MFTV: A Zoomable Multifaceted Tree Viewer

Nizar Habash
Computer Science Department
University of Maryland
College Park MD 20740 USA
+1 301 405 6746
habash@cs.umd.edu

Jin Tong
Computer Science Department
University of Maryland
College Park MD 20740 USA
+1 301 405 6746
jintong@cs.umd.edu

ABSTRACT
In this paper, we describe a zoomable visualization tool for large-scale hierarchical ambiguous data structures. The tool uses different techniques such as a multifaceted tree representation, semantic zooming, detail and overview windows, subtree hiding, a space-efficient layout scheme, and a search ability.

Keywords
Zoomable User Interface (ZUI), Multifaceted Tree, Semantic Zooming, Subtree Hiding, JAZZ, Linguistics, Natural Language Representation, Lexical Conceptual Structures.

INTRODUCTION

Natural language processing tasks, such as Machine Translation or Information Retrieval require the use of complex representations that can capture the semantics of real-world sentences. Such representations tend to be large-scale ambiguous hierarchical structures that are very hard to visualize for several interrelated reasons. Most important of all is that ambiguity is represented in these structures by explicit repetition. This increases the size of the tree. A second issue is the amount of information that is recorded at every node. It is rather large to account for semantic, syntactic and lexical features. These different pieces of information can clutter any visualization if given equal significance, which is not the case since some of the information is usually more relevant than others. Finally, the size of these representations, although partly dependent on the level of ambiguity and detail, can be enormous. In this paper, we describe a visualization tool that we call the Multifaceted Tree Viewer (MFTV). It attempts to address these issues by utilizing different techniques such as a multifaceted tree representation, semantic zooming, detail and overview windows, subtree hiding, a space-efficient layout scheme, and a search ability. This project was implemented using JAZZ, a Zoomable User Interface (ZUI) API that allows developers to quickly and easily build ZUI applications in Java2. JAZZ is the current generation of the ZUI APIs that started from Pad and Pad++ [2][3][7].

MOTIVATION

The need for such a visualization tool came out of work at the Computational Linguistics and Information Processing Lab (CLIP) at the University of Maryland College Park. Specifically, different groups (analysis, generation, lexicon, etc.) working on a Chinese-English interlingual machine translation system needed a tool to debug the interlingua that was used to represent the semantics of the translated sentences and to debug the group modules themselves. This interlingua is called Composed Lexical Conceptual Structure (CLCS) [4]. CLCSes have been used as the interlingual representation for many languages such as English, Arabic, Spanish, and Korean in different Natural Language Processing projects from Machine Translation to Foreign Language Tutoring. CLCSes share with other natural language representations all of the visualization issues described above. The only visualization currently available is a purely textual representation (Lisp format). For an average sentence of length 36 words, the CLCS is about 700 lines long. Ambiguity is explicitly represented by duplication. This makes visualizing an unambiguous instance of the CLCS very hard. Also, all of the information is displayed with equal significance on the screen, which is not true of the data itself. Figure 1 shows part of the current textual visualization of the following sentence:

In the 21st Session of the South-East Asia-Singapore-Macao Symposium, vice president of the People's Bank of China, Yin Jieyan issued an idea about the situation of large capital inflow and macroscopic economic policy.

SOLUTION

Background

In the beginning of our work, there was no clear visualization solution to deal with the data structure. The visualization needs of CLCSes are quite different from other large-scale hierarchical data visualized by other researchers using cone trees, zoomable hierarchically clustered networks, etc. [5][8][9]. We interviewed some of the people working on the machine translation project mentioned above and discussed with them the needs and limitations of the visualization they were using. Different solutions were explored to visualize the data structure.

