直接操作を用いたグラフィカルな図形文法編集システム

Graphical definition system for visual grammar using direct manipulation

亀山 裕亮 † 志築 文太郎 †† 田中 二郎 ††
Hiroaki KAMEYAMA Buntarou SHIZUKI Jiro TANAKA

† 筑波大学大学院工学研究科
Doctoral Program in Engineering, University of Tsukuba
†† 筑波大学電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba
{kame,shizuki,jiro}@iplab.is.tsukuba.ac.jp

一般のプログラミング言語の文法に対し，入力が四角形や円といった図形である文法を図形文法と呼び，図形文法を定義することで簡単にビジュアルシステムを作成することができる．我々は図形文法として，構成要素，制約，属性，アクションから構成される拡張 CMG を対象とし，図式表現を用いて拡張 CMG を定義するインタフェース GIGA を作成している．GIGA では，図形を画面上に直接描画することにより構成要素を定義する．定義した構成要素の位置を制約を満たすように配置することで制約を定義し，推論された制約が画面上に強調して表示されるため，インタラクティブに制約の定義を行うことができる．図形の書き換えを行なうアクションについては，入力した図式表現の他に，アクションが実行された後の図式表現を併せて編集することにより定義する．GIGA を用いることで，拡張 CMG をより直感的かつインタラクティブに定義することができる．本稿では状態遷移図の文法を定義する例を挙げ，GIGA のインタフェースについて述べる．

1 はじめに

一般のプログラミング言語の処理系では，入力されたテキストに対しパーサが文法に基づいて解析を行なう．また，プログラミング言語の文法の記述から，そのようなパーサを自動的に作成する生成系として Yacc[7] や Rie[6] などが存在する．これに対し，ダイヤグラムエディタのように図形を扱うビジュアルシステムでは，入力がある規則のもとに組み合わせられた長方形や円などである．この規則を定義する文法を図形文法と呼び，ビジュアルシステムの図形文法記述からパーサを自動的に作成する空間解析器生成系として，我々が開発を行なってきた恵比寿 [1][2][5] や，Chok らの Penguins[3][4] などがある．

従来，図形文法の記述にはテキストが用いられてきた．しかし，図形文法では図形の位置や大きさなど，2 次元的な情報を表現する規則を扱うことが多く，テキストによる記述だけでは文法が表す意味を直感的に理解しつつ定義を行なうことが困難であった．

我々は図式表現を用いてグラフィカルに図形文法を編集するインタフェースを提供することで，図形文法の定義と意味の理解の困難さを解決するシステム GIGA を作成している．GIGA では，図形を直接操作することによって図形文法を定義することができる．本稿では状態遷移図の文法を GIGA を用いて定義する例を挙げ，GIGA のインタフェースについて述べる．

2 拡張 CMG

Chok らは空間解析器生成系 Penguins で図形文法として CMG[8] を用いている．CMG は，記述することのできるクラスが広い，バックトラックなしに解析を行なうことができる，などの特徴を持つ．CMG を用いて記述されたビジュアルシステムの例として，Chok らはフローチャート，n 分木，数式，状態遷移図などのエディタを報告している [4].

しかし，実際には図形間の解析を行なうだけでは十分ではなく，解析結果に応じた動作を行ない，図形間に制約を課すことによって意味的関係を保存し，さらには描いた図形へのフィードバックを行ないたい．

そこで我々は，解析時に，図形の追加や削除，図形の属性の変更などを行なえるように，CMG にアクションを導入した拡張 CMG(Extended CMG) を提案した [1]．拡張 CMG で追加したアクションとは図形の生成（create），図形の削除（delete），図形

の属性値の書き換え（alter）である．

拡張 CMG のルールの構文を以下に示す．

$$
\begin{array}{lll}
T & ::= & T_1, \cdots, T_n \quad where(\\
 & & Constraints \\
 &) & \{ \\
 & & AttributeAssignments \\
 & \} & \{ \\
 & & Actions \\
 & \} &
\end{array}
$$

