Automatic Genre-Driven Layout Generation*

Renate Henschel
University of Stirling
Stirling, Scotland
rhenschel@uni-bremen.de

John Bateman
University of Bremen
Bremen, Germany
bateman@uni-bremen.de

Judy Delin
University of Stirling *and*
Enterprise IDU
Newport Pagnell, England
judy.delin@enterpriseidu.com

Abstract

It has long been recognized in natural language generation that, for generated texts to be realistic, they will have to consist of more than simple sets of strings separated by full-stops. A common assumption in this work has been that the form of information presentation straightforwardly follows the rhetorical structure. More recently, this assumption has been shown to be too simple to account for naturally occuring document design: not only the document content, but also the visual appearance of the presented information, serves to achieve the communicative goal of the author. The design of this visual appearance therefore needs to be controlled more flexibly. In this paper, we present a general algorithm for layout production which, dependent on features of the chosen genre, transforms a rhetorical structure into a not necessarily isomorphic layout structure. The algorithm has been implemented as the XSL stylesheet `gemLayout`. We illustrate `gemLayout` in two different genres drawn from a growing corpus of multimodal documents, bird guides and instruction manuals.

1 Introduction

It has long been recognized in natural language generation (NLG) that, for generated texts to be realistic, they will have to consist of more than simple sets of strings separated by full-stops. Hovy and Arens (Hovy and Arens, 1991), for example, presented a framework where a text plan expressed in the terms of rhetorical structure theory (RST: Mann and Thompson, (Mann and Thompson, 1988)) could guide selection of particular LaTeX elements such as enumeration, bullet lists, and emphasis. Approaches to multimodal generation have also been concerned with presenting more than linear text, with combinations of textual and graphical/pictorial information appearing side-by-side. Here again it has been common to appeal to rhetorical structure as providing a description of the communicative intentions that the composite communicative act is to fulfill (e.g., (André et al., 1993; de Carolis et al., 1997)). In most of this work, it has been assumed that the form of information presentation can follow the rhetorical structure relatively straightforwardly. More recently, this assumption has been shown to be too simple to account for naturally occuring document design decisions. Bouayad-Agha, Power and Scott (Bouayad-Agha et al., 2000) argue that there are often extreme mismatches between 'extended' punctuation (such as that addressed by Hovy and Arens) and the rhetorical structure; while Bateman, Kamps, Kleinz and Reichenberger (Bateman et al., 2001) show that this is equally true for the larger scale units of layout and formatting chunks presented on a page.

In this paper we build on the work presented by Bateman *et al.* (Bateman et al., 2001) and show how we are exploring a genre-based approach to constraining this very flexible aspect of multimodal document generation. Within the project 'Genre and Multimodality' (GeM: http://purl.org/net/gem), we are collecting a corpus of multimodal documents from several distinct multimodal genres or text types. This is serving as a source of systematic and motivated constraints for sophisticated layout generation. We present here a prototype implementation of a general algorithm for transforming an RST structure into a not necessarily isomorphic layout structure dependent on fea-

* The GeM project is a project funded by the UK ESRC.

tures of the chosen genre. The implementation experiments with the application of emerging industry-standard tools for processing XML-based documents—in particular, extended style sheet transformations (XSLT), XSL formatting objects (XSL-FO), and commercial renderers for presenting formatting object documents in, for example, the Adobe portable document format (pdf) or postscript such as RenderX (http://www.renderx.com). We illustrate the prototype with two different genres drawn from the GeM corpus, bird guides and instruction manuals. With this work, we are moving towards a position in which sophisticated document layout is a natural and expressive component of an overall NLG process in which not only the document content, but also the visual appearance of the presented information, may be deployed in order to achieve the communicative goals of the author.

2 Modelling layout

Starting from the hypothesis that the visual appearance of presented information also helps fulfill the communicative goals of an author, we consider each piece of information on a page (in a document) from two perspectives: its semantic content and communicative intention on the one hand, and its visual appearance or layout realization on the other. As most commonly adopted in NLG, we use Mann and Thompson's RST for the former. For the layout perspective, we have developed within the GeM project a new layout model which serves as a general descriptive framework for the documents placed in our corpus. The GeM layout model abstracts away from specific domains of application and describes chunks of information solely in terms of their visual properties. The layout of a document is then determined by three components: information about what the minimal layout elements are, about which typographical properties they have, and an account of how they are grouped into more complex layout chunks on the page.

