GOVE
Grammar-Oriented Visualisation Environment

Richard Webber and Aaron Scott

Department of Computer Science, University of Newcastle, Callaghan 2308, Australia

1 Introduction

Most Information Visualisations have been developed in an ad hoc manner. To overcome this, we have proposed an architecture that formalises the structure of Information Visualisations. By using this architecture, software developers will benefit from the use of tools that support this approach, and from the increased potential to reuse parts of visualisations built under the architecture. We have embodied this architecture in a software environment, GOVE (Grammar-Oriented Visualisation Environment) [7], that can be used to develop Relational Information Visualisations — visualisations that deal specifically with the entities and relationships found in the information source (which is currently limited to static, textual sources), usually presenting them as a graph. This form of Information Visualisation is very common, and accounts for much of the work in the field [1, 8].

2 Architecture

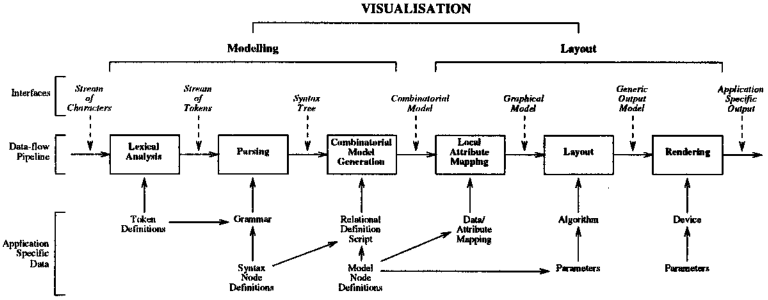


Fig. 1. Visualisation architecture

The traditional model of information visualisation consists of two parts [4, 5]: *Modelling* (extracting entities and relationships from an information source), and *Layout* (converting this into a graphical representation).

We propose an architecture which subdivides these parts into six modules, as outlined in Fig. 1. Together, these form a dataflow pipeline, which accepts an information source as input, and produces a visualisation as output. Unlike many systems, GOVE incorporates the process of obtaining relational information from the (textual) source.

Each module performs a specific task, and is made up of two logical components: *General Functionality* (the underlying engine for that module's part of the visualisation process), and *Application Specific Data* (the definitions that tailor a module to a specific visualisation). The interfaces between neighbouring modules are defined to allow modules from one visualisation to be inserted into the same position in another visualisation, greatly increasing the potential for reuse.

3 Modules

GOVE presents its user with an explicit visual representation of the Information Visualisation pipeline they are developing. A simple pipeline to visualise a graph description file is given in Fig. 2. The user interacts with this pipeline, supplying the application-specific data through a set of form-based interfaces (see Fig. 3), then instructing GOVE to build and execute the pipeline of modules to perform the visualisation.

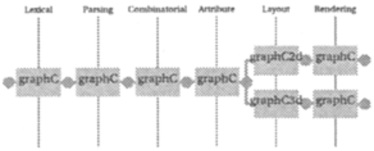
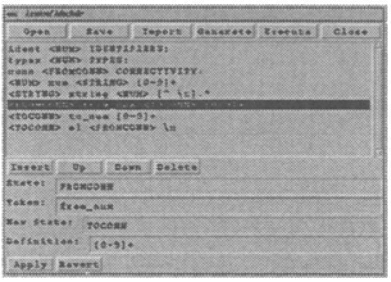


Fig. 2. Graph description pipeline



```
ident <NUM> IDENTIFIERS:
types <NUM> TYPES:
conn <FROMCONN> CONNECTIVITY:
<NUM> num <STRING> [0-9]+
<STRING> string <NUM> [^ \t]-*
<FROMCONN> from_num <TOCONN> [0-9]+
<TOCONN> to_num [0-9]+
<TOCONN> el <FROMCONN> \n
```

State: FROMCONN

Token: from_num

New State: TOCONN

Definition: [0-9]+

Fig. 3. Lexical Analysis interface

Lexical Analysis: divides a stream of characters (the textual source) into tokens. The application specific data for a Lexical Analysis module is a list of token names and definitions given as Lex [6] regular expressions.

Parsing: builds a syntax tree from the sequence of tokens it receives. The application specific data for a Parsing module is a Yacc [6] grammar.

Combinatorial Model Generation: analyses the syntax tree, and generates a Combinatorial Model (an Entity-Relationship Diagram). It does this by matching patterns in the syntax tree, and generating corresponding components in the Combinatorial Model. This process is similar to the construction of graphs through Graph-Grammars [3], and is shown in Fig. 4.

