Automatic Generation of On-Line Documentation in the IDAS Project*

Ehud Reiter, Chris Mellish, and John Levine
Department of Artificial Intelligence
University of Edinburgh
80 South Bridge
Edinburgh EH1 1HN BRITAIN
(e-mail: e.reiter@ed.ac.uk)

Abstract

The Intelligent Documentation Advisory System generates on-line documentation and help messages from a domain knowledge base, using natural-language (NL) generation techniques. This paper gives an overview of IDAS, with particular emphasis on: (1) its architecture and the types of questions it is capable of answering; (2) its KR and NL generation systems, and lessons we have learned in designing them; and (3) its hypertext-like user interface, and the benefits such an interface brings.

1 Introduction

The Intelligent Documentation Advisory System (IDAS) project is attempting to use natural-language (NL) generation and hypertext technology to produce an on-line documentation and help system that supports users of complex machinery. In this paper we present an overview of the most recent IDAS prototype developed at the University of Edinburgh, including descriptions of:

- IDAS's overall architecture, and particularly its *question space*, i.e., the set of queries it is designed to answer;
- IDAS's KR and NL generation components, with particular emphasis on lessons we have learned while building them, and related design decisions;
- IDAS's hypertext-like user interface, and the operations it is intended to support.

*The IDAS project is partially funded by UK SERC grant GR/F/36750 and UK DTI grant IED 4/1/1072, and we are grateful to SERC and DTI for their support of this work. We would also like to thank the IDAS industrial collaborators — Inference Europe, Ltd.; Racal Instruments, Ltd.; and Racal Research, Ltd. — for all the help they have given us in performing this research. Thanks also to Robert Dale and the anonymous reviewers for their very helpful comments

IDAS is a collaborative effort between the University of Edinburgh, Racal Instruments Ltd., Racal Research Ltd., and Inference Europe Ltd. As this paper is written, the project is about half-way through its 3-year lifespan. Several prototypes have been built to date; this paper describes the most recent one built at the University of Edinburgh, which consists of about 5000 lines of Lisp code. Work on IDAS continues as this paper is being written, with current tasks including the expansion of the existing domain knowledge base, and the integration of the documentation software with the actual hardware being documented. We have not yet carried out any formal evaluations of IDAS, although we hope to arrange such tests once the current expansion and integration tasks are completed; the existing system has been shown to many people informally, generally with quite favorable reactions.

The initial IDAS system documents an ATE (Automatic Test Equipment), a complex device made by Racal Instruments for testing potentially faulty electronic devices. The ATE contains an assortment of electronic instruments, a switching system that connects these instruments to a UUT (Unit Under Test), and a computer which runs test programs that test the UUT with the instruments. Potential IDAS users include operators who use the ATE to test UUTs; maintenance technicians who look for faults in the ATE itself; and programmers who create test programs. The current IDAS prototype is designed to support operators and maintenance technicians; support for programmers may be added later.

2 Architecture

A simplified version of IDAS's architecture is shown in Figure 1. Textual output from test programs and other ATE software is intercepted by the *Listener*,[1] which detects mentions of ATE components and extracts information about the user's task (e.g., what test program

[1]The Listener has not yet been implemented in the prototype.

Figure 1: Simplified IDAS architecture

he is running). Mentioned components are added to the discourse in-focus list, and are also made mousable in the output window; if the user clicks on one, he invokes IDAS and the Listener creates an initial query about that component, i.e., an initial point in *question space* (Section 2.1). The question space point is given to IDAS's NL generation system, which generates a response using three modules: *content determination*, which picks relevant information out of the knowledge base to communicate to the user; *text planning*, which converts this information into an expression in SPL, the ISI Sentence Planning Language [Kasper, 1989]; and *surface realization*, which produces a surface form, i.e., an annotated text string.

