LA-003

# LMNtal処理系におけるグラフ構造の操作機能の設計と実装

## Design and Implementation of Operation Constructs of Graph Structures in the LMNtal System

工藤 晋太郎* 加藤 紀夫†‡ 上田 和紀†
Shintaro Kudo Norio Kato Kazunori Ueda

## 1 はじめに

一般にプログラミング言語においてデータとは，数値や記号のようなアトミックなものと，内部構造を持つものとに分けられる．内部構造を持つデータは，その表現によく図を用いることからもわかるように，その表現と操作には，グラフ書き換えに基づく言語が適していると期待される．

階層的グラフ構造の書き換えに基づく言語モデル LMNtal[3] は，木構造だけでなく環構造も含む一般のグラフ構造も許しており，また膜による多重集合の表現も可能なため，柔軟なデータ表現が可能である．

LMNtal では，書き換え規則に従ってグラフ構造を書き換えることで計算を実現する．この書き換え規則において，特定の構造に対する操作を表現する仕組みとしてプロセス文脈と呼ばれる概念があり，それによって複数通りの書き換えを一本の規則で表現することができる．

本研究では，従来の LMNtal 言語を拡張してこのプロセス文脈にデータ型というグラフ構造に関する型を導入した．その設計においては，直感的な表現と操作を念頭に置いた．それにより，グラフ構造によるデータの表現とその比較・複製・破棄といった操作を可能にした．

また，現在 LMNtal には Java で実装された処理系がある.§ 本研究ではこの LMNtal 処理系を拡張し，設計した機能を実装した．

## 2 LMNtal の概要

### 2.1 アトムとリンク，膜

図 1 は LMNtal の基本要素である**アトム**とその引数に持つ**リンク**，および**膜**を図示したものである．丸がアトム，点線が膜，直線がリンク，矢印はアトムの第一引数の位置と引数の並ぶ方向を示す．またそのテキスト表現を下に示す．アトムは英小文字か数字から始まる名前とそれに続く括弧内に並ぶ引数 (リンクを持つ) で記述する．膜は { と } で記述する．リンクは英大文字で始まる名前で表現される

図 1 のテキスト表現

```
a,b(X),c(X,Y),{d(Y),e}.
```

### 2.2 プロセス

LMNtal におけるプロセスの定義を図 2 に示す．

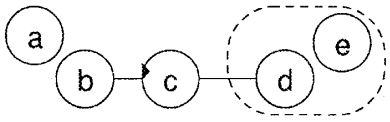


図 1 アトムとリンク，膜

| | |
|---|---|
| $P$ ::= | (プロセス) |
| 0 | (空) |
| \| $p(X1,...,Xm)$ | (アトム) |
| \| $P,P$ | (分子) |
| \| $\{P\}$ | (セル) |
| \| ルール | |

図 2 プロセス

**セル**とは，膜及びその内側のプロセスを指す．セルは入れ子になることができるが，セル $P$ において，入れ子になったセルに含まれない部分をその $P$ の**最外部**と呼ぶ．また，どのルールにも含まれない部分を**ルール外**という．

- **(リンク条件) プロセスのルール外には，同じリンクが 2 回を越えて出現してはならない．**

プロセス $P$ に 1 回出現するリンクを $P$ の**自由リンク**と呼び，$P$ に 2 回出現するリンクを $P$ の**局所リンク**と呼ぶ．

### 2.3 ルール

プロセスの書き換え規則を，**ルール**という．ルールは，ヘッド，ガード (省略可)，ボディの三部からなる．ルール適用とは，ルールのヘッドにマッチしたプロセスがボディに出現するプロセスに書き換えられることである．LMNtal における計算は，ルール適用によって進む．ガードにはルールが適用される条件を記述する．

ルールの構文

ヘッド :- ガード | ボディ

簡単なルールの例を示す．

ルールの例

```
{a(X),$p[X]} :- b(Y),$p[Y].
```

`$p[X]` と `$p[Y]` は**プロセス文脈**と呼ばれるものである．ヘッドに出現するプロセス文脈は膜内にのみ出現することができ，「膜内の残りすべて」にマッチする．つまり，不特定のグラフ構

* 早稲田大学理工学研究科情報・ネットワーク専攻
† 早稲田大学理工学部コンピュータ・ネットワーク工学科
‡ 現所属は産業技術総合研究所システム検証研究センター
§ http://www.ueda.info.waseda.ac.jp/lmntal/

造を扱うための概念である．プロセス文脈は名前を持ち(ここでは$p)，0個以上の引数を持つ．ここでは1引数である．ルールのヘッドでは，各々のプロセス文脈はちょうど一回出現し，各膜に高々一つである．また，ルールのヘッドの最外部には出現しない．

