This work was supported in part by:

The National Science Foundation
Grant Number IST 8104984

ACKNOWLEDGEMENTS

This dissertation owes its existence to the direction, encouragement, collaboration, and intellectual support I received from two people: Michael Arbib and David McDonald. Besides his crucial guidance in the beginning stages of this work, Dr. Arbib provided an orientation to Cognitive Science which is reflected throughout this work. Dr. McDonald's partnership in the design, implementation, and development of the program was essential, and it is to him that I owe everything that I know about language generation.

Likewise, Lyn Frazier was a wonderful linguistics mentor, and I am enormously grateful to her for pushing me to be scientific (i.e. "explicit") about which claims in this thesis I really believed in, and which of those were testable. Kate Ehrlich gracefully guided me through the maze of designing, running, and reporting on psychological experiments. I also thank David Waltz, Mitch Marcus, and Candy Sidner for their interest and input over the years.

The students, faculty, and staff of the Computer and Information Science Department have been a source of numerous dicussions and friendships (and volleyball games). In particular, Jeff Bonar and Steve Levitan were the source of many long and productive arguments during our "formative years" together. Terry Weymouth went out of his way to keep me honest about my notions of computer vision. And Bev Woolf kept me company on the computer for some of the most exciting nights of my intellectual career.

On the more personal side, I would like to express my appreciation to my parents for their loving support (financial and emotional) during what seemed like (and was) such a long time.

For the discipline I had to develop to finish this work and the sheer satisfaction of doing good work I thank Werner Erhard, who was instrumental in my discovery of myself.

Finally, I thank my wife Marie for her support, her assistance whenever I needed it, and for putting up so graciously with the other woman in my life (this thesis!). I couldn't have done it without you!

ABSTRACT

DATA-DRIVEN INDELIBLE PLANNING

OF DISCOURSE GENERATION USING SALIENCE

May 1983
E. Jeffrey Conklin
B.A., Antioch College, Yellow Springs, Ohio
M.S., Ph.D., University of Massachusetts
Directed by: Professor David D. McDonald

Natural language generation can be divided into two stages: deep generation, in which the content and style of the utterance are selected and a specification of the desired utterance is constructed, and surface generation (or "realization"), in which this specification is converted into natural language using the syntactic, lexical, and morphological rules of the language. Traditionally, the deep generation task of selecting the topics of discourse (when not trivial) has been done using a computationally expensive, goal-directed approach. In this thesis the salience of objects in a database is used to provide a way to explore data-driven selection – a computationally much less expensive approach to discourse planning.

The domain of generating descriptions of natural suburban scenes was used in this research. The phenomenon of salience in such pictures was explored through a series of psychological experiments – in some, subjects provided subjective ratings of the relative importance of the items in various photographs; in others, subjects wrote short textual descriptions of the same pictures. Analysis of the rating data provided the basis for the beginnings of a theory of visual salience as a perceptual phenomenon, while analysis of the combination of rating and textual data showed the strong influence of salience on the order of mention of objects in the descriptions.

Based on the data from these experiments a program (GENARO) was written which plans paragraph-length descriptions of visual scenes. The input perceptual representation, though hand-built, is designed to simulate the output of a fully operational computer vision system, and includes an annotation of objects' representations with the empirically-based salience values. GENARO uses production rules (each of which knows about some specific rhetorical or stylistic effect) to build "rhetorical specifications". Its processing is data-driven, i.e. a "current-item" is selected from a salience-ordered list of perceptual objects, and this object and its salience are the primary determiners for the actions of the rhetorical rules. In addition, all of GENARO's actions are indelible – the control structure provides no look-ahead or backup for the planning process.

The realization specification ("r-spec") built by GENARO is realized as an English sentence by a separate AI program, MUMBLE, written by McDonald as a general purpose realization component.

The "planning" done by GENARO is distinctly localized and short-sighted, and aims to take full advantage of the topic-ordering information captured in the data base by the salience annotation. The fact that this localized planning is able to devise quite natural-sounding paragraphs demonstrates that a data-driven approach to deep generation is viable, and that salience can be a powerful heuristic in guiding natural language generation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

In investigating the process of Natural Language Generation (NLG) there are two major aspects to be addressed:

1. How to determine *what* to say, and what *not* to say; and

2. Determining *how* to express that which has been selected.

Part of the difficulty in a broad approach to NLG is that, while the second issue is fairly well defined and is a straightforward extension of linguistic inquiry, the first question (which I will call the problem of "*selection*") opens the door to the little-understood realms of pragmatics, speech acts, and discourse theory. Grice observed ([Grice 1975], p. 67) that there is a conversational maxim to "Be relevant", but confessed that the issue of conversational relevance was complex and difficult.

Until recently the issue of how one decides what to say could not be discussed in any detail because it was underspecified. On the simplest model the process of deciding what to say requires a representation of "what is known" (in the sense of "available for saying") as the input, and what specifically is to be said and how to say it (somewhat akin to a "deep structure") as the output. Without specifications of the input "thoughts" or "knowledge" and this output "message", however, linguists, psychologists, and philosophers have not had the tools to have much of a conversation about the details of the selection process.

The field of Artificial Intelligence (A.I.) has at last provided the opportunity to construct the tools with which language generation may be investigated. However, NLG systems to date have generally avoided facing the selection problem squarely, either by skipping it entirely (e.g. [Friedman 1969]), or by "pre-wiring" the solution into the input data base (using "and-then-say" links between items) (e.g. [Davey 1979] and [Mann and Moore 1981]). The recent systems which do offer serious solutions to the Selection problem use powerful but costly search [Appelt 1982] and matching [McKeown 1982] techniques to construct the message.

This thesis is based on an observation that is so simple that it seems to have been overlooked: that speakers talk about what is *important* to them in a particular situation. More specifically, that the Selection process is guided (if not determined) by the evaluation of what in the data base is important, or "salient", and what is not. Hence, one of the central issues of this thesis is, What makes something

salient?[1] Not only does this notion simplify the problem of What to say, it captures an important aspect of human knowledge – that we have an intuitive sense of the relative importances of the different things that we know about. The thesis goes on to explore the strengths and limitations of a simple planning system which exploits salience as a heuristic for text planning.

1.1 Picking a domain

Since the generation process starts with a "meaning"[2] and finds linguistic expression for it, the selection of the domain of discourse is crucial. The issues of what is said and how it is said are so intertwined that the "How" cannot be usefully studied in isolation – the "What" must also be considered.[3] Thus, part of building an A.I. natural language generator for the purpose of studying how ideas are turned into language is the selection of a domain which can provide the "seed" semantic content.[4]

For my research on generation I chose the domain of descriptions of pictures, using as the domain data base a symbolic representation of the visual information in a picture. Here there is a natural and direct correspondence between the perceptual material presented in the picture image and the linguistic material in the textual description. This correspondence can be used as a motivation of and check on the rules and strategies that go into deciding what in the picture is important to mention – and at what point in the text – and what can be omitted. Indeed, there are levels of importance of items in a picture and there are degrees of rhetorical stress on items in a text, and this correlation between "meaning" and language is perhaps more accessible in studying scene descriptions than in most other kinds of textual material.

---

[1] It is tempting to define salience in terms of what people talk about, but this makes it vacuous by making it circular. To avoid this, salience must be approached as a *pre-linguistic* phenomenon.

[2] By "meaning" I mean simply a packet of information which is to be expressed by the NLG system.

[3] In studying parsing, on the other hand, the input data is clearly specified (as a string in the language), so one can more readily separate the conventional study of How things are said (e.g. syntax) from the study of semantic content.

[4] Ironically, there is considerable interaction between the domain chosen and what one learns about generation: the domain of discourse has a great deal to do with the style and structure of the text generated about it.

1.2 Deep and Surface Generation

Another aspect of NLG is that it is a process which naturally divides into two phases. In the first phase selection takes place, reflecting the speaker's goals, and the selected material is composed into a "realization specification" (abbreviated "r-spec")[5] according to high-level rhetorical and stylistic conventions. In the second phase the r-spec is "realized" -- the English text produced -- in accordance with the syntactic and morphological rules of the language.[6] I call the first phase "deep generation", instead of the more technique-oriented term "planning", to reflect the view that its use of actual planning techniques will be one of several methods available for high-level structuring of the output.

This thesis presents a model of deep generation which performs data-driven planning by taking advantage of the power of salience.

Rather than designing and building my own system for doing the second stage "surface generation" (also known as "realization"), I chose to adopt McDonald's MUMBLE system [McDonald, 1981]. MUMBLE was designed with a very flexible input specification, thus making it useful as a general purpose realization component. In fact, the process of using MUMBLE in the output part of a natural language interface mainly involves building for it a "dictionary" which specifies the possible English realizations for each term in the domain data base (see Section 4.2).[7]

1.3 The thrust of this research

The purpose of this research has been to explore an approach to the planning of text which is in some settings more natural than traditional methods. The exploration has taken two forms: 1) experimental studies of people looking at pictures

---

[5] I use this new term -- "realization specification" -- in place of the term "message", simply to avoid some of the "baggage" which the less technical term carries.

[6] This distinction has also been made by Levelt [1979] and Kempen [1977], using the terms "conceptualizing" and "formulating"; by McDonald [1980], using the terms "speaker" and "realization component"; and by Thompson [1977] and McKeown [1982], using a "strategic" component followed by a "tactical" component.

[7] This was actually a liability in doing this research (as further discussed in Chapter 4), because it left the output of the deep generation component almost completely unspecified. It would have been easier (although less flexible) if MUMBLE had had a rigid input language, since that would have provided more constraints on the design of the deep generation component.

and describing them, and 2) the construction of an A.I. program, GENARO, which is the deep generation component of a system which generates natural-sounding descriptions of scenes. The program incorporates insights derived from the studies into both its structure (by relying heavily on *salience* as a heuristic) and its knowledge (by using rhetorical conventions culled from human-generated texts). The program simulates human generation (speaking) performance: because it uses an efficient data-driven approach to planning the text, and because its decisions are indelible, it is effectively real-time in its planning, and it occasionally "talks itself into a corner" (i.e. builds unrealizeable rhetorical specifications).

GENARO also provides a testbed in which rhetorical and thematic conventions can be explored. The program uses a set of "rhetorical rules", expressed as production rules, to do its planning. Since changes in and additions to this body of rules show up in the structure of the text produced by the system, it is possible to discover specific rhetorical mechanisms, as well as to discover and test rhetorical conventions in a very precise framework.

