D6-4

視覚的項書換え環境のSMLによる実現

Implementation of Visual Term Rewriting Environment with Standard ML

河口 信夫 坂部 俊樹 稲垣 康善
Nobuo KAWAGUCHI Toshiki SAKABE Yasuyoshi INAGAKI

名古屋大学工学部
Faculty of Engineering, Nagoya University
E-mail : kawaguti@inagaki.nuie.nagoya-u.ac.jp

概要

項の構造の解析, 書換え系列の解析、項書換え系の変換など, 項やその計算に対する操作を視覚的に支援することを目的とする環境を, 最近注目されている関数型言語StandardMLを用いて実現した. 本稿ではインタラクティブなシステムの実現が難しいとされている関数型言語を用い, 実験的な視覚的システムが効率的に, かつ実用的なレベルで実現できることを示す. ユーザーインタフェースを実現するために, SMLを並列化したConcurrentMLと, eXeneライブラリを用いた.

1 はじめに

近年のハードウェア技術の急速な進歩により, これまでテキストベースであったシステムを, より複雑で多くのリソースを必要とするグラフィカルユーザーインタフェースを持つ視覚的なシステムに置き換えることが可能になった. しかしながら, ユーザーインターフェースの開発には使い易さなどのプログラムの正しさとは異なる意味でのトライアンドエラーが必要となり, 効率的な開発は難しい. また, 視覚的な表現力を持つシステムを作るためのノウハウは今だ十分とはいえず, 多くの工数を必要とする.

それらの欠点を補うために, 基本的なグラフィックインターフェースオブジェクトをライブラリ化したMotifライブラリや, 画面上で実際の画面を確認しながらユーザーインタフェースを作成するインターフェースビルダー, またTcl/Tkなどのインタフェース記述言語などを用いる方法がある. これらの手法を用いることで, 基本的なグラフィカルユーザーインターフェースを容易に作成することができる. しかし, 対象分野ごとに異なるデータの視覚化を行なうためにはどうしても低レベルのプログラムを書かざるを得ず, 特に様々な視覚化手法を試す必要がある場合には困難を伴う. 特に実験的なシステムの場合, 視覚化の手法を頻繁に変更することがあり, そのたびに低レベルのプログラミングを行なう必要がある.

これに対し, 本稿では一般にインタラクティブなシステムを作成することが困難であるといわれている関数型言語を用い, 実際の視覚的システムを作成し, プログラムの開発が容易であること, 十分に実用的な速度で動作すること, そしてシステムの変更が容易であり, 動作中でさえその振舞いを変更することができることを示す.

我々が実験の対象とした分野は, 同じ関数型言語のモデルの項書換え系である. 項書換え系は記号の書換えのみによって計算が進む単純な計算モデルであり, 視覚化の実験を行なうのに都合が良い. 我々の作成した視覚的環境は項書換え系の計算, 項の構造の解析, 書換え系列の解析, 数値データの解析など, 項やその計算に対する操作・解析を視覚的に支援することができる.

実現に用いた言語は強力な型推論の機構とモジュール化の仕組みを持つ関数型言語StandardML(SML)

を並列化したConcurrentML(CML)[3]と呼ばれる言語で, X Window System とのインターフェースには eXene ライブラリ [4] を用いた.

以下, 2節で CML 及び eXene についての簡単な説明を行ない, 3節では我々が実現したシステムについての解説を行なう. 4節ではシステムがどのような構成によって実現されているかを示し, 最後に, 5節で他のユーザーインタフェース作成システムとの比較を行ない, 本稿で提案した関数型言語を用いた視覚的システムの作成についての評価を行なう.