つまりこのルールは，aという名の1引数アトムのあるセルに反応し，セルをアトムaごと取り除いてaに繋がっていたリンクをbという名の1引数のアトム(を新しく作って)そこに繋ぎ直す，という操作を表現している．

また，アトム $p(A_1, A_2, ..., A_{n-1}, A_n)$ の $A_n$ とアトム $q(B_1, B_2, ..., B_{k-1}, B_k, B_{k+1}, ..., B_m)$ の $B_k$ が繋がっている(同じリンクである)とき，$q(B_1, B_2, ..., B_{k-1}, p(A_1, A_2, ..., A_{n-1}), B_{k+1}, ..., B_m)$ と略記することができる．つまり，次の二つは同じプロセスを表している．

```
a(X,Y), b(Z,X), c(Y), d(Z).
```

```
a(b(d),c).
```

また，"=" という名を持つ2引数のアトムは**コネクタ**と呼ばれ，その二つの引数に持つリンクを短絡させる目的で使われる．例えば，次の三つは同じプロセスを表す．

```
a(A),A=B,B=C,C=D,D=E,e(E).
```

```
a=e.
```

```
a(e).
```

## 3 型付きプロセス文脈

プロセス文脈は，1本のルールで複数通りの書き換えを記述する仕組みである．

本研究において，データの比較・複製・削除，および並行動作時のデータの待ち合わせのための仕組みとして**型付きプロセス文脈**を導入する．型付きプロセス文脈はその名の通り，従来のプロセス文脈の拡張的な概念である．型付きプロセス文脈は膜外に出現することができ，特定の構造にのみマッチする．この特定の構造を指定するために，**データ型**という概念を導入する．

### 3.1 データ型

例えばデータ型の最も単純な例として **int 型**がある．**int 型のプロセス**とは，「整数を名前として持つ，1引数の単一のアトムからなるプロセス」である．つまり具体的には，

```
1(X).
15(Y).
-3(N).
```

のようなプロセスである．

型付きプロセス文脈とは，特定のデータ型のプロセスにのみマッチするプロセス文脈である．型付きでない従来のプロセス文脈と違い，膜内に複数出現したり膜の外に出現してもよい．

例えば int 型のプロセスにのみマッチする型付きプロセス文脈を使ったルールは，次のように記述する．

```
a(A),$n[A]  :- int($n)  |b(B),$n[B].
```

このように，型付きプロセス文脈に対するデータ型の指定はガード部に記述する．このルールでは，aという名前の1引数のアトムと，その第一引数に繋がった int 型のプロセスにマッチする．

この型付きプロセス文脈を，次のように略記することができる．通常英大文字から始まる名前が示すのはリンクであるが，この場合はリンクに繋がる型付きプロセス文脈自体を表現している．

```
a(A)  :- int(A)  |b(A).
```

従来のプロセス文脈は，ルールのヘッド部において出現する膜の残りすべてにマッチする．一方，型付きプロセス文脈は指定されたデータ型のプロセスにのみマッチする．

その他のデータ型の例として，unary 型がある．unary 型は，**単一の1引数アトムからなるプロセス**である．つまり，次に示すようなプロセスはすべて unary 型である．

```
a(X).
data(Y).
longlongname(L).
15(N).
```

int 型のプロセスは unary 型のプロセスでもある．

現処理系では他にも，float 型，string 型というデータ型が用意されている．また，構造的なデータを扱うためのデータ型として，ground 型がある．ground 型については，4章で解説する．

### 3.2 ガード制約

型付きプロセス文脈に関するガードの機能として，その構造を限定する型検査だけでなく，他にもいくつかの検査ができる．

例えば，型指定以外のガード制約として "=" がある．型付きプロセス文脈同士が等しい(「等しい」の定義はデータ型により異なるが，int 型ならば同じ数値の名前であること，unary 型ならば同名であること)場合にのみマッチさせるようにルールを制限することができる．例えば

```
man(tommy,male),
man(jenny,female),
man(danny,male),
getout(male).

getout(W), man(Name, Sex)  :-
unary(W),unary(Sex),W=Sex  |getout(W).
```

というプログラムを実行すると，

```
getout(male),man(jenny,female).
```

という結果が得られる．

現処理系では，各データ型についてそれぞれに対応した比較が組み込みのルールとして実装されており，例えば次のようなルールが表記できる．

```
number(N), number(M) :- N>M | number(N).
```

“>” 制約を記述した場合，int 型制約を省略できる．

このルールは繰り返し適用されることにより最大値のアトムを引数に持つ number という名のアトムのみを残しそれ以外を消す．

### 3.3 ガード制約の実装

現在の LMNtal 処理系は，ルールが LMNtal コンパイラによって中間命令列に変換され，実行時処理系によって中間命令列が解釈実行されることで動作している．

