Final Report for Grant GR/M36960/01: ICONOCLAST

1 Background

ICONOCLAST (Integrating CONstraints On Content, Layout And STyle) grew from two points of dissatisfaction with earlier work on Natural Language Generation (NLG). The first was the lack of any attention to large-scale document structure: a 'text' was regarded as a series of sentences, any further organisation into sections, paragraphs, or indented lists being treated (if at all) as a subsequent formatting operation. Analysis of corpora for previous NLG projects such as DRAFTER[1] (Paris et al., 1995) convinced us that this was a mistake: since layout interacts with wording and meaning, at least an abstract specification of document structure should be integrated within the generator itself.

The second dissatisfaction was the inflexibility of most NLG systems, and in particular their inability to accommodate stylistic preferences of users, except through very broad distinctions like formal vs. informal. Like the failure to address layout, this limitation is a serious obstacle to commercial application of NLG. In all the application contexts that we have investigated, including software manuals, social security forms, and patient information leaflets, producers care a great deal about style, and many have taken the trouble of drawing up detailed guidelines on layout, punctuation and wording.

The main idea behind ICONOCLAST was that both these problems could be addressed by a system that formulated NLG as a Constraint Satisfaction Problem (Hentenryck, 1989). Such a generator proceeds in two main phases: first, it imposes constraints on the set of permitted solutions; secondly, it searches the space of solutions satisfying these constraints, looking for the candidate that best satisfies a set of preferences. The integration of layout with meaning and wording can be achieved by defining constraints across these data-types; detailed stylistic control can be achieved by allowing users to determine, at least to some extent, which of the potential constraints and preferences are applied.

2 Objectives

This section describes summarily how the four objectives listed in the proposal were addressed; a more detailed account follows in the next section.

A. Identify the constraints that authors need to control when designing documents, including constraints on layout and style. We collected and annotated a corpus of patient information leaflets — a genre for which layout and style are important concerns. From analysis of this corpus we identified some ways in which layout and wording interact, and some dimensions of style. Meetings were also held with the three user-group companies, each active in the field of document design.

B. Develop a model of how constraints interact in the process of language production. We introduced the notion of abstract document structure, a generalisation of Nunberg's 'text-grammar' (Nunberg, 1990), as a level of description that mediates between the wording of a document and graphical features of layout. This has allowed us to state formally some common ways in which layout and wording interact.

[1] DRAFTER was supported by EPSRC grant GR/J19221.

C. Represent and apply these constraints in a system that automatically generates high-quality documents. We showed that documents could be planned by defining constraints over the document-structure features that realize each rhetorical unit in the message. Formally, this method reduces document structuring to a Constraint Satisfaction Problem, which means that solutions can be enumerated and evaluated using the constraint-solving and optimization libraries of languages that support Constraint Logic Programming.

D. Investigate ways of helping the author to control the application of constraints in a generation system. A constraint-based generator is flexible both in the range of solutions that can be generated, and the control that can be given to developers or users in varying the constraints. We experimented with two interfaces through which document designers could control the constraints: first, WYSIWYM editing (Power and Scott, 1998) of a style profile; second, a set of parameters like paragraph length and semicolon frequency, each controlled through a slider.

3 Main contributions

3.1 Abstract Document Structure

An important innovation has been the development of the concept of *Abstract Document Structure*. In previous research, especially Nunberg (1990), this was used only as a means of assigning or interpreting punctuation. We have broadened the concept so that it applies also to the realisation of text structure through layout. This brings several benefits:

- It allows us to state constraints on layout and wording at the right level of abstraction, and hence to generate efficiently documents in which layout and wording are co-adapted.
- By including structures larger than paragraphs, it allows a generation system to plan documents of realistic size.
- By including indented structures like bulleted lists, it allows a generator to choose among a wider range of expressive patterns.