Tree diagrams have been used in different textbooks and articles on CLCSes but never including ambiguity. Initially, three ideas presented themselves as important elements for a visualization solution to the issues described above: multifaceted trees, detail+overview windows, and semantic zooming.

```
(:POSSIBLES 0
 (:ROOT NIL EVENT CAUSE NIL NIL 1314636
  ((VOICE ACTIVE) (CAT V) (GLOSS EXPRESS) (ROOT ·¢±í)
   (LEVIN 26.4))
  (:SUB NIL THING YIN_JIEYAN+ NIL NIL 1315184
   ((THETA AG) (CASE ACC GEN DAT NOM) (CAT N)
    (GLOSS YIN JIEYAN) (ROOT Òó½éÑ×))
   (:MOD NIL THING PEOPLE_S_BANK_OF_CHINA+ NIL NIL 1207776
    ((CASE ACC GEN DAT NOM) (CAT N)
     (GLOSS PEOPLE 'S BANK OF CHINA) (ROOT ÖÐ¹úÈËÃñÒøÐÐ)))
   (:MOD NIL PROPERTY VICE+ED NIL NIL 1208324
    ((CASE ACC GEN DAT NOM) (CAT ADJ) (GLOSS VICE)
     (ROOT ¸±)))
   (:MOD NIL THING PRESIDENT+ NIL NIL 1208872
    ((CASE ACC GEN DAT NOM) (CAT N) (GLOSS PRESIDENT)
     (ROOT ÐÐ³¤))))
                          ⋮
                  (690 lines)
```

Figure 1. List Format of one CLCS

The multifaceted tree data structure provides a paradigm to deal with ambiguity and reduce the visible size of the CLCSes to only unambiguous instances of it. A detail window is used to view the different details in every node. Thus reducing clutter and only displaying the most important information in the overview display. Zooming and semantic zooming are needed to provide detail-in-context, compensating for the problems of using a detail window separate from the overview.

Basic Solution Elements

Multifaceted Trees

Ambiguous branches in CLCSes are represented using a special node called a *possibles* node that signifies that there is a disjunction in the tree at the node's position and that the elements of the disjunction are the children of the *possibles* node. Nodes and branches that are not ambiguous are represented like any regular tree. In Figure 2, the black node is a *possibles* node. The tree to the left corresponds to an ambiguous CLCS and the trees to the right correspond to unambiguous instances of it. We decided to normalize this "ambiguity branching" over the whole tree thus abstracting to a tree where every node is a disjunction of faces and every face has its own children that are nodes themselves. The result is a multifaceted tree (Figure 3). It is a generalized tree since a regular tree is a special case of a multifaceted tree where every node has one face only. It is a generic data structure since is not specific to our CLCS data structure.

To visualize a multifaceted tree node, we use a box that displays the content of a single face only and different buttons allowing access to all faces available (Figure 4). Presenting a single face per node at a time reduces the clutter of ambiguity. The selected face is marked by a gray button. When the user clicks on a face button, the tree is reconfigured to show the new face and its children. In Figure 4, the two buttons to the left are different faces of a single node. The button to the right is the fourth face of a node with four faces.

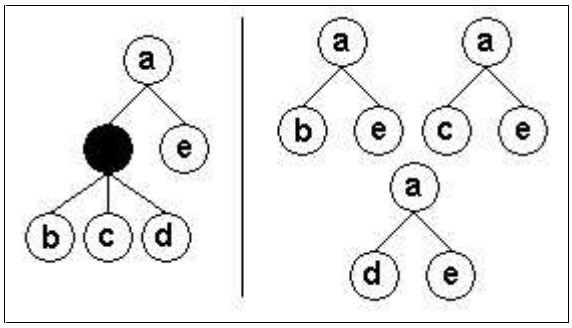

Figure 2. *Possibles* Node

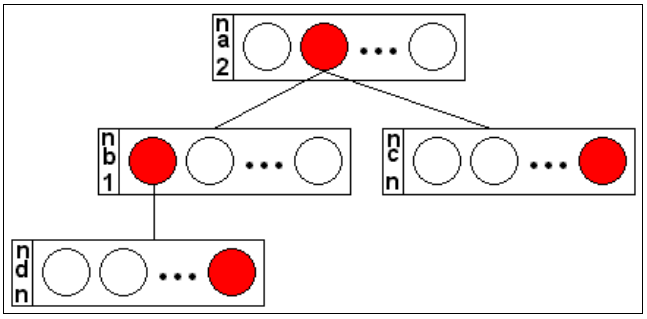

Figure 3. Multifaceted Tree Data Structure

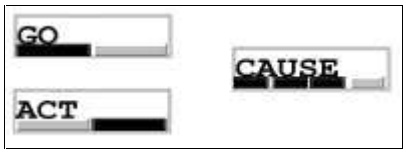

Figure 4. Nodes in MFTV

Detail Window

The detail window is provided to hide less significant information in the main view window (Figure 5). When a user left-clicks on a node box, the node is selected. A red box is drawn around the selected node. The contents of the selected node, which include all the information available on it in the CLCS, are displayed in the detail window. The user can make the information about the tree as a whole to be displayed in the detail window. Such information includes the English translation or statistical information on the number of objects in the tree. This is done by right-clicking on the background while holding the shift button (so as not to interfere with the panning mechanism

described in the next section). This action also causes the de-selection of any selected tree nodes.

