一般的な情報の視覚化モデル

2T-1

鎌田富久　川合　慧
(東京大学　理学部)

1．はじめに

情報（データ）を分かりやすく図で表現することはユーザインタフェースには欠かせない要素である。計算機で扱う様々なデータを様々な形式で視覚化したいという要求がある中、図化モジュール生成の効率を上げるためには、データの形式および図の形式に依らない一般的な図化の枠組みが必要である。このような図化システムには、汎用的な図の部品だけではなく、関係構造から図構造への高レベルな対応づけが必要不可欠である。

我々は、データ形式を記述する形式言語の生成規則に図化規則を対応させる手法[1, 2]を発展させ、データ形式に依らない関係構造の一般的な記述から図構造に変換する手法を考案した。本稿では、我々の視覚化モデルの基本構造とその実現について述べる。

2．視覚化（図化）モデル

図は、図形要素をある（幾何学的な）規則で並べたものという意味で一種の言語である。その図の中の規則が人間の自然に理解できる視覚的常識(visual metaphore)と対応しているとき、分かりやすい図ということができる。従って、あるテキスト形式で記述されるデータ(textual rep.)をあるタイプの図(visual rep.)に変換する問題は１次元の言語から２次元の言語への翻訳と見ることができる。

我々は、この図化過程を図１のような枠組みで考える。まず、様々な形式(syntax)で記述されるデータを、その内容の関係構造の表現に変換する。一般に情報の関係構造はネットワーク構造になり、現在のところその表現にはＰｒｏｌｏｇを用いている。次に、図化マッピングにより関係構造を図構造に変換する。図化マッピングには次の２種類がある。

・要素マッピング…関係構造を構成する要素を図形要素（長方形、円など）に対応させる。
・関係マッピング…要素間の関係をそれらの要素の対応する図形要素間の幾何学的関係（並列／直列／環状配置、包含など）に対応させる。

この図化マッピングを変えることによって、別のタイプの図に変換することができる。現在、図化マッピングの指定はＰｒｏｌｏｇで記述するようになっているが、さらに高レベルな指定方法を研究中である。また、いろいろな種類の図化マッピングをライブラリ化すれば、図化の指定を容易にすることができるだろう。

最後に、図形要素間の関係・拘束条件(constraint)を解いて図形の位置を計算し、実際に図を生成する。我々は、このために図の記述を図形要素間の拘束によって指定できるＣＯＯＬというシステムを開発した。ＣＯＯＬは我々の視覚化モデルの中で中心的な役割をはたすもので、次に説明するようにconstraint-basedグラフィクス言語としても興味深いものである。

我々の視覚化モデルでは、ある形式のデータをある形式の図に変換するのにデータをparsing する情報と図化マッピングの指定をするだけでよい。このような汎用的な図化システムは他のシステムとの組み合わせやプロトタイプの作成などにきわめて有効である。

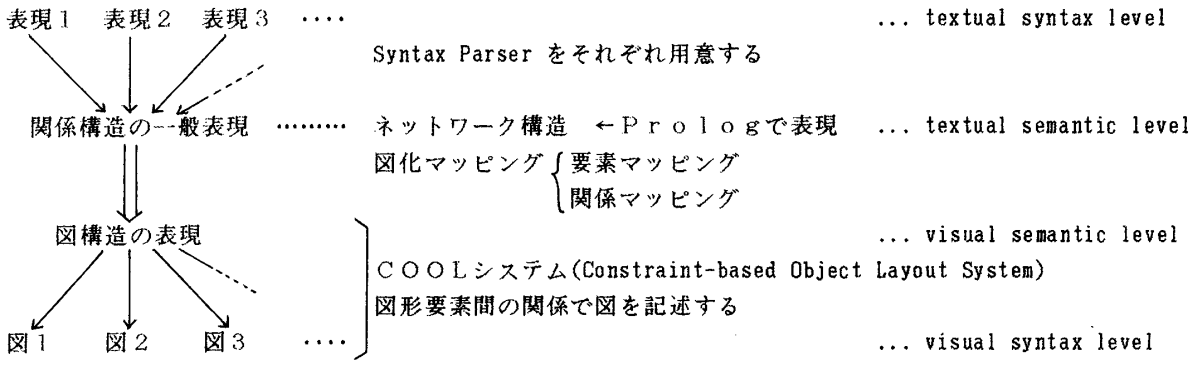

図１．図化の枠組み

On the Visualization of Abstract Objects and Relations
Tomihisa Kamada and Satoru Kawai
University of Tokyo

3．COOLシステム

図形要素は2（3）次元空間内で互いに拘束し合うので、それらの拘束条件、具体的には変数間の等式をすべて満たすようにシステムが値を決めなければならない。COOLは他のconstraint-basedシステムには見られない次のような特徴を持っている。

