ビジュアルシステム生成系 Eviss における Action の視覚化

Visualization of Action on Visual System Generator Eviss

西名　毅[†]　　田中 二郎[‡]

Tsuyoshi NISHINA　　Jiro TANAKA

†筑波大学院修士課程理工学研究科
Master's Program in Science and Engineering, University of Tsukuba
‡筑波大学電子情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba

概要

ビジュアルシステム生成系 Eviss は、図形文法に Action を導入することによってさまざまなビジュアル言語に対応できるシステムである。現在、Eviss では、図形言語を定義する際にテキストを用いて定義している。しかし、図形言語をテキストを用いて表現しているために理解をするのに困難な時があり入力にも時間がかかった。今回、よりユーザにとって直感的かつ一目で理解しやすいようにするために、定義に使用した図形言語をそのまま用いて Action を視覚化することを試みた。また、１つの CMG で連続した Action を扱う場合の表示手法を工夫した。その結果、テキストに比べて入力を簡略化することができ、テキスト入力時には難しかった Action の流れを容易に把握できるようになった。

1 はじめに

Eviss[1] [2] [5] [6] は、図形文法に Action を導入することによって、さまざまなビジュアルプログラミング言語に対応できるシステムである。

Eviss は、Spatial Parser Generator[3]と、制約解消系を持つ。Spatial Parser Generator は、図形言語の文法に従って図形言語を解析する部分(spatial parser)を生成し、制約解消系は、図形間に制約を課す。

また、図形言語の文法の記述方法としては Constraint Multiset Grammars(CMG)[4]が使われている。

2 Eviss の使用方法

Eviss のメイン画面を図１に示す。上側の Window は定義 Window で、下が実行 Window である。Eviss の使用法は、まず、上側の定義 Window で、図と対応させて１つの図形言語につき１つのルール(文法)を、CMG Window の形式で作成する。そして、下の実行 Window で解析したい図を描くと、文法に従って解析結果が実行されるようになっている。

3 従来の入力例

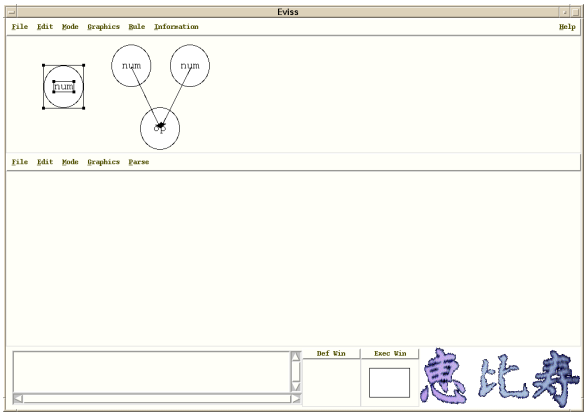

図１ システム概観

従来の入力方法を計算の木を例にとって説明する。計算の木は 2 つの文法から定義される。

①ノードは、円の中に数字が描いてあり、その円の中心が一致している。

②ノードは、2 つのノードが矢印によって円につながれ、その円の中には演算子が描かれていて、矢印の先と終点は、円とノードの中心と一致する。そして、計算を実行する。

図 1 の定義 Window には 2 つの図形言語の例が示されている。図形言語の左側が文法の①、右側が②と対応している。1 つの図形言語をセレクトし、メニューの rule 欄から make new production rule を選択すると、CMG Input Window が表示される。CMG Input Window は、図形の属性、アクション、制約を記述する Attributes、Action、Constraints 部分と、図形言語の構成要素の種類を表す normal、exist、not_exist、all の欄から成る。この時、CMG Input Window には、自動認識された制約と構成要素が記入されている。この中から必要ないものを削除し、足りないものを追加して書く。文法①と②を CMG Input Window に書いて定義した後に計算の木を下の実行 Window で描いてパースすると計算が実行され答えを得ることができる。図 2 に②に対応する CMG Input Window を示す。