1.3.1 A short example.

"A picture is worth a thousand words."
An old maxim

This section presents a trace of the generation of a short scene description, showing briefly the steps that GENARO takes to plan a paragraph of text. Figure 2 shows the specific paragraph which describes the picture shown in Figure 1.

The paragraph was generated from a (hand-simulated) perceptual representation in which the most salient objects, in order of decreasing salience, were:

House, Fence, Door, Driveway, Gate, Mailbox, and Lighting.

The deep generation component maintains this list as the "Unmentioned Salient Objects List" (*USOL*), and it is this data structure which mediates between GENARO and the domain data base (see Figure 3). It should be stressed that the USOL contains only objects -- not properties of objects or relationships between objects -- since I specifically claim that such an "object-driven" approach is both more natural and adequate to the task.[8]

[8] In general in this thesis I use the term "object" to refer to objects in the world as they are represented in the domain data base, whereas "item" is any entity in the data base (e.g. objects, properties, relations, etc.).

Figure 1: One of the pictures used in the experiments.

"This is a picture of a white house with a fence in front of it. The house has a red door and the fence has a red gate. It is a cloudy day."

Figure 2: A short description of the picture in Figure 1.

There is one primary register in the system: "Current-item". It contains the object currently in focus (and the most salient object which has not previously been mentioned). An object moves into focus by being "popped" from the USOL and placed in the Current-item register, along with its most salient properties and relationships (for ease of access). When formulating the r-spec, most of the rhetorical rules then look only at the Current-item. (Some rules look down "into" the USOL, or into the r-spec under construction, as elaborated below.)

GENARO stores its rhetorical conventions in the form of production rules.[9] The rules are organized onto several "packets" (e.g. Introduce, Shift-topic, Elaborate, Conclude), which are used for high-level control of the paragraph structure.

[9] A production rule is specialized parcel of procedurally-encoded information: there is a "precondition part" which, if its various conditions succeed, triggers the "action part", in which the actions of the rule are specified.

The control mechanism for the production rules is "Iterative Proposing": each of the rules whose precondition is satisfied makes a proposal and gives it a priority; the proposals are then ranked, and the one with the highest priority wins. "Winning" generally means that the element that was constructed by that rule gets added to the r-spec. This process is repeated until the r-spec is complete. (Repetition of this proposing stage is necessary because the environment in which the rules' conditions are evaluated changes as the r-spec grows.) The r-spec can thus be thought of as a "molecule", each of whose "atoms" is the result of a successful rule. These atoms are called "r-spec elements", and are the basic units which are processed by MUMBLE; they are either objects, properties, or relations from the domain, or rhetorical instructions that originate with GENARO.

In the course of producing a description many r-specs will pass from GENARO to MUMBLE. Each r-spec is produced "locally" within GENARO, without an awareness of previous r-specs or a planning of future ones, and MUMBLE produces a sentence for each r-spec it receives.

GENARO starts with an empty r-spec buffer and with Current-item set to House, the first item (in this example) in the USOL (see Figure 3). There is a rule which proposes to "Introduce(House)"; this rule's conditions are that this is the first r-spec in the description, and that the salience of the Current-item is above some specified threshold. In this example both of these conditions are met, and the r-spec element Introduce(House) is proposed at a high rhetorical priority, thus guaranteeing not only that it will be included in the first r-spec, but that it will be the dominant element in that r-spec. In the next round of proposing another rule proposes (and this proposal wins) including the color of the house (e.g. Color(House,White)), not because the color is itself salient, but to "flesh out" the introductory sentence. This rule is included because it was noticed that salient items were rarely mentioned as "bare" objects -- *some* property was always given. (Note also that there are other rules that propose mentioning properties of objects on other grounds, i.e. because the property itself is salient.) Finally, there is a rule which notices that Fence is both quite salient and directly related to the Current-item, and so proposes In-Front-Of(Fence, House).

The *USOL*: Fence Door Driveway Gate Mailbox Lighting

The *Current-item*: House

Figure 3: The initial data structures.

The values of the two main data structures after *House* has been popped off of the USOL and made the Current-item.

Since the r-spec now contains three elements and there are no strong grounds based on salience or considerations of style to continue adding to it, the r-spec is sent to the process MUMBLE, which immediately realizes it. MUMBLE's dictionary contains entries for all of the symbols used in the r-spec, e.g. Introduce, In-front-of, House, etc., and these are used to construct a linguistic phrase structure tree which then controls the realization process, outputing "This is a picture of a white house with a fence in front of it.". Back in GENARO, after the r-spec was sent, the Introduce packet was turned off, the message buffer cleared, Door (the next *unused* object) removed from the USOL and placed in the Current-item register, and the Iterative Proposing process started over.

In building the next r-spec, Part-of(Door, House) and Color(Door, Red) are inserted, by rules similar to the ones described above. However, there are no other salient relations or properties to mention about the Current-item Door: nothing of high rhetorical priority is left to be proposed.[10] There is, however, a rule called "$condense-prop" which looks for "rhetorical parallels" and proposes them at *low* priority (i.e. they only win when there are no more useful rhetorical effects which apply). $Condense-prop notices that both Door (the Current-item) and Gate (which is somewhere "down" in the USOL) have the property Red, and that the salience of Gate and of the property Color(Gate, Red) are above the appropriate thresholds, and so proposes that Gate be made the new focus. When this action is taken, a conjunction marker (a symbol know by MUMBLE to signal conjunction) is added to the r-spec, and Gate is pulled out of the USOL and made the Current-item. Processing continues until both the color of Gate and the fact that it is a part of the Fence are inserted. The r-spec created by these actions is realized (by MUMBLE) as "The house has a red door and the fence has a red gate.".

As the next USOL object (Driveway) is made the Current-item it is found to have a below-threshold salience value. This triggers the Conclude packet being turned on. In this packet is a rule which goes into the USOL looking for scene-level "objects", such as the lighting conditions in the picture (e.g. day or night), the season of the year, etc. This rule finds and proposes the object "Lighting", and this is sent to MUMBLE as a single element r-spec, resulting in the sentence "It is a cloudy day."

1.3.2 The claims of this thesis.

As in all A.I. research there are a myriad of details and decisions in the program itself. Some of these are deliberate and carefully chosen, while most are unimportant implementation details. The distinction is important: in order to evaluate

---

[10] Once a rule's proposal is accepted that rule turns itself off until that r-spec is complete.

the theory[11] offered here one must know which details of the program are unimportant and which are substantive claims of the model.

Therefore, Figure 4 offers a list of the central claims of this thesis. The fundamental claim being made by this thesis, of which each of these claims is a particularization, is that

> *deep generation can be done quickly and effectively using a data-driven, indelible planning phase, IF the domain data base is annotated with salience.*

The basis of this rapid planning is that the model is *data-driven* and has very little knowledge about global paragraph structure. Instead, it uses the salience in the data base to steer a very localized planner (i.e. one that does not look forward or back in deciding what to do next).

The 10 claims in Figure 4 specify precisely the important aspects of deep generation which this system is designed to address. They comprise an initial *theory* about NLG – proving a claim untrue constitutes an invalidation of the theory in the particular form embodied in the program. Most of these claims are technical, and require some discussion to be clear; this discussion is provided in Chapter 5, so these claims are presented here simply to give the reader an orientation to the important points to be discussed in this thesis.

1.4 The organization of this thesis

This chapter has presented an overview of the research reported in this thesis. Chapter 2 presents the previous work that has been done in NLG, with particular attention to the ways in which selection and deep generation have been handled.

An empirical exploration of the notion of visual salience, which is central to the work presented in this thesis, is presented in a different Technical Report (COINS TR 83-14). From the perceptual psychology experiments on scene descriptions described there, it was determined

---

[11] In this thesis I will use the following terminology: a *theory* is a formal claim or a set of claims about a system; a theory can be embodied in a *model*, which makes specific predictions about the input/output behavior of the system; finally, a *program* is a machine-runnable implementation of a model. This distinction is from McDonald [personal communication].

1. Some annotation of salience in the domain data base is necessary to organize descriptions based on that data base.

2. Natural-sounding descriptions can be generated using as the primary selection strategy: Mention the most salient things first.

3. I define the *Locality Constraint* to be the following limitation on the power of a deep generator: each domain item is made the Current-item once, and the system rules describe only the Current-item. The result is that an item is described at only one point in the text; the claim is that this is sufficient to cover a broad range of descriptive texts.

4. Descriptive paragraphs are not generated by a recursive (i.e. stack) mechanism.

5. No more or less than the power of Iterative Proposing is required to effectively use rhetorical conventions expressed as production rules.

6. The planning of r-specs can be effectively managed indelibly -- backtracking is not needed, because the domain of rhetorical planning is resilient enough that it is in fact difficult (at this level) to "paint yourself into a corner".

7. Salience is perceptual, not linguistic. The components of visual salience are computed as a *by-product* of seeing and "understanding" a picture, so that selection based on salience relies not on some prespecified order of presentation in the data base but rather reflects naturally the *processing* used to construct and maintain the data base.

8. Perceptual descriptions are oriented to the *objects* in the domain data base, while properties and relationships are secondary.

9. Feedback from surface to deep generation is expensive and unnecessary.

10. It is adequate to the interface between deep and surface generation to have the basic unit of planning at the deep level correspond to a single sentence at the surface level.

Figure 4: The claims of this thesis.

- that the notion of visual salience, as defined there, is a significant perceptual phenomenon, and that it possesses an internal structure consisting of high- and low-level and intrinsic and context-dependent components;

- that it can be studied by the simple experimental techniques described in the chapter;

- that salience organizes a perceptual data base in a way that allows greatly simplified rhetorical planning in natural language generation, by playing a powerful role in determining the order in which objects are mentioned in scene descriptions; and

- that there are other textual organizing forces than salience, and that these have to do with the relationship between objects in the domain.

Finally, it is suggested that the notion of salience, where it is applicable, can be a powerful additional dimension in a data base.

In Chapter 3 the deep generation component GENARO is described, along with a detailed example of its operation. In the example the extreme locality and "myopia" of the program are demonstrated, and specific examples of the strengths and failings of this approach are provided.

