ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/2845750

Generating Textual Diagrams and Diagrammatic Texts

Conference Paper · October 2002

DOI: 10.1007/3-540-45520-5_2 · Source: CiteSeer

CITATIONS
2

READS
222

2 authors, including:

Donia Scott
University of Sussex
129 PUBLICATIONS **2,006** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Reference Architecture for Generation Systems (RAGS) View project

All content following this page was uploaded by Donia Scott on 10 July 2013.

The user has requested enhancement of the downloaded file.

Generating Textual Diagrams and Diagrammatic Texts

Donia Scott and Richard Power

ITRI, University of Brighton, Brighton, UK
{Donia.Scott,Richard.Power}@itri.brighton.ac.uk

Abstract. There are obvious ways in which text and diagrams within a document should be coordinated: for instance, the placement of a diagram might influence the wording of the text. However, there is a more subtle interaction between text and diagrams, which has emerged from work on generating technical documents that make extensive use of layout. Constituents that would normally be classified as textual may contain diagrammatic features (e.g., when multiple indenting is used); conversely, non-pictorial diagrams usually contain short strings of text (e.g., labels within boxes). We argue that text and diagrams really lie on a continuum, and that for generating documents of this kind we need a descriptive framework that combines linguistic and graphical features in the same representation.

1 Introduction

In many genres it is normal for documents to contain diagrams as well as text. (By 'diagrams' we mean schematic illustrations in which logical relationships are expressed graphically — for instance, tables, networks, or embedded boxes; we are not concerned here with pictures or photographs.) Obviously, any system that automatically produces documents in such genres must go beyond the normal capabilities of Natural Language Generation (NLG) programs, not only by generating diagrams and placing them appropriately, but by adapting the wording of the text (McKeown et al., 1992, André and Rist, 1995, van Deemter and Power, 2000). However, there is a more subtle sense in which text and diagrams have to be coordinated. If we look closely, we find that document parts that would normally be labelled 'text' often employ techniques of graphical organization typical of diagrams; conversely, document parts that would normally be labelled 'diagrams' make essential use of text.

Our interest in this field dates from the DRAFTER[1] and GIST[2] projects (Paris et al., 1995, Power and Cavallotto, 1996), which began in 1993. In both projects, the aim was to develop an NLG system that would allow an author

[1] *A Drafting Assistant for Technical Writers.* EPSRC project J19221, 1993–1996. `http://www.itri.bton.ac.uk/projects/drafter`.

[2] *Generating Instructional Texts.* European Commission project LRE 062-09, 1993–1996. `http://ecate.itc.it:1025/projects/gist.html`.

H. Bunt and R.-J. Beun (Eds.): CMC'98, LNAI 2155, pp. 13–29, 2001.
© Springer-Verlag Berlin Heidelberg 2001

to define the content of instructional texts, which were generated in several languages; in this way, experts on the relevant domains could produce documentation even in languages that they did not know. For DRAFTER the domain was software manuals, specifically for word processors and diary managers; for GIST, the domain was social security forms. During the early stages of these projects we studied wide-ranging examples of actual manuals (Paris and Scott, 1994, Hartley and Paris, 1996) and forms (Scott et al., 1995), and were struck in each case by the prevalence of diagrams and graphical layout, and their close integration with the content and syntax of the text. In software manuals, diagrams of interface objects (icons or buttons) were sometimes inserted directly into sentences as the subject or object of the verb; in social security forms, whole sections of text might be organized as an indented tree representing dependencies among questions (examples will be given later).

Within DRAFTER and GIST there was no provision for addressing these issues, but a lesson drawn from both projects was that in many genres it makes no sense to generate a text without specifying as well how the text should be laid out, and which diagrams (if any) should accompany it. It might seem at first sight that a text could be generated as a punctuated string, the layout and illustrations being added later as a formatting task, but this overlooks the essential contribution of layout and illustrations to meaning. Since 1997 we have been exploring the interaction between wording and layout in the ICONOCLAST project[3] (Bouayad-Agha et al., 2000b); in addition, our colleague Markus Fisher has investigated the automatic generation of diagrams for user interfaces (Fischer, 1998, Fischer, 1999). In both these more recent projects, we used the technique of constraint logic programming, expressing interactions between graphical organisation and wording by means of constraints defined on linguistic and graphical features. By treating generation as a constraint satisfaction problem (Hentenryck, 1989), we were also able to produce multiple solutions, and so to consider the stylistic criteria by which one potential solution might be preferred to another.

In this chapter we describe an approach to document generation that has grown from all these projects. In what follows, we give examples suggesting that there is no sharp divison between text and diagrams; these concepts actually represent vaguely defined stretches along a continuum, and many document genres contain hybrid passages exhibiting a mixture of textual and diagrammatic features. The prevalence of such passages suggests that textual and diagrammatic features should be merged in a common descriptive framework; we will argue that this can be done through a level of representation called *abstract document structure* which has emerged from our work on ICONOCLAST (Bouayad-Agha et al., 2000a), and on the project RAGS[4], which aims at developing a reference architecture for NLG (Mellish et al., 2000). We finally describe

[3] *Integrating constraints on layout and style*, EPSRC Project L77102, 1997–2000. `http://www.itri.bton.ac.uk/projects/iconoclast`.