4 Action について

Action は、パースした結果を利用して図形の描き換えなどを行うような機構を CMG に導入したものである。

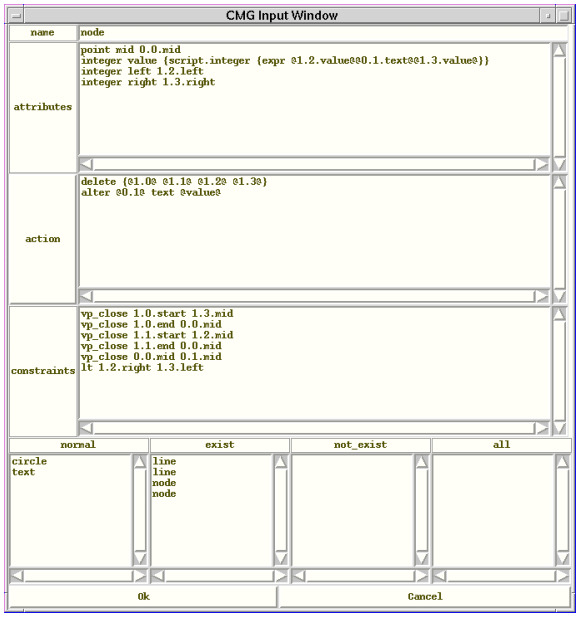

図 2　CMG Input Window の例

Eviss における Action の実行は、図形言語を認識するときに CMG Window の Action 欄に書いてある Action を実行することによって行われる。また、Action は「生成規則が適用された時に実行されるプログラム」として定義されていて、任意の Tcl のスクリプトを扱うことができる。

Eviss では Action の中で Tcl のスクリプトを書くのに便利な手続きとして、delete（図形の削除）、alter（図形の描き換え）、create（図形の生成）などを供給している。図形に関しての描き換えや生成の際には、この 3 つの Action でほとんど対応することができる。

以下に、3 章で述べた文法②で使われている Action のスクリプトを示す。上は、図形の削除でノード 2 つとライン 2 本を削除するのを表していて、下は、図形の書き換えで、演算子を計算結果に書き換えることを表している。

```
delete {@1.0@ @1.1@ @1.2@ @1.3@}
alter @0.1@ text @value@
```

5 視覚化した Action の入力と表示手法

Eviss では CMG をテキストで入力するが、これは、ユーザにとって直感的でない。そこで、ユーザにとって理解しやすいように入力を視覚化することが重要である。今回は、Constraint や構成要素などの視覚化が比較的容易にできると考えられる事から Action の視覚化を試みた。また、Action の中でも図形に関しては delete、alter、create でほとんど対応できるためこの 3 つの Action について視覚化する。

5.1 入力手法

我々は、Eviss を CMG Window の上から 3 番目にある action ボタンを選択すると、図 3 のような Action 定義 Window が表示されるよう改良した。この Window には左側と右側の両方に、制約や属性に従ってパースされた図形言語が表示されている。この Action の記述方法は実行前を左に実行後を右として表現する。よって、右側に描かれた図形言語を書き換えて Action を表現する。以下に 3 章で述べた計算の木で使われている delete と alter の描き方を示す。また、create を視覚化した例を、リストの例を使って示す。

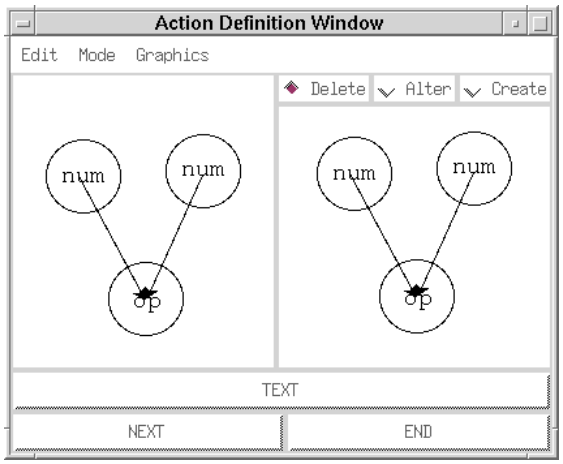

図 3　Action 定義 Window

delete は、まず、図 3 の状態でメニューから◇Delete を選ぶ。それから削除したい構成要素を右の実行後の Window から選択し、その図形言語を実際に削除することによって delete を表現する。図 4 は削除したい構成要素を消した後の Window である。