2 Concurrent ML/eXene

Concurrent ML(CML)[3] は SML の上で動作する並列関数型言語であり, 並列動作をするスレッド間ではチャネル及びイベントを用いた通信を行なうことができる. CML でも SML と同様に静的な型づけが行なわれ, 実行時には型エラーは発生しない. また SML のモジュール化の機構はそのまま利用することができる.

eXene[4] は CML 上に構築された X Window System とのインターフェースライブラリであり, Xlib と同等のレベルの図形描画関数のみでなく, Xtoolkit と同等のボタンやリスト, メニューなどの基本的なウィンドウオブジェクトを持つ. また, X サーバとの通信も C ライブラリを全く用いず, X プロトコルレベルからすべて CML で記述されている. 一つ一つのウィンドウ内には複数のスレッドを走らせることが可能であり, またマルチウィンドウも可能である.

3 項書換え視覚的支援環境 TERSE

視覚的項書換え支援環境 TERSE は「視覚化 (Visualization)」の概念を以下のように区別し, 詳細な考察に基づいて作成されている.

- **項の視覚化**: 項を様々な情報により拡張した木で表現する.
- **計算の視覚化**: 項書換えの計算の進行を視覚的に表現する.
- **操作の視覚化**: 項や規則に対する操作を視覚的に行なう.

これらに加え, 項書換え系の計算から得られる様々な数値情報をグラフ化することを情報の視覚化と呼び, 動的なデータの収集を支援することができる.

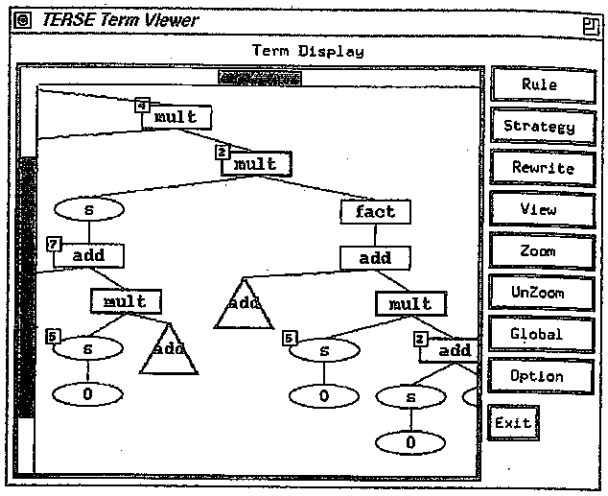

図 1: TERSE による項の視覚化

各々の視覚化の詳細については, [1, 2] に詳しいので参照されたい. 本稿では項の視覚化についてのみ説明する.

3.1 項の視覚化

図 1に示すのが, 視覚化された項とそれを操作するインタフェース TermViewer である.

TermViewer のウィンドウの右側にあるボタンを用いることで視覚化された項に対し様々な操作を行なうことが可能で, 後述する計算の視覚化やグラフ化のウィンドウもここから呼び出す. 図からも明らかなようにスクロール, 拡大, 縮小を自由に行なうことができる.

4 Standard ML による実現

TERSEの実現においてはデータ構造の決定やモジュール分割の設計を慎重に行なったため, 柔軟に拡張を行なうことができる. システム全体は図 2に示すような構造をしており、TERSEは大きく 3 つのモジュール (ユーザインタフェース部, TRS インタプリタ部, 木表現部) に分割できる.

それぞれはモジュールとして独立しており, 独立に変更することが可能である. また, 環境自身がコンパイラを含むため, 動作中に動的にシステムを拡張することができる. この仕組みによりユーザーが実行時に新しい機能を導入することが可能であり, 情報の視覚化に新しい計測関数を追加する機能が実