Layout units. In typography, the minimal layout element (in text) is the glyph. In the GeM project, we are primarily concerned with typographical and formatting effects at a more global level, and so consider as minimal layout elements text blocks of the paragraph level, pictures in their entirety, and all other layout elements which are differentiated as a whole from their environment *visually* (e.g. running head, title, caption, page number, list items, ...). We call these minimal layout elements **layout units**.

Typographical realization. The most obvious difference to be observed in realized layout units is the mode in which they are realized—typically linguistic or graphical. Dependent on the chosen mode, different sets of features describe other layout characteristics. For textual elements, we consider type family, type size, type weight, type style (italics or not), justification, color, case, and so on. Since our goal is to explore precisely which combinations of features are useful for discriminating between different genres in our corpus, we adopt only those features which show themselves to be useful in this respect; this means that the features enumerated are by no means comprehensive: for a full set, from which we make a motivated selection, we employ standards such as the XSL-formatting objects specification. For the graphical layout units, the only choice we have for their realization presently is their size, because we are currently working with ready-to-show pictures as input. This is modelled in the layout structure.

Layout structure. The layout structure describes how layout units are hierarchically grouped into larger layout chunks. For instance, the heading and its associated text form together a larger layout element, or the cells of a table form the larger layout element "table". The criterion for grouping layout units into chunks is that the chunk should consist of elements of the same visual realization (font-family, font-size, ...), or the chunk is differentiated as a whole from its environment *visually* (e.g. by background colour or a surrounding box); motivations and methods for identifying layout chunks have been discussed by Reichenberger *et al.* (Reichenberger et al., 1995). Any layout chunk can consist of layout elements involving different modes (text and graphics).

The layout structure of a document is a hierarchical structure, with the entire document being the root. Each layout chunk is a node in the tree, and the minimal layout units are the terminal nodes of that tree. The grouping

into complex chunks—the layout structure—is determined by: (i) the rhetorical structure of the information to be presented; and (ii) **canvas constraints**—constraints arising from the medium used (paper size and quality) and from presentation decisions imposed on a document as a whole. Examples for layout chunks derived from the RST structure are chapters, sections, and paragraphs. The chapter–section–paragraph hierarchy has also been explored by (Power, 2000), who labels it **document structure**. Examples for layout chunks generated by canvas constraints are pages and columns. Typical for this second kind of layout grouping is that even sentences are broken apart, and can readily belong to different layout chunks in the output document. The final layout which appears before the reader shows a layout structure which meets both types of constraints (RST constraints and canvas constraints).

The layout of a document is not fully determined by grouping layout units into a tree structure; further information is required about the actual position of each unit in the document (on or within its page). For this, we introduce an **area model**,[1] which recursively specifies rectangular sub-areas of the page area in a grid-like manner. These sub-areas then serve to determine the position of each layout chunk or layout unit. Two layout elements are called **adjacent**, if they are placed into two either horizontally or vertically adjacent subareas.

Figure 1 shows an example layout structure.

The described layout structure differs from Power's (Power, 2000) document structure in that

- it reflects the production and canvas constraints which the realization of a given document structure is subjected to (decisions about pagination, columns, margins, hyphenation, etc.);
- it specifies navigational elements—layout elements which are not derived from the content, but which serve to guide the reader through the document (e.g. page numbers, pointers, running heads, titles);
- it specifies the position of layout elements on the page.

[1]This should not be confused with the similarly named but different construct from XSL-FO.