Chapter 4 then presents the computational setting in which GENARO operates. At the input to GENARO is a visual representation, which is (as mentioned above) a mockup of the internal model which a full-scale computer vision system would construct during its analysis of an image. The representation is annotated with salience values which were derived from the experiments. At the output from GENARO is the MUMBLE surface generator. This system is described as an interpreter of GENARO's r-specs into English, with emphasis on the form of dictionary entries and the process of writing them.

Chapter 5 is about the theoretical implications of this work. It presents a thorough discussion of the above claims, as well as a discussion of the use of GENARO as a tool for doing research into the rhetorical conventions in English.

Finally, in Chapter 6 the strengths and shortcomings of this research are discussed, with an eye toward specific research topics for the future.

CHAPTER II

BACKGROUND

2.1 Deep Generation

The problems confronted by the builder of a generation system are intimately related to the domain which is chosen as the source of "ideas" to be expressed. The choice of domain brings with it its own range of special problems, exceptions, and heuristics. Work in the area of deep generation, including this thesis, is particularly sensitive to this interdependence, since the interface between perception, conception, and language is too rich to have allowed anything but spotty and partial treatments to date. Before the problem of deep generation can be considered to be "solved" we will need to have given the gift of language output to a computer system with sufficient world knowledge and computational power to be capable of *needing* to say everything that people might need to say.

In this chapter I review some of the previous efforts at deep generation. It is noteworthy that this is a relatively short discussion. In the brief history of computational linguistics, only a handful of workers have addressed the hard problems that occur upstream from grammatical and lexical realization. Indeed, the fact that the name "deep generation", and the two-staged view of generation that it implies, is relatively new reflects uncharted nature of the territory. Most work has either used an input domain which presented few problems, or has somehow skirted the problems presented by the input domain.

The following discussion is organized into three broad categories: systems which skipped most of the deep generation process by hand-feeding the generator with a very small packet of carefully structured information; systems which used a large input data base but which built the solution to the selection problem into that data base; and systems which did serious deep generation.

2.1.1 Systems with very restricted input.

The clearest examples of this kind of system are the various efforts in machine translation, i.e. Herskovits [1973] and Brown [1974]. These systems accept an utterance in one language, parse it into some internal representation, and feed this representation to a generator which uses the grammar of a different language. There is no need to reflect on or plan the contents or style of the output, since these are completely specified by the input.

Generators such as Friedman's [1969] which were primarily randomly driven surface generators had essentially no input from which to plan. They were useful for exploring grammatical issues in a generation framework. The systems that were driven by ATN's, i.e. Simmons and Slocum [1972] and Goldman [1974], also tended to focus on these surface generation issues.

2.1.2 Data bases with selection pre-wired.

Other workers chose to provide their system with an interestingly large domain data base from which to generate. Usually, however, the solution to at least the selection problem was pre-wired into the data base. Sometimes other deep generation issues, i.e. level of detail and where to place emphasis, were pre-wired as well.

Meehan's generation system [1977] "composed" simple stories by using world knowledge about the characters, the environment, and some rules of behavior and motivation to construct an internal plan for the story. The order in which events were described was largely prespecified, however. The level of detail was determined by the limited detail in the system's model of the world.

Davey's system [1979] for describing a game of tic-tac-toe had both deep and surface generation components. The input data base -- a move-by-move description of a game -- had the kind of temporal links between the objects of description (i.e. the moves of the game) that make the Selection process simple, since it is not only adequate but desirable to simply mention the moves in the order in which they occured. However, Davey's system also determined how to *group* the moves for each sentence -- a crucial issue, since too few or too many moves made for awkward text. Each move's strategic value had to be taken into account, so that a threat/response pair, for example, did not get separated by a sentence boundary. Most importantly, Davey's system tackled the problem of how to express each move in the context of its relationship to previous moves.

Mann and Moore [1981] used as their input data base a flow-chart for responding to a fire alarm. The order of mention was, as with Davey, specified essentially in terms of explicit "and-then" links in the domain data base. Decisions about sentence size and style were made on the basis of general purpose heuristics (and not on discourse context or sentence content).

2.1.3 Planning and deep generation.

A few systems have taken on the substantial problems in deep generation. They use the generally thorough and expensive process of planning to do the selection job. Cohen [1978] studied planning as a solution to the Selection problem in deep generation. His system, which planned a speech act in response to a user's question, did its planning by using backwards chaining to search through a space of possible

utterances and the goals they could achieve. More recently, in [Cohen et. al. 1981], he has described a model (based on recognition of "intended plans") which is powerful enough to enable a computer program to act as a truly helpful partner in a dialogue.

Appelt's KAMP [1982] system features the planning of utterances which satisfy several goals. It uses planning in all phases of generation, from the selection of the high-level illocutionary act (e.g. inform or request) to the syntactic structure and lexical items of the text. Appelt claims that by treating the entire generation process in this uniform way sub-goals at different levels of the process can interact, allowing low-level decisions to influence high-level selection decisions. His approach also explicitly rejects the view of generation as a multi-stage linear process -- the planning of what to say and how to say it are concurrent processes in a uniform framework. Appelt's system also features reasoning about what to say based on what the hearer knows and wants, using a formalism based on possible worlds semantics of an intensional logic of knowledge and action.

Finally, McKeown's TEXT system [1982] generates explanatory paragraphs in response to a data base query. This work represents a substantial advance in the use of sophisticated selection strategies in the generation of multi-sentence texts, as well as being an interesting approach to maintaining coherence relations between successive sentences. Because of the similarities with the goals of my system, her system will be reviewed in somewhat more detail than the others.

The inputs to the system are a data base and a user-generated query concerning the data base. The data base is an Office of Naval Research data base on ships, missiles, etc. expressed in a semantic-net formalism. There are three basic types of questions the system can answer: "Describe X", "Define X", and "Compare X and Y". The specific question asked is used initially for two things: to select a small set of schemas, each of which might be used to answer the question (which schema to use is determined later); and to select a small subset of the data base, called the Relevant Knowledge Pool, from which the data for the answer are selected. The type of data in the Pool is then used to determine precisely which schema is to be used in constructing the answer.

Since schemas play such a central role in this system it is useful to describe them in some detail. McKeown constructed her schemas by studying actual samples of text and generalizing their structure. An example is shown in Figure 5. McKeown writes "A schema is a representation of a standard pattern of a discourse structure which efficiently encodes the set of communicative techniques that a speaker can use for a particular discourse purpose" [McKeown 1982]. Despite the fact that the schema notation is highly suggestive of a context free grammar, McKeown claims that her schemas are strictly *descriptive* of some observed textual phenomena, and are

Identification Schema

Identification (class & attribute/function)
{Analogy/Constituency/Attributive/Renaming}*
Particular-illustration/Evidence+
{Amplification/Analogy/Attributive}
{Particular illustration/Evidence}

Example

"*Eltville* (Germany) 1) An important wine village of the Rheingau region. 2) The vineyards make wines that are emphatically of the Rheingau style, 3) with a considerable weight for a white wine. 4) Taubenberg, Sonnenberg and Langenstuck are among vineyards of note."

Classification of the Example's Sentences

1. Identification (class & attribute)
2. Attributive
3. Amplification
4. Particular illustration

In the schema "{}" indicates optionality, "/" indicates alternatives, "+" allows its item to appear 1-n times, and "*" allows its item to appear 0-n times.

Figure 5: The Identification Schema in McKeown System.

---

not "prescriptive".[1]

---

[1] From a cognitive science perspective there is some question about the precise role played by schemas in the TEXT system. On the one hand they are derived in an empirical manner from actual samples of text, and in the computer implementation they stand squarely in the middle of the processing – functioning both as data (or more accurately the "filter" through which the data comes) and as control (guiding the actual structure of the paragraph being generated). On the other hand, McKeown is careful to state that they do *not* function as grammars of English text. While it may be a minor point, this makes it questionable that schemas are testable (i.e., falsifiable), and, more generally speaking, leaves some doubt about what their theoretical status is. This problem is an example of the tension between traditional A.I. and cognitive science. To enter the cognitive science arena requires making claims beyond the gross input/output behavior of the system to the actual structures and processes used in the model.

In TEXT schemas are implemented as Finite State Automata (FSA), which is parallel to the Iterative Proposing control structure in GENARO: in TEXT all arcs leaving a state are "taken", but the best one is selected by the focus constraints, and the state that it leads to becomes the next state. (Thus, although all possible next states are explored, only one is actually taken.) This is similar to what happens in a round of proposing of the rhetorical rules in GENARO: all of the rules which are enabled make a proposal, the best of which is chosen to be actually done. The major difference is that GENARO's "state space" is not explicitly specified -- it is implicit in the possible orderings among the rule firings.

Once a single schema is selected to guide the construction of the text which answers the user's question it is "layed down" on the data in the Pool and a matching process seeks to fill the slots in the schema with facts ("propositions") in the data base. Here is where the focus mechanism is used -- it helps to resolve conflicts when there are several propositions competing for a slot in the schema. The filled schema constitutes the r-spec -- it is then sent to the surface generation ("tactical") component to be realized.

Selection is performed in TEXT during the process of schema filling. Three processes collude to fill a schema: 1) if there are several competing slots then all of them are filled; 2) if a given slot can be filled by several data base propositions then all possible data points are taken to form a set from which one element will be selected; and finally, 3) a function which embodies a set of FOCUS constraints is used to select, from among these many propositions and slots, which ones provide the most coherence. The first two of these processes actually function to increase the number of alternatives of what to say next -- if the nodes in the data base are "bushy" there could be a large number of alternatives at this point. Thus the real work of selection is left to the focus mechanism.[2]

2.1.4 Summary.

Mann et al [1981] suggested that there are currently four aspects of text generation which require concentrated study in order to develop flexible and portable text generation facilities:

1. A comprehensive, linguistically justified grammar,

---

[2] As might be expected, some of the choices made on this basis are arbitrary, and must be made by "tossing a coin". It is precisely this problem that the notion of salience addresses: many of these choices would not, in fact, be arbitrary if the data base were augmented with a salience annotation on its objects, and the deep generator had access to this salience information.

2. A knowledge-representation formalism that can encode diverse kinds of information,

3. A model of the intended reader of the text, and

4. A model of discourse structure and control.

The work described in this thesis most potently addresses the fourth component, although it connects strongly with the second component as well. In GENARO the key is the use of salience as a heuristic which greatly reduces the cost of the (generally expensive) process of planning what to say, what to leave out, and in what detail to discuss what is to be said.