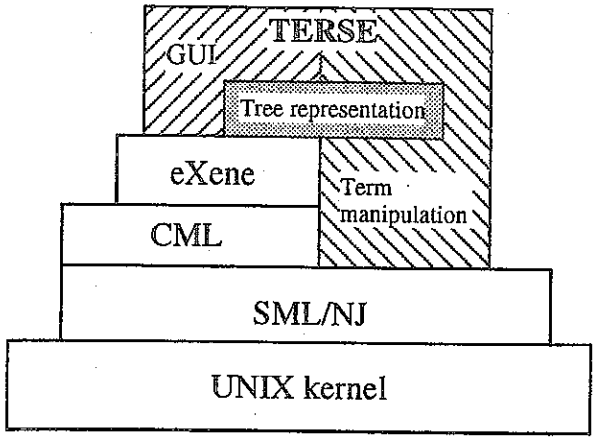

図 2: TERSE の構造

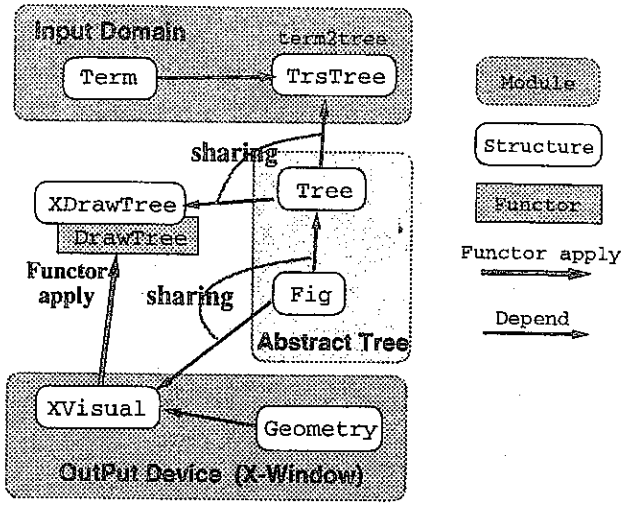

図 3: 木構造描画モジュールの依存関係

現されている. 例えば項の大きさ, 深さと幅の和のグラフを書きたければ以下のプログラムを読み込めばよい.

```
fun g_fun (t:term)=[term_size t
                   ,(depth t)+(width t)];
TermDialog.graph_fun :=  g_fun;
```

4.1 木構造の描画

TERSE では関数型言語による実現の利点を生かし, 多くの部分で高階関数や Functor を用いた実現を行なっている. ここではその一例として項の木構造を描画するアルゴリズムを挙げる. 描画モジュールの依存構造は図 3のように表せる.

DrawTree Functor は子供の木構造の幅を計算し, その中心に親を描くアルゴリズムを実現しており, 描画対象への描画関数を入力モジュールとし, 描画方向のベクトルを返す関数を引数にとることで任意の対象に対し任意の方向で木構造を描画することを実現している. 例えばポストスクリプトプリンタに出力する場合, ポストスクリプト用の描画モジュールを用いるだけで他の部分は全く変更する必要がない. TERSEでは項の視覚化, 計算の視覚化, ポストスクリプト出力など多くの視覚化において共通の木構造描画モジュールを使っており, 変更も容易である. また, DrawTree Functor は他のシステムでもライブラリとして容易に利用可能である.

4.2 実行速度

TERSEの実行速度についての実験結果を示す. 実験システムは SparcCenter 1000 であり, X 端末として XMiNT CSL を用いている. TERSEを用いて木構造の描画を行なわずに fact(4) を最外戦略で計算すると, 330 回の書換えを行ない約 2 秒で計算を終える. 一方, 描画を行ないながらの実行では 100 秒程度である. つまり 1 枚の画面の描画に約 0.3 秒を必要とする. この時, 描画された木のノード数は平均 38.5 であり, ある程度の大きさを持つ木を十分な速度で表示している.

インタラクティブな視覚的システムとしては操作を行なってから対応が起こるまでのレスポンスタイムは 1 秒以内が望ましいとされているが, TERSEは十分にこの条件を満たしており実行速度に関して実用的であるといえる.

4.3 ML の利点

ML をTERSEの実現に用いたのは以下に示すような多くの利点があるからである. 第 1 に開発効率が非常に高いことが挙げられる. 高階関数や様々なデータ型を自由に用いたプログラミングは簡潔な記述を可能にしている. また, 一般にユーザインタフェースの記述は複雑になりがちであるが, 並列関数型言語とイベントを用いた実現により, 単純でわかりやすい実現が可能である.