3 RST structure and layout structure

In its original form, RST investigates the relations which hold between consecutive clauses, or bigger adjacent fragments of a text—the so-called text **spans**. An RST relation typically combines two adjacent text spans of unequal importance, a text span which is central to the writer's communicative goal, the **nucleus**, and a text span which supports the message of the nucleus, the **satellite**. Multinuclear relations between two or more spans of equal importance (e.g. list, joint, sequence, ...) are also possible. Adjacent text spans which are related by an RST relation form then together a bigger text span, which itself is the nucleus or satellite of an RST relation. So the RST relations are recursively applied on progressively larger text portions, resulting in a tree structure the root of which is the whole text, and whose terminal nodes are the clauses of this text. RST structures have been used as data structures mediating between text planning and tactical generation in pipeline organized NLG systems; the terminal nodes of the RST tree are then semantic propositions. We adopt this latter model for the generation process developed here, but generalize the RST structure to hold over multimodal presentations.

One central question of our investigation is the relation which holds between the RST structure behind a document and the corresponding layout structure found on the page. Inspection of our corpus has so far led to two insights. First, there are layout elements which do not contain information from the input RST structure, but serve to help the reader navigate through the document (e.g. page numbers, running heads, pointers), which we call *navigational elements*. Second, the output layout structure does not generally preserve the RST input structure. We observe three principle types of structural transformation:

1. **Sequential layout (concatenation):** The terminal nodes of an RST tree are all realized inside one and the same layout block (in case of text, with identical typography) maintaining the adjacency of nucleus/nuclei and satellites of one and the same relation.

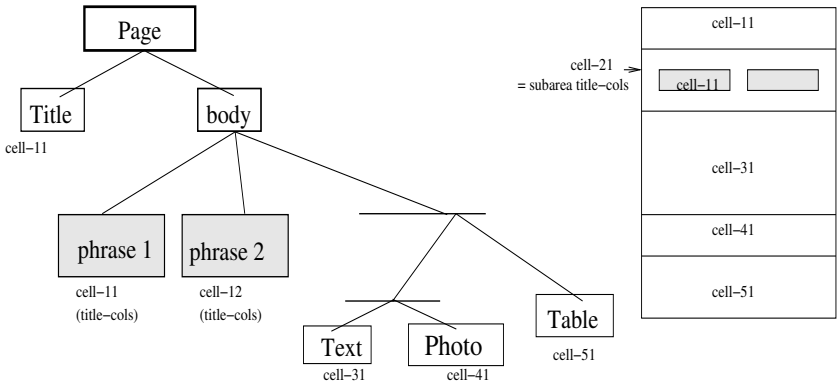
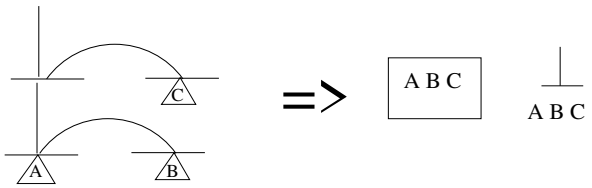
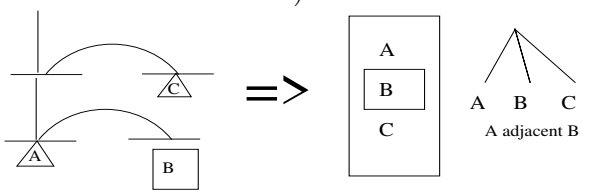


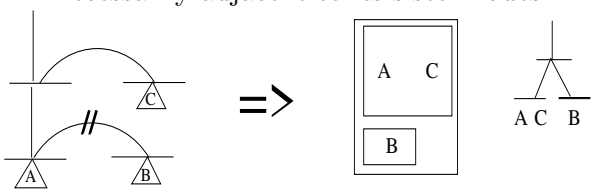
Figure 1: Layout structure of a page similar to that given in Figure 3 below



2. **Emphasis:** A certain satellite or nucleus is realized with different layout properties than its sister nodes, thus creating an extra layout block, but maintaining adjacency with the other relation constituents (nucleus and satellites).



3. **Extraposition:** A certain satellite or nucleus is cut from the RST tree and realized at a different place in the document, not necessarily adjacent to its sister nodes.



For multinuclear relations, we state that they should be of equal layout status: if one nucleus is cut, all are cut; if one nucleus is emphasized, all of them are emphasized. Thus 'emphasis' here cannot be seen as a mechanism to highlight something against a context, but to distinguish constituents from one another.