T はルールが適用された時に新しく作成される**図形単語**である．$T_1, \cdots, T_n$はルールの**構成要素**である．構成要素 T_i は図形単語か図形である．図形単語は図形や図形単語自身の再帰的な組み合わせである．図形は文法を構成する基本単位であり，円 (circle) や長方形 (rectange)，テキスト (text)，直線 (line) などがある．*Constraints* は**制約**である．制約は，構成要素の属性の間の等式関係や不等式関係を組み合わせた条件式である．ルールが適用されるためには，構成要素が存在し，かつ制約が満たされる必要がある．*AttributeAssignments* は**属性**を定める．T の属性がここに記述された代入式により決まる．*Actions* はこのルールが適用された時に実行される**アクション**である．

3 GIGA

我々は図式表現を用いてグラフィカルに拡張 CMG を編集するインタフェースを提供することで，文法の定義と意味の理解の困難さを解決するシステム GIGA を Java 言語を用いて作成している．GIGA では図形を直接操作することによって，拡張 CMG の構成要素，制約，属性，アクションを定義することができる．

3.1 構成要素の定義

構成要素となる基本図形もしくは図形単語を定義するために，GIGA の下部にあるボタンから目的の図形を選択し，図形を画面上に描画する．基本図形として使用できる図形には，円 (circle) や長方形 (rectange)，テキスト (text)，直線 (line)，画像 (image) がある．あらかじめ定義した図形単語を入力することもできる．

なお，図形単語は図 1 に示すように中心に図形単語の名前の付いた長方形として表示され，座標値を表わす属性もあわせて表示される．例えば node という図形単語が 3 つの属性 (中心の座標値，左端の座標値，右端の座標値) を持つ場合には，図 1 に示すように，各座標に円が表示される．

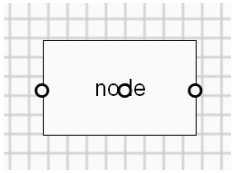

図 1: 図形単語の属性表示

3.2 制約の定義

制約の定義を行なうには，構成要素の図形を制約を満たすように配置する．GIGA は配置された構成要素間の属性を比較し制約を自動生成する．

3.2.1 重ね合わせによる図形の関連付け

入力されたすべての構成要素間で属性の比較を行なうと，不必要な制約が多数生成されるという問題がある．GIGA では制約を定義したい構成要素同士を重ね合わせることで**関連付け**を行ない，関連付けされた構成要素間でのみ制約を出力することで，不必要な制約の出力を抑えている．

3.2.2 基本図形の属性指定

基本図形にはシステムにより様々な属性が用意されている．例えば長方形には座標値や幅，高さ，色が属性として用意されている．これらの属性はルールを定義するために用いられるが，基本図形が持つすべての属性が定義に使用されることは少ない．「長方形であればどのような色や幅を持っていても解析を行なう」というルールのように，ルールに必要な属性だけが使用される．GIGA では制約を推論する際に，入力された図形の属性をすべて使用せずにあらかじめユーザに指定された属性のみを用いて制約を出力する．

例えば何も属性が選択されていない状態では，長方形は図 2-a に示すように半透明で表示されている．

長方形の上に表示されている 9 つの円は長方形の 4 つの頂点と各辺の中心，対角線の交点の座標を表わしている．ユーザはこの中からルールを定義するために必要な属性のみを選択する．例えば長方形の中心の座標をルールで使用したい場合には，中心にあ

