Thistle and Interarbora

Jo Calder
University of Edinburgh
Division of Informatics
Language Technology Group
2 Buccleuch Place
Edinburgh Scotland EH8 9LW
J.Calder@ed.ac.uk

Abstract

We present a system for manipulating a wide class of linguistic diagrams, which is configurable and extensible, and allows deployment as a web-delivered system. A major theme of this work is the transfer of the devices of formal grammar into the analysis and construction of diagrams.

1 Introduction

Diagrams play a crucial role in (computational) linguistics, in presenting analyses and characterizing fragments of theories. This role has not to date been adequately supported by programs for the creation, maintenance and delivery of diagrams. We conjecture that this has to do with three main factors. First, in a changing field, obsolescence may be a concern. Second, it may be difficult to see how to provide a uniform interface to an appropriately wide range of kinds of diagrams. Third, integration with delivery systems may be difficult to achieve. We argue below that the design of the Thistle diagram editor provides mechanisms for obviating each of these problems. We start with a brief description of the design of the editor, stressing design decisions that avoid the problems just mentioned. We then turn briefly to some implementation details, before describing and exemplifying the classes of diagrams which have been developed so far. We end with a discussion of current and future directions for this work. All of the examples can be accessed on-line[1].

Some of our practical considerations are worth emphasising. First, we aim for typographic quality as close as possible to standard print presentations of the diagrams in use. The diagrams shown in this paper are presented using the PostScript generated by Thistle. They have essentially the same form as delivered by a web browser. Second, the system should be lightweight in several senses. It should be usable without specialist knowledge of the diagrams in question. The user interface should be simple. It should be deployable with minimal assumptions about the hosting environment. These considerations mean that other programs for manipulating diagrams, such as more general purpose graph editors (for example daVinci[2], DiaGen (Viehstaedt and Minas 1995) or VGJ[3]) are generally unsuitable, as are more complex tools for data annotation, such as the MATE workbench (Dybkjær *at al* 2000). Such systems may of course be able to present more complex diagrams than Thistle, or offer alternative functionality.

Crucial to the simplicity of Thistle is the assumption that many diagram classes of interest can be characterized using only context free methods. As we will demonstrate below, this assumption is consistent with a usefully wide range of classes. We first discuss motivation for the design of Thistle, and describe the grammars that characterize classes of diagram. We then discuss briefly some example classes and the Interarbora service. After giving details of the current implementation and recent enhancements, we describe the settings in which these tools have been exploited. Finally, we describe our current work, and possible strategies for usefully broadening the kinds of diagram that Thistle can describe.

2 Design

Thistle is a parameterizable diagram editor. A class of diagrams is selected by providing Thistle with a grammar which characterizes the diagrams of interest. The grammar describes the hierarchical structure of diagrams, and provides information about layout.

Grammars for diagram classes utilize a particular form of context free grammar, in which there are two kinds of statement. In the first, the left hand side of a rule names a particular type of diagram, and its rewrite describes the abstract structure and concrete layout of a diagram type. In the second, the rewrite is a set of names of other diagram types, representing a disjunctive choice between the latter. Left hand sides are required to be unique throughout. (It is straightforward to show that any context free grammar can be encoded in this form.) Figure 1 shows a fragment of the grammar used to generate the diagram in Figure 2. This fragment can be used to analyse that part of the diagram expressing the value of the feature CONTENT.