[4] *A Reference Architecture for Generation Systems*, EPSRC projects GR/L77041 and GR/L77102, 1998–2001. `http://www.itri.bton.ac.uk/projects/rags`.

an application demonstrating the generation of text and diagrams from rules defined in the same representational framework, and mention plans for future work.

2 Text and Diagrams

We will argue here that the distinction between text and diagrams is not as straightforward as it is often conceived to be: few documents are purely textual and most diagrams have a critical textual element (apart from their captions).

2.1 Diagrammatical Features within Texts

It is not simply the case that some texts *contain* diagrams: many texts are *presented* diagrammatically, with their linguistic and graphical elements jointly contributing in rather crucial ways to their meaning. Let us look at some fairly common examples.

Forms. Figure 1 shows a simplified section in the style of the form BR1 for retirement pensions, produced by the Document Design Unit of the British Department of Social Security. It belongs to a series of forms that has won awards for clarity and approachability. Compared with a conventional text, the outstanding feature of the section is its layout, obviously designed to express dependencies among questions. The placement of the second question 'Has your spouse applied for a pension before' shows that it is relevant only if the reply to the first question is 'Yes'. Unmarried users, having answered 'No' to the first question, can see immediately from the layout that *there is no need to read the second and third questions at all*, let alone answer them.

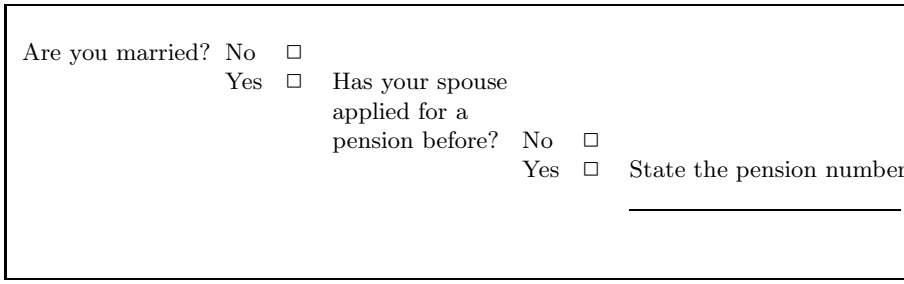

Fig. 1. Part of a Social Security Form (British Version).

To describe the structure of Figure 1 formally, the conventional textual hierarchy of sentence, paragraph, subsection, section, is plainly inadequate. At a superficial level, the section might be regarded as a table, with seven points of vertical alignment; at a deeper level it might be regarded as a binary tree

which the user can navigate by following either the 'Yes' arc or the 'No' arc at each decision point. Layout apart, it also differs from conventional texts through the inclusion of ticking boxes next to the 'Yes' and 'No' answers. Is it a text or a diagram? Rather than refining the definitions of these informal terms, we would prefer to classify it as a hybrid, containing both textual and diagrammatic features.

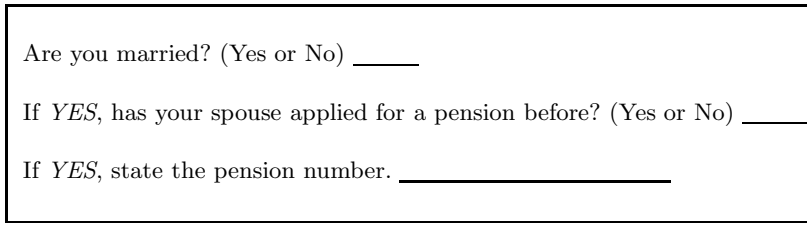

Fig. 2. Part of a Social Security Form (Italian Version).

Interestingly, we came across an Italian version of the form (including English translations), produced by a publisher that for one reason or another preferred a more conventional layout. Figure 2 gives the section of this form corresponding to Figure 1; again the content has been simplified, but the style remains faithful to the original. As can be seen, information previously shown by layout is now shown by additional wording. Instead of representing alternative answers by vertically aligned ticking boxes, the Italian version explicitly states 'Yes or No'. Instead of showing dependencies among questions by a graphical relationship to the ticking boxes, it words all follow-up questions in the conditional form — thus perhaps introducing ambiguities: is the third question conditional on the second question, or perhaps on the first? (Working this out is not too difficult, but we have to read the questions.) Whichever version is preferred, the important point is that each version adapts wording to layout: information expressed graphically in Figure 1 is expressed linguistically in Figure 2.