2.2 Salience

Although there has been considerable study of the processes of perceiving, recalling and recognizing visual information (cf [Loftus 1982]), I have found little work directly addressing the issue of salience in pictures. Hooper [1980] studied the visual factors that contributed to the *recognition* of objects in scenes, but little of this study can be applied directly to the problem of determining what makes a particular object *salient*. Some of the work that has been done on image abstraction for image storage and retrieval is relevant to visual salience. Brush [1979] used a numerical rating technique in the study of pictures (as did I), but his subjects rated the entire picture and major picture elements, and not the specific objects in the scenes. Firschein and Fischler [1971] discussed methods of "content analysis" in library pictures, including tackling the "problem of aboutness". They note that "what a document is about depends on what its reader will use it for". Although the influence of purpose and context on salience and descriptions is considerable, it is also quite complex. I found that I was able to standardize this aspect of the perception process in my experiments by instructing subjects to adopt a particular perspective.

CHAPTER III

THE AI PROGRAM GENARO

In this chapter the LISP program GENARO[1] will be discussed in terms of its operation as an Artificial Intelligence program (as opposed to its embodiment of a theory of generation, which is discussed in the next chapter). This will include both a description of the various assumptions underlying the system design and a detailed discussion of how those assumptions were implemented in the program. This chapter is divided into four sections: 3.1 presents a brief explanation of the operation of MUMBLE, 3.2 is a technical discussion of the control structure of the program, 3.3 elaborates on the design of the rhetorical rules and their interactions, and 3.4 presents a detailed trace of the program planning a paragraph, as an example of how all this machinery works together.

3.1 A brief explanation of MUMBLE

In this thesis I have adopted the traditional view (in A.I.) that natural language generation is divided into two successive phases: a first phase in which *selection* takes place, reflecting the speaker's goals, and the selected material is composed into an r-spec ("realization specification") according to high-level rhetorical and stylistic conventions, and a second phase in which the r-spec is realized -- the text actually produced -- in accordance with the syntactic and morphological rules of the language. I call the first phase "deep generation" (after Brown [1973]) because it is the earliest stage in generation at which non-linguistic domain information gets augmented with linguistic information.

In this section enough will be said about the surface generation system MUMBLE to allow the reader to appreciate some of the issues involved in the process of jointly inventing rhetorical rules for GENARO while writing the "dictionary entries" for MUMBLE which allow it to "realize" GENARO's specifications.

MUMBLE is a transducer from a symbolic representation (a "meaning") into an English utterance (that expresses the "meaning"). The input to this transducer is an r-spec which is constructed by a "speaker" which knows what is to be said and how it is to be expressed. The r-spec is written in a language which is defined by

---

[1] I will refer to the *program* GENARO (the LISP code written for this thesis) as distinct from the larger generation *system*, which includes the simulated perceptual data base and MUMBLE.

MUMBLE's "*dictionary*"[2] : the legal elements of r-specs, and their legal combinations, are specified by entries in that dictionary, which also indicates how each element is to be "realized" in English. A dictionary entry makes *its* specification using the terms in MUMBLE's *grammar*. The grammar simply specifies the legal syntactic constructions available to the dictionary entries.

The major problem in realizing an r-spec is that both the r-spec and the grammar are sources of *constraints* on the final output. Hence the process of finding a realization is one of finding a construction (i.e. an utterance) that satisfies two sets of orthogonal constraints. One of the most interesting aspects of MUMBLE's operation is that it is "data-driven" – as the r-spec is interpreted a syntactic tree is constructed top-down and left-to-right, and at each node the question is "What grammatical construction can express this element of the r-spec". This is to be contrasted with a grammar-driven system in which the question at each node is "What element of the semantic input corresponds to the current grammatical term?"

Suppose one desired to have MUMBLE say the sentence "The fence is white". One extreme approach would be to have a single dictionary entry called "The-fence-is-white" which, when encountered as an element in an r-spec, directed the construction of a phrase structure tree whose leaf nodes were the words "the", "fence", "is", and "white". However, to assert that the fence is green (or any other color) would require a new dictionary entry. Even if this output were given internal linguistic structure by MUMBLE, it would nonetheless be a "canned" phrase.

A more flexible approach would be to take advantage of the fact that dictionary entries can have arguments (like functions). That is, have in the dictionary an "Assert-property(Object, Property)" entry which could then be invoked with the (input) r-spec element "Assert-property(Fence, White)". This entry would both invoke the appropriate grammar rules to construct the tree, and specify where to put its "object" argument (i.e. as the head of the subject noun phrase) and its "property" argument (i.e. as the adjective in the predicate adjective verb phrase).[3] Hence this entry will serve to express the possession by an object of any attribute which could be lexicalized as an adjective.

---

[2] N.B. While the dictionary defines the language in which the r-spec's are written, it is often the case that the dictionary entries are written to satisfy the needs of the r-spec elements. In fact the process of tuning the interface between GENARO and MUMBLE is a dialectical one of adjusting both the way in which r-spec elements are structured and the details of realization in the dictionary entries. See Chapter 4.2.

[3] The details of this tree construction and mapping semantic elements into structural slots is covered in detail in Chapter 4.2.

This level of detail is adequate for an understanding of the interactions between GENARO and MUMBLE that are discussed in this chapter. The operation of MUMBLE is discussed in much greater detail in Chapter 4.

3.2 The Control Structure of GENARO

Programs can be roughly divided into the part that knows things about the world, called "world knowledge", and the part that controls how this information is used, called the "control structure". Often the world knowledge is broken up into "modules" for flexibility and efficiency. The control structure does not "care" about the content of the world knowledge itself, it simply arbitrates which modules in the program are used and where their information flows. However, in A.I. the choice of control structure sometimes cannot be divorced from the claims being made about the knowledge contained in the modules which it controls. The designer of an A.I. system can either make the weak claim that the output behavior of the system captures some aspect of intelligent human behavior, or the much stronger claim that the system's internal mechanisms are operationally equivalent to human cognitive mechanisms. For example, under the strong claim interactions between modules which are made impossible by the control structure amount to a claim that these interactions do not occur in people performing the same task. In this thesis some of these stronger claims are made about the implementation of GENARO, and these are detailed in Chapter 5.

3.2.1 Packets and Iterative Proposing.

GENARO's control structure is basically a production system: a collection of independent production rules, each of which has a precondition and an action part. If the precondition part is satisfied, the action part "runs", producing whatever effects it was written to achieve.

Production systems provide a nice control paradigm because:

- the production rules are modular and independent "chunks" of knowledge about the domain, and are therefore quite flexible -- individual rules can be added and removed from the program without requiring any other changes; and

- since rules are independent they can be expressed in any order, thus they simulate a parallel machine, making this control regimen particularly interesting cognitively.

However, there are some disadvantages with production systems:

- exactly because of their independence, rules can be quite difficult to coordinate -- different rules can, in theory, cancel each other's effects, undermine each other's preconditions, and "cooperate" in complex ways that are difficult or impossible to predict; getting such systems to work well in practice thus requires considerable experimentation with and crafting of the rules; and

- in traditional production systems only one of the rules whose preconditions are satisfied is allowed to run [Nilsson 1980] -- part of the job of the control system then is to select which one gets to run, which can smear the distinction between what the rule knows and does and what the control system knows and does. (I.e., if the control system must use domain knowledge to choose among "winning" rules, then the control system is no longer domain independent.)

The last problem, of determining which rule to run next, is addressed in GENARO by two specific additions to the basic production system paradigm. One is the parcelling of the rules into "packets" (a la Marcus [1980]): each packet represents a situation in which its rules are appropriate, and packets are then turned on and off (thereby turning on and off the rules they contain) by a "driver" which is sensitive to the development of the paragraph. This technique provides a level of description in the control structure which often proves very useful (e.g. the "Introduce" packet in GENARO contains rules which introduce a paragraph description). The other enhancement to production systems that GENARO uses is called Iterative Proposing, which provides that the "action" of each rule is simply to *propose* its actual action on the data along with the "*priority*" with which it is making its proposal. The proposals of all of the rules are collected, and the one with the highest priority is chosen to perform its actual action. This is similar to the style of rule competition that was used in PANDEMONIUM [Selfridge, 1959].

The splitting of the traditional action part of a rule into a proposal and an action requires that one be specific about terms: when a rule's packet is on, I will call that rule "enabled"; when the precondition of a rule succeeds, that rule is "active" or "running"; when a rule runs, a "proposal" is submitted; finally, the result of choosing a proposal is that the rule's "action" is performed (sometimes referred to as the rule "winning").

Since the main work of GENARO is the construction of r-specs, the fundamental action is the insertion of a new rhetorical element into the r-spec. This is the action of most of the rules, although some rules have actions which are more "control-like" (e.g., shifting the topic).

The advantage of Iterative Proposing is two-fold: it leaves the choice of which of the active rules to run to the rules themselves (i.e. arbitration among the rules is internal to the set of rules), and it provides the system with a priority metric, which is recorded in the r-spec elements and which turns out to be useful to MUMBLE (see

Chapter 4).

3.2.2 The Organization of the System.

The overall organization of GENARO (presented in Figure 6) consists of 3 main parts:

- the 2 primary input data registers (through which the program examines the domain data base): the *Unused Salient Object List* (USOL) contains the *objects* in the domain, in order of decreasing salience, which have not yet been mentioned in the description; and the *Current-item*, which contains the object from the USOL which is currently under descriptive scrutiny by the rules. Two subregisters are part of the Current-item register: (1) a list of the properties of the Current-item object, in order of decreasing salience (called Current-properties), and (2) a list of the relations the object is part of (called Current-relations), also ordered.