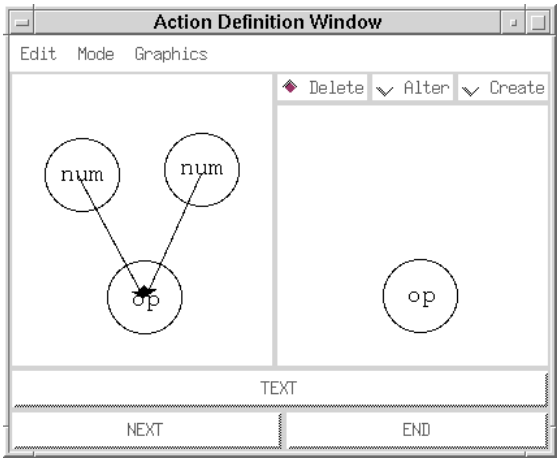

図 4　delete の視覚化

delete 入力終了後に左下の NEXT を押すと、前の Action の結果が左側と右側の両方に描かれている画面がさらに現れる。alter は、◇Alter を押し、右に表示されている図形言語を書き換えることによってそれを表現する。描き換える際に消したものを“描き換えられるもの”、描き加えたものを“描き換えるもの”という具合に認識させる。ここで、@value@ というのは計算結果の値の事で、前後の@は CMG で構成要素の属性の値を参照するために用いられる記号である。図 5 に、書き換えた後の Window を示す。

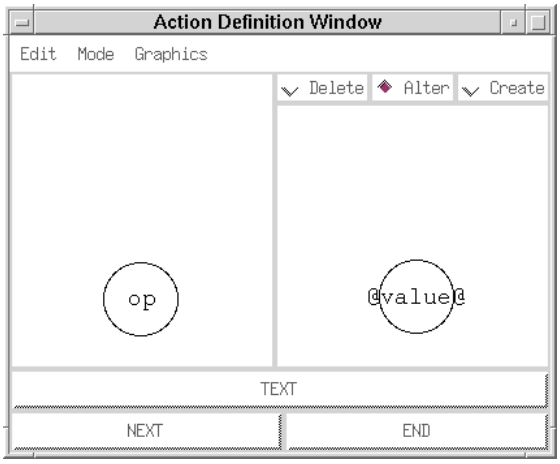

図 5　alter の視覚化

create は、計算の木ではなくリストを使って説明する。◇create を押して実行後の Window の図形言語に、実際に生成したい図形言語を書き加えることによって create を表現する。このとき、書き加えられた図形言語の座標は、既にある図形言語の座標との相対的な位置から計算される。図 6 に図形言語を生成した後の図を示す。

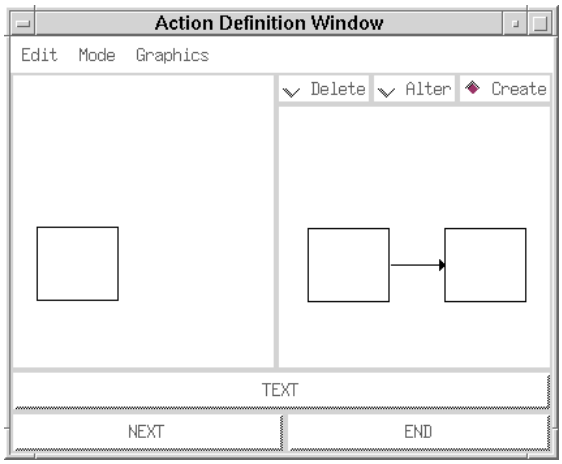

図 6　create の視覚化

このように定義すると四角を 1 つ描けばリストがどんどん生成される。制約の欄に描かれる四角の数の上限を指定すると指定した数のリストを作ることができる。

5.2 Action 連続時の表示手法

5.1 に述べた方法では Action が連続でいくつか続いた場合に複数の Window を表示しなければならないので Window が重なってしまうために、見づらか

ったり Action の流れを把握するのが困難である。そこで、連続時には複数の Action 定義 Window を縦につないでミニチュアで表示する方法を提案する。