Tables. We normally think of text as a linear sequence, so that each unit has a single successor. A table, instead, is organized in two dimensions, so that each cell has two successors — the cell on the right if we follow rows, or the cell underneath if we follow columns. Some tables are not text-like: a table of numerical data, for example, would typically serve as an illustration, intended for reference rather than exhaustive reading; such tables are usually given a label and a caption, allowing them to float in relation to the text. However, in instructional documents we have often found tables in which the cells contain paragraphs of text, the columns of the table representing a repeated rhetorical relationship. Figure 3 shows an example from a patient information leaflet explaining how to use an insulin pen (ABPI, 1997pg. 535).

Problem	**Action**
Insulin is not appearing.	The needle may be clogged. Change the needle.
The dose window only shows half a number.	The pen was not reset to ⋆ before you dialled the dose. Reset to ⋆.
You cannot reset to ⋆.	Hold the pen firmly around the white cylinder ...

Fig. 3. Section from a Patient Information Leaflet.

The actual table contains several more entries, and makes up the whole of a section called 'Hints and tips'. It has no label and no caption, and clearly does not float in relation to the surrounding text. In short, apart from its organisation into rows and columns, it behaves like a conventional section of text.

From analysis of a corpus of over 500 patient information leaflets in (ABPI, 1997), we have observed that 'tables of text' are employed most commonly when there is a list of points with a parallel semantic or rhetorical structure. For example, in Figure 3 each point comprises a problem and a remedy, and by using tabular form the author is able to mark this rhetorical relationship just once, in the column headings. Even when a relationship is marked repeatedly, we have seen examples such as Figure 4 in which an informal tabular layout is still used in order to emphasize the parallelism.

Unless advised by your doctor, the maximum dose is

3-4 tablets	for adults,
2 tablets	for children over 12,
1 tablet	for children from 2-12 years, and
half a tablet	for babies and children under 2.

Fig. 4. Informal Tabular Layout.

Icons within Sentences. The '⋆' icon in Figure 3 illustrates the practice of integrating a small picture into a sentence in place of a descriptive noun phrase; the star sign here refers to a marking on the actual insulin pen. We have

also noticed this intrusion of diagrams into syntax in software manuals, where a schematic drawing of a button or icon may be employed instead of a noun phrase like 'the Cancel button'

> Click on **Cancel** if you want to close the dialogue box without performing an action.

2.2 Text within Diagrams

Most diagrams contain strings of text which contribute essentially to the meaning. Entity-relationship diagrams, for example, typically have labels on the nodes and arcs; trees have node labels; tables have alphanumerical characters (words or numbers) within the cells. We have been emphasizing the contribution of layout to the meaning of text, *but the contribution of text to the meaning of diagrams is often far more important.* Strip the labels from the system architecture diagram shown in Figure 5, and the result is virtually meaningless (Figure 6). Similarly, separating the labels from their diagrammatic context may leave some important relationships unexpressed, but in most cases it will be easier to reconstruct the meaning from the text alone than it would be from just the graphics.

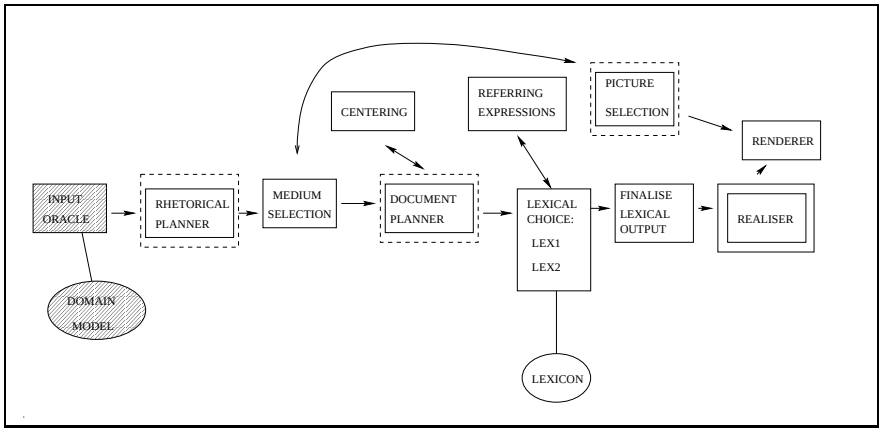

Fig. 5. A Diagram with Text.

3 Abstract Document Structure

The examples that we have discussed suggest the following radical conclusion: *diagrams are texts with rich layout.* This may be an exaggeration, especially for some types of diagram (e.g., ones containing schematic pictorial elements), but it is far nearer the truth than an approach that regards diagrams and text as

