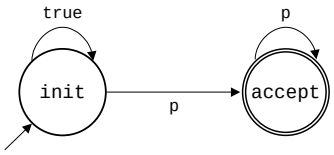


図 2: 性質オートマトン

```
ヘッド: R=[X,Y|T],$p[R,T],@p
ガード: X>Y
```

図 3: ヘッドとガードの記法による原子命題 $p$ の意味定義

て性質オートマトン $S$ の遷移ラベル集合 $S.L$ を図 3 で示すような記述を左辺として持つルールとして定義することで，図 2 のオートマトンを表現する図 1 の 3 つの性質ルールが得られる．

性質ルールの生成は，Ltl2Ba の出力と，ユーザが与える図 3 のような LMNtal の記法による原子命題の意味定義を合成することで行う．

#### 2.2.3 同期積オートマトン

同期積オートマトン $B = A \otimes S$ の遷移ラベル集合 $B.L$ は $A.L \times S.L$ として定義される．すなわち遷移ラベル $l \in B.L$ は性質ルールとシステムルールの順序対となる．そしてルールの順序対の意味論を，2 つのルールをその順に実行することであると定義することで，状態遷移 $t \in B.T$ を，性質ルールとシステムルールをこの順に実行することによるプロセス書換えとして解釈することができる．初期プロセスから性質ルールとシステムルールの順序対を網羅的に実行することで同期積オートマトン，すなわち探索対象となる状態遷移グラフが得られる．

# 3 LMNtalMC 実装

図 1 で示すようなシステム膜と性質ルールからなる LMNtal プログラムを入力として，LTL モデル検査を行って反例を返す機能を，LMNtal 実行時処理系 SLIM に組み込む形で実装した．LMNtalMC は主に次の 2 つの部分で構成される．

## 3.1 探索アルゴリズム

受理実行探索には nested DFS を用いる．nested DFS とは受理実行を発見するためのアルゴリズムであり，受理状態を探索する第一 DFS と，その受理状態を含むループが存在するかを探索する第二 DFS から構成されるアルゴリズムである．

## 3.2 状態展開

状態遷移グラフのある状態において，適用可能な性質ルールとシステムルールの組合せがある場合は，それらをその順に適用した結果を新たな状態とする．1 つのシステムルールについて複数のマッチング候補がある場合は，その全ての候補についてシステムルールを実行して新たな状態を生成する．

状態展開をする際には，新く生成する状態が過去に出現したかどうかを判定する必要がある．この履歴管理をハッシュテーブルによって行うために，階層グラフ構造のハッシュ関数および同型判定関数を実装した．

なお状態展開は探索と同期して行われる (on-the-fly)．これにより反例がある場合は，状態遷移グラフ全体が構成される前にそれを見つけることが可能となる場合がある．

# 4 モデル検査例と関連研究

## 4.1 MSR

多重集合書換えに基づくセキュリティプロトコル記述言語 MSR で記述された，簡易版 Needham-Schroeder プロトコルと攻撃者モデルを LMNtal にエンコーディングしてモデル検査を行った．MSR では右辺の ∃ によってプロトコルにおけるノンス生成などを表現しているが，LMNtal によるエンコーディングでは，ノンス生成を膜の新規作成によって，またその参照を膜への入射リンクによって表現している．

## 4.2 SPIN

SPIN は最もよく使われているモデル検査器の一つであり，高速かつ軽量である代わりにモデル記述言語の記述力が犠牲となっている．本研究では分散プロトコルなど SPIN が得意とするものに加えて，非決定的な計算モデル（4.3 節）など SPIN では記述が困難なものについてモデリングおよびモデル検査を行うことで，SPIN と比較して LMNtalMC の記述力が強いことを確認した．

## 4.3 $\lambda$ 計算

過去に $\lambda$ 計算の階層グラフ書換えへのエンコーディング手法が研究された．この研究ではエンコーディングされた $\lambda$ 式の合流性が示されているが，たとえば Church 数のべき乗などを計算すると，グラフ構造としては異なるが，$\lambda$ 式としては同じ意味であるような複数通りの結果が得られることが確認されている．非決定的実行（プログラムの全実行経路を出力する LMNtalMC の実行モード）により，Church 数 $2^2$ の結果として以下のような 2 通りのグラフ構造が存在することを確認した．

```
> ./slim --nd_result lambda_flat.il
execution result:
0(    450858):  go. 'next_color'(1). r(apply(two,two)). 'co
lor_count'(0,0). 'sub_count'(0,0). @1
...
34(  18768491): go. 'next_color'(4). r(lambda(cp(cp(L0,L1,0)
,cp(L2,L3,0),0),lambda(L4,apply(L0,apply(L2,apply(L1,apply(L
3,L4)))))). 'color_count'(0,3). 'sub_count'(0,0). @1

execution result:
...
43(  18761129): go. 'next_color'(5). r(lambda(cp(cp(L0,L1,0)
,cp(L2,L3,0),0),lambda(L4,apply(L0,apply(L1,apply(L2,apply(L
3,L4)))))). 'color_count'(0,3). 'sub_count'(0,0). @1
```

# 5 まとめ

LMNtal 実行時処理系 SLIM に LTL モデル検査器 LMNtalMC を実装した．LMNtalMC は階層グラフで表現されたシステムを，その構造や制約に関する性質について検証する．LMNtalMC では記述力の強い階層グラフ書換えをシステムや性質の記述に用いており，様々な応用例があることを確認した．