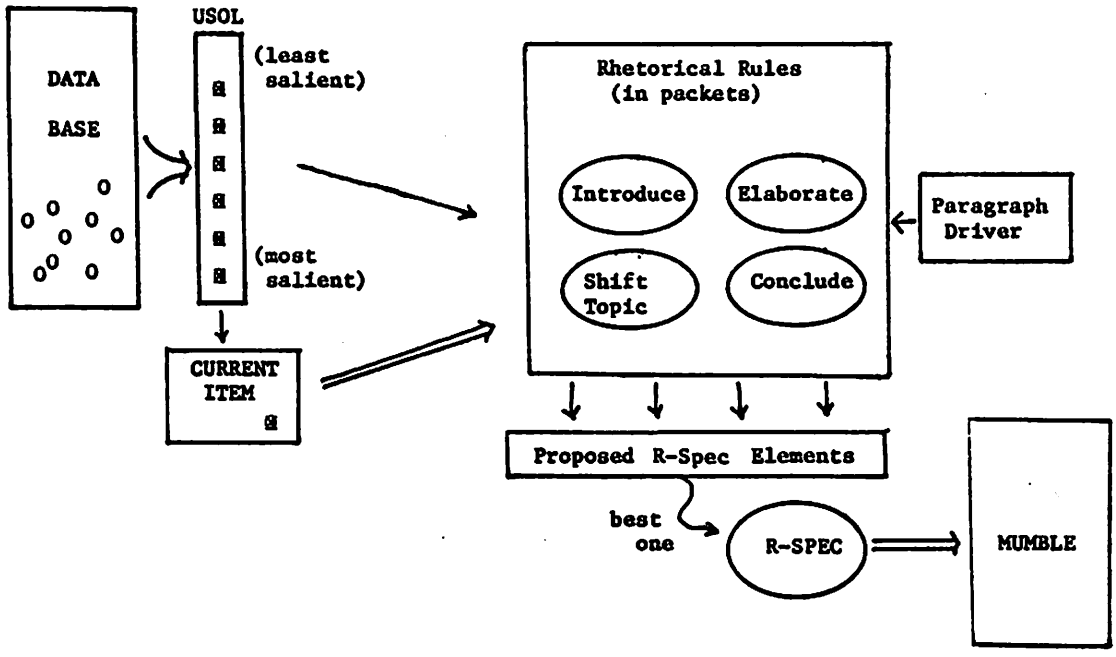
- the *rhetorical rules*, which are organized into packets. The precondition parts of the rules look primarily at the Current-Item, although they can also look down into the USOL, and even back into the domain data base, as well as into the R-spec.

- the current rhetorical-specification (i.e. the one under construction), or simply *r-spec*. The r-spec contains the information being assembled to send to MUMBLE, and is a list of r-spec elements (often just called "elements"), each of which is the result of the action of one rhetorical rule, and which specifies to MUMBLE the rhetorical effect intended by that rule. A paragraph is generated in a very local way: each r-spec is constructed largely on the basis of the domain data, without history about past r-specs or planning of future ones, and each r-spec is realized *as a single sentence* by MUMBLE.

Each of these parts has a deliberate connection to the theoretical claims about this program; however, in this chapter I will largely just describe how the program works (see Chapter 5 for a theoretical discussion of the system). The remainder of this section gives a more elaborate description of the data structures, ending with a brief description of GENARO's algorithm.

The USOL and the Current-Item.

The Unused Salient Object List (USOL) and the Current-item are the driving data structures of this program. Functionally they represent a view of the domain data base in terms of salience, almost as if one could pick up the data base and shake it so that it dangled in order of decreasing salience. The USOL can be



A block diagram of the GENARO system. The "O"s in the "Data Base" represent objects in the domain representation, whereas the "▣"s are the thematic "shadows" of these objects used by GENARO for its rhetorical processing. Each of the ovals in the "Rhetorical Rules" box is a packet containing one or more rhetorical rule.

Figure 6: The organization of GENARO.

thought of as the program's window into the data base, and the Current-item as the specific object being "viewed". Structurally, the USOL is simply an ordered list of

pointers to objects[4] in the domain; it is constructed at the beginning of the generation of a description by scanning the domain data base, collecting all of the individuated objects, and sorting them by their salience values.

The USOL is a programming convenience -- it saves repetitive and time-consuming scanning of the domain data base. However, if the data base were a dynamic one, in which concepts were being created and destroyed and salience values were changing (e.g. the visual data base of a moving robot), then the program would have to be modified to either (1) require that such changes be reflected in the USOL or (2) forego the USOL and scan the data base directly. Neither of these alternatives is inconsistent with the theory or operation of GENARO (although some care would have to be taken to prevent changes from taking place during critical periods of a rule's examination of the data base).

Objects are removed from the USOL in two ways during the course of a description. Normally, the most salient object on the list (the "top" of the list) is made the Current-Item, and is removed from the list (thus keeping the USOL a list of *unmentioned* objects). The other way, less often used, is that a rule "reaches down" into the USOL for the Current-Item (again, removing it).

The Rhetorical Rules.

Functionally each of the rhetorical rules captures a rhetorical or stylistic convention of descriptive paragraphs. The intention of the system design is to provide not only a testbed for prospective rhetorical rules but also a language in which they can be expressed. At present the system contains only about 8 rules[5] ; as the number of such rules grows there will be increasing evidence from which to specify this language precisely. Ultimately, writing the rules in a restricted language (the language of rhetorical rules) makes an implicit claim that the terms of the language are necessary and sufficient: the more constraining the language, the stronger (viz. the more predictive) the claim. I claim that the language described below does constitute such a theory for texts describing static situations.

Rules have two parts: a *precondition* and an *action*. These precondition contains one or more predicates (a term which is either true or false when applied to an argument), all of which must be true for the action to be performed. These predicates variously examine:

---

[4] By "objects" I mean simply the domain entities which denote actual objects in the world, e.g. houses, fences, trees, etc.

[5] This number reflects both the youth of this research endeavor, and the relative power of the individual rules.

- the salience of an item (the Current-item, its properties and relations, or some other object on the USOL);

- the "size" or contents of the r-spec (see below);

If the precondition is met, the rule proposes an action, but does not execute it – the action is merely *proposed*. The proposal has a *priority*, which reflects how important the rule regards its proposal to be, based on such things as the salience of the item being proposed and the "intrinsic priority" of proposals made by that rule. When the proposal with the highest priority is executed, it can:

- add one or several elements to (the end of) the r-spec (this is the most common action of a rule);

- take an item off of the USOL and make it the Current-Item;

- end the construction of the r-spec.

Each of the rules is contained in one of four packets: Introduce, Shift-topic, Elaborate, and Conclude. Packets are controlled by a part of the control structure called the *Paragraph Driver*. This is simply a demon-like routine which turns the various packets on and off, thus orchestrating the high-level structure of the output text.[6] At the beginning of a description the Paragraph Driver has the first three packets on (Conclude is off). Once any rule in the Introduce packet has (successfully) run that packet gets turned off. At the end of the description, when there are no more objects salient enough to describe, the Conclude packet is turned on. When the rules in the Conclude packet have nothing more to say the program ends.

Two of the packets are never used for high-level control: both the Shift-topic and Elaborate packets are left on for the whole description generation process. That is, the rules in these packets are not controlled by being turned on and off via their packets (they could in fact be combined into one "Develop-description" packet). Rather, there are two global numerical parameters, "*shift-factor" and "*level-of-detail", which are used by the rules in the Shift-topic and Elaborate packets, respectively, and which provide a more sensitive control than simply enabling and disabling the rules (and which are described below in section 3.2).

---

[6] It turns out that paragraphs describing scenes have very little high-level structure -- they are largely *"flat"*, and the structure they do have derives almost entirely from the domain data base (specifically, from the salience values there). Thus, the Paragraph Driver does very little work in the current version of GENARO. I expect, however, that it will serve an important function both in other, richer domains and in the generation of multi-paragraph texts.

Following is a list of the rules currently used by GENARO. These rules evolved through a largely empirical process: starting with about a half dozen rules (which were written on the basis of my own intuition), the program was run on the input data base, each time generating the r-specs for descriptive paragraph.[7] As weaknesses in these descriptions were observed, existing rules were "tuned up" and new rules were written. In later stages of development there have been several data bases on which to test the system, so that a greater range of rule interactions could be explored.

Rather than describe the actions of rules in terms of the intermediating processes of proposing and competing, I will simply say here what happens if the rule's proposal *wins*. Each of the rules listed here will be discussed in greater detail later in the thesis:

Introduce packet rules

$intro – This is currently the only rule in the Introduce packet; it proposes an r-spec element which essentially becomes, through MUMBLE, "This is a picture of ...". This rule has no preconditions, and its proposal is posted at a very high priority.

Elaborate packet rules

$prop-salience – Adds a property of the Current-item to the r-spec. The precondition is that the property be salient enough.

$prop-sal-obj – Specifically sees to it that the most salient property (if there is one) of prominent objects gets put into the r-spec.

$prop-color – This rule specifically adds the *color* of the Current-item to the r-spec. Its preconditions are that the color of the object exist and be sufficiently salient, and that the r-spec is not already too large.[8]

$reln-salience – This rule adds the Current-item's most salient relation to the r-spec. Its precondition is that the relation be salient[9] enough.

---

[7] Recall that the input data base represents a "machine's eye view" of the contents of a visual scene, and that the system's objective is to formulate a textual description that aptly describes the scene so represented.

[8] As will be discussed below, this rule reflects my observation that color was used in subject's descriptions out of proportion to the salience of the color, i.e. as a result of a stylistic device.

[9] Recall that the salience of a relation is based on the salience of the objects it relates, and on some properties of the relation itself (such as the 3-D distance between the objects).

Shift-topic packet rules

$newitem -- Takes the most salient object from the USOL and makes it the Current-Item. The only precondition for this rule is that the r-spec not be empty (i.e., if you have not said anything about the Current-Item yet, do not move on to a new item).

$condense-prop -- This is one of several very powerful rules for "condensing" the description of several objects into one r-spec based on some shared attribute of the objects. That is, these rules locate similar objects and propose describing them compactly using some parallel construction, e.g. "Both X and Y are Z". This rule condenses on the basis of a shared *property* between the Current-item and some object on the USOL (recall that the "U" stands for Unmentioned). Its preconditions are: that such a similar object exists, that the property shared by the two objects be salient enough, and that the r-spec is not already too large. The rule proposes at a low enough priority to assure that the Current-item has been reasonably described *before* this rule's proposal wins and the condensation occurs. The object that shares a property with the Current-item object is pulled out of the USOL and made the Current-item. Since there is a new Current-item, all of the rules come into play just as if it were the beginning of a new r-spec, and the descriptions of both objects are packed into one r-spec. (Condensing is discussed fully in Chapter 5).