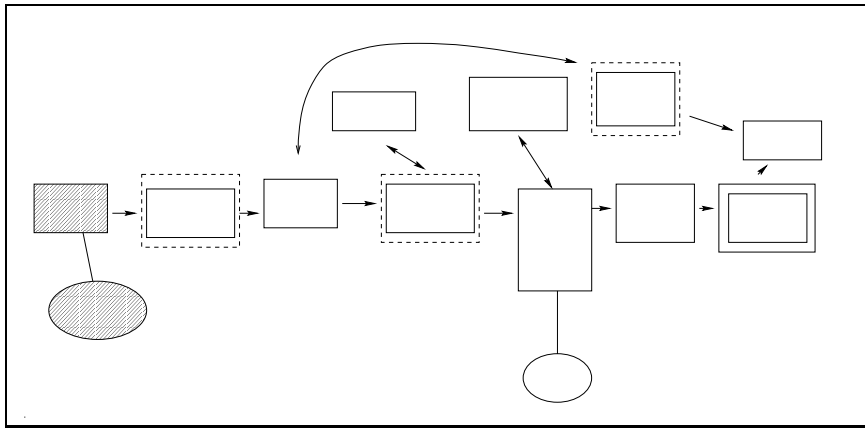

Fig. 6. A Diagram without Text.

distinct presentational forms. Let us adopt this idea provisionally and see how far it can be pushed.

As a foundation, we need a conceptual framework for describing the structure of plain text; with this foundation in place, we can extend the framework so that it covers more and more advanced layout features, leading eventually to diagrams. The most useful starting point, we believe, is the theory of text structure proposed by Nunberg (Nunberg, 1990) in his book 'The Linguistics of Punctuation'. This book introduces two crucial clarifications. First, it distinguishes *text structure*, which is realised by punctuation and layout, from *syntactic structure*. Secondly, it distinguishes *abstract* features of text structure from the *concrete* (or graphical) features by which they are expressed.

The distinction between text structure and syntax can be explained by considering two interpretations of the word 'sentence'. In linguistics, 'sentence' is used mainly as a syntactic category, defined by phrase-structure rules such as $S \rightarrow NP + VP$. However, a sentence can also be viewed as a portion of text starting with a capital letter and ending in a full stop; to distinguish this from the syntactic category, Nunberg calls it a 'text-sentence'. Sometimes the two categories of sentence coincide, but often they do not. Thus in the following passage:

> He entered the office. Disaster. The safe was open and the money had gone.

the first text-sentence is also a syntactic sentence, but the second is merely a noun, while the third comprises two syntactic sentences (or three if we count the whole as well as its parts). Nunberg argues that if we have two kinds of category, then we need *two kinds of grammar*: he calls them the 'lexical' grammar (we prefer 'syntactic') and the text-grammar. In addition to text-sentence, the text-

categories include 'text-clause', 'paragraph', and 'section', and the text-grammar allows us to formulate constituent structure rules such as

$$S_t \rightarrow {C_t}^+$$

meaning that a text-sentence comprises one or more text-clauses.

In introducing the concepts 'text-sentence', 'text-clause', etc., it is convenient to explain them in terms of their realisation in punctuation and layout: thus a text-sentence starts with a capital letter and ends in a full stop; a text-clause ends in a semicolon; a paragraph begins on a new line with a tab. However, this is not strictly correct. In Nunberg's theory, these concepts represent *abstract* structural properties of the text which may be realised differently according to context or convention. In the case of 'paragraph' this distinction is obvious, since we are all familiar with several devices for expressing paragraph boundaries: instead of a new line with a tab, for example, an editor might prefer two new lines (or some other vertical space) with no tab. However, the abstract/concrete distinction also applies to the other text-categories. For example, the passage

> The safe was open; the money had gone.

contains two text-clauses, but the second has no semicolon because its ending coincides with the closure of a larger unit, a text-sentence, which is marked by a full-stop. Similarly, the stop at the end of a text-sentence is often dropped when the sentence is an item in a vertical list, for instance in a sequence of instructions:

> To save the file:
> 1. Open the Save dialogue-box
> 2. Enter the filename
> 3. Click on the Save button

Thus text structure is *realised* by punctuation and layout, but the two are not equivalent.

The relationship between text structure and syntax is a difficult issue, but to simplify a complex story we can assume that syntactic relations hold only within text-clauses. It is easy to find exceptions, for instance in an informal style of writing

> He felt beaten. Battered. And bewildered.

but in most cases this assumption holds. Within text-clauses, then, the structural features guiding interpretation are mainly syntactic; at higher levels, this role is taken over by the text structure.

Since his book is about punctuation, Nunberg focusses on text-categories like text-clause and text-sentence which are realised by marks like semicolon and full-stop. In ICONOCLAST, our aim has been to extend the concept of text grammar so that it covers all significant higher-level structure found in documents, including pictures and diagrams as well as layout patterns like bulleted lists. On this wider

interpretation, the term 'text-structure' becomes misleading, so in the RAGS guidelines the term 'document structure' is preferred.

