Building Natural Language Generation Systems

Ehud Reiter
University of Aberdeen

Robert Dale
Macquarie University

CAMBRIDGE
UNIVERSITY PRESS

PUBLISHED BY THE PRESS SYNDICATE OF THE UNIVERSITY OF CAMBRIDGE
The Pitt Building, Trumpington Street, Cambridge, United Kingdom

CAMBRIDGE UNIVERSITY PRESS
The Edinburgh Building, Cambridge CB2 2RU, UK http://www.cup.cam.ac.uk
40 West 20th Street, New York, NY 10011-4211, USA http://www.cup.org
10 Stamford Road, Oakleigh, Melbourne 3166, Australia
Ruiz de Alarcón 13, 28014 Madrid, Spain

First published 2000

Printed in the United States of America

Typeface Times Roman 10/12 pt. *System* LaTeX 2ε [TB]

A catalog record for this book is available from the British Library.

Library of Congress Cataloging in Publication Data

Reiter, Ehud, 1960–

Building natural language generation systems / Ehud Reiter, Robert
Dale.

p. cm. – (Studies in natural language processing)

ISBN 0-521-62036-8

1. Natural language processing (Computer science) I. Dale,
Robert, 1959– . II. Title. III. Series.

QA76.9.N38R45 1999
006.3′5 – dc21 99-14937
CIP

ISBN 0 521 62036 8 hardback

Contents

List of Figures

1 Introduction

NATURAL LANGUAGE GENERATION (NLG) is the subfield of artificial intelligence and computational linguistics that focuses on computer systems that can produce understandable texts in English or other human languages. Typically starting from some nonlinguistic representation of information as input, NLG systems use knowledge about language and the application domain to automatically produce documents, reports, explanations, help messages, and other kinds of texts.

NLG is both a fascinating area of research and an emerging technology with many real-world applications. As a research area, NLG brings a unique perspective on fundamental issues in artificial intelligence, cognitive science, and human–computer interaction. These include questions such as how linguistic and domain knowledge should be represented and reasoned with, what it means for a text to be well written, and how information is best communicated between machine and human. From a practical perspective, NLG technology is capable of partially automating routine document creation, removing much of the drudgery associated with such tasks. It is also being used in the research laboratory, and we expect soon in real applications, to present and explain complex information to people who do not have the background or time required to understand the raw data. In the longer term, NLG is also likely to play an important role in human–computer interfaces and will allow much richer interaction with machines than is possible today.

The goal of this book is to explain the central ideas in NLG both to the advanced student who is interested in learning about NLG as a research area and to the software developer who is building sophisticated document generation or information presentation systems and wants to learn about new technologies developed by the NLG community. We hope that students will be as fascinated as we are by the intellectual problems and insights of NLG and that developers will be able to exploit the techniques we describe to improve the systems they build.

1.1 The Research Perspective

From a research perspective, NLG is a subfield of natural language processing (NLP), which in turn can be seen as a subfield of both computer science and cognitive science. The relation between NLG and other aspects of NLP is further discussed below. From the broader perspective of computer science and cognitive science as a whole, NLG provides an important and unique perspective on many fundamental problems and questions, including the following:

- How should computers interact with people? What is the best way for a machine to communicate information to a human? What kind of linguistic behaviour does a person expect of a computer he or she is communicating with, and how can this behaviour be implemented? These are basic questions in human–computer interaction, an area of computer science which is becoming increasingly important as the quality of computer software is judged more and more by its usability.
- What constitutes 'readable' or 'appropriate' language in a given communicative situation? How can the appropriate pragmatic, semantic, syntactic, and psycholinguistic constraints be formalised? What role does context in its many aspects play in the choice of appropriate language? These issues are basic questions in linguistics, and also important in philosophy and psychology; as such they are core issues in cognitive science.
- How can typical computer representations of information – large amounts of low-level (often numeric) data – be converted into appropriate representations for humans, typically a small number of high-level symbolic concepts? What types of domain and world models and associated reasoning are required to 'translate' information from computer representations to natural language, with its human-oriented vocabulary and structure? These questions are aspects of the larger question of how humanlike intelligence can be modelled and simulated on a computer, which is one of the main goals of artificial intelligence (AI).

Although work in natural language generation can provide insights in these related fields, it also draws on these fields for ideas. For example, many NLG systems use ideas developed within artificial intelligence, such as planning techniques and production rules, to determine the information content of a text; and most NLG systems use formal linguistic models of syntax to ensure that their output text is grammatically correct.

1.1.1 Differences between NL Generation and NL Understanding

Natural language generation is of course closely related to natural language understanding, which is the study of computer systems that understand English and other human languages. Both NL understanding and NL generation are concerned with computational models of language and its use; they share many of the same theoretical foundations and are often used together in application programs. Together, natural language understanding and natural language generation form the field of natural language processing (NLP).

At a rather abstract level, one can think of the process of natural language generation as being the inverse of the process of natural language understanding. Natural language generation is the process of mapping internal computer representations of information into human language, whereas natural language understanding is the process of mapping human language into internal computer representations. Thus, at least in general terms the two processes have the same end points, and their difference lies in the fact that they are concerned with navigating between these end points in opposite directions.

But the internal operations of these processes are quite different in character. Most fundamentally, as has been observed by McDonald (1992), the process of natural language understanding is best characterised as one of HYPOTHESIS MANAGEMENT: Given some input, which of the multiple possible interpretations at any given stage of processing is the appropriate one? Natural language generation is best characterised as a process of CHOICE: Given the different means that are available to achieve some desired end, which should be used?

1.1.2 Sharing Knowledge between Generation and Understanding

Some research in the area has attempted to construct 'reversible' components which can be used in both NLG and NLU systems. This idea has been explored in particular for the mapping between semantic representations and the surface-form sentences that correspond to those representations: a BIDIRECTIONAL GRAMMAR uses a single declarative representation of a language's grammar to perform both parsing in NLU systems (that is, mapping sentences into internal semantic representations) and linguistic realisation in NLG systems (that is, mapping semantic representations into surface sentences). Bidirectional grammars are discussed in Section 6.8.

Unfortunately, despite the elegance and intuitive appeal of the idea, it is difficult to build effective bidirectional systems in practice. This is largely because many important NLU problems are not issues in NLG and vice versa. For example, a major problem in building real NLU systems is the need to be able to handle grammatically incorrect or ill-formed input, which is not a problem in NLG, whereas a major problem in NLG is ensuring that generated text is easily comprehensible by humans, which is not a problem in NLU. A related problem for bidirectional systems is that the internal representations used by NLG and NLU systems are very different. For example, the representations which most NLU parsers produce as output are quite different from the input representations required by most NLG realisers. This basic incompatibility makes it difficult to build a system that does both parsing and realisation.