Abstract document structure has filled a gap in previous theory: while the realisation of semantics by syntax had been explored in detail, little attention had been paid to the realisation of rhetorical structure. Abstract document structure has been adopted as basic data-type in the RAGS[2] specification (RAGS, 1999), and was highlighted in the final address of the International Conference on NLG at Mitzpe Ramon in June 2000 as a significant development in the field.

3.2 Constraint Satisfaction

We have succeeded in formally defining document structuring as a Constraint Satisfaction Problem. In doing so, we have provided a method of more general applicability which can be considered whenever one structure is realised by another: for example, it has been used by Fischer (1998) in a system that automatically generates diagrams for user interfaces.

Essentially, the idea is to associate each object in the input with a set of features describing the various ways in which it can be realised, each feature being implented as a variable with a finite domain of possible values. Constraints are then imposed among the realisation features (thus ensuring that the realisation is well-formed), and also between the realisation and the input (thus ensuring that the realisation expresses the right content). Further features are assigned in order to evaluate the solution; these work by counting (and weighting) non-fatal defects, thus allowing users to impose and prioritize their preferences.

Once the problem has been set up in this way, it can be solved by standard libraries for constraint solving and search, found in any language supporting Constraint Logic Programming (e.g., Eclipse or Sicstus). The method does not depend on any particular search technique: the

[2]RAGS, an emerging reference architecture for NLG systems, is supported by EPSRC grant GR/L77102.

same constraints and preferences would serve for generate-and-test, branch-and- bound, repair methods, or stochastic search (e.g., through a genetic algorithm or simulated annealing). Thus advances in the technology of constraint reasoning can be incorporated without altering the system in any other way.

3.3 Multiple solutions

By following this approach, we have developed an NLG system that generates enormously more potential solutions than any other program that we are aware of. Even for a simple semantic input comprising five propositions, the document structurer typically generates around 500 acceptable plans – and this ignores further variations due to the wording of individual propositions.

Of course there are contexts in which we do not wish to see 500 solutions; one solution will suffice, and the important issue is speed. For such contexts, we can if necessary impose more stringent constraints. However, the flexibility to enumerate a wide range of alternatives brings two advantages, one foreseen, and the other unexpected.

- We original saw this flexibility as a means of obtaining fine-grained stylistic control: since the system was willing to consider many solutions, it would be able to find solutions satisfying particular individual preferences.
- However, the ability to enumerate many solutions also makes the program into a valuable research tool. At least for short examples, we are now able to enumerate all solutions permitted by a given set of constraints, and to order them according to a given set of preferences. This allows us to explore quickly the consequences of adopting any proposed constraint or evaluation metric. We have used this method, for example, in comparing various evaluations of centering assignments (Walker et al., 1998) in order to improve continuity of reference throughout a text.

3.4 Corpus for studying layout

Early in the project, we decided to study the domain of Patient Information Leaflets (PILs). This domain has several advantages: restricted semantics, practical importance, intense use of layout, and distinctive house styles (different leaflets by the same company clearly resemble one another).

A major result of the project has been to collect a corpus of several hundred British PILs and to annotate them for layout features. This corpus will be made available for use by other researchers when copyright issues with the companies have been checked. The corpus comprises about 500 leaflets, each available in four formats: Word, RTF, HTML, and an SGML markup that we have developed for annotating layout features.

3.5 Rhetorical transformations

Patient Information Leaflets contain many passages in which the document structure, articulated through layout and punctuation, is incompatible with the rhetorical structure that would be assigned on a purely logical basis. For instance, we find conditional instructions in which a complex antecedent has been raised by one level so that it is coordinated with the conditional instead of embedded within it:

> Are you pregnant?
> Are you breast-feeding?
> Are you allergic to penicillin?
>
> Tell your doctor if the answer to any of these questions is YES.