The significance of abstract document structure in ICONOCLAST (and RAGS) is that it mediates between rhetorical/semantic structure and the details of graphical layout and punctuation. As a simple illustration, consider the concept of 'emphasis'. On a rhetorical/semantic level, this concerns the information structure of the message: perhaps a particular element is focussed, as in the following formula:

$$contrast(likes(john, meat), ordered(john, \overline{fish}))$$

in which focussing is marked by an overline. At a later stage of generation, this message has been realised by an abstract document structure, perhaps comprising two text-clauses, and the focus on the element $fish$ has been realised by an emphasis feature on the word 'fish'. In XML notation, this abstract document structure could be represented as follows:

```
<text-sentence>
  <text-clause>
    John likes meat
  </text-clause>
  <text-clause>
    but he ordered <emphasis>fish</emphasis>
  </text-clause>
</text-sentence>
```

Finally, in the concrete document structure, these features of abstract document structure are realised by punctuation and formatting decisions. Since emphasis can be shown in several ways, the same abstract structure can be realised by a number of different concrete structures, depending on which convention is preferred:

John likes meat; but he ordered *fish*.
John likes meat; but he ordered **fish**.
John likes meat; but he ordered FISH.

Why postulate an intermediate abstract document structure, rather than passing directly from rhetorical structure to detailed layout and punctuation? The answer, we believe, is that through the concept of abstract document structure *we can capture those features of layout and punctuation that interact with meaning, and hence with wording.* Having decided to put focus on $fish$, the first issue that arises in expressing the message is whether to express this focus by wording (e.g., 'what he ordered was fish'), or by emphasis expressed through formatting. At this stage it does not matter whether the emphasis will be marked by italics, bold face, or capital letters: the choice of wording will remain the same no matter which of these alternatives is eventually used.

3.1 Advanced Layout

The implication of our argument is that a common architecture can be used for generating text and diagrams — or at least, some kinds of diagrams. Indeed, we have suggested that such an architecture is not only possible, but desirable, since the 'text' and 'diagrams' really lie on a continuum, with many hybrid forms. Minimally, this common architecture envisages two stages, although these could be further divided:

1. Starting from a rhetorical/semantic input, the wording and abstract document structure of the text/diagram is selected.
2. The output document is fully specified by decisions about punctuation, formatting, and graphical layout, which are guided by the abstract document structure.

In such a system, wording is adapted to layout through *co-occurrence constraints on syntax and abstract document structure*. Thus, if focus is to be expressed through a cleft construction, an 'emphasis' feature in abstract document structure might be ruled out (unless a redundant style was preferred). Or if an entity was to be expressed in abstract document structure by a node in an entity-relationship diagram, its syntactic realisation might take the form of a concise label rather than a full noun phrase (e.g., 'company location' rather than 'the location of the company').

If this approach is viable, the crucial next step is to extend Nunberg's text-grammar so that it encompasses the abstract forms of a wider range of layout patterns. Our first move in this direction has been to admit vertical lists through a new abstract feature called INDENTATION. In Nunberg's text-grammar, the categories form a hierarchy ordered by size: sections contain paragraphs, paragraphs contain text-sentences, text-sentences contain text-clauses, and so forth. Vertical lists complicate this picture because *lower units may contain higher units*, provided that the higher units are indented items. In the following passage, for example, a text-clause contains indented paragraphs (coordinated syntactically with the subordinating conjunction 'since'):

> In rare cases the treatment can be prolonged for another week; however, this is risky since
> – The side-effects are likely to get worse. Some patients have reported severe headache and nausea.
> – Permanent damage to the liver might result.

To formalise this extension, the ICONOCLAST system represents document units by two features called TEXT-LEVEL and INDENTATION. TEXT-LEVELS are represented using the familiar hierarchy from section to text-phrase; INDENTATION is represented by integers in the range $0..N$, so that for example an indented item within an indented item would have INDENTATION=2. For short, we can write for example Sen_0 for an unindented text-sentence and Par_1 for an indented paragraph. Within the same level of indentation, Nunberg's constituent structure rules hold good; thus we can write

$$Par_i \rightarrow Sen_i{}^+$$

meaning that a paragraph with indentation i may comprise one or more text-sentences also with indentation i. This rule (with $i = 1$) exactly describes the first indented item in the above passage. However, the children may also be indented one degree higher than the parent, in which case the usual constraint on TEXT-LEVEL hierarchy no longer applies. Thus we have for example:

$$Phr_i \rightarrow Par_{i+1}{}^+$$

meaning that a text-phrase (the constituent of a text-clause) may comprise one or more paragraphs at a higher indentation; with $i = 0$ this describes the whole indented list.

To specify abstract document structure for tabular layouts is more difficult, since it is unclear what is the right level of abstraction. Our first experiments have been based on the concepts 'row', 'column' and 'cell' that are used for defining tables in HTML and CLIM (Common Lisp Interface Manager); these define the schematic structure of a table, the exact spacing being determined by graphical features (e.g., distance between cells) which can be regarded as part of concrete document structure. Thus an abstract document structure for Figure 1 (the excerpt from a social security form), encoded in XML, might run as in Figure 7.

4 Applications

In the DRAFTER system, which generates instructions for using word processors and diary managers, the user specifies the desired content by interacting with a diagrammatic representation of the instructional procedure. The diagram is made up of a number of embedded boxes (Figure 8), orange boxes representing actions, defined by labels in the upper left, and violet boxes representing methods for achieving them; since a method includes a sequence of sub-actions, these violet boxes will contain smaller orange boxes in their turn.