Zooming and Semantic Zooming

Zooming has been used in many applications and has been shown to be an effective tool in navigating large-scale data [1][5][9]. The advantages of using a zoomable interface for visualizing CLCSes include the ease of navigation and providing the user with the ability to view the trees at any non-discrete scale. In addition, the use of semantic zooming complements the restrictions of the detail window by providing the ability to view details of a node in context of other nodes. Figure 6 shows a CLCS subtree at three different zoom scales. The ability to zoom and pan was provided by the JAZZ API. Zooming is handled by right-dragging the tree and panning is handled by left-dragging it.

Additional Solution Elements

The basic features above solved a large part of the visualization problems. However some of the old problems are still not solved. For example, large text size consumes too much screen real estate and hiding information under different faces made search harder. Also, using zooming introduces the problem of desert fog. So we added several other features to our system to deal with these issues.

Tree Layout

Our initial tree layout algorithm gave every subtree its own rectangular bound that does not intersect with others. This turned out to be quite wasteful of screen real estate (Figure 7). The solution was to implement a more complicated algorithm that kept track of the borders of sibling subtrees and tried to position them such that the space between them is minimized without causing intersections. The results were very encouraging (Figure 8).

Subtree Hiding

Subtree hiding is another enhancement feature to save screen real estate. We implemented this feature such that when a user right-clicks on a node while holding the shift button, the whole subtree under this node is "collapsed" into it, thus hiding the subtree. A node with collapsed children is distinguishable from other nodes by a grayish background that highlights it. Subtree hiding is especially useful for hierarchical data, where parent nodes often carry enough information for the whole subtree. With subtree hiding, we can show an overview of the whole tree by showing a few top-level nodes. This feature also proved very effective for displaying CLCSes that tend to have repeated subtrees. This "redundancy" can be avoided in the visualization if all repeated subtrees were hidden except for one. Thus users will not be daunted by the number of repetitions of the same subtrees and the overall size of the tree. Figures 9 and 10 showed the same CLCS with and without hiding.

Search Ability

A large amount of data in our visualization is hidden. It is either in faces other than the chosen ones or being collapsed in ancestor nodes. This results in a need for a way to locate nodes anywhere in the tree. Our query menu option allows users to search the tree on a specific string. All nodes matching the query are enclosed in blue boxes. If a node matched but was in a face other than the current one, the button that allows going to that face is highlighted with blue too. Also, if a subtree was collapsed and something within it matched the query, the collapsed node is enclosed in a blue box. Figure 11 displays the tree from Figure 8 after a search was done on the string *NIL*.

Camera Restoration

The use of zooming introduces its own visualization problems. One such problem is desert fog, or the situation when users are lost after having zoomed too far in or out that their immediate environment is devoid of navigational cues [6]. Our simple solution for this was to provide a "Restore Camera" menu option that returns the tree to its default zoom scale and location.

USABILITY TESTING

We did not conduct a formal usability test of this visualization tool. However, every one who was interviewed initially to help with the design issues was very enthusiastic and appreciative of MFTV. Some members of the CLIP lab have expressed interest in extending our tool to visualize other linguistic data structures that they deal with.

SCALABILITY

Because of the dense tree data structure inherited in multi-faceted trees, MFTV can handle trees with a large number of nodes. In the CLCS data set we used, the average number of objects (nodes and faces) of a tree is 362 with a range from 16 to 2307. Coupled with subtree hiding, MFTV can accommodate significantly large trees.

SOFTWARE ARCHITECTURE

The overall architecture of MFTV is described in Figure 12. The application consists of two distinct parts: the LISP writer and the JAZZ visualization application. The interface between the two parts consists of a textual representation of the multifaceted tree to be visualized.

Lisp Writer

The LISP writer is a program that converts CLCS data into multifaceted tree data structures and writes the results to a file in a JAZZ parseable format.

JAZZ Visualization Application

This part constitutes the full MFTV application. When users open files created by the Lisp writer, the multifaceted tree is parsed and constructed. Then a JAZZ scene graph extended with our own visual components is created and rendered. When the user selects a face or hides a subtree, the multifaceted tree data structure is modified to track these choices and a new scene graph is constructed. Semantic zooming is handled by our visual components together with JAZZ.