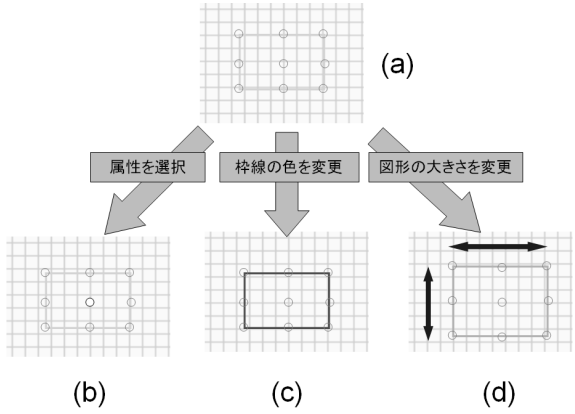

図 2: 基本図形の属性表示

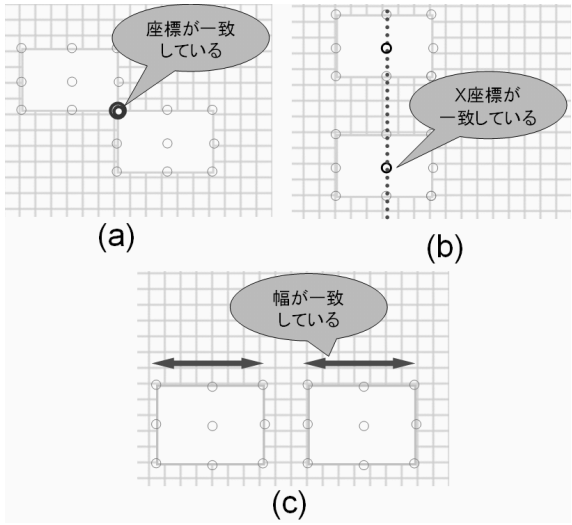

図 3: 制約の強調表示

る円をマウスでクリックする．これにより図 2-b に示すように，選択された円 (属性) が半透明ではなくなり，この属性を制約の定義を行なう際に使用することができるようになる．

長方形の枠線の色をルールで使用したい場合には，長方形の枠線の色を変更する．枠線の色を変更すると図 2-c に示すように，枠線の色が実際の色で表示されるため，枠線の色を属性として使用することを確認することができる．

長方形の幅や高さをルールで使用したい場合には，長方形の大きさを変更する．大きさを変更すると図 2-d に示すように，変更された部分が属性として使用されることを矢印を用いて表示される．

3.2.3 制約の視覚化

GIGA では図 3 に示すように制約を強調して表示することで，定義した制約を直感的に理解する手助けをしている．

図形の座標が他の図形の座標と完全一致している場合には図 3-a のように，一致している座標を表わす属性を強調表示する．

図形の座標が他の図形の座標とと完全に一致してはいないが，X 座標 (もしくは Y 座標) が一致している場合には図 3-b のように，座標を表わす属性同士を通る太い点線でガイドラインを表示する．

図形の大きさを表わす属性が，他の図形の大きさの属性と完全に一致している場合には，3-c のように大きさの一致している部分を表わす属性を強調表示する．

3.3 アクションの定義

拡張 CMG のアクションとは，解析が行なわれた時に実行される動作の集合である．アクションの中で図形の書き換えに関する規則については，入力した図式表現の他にアクションが実行された後の図式表現を併せて編集することにより，アクションの定義を行なう．

GIGA では構成要素，制約についての定義を行なった後，画面下にあるアクションと書かれたボタンを押すことで，作成した図形がそのまま複製される．この図を編集し，アクションが実行された後の図を作成することで，アクションの定義を行なう．GIGA ではこれらの 2 つの図の違いと図形に対する操作履歴から拡張 CMG のアクションを推論し生成する．例えば delete アクションを定義するには，図 4 に示すように削除したい図形を削除する．

