修 士 論 文 概 要 書

CD

2008 年 2 月提出　　学籍番号 3606U070 – 4

専攻名	情報・ネットワーク	氏 名	中野　敦	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	スケーラブルな可視化機能を備えた LMNtal 開発環境の設計と実装				

1　本研究の目的と背景

LMNtal は階層グラフの書き換えに基づく並列言語モデルである．簡潔な構文を持つ言語であるものの，規模が大きくなるにつれ処理の進行の把握が困難となる．一方で，少ない基本要素から階層グラフ構造をなしているプログラムであるため，可視化が容易に行える．これらの理由から， LMNtal プログラムの理解にはテキスト表現によるグラフ構造の提示よりも，可視化による視覚的な情報提示が有効であると考える．

現在 Java による実装が進んでいる LMNtal 処理系には，以前から可視化機能が組み込まれており，ある程度の可視化が可能となっている．しかしながら，この可視化機能は，大規模なグラフを扱うことを前提としておらず，規模の大きいグラフや，リンクが複雑なグラフなどではグラフが振動や発散により安定しないという問題があった．そこで，本研究では，階層グラフの可視化と同時に開発環境を備えたツール UNYO UNYO の設計と開発を行う．

2　言語モデル LMNtal

LMNtal は主に，基本の要素であるアトム，要素を繋げるリンク，要素の多重集合である膜，書き換え規則であるルールの 4 つの要素から成る．アトムは，LMNtal の基本要素であり，名前と 0 本以上のリンクを保持する．2 本のリンクを持つ a という名のアトムを a(X, Y) と表記し，リンクの順番が異なる a(X, Y) と a(Y,X) のようなものは別物として扱われる．また，リンクは要素を 1 対 1 で接続するものである．膜は，アトムや膜など要素の多重集合を保持する仕組みである．膜を要素として保持できるため，階層的なグラフ構造を実現する．ルールはグラフ構造のある特定部分を書き換える規則を表す．

3　UNYO UNYO の設計

3.1　システム概要

本システムは階層グラフ書き換え言語である LMNtal のスケーラブルな開発環境として設計されている．

本研究のスケーラビリティとは，より大規模なグラフの描画や再配置などがスムーズに行えることをさし，また大規模なグラフにおいての可読性をさす．また，以前の可視化機能では振動や発散により再配置が不安定な状態に陥ってしまうような複雑な構造をなしたグラフ構造も安定した状態で再配置が行えることを目指す．

UNYO UNYO は，再配置の過程もリアルタイムに描画するため，グラフ構造の変化も視覚的にとらえやすく，ユーザによる直観的な操作で再配置を制御するインタラクティブなインタフェースを有する．

3.2　各システムのモジュール化

UNYO UNYO では，機能ごとにシステムを分離しモジュールとして独立させ，拡張性や運用性などを高めている．

システムは以下に示すの構造に分けることができる．（各モジュールの詳細については後述する．）

モジュール名	主な役割
Mediator	処理系と UNYO UNYO の仲介を担い結合度を低く保つ
Graph Container	処理系と UNYO UNYO の持つグラフ構造を矛盾なく管理する
Reallocator	目的地算出と移動処理を担う
Interface	ユーザに対するすべての入出力を担う

4　Mediator の設計と実装

LMNtal 処理系との仲介を担当し，処理系の変更に伴う影響を吸収する．

処理系からグラフ構造の取得　LMNtal 処理系に実行するプログラムを渡しグラフ構造に変換させ，処理系が変換したグラフを受け取る．

また，Mediator は処理系によるグラフ書換えの通知も受け取る．グラフが書き換えられた場合は，Graph Container へ通知し，Active Node の情報を更新させる．ただし，Active Node と Node の同期をとるのは Graph Container の役割であるため，Graph Container への通知までが Mediator の役割となる．

差分の利用　LMNtal 処理系により書き換えらたグラフは Graph Container で解析され変更を Acitve Node に反映させることになるが， Graph Container でグラフ全体をチェックし，変更箇所を見つけ出すには高いコストがかかる．そこで，Meidator は処理系からグラフの差分情報をのみを受け取る．

差分情報は「生成されたアトム」,「削除されたアトム」,「変更されたアトム（リンクの張り替えを含む）」,「生成された膜」,「削除された膜」,「変更された膜」

5　Graph Container の設計と実装

Node　Graph Container 部の役割の一つに LMNtal 処理系を意識しないグラフへのインタフェースを提供することがる．そこで， LMNtal 処理系のグラフへリアルタイムにアクセスするメソッド群を用意した．これによりシステムはグラフのアトムや膜といったグラフ構造にたいして， Node として容易にアクセスすることが可能となる．

Active Node　Active Node は UNYO UNYO が LMNtal 処理系のグラフを元に生成するグラフである．UNYO UNYO は必ずしも LMNtal 処理系の保持するグラフをすべて可視化するわけではなく，必要に応じて必要な Node のみ Active Node として生成し，可視化する．不要な Node の Active Node 化を行わないことで，メモリや処理コストを軽減する．