[1] http://www.ltg.ed.ac.uk/software/thistle

[2] http://www.informatik.uni-bremen.de/daVinci/

[3] http://www.eng.auburn.edu/csse/research/research_groups/graph_drawing/graph_drawing.html

```
diagram_spec(plain_avm,
             bracket([delimiter(square)],
                     vbox(var(avpairs, [avm_line]))))

diagram_union(avm_line,
              [avpair, path_value])

diagram_spec(avpair,
             array_element([align(baseline)],
                           [var(attribute, attribute),
                            var(value, value)]))

diagram_spec(attribute,
       smallcaps(var(name, Text)))
```

Figure 1: A simplified fragment from the grammar used to generate the diagram in Figure 2

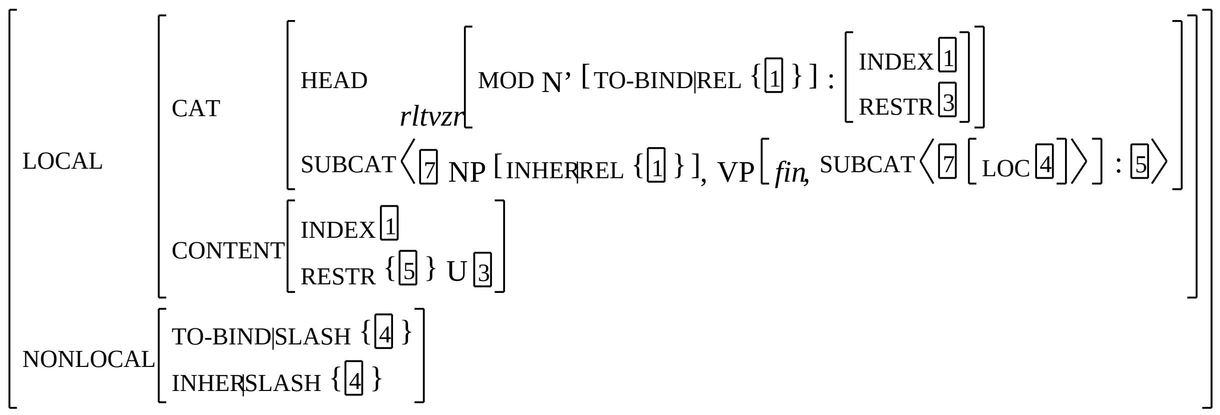
LOCAL
CAT
HEAD
rltvzr
MOD N' [TO-BIND|REL {1}] : INDEX 1 RESTR 3
SUBCAT ⟨7 NP [INHER|REL {1}], VP [fin, SUBCAT ⟨7 [LOC 4]⟩] : 5⟩
CONTENT
INDEX 1
RESTR {5} ∪ 3
NONLOCAL
TO-BIND|SLASH {4}
INHER|SLASH {4}

Figure 2: A diagram, reproduced using Thistle, from Pollard and Sag (1994).

The first and third statements here express the hierarchical structure of and layout of attribute-value matrices. One can gloss the first as: "A diagram of type `plain_avm` consists of any number of diagrams of type `avm_line`.[4] The subdiagrams are arranged vertically and enclosed by a pair of square brackets." In other words, `var` elements stand for a variable subpart of a diagram and indicate the type of diagram that can appear at that location. Note that such elements also assign a label to each variable subpart. The second statement above indicates that a diagram of type `avm_line` can be realized as either of the named types. The fourth statement indicates how diagram types may introduce sequences of characters.

This form of CFG leads directly to a user interface based on top-down rewriting,[5] where a rule of the first kind is invoked, leading to choices in the diagrams introduced as subparts, and so on. In practical terms, then, given a class of diagrams, a particular instance may be constructed by selecting a location in a diagram, and choosing among the possible types of diagram for that location. What the user sees on the surface is a WYSIWYG presentation of the consequences of the particular arrangements of diagram types.

These aspects of the design address at once problems of obsolescence and of providing a uniform user interface. In order to provide a new class of diagram, one has only to construct a grammar for that class, providing the class is amenable to context free treatment (see §6 below). We make use of existing standards in tackling the problem of integrating with other systems. Any instance of the editor may be used via a web browser, so that local installation of software is not essential. The graphical presentation of a diagram may be saved in PostScript, while the logical content of a diagram is stored as SGML.[6] The precise format of a diagram's logical content exploits the fact that each variable subpart of a diagram is assigned a unique name.

In addition to the construction of static diagrams, This-

[4]The square brackets in `[avm_line]` are an *ad hoc* way of expressing the Kleene star.

