小規模制御系向けLMNtal処理系の設計と実装

Design and Implementation of LMNtal Runtime for Small-memory Control Systems

3603U147-0　矢島 伸吾　Shingo Yajima
指導：上田 和紀 教授

概要： 階層グラフ書き換え言語 LMNtal は大規模から極小まで様々な計算環境を統一的に扱えるスケーラブルなソフトウェア基盤を目指しており，分散など大規模への応用研究は進められているが，組み込み分野など小規模な計算環境と LMNtal の関係はまだほとんど考えられていない．
本研究では，汎用組み込み制御システム eyebot 上で動作する LMNtal 処理系の設計，実装を通して，メモリ消費量の少ない LMNtal データ構造管理，LMNtal 上における入出力制御やタイマの表現，一連のルール適用の優先実行，など LMNtal と組み込み制御分野の関係を考える上で重要となる問題について議論し，その一方式を提案する．
本研究の成果として，膜，プロセス文脈の一部機能を含む LMNtal プログラムによって周期タスクや制御処理を記述し，eyebot を制御，動作させることに成功した．

キーワード：LMNtal, グラフ書き換え言語, eyebot, 実行時システム, 小規模システム, 組み込みシステム
keyword：LMNtal, graph rewriting language, eyebot, runtime system, small-memory systems, embedded systems

1　研究の背景と目的

LMNtal は膜，アトム，リンクからなる階層グラフ構造をルールによって書き換えることで計算を行う言語モデルである．現在 Java による処理系が公開されており，膜で計算ノードを表し，ルールで通信を記述することによる分散処理の実行にも対応している．このように LMNtal の大規模計算への応用は研究されているが，LMNtal が真にスケーラブルな言語モデルであるためには，組み込み分野などの小規模な計算環境へも応用可能であることを言う必要がある．

組み込み機器の制御プログラムの特徴としては

- 利用可能なメモリが少ない
- 制御対象がある
- (優先度に基づくスケジューリングのために) 一連の処理 (タスク) を優先実行することができる

があげられるが，これらの問題と LMNtal の関係はわかっていない．本研究では，小規模な制御系上で動作する LMNtal 処理系 eLMNtal の設計，実装を通して，これらの問題の解決方法を模索，提案する．

小規模な制御系として，ロボット制御用組み込みシステム eyebot を用いた．eyebot には，

- 組み込み機器にしては大容量なメモリ (1MB)
- 様々な入出力インターフェース
- C++開発環境と豊富なライブラリ

が用意され，処理系など複雑なプログラムの開発に適している．

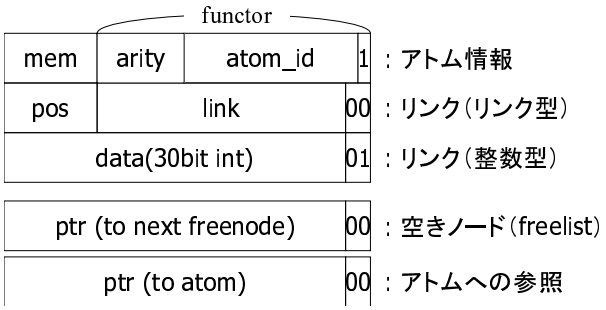

図 1: 風上に向かって走る車の動作例

2　省メモリデータ構造

Java 版処理系ではデータ構造は膜，アトム，リンクオブジェクトの相互参照により構築されており，これは空間量的には効率の悪い設計である．eLMNtal では

データ構造を図 1 のように表現することで、アトムとリンクを arity+1 word で表現することに成功した．また，pooled allocation(一定量のアトム領域の事前確