Originally, we did not regard the production of diagrams like Figure 8 as a NLG problem. The diagram belonged to a user interface for creating and maintaining a knowledge base, from which output texts in natural languages (English and French) would then be generated. However, as pointed out in Section 2, strings of text play a vital role in such diagrams, and we had to implement at least a rudimentary syntax in order to produce action labels like 'Enter document name'.

Thus DRAFTER paradoxically had *two* NLG systems, not one. The knowledge-editing interface was able to produce diagrams (similar to Figure 8) from a knowledge-base in any state of completion; as well as indicating the current state of the knowledge, these diagrams could be manipulated interactively in order to perform editing operations — for instance, by adding further sub-actions to a sequence. In a way, the production of the diagrams was a more impressive application of NLG than the production of the output texts: the syntax

may have been impoverished, but the use of layout was far more complex, the generator worked even when the knowledge was only partly specified, and the generated diagram showed extra information — not only the existing content, but the options for adding further content.

```
<section>
  <row>
    <text-sentence force=question>
      are you married
    </text-sentence>
    <column>
      <row>
        <text-sentence>no</text-sentence>
        <ticking-box/>
      </row>
      <row>
        <row>
          <text-sentence>yes</text-sentence>
         <ticking-box/>
        </row>
        <row>
          <text-sentence force=question>
            has your spouse applied for a pension before
          </text-sentence>
          <column>
            <row>
              <text-sentence>no</text-sentence>
              <ticking-box/>
            </row>
            <row>
              <row>
                <text-sentence>yes</text-sentence>
                <ticking-box/>
              </row>
              <text-sentence>
                state the pension number
              </text-sentence>
            </row>
          </column>
        </row>
      </row>
    </column>
  </row>
</section>
```

Fig. 7. Abstract Document Structure.

From this experience we drew two lessons. Firstly, if we could support knowledge editing through a generated diagram, why not support it through a generated *text*, thus freeing the user from any need to learn new diagrammatic conventions? Secondly, if the tasks of generating text and diagrams had considerable overlap, why not approach them by applying the same methods and resources? The first conclusion led to the development of the WYSIWYM[5] technique of knowledge editing (Power et al., 1998, Scott et al., 1998), which depends on interaction with a *feedback text* which plays the same role as the diagram in Figure 8. The second conclusion led to the attempt, in the ICONOCLAST project, to unify the generation of text and diagrams by treating diagrams as an enhancement of normal textual layout.

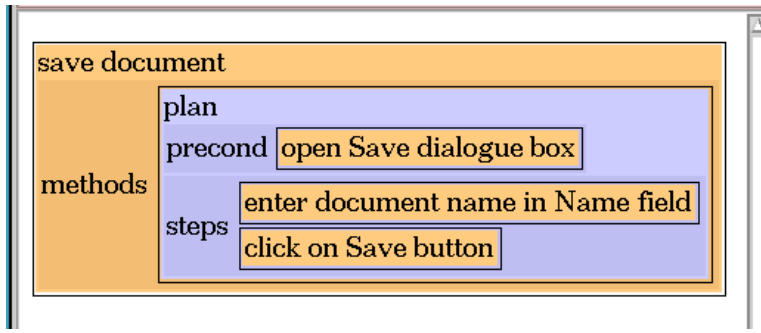

Fig. 8. Diagrammatic Representation of Instructions.

To express the point schematically, the two generators in DRAFTER produced presentations of the knowledge base which differed on two dimensions: *purpose*, and *appearance*:

Purpose. Presentations in the user interface served the purpose of *supporting editing*, by providing feedback on the current state of the knowledge base, along with options for editing. The texts produced by the English and French generators served the purpose of system output, to be incorporated into manuals. For short, we call these purposes *feedback* and *output*.

Appearance. Presentations in the user interface had a highly *diagrammatic* appearance; the output texts had a predominantly *textual* appearance.

Accordingly, in the first prototype of the ICONOCLAST system, we aimed to build a unified generator with all these properties: it could generate both output and feedback documents (thus allowing WYSIWYM editing), and it could produce both DRAFTER 'diagrams' and DRAFTER 'texts' (as well as various hybrids in between the two).

[5] WYSIWYM stands for "What You See Is What You Meant". A demonstration of how it works can be found at `http://www.itri.bton.ac.uk/projects/WYSIWYM/wysiwym.html`.