Typical examples for sequential layout are paragraphs. The layout of diagrams or photos in a textual environment already requires emphasis, because the size of the picture usually does not fit into the fixed line-height of a paragraph. Extraposition can be found, for example, in books where illustrations are placed in an extra section at the end of the book, or on a page where illustrating material is generally placed at the top or bottom and referred to from the text. A typical example for multinuclear emphasis is the layout of a 'sequence' or 'list' relation as a list.

These three structural possibilities can occur at different places in one and the same RST tree. We assume here sequential layout as the default layout structure. We will mark the cutting (extraposition) of an RST subtree by vertical double lines which cross the arc between the cut tree and its parent node, and the emphasizing of an RST subtree by a box around the subtree's root node. We call an RST tree marked in such a way *layout-marked*. Note that the layout structure generated by a cut is not homomorphic. It does not maintain adjacency relationships between the constituents (nuclei, satellites) of one relation. Layout structures which are isomorphic or homomorphic to the RST structure are special cases of the above definition. Isomorphic layout can be obtained by cutting or emphasizing every satellite. It can also be obtained, if e.g. every satellite in the top part of an RST tree is cut, but not in the bottom part.

Crucially, we argue that this breaking of RST relations is a *genre dependent process*. It is then a primary aim of the GeM project, and its corpus collection, to ascertain which constraints on the decomposition process can be allocated to genre considerations, which to canvas con-

straints, and which remain free.

4 The gemLayout algorithm

The `gemLayout` algorithm creates from a given RST structure a layout structure formed out of layout units, and attributes them with certain typographical features and relative positions on the page. It allows a conditional break (cut or emphasis) for all satellite arcs, as well as emphasizing the nuclei of multinuclear relations. It is in these break conditions that we bring the results of the corpus analysis to bear: break conditions are considered genre dependent and are therefore given as extra input. `gemLayout` without input break conditions will generate a flat layout structure, thus simulating traditional text generation programs. `gemLayout` has been implemented as a cascade of three XSL stylesheets: `gemTrans`, `gemExtrapose` and `gemFO`. It operates on the input RST tree and the genre-specific break conditions. Both are represented as XML descriptions. The RST representation follows the GeM RST format (see the GeM website for the DTD) augmented by an XML representation of the propositional content of the leaves. The first two transformations produce a layout structure specified in the form of an XML tree with the elements <pro> (to indicate terminal propositions), <cut> (for extraposed subtrees), and <block> for composite layout elements and emphasized subtrees. `gemTrans` is responsible for the structural transformation of the input RST tree (flattening the RST structure and adding navigational elements) as well as for the order of layout siblings. The flow-chart in figure 2 shows in an informal manner one step of `gemTrans`'s top-down recursive traversal through an RST tree. In the diagram, $node is a node in the input RST tree. $node can either be terminal or denote an RST relation. This relation can be a mononuclear relation, $nucleus and $satellite are then the spans it combines. If $node is a multinuclear relation, $\$nuc_1$ ... $\$nuc_n$ are the nuclei which build $node. The alternative actions for mono- and multinuclear nodes are separated by a dashed line in the figure. Note that in the figure we have always given a nucleus before satellite order, although the sequence of these constituent spans is in fact implemented as dependent on the stated break conditions in the last two action blocks (extraposition, isomorphic layout).

`gemExtrapose` serves solely to extrapose all cut material out of its original RST neighbourhood. Finally the third `gemLayout` stylesheet, `gemFO`, transforms the GeM layout structure specified in XML into an XML formatting objects specification, and carries out tactical generation of the layout leaves.

This first prototype transfers the RST tree directly into XML formatting objects; a planned second prototype will create a layout structure in the form of the GeM layout model (Section 2) for more complex layouts.

5 Examples — bird guides and telephone manuals

This section illustrates the layout generation algorithm with two different genres present in the GeM corpus: (1) bird guides, and (2) telephone instruction manuals. The implementation is, for the time being, restricted to a selection of simple layout decisions in order to show the principle behind the algorithm. The decisions considered are: content-determined layout structure generation, simple layout element positioning, title generation, typographic similarity/distinction, and linguistic realization. In all cases, we are concerned here with the generation of the layout structure and the corresponding page, not with the generation of the content for the layout elements. The content generation is to be treated in part by traditional NLG components and so is not a major focus here; for example, 'title generation' means here the decision where to add a title and where not (i.e. generation of an extra layout element or not)—not the generation of the actual words of the title.