FUTURE WORK

Animation

With the change-of-face mechanism, the user is allowed access to more data than what a screen area can usually hold. However, the change of an entire subtree (sometimes the whole tree when the root is multifaceted) can be confusing to the user if it happens all of a sudden. It would be nicer and less disorienting if the change of face is animated so that the user will see a more vivid trace of change.

Visual Design Enhancement

We also plan to explore enhancements to the visual design of the tree. This includes possible changes to color-coding, better text layout in each node and other changes of label look-and-feel.

Exploring Visualizing Other Tree Data Structures

The key features of MFTV should also be applicable to other types of multifaceted tree data structures. There are data structures used in NLP that are similar to CLCSes. We can visualize these structures with slight modifications to MFTV. We are also interested in exploring other data structures with similar encoded ambiguities and how they can be properly visualized using the MFTV framework. Moreover, we are open to exploring other uses of the change-of-face and subtree hiding features in a multifaceted tree visualization. One possibility is using the data structure to provide a generalized version of the subtree hiding mechanism. For example, allowing only a subset of the subtrees to be active (visually or functionally) at one time. Through a proper user interface, this data structure can be exploited to present any hierarchical data, especially large and dense trees.

ACKNOWLEDGMENTS

We thank Dr. Ben Bederson for all his help and support at every point in the development of this project. Also, we thank the members of the Computational Linguistics and Information Processing Lab and fellow students in *CMSC 838 B: Zoomable User Interfaces* for their great help and advice.

REFERENCES

1. Bartram, L., Henigman, F., & Dill, J. (1995). Intelligent Zoom As Metaphor and Navigation Tool in a Multi-Screen Interface for Network Control Systems. *Proceedings of IEEE International Conference on Systems, Man and Cybernetics IEEE*, pp. 3122-3127.
2. Bederson, B. B., Hollan, J. D., Perlin, K., Meyer, J., Bacon, D., & Furnas, G. W. (1996). Pad++: A Zoomable Graphical Sketchpad for Exploring Alternate Interface Physics. *Journal of Visual Languages and Computing*, 7, 3-31.
3. Bederson, B. B., McAlister B. (1999). JAZZ: A Toolkit for Object-Oriented 2D Graphics in Java (with zooming). *Submitted to CHI 99.*
4. Dorr, Bonnie. *Machine Translation: A View from the Lexicon*. MIT Press: Cambridge, 1993.
5. Hightower, R. R., Ring, L., Helfman, J., Bederson, B. B., & Hollan, J.D. (1998). Graphical Multiscale Web Histories: A Study of PadPrints. *In Proceedings of ACM Conference on Hypertext (Hypertext 98)* ACM Press, pp. 58-65.
6. Jul, S., & Furnas, G. W. (1998). Critical Zones in Desert Fog: Aids to Multiscale Navigation. *In Proceedings of User Interface and Software Technology (UIST 98)* ACM Press, pp. 97-106.
7. Perlin, K., & Fox, D. (1993). Pad: An Alternative Approach to the Computer Interface. *In Proceedings of Computer Graphics (SIGGRAPH 93)*. New York, NY: ACM Press, pp. 57-64.
8. Robertson, G. G., Mackinlay, J. D., & Card, S. K. (1991). Cone Trees: Animated 3D Visualizations of Hierarchical Information. *In Proceedings of Human Factors in Computing Systems (CHI 91).* ACM Press, pp. 189-194.
9. Schaffer, D., Zuo, Z., Greenberg, S., Bartram, L., Dill, J., Dubs, S., & Roseman, M. (1996). Navigating Hierarchically Clustered Networks through Fisheye and Full-Zoom Methods. *ACM Transactions on Computer-Human Interaction*, 3(2), 162-188.