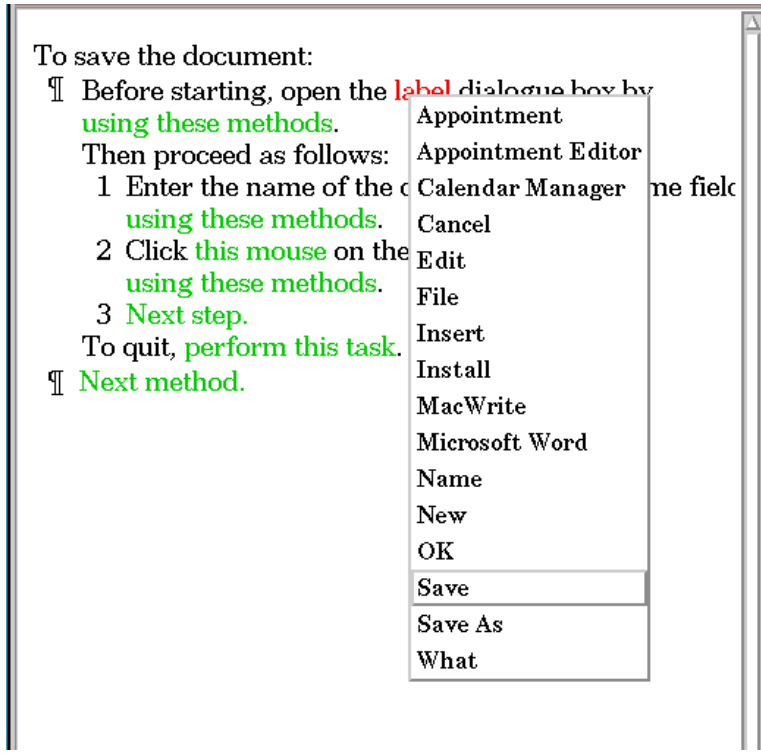

Fig. 9. WYSIWYM Editing.

Figure 9 shows a snapshot of the system during WYSIWYM editing of a procedure. As can be seen, the goal of the procedure (saving a document) has been fully defined, and some sub-actions of a method have been nearly defined: all that remains is to specify the label on a dialogue-box (i.e., the *Save* dialogue-box). The feedback text shows options for adding further information either by green 'anchors' (e.g., *Next step*) or by red anchors (*label*). These anchors are mouse-sensitive, and by clicking on the coloured spans the author can obtain a menu listing the types of object that may be inserted at that location. Red anchors signal that the insertion operation is obligatory; green anchors signal that it is optional. Insertion might appear to be an operation on the feedback text, but actually it is an operation on the underlying knowledge base; the feedback text is then completely regenerated, and might in some cases need to be re-organized (e.g., because information previously conveyed by a single sentence has become too voluminous). In Figure 9, the author has opened in the menu listing the options for the label on the dialogue-box, and is about to choose 'Save'.

Having obtained a complete knowledge base (i.e., having made all obligatory insertions), the author can switch the purpose setting to 'output' rather than

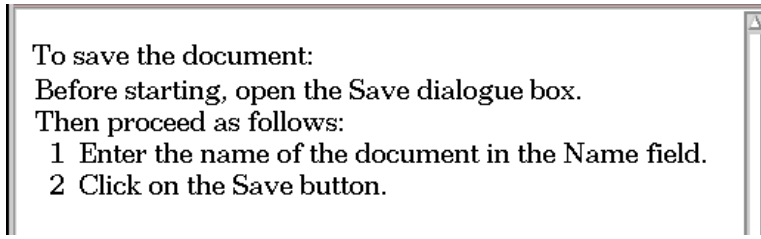

Fig. 10. Output with Textual Appearance.

'feedback'. The result will be a text which conveys the instructional procedure defined in the knowledge base, without signalling options for editing the knowledge (Figure 10). The difference that hits the eye is that the green anchors have gone, but of course this does not mean that the output text was derived merely by deleting the anchors from the feedback text. It is completely regenerated with a different pragmatic purpose setting; again, in some cases, this could lead to radical differences in text structure and syntax.

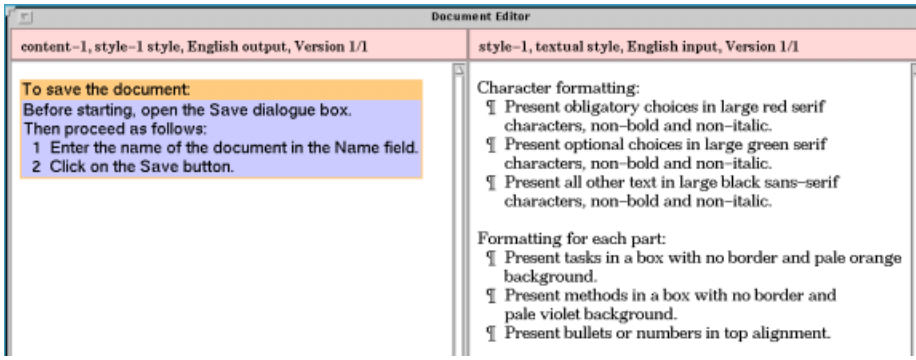

Fig. 11. Output with Hybrid Appearance.