The annotations consist of text-formatting commands (e.g., Begin-New-Line) and hypertext specifications. The annotated text string is given to the *Hypertext Interface* system, which presents it to the user in a hypertext window; this window also includes buttons for *hyperschema* follow-up questions (Section 4.1). If the user clicks on a mouse-sensitive word or a button, the point in question space that corresponds to this query is passed to the NL generation system, and the process iterates.

2.1 Question Space

Question space is the set of queries that can be given to IDAS's NL generation system; IDAS's hypertext system can be viewed as a tool that enables a user to move around question space until he finds a point that gives him the information he is looking for. A point in question-space is a tuple with five components:

- *Basic-question:* Currently includes What-is-it, Where-is-it, What-is-its-purpose, What-are-its-specifications, What-are-its-parts, What-is-it-connected-to, How-do-I-perform-the-task.[2]
- *Component:* the target of the question, e.g., **Printer-36** or **Computer-3**; components are usually physical ATE components, but can in some cases be actions or other knowledge-base entities.
- *Task:* The user's task, e.g., Operations or Replace-Part.
- *User-Expertise:* The user's expertise level, e.g., Novice or Skilled.
- *Discourse-in-focus:* The set of in-focus objects for referring expression generation [Grosz and Sidner, 1986].

For example, the question space point ⟨What-is-it, **DC-Power-Supply-23**, Operations, Skilled, {**VXI-Chassis-36**, **DC-Power-Supply-23**}⟩ represents the query "What is the DC Power Supply" when asked by a user of Skilled expertise who is engaged in an Operations task with the discourse context containing the

[2]How-do-I-perform-the-task is interpreted as How-do-I-use-it for Operations tasks, How-do-I-replace-it for Replace-Part tasks, etc.

objects **VXI-Chassis-36** and **DC-Power-Supply-23**. The NL Generation component would in this case produce the response

> "The DC power supply is a black Elgar AT-8000 DC power supply."

Variations in the above tuple would be processed as follows:

- *Component:* If a different component had been specified, IDAS would have generated another response that communicated colour, manufacturer, and model-number information, as specified by the content-determination rule for What-is-it questions asked during Operations tasks (Section 3.2). For example, if the component had been **Printer-12**, the generated text would have been

  > "The printer is a white Epson LQ-1010 printer."

- *Basic Question:* A different response pattern (i.e., content-determination rule) would have been used for a different basic question. For example, if the basic question had been What-is-its-purpose, the response would have been

  > "The DC power supply provides DC power for the UUT."

- *Task:* A different response pattern would also have been used if a different task had been specified. For example, for the What-is-it question, if the user's task had been Replace-Part instead of Operations, colour would have been omitted but a part number would have been included, e.g.,

  > "The DC power supply is an Elgar AT-8000 DC power supply with part number OPT-EP2."

- *User-Expertise:* The What-is-its-purpose response would have been phrased differently if the user's expertise level had been Novice instead of Skilled: 'unit under test' would have been used instead of 'UUT', 'power' instead of 'DC power', and 'the black power supply' instead of 'the DC power supply', giving:

  > "The black power supply provides power for the unit under test."

- *Discourse-in-focus:* The discourse-in-focus list does not affect the above responses, but it would affect the response to Where-is-it. The response to Where-is-it under the original discourse-in-focus list would have been:

  > "The DC power supply is below the VXI chassis."

  If the discourse-in-focus list had included **Mains-Control-Unit-29** instead of **VXI-Chassis-36**, the location would have been given relative to the mains-control-unit instead of the VXI-chassis, i.e., the text would have been:

  > "The DC power supply is above the mains control unit."

Question space is quite large: the current prototype has 40 components, 7 basic questions, 6 user-tasks, and 3 user-expertise models, so there are over 5000 points in its question space even if variations in the discourse context are ignored.[3] A more realistically sized system would document several hundred components and probably would have additional user-task and user-expertise models as well; its question space could therefore easily contain several hundred thousand points. Many points in question space represent queries that produce the same text (e.g., responses to Where-is-it do not depend on the user's task); even if only 10% of the points in question space produce distinct responses, however, this still means that a realistically-sized IDAS system must be able to generate tens of thousands of different responses. The justification for using natural language generation in IDAS is that it would be difficult to enter 20,000 different canned text responses for 200,000 different queries, and almost impossible to maintain this documentation database as new ATE configurations were announced; using NL generation from a domain knowledge base accompanied by explicit task, expertise, and discourse models makes it feasible to supply appropriate answers for this multitude of possible queries.