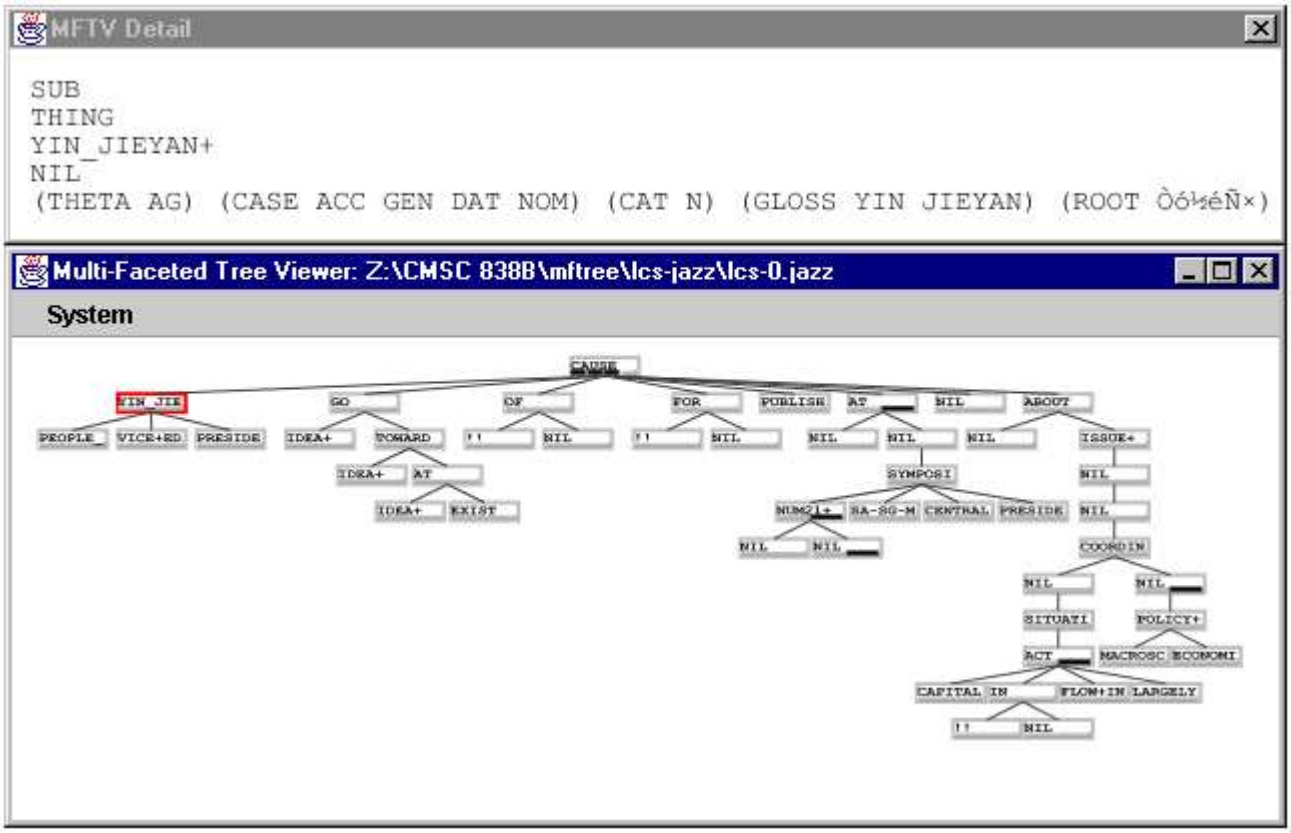

Figure 5. Overview + Detail

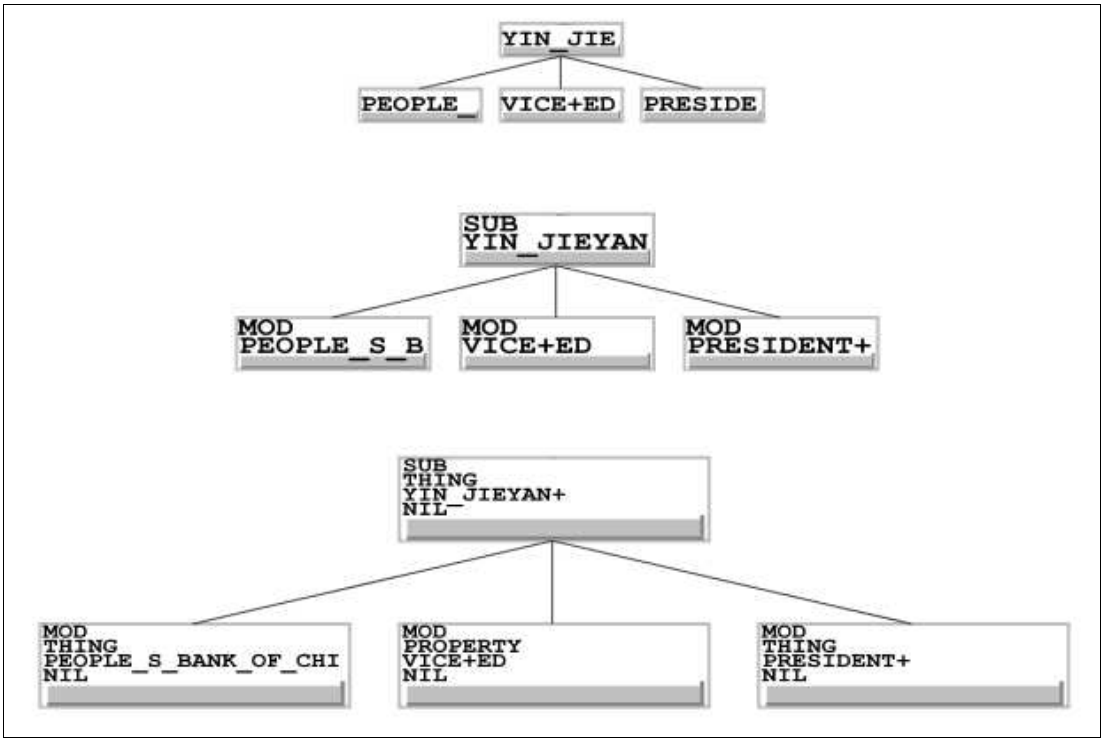

Figure 6. Semantic Zooming