The determination of the position of layout elements is in this first implementation restricted to four possibilities for arranging nucleus and satellite, or for arranging a cut satellite and the remaining RST tree material. These are nucleus-first and satellite-first in both dimensions: horizontally and vertically. The question of nucleus-satellite order inside a basic layout unit (a paragraph) has in our examples been simplified as 'nucleus-first' always; further research concerning this problem can be found elsewhere (e.g., (Bouayad-Agha et al., 2000)).

The genre-specific break conditions, which

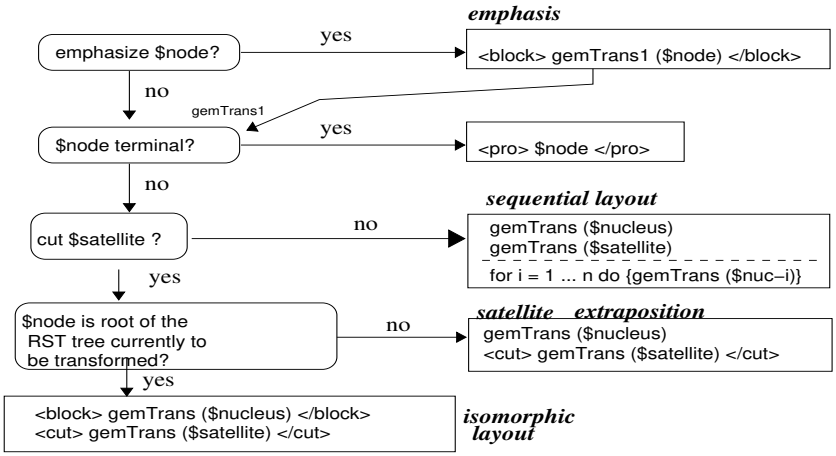


Figure 2: Algorithm `gemTrans`

trigger the start of an extra layout element and break the traditional sequential text arrangement—can be found in different sources: semantic content of the satellite, mode in which the satellite should be expressed, RST relation, and structural properties of the RST structure. The examples following utilise all four of these types of information. We have extracted the most apparent break conditions for bird guides and telephone manuals from the material in our corpus; at present this is done manually, although the corpus has been extensively annotated with respect to layout model and RST structure in order to derive break conditions automatically in the future.

Bird guides. A bird guide is a list of similarly appearing document pieces, each of which is dedicated to one particular bird. To simplify matters further, we assume here a one bird per page layout (which is the situation in around half of our investigated bird guides). We consider now the layout generation for one variable example bird—the bird in focus. Within this genre, we observe that the Latin name of the bird in focus and the information about its family are typically separated layout units. In addition, each bird page has at least one central picture showing the bird (realized in older books as a line-drawing, in more recent ones as a photograph); this graphical layout element is also separated from the textual information. And, finally, in one bird per page books, the information to be presented in textual form is usually split into two layout units—one represented as linear text, the other as an itemized list. These two layout elements have differing typographic realization, indicating the nucleus-satellite distinction, but are otherwise of relatively equal weight. Typically, each bird page has its own title. In Table 1, we summarize the stated break and title conditions as XSL paths that trigger particular mappings into layout structure. The table also shows their consequences for typographical and linguistic realization.

An application of `gemLayout` equipped with the break conditions given in Table 1 to the represented RST structure for the communicative intention of the page results in a tree of XSL formatting objects, which may then be rendered straighforwardly into a PDF output file (for this we have used RenderX). Figure 3 shows the corresponding output page.