Bidirectional grammars have been used in machine translation systems, where the input representations are often already in sentence-sized pieces, but they have not been widely used in NLG systems, which generate text from some underlying nonlinguistic representation of information, which is our focus in this book. Of course, this may change in the future, as our understanding of both NLG and NLU grows; it seems intuitively plausible that the same representations of knowledge about language should be used for both generation and analysis. However, it is unlikely that we will ever view NLG as NLU 'in reverse'. Given our current understanding of the processes involved in using language, many of the fundamental tasks of NL generation, such as deciding what information should be communicated in a text, have no clear analogue in NL understanding.

An important difference between NLG and NLU comes from a rather pragmatic aspect of work in the two areas. In very general terms, much of NL understanding is driven by the need to be able to handle the large variety of complex linguistic structures that make up any natural language. This aspect of coverage is of somewhat less importance in NL generation, and indeed the texts produced by most NL generation systems are much simpler in linguistic terms than the texts that NL understanding systems aspire to process. For work in NLG, it is often good enough to have one way of saying something, whereas any experiment in NLU has to take account of the fact that many different inputs may be used to express much the same thing. This again emphasises the difference in characterisation of process we described earlier: Work in NLU is often concerned with clearing away what are thought of as 'superficial differences', so that, for example, *John ate the biscuit* and *The biscuit was eaten by John* receive the same interpretation. Work in NLG is instead often concerned with choosing between such utterances in a principled way.

1.2 The Applications Perspective

From an applications perspective, most current NLG systems are used either to present information to users, or to (partially) automate the production of routine documentation. Information presentation is important because the internal representations used by computer systems often require considerable expertise to interpret. Representations such as airline schedule databases, accounting spreadsheets, expert system knowledge bases, grid-based simulations of physical systems, and so forth are straightforward for a computer to manipulate but not always easy for a person to interpret. This means that there is often a need for systems which can present such data, or summaries of the most important aspects of the data, in an understandable form for nonexpert users. When the best presentation of the data is in English, Spanish, Chinese, or some other human language, NLG technology can be used to construct the presentation system.

Automating document production is important because many people spend large proportions of their time producing documents, often in situations where they do not see document production as their main responsibility. A doctor, for example, may spend a significant part of his or her day writing referral letters, discharge summaries, and other routine documents. Similarly, a computer programmer may spend as much time writing text (code documentation, program logic descriptions, code walkthrough reviews, progress reports, and so on) as writing code. Tools which help such people quickly produce good documents may considerably enhance both productivity and morale.

When we build a complete NLG system, a major design decision that has to be taken is whether that system will operate in standalone mode, generating texts without any human information, or whether it will operate by producing what are in effect drafts of texts that are subsequently modified by a human author. This distinction is essentially the same as that found more broadly in work in expert systems, where it is often deemed appropriate to maintain the presence of a 'human in the loop', especially in contexts where a mistake on the part of the system could be life-threatening or disastrous in some other way. In the case of NLG, the reasons for including a human in the authoring process are generally less dramatic. Put simply, in many contexts it is simply not possible to create texts of the appropriate quality or with the required content without human intervention.

1.2.1 Computer as Authoring Aid

In practice, current NLG technology and the limitations of the information available in host systems mean that it is often not possible for the system to create the final product; instead, the NLG system is used to produce an initial draft of a document which can then be further elaborated or edited by a human author. A variant of this approach is for the NLG system to focus on producing routine factual sections of a document (which human authors often find monotonous to write), leaving analytical and explanatory sections to the human author.

Examples of NLG systems used as authoring assistants are the following:

- **FOG** (Goldberg, Driedger, and Kittredge, 1994), which helps meteorologists compose weather forecasts; see Section 1.3.2.
- **PlanDoc** (McKeown, Kukich, and Shaw, 1994), which helps telephone network engineers document the results of simulations of proposed network changes.
- **AlethGen** (Coch, 1996a), which helps customer-service representatives write response letters to customers.
- **DRAFTER** (Paris et al., 1995), which helps technical authors write software manuals.

This list is indicative only, and by no means complete; numerous other authoring-assistant systems have been described in the NLG literature.

A significant measure of success in technology transfer is the extent to which an idea or system has been actively deployed in the field. As of mid-1998, FOG and PLANDOC are fielded and in everyday use. ALETHGEN has passed the sponsor's acceptance tests but has not yet been deployed. DRAFTER's developers have applied for a patent and intend to commercialise the system once the patent is granted.

1.2.2 Computer as Author

As indicated above, it is probably true to say that most practical systems are required to operate in the 'computer as authoring aid' mode, sometimes for legal or contractual reasons. However, many experimental NLG systems have been developed with the aim of operating as standalone systems that require no human intervention when creating texts. Some such systems are as follows:

- **MODELEXPLAINER** (Lavoie, Rambow, and Reiter, 1997), which generates textual descriptions of classes in an object-oriented software system, using information extracted from a computer-aided software engineering database; see Section 1.3.4.
- **KNIGHT** (Lester and Porter, 1997), which explains information extracted from an artificial intelligence knowledge base.
- **LFS** (Iordanskaja et al., 1992), which summarises statistical data for the general public.
- **PIGLET** (Cawsey, Binstead, and Jones, 1995), which gives hospital patients explanations of information in their patient records.

Again this list is indicative only, and by no means exhaustive.

None of the systems listed above have been fielded yet (as of mid-1998). Given the current state of NLG technology, systems which work in collaboration with a human author are more practical, allowing a symbiotic approach where the machine produces the more routine aspects of documents and leaves the more difficult-to-automate aspects to the human. However, advances in NLG technology should make it possible in the near future to build fieldable standalone systems in some domains, just as mail-merge systems are used to customise letters to individual recipients without a perceived need for each letter to be checked by a human.[1]

1.2.3 Uses of NLG Technology

When developers build a complete NLG system, they do so with a certain application context in mind. The system serves some purpose. Currently, most NLG systems

[1] Of course, stories of such systems generating meaningless letters to deceased persons or other inappropriate categories of recipients abound. The difficulty of predetermining all the relevant circumstances is one very good reason for maintaining the role of a human overseer in systems of any complexity.

are built with the purpose of presenting information to the user in a straightforward and unbiased manner. However, some experimental systems have explored other uses of NLG technology, including the following:

- **Teaching.** The ICICLE system (McCoy, Pennington, and Suri, 1996) helps deaf people learn the syntax of written English.
- **Marketing.** The DYD system (van Deemter and Odijk, 1997) generates descriptions of a music CD which are intended to increase interest in and sales of that CD.
- **Behaviour Change.** The STOP system (Reiter, Robertson, and Osman, 1997) generates personalised letters which encourage people to stop smoking; see Section 1.3.6.
- **Entertainment.** The JAPE system (Binstead and Ritchie, 1997) generates jokes (more specifically, punning riddles).