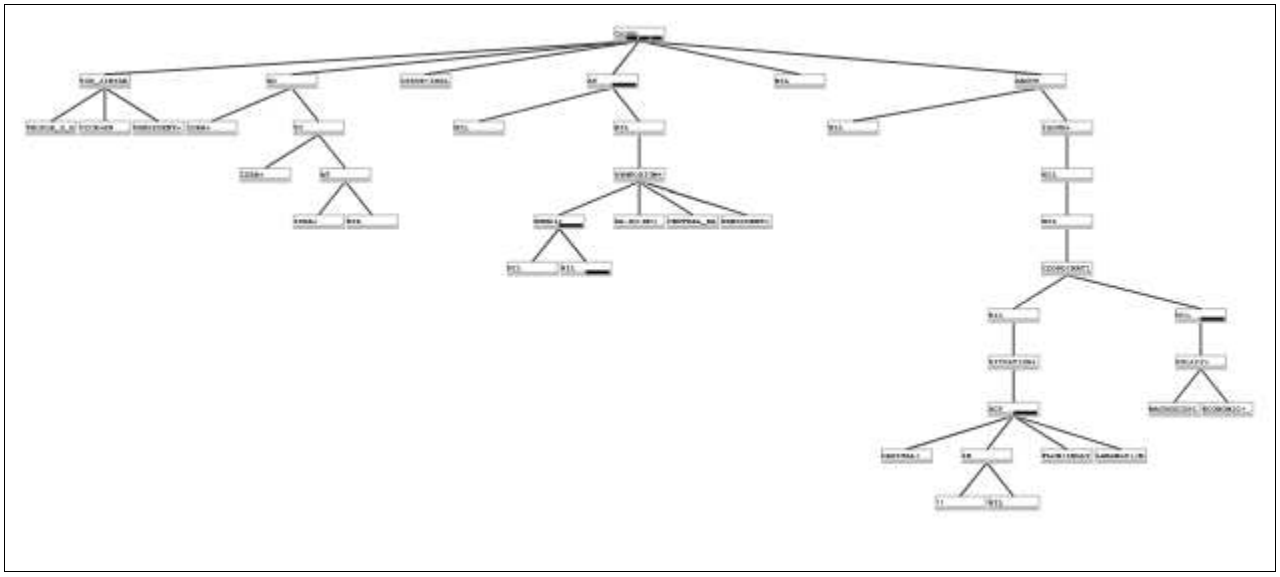

Figure 7. Old Tree Layout

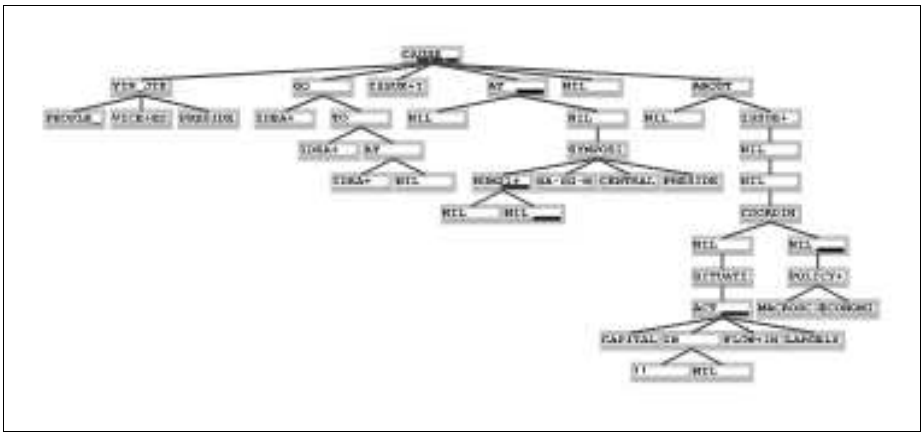

Figure 8. Efficient Tree Layout

