# Interarbora and Thistle — delivering linguistic structure by the Internet

**Jo Calder**

Language Technology Group
Division of Informatics
University of Edinburgh
J.Calder@ed.ac.uk

**Abstract**

I describe an Internet service *Interarbora*, which facilitates the visualization of tree structures. The service is built on top of a general purpose editor *Thistle*, which allows the editing of diagrams and the generation of print format representations.

## 1. Introduction

As linguistic resources grow in volume and internal complexity, so demand for novel ways of interacting with those resources increases. We describe a suite of tools which are intended to provide general mechanisms for the visualization, creation and editing of hierarchically structured data. One part of this suite is "Interarbora",[1] a Web-based system for converting labelled bracketings to standard print-like representations of trees for display by a Web browser or incorporation in printed documents.

This facility is built on top of a much more general system for the creation and manipulation of structured data, namely a parameterizable editor for structured data called "Thistle".

In this paper, I describe first the surfaces of Interarbora, as seen by the user. I then indicate how Thistle is deployed to support this service, and briefly mention current developments. I summarize the system's current usage, limitations and some possible future directions.

## 2. Interarbora

### 2.1. Surfaces

Interarbora presents two surfaces to the user. The first is a Web page incorporating an HTML form. The form provides an area in which a user may type (or paste in) appropriate data for processing.[2] One example of user input is shown in Figure 1.[3] Interarbora delivers the display as shown in Figure 2. That figure was generated by submitting the above example to Interarbora and, on receipt of the tree, requesting its PostScript representation, which was then included in this document by standard mechanisms. The appearance of the tree as delivered within a Web browser is (for all practical purposes) identical to that shown in the figure.

```
[S [NP LREC]
   [VP [VP [Vcop is]
           [PP [P in]
               [NP [PN Athens]]]]
       [TAdv [NP [Det this]
                 [N year]]]]]
```

Figure 1: Input supplied by the user

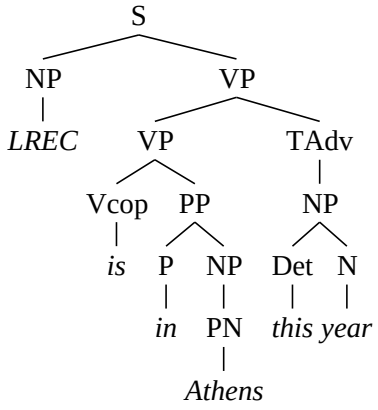


Figure 2: A tree generated from the example in Figure 1.

### 2.2. Architecture

The core of Interarbora's processing uses a server-side CGI program, which processes user input and generates a representation appropriate for display by a Java applet in the user's browser. The applet is an instance of the general display engine from Thistle, specialized for the class of diagrams based on trees whose nodes contain only symbols.

The CGI program is in Perl and uses standard utilities for the initial decoding of user input. At that point, the input is analysed by a simple parser — in essence, a push-down automata — which verifies that the input represents a balanced structure and generates the format required by Thistle. The core program is extremely simple, and the bulk of the work is in fact in managing server-side files and their relation to Web-delivered material. The format generated is the one used by Thistle applets, and represents a sequence of editing steps required to produce the desired diagram.[4]

[1] `http://www.ltg.ed.ac.uk/~jo/interarbora/`

[2] The user may also supply data by file upload.

[3] The whitespace and indentation used here are intended merely to better convey the structure to the human reader. Interarbora does not require any such layout, except that needed to distinguish tokens in the input.

[4] This is Thistle's "dynamic" format, intended to represent di-

In order to work within the Web's standard security models, these operations are all performed on the server that hosts the Interarbora system. An advantage of this organization is that the browser in use handles the acquisition and installation of the application software. Additionally, a further minor advantage is that once a URL has been constructed for a particular diagram, it can be reused as is, and so provides a static, flexible means of referring to a diagram with known content and presentation.

The final stage in processing is to generate a page of HTML, containing instructions to include the Thistle applet, and providing a pointer to the generated representation of the diagram. The page can be configured in a variety of ways, including

- whether the recipient of the diagram should be allowed to invoke the Thistle editor on the displayed structure,
- the overall amount of space to devote to the diagram,
- whether to produce a version of the diagram with no accompanying text.[5]

### 2.3. Input formats

A number of ways of representing labelled bracketings are to be found in text books, corpora or programs. We attempt to be relatively general in the formats we accept. In general, acceptable formats indicate bracketing by pairs of left and right symbols and labels are detectable as a sequence of non-whitespace characters following a left symbol. Formats which respect these general constraints include the Penn Treebank (Marcus *et al* 1993 , ), and are also found in standard linguistics text books, for example (Radford 1997, p87). Elements from such formats can be copied directly for processing by Interarbora.