3 KR and NLG

The fundamental purpose of IDAS's knowledge representation (KR) and natural-language generation (NLG) components is to represent domain information in a more efficient form than thousands of canned text responses. For example, a component's model number will typically appear in at least 30 different query responses (i.e., question space points); representing it in the knowledge base and using NLG to produce text from the knowledge base allows the documenter to enter (and update) this information only once, instead of 30 times.

Many of the theoretically interesting aspects of IDAS's KR and NLG systems are discussed elsewhere, e.g., [Reiter and Mellish, 1992; Reiter and Dale, 1992]. Here, we

[3] The prototype's knowledge base is not complete; currently only about 2/3 of the potential queries can be answered.

present a brief overview of the KR and NLG systems, and then discuss three design decisions that we made during the course of development: allowing canned text and other 'cheats'; generating short and focused replies; and stressing authorability instead of deep reasoning in content-determination. These decisions were not part of the original IDAS design, but rather were made as a result of experience in developing prototypes and interacting with our industrial collaborators; hence, they are 'lessons' we have learned that may be of interest to other researchers and developers working on similar applications-oriented projects.

3.1 Knowledge Representation

The IDAS knowledge-representation system uses a KL-ONE type taxonomy [Brachman and Schmolze, 1985] to represent domain entities (e.g., companies, ATE components, user actions) and linguistic knowledge (grammatical units, lexical definitions, etc.). The knowledge-base is supported by a KL-ONE-like automatic classifier; a superclass-to-subclass attribute inheritance system, based on Touretzky's minimal inferential distance principle [Touretzky, 1986]; and a graphical browse/edit tool. We currently use a small demonstration knowledge base that contains about 200 classes that represent domain entities (e.g., the company **Racal**, the component **Counter-timer**, and the user-action **Clean**), and 50 roles that represent domain attributes (e.g., **colour** and **manufacturer**). The knowledge-base will, of course, need to be substantially enlarged before it is of much use to real users.

The knowledge-base also contains user-expertise and user-task models. The user-expertise models overlay the class taxonomy, and specify what words a user knows and what primitive actions he can execute; they are in some ways similar to the user-models used in the FN system [Reiter, 1990]. The task models do not contain any structure themselves, but affect which content-determination rule is chosen (Section 3.2), and hence the system's decision as to what information the response should communicate to the user. The current prototype contains 3 user-expertise models and 6 task models. More expertise and task models will probably be added with time, but we expect our final system to have at most tens of such models, not hundreds; our objective is provide expertise and task models that are a reasonable fit to most circumstances, not to be able to cover all possible users performing all possible actions.

An authoring tool for the knowledge base is currently being developed by one of our industrial collaborators. Such a tool, which we hope will be directly usable by technical authors and domain experts, is of course vital to the ultimate success of the project.

3.2 Natural Language Generation

Natural-language generation is performed in IDAS in three stages:

Content Determination: The basic-question, component, and user-task components of the question-space tuple are used to pick a *content-determination rule*, which specifies which information from the domain knowledge base should be communicated to the user.

Text Planning: The KB information is turned into an SPL term, in a process which is sensitive to the user-expertise and discourse components of the question-space tuple. This process involves, for example, generating referring expressions and choosing open-class lexical items.

Surface Realization: The SPL term is converted into a surface form, i.e., a set of words with formatting and hypertext annotations. Except for its hypertext-related abilities, the IDAS surface-generation system has a similar functionality to a subset of the PENMAN system [Penman Natural Language Group, 1989].