To analyse such cases, we distinguish *deep* rhetorical structure from *surface* rhetorical structure. The semantic input to the NLG system is a deep rhetorical structure which may be converted by transformations to a surface rhetorical structure before it is mapped on to document structure. We have observed that passages including vertical lists are very often the result of such transformations. So far, three transformations have been investigated: *abstraction* (illustrated above), *factoring* (a type of aggregation) and *extraposition* (the raising of an embedded satellite so that it becomes a kind of after-thought).

3.6 Coherence

Our approach to text planning has allowed us straightforwardly to include some ideas from GNOME[3] on favouring coherence and salience in centering transitions. The text planner maps a surface rhetorical structure on to *all* abstract document structures that satisfy a set of formation rules. (This does not of course mean that we have a complete theory of rhetorical structure or document structure; however, within the limits of our theory, the text planner is exhaustive.) By generating multiple document structures, the text planner allows for many possible orderings of the elementary propositions; some of these will favour continuity of reference, by grouping together propositions about the same discourse referent, while others will chop and change. To find the most coherent document structures, we enumerate possible assignments of backward and forward centers *during text planning*, so that centering transitions can be evaluated before any propositions are passed forward to the tactical generator. The implementation of this method again depends on Constraint Logic Programming.

3.7 Logical complexity

An early finding of ICONOCLAST was that layout is often used when the content is logically complex. For instance, a conditional instruction may be expressed by a sentence containing an indented list. It is characteristic of logically complex material that many of the situations described are *hypothetical*. A leaflet may say 'If a rash develops, tell your doctor', without implying that the rash actually exists or will ever exist. Unfortunately, the methods of knowledge representation usually employed in NLG (e.g. languages like LOOM; MacGregor and Bates, 1987) cannot represent logically complex material; in particular, they have no way of distinguishing between hypothetical entities (such as the rash) and real ones (such as the medicine).

Because conditionals are so common, and because they have interesting implications for layout, we devoted some effort to extending our methods of knowledge representation and knowledge editing. This was done by developing a representation based on Discourse Representation Theory (DRT) (Kamp and Reyle, 1993) in which referents are bound to logical contexts (the DRT 'boxes'); a new knowledge-editing operation allows the author to distinguish local scope (for a hypothetical entity) from global scope (for a real entity). This operation does not allow full control over logical form, but it seems at least sufficient to cover the distinctions that can be made in ordinary language.

4 Methodology

The ICONOCLAST demonstrator is not intended as a commercial application. It would not be appropriate to judge it as a tool already able to generate good patient information leaflets, given suitable input from a document designer. We think it should be assessed as an innovative approach to generation that might potentially allow developers to build diverse systems, tailored to particular contexts, very quickly. The properties of the generator can be changed simply by adding or removing constraints; if necessary, the balance between time and quality can also be controlled by allowing a choice of search methods (e.g., exhaustive search for short passages with no time pressure, and partial or stochastic optimization for longer documents).

To develop a good constraint-based NLG system several requirements must be met:

Framework
: It is not at all obvious whether NLG can be treated as a Constraint Satisfaction Problem, and if so, what is the best way of setting up the problem.

[3]GNOME (Generating NOMinal Expressions) was supported by EPSRC grant GR/L51126.

Constraints and Preferences

Once a framework has been established, one can address the task of creating a large stock of useful constraints and preferences which embody general principles of good style and allow users to impose their individual preferences.

Search

The performance of the system will depend on which search method is used: using Constraint Logic Programming, this aspect can be largely decoupled from the definitions of constraints and preferences.

Our focus has been on the first requirement. The second can be added gradually with contributions from many researchers and many projects. The third depends mostly on progress in more mathematical fields of computer science.

5 Project planning

The original project plan was maintained except for a change in emphasis during the second year. Originally, we had expected to identify a substantial 'catalogue' of constraints by 18 months, codifying constraints identified by other researchers as well as discovering new constraints relating to layout and style. However, experiences with early prototypes persuaded us that a more important objective was to find effective ways of formalizing and controlling a small number of basic constraints on the relationship between rhetorical form and document structure.