One of the advantages of the current setting is, then, that there is a range of resources which operate in very much the terms Interarbora expects. There are, of course, many other diagram types in common use in computational linguistics for which Interarbora-style support is potentially useful. On the other hand, many don't have standard encodings, either in "plain text" or in some other format, which would allow the implementation of a parser to produce appropriate input to Interarbora.

As a final, general comment, Interarbora can be seen as one of a growing number of Web-based format conversion and validation services. Under one particular view, Interarbora consumes partially marked up data, and performs an "up translation", so as to produce a structured representation which is guaranteed to conform to some external expression of well-formedness.

agrams either as they are created or for delivering animations. An alternative static format uses SGML.

[5]Thistle's PostScript generation relies on functionality not supported in some browsers and this option provides an alternative method for generating a print representation. Some users have suggested using screen dumps as a means of exporting print representations, but there is a risk of compromising print quality.

## 3. Thistle

### 3.1. Overview

As mentioned above, Thistle is a parameterizable editor. That is, one supplies Thistle with a characterization of legal diagram types, in terms of permissible hierarchical organization and the layout of each type of diagram, (Calder 1998). (This division corresponds directly to that between 'structure' and 'styling' seen in XML and XSL(T).)[6] In particular, the hierarchical organization of diagrams is described by productions in a context free grammar, while styling is described in terms of graphic primitives for text, shapes and grouping associated with each production. This parameterizes the general purpose editor so that

- only well-formed diagrams can be constructed and
- the styling of particular diagrams can be made conformant to that used in other media, most notably in text book form.

The concern with adherence to typographic conventions used in other media is something that sets our approach apart from related work, such as syntax-directed editing and general "visual programming", for example (Viehstaedt and Minas 1995). It is our sense that the existence of representation formats which are consistent across a range of media is particularly important when one's goal is to educate about basic concepts (Calder 1999). Results presented by (Cox *et al* 1999, ) suggest that this form of representation is a powerful addition to teaching methods.

Example diagram classes have been constructed for a wide range of linguistic theories, including Head-Driven Phrase Structure Grammar (Pollard and Sag 1994), Discourse Representation Theory (Kamp and Reyle 1993) and Rhetorical Structure Theory (Mann and Thompson 1987). Demonstrations of the system, including the above functionalities, are available via

http://www.ltg.ed.ac.uk/software/thistle/demos/index.html

### 3.2. Advantages

One of Thistle's advantages is that, from the perspective of the user, interaction is uniform, regardless of the actual diagram class in use. This is because there are really just two basic operations: 'selecting' a part of the diagram, and choosing amongst the legal possibilities for that location in structure. Other obvious functionality is also available, such as cutting and pasting. Our current implementation also does grammatical inference over the structure of diagrams to determine more complex operations. These operations are equivalent in effect to adjunction, as seen in Tree Adjoining Grammars (Joshi and Vijay-Shankar 1985).

There are many advantages to the design we have developed. First, as we use a general editor and parameterize this with a grammar describing the diagrams that are of interest to us, if we choose to enhance our diagrams in some way, only the grammar needs to change, and no change to program code is necessary. Further, if the change really is an addition, existing data is compatible with the enhanced

[6]See `http://www.w3.org/XML/` and `http://www.w3.org/Style/XSL/`, *inter alia*.

class. Furthermore, construction of a diagram class for a new collection of diagrams or annotation scheme is much less costly than the coding required to produce a bespoke editor from scratch.

Second, the use of standard persistence formats such as SGML and XML decreases the amount of work required for interoperability with other systems. General purpose tools can be provided to support this interoperability.

To address the issue raised at the end of the previous section, Thistle provides an answer to the non-existence, in many cases, of standard formats for representing the content of other linguistic diagram types. Given that the cost of producing a grammar for new diagram types is very low, this means that a relatively *ad hoc* design can be used for new diagram types. Such designs will all the same tend to give rise to content whose structure is isomorphic to the data structures required by some theory. Thistle supports this activity by automatically generating *document type definitions* (DTDs) which provide an external characterization of well-formedness, independent of any layout.

### 3.3. Extensions

Current work investigates the extension of the system to the creation and manipulation of graph structures. Many of the existing properties of Thistle are preserved in this enhancement, most notably only minor additions are required in the area of user interactions. This extension is important, given that non-tree like structures are proposed in many annotation schemes, and that standard presentation of such schemes require that non-tree like structure is conveyed directly, rather than through implicit methods such as providing identifiers for parts of structure.