IDAS's NL generation system is only designed to be able to generate small pieces of text (a few sentences, a paragraph at most). This is because IDAS's hypertext system should enable users to dynamically select the paragraphs they wish to read, i.e., perform their own high-level text planning [Levine *et al.*, 1991], thereby eliminating the need for the generation system to perform such planning.

3.3 Design Decisions

3.3.1 Canned Text and other Cheats

We decided fairly early on *not* to put a great deal of effort into 'properly' handling rarely-occurring special cases, but instead to support canned text and other 'cheats' as a way of handling these cases. If a particular response is difficult for our KB system to represent or our generation system to generate, we simply enter canned text for this response, or (preferably) generate as much of the response as possible with straightforward applications of our knowledge-based techniques, and then add canned text annotations to convey things it would be difficult to generate.

For example, one instructional action in the knowledge base is "mount the ITA against the test head with the four lugs of the ITA resting in the four hooked receptacles of the test head". This is currently represented as a

Mount action where the **actee** is the **ITA**, the **location** is the **Test-head**, and the **manner** is the canned text "`with the four lugs of the ITA resting in the four hooked receptacles of the test head`". The system could be augmented to 'properly' represent this manner modifier, but we felt development efforts could more productively be spent elsewhere, and hence have left this modifier as canned text. Since IDAS's domain KB is only used for generating text and does not have to support general domain reasoning, the decision on when to use canned text can be made on such engineering grounds.

We believe that 'cheating' in this and other manners (e.g., by only supporting a small number of user task and expertise models) is unavoidable, given our goal of building a usable NL generation system with non-trivial domain coverage. As in so many other fields, there is something like a 90%–10% law in operation; properly handling the 10% of special and unusual cases would require 90% of the development effort. With current-day NL generation technology, it is difficult enough to do a good job on the 90% of common cases; spending ten times this effort to handle the remaining 10% of unusual cases would not be justifiable, since we would get a much better usability payoff by spending a fraction of this effort in improving the handling of common cases.

3.3.2 Short Targeted Responses

When the project started, our industrial collaborators gave us an initial list of sample responses they wanted us to try to generate. These responses were fairly general, and subsequent discussions revealed that using more context-specific responses was preferable both for our collaborators and for us. Our collaborators preferred such responses because they were more likely to give users the information they really needed, while we found that the context-specific responses were in many ways easier to generate than the original responses; this was largely because they tended to have simpler linguistic and rhetorical structures.

For example, the original human-generated response for the query "What is the ARINC-429 Interface" was

> "The ARINC-429 interface is a serial Avionics bus interface for communicating with a UUT fitted with this bus interface. ARINC 429 is a single source, multi-sink unidirectional data transmission standard. It is used to interface digital avionics equipment in commercial applications, but is also seen in military equipment where there is a commonality with commercial equipment."

In our current prototype, if this query was asked by a user with a Skilled expertise level who was engaged in a Replace-Part task, the response would be

> "It is a Racal 10500-130 ARINC-429 interface with part number RIL-523."

The second response is intended to inform the user of the interface's manufacturer, model number, and part number, since that presumably is the information someone performing a Replace-Part task most needs to know. Hypertext follow-ups enable the user to get the location of the ARINC-429 interface and a list of its subcomponents; these also might be important pieces of information for a Replace-Part task.

The second response thus gives the user the information he most needs to know to perform his task, and uses hypertext follow-ups to enable him to obtain other possibly important pieces of information; it does *not* give him a general description of the ARINC standard, as was present in the original human-generated response. This information is not directly relevant to the Replace-Part task, and could be as much of a distraction as a help to a maintenance technician;[4] it also would be difficult to represent and generate this text in the current IDAS architecture, except as canned text (e.g., it is difficult to represent the concept of 'military equipment that has a commonality with civilian equipment' in the IDAS knowledge base).

Short, specific, and targeted responses were thus felt to be both more useful and in many ways easier to generate. There is a danger that such responses might be inappropriate, if one of the contextual factors (task, expertise, discourse) is incorrect. We will investigate this in more detail when we perform user-evaluation trials; our hope is that users will be able to use IDAS's hypertext follow-up capabilities to obtain the information they need if an inappropriate response is generated.