All of the aforementioned systems are research prototypes. To be really successful in their domains of application, systems such as these need to embody models of how to teach effectively, how to persuade people to buy things, how people break addictive habits, and what people find amusing. Currently we do not have precise computational models in any of these areas, which means that building effective systems is partially a research project in the computational modelling of teaching, persuasion, and so forth, as well as in language generation.

It is perhaps unfortunate that the popular media pay far more attention to systems which generate fictional forms such as jokes, novels, or poetry than to any other kind of NLG system. For example, the JAPE system mentioned above has been reported upon in the UK in both the tabloid and the more serious press, and in radio and television broadcasts. In contrast, most of the NLG systems we describe in this book have never been mentioned in the popular media at all. There is no doubt that 'What happens if you cross a comic with a computer'[2] is a more appealing headline than 'Computer helps telephone engineer document simulation results'; nevertheless, the media's bias may tend to give the lay population the mistaken impression that NLG is mostly concerned with producing fictional material whose purpose is to entertain readers. This is certainly an interesting potential use of NLG technology, but, as should be clear by now, it is by no means the only, or even the primary, use of NLG.

1.3 Some Example NLG Systems

To make our discussion more concrete, in this section we introduce the WEATHER-REPORTER design study and also give brief descriptions of several implemented NLG systems: FOG, IDAS, MODELEXPLAINER, PEBA, and STOP. We will use these

[2] *Sunday Telegraph*, 29 August 1994.

systems throughout the book to illustrate technical points and discussions. We also occasionally use examples from other systems, but we try when possible to base examples on one of the systems described in this section. References for and very brief descriptions of all the NLG systems mentioned in this book are given in Appendix A.

Particular emphasis is placed in the book on the WeatherReporter design study, which is intended to serve as a unifying case study for the book. It is impossible to illustrate all aspects of NLG by means of a single case study, and hence we discuss and draw examples from many other systems as well; but by focusing on a specific system, we hope to give an idea of how the different aspects of NLG fit together.

1.3.1 WeatherReporter

The WeatherReporter design study is used throughout this book as a central case study to illustrate the process of developing an NLG system. We have implemented several prototype versions of various WeatherReporter components using a variety of algorithms and data structures. However, these implementations are early prototypes lacking the maturity and robustness of systems such as FOG, IDAS, ModelExplainer, and PEBA. We describe how corpus analysis is used to analyse the requirements of WeatherReporter and to construct a domain model and message definitions. We also illustrate various implementation options with regard to algorithms and data structures using examples from the WeatherReporter domain.

The purpose of WeatherReporter is to provide retrospective reports of the weather over periods whose duration is one calendar month. It does this by taking as input a large set of numerical data collected automatically by meteorological devices, from which it produces short texts of one or two paragraphs in length. The application is quite simple in many regards, thus allowing us to demonstrate how the techniques presented in this book can be used while avoiding the relative complexity of more sophisticated systems.

Figure 1.1 shows an example text that might be produced by WeatherReporter. This is fairly typical of the kinds of texts required. It provides information on temperature and rainfall for the month as a whole, and it provides descriptions of spells of weather that are noteworthy for one reason or another. Some of the texts can be quite different from this general form, depending on the nature of the data to be reported; these variations are explored in more detail later in the book.

> The month was cooler and drier than average, with the average number of rain days. The total rain for the year so far is well below average. There was rain on every day for eight days from the 11th to the 18th.

Figure 1.1 The Macquarie weather summary for February 1995.

Figure 1.2 shows a fragment of the data that WEATHERREPORTER uses as input. Each line represents one data record, these being collected at 15-minute intervals by an automatic weather station.

Retrospective weather reports similar to the one shown in Figure 1.1 are currently written manually for the staff newsletter of Macquarie University, where one of us (Dale) works, and the data shown in Figure 1.2 is real data collected automatically by meteorological data gathering equipment on the university campus. The WEATHERREPORTER design study is thus based on real input data and a real corpus of human-written texts, and our description of WEATHERREPORTER in this book shows how humanlike texts can be generated from real input data using NLG techniques.

1.3.2 FOG

The FOG system (Goldberg et al., 1994) generates textual weather forecasts from numerical weather simulations produced by a supercomputer and annotated by a human forecaster. More precisely, it takes as input a numerical prediction of how wind speeds, precipitation type and intensity, and other meteorological phenomena vary over a given region in a specified time interval, and produces as output a textual summary of this information. An example of a graphical rendition of some of FOG's input is shown in Figure 1.3, which depicts the predicted weather system over Northern Canada at 0600 GMT on 14 August 1998. An example of FOG's output is shown in Figure 1.4; this is a marine forecast issued at 04:25 EDT (09:25 GMT) on 13 August 1998, which describes the predicted weather over various points in Northern Canada. Because it is a marine forecast, it emphasises wind and precipitation information and says nothing about temperature; other types of forecasts would emphasise different information.

The texts produced by FOG are not very exciting, but they are useful and more difficult to generate than might at first seem to be the case. One complexity in FOG that may not be immediately apparent is that it must decide how detailed the information it provides should be. In the example in Figure 1.4, for instance, FOG has decided to produce a single summary forecast that covers both East Brevoort and East Davis but a separate forecast for East Clyde. Also, FOG needs to decide when it can use inexact temporal terms like *late this evening* or *early Friday evening* and when it should use a more precise phrase such as *at midnight*.

Another technically interesting aspect of FOG is that it can produce forecast texts in both English and French; in other words, it is a MULTILINGUAL system. Internally, FOG first produces a language-independent abstract representation of the forecast text and then maps this into each output language using appropriate grammatical and lexical resources.