1. Generic Geometric Relations

各図形要素に共通の変数を持たせ、幾何学的な関係づけはそれらの変数を参照する。こうすることにより拡張が容易になる。例えば、box,circleは次のような内部変数と内部条件及び描画命令から定義されている。

```
box(x) {
   /* variables lx,rx,ty,by,cx,cy,wd,ht */
   constraint("$.cx = ($.lx + $.rx)/2", [x])
   constraint("$.cy = ($.by + $.ty)/2", [x])
   constraint("$.lx + $.wd = $.rx", [x])
   constraint("$.by + $.ht = $.ty", [x])
   drawing("box( $.lx, $.rx, $.by, $.ty)", [x])
}
circle(x) {
   /* variables lx,rx,ty,by,cx,cy,r */
   constraint("$.lx = $.cx - $.r", [x])
   constraint("$.rx = $.cx + $.r", [x])
   constraint("$.by = $.cy - $.r", [x])
   constraint("$.ty = $.cy + $.r", [x])
   drawing("circle( $.cx, $.cy, $.r)", [x])
}
```

2つの図形要素を水平・垂直に並べる条件は次のように定義される。

```
horizontal(a,b) {
   constraint("$1.cy = $2.cy", [a,b])
}
vertical(a,b) {
   constraint("$1.cx = $2.cx", [a,b])
}
```

2. Rigid and Pliable Constraints

条件が正しく解けるように過不足なく条件を記述することは簡単ではない。とくにネットワーク状の複雑な関係構造を何らかの幾何学的関係に置き換えて2次元平面に埋め込む際にこの問題は生じる。そこで、我々は条件を厳密な(rigid)ものと柔軟な(pliable)ものに分け、後者は近似的に求めるようにした。

つまり、条件が多すぎて全体として矛盾する(over-constrained)場合には、rigid constraintをまず解いて、次にpliable constraintを最小2乗法で近似的に解く。現在、constraintは1次式に制限しているが、点の距離などの2次のconstraintも入れる予定である。

図2に関係構造を図化した簡単な例を示す。いくつかの関係が別々の図形式で表現され、全体としてうまく組み上がっている。この例で、118 変数の間の等式が解かれている。

4．今後の拡張

我々の視覚化モデルは非常に一般的なものなので、反対方向の翻訳、つまり図からテキストへの変換に応用することができる。その1つの例は Direct Manipulation で、これはvisual semanticsから図化マッピングを調べてtextual semantics に逆変換し、その結果をまたvisualsemantics にfeedbackするという操作になる。また、図面認識などはちょうど逆向きの流れとして捉えることができる。このように図言語とテキスト言語の間の相互変換を統一的に扱い、両者を結ぶ統合システムを構築することが最終的な目標である。

参考文献

[1] 鎌田、川合：情報の図化インタフェース、情報処理学会第35回全国大会、1987.

[2] 鎌田、川合：情報の図化インタフェース－形式文法を用いた一手法、グラフィクスとCAD 30-1, 1987.

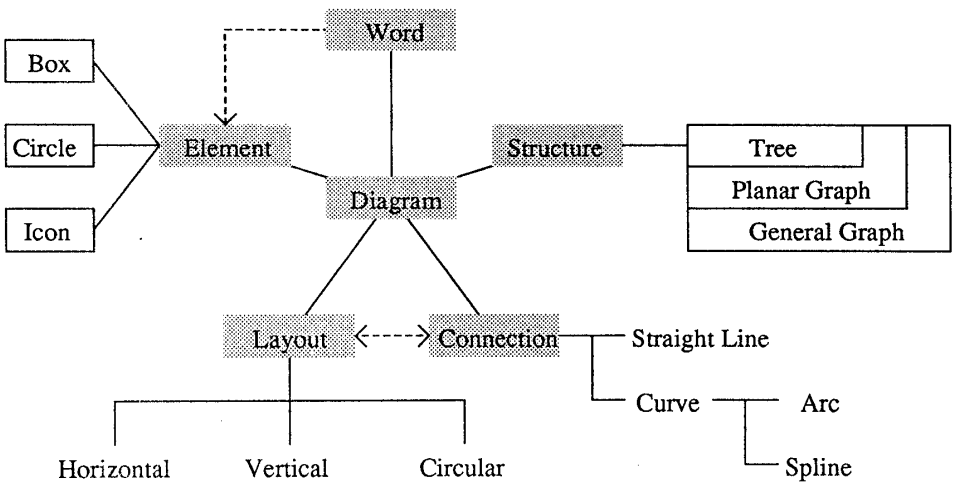

図2．関係構造の図化の例

使われている図化形式
- 並列配置
- 直列配置
- 環状配置
- x，y方向中心配置
- 包含
- 直線連結
- 折れ線連結
- 矢印連結