3.3.3 Content Determination

IDAS uses a much simpler content-determination system than other generation systems with somewhat similar goals (e.g., COMET [McKeown *et al.*, 1990] and WIP [Wahlster *et al.*, 1991]). Instead of using planning [Moore and Paris, 1989] or schemas [McKeown, 1985] to determine what to communicate, IDAS's content-determination system is based on rules (created by domain experts) of the form 'if a user asks question Q about a component of type C in the context of task T, he should be told facts F'.[5] In other words, it is intended to sup-

[4] The author of the original human-generated response agrees with this assessment.

[5] The rules are actually represented as KB classes with appropriate **basic-question**, **component**, and **task** role

port easy authorability, instead of reasoning from basic principles. We felt this was the most appropriate way in which to achieve IDAS's goal of fairly broad, but not necessarily deep, domain coverage.

One drawback of the lack of plan-based content-determination is that IDAS can not in general answer Why questions about its suggested actions, in the manner described by [Moore and Swartout, 1990]. Indeed, because IDAS does not have access to an underlying domain reasoning system (such as the EES system used by Moore and Swartout), it can only respond to a Why question if a 'purpose' plan for the relevant object or action has been explicitly added to the knowledge base by a domain expert.

4 Hypertext

4.1 Hypertext in IDAS

IDAS's hypertext interface allows users to issue new queries by mouse-clicking on pieces of text. Hypertext links are automatically added to referring expressions and action descriptions; clicking on a referring expression pops up a menu of basic questions that can be asked about the referred-to component in the current context, while clicking on an action issues a request for IDAS to explain this action in more detail (i.e., issues a How-do-I-perform question for this action).

Responses can also contain *hyperschema* follow-up buttons. These are specified by the content-determination rules (Section 3.3.3), i.e., by rules of the form 'if a user asks question Q about a component of type C in the context of task T, he should be given the opportunity to ask follow-up question F'. These follow-up questions were originally intended to implement a variant of McKeown's schema system [McKeown, 1985] where the user, instead of the system, decided which ATN arc to traverse; hence the name hyperschema. The mechanism is quite general, however, and can be used to add any useful follow-up question to a hypertext node. The current IDAS system also adds a special **MENU** button to all nodes, which allows users to explicitly modify their question-space coordinates in any way they choose; this button is primarily a development aid, and may not appear in the final system.

Users can utilize the hypertext interface for many purposes, including:

- *Elaboration:* If a user wants further information about an object or action mentioned in the text, he can obtain it by clicking on the textual description of that entity.

- *High-level text planning:* Hyperschemas and the other follow-up mechanisms allow users to dynamically specify which paragraphs they are interested in reading; this effectively means they can perform their own high-level text planning (Section 3.2).

- *Browsing:* The hypertext interface provides some support for general browsing, which may be necessary if a user is not entirely sure which question he should ask. We may add more support for browsing in the future, such as a hypertext-based structural browser similar to the one proposed for the IMAD system [Hayes and Pepper, 1989].

IDAS's hypertext interface is in some ways similar to the one presented by [Moore and Swartout, 1990], although it has a broader scope; Moore and Swartout used hypertext primarily to enable users to ask for clarifications of explanations, while IDAS uses hypertext as a general input mechanism which users can use to pose any question the system is capable of answering.

The current IDAS prototype does not do anything interesting in modeling the discourse structure of hypertext dialogues; it simply assumes that each hypertext node corresponds to a separate closed focus-space [Grosz and Sidner, 1986], and hence that an object introduced in one node cannot be referred to in another node unless it is re-introduced. We suspect this may be an overly conservative approach, and hope to do more research in the future on the relationship between the focus spaces of different hypertext nodes.