A final novelty of the ICONOCLAST prototype is that it allows the author to edit the *presentational style* of the instructional text/diagram as well as its content. This is done through a style profile, presented in an accompanying pane, which can be modified in the usual way through WYSIWYM editing. In Figure 11, the author has requested a basically textual appearance, except that whole procedures should be presented on an orange background, while their constituent methods (the actions for achieving a goal) should be presented on a violet background: we thus obtain a hybrid between the diagrammatic output of Figure 8, and the textual output of Figure 10.

5 Summary and Conclusions

We have argued that text and graphics are not as distinct as they are generally conceived to be. While we would agree that speech and graphics are clearly discrete categories, we do not believe this to be the case for text and diagrams. Certainly, within the context of documents, these two media are very closely related. If clear categorisation is what we seek, then this is more likely to be between linguistic material – not text – and graphics. A given text or diagram will fall somewhere on a continuum ranging from the purely linguistic to the purely graphical. Texts will tend to fall closer to the linguistic end, and diagrams to the graphical end, clearly some textual genres (e.g., letters) are more purely 'linguistic' than others (e.g., instruction manuals), just as some diagrams (e.g., Venn diagrams) are more graphical than others (e.g., maps). Applying these insights to natural language generation systems, we have shown that these two media can be generated from a common architecture.

If we want to describe (or generate) documents, we have to use a framework that goes beyond purely linguistic features and includes graphical ones. In doing so, we obtain a framework applicable to presentations that would normally be thought of as multimedia, including graphical user interfaces.

References

ABPI, editor (1996–1997). *Compendium of Patient Information Leaflets*. Association of British Pharmaceutical Industry.

André, E. and Rist, T. (1995). Generating coherent presentations employing textual and visual material. *Artificial Intelligence Review*, 9:147–165.

Bouayad-Agha, N., Power, R., and Scott, D. (2000a). Can text structure be incompatible with rhetorical structure? In *Proceedings of the International Conference in Natural Language Generation (INLG-2000)*, pages 194–200, Mitze Ramon, Israel.

Bouayad-Agha, N., Scott, D., and Power, R. (2000b). Integrating content and style in documents: a case study of patient information leaflets. *Information Design Journal*, 9(2–3):161–176.

Fischer, M. (1998). A framework for generating spatial configurations in user interfaces. In *Proceedings of the 1998 meeting of Design, Specification and Verification of Interactive Systems*, pages 225–241.

Fischer, M. (1999). *Automatic Generation of Spatial Configurations in User Interfaces*. PhD thesis, University of Brighton. Also available as ITRI Technical Report ITRI-99-02.

Hartley, A. and Paris, C. (1996). Two sources of control over the generation of software instructions. In *Proceedings of the 1996 Meeting of the Association for Computational Linguistics*, Santa Cruz, California, USA.

Hentenryck, P. V. (1989). *Constraint Satisfaction in Logic Programming*. MIT Press, Cambridge, Mass.

McKeown, K., Feiner, S., Robin, J., Seligman, X., and Tanenblatt, Y. (1992). Generating cross-references for multimedia explanation. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI'92)*, pages 9–16.

Mellish, C., Evans, R., Cahill, L., Doran, C., Paiva, D., Reape, M., Scott, D., and Tipper, N. (2000). A Representation for Complex and Evolving Data Dependencies in Generation. In *Proceedings of the 6th Applied Natural Language Processing Conference (ANLP 2000)*, pages 119–126.

Nunberg, G. (1990). *The Linguistics of Punctuation*. Number 18 in CSLI Lecture Notes. CSLI Publications, Stanford, CA.

Paris, C. and Scott, D. (1994). Intentions, structure and expression in multilingual instructions. In *Proceedings of the Seventh International Workshop on Natural Language Generation*, pages 45–52, Kennebunkport, Maine. Also available as ITRI Technical Report ITRI-94-2.

Paris, C., Vander Linden, K., Fischer, M., Hartley, A., Pemberton, L., Power, R., and Scott, D. (1995). A support tool for writing multilingual instructions. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1398–1404, Montreal, Canada.

Power, R. and Cavallotto, N. (1996). Multilingual generation of administrative forms. In *Proceedings of the 8th International Workshop on Natural Language Generation*, pages 17–19, Herstmonceux Castle, UK.

Power, R., Scott, D., and Evans, R. (1998). What you See Is What You Meant: direct knowledge editing with natural language feedback. In *Proceedings of the 13th Biennial European Conference on Artificial Intelligence (ECAI'98)*, pages 677–681.

Scott, D., Gorman, L., Hartley, A., Paris, C., Pemberton, L., Power, R., and Vander Linden, K. (1995). Characteristics of good administrative forms. Technical Report ITRI-95-3, Information Technology Research Institute, `ftp://ftp.itri.bton.ac.uk/reports/ITRI-95-3.ps.gz`.

Scott, D., Power, R., and Evans, R. (1998). Generation as a Solution to its own Problem. In *Proceedings of the 9th International Workshop on Natural Language Generation*, pages 256–265.

van Deemter, K. and Power, R. (2000). Authoring multimedia documents using WYSIWYM editing. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, pages 222–228.