The FOG system was developed by CoGenTex, a specialist NLG software house, for Environment Canada (the Canadian weather service). It has been in everyday use since 1993.

```
96,122,1,5,2.00,200,-14.41,-3.668,-1.431,.345,1023,15.41,15.82,20.07,-11.1,-2.878,104.2,.28,153.6,53.19,0,16.26
96,122,1,5,2.25,215,-10.72,-3.241,-1.35,.152,1023,15.3,15.78,20.07,-11.42,-2.762,105,.208,98.2,.822,0,17.05
96,122,1,5,2.50,230,-8.37,-1.282,-.904,2.15,1022,15.3,15.71,20.05,-11.66,-3.206,104.4,.2,141.6,42.96,0,17.7
96,122,1,5,2.75,245,-12.81,-2.11,-1.067,2.119,1022,15.33,15.79,19.99,-11.15,-3.093,104.8,.2,186.5,11.32,0,17.81
96,122,1,5,3.00,300,-13.68,-3,-1.35,1.075,1022,15.36,15.79,19.96,-10.63,-3.005,104.6,.402,285.8,61.45,0,18.47
96,122,1,5,3.25,315,-10.2,-2.457,-1.13,-.73,1022,15.32,15.66,19.92,-11.17,-3.263,103.6,.304,354.7,36.29,0,19.03
96,122,1,5,3.50,330,-9.33,-1.353,-.942,.902,1022,15.21,15.62,19.9,-10.95,-2.903,104.3,.313,302.2,34.69,0,19.16
96,122,1,5,3.75,345,-7.29,-.285,-.76,2.048,1022,15.24,15.63,19.87,-10.68,-3.27,104,.252,313,29.7,0,19.61
96,122,1,5,4.00,400,-6.822,-.365,-.653,1.531,1022,15.25,15.63,19.83,-9.93,-3.316,104,.331,274.2,52.98,0,20.42
96,122,1,5,4.25,415,-8.78,-.65,-.747,1.602,1023,15.35,15.66,19.79,-9.77,-2.656,103.3,.253,247.7,10.99,0,21.08
96,122,1,5,4.50,430,-8.73,-.641,-.741,1.785,1023,15.46,15.81,19.75,-9.16,-2.782,103.7,.2,295,29.15,0,21.3
96,122,1,5,4.75,445,-11.45,-2.671,-1.03,-.456,1022,15.46,15.82,19.74,-8.81,-2.464,103.7,.2,355.3,23.98,0,21.65
96,122,1,5,5.00,500,-13.12,-4.3,-1.306,-1.359,1022,15.42,15.75,19.76,-9.39,-2.49,103.4,.2,20.67,.188,0,21.83
96,122,1,5,5.25,515,-13.62,-4.621,-1.344,-.842,1022,15.32,15.67,19.81,-9.47,-2.703,103.7,.2,20.65,.183,0,21.98
96,122,1,5,5.50,530,-13.8,-3.534,-1.325,.943,1022,15.23,15.61,19.86,-10.92,-3.384,103.9,.2,20.65,.183,0,22.14
96,122,1,5,5.75,545,-14.7,-3.748,-1.419,.385,1022,15.06,15.47,19.9,-11.62,-2.868,104.4,.2,341.6,18.6,0,22.36
96,122,1,5,6.00,600,-13.61,-2.315,-1.287,2.038,1022,14.98,15.42,19.9,-12.37,-3.092,104.7,.2,298.6,5.173,0,22.54
96,122,1,5,6.25,615,-14,-2.894,-1.293,.669,1022,14.92,15.36,19.88,-12.48,-3.808,104.7,.591,320.3,21.07,0,22.87
```

The 22 data values in each record are as follows: the year; the day of the year as a value between 1 and 365; the month of the year; the day of the month; the time in 24 hour notation; the time in hours and minutes; four radiation readings; the barometric pressure; the temperature, referred to as ‘dry bulb temperature’; the wet bulb temperature, which is used to calculate humidity; the soil temperature; the soil heat flux at two depths; relative humidity; average wind speed; wind direction; the standard deviation of wind direction; precipitation within the 15 minute period; and amount of sunlight.

Figure 1.2 Data input for the WeatherReporter system.

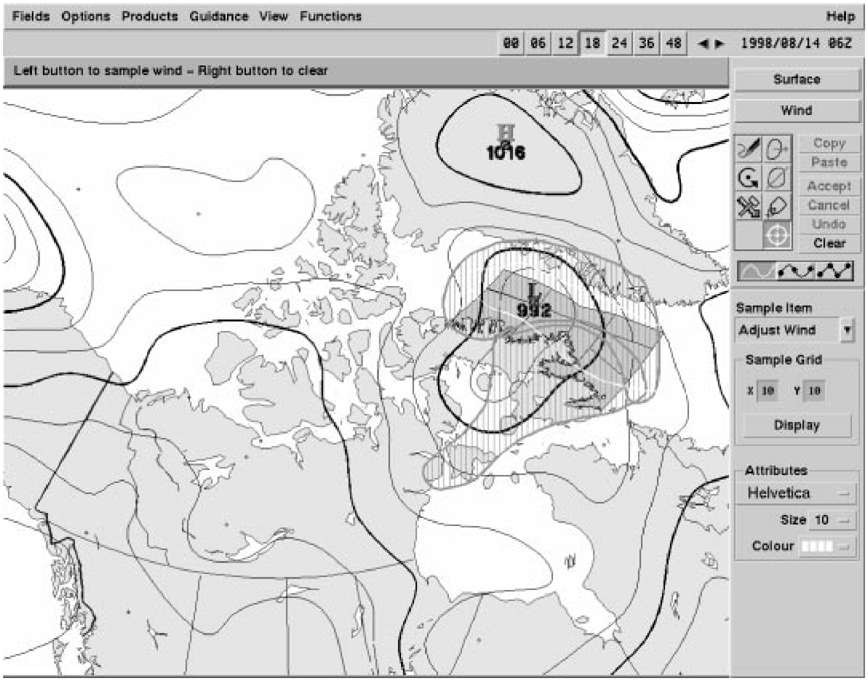
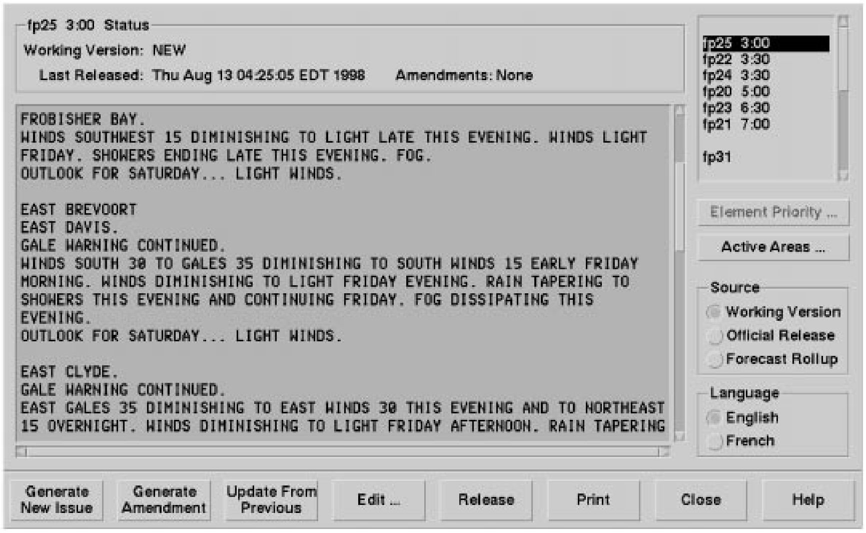