[5]Other ways of constructing a diagram are possible, as discussed in §5 below.

[6]See also §6 below.

tle may also be used to construct step-time sequences of diagrams. A 'diagram player' can be used to step through (or jump between) diagrams in the sequence. One example shows the states visited by a top-down backtracking parser, on some input and with respect to a given grammar.

3 Example diagram classes

There is a wide range of diagram classes currently available, ranging from an essentially complete treatment of the diagrams in Pollard and Sag (1994) (Figure 2), and in Kamp and Reyle (1993) (Figure 3), to small but useful classes for diagrams from particular areas of linguistics, such as metrical trees and categorial derivations. There are also a number of generic diagram classes such as trees with unlimited or fixed branching.

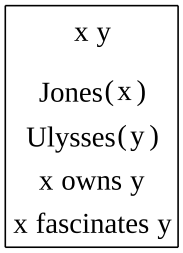

Figure 3: A diagram based on Kamp and Reyle 1993

4 Interarbora

Interarbora[7] is an internet based service allowing the construction and display of tree diagrams via Web browsers. The user supplies a tree specification as a labelled bracketted string, which is then analysed to produce a specification of a Thistle diagram for a simple diagram class. This information is then passed back to the Web browser, which computes a Thistle diagram for display.

The analyser for bracketted strings attempts to be quite liberal. One target format that we handle successfully is that of the Penn Treebank[8]. Figure 3 shows a simple example from Interarbora. As with the other diagrams in this paper, this example is formatted here using Postscript generated by Interarbora. There is no discernable difference between this presentation and that delivered by a web browser. Interarbora is described in more detail by Calder (2000).

5 Current status

The system described above is fully implemented and is available at no charge for non-commercial purposes. As

[7]http://www.ltg.ed.ac.uk/˜jo/interarbora/
[8]http://www.cis.upenn.edu/ treebank/home.html

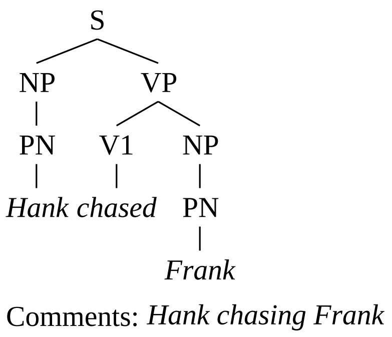

Figure 4: An example tree from Interarbora

our implementation platform is Java, there are relatively few portability issues.[9] In addition to the mode of operation described above, where a user selects a location in a diagram and chooses a type for that location, we have also investigated modes which are not strictly top-down. Such modes are essential in tasks such as annotation, where one has, for example, a given string or text to mark up. In this case, one is interested in adding to the (possibly minimal) existing structure, and this cannot be straightforwardly done under a pure top-down model. Consequently, we have added a range of operations over diagrams, including:

split a sequence of characters is replaced by two (or more) of its subsequences with appropriate structural adjustments

join the inverse of split

demote a diagram is adjoined into the diagram at the current location

promote the diagram at the current location replaces its mother.

There are a number of interesting points to these operations. First, the possibility of such operations is in general determined through grammatical inference. So it is not possible to split a sequence of characters in a location where only one such sequence is allowed by the grammar. Second, the *demote* operation is the exact analog of adjunction in Tree Adjoining Grammars (see e.g. Joshi *et al*, 1991). A demote operation is only allowed if the type of diagram at the current location is permitted within some other diagram type *t* and the type *t* is also permissible at the current location in structure. In general, having selected a location for a demote operation, there may be several ways of executing the operation. For example, the user may be asked to choose which daughter in a finite branching local tree should receive the diagram at the currently selected location. Finally, these operations are not grammar specific, so that

[9]Our implementation predates later versions of Java which provide a tree abstraction, and so our current implementation does not make use of this facility.

the same kinds of operations are available, whether one is dealing with corpus annotation tools or an editor for HPSG diagrams.