コンパイラは，ルールのヘッド部，ガード部，ボディ部のそれぞれに対応する命令列を発行する．

ガード部に対応して発行されるガード命令は，ヘッド命令によって特定されたプロセスに対してガードで指定された条件を満たすかどうかの検査をする命令である．

ヘッド命令とガード命令が実行時処理系により解釈実行され，マッチング成功となればボディ命令が実行される．ヘッド部に出現するプロセスにマッチしたプロセスは取り除かれ，ボディ部に出現するプロセスが構築される．マッチするプロセスが無ければボディ命令は実行されない．

ルールのコンパイルと解釈実行処理系については [1] を参照されたい．

型付きプロセス文脈に関するガード制約は，各制約に対応するガード命令によって実現される．

データ型を指定するガード制約に対しては，ヘッド部に出現する型付きプロセス文脈にマッチしたプロセスが各データ型のプロセスであるかどうかの検査命令が発行される．int 型であれば整数を名前に持つ単一のアトムであることが，unary 型であれば 1 引数の単一のアトムであることが，それぞれ検査される．

“=” 制約はデータ型ごとに定義が異なるため，各データ型ごとに対応する比較命令が発行される．int 型と unary 型であれば，同名であることを検査する．

その他，各種数値型についての大小比較も，それに対応したガード命令が定義されている．

## 4 構造型

記述経験から，3 章で述べた各データ型だけでは不足なことがわかった．unary 型を用いれば，1 引数の単一アトムについては比較 (等しいかどうかの検査) が可能である．しかし，より規模の大きい構造に対して比較を行いたいという欲求がある．そこで ground 型というデータ型を設計した．

### 4.1 ground 型

ground 型の定義を示す．

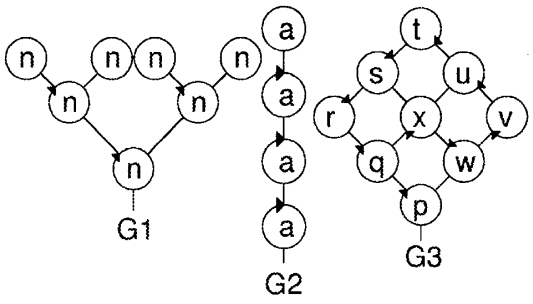


図 3 ground 型のプロセスの例

> ground 型のプロセスが持つ自由リンクは 1 つだけである．このリンクを根という．
> ground 型のプロセスは，膜をまたぐリンクを持たないアトムとそれらの持つリンクから成る．
> ground 型のプロセスを構成する全アトムは，根からリンクを辿って到達可能である．

具体例を図 3 に示す．また，これらのテキスト表現は次のようになる．

```
G1=n(n(n,n),n(n,n)).
G2=a(a(a(a))).
G3=p(q(r(s(t(u(v(w(x(Q,S,U),W)),U)),S)),Q),W).
```

直感的には，閉じたネットワークで出口が一つあるグラフ構造である．

設計上重要な点は，膜をまたぐリンクを含まないこと，つまり膜をその構造の内部に持つことも膜をまたいで存在することもない，ということである．

膜は多重集合を扱うための概念である．比較が目的である ground 型に膜を含むことにした場合，その目的は多重集合を比較することである．しかしながら，2 つの多重集合の比較は非決定性を含むので，言語に基本操作として導入するには大きすぎる．また，単に構造的なデータを比較することが目的ならば膜を用いる必要はなく，上に挙げた仕様で十分であると判断し，ground 型は膜を含まない仕様とした．

### 4.2 グラフ構造の検査・比較機能の設計

ground 型の検査・比較はガード命令により実行される．

すなわち，unary 型のプロセスを検査 (unary 型かどうかの検査)・比較する命令と同じように，ground 型のプロセスを検査 (ground 型かどうかの検査)・比較する命令を追加した．unary 型について発行されていたのと同じように，ground 型に対してはこれらの追加命令を発行するようにした．それにより従来のコンパイラの命令発行処理を最小限変更するのみで実装することができた．

検査・比較それぞれを行う命令は，根から再帰的に辿る方法で実装されている．すなわち，検査ならば構造に膜をまたぐ部分が含まれないことを，比較ならば同じ構造であることを検査

する．

ground 型のプロセスの特定は根のみで行われる．つまりどこからどこまでが ground 型のプロセスか，ground 型を構成するアトムがどれなのかという情報は持たず，根を対象として検査命令を実行する．これは，基底項プロセスの内部構造が分断されていないことを利用している．ただしこのとき，ヘッド部に出現する膜内の他のアトムと重複してマッチすることのないよう，ルール内の他のマッチング情報を参照する．

