階層グラフ書換え言語LMNtalによるモデル検査

岡部 亮 †　　上田 和紀 ‡
† 早稲田大学理工学研究科　　‡ 早稲田大学理工学術院

1 はじめに

階層グラフ書換えに基づく言語モデルLMNtal[1]は，リンクによる接続構造と膜による階層構造に由来する表現力，そしてルールと呼ばれる階層グラフ書換え規則の非決定性（書換え対象選択における非決定性とルール選択における非決定性）を特徴としており，これまでにこれらの特徴を生かして，各種計算モデルのエンコードや，分散プロトコルのモデリングなどが行われてきた．モデル検査の分野ではモデル記述言語の表現力が重要な問題の一つとして挙げられており，このような特徴を持つLMNtalをモデル記述言語としたモデル検査器は有用であると考えた．

本発表ではLMNtal実行時処理系SLIM上に実装したモデル検査器LMNtalMCについて述べる．またLMNtalによるモデル検査例を紹介する．

2 LMNtalMC概要

LMNtalMCでは時相論理としてLTLを採用した．LTLによるモデル検査[2]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）との同期積が受理実行を持つかどうかの探索問題に帰着される．

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．

図1はリストの連結と整列を非同期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ソースコードである．1行目から4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に対応するルールをシステムルールと呼び，初期状態とともにシステム膜内に記述される．5行目から9行目は，システムの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する．

これらのルールを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．

Model Checking by Hierarchical Graph Rewriting Language LMNtal
†Ryo OKABE ‡Kazunori UEDA
†Graduate School of Science and Engineering, Waseda University
‡Faculty of Science and Engineering, Waseda University

```
init{ l=append([3,2,5],[1,4]).
a@@ R=append([H|T],L) :- R=[H|append(T,L)].
a@@ R=append([],L) :- R=L.
sort@@ R=[X,Y|T] :- X>Y | R=[Y,X|T]. }.
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によるモデル検査例

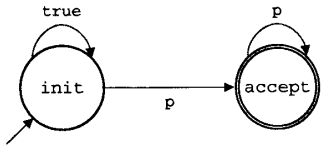

図2: 性質オートマトン

2.2 オートマトンのLMNtalによる表現

オートマトンにおける遷移ラベルをルールと定義すると，その状態遷移は遷移ラベルが表すルールの実行と解釈できる．これによりオートマトンをルールで表現することができる．システムルールを遷移ラベルとするオートマトンをシステムオートマトンとする．また性質ルールを遷移ラベルとするオートマトンを性質オートマトンとする．

図2はLTL式<>[]pと等価な性質オートマトンである．ここでpが「隣り合う要素が降順であるリスト構造が存在する」という命題であるとき，グラフ書換えのマッチング条件指定部であるルール左辺（構造を指定するヘッドと制約を指定するガードからなる）を用いると，命題pは図3のように表現できる．そして「遷移ラベルが表す命題を左辺として満たすときにシステム膜名を遷移前の状態ラベルから遷移後の状態ラベルに書き換えるルール」として状態遷移を表現することで，図2から図1の3つの性質ルールが得られる．

性質ルールの生成は，LTL式をPromela[3]で表現されたオートマトンに変換するツールの出力と，ユーザが与える図3のような命題定義とを合成することで行う．

2.2.1 同期積オートマトン

以上のように各オートマトンとルールとを対応付けると，同期積オートマトンの遷移ラベルは性質ルール

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

図3: ヘッドとガードによる命題表現

とシステムルールの直積となり，その状態遷移は遷移ラベルで表現された直積ルールを実行することと解釈できる．ルールの直積とは，与えられた2つのルールについて左辺の論理積をとり，右辺を連結したルールのことである．ただし性質ルールとシステムルールの直積をとる場合は，前者がシステム膜の膜名のみを書き換え，後者がシステム膜の膜内のみを書き換えるため，これは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等しい．性質ルールとシステムルールの直積を網羅的に実行することで同期積オートマトン，つまり探索対象となる状態遷移グラフが得られる．

2.3 決定的ルール

非決定性はLMNtalの特徴の一つであるが，それは同時に状態数の爆発的な増加を招くことがある．そこでLMNtalMCでは不必要な非決定性を排除するため決定的ルールを導入した．決定的ルールはルール名が`det_`で始まるシステムルールである．システム膜内に適用可能な決定的ルールがある場合は，処理系依存の順序で，全ての決定的ルールが適用できなくなるまで適用した結果を新たな状態とする．

決定的ルールの実行は，意味的にはシステムオートマトンにおける一回の状態遷移と考える．つまり全ての決定的ルールが適用できなくなるまで適用したときの，その決定的ルールの実行列を一つのシステムルールとして考える．

3 LMNtalMC実装

3.1 探索アルゴリズム

Büchi オートマトンの受理実行探索アルゴリズムはnested DFS[3]を用いる．

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複数の書換え候補がある場合は，その全ての場合の結果について新たな状態を生成する．ただし展開される状態が適用可能な決定的ルールを持つ場合は，性質ルール適用後に決定的ルールが適用できなくなるまで適用した結果を新たな状態とする．

状態展開の際には生成された状態が過去に出現したかどうかを判定する必要がある．LMNtalMCでは状態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．これにより状態遷移グラフの全体が構成される前に反例を見つけることが可能となる．

4 適用例

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であるMSR[4]をLMNtalにエンコードした．MSRでは右辺のヨによってプロトコルにおけるノンス（乱数）生成などを表現する．MSRのLMNtalによるエンコーディングにおいては，ノンス生成を膜の新規作成によって，またノンスの参照を膜への入射リンクによって表現した．上記論文で紹介されているNeedham-Schroeder公開鍵プロトコルとDolev Yao攻撃者モデルをLMNtalにエンコーディングして，LMNtalMCを用いてモデル検査を行った．

4.2 λ計算

[5]においてλ計算の階層グラフ書換えへのエンコーディング手法が紹介されている．この論文ではエンコーディングされたλ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，λ式の複製を表すアトムであるcpの交換則や結合則によってλ式としては同一であるような複数通りの結果が得られる．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2通りのグラフ構造が存在することを確認した．

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．これにより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ことが可能となった．LMNtalMCでは表現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，様々な応用例が考えられる．

今後の課題としては，実装およびアルゴリズムの最適化や，LMNtalの特徴であるプログラム可視化をモデル検査に取り入れることなどが考えられる．

参考文献

[1] 乾敦行, 工藤晋太郎, 原耕司, 水野謙, 加藤紀夫, 上田和紀. 階層グラフ書換えモデルに基づく統合プログラミング言語LMNtal. コンピュータソフトウェア, Vol. 25, No. 1, pp. 124–150, 2008.

[2] E. M. Clarke. *Model Checking*. The Mit Press, 2000.

[3] G. J. Holzmann. *The Spin Model Checker*. Addison-Wesley, 2003.

[4] I. Cervesato, N.A. Durgin, P.D. Lincoln, J.C. Mitchell, and A. Scedrov. A Meta-notation for Protocol Analysis. In *12th Computer Security Foundations Workshop — CSFW-12*, pp. 55–69, 1999.

[5] 上田和紀. 純粋λ計算の階層グラフ書換えへのエンコーディング. 第9回プログラミングおよびプログラミング言語ワークショップ論文集, 2007.