Figure 1.3 FOG input: a weather system over Canada. (Courtesy of Environment Canada.)



```
FROBISHER BAY.
WINDS SOUTHWEST 15 DIMINISHING TO LIGHT LATE THIS EVENING. WINDS LIGHT
FRIDAY. SHOWERS ENDING LATE THIS EVENING. FOG.
OUTLOOK FOR SATURDAY... LIGHT WINDS.

EAST BREVOORT
EAST DAVIS.
GALE WARNING CONTINUED.
WINDS SOUTH 30 TO GALES 35 DIMINISHING TO SOUTH WINDS 15 EARLY FRIDAY
MORNING. WINDS DIMINISHING TO LIGHT FRIDAY EVENING. RAIN TAPERING TO
SHOWERS THIS EVENING AND CONTINUING FRIDAY. FOG DISSIPATING THIS
EVENING.
OUTLOOK FOR SATURDAY... LIGHT WINDS.

EAST CLYDE.
GALE WARNING CONTINUED.
EAST GALES 35 DIMINISHING TO EAST WINDS 30 THIS EVENING AND TO NORTHEAST
15 OVERNIGHT. WINDS DIMINISHING TO LIGHT FRIDAY AFTERNOON. RAIN TAPERING
```

Figure 1.4 Some example forecasts from FOG. (Courtesy of Environment Canada.)

1.3.3 IDAS

The IDAS system (Reiter, Mellish, and Levine, 1995) produces on-line hypertext help messages for users of complex machinery, using information stored in a knowledge base that describes that machinery. Figure 1.5 shows a series of texts produced by IDAS in response to user queries; in this case the machine being described is a bicycle. All underlined words in the system's responses are hypertext links which can be clicked on; each click results in a new text being produced.[3] For example, if the user clicks on *front gear mechanism* in the *What are the parts of the gearing system* box (in the top-left corner of Figure 1.5), she will see a pop-up menu which lists a number of questions she can ask about this device, including *What is its purpose*. Selecting this question results in the system producing a text on *What is the purpose of the front gear mechanism*; this box is shown in the middle of the top row of boxes in Figure 1.5. The bold font words at the bottom of each box enable further questions to be asked about the same component. For example, clicking on USE in the *What is the purpose of the front gear mechanism* box results in a text on *How do I use the front gear mechanism*; this box is shown in the top-right corner of Figure 1.5. IDAS is a DYNAMIC HYPERTEXT system, in that all texts are generated dynamically from the knowledge base at run-time. The links are not pointers to static files, but rather requests to run a program (IDAS) to generate an appropriate response in the current context.

IDAS can vary the REFERRING EXPRESSIONS it uses to refer to objects according to context. For example, consider the response text for *How do I use the front gear mechanism* (top-right box in Figure 1.5):

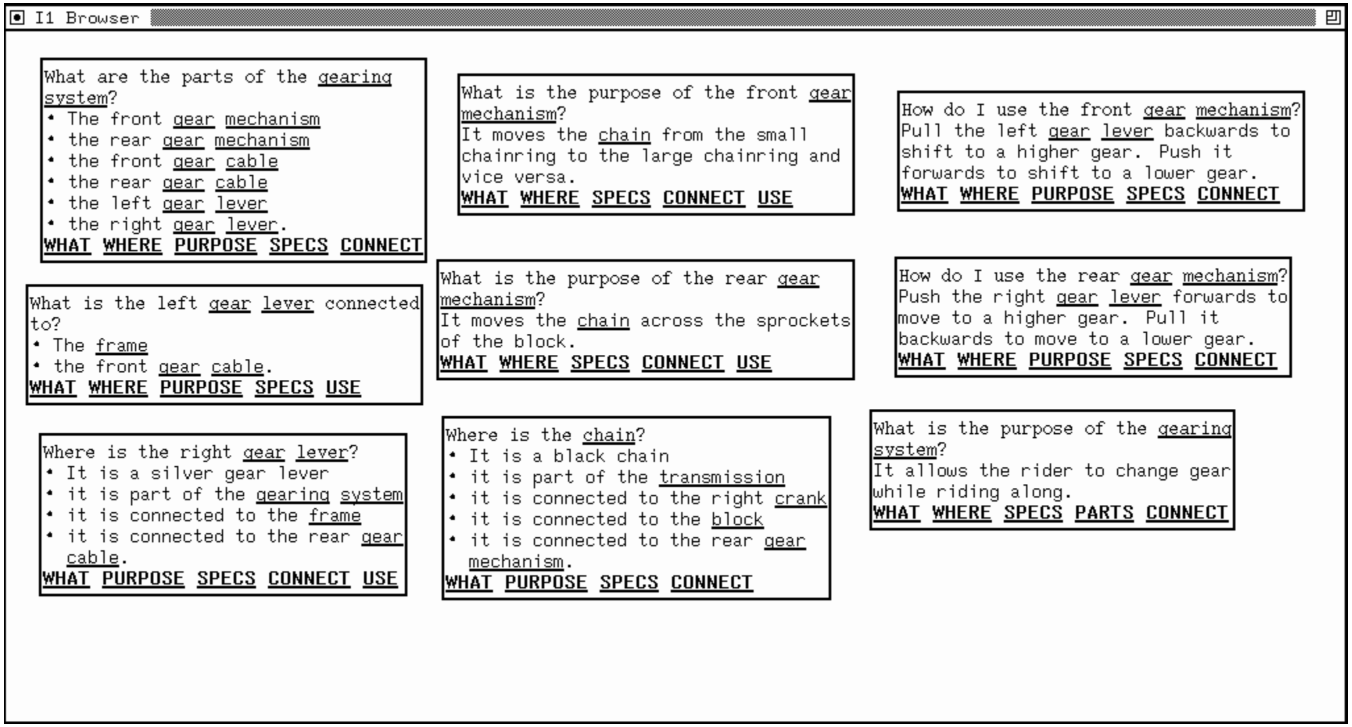
(1.1) Pull the left gear lever backwards to shift to a higher gear. Push it forwards to shift to a lower gear.

In the second sentence of this example, IDAS has used the pronoun *it* to refer to the left gear lever of the bicycle; this is possible because this object was introduced in the first sentence of this text.