Instruction manuals. Our second example is the generation of a page of a telephone manual instructing the user how to install a new telephone device. In the GeM corpus, it is apparent that most pages from telephone manuals have highly structured layout. It is rare that one can find a paragraph formed by more than one sentence. The analysis of the RST structure of such a page reveals that it consists of a high percentage of multinuclear relations (sequences, lists and restatements). All these relations undergo an isomorphic transformation into layout, i.e. each of their nuclei is realized as a separate layout element, they do not form a sequential text. However, these separate elements are generally realized preserving adjacency. Thus we consider

cut condition	satellite realization

`//proposition/[@id=$satellite/]/pred='latin'`	`font-style='italic'`
	`lingustics='ellipsis'`
	`dimension='horizontal'`

`//proposition[@id=$satellite]/pred='family'`	`font-style='italic'`
	`lingustics='ellipsis'`
	`order='sat-first'`
	`dimension='vertical'`

`//proposition[@id=$nucleus]/@mode='graphics'` `//proposition[@id=$satellite]/@mode='graphics'`	

`count(id($nucleus)//proposition)`
$\cong$ `count(id($satellite)//proposition)`
`and count(id($nucleus)//proposition) > 6`
	`font-family='arial'`
	`font-size='smaller'`
	`lingustics='itemized-list'`
	`order='nuc-sat'`

title condition	**realization**
`//rst-root[@id=$node]`	`font-weight='bold'` `font-size='16pt'`

Table 1: Break conditions for bird guides

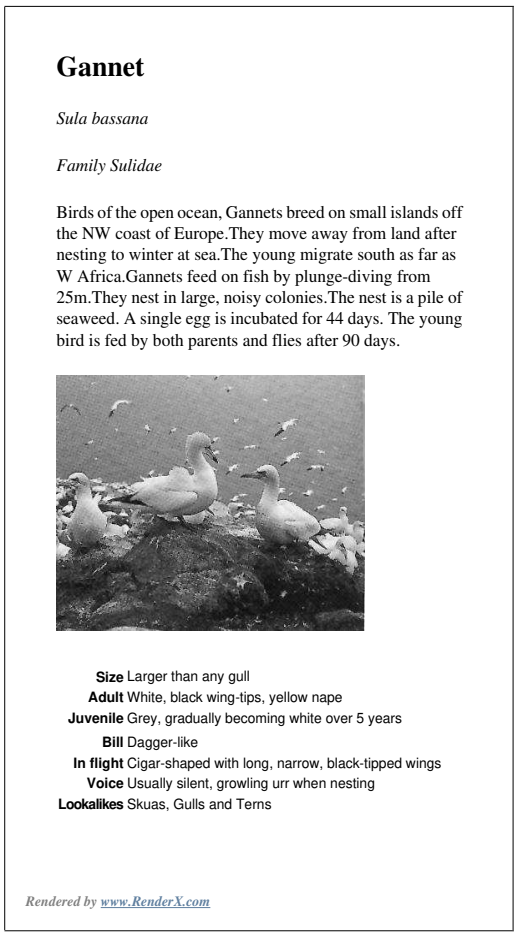


Figure 3: Generated bird page

this break as an emphasis break. So, the first and most powerful break condition for instruction manuals is to trigger an emphasis break for all nuclei in any multinuclear relation. For the realization of these nuclei, we have to distinguish between different RST relations. The nuclei of a multinuclear restatement with one of its nuclei in graphical mode are simply rendered in two adjacent blocks arranged horizontally, text first, diagram second. Multinuclear relations of type 'sequence' are usually layouted as lists. Our corpus analysis suggests three main possibilities: the list items are separated against each other by a horizontal rule, the list is realized as an enumerated list, or the list is realized as a bulleted list. These three possibilities are applied during the RST-tree traversal one after another from top to bottom. The list items (nuclei) are always arranged vertically. The break conditions for instructions are as shown in Table 2. Also, as in bird guides, each telephone manual page has its own title. A resulting page is given in figure 4.

6 Conclusion

We have presented a brief overview of a general algorithm, implemented as a cascade of XSLT style sheets, for generating layout on the basis of corpus-derived design decisions. The algorithm `gemLayout` is capable of generating homomorphic (as in (Power, 2000)) as well as non-homomorphic layout structures, and moreover it has made the crucial step from traditional linear top-to-bottom arrangement of derived layout elements (again as in (Power, 2000)) to a two-dimensional arrangement. For our next round of implementation, we will fold in the modules which for the present have been fac-

emphasis condition	realization
`//multi-span[@id=$node] and $break-count=0`	`realization='rule'`
`//multi-span[@id=$node] and $break-count=1`	`realization='olist'`
`//multi-span[@id=$node] and $break-count=2`	`realization='ulist'`
`//proposition[@id=$nucleus-1]/realization/full and //proposition[@id=$nucleus-2]/@mode='graphics'`	`dimension='horizontal'`