Another new direction taken during the project was the decision to compile a corpus for studying layout. Originally we assumed that constraints and preferences on layout and style would be identified by interviewing document designers, but after discussions with collaborators in the user group we came to accept their view that there are three kinds of principles – ones document designers *say* they use, ones they *think* they use, and ones they *actually* use; the latter can only be revealed by studying the documents themselves. Having decided that a corpus of patient information leaflets would be a useful resource both for the project and for the community, we devoted some effort to providing such a corpus on-line.

6 Dissemination, exploitation and further work

About 20 articles and conference papers reporting ICONOCLAST work have been published during the project. More detailed presentations can be found in some ITRI Technical Reports and in Nadjet Bouayad-Agha's forthcoming Ph.D. thesis

Papers reporting ICONOCLAST work have been presented at all the major conferences and workshops in computational linguistics and NLG over the period of the project: COLING-98, INLG-98, EACL-99, EWNLG-99, INLG-00, COLING-00 and ACL-00. We also organised the AAAI Symposium on Layout at Cape Cod in November 1999 (co-chaired by Donia Scott and Richard Power), where two papers relating to ICONOCLAST were presented. An article describing ICONOCLAST has appeared in the *Information Design Journal*, an important forum for the document design community (Bouayad-Agha et al., 2000). We are at present writing an article on document structure, to be submitted to the *Computational Linguistics* journal.

Our work on ICONOCLAST has inspired a new European project, PILLS (Patient Information Leaflet Localisation System) which starts in December 2000. PILLS will investigate whether a larger project for generating patient information leaflets in several languages would be feasible, and is likely to lead to commercial applications. The coordinator of this project, Berlitz, is interested in collaborating with us in commercial exploitation if this proves possible.

Several other spin-offs will be mentioned briefly:

- Daniel Paiva's Ph.D. work on linguistic style has been directly influenced by ICONOCLAST, and studies the same genre (patient information leaflets).

- Markus Fischer's Ph.D. work on automatic generation of diagrams in user interfaces (Fischer, 1998) made use of the constraint-solving techniques introduced in ICONOCLAST.
- The concept of abstract document structure, developed in ICONOCLAST, has been adopted as a basic data type in the RAGS project.
- The constraint-based document planning method developed in ICONOCLAST has been used in the GNOME project.

To assess potential developments of our approach we have discussed the ICONOCLAST demonstrators with professional document designers[4]. It was felt that our present technology could be applied for the automatic production of *routine* documentation, especially when a company wanted to control the style and content of documents in many languages. For applications requiring very high quality, the program might play a rather different role — that of sketching a wide range of potential solutions as an aid to creativity in the human designer; we discussed for example how a constraint-based program might explore the options in laying out a complex form. The approach could also be used for rapid prototyping, even if the program's output required substantial revision by a human editor. Finally, the program's ability to list the stylistic defects in any generated solution might provide the basis for a useful training tool.

References

Bouayad-Agha, N., Scott, D., and Power, R. (2000). 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 Design, Specification and Verification of Interactive Systems*, pages 225–241.

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

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

MacGregor, R. and Bates, R. (1987). The LOOM knowledge representation language. In *Proceedings of the Knowledge-Based Systems Workshop,* St. Louis, April 21–23.

Nunberg, G. (1990). *The Linguistics of Punctuation.* CSLI, Stanford, USA.

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 Scott, D. (1998). Multilingual authoring using feedback texts. In *Proceedings of the 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics*, pages 1053–1059, Montreal, Canada.

RAGS (1999). The RAGS project: towards a reference architecture for natural language generation systems. Technical report, Information Technology Research Institute, Brighton, UK.

Walker, M., Joshi, A., and Prince, E. (1998). *Centering theory in discourse.* Clarendon Press, Oxford.

[4]We thank in particular David Lewis and Rob Waller of the Information Design Unit.