4.2 Example

Figure 2 shows some example IDAS texts produced by the various follow-up mechanisms. The initial query was What-are-its-parts, asked about the complete ATE by a Skilled expertise person performing an Operations task; this produces the text shown in Response 1. The underlined part names (which are in fact referring expressions) are all mousable, as is `ATE` in the title question and the buttons on the bottom line. Response 2 was produced by clicking on `test head` in Response 1, and selecting What-is-it from a pop-up menu of basic questions; this response was generated using the same user-task, user-expertise, and discourse-in-focus question-space components as Response 1.[6] The hyperschema follow-ups for

fillers, and attached data that indicates the facts to be communicated; content-determination is done by classifying the current question-space point into the rule taxonomy, and inheriting the attached data [Reiter and Mellish, 1992].

[6] As mentioned above, IDAS currently assumes that discourse focus-space changes within one hypertext node do not have any effect on other nodes.

Figure 2: Example IDAS Texts

⟨What-is-it, ?Component, Operations⟩ are ⟨Where-is-it, ?Component, Operations⟩ and ⟨How-do-I-perform-the-task, ?Component, Operations⟩, so `WHERE` and `USE`[7] follow-up buttons are added to the response.[8] The `MENU` button was described above; it allows the user to explicitly specify a new point in question space. Response 3 was obtained by clicking on `WHERE`; it answers 'Where is the test head'.

Response 4 comes from clicking on the `USE` button in Response 2;[9] it is a response to 'How do I use the test head'. In this response the underlined nouns `test head`, `ITA mechanism`, and `ITA` are all linked to pop-up menus of basic questions about these components, while the verbs `unlock`, `mount`, and `lock` are all linked to How-do-I-perform queries for the relevant action. Clicking on `unlock` produces Response 5, which presents a step-by-step decomposition of the action of unlocking the ITA mechanism. Response 6 was obtained by clicking on `lever` in Response 5, and selecting What-is-it from the pop-up menu.

4.3 Hypertext vs NL Understanding

From IDAS's perspective, hypertext is a technique for enabling users to specify an input (i.e., a question-space point) to the NL generation system. As such, it is natural to compare it to other input mechanisms, particularly natural language text input. The advantages of hypertext over NL input systems include:

- *Implementation:* A hypertext interface is easier to implement than a NL input system; indeed, we have found that generating hypertext is only marginally more difficult than generating conventional text (if appropriate graphics support software is available). Implementing an NL input system, in contrast, is a major undertaking.

- *Deictic References:* As [Moore and Swartout, 1990] point out, a hypertext interface makes many (although not all) kinds of references trivial for a user; he simply points to the phrase that describes the object or action he wants more information about. The user does not have to construct a complex referring expression (e.g., "the board I was told to remove in the second sentence"), and the system does not have to try to resolve such complex references.

- *Transparency of Capabilities:* NL understanding systems are in general only capable of answering a subset of the questions the user is able to pose, and the user may become confused if he is not aware of the boundaries of this subset [Tennant *et al.*, 1983]. This problem does not arise with hypertext, where the user is only allowed to issue questions that the system can answer.

[7] How-do-I-use-it is the interpretation of How-do-I-perform-the-task under an Operations user-task

[8] Other questions, e.g., What-are-its-parts, can be asked by clicking on `test head` in the title question, and selecting from the pop-up menu.

[9] An identical response would have been obtained by clicking on `USE` in Response 3, since Responses 2 and 3 have the same task, expertise, and discourse question-space components.

Perhaps the primary disadvantage of hypertext systems is their lack of flexibility; hypertext systems typically limit the user to pointing to a single entity and asking one of a small number of questions about it, while NL input systems allow queries to be stated using the full power of English or other human languages. While this is a severe, perhaps crippling, drawback in many applications, we believe it is less of a problem in IDAS, because IDAS is only capable of responding to a small number of basic questions about entities in any case. Hypertext clicking can in fact be used in IDAS to pose almost all questions that IDAS is capable of answering; if the user is allowed to use the **MENU** button, then any answerable query can be issued through the hypertext interface. Accordingly, IDAS does not currently include an NL understanding component, and there are no plans for adding one; we believe that hypertext mechanisms will provide a sufficient query input mechanism for IDAS.