Table 2: Break conditions for instruction manuals

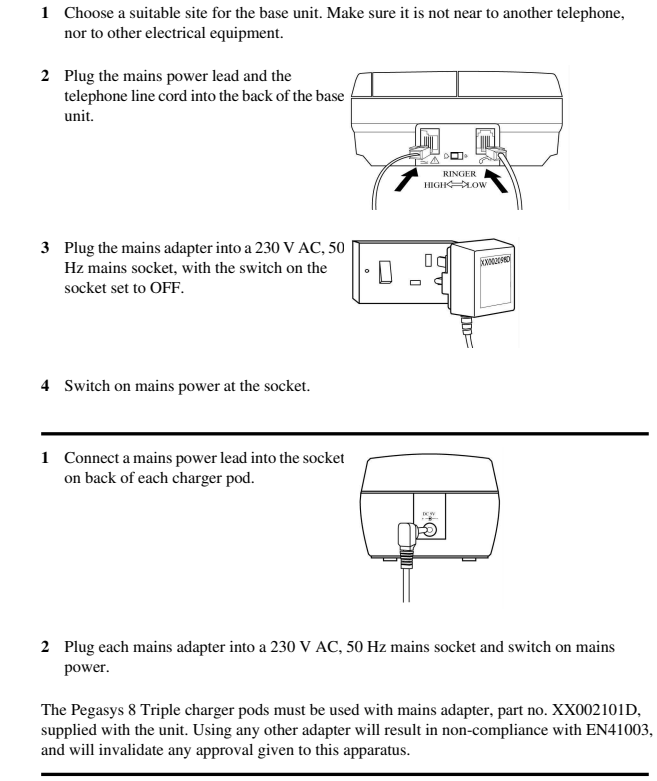


Figure 4: Generated instruction page

tored out of the discussion; in particular: mode allocation (graphics or text), more possibilities for the position of layout elements, generation of pointers/links, generation of captions, typographic features, color, background and margins, and canvas determined layout decisions. For the present, we have demonstrated that NLG will soon be capable of employing general techniques for the generation of motivated multimodal documents approaching, at least for some genres, those produced by design professionals.

References

Elizabeth André, Wolfgang Finkler, Winfried Graf, Thomas Rist, Anne Schauder, and Wolfgang Wahlster. 1993. WIP: the automatic synthesis of multimodal presentations. In Mark T. Maybury, editor, *Intelligent Multimedia Interfaces*, pages 75–93. AAAI Press, Menlo Park (CA).

John A. Bateman, Thomas Kamps, Jörg Kleinz, and Klaus Reichenberger. 2001. Constructive text, diagram and layout generation for information presentation: the DArt$_{bio}$ system. *Computational Linguistics*, 27(3):409–449.

Nadjet Bouayad-Agha, Richard Power, and Donia Scott. 2000. Can text structure be incompatible with rhetorical structure? In *Proceedings of the International Natural Language Generation Conference (INLG-2000)*, pages 194–200, Mitzpe Ramon, Israel.

Berardina de Carolis, Fiorella de Rosis, and Sebastiano Pizzutilo. 1997. Generating user-adapted hypermedia from discourse plans. In *Proceedings of Fifth Congress of the Italian Association for Artificial Intelligence (AIIA'97)*, pages 334–345, Rome.

Eduard H. Hovy and Yigal Arens. 1991. Automatic generation of formatted text. In *Proceedings of the 8th. AAAI*, pages 92–96, Anaheim, California.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Richard Power. 2000. Mapping rhetorical structures to text structures by constraint satisfaction. Technical Report ITRI-00-01, ITRI, University of Brighton.

Klaus Reichenberger, Klaas Jan Rondhuis, Jörg Kleinz, and John A. Bateman. 1995. Effective presentation of information through page layout: a linguistically-based approach. In *Proceedings of ACM Workshop on Effective Abstractions in Multimedia, Layout and Interaction*, San Francisco, California. ACM.