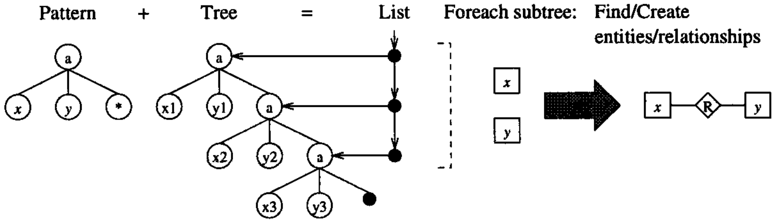


Fig. 4. Combinatorial Model Generation

Local Attribute Mapping: assigns graphical attributes to the entities and relationships of the Combinatorial Model. The application specific data consists of expressions that assign a value to each attribute for each *type* of entity and relationship, possibly using the data stored in the Combinatorial Model.

Layout (Global Attribute Mapping): applies the user's choice of layout algorithm [1] to assign positions to the nodes. GOVE has support for three-dimensional graph drawing, with an increasing set of layout algorithms.

Rendering: converts the resulting graph to the desired format for storage or viewing. The two visualisations resulting from the pipeline in Fig. 2 are shown in Figs. 5 and 6 (two- and three-dimensional spring layout respectively).

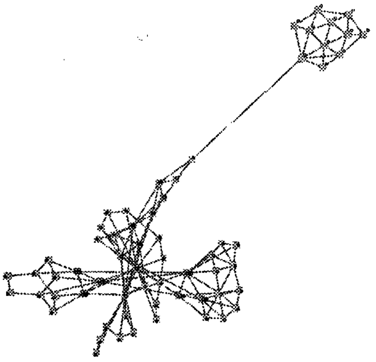


Fig. 5. Two-dimensional spring

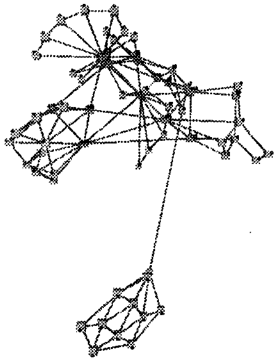


Fig. 6. Three-dimensional spring

As shown in Fig. 2, GOVE can develop pipelines that branch to produce multiple layouts of the same Combinatorial Model. The same principle can be applied to any stage of the pipeline. Fig. 7 shows a pipeline that visualises the naming scope and procedure calls in a pseudo-pascal source file.

In addition to the spring algorithms shown in Figs. 5 and 6, GOVE currently supports a three-dimensional DAG algorithm [2] shown in Fig. 8.

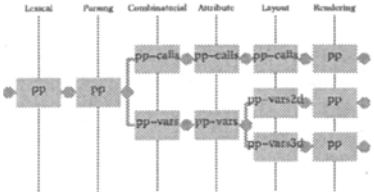


Fig. 7. Pseudo-pascal pipeline

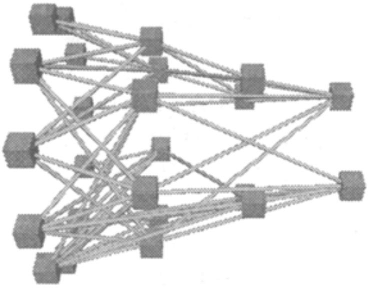


Fig. 8. Three-dimensional DAG

4 Future Improvements

GOVE is currently undergoing redevelopment. Some of the possible improvements include: the use of *diagrammatic* interfaces; interpreted modules to speedup development; more specific/efficient Combinatorial Models; and many ***useful*** layouts algorithms. Allowing interaction with the resulting visualisations is also desirable — simply manipulating the output is easy, while supporting feedback to the information source is much more difficult.

References

1. G. Di Battista, P. Eades, R. Tamassia and I.G. Tollis, "Algorithms for Drawing Graphs: An Annotated Bibliography". Jun, 1994. `ftp://wilma.cd.brown.edu/pub/papers/compgeo/gdbiblio.tex.Z`
2. R.F. Cohen, D. Fogarty, P. Murphy and D.I. Ostry, "Animated Three-Dimensional Information Visualizations". Submitted to ACSC '96. Aug, 1995.
3. H. Ehrig, H.-J. Kreowski, G. Rozenberg, *Graph Grammars and their Applications to Computer Science.* Lect. Notes in Comp. Sc., vol. 532, Springer-Verlag. 1991.
4. T. Kamada, *Visualizing Abstract Objects and Relations: A Constraint Based Approach.* Series in Comp. Sc., vol. 5. 1989.
5. J. Mackinlay, "Automating the Design of Graphical Presentations of Relational Information" in *ACM Trans. Graphics*, vol. 5, no. 2, pp. 110–141. 1986.
6. T. Mason and D. Brown, *Lex and Yacc*, A Nutshell Handbook. 1990.
7. A. Scott and R. Webber, "Grammar Oriented Visualisation Environment (GOVE)". Hons. Proj., Dep. Comp. Sc., Uni. Newcastle, Callaghan 2308, Australia. Nov, 1993.
8. A. Scott, "A Survey of Graph Drawing Systems". Tech. Report 95-1, Dep. Comp. Sc., Uni. Newcastle, Callaghan 2308, Australia. Dec, 1994.