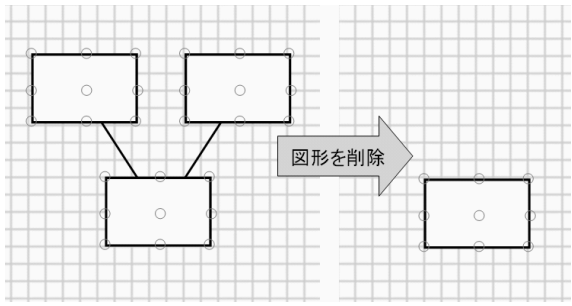

図 4: アクションの定義

3.4 属性の定義

構成要素の属性値を図形単語の属性として定義する場合，構成要素の属性を直接指定することで新しい属性の定義を行なう．具体的には，図 5 に示すように属性を表示している円をクリックすることでその円が黄色で表示され，その座標が図形単語の属性として定義される．また，座標値以外の値を持つ属性を定義する場合には，画面上で何もない部分をクリックすることで，属性を入力するためのボックスが出てくるので，ボックスの中に属性の計算式を入力することで図形単語の属性を定義することができる．

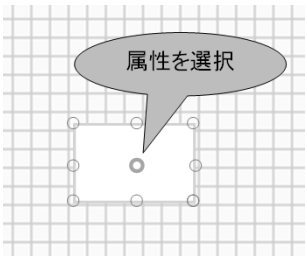

図 5: 属性の定義

4 図形文法の定義例

GIGA を用いて図形文法を定義する例として，文献 [4] に挙げられている，図 6 のような状態遷移図を編集するダイアグラムエディタを定義する．

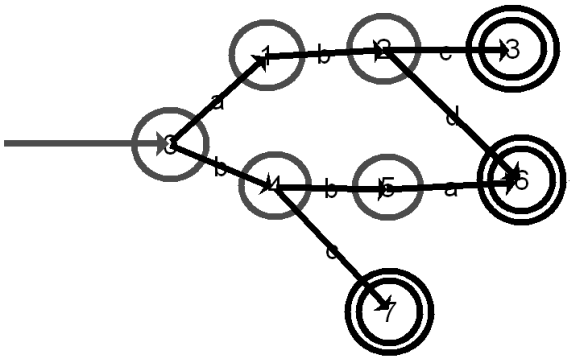

図 6: 状態遷移図

状態遷移図は以下の 6 つのルールで定義することができる．GIGA を用いて定義した状態遷移図のルールを図 7 に示す．

(a) 遷移線
状態の遷移を表わす線は，遷移の条件を表わすテキストと状態同士を結ぶ直線の組合わせとして定義を行なう (図 7-a)．まずルールに arc という名前を付ける．次に構成要素としてテキストと直線を 1 つづつ描画した後，それぞれの中心の座標値を属性として使用することを指定するため，図形の中心にある円をクリックする．テキストと直線の中心を重ねる．あらかじめ指定した属性同士を重ねることで，GIGA により制約が推論され，強調表示が行なわれる．遷移線の属性として直線の両端と中心の座標を定義するために，各座標に描画されている円をクリックする．これにより，それらが属性として定義される．

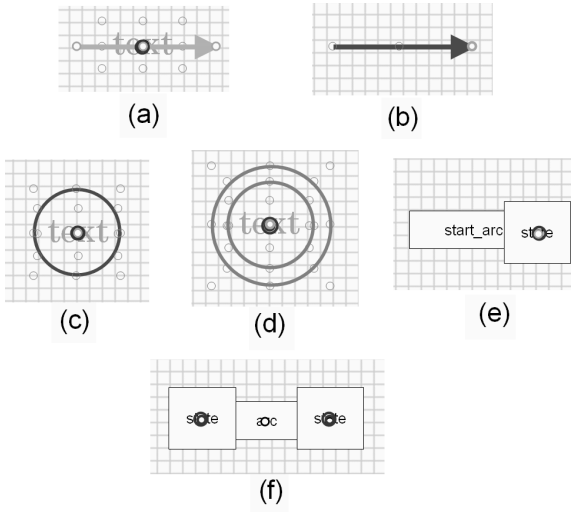

図 7: 状態遷移図のルール

(b) 開始線
状態遷移の開始を表わす線は，通常の線と区別するために赤い直線として定義を行なう (図 7-b)．まずルールに start_arc という名前を付ける．構成要素として直線を 1 つ描画し，線の色を赤に変更する．開始線の属性として直線の終点の座標を定義するために，直線の終点に描画されている円をクリックする．