We are also investigating the enforcement of other constraints on well-formed diagrams. Discourse representation theory, for example, contains a notion similar to that of bound variable in systems of first order logic. This constraint cannot be enforced in context free systems, and so there is interest in seeing what mechanisms are required to enforce such constraints for particular diagram types. Finally, for many annotation tasks, while particular schemes may be fundamentally context free in nature, direct expression of such schemes within a context free grammar may be unwieldy. (Examples of this include characterizing the sets of appropriate features for some type in a typed feature logic, selecting a text element from some very large set of choices, and instantiating multiple subparts of a diagram at once. As an example of the latter two situations, one can imagine selecting an item from a lexicon, as a consequence of which a significant amount of structured data to do with the item also needs to be displayed.) We approach this latter problem by providing an interface through which application-specific data may be imported.

## 4. Current use and limitations

Since the introduction of the service in June 1999, Interarbora has delivered more than 2500 trees to over 500 distinct Internet locations. Analysis of logs suggest that there is a variety of user types, from students thinking about the analysis of simple sentences to researchers visualizing the results of automatic processing of texts.

There are a number of minor limitations. Support for different character sets is limited, mainly in the area of PostScript generation. We expect this aspect of the system to be enhanced as the underlying platform stabilizes. There is obvious functionality, as examples the representation of multitoken terminal elements, or subparts of trees which are not assigned analyses, which could be provided, but for which demand has not yet been seen.

## 5. Future directions

As suggested above (and as with any experimental software), there is any number of enhancements which could be made here. At the simpler end of things, one could introduce new diagram types for Interarbora, perhaps for the presentation of Discourse Representation Structures, or rhetorical structure trees, although as mentioned above the lack of consensus standard markup for such types makes one question the usefulness of so doing. More useful and interesting would be the provision of facilities within Thistle to allow hyperlinking from diagram components to other diagrams. This would provide a means of navigating through linguistic resources, whose structure is explicitly exposed, using a standard Web browser.

## 6. Conclusions

We have presented the Interarbora system for the Web-based delivery of structured representations. We have seen that the underlying components are well-suited to supporting this task.

## Acknowledgements

Dafydd Gibbon originally pointed out to me the need for the service Interarbora provides. Nadjet Bouayad-Agha provided the diagram class specification for RST. Richard Tobin programmed the first version of Thistle. Roberto Zamparelli first pointed out the usefulness of having hyperlinks from diagram components. The work reported here was supported in part by: grant TTT *Text Tokenization Tool* GR/L21952 from the Engineering and Physical Science Research Council; grant *The Vicarious Learner* from the Economic and Social Research Council (Cognitive Engineering Programme grant number I. 127251203); a grant to Edinburgh University from Microsoft Corporation.

## 7. References

Calder, J. (1998) How to build a (quite general) linguistic diagram editor. In *Content Visualization and Intermedia Representations*, 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics, Montréal, August 15, 1998.

Calder, J. (1999) 'Diamonds on my Windshield': The Use of Computer-based Instruction in Computational Linguistics. Keynote lecture to the *Workshop on Computer and Internet supported education in language and speech technology*, 8th Conference of the European Chapter of the Association for Computational Linguistics, Bergen, June 12, 1999

Cox, R., McKendree, J., Tobin, R., Lee, J. & Mayes, T. (1999) Vicarious learning from dialogue and discourse: A controlled comparison. *Instructional Science* 27, pp431–458.

Joshi, A. K. and Vijay-Shankar, K. (1985) Some Computational Properties of Tree Adjoining Grammars. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, University of Chicago, July 8–12, 1985, pp 82–93.

Kamp, H & Reyle, U. (1993). *From Discourse to Logic*, Kluwer Academic: Dordrecht and London.

Mann, W. and Thompson, S. A. (1987) Rhetorical Structure Theory: A theory of text organisation. In L. Polanyi (ed), *The Structure of Discourse*, Ablex: Norwood, NJ.

Marcus, M. P., Santorini, B. and Marcinkiewicz, M. A. (1993) Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics* 19, pp313–330.

Pollard, C.& Sag, I.A. (1994). *Head-Driven Phrase Structure Grammar*. CSLI: Stanford and University of Chicago Press: Chicago and London.

Radford, A. (1997). *Syntactic theory and the structure of English: a minimalist approach*, Cambridge University Press: Cambridge and New York.

Viehstaedt, G. & Minas, M. (1995). Generating editors for direct manipulation of diagrams. In B. Blumenthal, J. Gornostaev & C. Unger, editors, *Proc. 5th International Conference on Human-Computer Interaction* (EWHCI'95), Moscow, Russia, LNCS 1015, pp17–25. Springer-Verlag.