From a technical perspective, IDAS is interesting for its use of hypertext, and for the way it selects appropriate referring expressions. With some knowledge bases (but not the bicycle KB used to produce Figure 1.5, which is relatively simple), it can also vary response texts according to a user model which describes the user's expertise and task.

The IDAS system was developed as a research prototype at the University of Edinburgh in the early 1990s. It was never put into everyday use, but many of the ideas developed in IDAS have influenced later NLG systems.

[3] The hypertext interface used in IDAS is somewhat primitive, since the system predates the World Wide Web and its associated hypertext browsers.



```
I1 Browser

What are the parts of the gearing
system?
• The front gear mechanism
• the rear gear mechanism
• the front gear cable
• the rear gear cable
• the left gear lever
• the right gear lever.
WHAT WHERE PURPOSE SPECS CONNECT

What is the left gear lever connected
to?
• The frame
• the front gear cable.
WHAT WHERE PURPOSE SPECS USE

Where is the right gear lever?
• It is a silver gear lever
• it is part of the gearing system
• it is connected to the frame
• it is connected to the rear gear
  cable.
WHAT PURPOSE SPECS CONNECT USE

What is the purpose of the front gear
mechanism?
It moves the chain from the small
chainring to the large chainring and
vice versa.
WHAT WHERE SPECS CONNECT USE

What is the purpose of the rear gear
mechanism?
It moves the chain across the sprockets
of the block.
WHAT WHERE SPECS CONNECT USE

Where is the chain?
• It is a black chain
• it is part of the transmission
• it is connected to the right crank
• it is connected to the block
• it is connected to the rear gear
  mechanism.
WHAT PURPOSE SPECS CONNECT

How do I use the front gear mechanism?
Pull the left gear lever backwards to
shift to a higher gear.  Push it
forwards to shift to a lower gear.
WHAT WHERE PURPOSE SPECS CONNECT

How do I use the rear gear mechanism?
Push the right gear lever forwards to
move to a higher gear.  Pull it
backwards to move to a lower gear.
WHAT WHERE PURPOSE SPECS CONNECT

What is the purpose of the gearing
system?
It allows the rider to change gear
while riding along.
WHAT WHERE SPECS PARTS CONNECT
```

Figure 1.5 Some example texts from IDAS. (Courtesy of John Levine.)

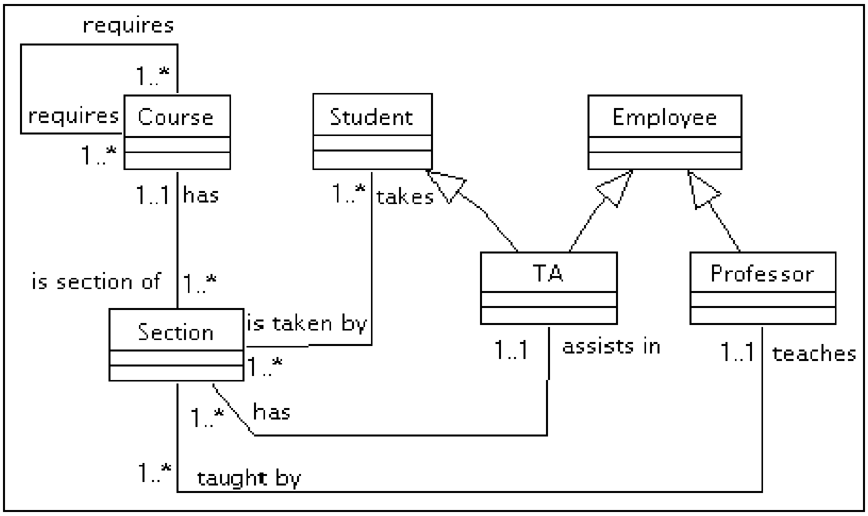


Figure 1.6 An example MODELEXPLAINER input: an object-oriented class model. (Courtesy of CoGenTex, Inc.)

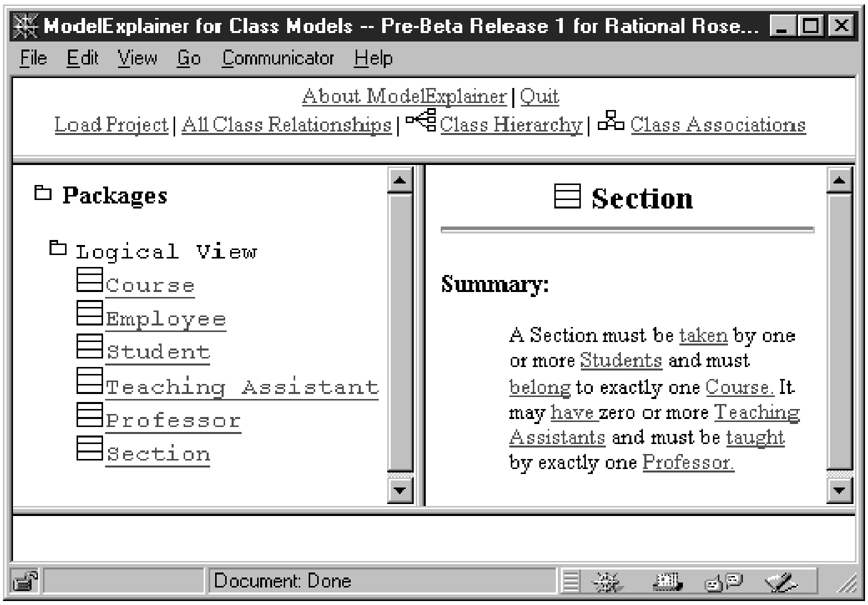
1.3.4 ModelExplainer

The MODELEXPLAINER system (Lavoie et al., 1997) generates textual descriptions of information in models of object-oriented (O-O) software. More specifically, it takes as input a specification of an object-oriented class model and produces as output a text describing the model or portions of it. An example of MODELEXPLAINER's output is shown in Figure 1.7; this describes the class Section in the O-O model which is graphically shown in Figure 1.6.

O-O models are usually depicted graphically, and MODELEXPLAINER is intended to be used as a supplement to, rather than as a replacement for, graphical depictions; this is useful because certain kinds of information are better communicated textually. For example, the text in Figure 1.7 makes it clear to the reader that a Section must be taught by exactly one Professor; this is perhaps less immediately obvious in the graphical depiction of this object model, especially for people who are not familiar with the notation used in the graphical depiction.

MODELEXPLAINER is a highly customisable system. Whereas most NLG systems can be modified only by their developers, MODELEXPLAINER also allows users (or system administrators at user organisations) to modify the content of its descriptions. Like IDAS, it makes extensive use of hypertext; all underlined words in the output text shown are hypertext links on which the user can click.