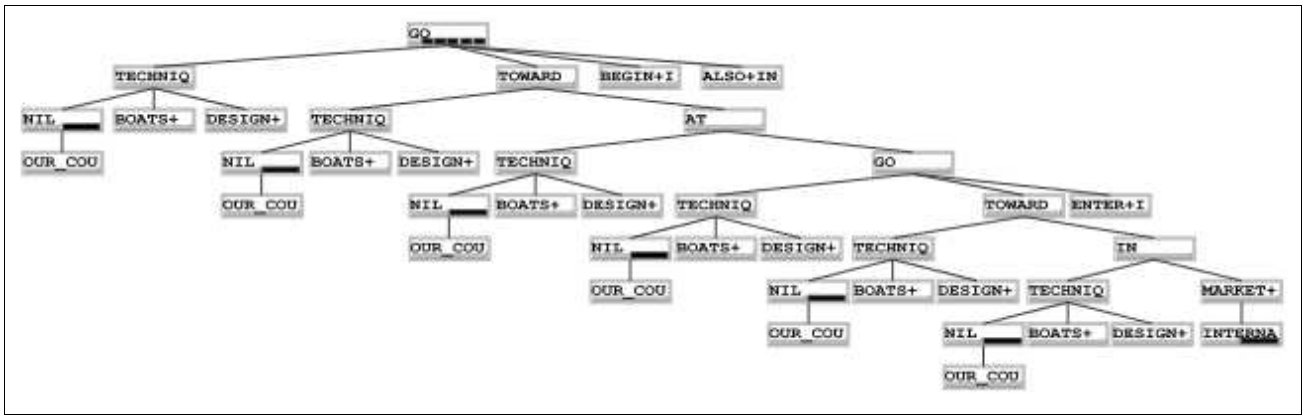

Figure 9. CLCS without subtree hiding

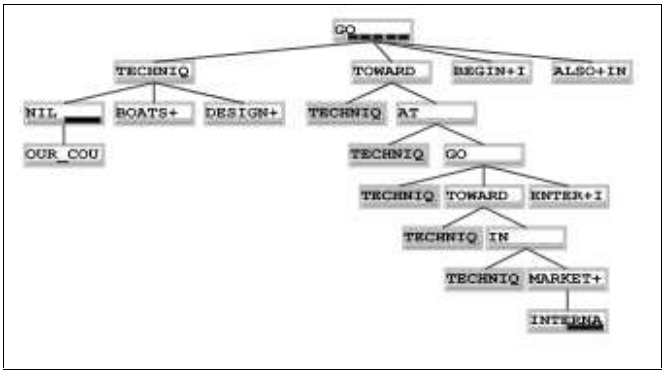

Figure 10. CLCS with subtree hiding