6 Current use and on-going and future work

The system is in use in the support of teaching in a variety of settings. Cox *et al* results about the effectiveness of Thistle in teaching concepts to do with phrase structure and category membership. Understanding of these concepts seems to have been improved simply by viewing a video capture of trees being editing. Interarbora is used at several institutions in junior level courses. We have used Thistle as a front end to a variety of rule formats, including those for the tokenization tool TTT (Grover *et al* 2000). The diagram player has been used for the visualization of the results of corpus searches in GSEARCH[10] and of dialogue states, in concert with software developed in the TRINDI project[11].

On-going support work includes changing the persistence format of diagrams from SGML to XML, and bringing diagram classes within the same format. There are a large number of minor improvements we intend to make, including generalizing the Web interfaces so that diagram classes and persistence formats may be supplied by the user.

Our current research has a number of aspects. The limitation to context free diagram classes simplifies many aspects of implementation, most notably in the area of layout. On the other hand, many diagram classes require greater than context free power for their adequate description. Important classes include state transition diagrams, systemic functional networks and autosegmental diagrams. We are looking at compromises which will allow the construction and display of such diagrams while avoiding difficult layout problems.

Another area in which the context free assumption is being examined has to do with diagrams where constraints such as equality are required to hold within a diagram. An example of this is the notion of proper binding in Discourse Representation Theory — a variable occurring as an argument must be appropriately introduced (and *vice versa*). A further example is the enforcement of appropriateness conditions within a typed feature framework. Strictly speaking, this case doesn't violate our context free assumption, but encoding such conditions in a context free way is cumbersome. In these cases, we are interested in looking at ways of further constraining the content of diagrams. One possibility, which sits happily enough with Thistle's background of formal language theory, is to exploit the notion of *path*, a sequence of variable-type pairs. Any Thistle diagram corresponds to a set of such paths, and, because these are generated by a context free grammar, the language of such paths is regular. We could enforce appropriateness in a typed feature setting, for example, by expressing further regular constraints over paths. Using greater than regular power would result in diagrams whose structure was no longer context free.

[10] http://www.hcrc.ed.ac.uk/gsearch/

[11] http://www.ling.gu.se/research/projects/trindi/trindikit.html

Other possibilities include looking at logics to express constraints over diagrams. We can view the set of paths as a model of some logical theory. As our diagrams are necessarily finite, this means that logical frameworks of considerable power could be invoked.

One further element of our work examines ways of providing programmatic control of diagrams, with applications in interactive diagram design, where a cooperating program may fill in details which are logically implied, and debugging of complex representations.

7 Conclusions

We have seen above that Thistle provides a flexible, lightweight interface to a wide variety of diagram types. Furthermore, it can be used for the delivery of diagrams (and sequences of diagrams) in a variety of settings. The Interarbora service provides a way of allowing visualization of tree structures suitable for a wide variety of users.

References

Calder, J. (2000) Interarbora and Thistle: Delivering linguistic structure via the Internet, in *Proceedings of the Second Language Resources and Evaluation Conference*, 31 May–2 June 2000, Athens, Greece.

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.

Dybkjær, L., Møller, M. B., Bernsen, N. O., Grosse, M., Olsen, M. and Schiffrin, A. (2000) Annotating Communication Problems Using the MATE Workbench, in *Proceedings of the Second Language Resources and Evaluation Conference*, 31 May–2 June 2000, Athens, Greece.

Grover, C., Matheson, C., Mikheev, A. and Moens, M., (2000) LT TTT - A Flexible Tokenisation Tool, in *Proceedings of the Second Language Resources and Evaluation Conference*, 31 May–2 June 2000, Athens, Greece.

Joshi, A. K., Vijay-Shanker, K. and Weir, D. J. (1991) The convergence of mildly context-sensitive grammatical formalisms. In P. Sells, S. M. Shieber and T. Wasow (eds.) *Foundational Issues in Natural Language Processing*. MIT Press: Cambridge, MA.

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

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

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.