第 2 に対話形式の開発環境であるため, 実験的な実現が容易であることが挙げられる. 視覚化の実現

を行なうためには様々な試行錯誤が必要であるが，試験的なデータの入力や，プログラムの部分的実行，変数値の確認などを容易に行える．

第3に強力なモジュール化の機構により大規模なプログラミング (TERSE は CML で約 7000 行) に伴う困難が軽減されている．また，CML は多くの計算機上に処理系が実装されており，プログラムの移植性が高いことも利点の一つであろう．逆に欠点としては，現在の CML の実現では非常に多くのメモリを必要とすること（12-16MByte）などが挙げられる．

5 他の視覚的環境構築システムとの比較

視覚的（視覚化）システムの開発を行なう場合，その開発効率について，他のグラフィカルユーザーインタフェース構築システム（ライブラリ）を用いた場合と我々が採用した関数型言語 (Standard ML) を用いたアプローチとの比較を行なう．

5.1 Motif,NextStep(IB)

Motif ライブラリや NextStep Development システムは，現在最も広まっている GUI であり，どちらも多くのライブラリと優秀なインタフェースビルダー (IB) を持つ．見栄えと IB については我々のアプローチは劣っている．ただし，新しい eXene のリリースはより Motif に似た外見を持つはずである．しかし，IB を用いて得られるソースコードは多くの場合冗長であり，これを人の手で編集することは忍耐を必要とする作業である．

また，問題に特定な視覚化を行なうためのコードを書くには結局 Xlib を使う必要がある． C 言語 (及び Objective-C) と ML では圧倒的に ML のほうが容易にプログラムを構築することが可能である．これはデバッグの手間が少なくて済むことや，抽象度が高いことなどが理由であろう．ML には実行時エラーはないため，期待どおりに動かない時はアルゴリズムが間違っている場合であり，原因の追求が容易に行なえる．

5.2 Tcl/Tk

Tcl/Tk などの簡易言語（専用言語）を用いての開発は，ユーザーインターフェースに限れば非常に効率が良く，テキストアプリケーションをグラフィック化するフロントエンドの開発に適した手法である．しかし，専用言語であるがために汎用の型やレコードの概念に乏しく，複雑な構造を表現するのは難しい．項の視覚化などにおける複雑なデータ構造に密接した視覚化の機能を実現するためには強力なデータ型を持つ必要がある．ML では型を自由に追加できる柔軟な型システムを採用しており，複雑なデータ構造を容易に表現することができる．

6 まとめ

項書換え系の計算，項の構造の解析，書換え系列の解析，項書換え系の変換など，項やその計算に対する操作を視覚的に支援することを目的とする環境を，近年注目されている関数型言語 StandardML 上の ConcurrentML を用いて実現した．この環境は，マルチウィンドウを用いて複数の項，規則を同時に扱うことが可能であり，色や図形などにより修飾された木表現を用いて項を視覚的に表現する機能を持つ．CML を用いた実現が十分実用的な速度で動作することは，関数型言語がユーザインタフェースの実現に応用可能であることを示している．また，他のシステム構築ツールとの比較でも，速度的な問題は若干あるが，開発効率の点や抽象度の点では非常に有利であることを示した．

今後の課題としては，視覚的システムの作成を支援するための様々な視覚化ライブラリの作成，インターフェースビルダーの作成などが考えられる．

謝辞

日頃御指導いただく平田富夫教授，御討論下さった山本晋一郎助手，外山勝彦中京大学助教授，並びに研究室の皆様に感謝します．

参考文献

[1] 河口，坂部，稲垣: “グラフィカルユーザーインタフェースを持つ項書換え系の解析・変換支援環境”，信学技法, SS93-44(1994).

[2] N. Kawaguchi, T. Sakabe,Y. Inagaki:“TERSE: TErm Rewriting Support Environment”, *Proceedings of the 1994 ACM SIGPLAN Workshop on Standard ML and its Applications*, pp91-100 (1994).

[3] Reppy,J.H.: “CML: A higer-order concurrent language” ,*Proceedings of SIGPLAN'91*, pp293-305 (1991).

[4] Reppy,J.H.,Gansner,E.R.: “The eXene Library Manual(Version 0.4)” ,AT&T Bell Laboratories (1993).