From a technical perspective, among the many interesting aspects of MODELEXPLAINER are the AGGREGATION processes it performs in order to produce sentences which contain several clauses. An example of this is the first sentence in the


ModelExplainer for Class Models -- Pre-Beta Release 1 for Rational Rose...
File Edit View Go Communicator Help
About ModelExplainer | Quit
Load Project | All Class Relationships | Class Hierarchy | Class Associations
Packages
Logical View
Course
Employee
Student
Teaching Assistant
Professor
Section
Section
Document: Done

Summary:

A Section must be taken by one or more Students and must belong to exactly one Course. It may have zero or more Teaching Assistants and must be taught by exactly one Professor.

Figure 1.7 An example description produced by MODELEXPLAINER from the model in Figure 1.6. (Courtesy of CoGenTex, Inc.)

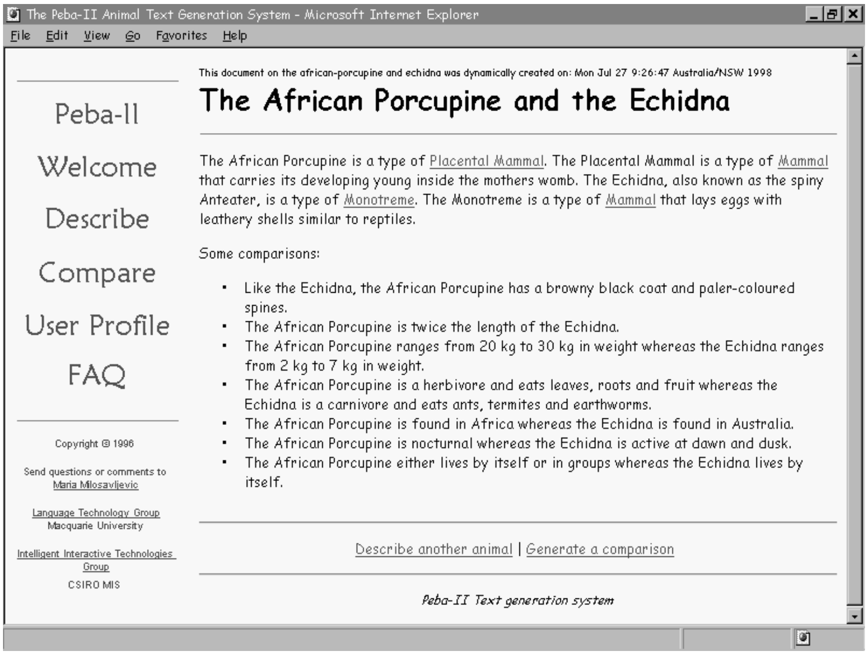
Summary section of Figure 1.7, which was produced by aggregating the information that might otherwise have been presented in the separate sentences *A Section must be taken by one or more students* and *A Section must belong to exactly one Course*. Another interesting aspect of MODELEXPLAINER is that it must express relations from the object model (such as Teaches) in a variety of linguistic contexts; for example, *a Professor* <u>*teaches*</u> *a course*, *a Section* <u>*must be taught by*</u> *a Professor*, and *Professor Smith* <u>*does not teach*</u> *any Sections*.

MODELEXPLAINER was developed by CoGenTex, the same company which developed the FOG system.

1.3.5 PEBA

PEBA (Milosavljevic and Dale, 1996) is a natural language generation system which interactively describes entities in a taxonomic knowledge base via the dynamic generation of hypertext documents, presented as World Wide Web pages.

In PEBA, the information provided to the user varies depending upon the context of use: The system produces different texts for novice and expert users and varies the content of the text presented depending on what material the user has seen before. Figure 1.8 shows a Web page generated by PEBA; this provides a comparison of two objects in the knowledge base, as requested by the user. The


The Peba-II Animal Text Generation System - Microsoft Internet Explorer
File Edit View Go Favorites Help
Peba-II
Welcome
Describe
Compare
User Profile
FAQ
Copyright © 1996
Send questions or comments to Maria Milosavljevic
Language Technology Group
Macquarie University
Intelligent Interactive Technologies Group
CSIRO MIS
Describe another animal | Generate a comparison
Peba-II Text generation system

This document on the african-porcupine and echidna was dynamically created on: Mon Jul 27 9:26:47 Australia/NSW 1998

The African Porcupine and the Echidna

The African Porcupine is a type of Placental Mammal. The Placental Mammal is a type of Mammal that carries its developing young inside the mothers womb. The Echidna, also known as the spiny Anteater, is a type of Monotreme. The Monotreme is a type of Mammal that lays eggs with leathery shells similar to reptiles.

Some comparisons:

- Like the Echidna, the African Porcupine has a browny black coat and paler-coloured spines.
- The African Porcupine is twice the length of the Echidna.
- The African Porcupine ranges from 20 kg to 30 kg in weight whereas the Echidna ranges from 2 kg to 7 kg in weight.
- The African Porcupine is a herbivore and eats leaves, roots and fruit whereas the Echidna is a carnivore and eats ants, termites and earthworms.
- The African Porcupine is found in Africa whereas the Echidna is found in Australia.
- The African Porcupine is nocturnal whereas the Echidna is active at dawn and dusk.
- The African Porcupine either lives by itself or in groups whereas the Echidna lives by itself.

Figure 1.8 A WWW page generated by PEBA. (Courtesy of Maria Milosavljsevic.)

underlying knowledge base contains information about individual objects – here, this is a representation of a fragment of the Linnaean taxonomy of animals – expressed in a simple knowledge representation formalism that permits the system to automatically construct descriptive texts. The system was developed to explore what the next generation of intelligent encyclopaedias might look like. By taking into account some characterisation of the user's background knowledge, and also by maintaining a record of the information already presented to the user, the system can tailor texts appropriately; this means that a description of an object may be different for different people, and may vary depending on the path they have taken through the hypertext system. The ability to automatically generate arbitrary comparisons of pairs of entities on demand means that the system can generate many more individualised texts than it would be feasible to write by hand.

The PEBA system can be accessed on the Web at `http://www.mri.mq.edu.au/peba`.

1.3.6 STOP

STOP (Reiter et al., 1999) is a natural language generation system which produces personalised smoking-cessation letters. Personalisation is based on an 'Attitudes

Towards Smoking' questionnaire which smokers fill out; this includes questions about topics such as health problems, previous attempts to quit, and what the person likes and dislikes about smoking.

