6J-07

階層グラフ書換え言語 LMNtal の表現力向上に向けたパターンマッチングモデルとその実現

齋藤 諒人 †
早稲田大学 基幹理工学部 †

上田 和紀 ‡
早稲田大学 理工学術院情報理工学科 ‡

1 はじめに

LMNtal は接続構造と階層構造を表現することのできるグラフ書換え言語であり [4]，グラフと書換え規則を簡潔に記述できる言語である．書換えはパターンマッチングによって取得した部分グラフを規則に従って変換する操作である．現状の LMNtal では全称量化や否定のパターンマッチを行うことができない．本論文では，LMNtal におけるパターンマッチの表現力を向上するために導入すべきパターンマッチングモデルと，その実装について論じる．

2 LMNtal

2.1 言語モデル

LMNtal はリンク（無向辺）でグラフを構成する言語である．また，膜 {} を用いることで階層構造を表現できる．LMNtal の構文は以下の通りであり，P をプロセス，T をプロセステンプレートと呼ぶ．

$$P ::= 0 \mid p(X1, ..., Xm) \mid P, P \mid \{P\} \mid T\text{:-}T$$
$$T ::= 0 \mid p(X1, ..., Xm) \mid T, T \mid \{T\} \mid T\text{:-}T \mid @p \mid \$p$$

LMNtal の書換え規則はルールと呼ばれる．言語モデルでは T:-T と定義され，左辺を Head，右辺を Body と呼ぶ．Head でのマッチングは全て存在量化子ヨに相当する．また，LMNtal は各ルールのマッチングが非決定的であり，グラフがいずれのルールによっても簡約不可能になるまで適用が繰り返される．

LMNtal の処理系では拡張構文として，マッチングの制約条件を Guard に記述することができる [5]．

$$Head \text{ :- } Guard \mid Body$$

2.2 ガード否定条件

LMNtal では，膜に対する否定条件を記述できる．

$$\{\$p\} \text{ :- } \backslash+(\$p = (a(X), \$pp)) \mid \$p.$$

これは Head 部でマッチングした膜が 1 価の（リンクを 1 つ持つ）a アトムを持たない場合に，その膜からプロセスを取り出すルールである．\+ が否定条件を意味し，括弧内は膜 {$p} のプロセス文脈 $p（膜の中に存在する全てのプロセスを意味する）が $a(X)$ と $pp に分割できることを意味する．この構文は膜の存在が前提となり，世界的ルート膜（プロセス全体を指す仮想の膜）への適用や膜同士の関係の言及，否定のネストはできない．

3 関連研究

パターンマッチを行う言語モデルとして，GROOVE と VIATRA を取り上げる．

3.1 GROOVE

GROOVE は動的システムを視覚的にモデリングするツールセットであり，プログラムの仕様と挙動を自動で検証するモデル検査を実行できる [2]．GROOVE における初期グラフ (host graph) は名前付き node と名前付き有向 edge で構成される．書換え規則 (rule) は host graph と同様に記述され，それぞれの node と edge は rule 適用の存在条件である Readers，rule 適用で削除される Erasers，rule 適用によって生成される Creators，rule 適用の存在否定条件となる Embargoes，Embargoes と Creators の両方を意味する Conditional creators のいずれかに分類される．

Rule のグラフは複数作成することができ，それぞれに名前が付く．各 rule の適用順序や反復の制御，条件分岐はテキストで記述する．また，rule では量化子を扱うことができ，通常 node が量化子 node を edge で参照することによって全称量化，存在量化ができる．量化子 node が別の量化子 node を参照することで入れ子にできる．

3.2 VIATRA

VIATRA はモデル駆動開発を支援するモデル変換 (Model transformation) フレームワークであり [1]，グラフ書換え言語と厳密には分類が異なるが，接続構造や階層構造を操作する点において類似している．VIATRA はグラフ書換え言語のグラフに相当するモデルおよびその書換えによって構成されるモデルが，ドメインの制約下で構成される特徴を持つ．VIATRA は統合開発環境 Eclipse 上で動作し，複数言語およびフレームワークで構成される．モデリングには Eclipse Modeling Framework (EMF)，パターンマッチングには VIATRA Query Language (VQL)，モデル変換には

An enhanced pattern matching model and its implementation of LMNtal, a hierarchical graph rewriting language

† Ryoto Saito, Dept. of Computer Science and Enginerring, Waseda University
‡ Kazunori Ueda, Dept. of Information and Computer Science, Waseda University

```
1  not [[
2          findatom [1, 0, 'a'_1]
3          deref [2, 1, 0, 0]
4          not [[
5                  func [2, 'b'_1]
6                  proceed [] ]]
7          proceed [] ]]
```