(c) 状態
状態遷移図の各状態は，テキストと赤い円の組み合わせとして定義を行なう (図 7-c)．まずルールに state という名前を付ける．構成要素として円とテキストを画面上に描画した後，円の色を赤に変更する．次に中心の座標値を属性として使用することを指定するため，それぞれの図形の中心にある円をクリックし，円とテキストの中心を重ねる．円の中心座標とテキストの値を

終了状態の属性として定義するために，円の中心にある円とテキストの文字列をクリックする．

(d) 状態 (開始状態)
状態遷移の開始を表わす開始状態は，開始線と状態の組み合わせとして定義を行なう (図 7-d)．まずルールに state という名前を付ける．構成要素として開始線と状態を 1 つづつ描画し，開始線の端点と状態の中心に描かれている属性同士を重ねる．状態の中心座標を開始状態の属性として定義するために，状態の中心にある円をクリックする．

(e) 状態 (終了状態)
状態遷移の終了を表わす終了状態は他の状態と区別するために，テキストと二重の円の組み合わせとして定義を行なう (図 7-e)．まずルールに state という名前を付ける．構成要素として円を 2 つとテキストを 1 つ画面上に描画した後，中心の座標値を属性として使用することを指定するため，それぞれの図形の中心にある円をクリックする．次に 2 つの円とテキストの中心を重ねる．円の中心座標とテキストの値を終了状態の属性として定義するために，円の中心にある円とテキストの文字列をクリックする．

(f) 状態遷移状態の遷移は 2 つの状態と遷移線の組み合わせとして定義を行なう (図 7-f)．まずルールに transition という名前を付ける．構成要素として状態を 2 つと遷移線を 1 つ画面上に描画した後，線の端点と状態をそれぞれ重ねる．

5 まとめ

本論文では，拡張 CMG をグラフィカルに編集するために作成したシステム GIGA について述べた．

GIGA では，ルールの各構成要素を視覚的に表現し，各種の手法を用いてそれらの要素に対する直接操作を行なうことで，ルールの定義を行なうことができ，さらにひとまとめに図式表現された図形文法のルールからその意味を容易に把握することができる．

GIGA を用いてグラフィカルに文法の編集を行なうことで，図形文法をより直感的かつインタラクティブに編集することができる．

参考文献

[1] 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, No. 5, pp. 1385–1394, Jul 1998.

[2] 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, No. 2, pp. 497–506, 1999.

[3] Sitt Sen Chok and Kim Marriott. Automatic Construction of User Interfaces from Constraint Multiset Grammars. pp. 242–249. Proceedings of IEEE Symposium on Visual Language, 1995.

[4] Sitt Sen Chok and Kim Marriott. Automatic Construction of Intelligent Diagram Editors. pp. 185–194. Proceedings of ACM Symposium on User Interface Software and Technology, 1998.

[5] Kazuhisa Iizuka, Jiro Tanaka, and Buntarou Shizuki. Describing a Drawing Editor by Using Constraint Multiset Grammars. pp. 119–124, Zhengzhou, China, Nov 2001. Proceedings of the International Symposium on Future Software Technology (ISFST2001).

[6] 石塚治志, 佐々政孝, 中田育男．1 パス型属性文法に基づくコンパイラ生成系 Rie．コンピュータソフトウェア, Vol. 10, No. 3, pp. 20–36, 1993.

[7] Stephen C. Johnson. Yacc: Yet Another Compiler-Compiler. *UNIX Programmer's Manual Seventh Edition*, Vol. 2, pp. 353–387, 1979.

[8] Kim Marriott. Constraint Multiset Grammars. pp. 118–125. Proceedings of the IEEE Symposium on Visual Languages, 1994.