5 Conclusion

The IDAS project is attempting to use natural-language generation and hypertext technology to build a prototype of an on-line documentation and help system for complex machinery. IDAS is based around the ideas of (1) having a well-structured question space; (2) using KR and NL generation systems that produce short targeted responses and allow canned text to be used when necessary; and (3) presenting users with a hypertext-like interface that allows them to pose follow-up and elaboration questions. Our hope is that this combination will allow us to construct a system that demonstrates that current-day natural-language generation technology can be used to build a useful on-line documentation facility; this, indeed, is the ultimate goal of the IDAS project.

References

[Brachman and Schmolze, 1985] Ronald Brachman and James Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9:171–216, 1985.

[Grosz and Sidner, 1986] Barbara Grosz and Candace Sidner. Attention, intention, and the structure of discourse. *Computational Linguistics*, 12:175–206, 1986.

[Hayes and Pepper, 1989] Phil Hayes and Jeff Pepper. Towards an integrated maintenance advisor. In *Hypertext 1989 Proceedings*, pages 119–127, Pittsburgh, 1989.

[Kasper, 1989] Robert Kasper. A flexible interface for linking applications to Penman's sentence generator. In *Proceedings of the 1989 DARPA Speech and Natural Language Workshop*, pages 153–158, Philadelphia, 1989.

[Levine *et al.*, 1991] John Levine, Alison Cawsey, Chris Mellish, Lawrence Poynter, Ehud Reiter, Paul Tyson, and John Walker. IDAS: Combining hypertext and natural language generation. In *Proceedings of the Third European Workshop on Natural Language Generation*, pages 55–62, Innsbruck, Austria, 1991.

[McKeown *et al.*, 1990] Kathleen McKeown, Michael Elhadad, Yumiko Fukumoto, Jong Lim, Christine Lombardi, Jacques Robin, and Frank Smadja. Natural language generation in COMET. In Robert Dale, Chris Mellish, and Michael Zock, editors, *Current Research in Natural Language Generation*, pages 103–139. Academic Press, London, 1990.

[McKeown, 1985] Kathleen McKeown. Discourse strategies for generating natural-language text. *Artificial Intelligence*, 27:1–42, 1985.

[Moore and Paris, 1989] Johanna Moore and Cecile Paris. Planning text for advisory dialogues. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, pages 203–211, 1989.

[Moore and Swartout, 1990] Johanna Moore and William Swartout. Pointing: A way toward explanation dialogue. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 457–464, 1990.

[Penman Natural Language Group, 1989] Penman Natural Language Group. The Penman user guide. Technical report, Information Sciences Institute, Marina del Rey, CA 90292, 1989.

[Reiter and Dale, 1992] Ehud Reiter and Robert Dale. A fast algorithm for the generation of referring expressions, 1992. Submitted to COLING-1992.

[Reiter and Mellish, 1992] Ehud Reiter and Chris Mellish. Using classification to generate text, 1992. Submitted to ACL-1992.

[Reiter, 1990] Ehud Reiter. Generating descriptions that exploit a user's domain knowledge. In Robert Dale, Chris Mellish, and Michael Zock, editors, *Current Research in Natural Language Generation*, pages 257–285. Academic Press, London, 1990.

[Tennant *et al.*, 1983] Harry Tennant, Kenneth Ross, Richard Saenz, Craig Thompson, and James Miller.

Menu-based natural language understanding. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, pages 151–158, 1983.

[Touretzky, 1986] David Touretzky. *The Mathematics of Inheritance Systems.* Morgan Kaufmann, Los Altos, California, 1986.

[Wahlster *et al.*, 1991] Wolfgang Wahlster, Elisabeth André, Som Bandyopadhyay, Winfried Graf, and Thomas Rist. WIP: The coordinated generation of multimodal presentations from a common representation. In Oliverio Stock, John Slack, and Andrew Ortony, editors, *Computational Theories of Communication and their Applications.* Springer-Verlag, 1991.