Conclude packet rules

$light -- This is the only rule in the Conclude packet at this point. Its r-spec element describes the lighting of the scene (i.e. nighttime, cloudy day, etc.).

The rules will be further discussed below, both in section 3.2 and in Chapter 5.

The rule environment.

The rules listed above are *the* vessels of linguistic knowledge in the program -- if the program were a car, these rules would be the engine. In this section, the environment in which the rules operate -- the engine compartment -- will be described.

The gasoline of this system is the input data base. That is, GENARO is "*data-driven*" by the objects, properties, and relationships that it finds in the visual representation. These entities determine, via their salience values, what gets mentioned, when, and in what detail. This is in direct contrast to a top-down approach in which text is structured by the rhetorical rules (i.e. [McKeown 1982]). In GENARO the information in the rules gives a rhetorically and stylistically pleasing "shape" to the text, but it is the data base entities, and particularly their salience, that determine the length and level of detail of the text.

GENARO does not, however, make any changes in the domain data base.[10] Instead, it makes a copy of each entity in the data base and uses this new data object for its internal manipulation and book-keeping. These are called *"thematic-objects"*. There are four kinds: those for the objects, properties, and relations in the data base, and a fourth kind, called "rhetorical thematic-objects", which are not a copy of anything and which are created by GENARO to become r-spec elements with specific rhetorical effects. For example, the rule $intro constructs a rhetorical thematic-object which has an entry in MUMBLE's dictionary and which is realized as something like "This is a picture of ...". Other such thematic-objects can signal MUMBLE to perform conjunction and other syntactic constructions.

Finally, the r-spec is a simple list composed entirely of r-spec elements, and all elements are thematic-objects: an r-spec is constructed through the addition, by the rules, of thematic-objects to it. (However, only a few of the thematic-objects created for a description ever make it into an r-spec -- the rest are simply discarded.) The r-spec is built in strictly linear fashion: new elements are added to the end of the list.[11] (The distinction between domain objects, thematic-objects, and r-spec elements will become clearer in the discussion of GENARO's algorithm below.)

Another important element of the environment is a mechanism for *measuring* the r-spec. Starting from the assumption that small r-spec's produce simple sentences and large r-spec's produce complex sentences, GENARO requires a means of estimating the complexity of the text that MUMBLE will be generating from the r-specs that it receives. This is done through the notion of the "weight" of the r-spec. A simple numerical value -- a "weight" -- is assigned to each kind of thematic-object, based on the textual size and complexity that it is liable to incur. As an r-spec is contructed, it can be weighed at any point simply by summing the weights of its elements.

For example, since properties of objects are almost always realized simply as prenominal adjectives, properties contribute relatively little weight to the r-spec. On the other hand, rhetorical thematic-objects signalling conjunction and other complex syntactic constructions contribute relatively large amounts of weight. Interestingly, one exception to the large weight of rhetorical thematic-objects are those which signal condensation, since the whole idea of a condensation is the application of essentially the same descriptive text to several objects. For example, the sentence "The door and the gate are red" is not much more complex than "The door is red", although the r-spec specifying the former sentence is considerably larger.

---

[10] See Chapter 5.2 for a proposed exception to this convention.

[11] This leaves the burden of any "sorting" that must happen among r-spec elements to MUMBLE, a job which it is already well suited to perform because it does a quick first-pass through each new r-spec looking for elements which it knows to contain global constraints on the syntactic structure.

In the research reported here thematic-objects were given the following weights: Thus, an r-spec consisting of an object[12] (weight = 0), an Introduce element (weight = 2), a property (weight = .5), and a relation (weight = 1) will weigh 3.5. This has been determined, through experimentation with the system, to be an optimum weight for an r-spec. For example, this is the weight of "This is a scene of a suburban house with a fence in front of it". The sentence "The house has a red door" has a weight of 1.5 (a property plus a relation), while "In front of the house is a white picket fence with a red gate and in front of that is a sidewalk with a person walking on it" has a weight of 8.0 (four properties plus four relations plus a rhetorical element for the conjunction).

Another element of the rule environment is the distinction between "unique" and "common" r-spec elements. In an average round of proposing, several of the rhetorical rules listed above will be active, producing several proposals. Only the best of these proposals "wins", resulting in either a new element added to the r-spec (usually) or some other action. However, the "losers" are free to run over and over again, until their proposal succeeds or the Current-item is changed. For rules which specify their proposal to be *unique*, success disables that rule for the duration of the construction of that r-spec (e.g. the $intro rule only wins once), while other rules, whose proposals are *common*, depend on predicates in their preconditions to keep them from winning excessively (e.g. condensing can occur several times in one r-spec, and is limited only by the condense rules being sensitive to the "weight" of the r-spec).

thematic-object type	*Weight*
object	0
property	.5
relation	1.0
rhetorical	2.0
condense	.5

These values have been arrived at through repeated experiments with the program and although not precise, they seem to be adequate to the coarse function that they serve.

Table 1: Thematic-object weights.

[12] In this thesis objects were not inserted as their own r-spec elements, as implied in Table . Whether or not objects are needed as independent elements depends on the processing demands (by MUMBLE) on r-specs which have condense relations in them, and this will become clearer as more experimentation is done with the various $condense rules. In any case, such r-spec elements would serve a strictly syntactic function as place-holders in the r-spec, and thus would have zero weight.

The designer of a rhetorical rule thus has two mechanisms available for determining when the rule will win. One is to place predicates in the rule's precondition that block the rule from even proposing. For example, $prop-salience has a predicate in its precondition for testing if the salience of the property that it might propose is salient enough. This is a *local* constraint in the sense that the precondition is based just on the salience of some property, and not on what other rules are doing. Constraining rules through their predicates is a useful in cutting down on the number of "serious" proposals which the control structure must handle.[13]

The other mechanism for controlling a rule is through the manipulation of the priority at which it posts its proposals; this strategy is useful for more global decisions[14] , since the arbitration of priority values happens among all active rules and hence takes all active rhetorical factors into account. For example, $prop-color proposes mentioning the color of the Current-item regardless of the salience of the color – but it posts its proposal at a low enough priority that it will only win if there is nothing more important to say about the Current-item. In other words, although the rules are independent and highly localized sources of rhetorical information, they communicate weakly through the process of their competition during the proposing process. When $prop-color wins it has essentially received the "message" from all other active rules that there is room in the r-spec for the color of the Current-item and that it may now insert that element into the r-spec.[15]

The various rules in the system to date are listed with the type of control they use in Table 2. "Local control" is via rule predicates checking salience values of items, whereas "Global control" is via rules competing through the priority values with which they post their proposals, Note that since *all* rules are controlled through the competition of priority values, this mode of interaction is only marked here as "Global control" when there is no "Local" use of salience values.

---

[13] This is a different level of locality from that in the claim that GENARO's planning is local in scope. In the latter use of the term, the point is that GENARO does not keep track in any explicit way of the structure of the text being generated, nor does it have any sense of a "goal" towards which is it working.

[14] These local and global mechanisms are not completely separate: for a rule's action to take effect it needs *both* (a) all of its preconditions to succeed and (b) its proposal to have the highest priority. These two mechanisms thus act in series, and a failure on either account prevents the rule's effect from being achieved.

[15] In order to prevent this rule from putting the Current-item color into full r-specs it has a predicate which checks the r-spec's weight in its precondition part. There is an alternative to having each rule monitoring the size of the r-spec: a production rule called $finish-building-rspec, which proposes ending the construction of an r-spec at a priority which is a function of the weight of the r-spec. This idea is discussed in Chapter 6, and in fact is illustrated in the sample program runs in Appendix 2.

	Control Style		
Rule	*Packet*	*Local*	*Global*
$intro	Intro		x
$prop-salience	Elaborate	x	
$prop-sal-obj	"	x	
$prop-color	"		x
$reln-salience	"	x	
$newitem	Shift-topic		x
$condense-prop	"	x	
$lighting	Conclude		x

Table 2: Rules and their control style.

The rules which use local control are more data-driven than the global control rules -- they are very sensitive to what is in the visual representation, especially to the salience values. The global control rules, on the other hand, are the ones which are sensitive to more rhetorical concerns, and which interact primarily in the realm of the rhetorical priorities. Thus there are two broad classes of rules in the system, one for rules which are data- and salience-sensitive and which function to determine the *contents* of the r-spec (i.e. *what* is said), and one for rules which function to make the r-spec *rhetorically well-formed* (i.e. more *how* it is said) and which compete with each other via their rhetorical priorities.

This illustrates that at the level of deep generation both the What and How issues are important, although there is not a rigid separation between how these two concerns are processed. It also leads to a formulation of the deep generation process as an interaction between two forces: one which seeks to have the structure of the paragraph reflect the structure of the data (i.e. the data-driven part), and one which operates to impose stylistic, thematic, and rhetorical (all of which I have generally lumped into the term "rhetorical") constraints on the paragraph structure. This is intriguingly parallel to McDonald's formulation of the surface generation process (i.e. realization) as the resolution of a force to fully express the contents of the r-spec with a force which constrains the utterance to remain within the grammar.

The R-spec.

The r-spec can be thought of as the "idea" -- the essential semantic content --

which is intended to be expressed by the speaker.[15] It is constructed to convey all of the important decisions as to content and style to the realization component. In this system I have further specified that it form a channel with as narrow bandwidth as possible between the planner and realization components. The r-spec should contain all and only the information needed, and the realization component cannot ask for disambiguation or elaboration on an r-spec it has received. In fact, if the realization component does not succeed in finding a realization for an r-spec it can only signal this failure. (See Chapter 5.1 for a discussion of these GENARO/MUMBLE interactions.)

In GENARO the r-spec is a rhetorical "molecule" which is constructed in a register (called "R-spec"). The "atoms" of this molecule are the "specification elements" (sometimes called simply "elements"). Every element in an r-spec has some specific rhetorical function, and every element was added to the r-spec by some rhetorical rule (though some rules can add several elements). Elements cannot be changed or deleted once they are in the r-spec. Each element has a name (so that they can refer to each other), and each has a "themeobj-operator" which specifies a realization action via MUMBLE's "dictionary".

R-spec elements can be of four types: object, property, relation, and "rhetoreme". The first three correspond to the three classes of domain entities in the domain data base; to mention that an object has the property Red, for example, an element of type *property* would be inserted in the r-spec. The fourth type, "*rhetoreme*", is for elements which have no domain correlate and which are inserted only for their rhetorical function (e.g. Introduce(x)). The name (which I coined) indicates that these are the smallest units of rhetorical information which the system has.