6　再配置 の設計と実装

Graph Container に格納されている Active Node に対し，目的地の座標の算出と，実際の移動処理を行う．また，目的地の算出と実際の移動処理とは別スレッドで非同期的に行われている．

移動処理と目的地算出処理の分離　目的地座標の計算には，ばねモデル，リンクの角度調節，斥力，引力など複数のアルゴリズムを使用し，結果を平均化することで決定する．処理には高いコストがかかるため，大規模なグラフをリアルタイムに再描画するための課題の一つとなっていた．そこで，最適と思われる座標を記憶させ，移動の際はこの最適座標へ向かって一定距離ずつ移動させる．この処理の利点は，コストの高い座標計算演算処理と比較的コストの低い移動処理とを切り分けれるところにある．これにより，目的地の算出処理と，移動処理とを別スレッドとして非同期的に実行することが可能である．移動処理が実行されると位置関係が変化するため再計算の必要性がある．ただし，一度の移動処理で移動する距離は一定に制限されているので，位置関係が大きくは変化しないと期待でき，移動処理よりも低い優先度で実行することがで出来る．

活性度による負荷軽減　Active Node は活性状態に関するフラグを保持している．活性状態にあるものは，目的地座標の計算を行い，移動処理を行うが，活性状態にないものはいずれも行わない．これにより，処理が不要なものに対して，不要なコストをかけなくて済むようになる．移動処理を行う際，現在地と目的地が等しかった場合，移動が完了したものとみなされ，非活性状態に置かれる．Interface 部はこの活性度を監視しており，すべての Active Node が非活性状態になった場合，再描画を停止する．これにより，すべての Active Node の移動処理が終わった場合，再描画にかかるコストがほぼなくなる．

7　Interface の設計と実装

グラフの理解についてのスケーラビリティを向上させるためには，理解を助けるための豊富なインタフェース機能が不可欠である．以下に実装したインタフェースの例をあげる．

機能	効果
縮小地図	別ウィンドウで全体表示
膜の開閉	膜の中身を隠蔽・開示する
Active Node の固定	システムによる再配置を 行わすその場に固定する
グラフの回転	配置の回転移動
グラフの編集	ユーザによるグラフ構造の書換え

8　考察

8.1　差分の利用による効果

a という名のアトムを一個ずつ増やすのプログラムを 5000 回実行するのにかかった時間を計測する．時間は 100 回 毎に計測し，三回の平均をとった．時間は処理系で実行された結果を UNYO UNYO が受け取ってから，UNYO UNYO の保持するグラフ構造の更新にかかる時間とする．つまり，LMNtal 処理系が要する時間は含まない．
結果を図 1 に示す．

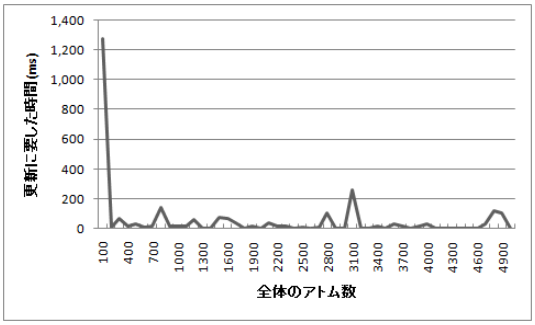

図 1: アトム一つ生成の所要時間

8.2　深さ 3 の四分木の可視化

```
a.
a :- a(b, b, b, b).
b(X) :- b(c, c, c, c, X).
```

上記のプログラムを LMNtal で実行した例を次に示す．

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

また，LMNtal ではリンクを明記することも可能であるため，以下のコード上記のコードと同義である．

```
b(_8,_10,_12,_14,_1), b(_17,_19,_21,_23,_3),
b(_26,_28,_30,_32,_5), b(_35,_37,_39,_41,_7),
c(_8), c(_10), c(_12), c(_14), c(_17), c(_19),
c(_21),c(_23), c(_26), c(_28), c(_30), c(_32),
c(_35),c(_37), c(_39), c(_41),a(_1,_3,_5,_7).
```

同じプログラムを UNYO UNYO で実行した結果を図 2 に示す．可視化による結果表示はテキスト表現とはことなり，配置が異なることはあっても，表記方法がことなることはない．また，再配置機能を使い，配置したもので手動での再配置は行っておらず，常に同じ配置を再現することも可能である．

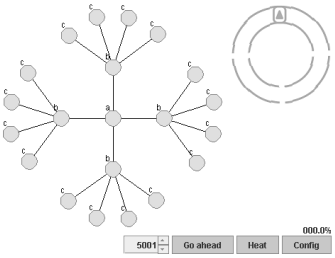

図 2: 深さ 3 の四分木