比較検査においても，分断されていない性質を利用し根から辿ることで効率の良い検査を実現している．

### 4.3 記述例

ground 型に関する型指定も，unary 型の場合と同様に記述できる．

```
g(F),g(G) :- ground(F),ground(G),F=G | ().
```

このルールは，等しい構造を第1引数に持つ g アトムを見つけ，それらの構造ともども削除する．ground 型の比較・破棄機能を使っている．

## 5 非線形プロセス文脈

### 5.1 型付きプロセス文脈の複製・破棄

プロセス文脈は，ルールのボディ部での出現が複数ならば複製を意味し，出現が一回ならば移動，出現が無ければ破棄を意味する．

```
cp(X),{$p[X]} :- done(Y,Z),{$p[Y]},{$p[Z]}.  % 複製
mv(X,Y),{$p[X]},{$q[Y]} :-
  done(Z,W),{$p[Z],$q[W]}.  % 移動
cn(X),{$p[X]} :- done(Y),{$p[Y]}.  % 存続
rm(X),{$p[X]} :- ().  % 破棄
```

ヘッドとボディで一回ずつ出現するものを**線形プロセス文脈**と呼び，ボディでの出現が0回ないし複数回であるプロセス文脈を**非線形プロセス文脈**と呼ぶ．

本研究では，非線形プロセス文脈を扱うための，プロセスの複製・破棄操作を設計，実装した．

実行時に，ヘッドでプロセス文脈にマッチしたプロセスは，まずボディでのプロセス文脈の出現位置にコピーされる．その後プロセス文脈にマッチしたプロセスは削除される．

本研究では，プロセス文脈を複製する命令と破棄する命令を追加した．ルールのボディのコンパイル時にそれらをコンパイラによって発行させ，また実行時処理系にそれらを解釈実行させるようにそれぞれを拡張した．

unary 型や各種数値型，文字列型など構造が単一のアトムの場合は，名前と引数の数を記録してボディ部で構築する．これは通常のアトム(ルールボディ部に表記されたアトム)が生成されるのと同じ処理であり，破棄も通常のアトムと同様である．コンパイラは通常のアトムと同様の命令を発行する．

一方，ground 型の場合はグラフ構造が不特定である．そこで ground 型の複製/破棄命令を定義し，その実行時処理系による解釈実行は，根から辿り再帰的に複製/破棄をすることで行う．すべての構造が根から辿れるという性質を利用している．

### 5.2 計算量

複製・破棄操作の計算量は，従来のプロセス文脈も型付きプロセス文脈もほぼ規模に比例する．また，型付きプロセス文脈の比較検査についても規模に比例した計算量で実現できている．これはすべての構造が根から辿れるためである．

### 5.3 用途

これまでの記述経験において，型付きでないプロセス文脈は多くの場合，プロセスを書き換える際に膜内の他のプロセスをそのまま存続させる目的で利用されていた．また，他の膜に膜内プロセスを移動するために利用することもあった．

しかし，型付きプロセス文脈は特にデータを扱うための機能であり，その複製や破棄といった操作も必要であると思われる．また，型付きでないプロセス文脈についても，膜でデータを表現することがあるため，その複製・破棄は便利な機能である．また，計算状態の複製・破棄操作も可能になり，プロセス文脈の複製・破棄操作が有用であることがわかった．

## 6 まとめ

本論文では，LMNtal においてグラフ構造の形を指定するためのデータ型を導入し，プロセス文脈の拡張として型付きプロセス文脈を導入した．特にデータ型の一つとして設計した ground 型により，特定構造の比較が可能になった．また，データ型を指定し比較を行うガード制約を実現するべく中間命令を追加し，処理系を拡張して追加した命令を解釈実行するようにした．そしてこれらの機能を使ったプログラム例を記述し，その有用性を確かめた．

本研究では膜による多重集合の比較機能は，その処理の大きさゆえに対象外とした．また，構造データの表現は多くの場合ルートから辿れる構造にすることで，実装した機能で代用可能な操作と思われるためである．その意味で，単なるデータの表現に膜を使うことは適切でないかもしれない．しかし，多重集合としてのデータ表現には膜は不可欠であり，その比較機能の必要性については議論の余地がある．LMNtal 言語におけるデータ表現のスタイルを包括的に検討する必要があるだろう．

## 参考文献

[1] 水野謙, 永田貴彦, 加藤紀夫, 上田和紀. LMNtal ルールコンパイラにおける内部命令の設計. 情報処理学会第 66 回全国大会, 5G-2, 2004

[2] 工藤晋太郎. LMNtal 処理系におけるグラフ構造の比較および複製・破棄機能の設計と実装. 早稲田大学卒業論文, 2004.

[3] 上田和紀, 加藤紀夫. 言語モデル LMNtal. コンピュータソフトウェア, Vol. 21, No. 2, pp. 44-60, 2004.