The grammar of the r-spec is shown in Figure 7, and an example of an r-spec (for "This is a picture of a white house.") is presented in Figure 8. The first major part of the r-spec, the "body", specifies those elements which must be expressed in the realization, while the "optional" contains the elements which are optional. This distinction allows the planner to build large, complex r-specs but to "soften" them by

---

[15] It could also be considered as a hypothesis on the form of the "language of thought". I think, however, that current r-spec format is too poor even to be considered as a hypothesis on the "universal form". (It does not address the issues of intentionality and self-reference, for example.)

```
<r-spec> :== (RSPEC NO.i BODY [<rspec-elmt>]
                              OPTIONAL [<rspec-elmt>])
<rspec-elmt> :== (ELMT.j <elmt-type>
                         <elmt-header>
                         <elmt-body>)
<elmt-type> :== OBJECT, PROPERTY, RELATION, RHETOREME
<elmt-header> :== <themeobj-operator>
                  ([<themeobj-operand>])
<elmt-body> :== [<themeobj-spec>]
<themeobj-spec> :== (<themeobj-operand>
                         [<themeobj-qualifier>])
<themeobj-qualifier> :== (SUPERC <themeobj-superconcept>),
                              NEWITEM,
                              IBID
<themeobj-operator> :== a thematic object
<themeobj-operand>  :==     "     "       "
<themeobj-superconcept> :== "     "       "
```

Terms in capital letters are constants; terms in <>'s are nonterminals; square brackets ([]'s) indicate *"one or more of"* the enclosed term; commas separate equal alternatives; the letters "i" and "j" are used as index variables, e.g. "NO.i" would become "NO1" the first time, "NO2" the second time, etc.

Figure 7: The grammar of r-specs.

```
(RSPEC NO1
   BODY
     (ELMT1 RHETOREME introduce (house-1)
          (house-1 (SUPERC house)
                   NEWITEM))
     (ELMT2 PROPERTY  color-of (house-1 white-1)
          (house-1 IBID)
          (white-1 (SUPERC white)))
   OPTIONAL nil)
```

This example r-spec might result in the sentence "This is a picture of a white house". "SUPERC" names the super-concept (from the domain data base) of the object (for use in lexicalization). "IBID" indicates that there is an entry for the object already in the r-spec. "NEWITEM" is explained in the text.

Figure 8: An example r-spec.

allowing the realization component to ignore difficult and unnecessary elements.[17]

The <elmt-header> is the part of the specification element that makes direct contact to the dictionary in MUMBLE, and can be thought of as a function call: the themeobj-operator is the name of the "function", and the themeobj-operands are the parameters with which it is called. (This is in fact literally the case within MUMBLE, where each dictionary entry is implemented as a LISP function.)

For each themeobj-operand in the header there is a corresponding <themeobj-spec> which provides further information to the dictionary about how to treat the operands. For example, when an object in the data base is being introduced into the text for the first time it is marked as a "NEWITEM" so that MUMBLE can provide the appropriate sentence structure and determiner.

What is surprising, perhaps, is that the single marker NEWITEM can carry as much information as it does to MUMBLE. It would certainly be possible for GENARO to specify the details of the actions MUMBLE should take with a new item, e.g. using an indefinite determiner with that object's noun phrase and making it subject of the sentence. But the fact that this information can be more parsimoniously represented in MUMBLE (in terms of how to process the NEWITEM marker), and that the channel between programs is much narrower simply passing the NEWITEM marker, suggests that this is a natural and important division of labor between deep and surface generation. (For further discussion, see Chapter 5.)

Note also that none of the examples presented in this thesis use the "optional" part of the r-spec. However, it is offered here because it appears to provide considerable power and flexibility to the GENARO/MUMBLE interface while increasing the interface bandwidth only minimally. Also, its use is supported by the intuitive sense of foregoing saying something that would fit semantically in the current utterance but which would make the sentence syntactically too complex or overloaded. (Note also that its use would have to be constrained so that it could hold only a few elements: otherwise, it could be used to "sneak" the whole data base into the r-spec, leaving deep generation to be done by MUMBLE.)

---

[17] It may be that there is a need for a part of the r-spec to be about more global rhetorical parameters, such as tense, aspect, mood, tone, as well as temporal anaphora, quantifier scope, negation, etc. I have chosen to deliberately exclude these issues from my research (see Chapter 6). Eventually it will be important to study how these more global rhetorical concerns can be incorporated into the simple machinery presented here (my hope is that they can be added as independent self-contained modules), but for the present they must be ignored.

3.2.3 The Algorithm.

Like any production rule based system, GENARO has a very simple algorithm: pick which rule to run and then run it, while maintaining the data structures used by the rules. Through the use of Iterative Proposing GENARO goes one step further: the rules themselves pick which one runs.

The basic algorithm for the program GENARO is shown in Figure 9 (see Appendix 1 for a more detailed listing of the program). Readers unfamiliar with LISP may find Figure 10 helpful in understanding the algorithm. The top-level routine, GENARO, simply iterates on building r-specs and sending them to MUMBLE until the end of the description is signalled.

Building an r-spec is done as follows: all of the proposed actions are gathered together on the "proposed-rspec-elmts-list" and the register Rspec-elmt is set to the one with the highest priority. Then, the function "insert-in-rspec" runs, either adding the contents of Rspec-elmt to the end of R-spec, or, occasionally, performing the specialized action described next.

```
GENARO
    (initialize-system)
    (setq USOL (Order-by-salience
                        (Get-objects Domain-data)))
    (set-current-item (pop USOL))
    (repeat
            (init-for-rspec)
            (build-rspec)
            (send-rspec-to-mumble)
     until  (end-description?))

BUILD-RSPEC
     (repeat
            (initialize-for-proposing)
            (propose-rspec-elmts)
            (setq rspec-elmt (pick-best-proposal))
            (insert-in-rspec rspec-elmt)
      until (rspec-complete?))

PROPOSE-RSPEC-ELMTS
     (mapcar *rule-list eval)  ; Runs the rules
```

Figure 9: The basic algorithm of GENARO.

In LISP notation each call to a function (termed "evaluation of the function") has the form "(<function-name> <arguments>)", where the name of the function comes first and is followed by the arguments, or "parameters", which the function is being asked to operate on. For example, the expression "(setq X (add1 5))" means to first find the value of "(add1 5)", which is 6, and then to set the variable "X" to this value; the result is that the value of X is 6.

Good LISP programming style dictates that function names and their arguments be highly descriptive, but some of the functions in Figure 9 may not be transparent. "Pop" is a function which returns an item on the top of a stack while removing that item from the stack. "Mapcar" takes the function named in the second argument and applies it to each item in the first argument list. "Eval" is the basic LISP function which "evaluates" (i.e. for functions, runs) LISP objects.

Figure 10: A few notes on LISP.

Sometimes Rspec-elmt is not really a rhetorical element destined for the r-spec – sometimes it is a *control action* to be performed. For example, the action of the rule $newitem adds nothing to the r-spec, but rather resets the Current-item register to the next unmentioned object on the USOL. Therefore, special provision has to be made for such "active" "rspec-elements".[18] If Rspec-elmt is specially marked (as a function), then "insert-in-rspec" *evaluates* the element, performing the action. The action can be either setting "Cur-item" to be a new object or setting a flag which causes the predicate "rspec-complete?" to be true.[19]

3.3 The Rules and their Interactions

The use of production rules is an instance of data-directed control, a style of programming in which control actions are explicitly bound to specific structures in the input. The advantage of representing the rhetorical rules of the system in this way is that they are compact, independent, and flexible units. They represent my best

---

[18] This "hack" is the mechanism that allows rhetorical rules to take on elements of the control process, manipulating the program's registers and stacks. It is not cheating, however – none of these control actions goes beyond the limits specified above (page 23). The names of the functions in this part of the code should perhaps be revised to reflect the fact that more is going on than just inserting elements in the r-spec.

[19] The use of a flag, normally a poor practice, here provides a bit more clarity on the termination condition of the repeat/until loop.

attempt to date on the formalization of rhetorical and stylistic conventions of descriptive texts, and as such constitute a "young" linguistic theory. One design goal of the system, in fact, is that it provide a test-bed in which "rhetorical grammars" (i.e. a set of rhetorical-rules) can be constructed and tested. Hence, what is perhaps more interesting than the specific rules presented here is the *process* by which they were developed, and the contraints and tradeoffs which this process revealed. This is a somewhat intuitive and subjective process -- as the specification of rhetorical and stylistic conventions will probably always be.

In the past the problem with specifying rhetorical conventions in any detail has been that there was no *language*, save English, with which they could be described -- no clearly specifiable "hooks" into the semantic world of thoughts and ideas from which the text was to flow, and no precise delineation of where rhetorical planning ended and grammatical started. With GENARO these hooks are provided, at the cost of a very specific set of computational constraints. What makes the exercise of rule-writing interesting is that the language in which rules may be written for GENARO is quite constrained, hopefully in natural ways. (To facilitate the discussion of the rules, the following section outlines the primitives of their language.)

There is another important point here, one that will be elaborated in Chapter 5 -- that the writer of rhetorical rules for GENARO will also be writing the grammatical rules for MUMBLE, and that there is an intimate connection between these two sources of knowledge. (Particularly interesting are issues about where to put decision-making processes which *could* be placed either in GENARO or MUMBLE.) Operationally, finding ways to give linguistic expression to facts in the domain data base is a process of coordinating knowledge at the distinctly different levels of deep and surface generation.

For example, in GENARO the r-spec is built in a strict linear fashion, without any restructuring or editing before it is sent to MUMBLE. This fact, along with the lack of any kind of backtracking in the control structure, constitutes a strong processing claim. Not only is GENARO doing deep generation without any attempt at planning-like optimization of the anticipated rhetorical effects, but its decisions are indelible -- MUMBLE receives rhetorical elements in exactly the order in which they were offered by the rules.