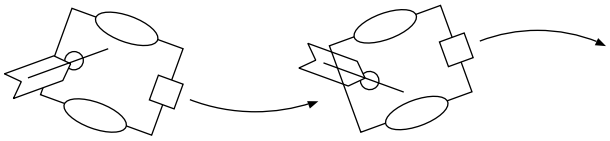

図 2: 風上に向かって走る車の動作例

保) によって，頻出するアトムの生成，消去時のメモリ割り当ての時間，空間効率を改善した．

3 制御の記述

eLMNtal では，並列タスクを扱うために,「膜 (タスク膜) を 1 個以上定義し，その中にタスクの制御内容を書く」という形式の LMNtal プログラムを書く．各タスク膜のルール適用を平等に行うことで擬似的に並列タスクを実現している．

LMNtal 上で制御を扱うために,「事前に定義されたアトム (システムアトム) が生成された時に特別な処理を行う」という方式をとった．例えば，

```
analog1(get)
```

というアトム構造を生成すれば，直後に解釈され，analog1(N) という構造に消去，再生成される (N は入力値)．また，周期タスクの実現に不可欠なタイマは，

```
analog1(timer(50))
```

というように書く．生成直後に消去，タイマ機構にデータ登録され，以降 50msec 毎に analog1(N) という構造が生成される．

get, timer などをコマンド型システムアトムと呼び，生成時に捕捉し，すぐに値取得などの処理を行う．analog1 などは生成型システムアトムと呼ばれ，タイマや get などにより生成され，後述のスタック機構に積まれて優先実行される．

応用例として,「風上に向かって走る車」を動作させるプログラム例をあげる．ロータリーエンコーダの回転角によって風上を認識している．(図 2)

```
{
quadleft_reset, quadleft(timer(3)), analog3(timer(3)).
quadleft(X) :- int(X), int(Y),
    Y = (((X + 1000100) mod 200) / 2) - 50 | wind(Y).
wind(Y) :- int(Y), int(VL), int(VR), VL=50+Y, VR=50-Y |
    leftmotor(VL), rightmotor(VR).
analog3(X) :- int(X), X > 500 |
    end, analog3(timerkill(3)), quadleft(timerkill(3)).
analog3(X) :- int(X), X =< 500 | .
end :- leftmotor(0), rightmotor(0).
}
```

	a(0) の表現にかかるメモリ	静的プログラムサイズ
eLMNtal	8 byte	54KB
Java 版	620 byte	300KB

表 1: 動的，静的メモリ使用量の比較

4 優先実行の仕組み

生成された analog1 などの生成型システムアトムは，タスクの持つスタックに積まれ，通常のルール適用に優先して実行される．各タスクは，ルール優先適用を行うアトムが積まれる実行スタックと，他タスクやタイマなどからのアトムの生成要求をうけとる生成待ちスタックの 2 種類のスタックを持つ．タスクは通常のルール適用前に，生成待ちスタックが空でなければ生成を行い実行アトムスタックに積み，それから実行アトムスタックが空になるまでルール適用を繰り返す．また，通常のルールは左辺解釈時であれば，優先度の高い処理のために中断終了させることが可能である．

以下にルール適用ループのアルゴリズムを記す．

```
do{
  for(各タスク){
    if(タイマ割り込み発生 ||
       スタックが空でない) break;
    タスクのルールを 1 つ適用試行
  }
  if(タイマ割り込み発生)
    アトム functor を生成待ちスタックに追加
  if(スタックが空でない){
    for(各タスク){
      生成待ちスタックに積まれたアトムを生成
      実行スタックに積まれたアトムの優先実行
    }
  }
}while(全てのタスクが stable になるまで);
```

この機構によって，タイマなどのイベント発生後ただちに生成されたアトムに対するルール適用を行うことができる．

5 まとめと今後の課題

LMNtal によって周期タスクや制御処理を記述し，eyebot を動作させることが可能な処理系 eLMNtal の設計，実装を行った．必要なメモリ容量は表 1 のように削減することができた．

今後は，システムアトムのユーザ定義による複雑な処理の優先実行や，より細かな優先度の指定，タスクのデッドラインの指定やスケジューリング機構の実装などを通して，LMNtal とリアルタイム性との関係について考えていきたい．