図 1　二重否定による全称量化中間命令列の例

Java の DSL フレームワーク Xtend による独自 DSL が用いられる．実際にはいずれも Java に変換され，Java で実行されている．

VQL は要素の型（クラス），関係，属性（フィールド）についてのマッチングを行う．それぞれのパターンには名前がつき，予約語 find によって他パターンを呼び出すことができる．パターンのステートメントを or で接続することで，複数条件からいずれかのマッチングを満たす要素を取得することができる．また，予約語 neg を付加することで呼び出したパターンを満たさない要素にマッチングする．

4　LMNtal 構文拡張

4.1　全称量化

全称量化 $\forall x P(x)$ は $\neg\exists x \neg P(x)$ と同値である．否定の構文を定義し，二重否定を記述することによる全称量化マッチングの実現を検討する．

4.2　否定の中間命令列表現

LMNtal の処理系はコンパイラと実行時処理系に分かれる．コンパイラは LMNtal ソースを実行時処理系が解釈する中間命令列に変換し，実行時処理系は中間命令列で記述されたルールの実行制御を行う．

マッチングを行う中間命令の例として *findatom* を挙げる．*findatom* は指定されたファンクタ（アトム名と価数）である任意のアトムを読み込む．後続の命令が失敗すると同命令までバックトラックが発生する．この命令は存在量化子 ∃ に相当する．中間命令 *not*[*instructionlist*] は中間命令列 *instructionlist* を実行し，中間命令 *proceed* を実行すると失敗する．この命令はすでに予約されており，ガード否定条件の実装に使用されている．否定構文を定義する際，その中間命令列表現は構文内プロセスのマッチング命令列と命令 *proceed* を *not* 内に記述することで実現する．

処理系はネストされた *not* の解釈が可能である．*not* を二重にすることで，全称量化マッチングが可能であることが確認できた．全ての 1 価 a アトムが 1 価 b アトムに接続されることを示す中間命令列を図 1 に示す．

5　Petri Net の発火可能マッチング

グラフによる数理モデルとして Petri Net が挙げられる．これは離散分散システムを有向 2 部グラフで表現するものである [3]．Petri Net を LMNtal グラフにエンコードする場合，発火可能である（接続される任意の place に token が存在する）transition にマッチするパターンを記述することが困難である．否定の構文をネストした全称量化マッチングを導入すれば，このようなパターンを記述することができる．発火可能な transition アトムを fireable アトムに変化させるプログラムを図 2 に示す．ここで，中間命令の *not*[*instructionlist*] に相当する LMNtal 文法を，プロセス P を用いて一時的に $![P]$ としている．また，このプログラムは LMNtal から拡張された階層ハイパーグラフ書換え言語 HyperLMNtal で記述している．詳細は [6] を参照されたい．また，この時の Petri Net モデルを図 3 に示す．

```
1  transition($t, $t2)
2    | hlink($t), hlink($p),
3      ![place($p), arc($p, $t), ![token($p)]]
4   :- fireable($t, $t2).
```

図 2　HyperLMNtal を用いた Petri Net の発火マッチングルール

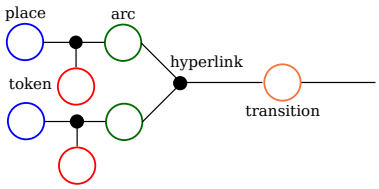

図 3　HyperLMNtal による Petri Net モデルの例

6　まとめ

LMNtal 処理系で使用されている中間命令を用いてマッチングの否定および全称量化の表現が可能であることを確認した．LMNtal 拡張構文の定義およびコンパイラの実装が今後の課題である．

参考文献

[1] Andras, B. and Daniel, V.: Advanced Model Transformation Language Constructs in the VIATRA2 Framework, *ACM Symposium on Applied Computing - Model Transformation Track*, ACM Press (2006).

[2] Arend, R., Iovka, B., Harmen, K. and Tom, S.: *User Manual for the GROOVE Tool Set* (2014).

[3] Desel, J. and Reisig, W.: *Place/transition Petri Nets*, pp. 122–173, Springer Berlin Heidelberg (1998).

[4] Kazunori, U.: LMNtal as a hierarchical logic programming language, *Theoretical Computer Science*, Vol. 410, No. 46, pp. 4784–4800 (2009).

[5] Kei, M., Shintaro, K., Ken, S., Ken, M., Norio, K. and Kazunori, U.: Implementation of the Hierarchical Graph Rewriting Language LMNtal, *Computer Software*, Vol. 25, No. 2, pp. 47–77 (2008).

[6] Seiji, O., Manabu, M. and Kazunori, U.: HyperLMNtal: an extension of a hierarchical graph rewriting language model, *The 25th Annual Conference of the Japanese Society for Artificial Intelligence* (2011).