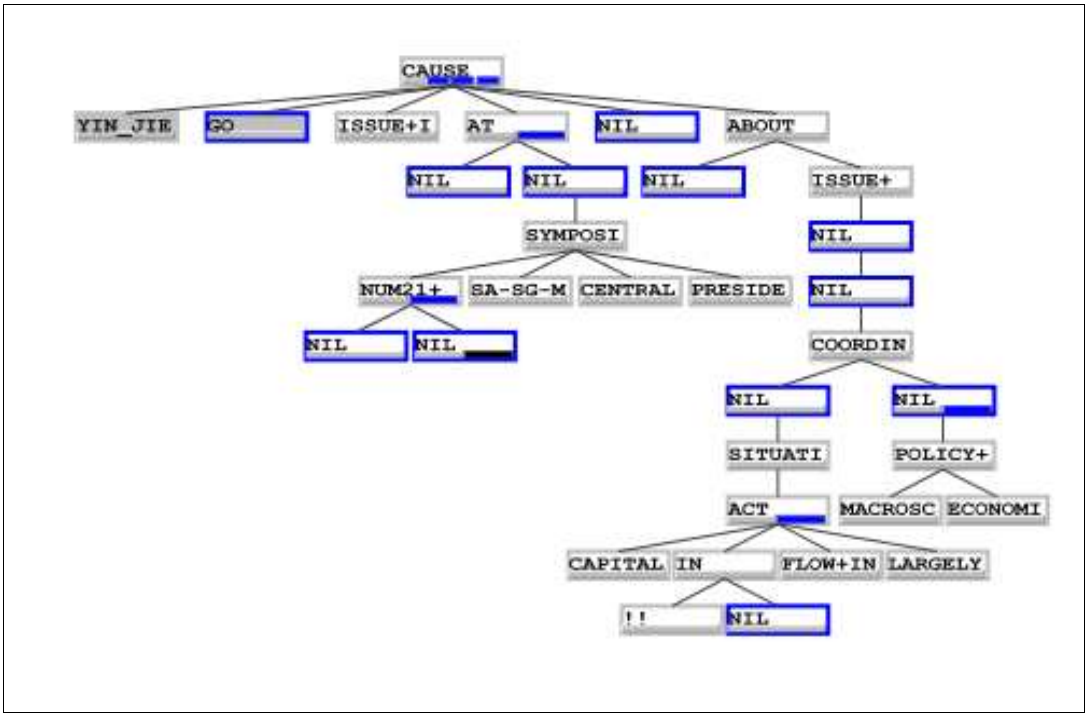

Figure 11. MFTV after a search is done.

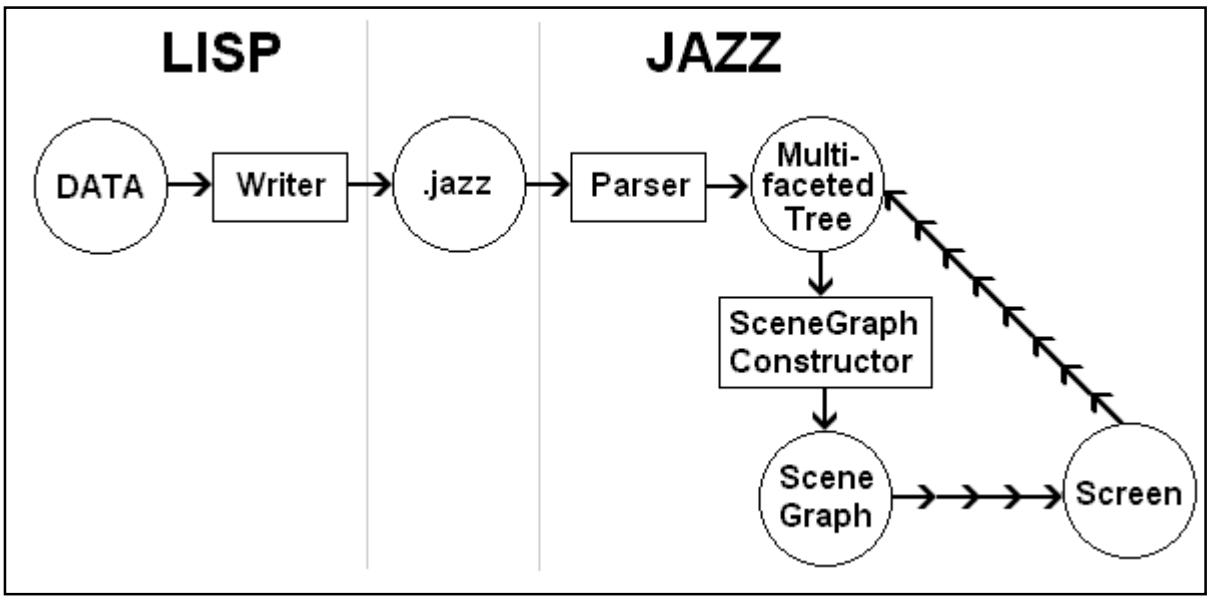

Figure 12. Software Architecture