ここでは、縦に隣り合った 2 枚で 1 つの Action を表している。5.1 節で示されているの計算の木の場合を例にあげる。まず、図 4 の左右の Window を上下に組み合わせる。そして、図 5 の Window も同じようにする。図 4 と図 5 を縦に組み合わせる。この時に図 4 の右（下）と図 5 の左（上）は同じ表示なので省略する。そして、ミニチュアにして全体の Action を表示する事によってすべての Action を見ることができる。マウスのカーソルをミニチュアの Window にあわすとその部分が拡大表示されて見える。また、このようにすれば Action の流れがとぎれず、理解しやすい。図 7 に計算の木の例を用いた連続時の表示図を示す。

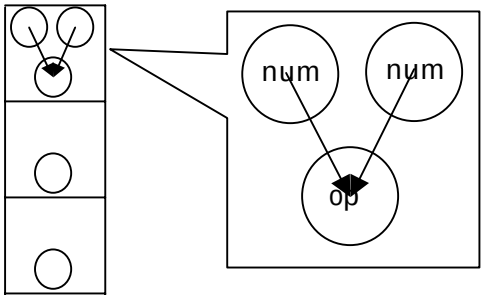

図 7　計算の木の表示図

図 8 に別の Action 連続時の表示例を示す。この図は、計算の木を使って X のｎ乗を実行するための図である。最初に、ノードに同じ数字を入れて掛け

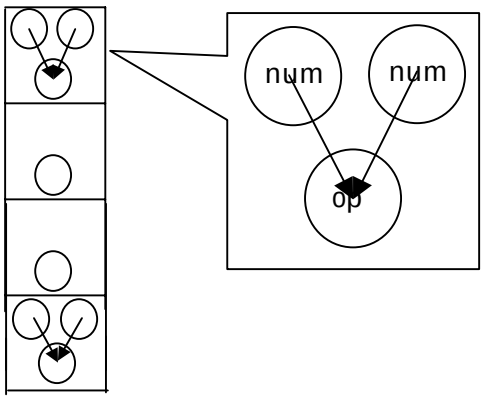

図 8　X のｎ乗の表示図

算をする。その計算結果に、最初に入れた数字と同じ数字であるノードを１つと、演算子のノードを生成して計算の木を作り、計算を繰り返す。これを制約の欄でｎ回以上いかないように指定することによって表現している。

6 まとめ

我々は、図形言語を描き換えたりする際によく用いられる Action の delete、alter、create を視覚化し、編集できるようにした。視覚化することによってより直感的になり、Action の実行結果を見ることができるようになった。そして、テキスト入力ではわかりにくかった構成要素の欄から参照して対応を調べることや、定義したい図と比較しながらの入力が、1 個所で理解できるようになった。また、視覚化するだけでなく、Action 複数時の表示を工夫することによって Action の流れを把握しやすくなった。

なお、飯塚和久氏には多くの貴重なコメントをいただいた。ここに感謝の意を表します。

参考文献

[1] 馬場昭宏, 田中二郎: 「恵比寿」を用いたビジュアルシステムの作成, 情報処理学会論文誌, Vol.40, No2, pp497-506, 1999

[2] 馬場昭宏, 田中二郎: Spatial Parser Generator を持ったビジュアルシステム, 情報処理学会論文誌, Vol.39, No5, pp1385-1394, 1998

[3] E. J. Golin and T. Magliery: A Compiler Generator for Visual Languages. *Proceedings of the 1993 IEEE Symposium on Visual Languages*, pp.314-321, 1993

[4] S. S. Chok and K. Marriott: Automatic Construction of User Interfaces from Constraint Multiset Grammars. *Proceedings of the 1995 IEEE Workshop on Visual Languages*, pp.242-249, 1995

[5] A. Baba and J. Tanaka: Eviss: A Visual System Having a Spatial Parser Generator, *Proceedings of Asia Pacific Computer Human Interaction* 1998 (APCHI'98), July, pp.158-164, 1998

[6] 馬場昭宏, 田中二郎: GUI を記述するためのビジュアル言語. インタラクティブシステムとソフトウェアⅤ, pp.135-140. 近代科学社, 1997