STOP was still under development at the time this book was written. Figure 1.9 shows a page generated by a prototype version of STOP.[4] This letter was generated for a smoker who does not currently intend to quit but who would quit if quitting were an easy process – in short, someone who would like to quit but doesn't feel she will be able to, and hence is not planning to try. In this case, STOP gently encourages the smoker to consider making another attempt to quit, by

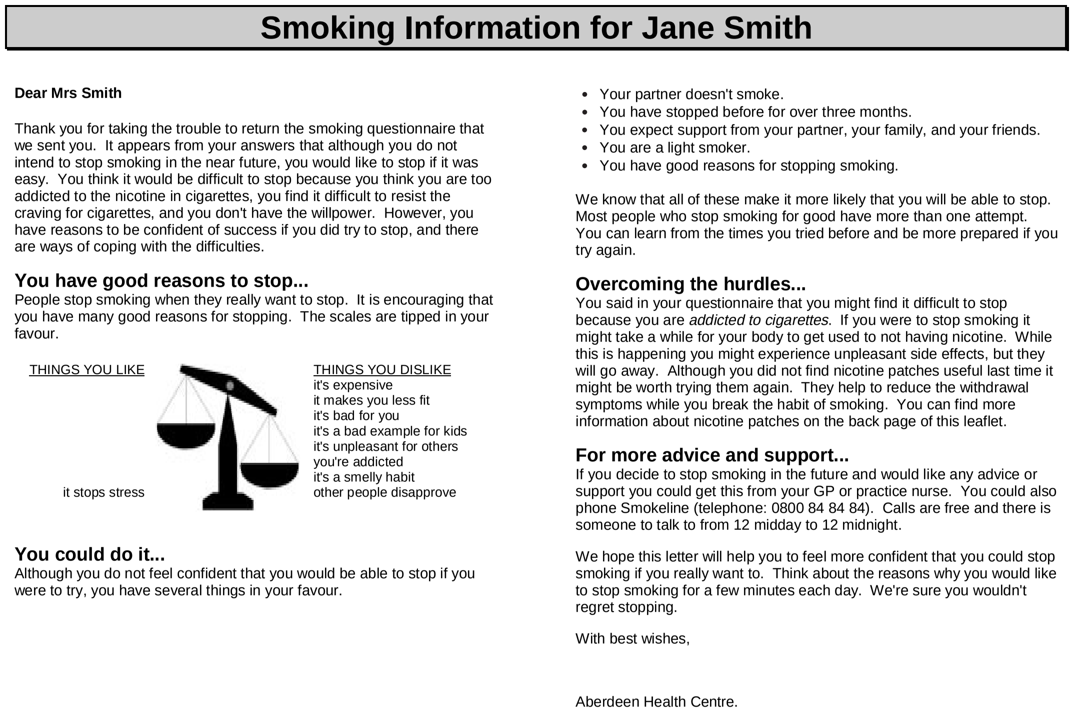
- pointing out to her that she dislikes many more things about smoking than she likes, using information about likes and dislikes extracted from the questionnaire;
- emphasising that she can quit if she really wants to, even if she has failed in the past;
- giving some practical advice about addiction, which is a special concern to this smoker (this includes an additional nontailored 'back page', not shown in Figure 1.9, on nicotine patches); and
- pointing her towards further sources of advice and help.

STOP deliberately does not hector or argue, since this may cause some people to 'switch off'. Instead, it adopts a positive tone, stressing benefits of quitting, encouraging people who lack confidence, addressing problems or concerns the smoker may have (for example, addiction, or fear of gaining weight after quitting), and giving practical advice.

From a technical perspective, perhaps the most interesting aspect of STOP is the knowledge acquisition techniques that the developers used to interact with the domain experts. Based on techniques developed by researchers on expert systems, these included protocol analysis, sorting, and role playing. STOP is also interesting for its use of typographic and multimodal elements such as different typefaces, itemised lists, and embedded graphics.

STOP will be evaluated in a controlled clinical study begun in late 1998; this will measure how many recipients of STOP letters quit as compared to control groups who received a nonpersonalised letter or no letter at all (but who did fill out a questionnaire).

[4] In fact, STOP generates four pages of output, not two: an initial page which contains various logos, a salutation, and an introductory paragraph; two pages which contain personalised text; and a final 'back' page which is selected from one of a dozen possible back pages but not otherwise personalised. To simplify the exposition, we show here only the personalised text produced by STOP in Figure 1.3.6; we have omitted the introductory logos and back page and have presented the salutation and introductory paragraph on the same pages as the rest of the personalised text.



Smoking Information for Jane Smith

Dear Mrs Smith

Thank you for taking the trouble to return the smoking questionnaire that we sent you. It appears from your answers that although you do not intend to stop smoking in the near future, you would like to stop if it was easy. You think it would be difficult to stop because you think you are too addicted to the nicotine in cigarettes, you find it difficult to resist the craving for cigarettes, and you don't have the willpower. However, you have reasons to be confident of success if you did try to stop, and there are ways of coping with the difficulties.

You have good reasons to stop...

People stop smoking when they really want to stop. It is encouraging that you have many good reasons for stopping. The scales are tipped in your favour.

THINGS YOU LIKE		THINGS YOU DISLIKE
		it's expensive
		it makes you less fit
		it's bad for you
		it's a bad example for kids
		it's unpleasant for others
		you're addicted
		it's a smelly habit
it stops stress		other people disapprove

You could do it...

Although you do not feel confident that you would be able to stop if you were to try, you have several things in your favour.

- Your partner doesn't smoke.
- You have stopped before for over three months.
- You expect support from your partner, your family, and your friends.
- You are a light smoker.
- You have good reasons for stopping smoking.

We know that all of these make it more likely that you will be able to stop. Most people who stop smoking for good have more than one attempt. You can learn from the times you tried before and be more prepared if you try again.

Overcoming the hurdles...

You said in your questionnaire that you might find it difficult to stop because you are *addicted to cigarettes*. If you were to stop smoking it might take a while for your body to get used to not having nicotine. While this is happening you might experience unpleasant side effects, but they will go away. Although you did not find nicotine patches useful last time it might be worth trying them again. They help to reduce the withdrawal symptoms while you break the habit of smoking. You can find more information about nicotine patches on the back page of this leaflet.

For more advice and support...

If you decide to stop smoking in the future and would like any advice or support you could get this from your GP or practice nurse. You could also phone Smokeline (telephone: 0800 84 84 84). Calls are free and there is someone to talk to from 12 midday to 12 midnight.

We hope this letter will help you to feel more confident that you could stop smoking if you really want to. Think about the reasons why you would like to stop smoking for a few minutes each day. We're sure you wouldn't regret stopping.

With best wishes,

Aberdeen Health Centre.

Figure 1.9 A letter generated by the prototype STOP system.