3.3.1 The Rhetorical Primitives.

It was stated above (page 23) that the rhetorical rules make a series of linguistic claims about the structure and content of English descriptive paragraphs. Similarly, the terms in which the rules are written specify a set of operations which can be thought of as the primitives of examining and asserting rhetorical elements for the r-spec. Of course, these operations are dependent on the "machinery" (i.e. the control stucture) of this particular system -- they could not be "plugged in" to another deep generation program without considerable translation effort. It is nonetheless

desirable to formulate these operations as abstractly and generally as possible, since it is at least possible that these primitives are more widely applicable than within the specific framework of GENARO. Therefore, below are listed the primitive operations which are used by the rhetorical rules.

The *Preconditions* of the rules can examine:

1. the *salience* of the Current-item;
2. the *salience* or *existence* of properties and relations of the Current-item;
3. the *salience* or *existence* of objects, properties, or relations on the USOL;
4. the *"weight"* of the r-spec under construction (see discussion page 27); and
5. the *contents* (elements) of the r-spec under construction.

The rules specifically *cannot* examine:

- previous r-specs;
- the pending proposals of other rules (during a round of proposing);
- any future event -- rules are not allowed to post "demons" which wait for some condition in the future which, when true, triggers an action or proposal.

The action parts of the rules can:

1. propose an r-spec element (this is the most common action);
2. get a new Current-item, either through the normal action of simply "popping" the USOL or, in the case of the condense rules, by reaching down into the USOL and pulling out an object; and
3. terminate the construction of the current r-spec.

Part of the methodology employed in this research has been the assumption that the weaker and more restrictive the primitive operations are, the more potent is the theory which they implicitly describe. Conversely, if arbitrary LISP functions could be expressed in the language of the rules (i.e. if there was a mapping from LISP to the rule language), then the virtual machine in which the rules run would have the power of a Turing machine, weakening considerably the claims of the theory.

The operations listed above describe a virtual machine that can be characterised as being very *localized* -- it only supports rules which are "short-sighted" and which have immediate effects. To summarize this machine:

> There are three data structures: the USOL, the Current-item, and the R-spec; the first two represent successive distillations of the domain data base, and are available only for inspection and a very limited form of modification by the system's rules (i.e. movement of items from the USOL to the Current-item); the R-spec is also examinable, and is the only place where planned rhetorical elements can be placed. The rules must perform their action at the moment that their proposal wins, and there is no backup or lookahead.[20] Once construction of an r-spec is completed it is sent immediately to MUMBLE, and the process is begun again with an empty r-spec and a new Current-item from the USOL.

There is one exception to the extreme locality reflected in this architecture, and that happens in the Paragraph Driver. Since this part of the machine turns the packets on and off, it has the ability to impose a global structure on the generated text. Currently this mechanism is used only to introduce and conclude the paragraph, using the packets Introduce and Conclude. (See Chapter 6 for a discussion of using this mechanism for multi-paragraph texts.)[21]

The advantage of this approach is

- that this machine is adequate to do salience-based rhetorical planning,
- that exactly because of the weakness of the machinery it can perform its planning very quickly, and
- that because of the possibility of "dead-ending" in its interaction with MUMBLE it presents a testable and psycholinguistically interesting theory of deep generation.

---

[20] Backup provides a program with the ability to make a "tentative" decision -- if it turns out to be wrong later on, the program can return to the point where the decision was made and make another choice. Lookahead allows a similar kind of flexibility -- the program can "look ahead" to some extent, anticipating the consequences of the alternatives at a decision point, and hence making a more informed choice. However, GENARO's decisions, like MUMBLE's, are indelible and unforeseeing.

[21] Another way of viewing the overall architecture of GENARO is that it is comprised of two very simple and weak machines, an Iterative Proposing rule driver and a Paragraph driver, connected in a way to provide performance well beyond either machine alone.

3.3.2 Writing Rhetorical Rules.

The experiments on salience (reported in [Conklin, Ehrlich, and McDonald 1983]) yielded a substantial body of texts and their associated salience ratings. These texts were used as the starting point in the design of the rules, although my own intuitions as a native speaker also played a large role. The rest of this section will present the process by which two of the simplest rhetorical rules evolved, illustrating the empirical approach which GENARO was designed to support. Familiarity with this section should allow the reader conversant in LISP to formulate their own GENARO-style rhetorical rules. The section following this one will present a detailed example of the generation of a short description, so questions about the *operation* of the rules presented next should be suspended until both sections have been read.

Many of the subjects' descriptions began with the phrase "This is a picture of ..." or "This picture shows ...". The job of designing rhetorical rules for GENARO generally has two parts: deciding *when* to use a rhetorical construction, and specifying *how* to the realization component in a concise form. In this case (i.e., at the beginning of a description), the decision to use it is could be simple: simply detect that nothing has been said yet.[22] In fact, it is even easier to simply start the system up with the Introduce packet turned on, and then to turn it off at the end of the first r-spec. The rule which introduces the description, $intro, therefore has no preconditions (it is controlled via its packet).

The action part of the rule must somehow tell MUMBLE to construct the tree structure for the phrase "This is a picture of" and to leave a hook in the structure for the noun phrase describing the theme of the picture. The difficult part of this choice is that there are so *few* constraints on it. The simplest action is to pass to MUMBLE an r-spec like "Introduce (X)", where MUMBLE's dictionary has an entry for "Introduce" which constructs the phrase marker shown in Figure 11, and which binds "X", the thing introduced, to the final NP position. A similar structure could be built for the phrase "This picture shows", and the choice between them could be left as a random decision to be made by MUMBLE (barring some criterion on which to base the decision).[23]

---

[22] This *could* be done by using a "nothing-has-been-said-yet" flag, but no such flag is available to the rules, nor do they have the ability to set or check it. (This is not to say that it could not be added, but to do so would be to extend the rhetorical rule language, and, in the interest of parsimony, such extensions should only be made as the result of a careful argument.)

[23] As will be explained in Chapter 4, MUMBLE always forms its phrases grammatically, using a tree representation of the internal structure. While a phrase could be represented as a "*word*" in MUMBLE, this is a bad way to capture idioms and other canned phrases since it forbids any morphological modifications to words inside the phrase.

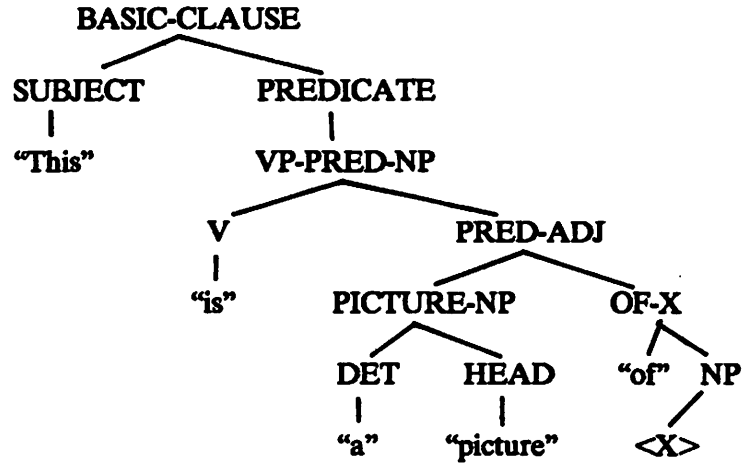


Figure 11: The tree for Introduce.

The rule must specify what object in the domain data base to use as the argument to this "Introduce" term. The obvious choice is the Current-item, since at the beginning of the description process the Current-item will be the most salient object in the domain data base.[24]

The Introduce specification element is proposed at a high priority because it is an *obligatory* element in the first r-spec (i.e. if the rule *can* fire, it *must*). Priorities normally range from zero to one, so I chose to propose this element with a priority of two times the salience of the Current-item (which salience will normally be close to 1.0), assuring that it will normally be the first to win. (If there are *no* very salient objects in the picture, however, the priority will be low enough that some other introductory rule which is designed to handle such situations can win.)

Finally, the rule must be prevented from firing more than once. Within the first r-spec this is done by proposing the Introduce element as "unique" (see discussion, page 28): once this proposal has won, the control system will prevent it from being proposed again during the construction of the current r-spec. To prevent this rule from firing in *subsequent* r-specs, it is in a packet (the Intro packet) which is turned off when the first r-spec is sent.

---

[24] There are complications to this simple scheme. If the picture is of *two* houses, of roughly equal salience, the desired output would be "This is a picture of two houses ...". However, the Introduce term would have already been bound to only one of these houses (the more salient one). To keep this section simple such complications will be discussed in Chapter 5.

Thus, this rule could be paraphrased as saying simply "Propose to introduce the Current-item as the topic of the picture." The actual form of the rule used in GENARO is shown in Figure 12. The function "propose" takes two arguments: the first is the r-spec element which is created by "create-rspec-elmt", and the second is the numerical value of the priority at which the proposal is to be made (the "at" is ignored). The function "create-rspec-elmt" takes four arguments: the first two specify the type and name of the r-spec element, the third one names the theme-object which is the "argument" of this r-spec element (i.e. the one it refers to), and the fourth argument specifies that this element is to be unique (i.e. if it wins, no other such proposals will be entertained). When this rule's proposal wins, the specification element that is inserted into the r-spec is shown in Figure 13 (see Figure 8 above).

```
($intro
    preconditions
    actions
       (propose
            (create-rspec-elmt 'rhetoreme
                               'introduce
                               Current-item
                               'unique)
        at (* 2.0 (get-salience Current-item))))
```

This rule is controlled by being in the Introduce packet. The function "create-rspec-elmt" is explained in the text. (Keep in mind that the action of this rule is simply to make a proposal -- the created r-spec element is included in the r-spec only if that proposal has the highest priority.)

Figure 12: The rule $intro.

```
(ELMT1 RHETOREME introduce (house-1)
    (house-1 (SUPERC house)
             NEWITEM))
```

This element specifies to MUMBLE that the dictionary entry *introduce* be run with the single "argument" *house-1*. "SUPERC house" names the immediate superconcept of house-1, and NEWITEM is added because the theme-object house-1 is marked as "unmentioned".

Figure 13: The first r-spec element.

As another illustration of capturing rhetorical conventions, it was observed that people rarely mentioned prominent objects without modifying the object's noun phrase in some way. "This is a picture of a *white* house" or "... a *New England* house" were preferred to the more austere "This is a picture of a house". To capture this convention I wrote a rule called "$prop-sal-obj" (because it proposes a property of a salient object). It had the following requirements:

1. that it only propose mentioning a property if the Current-item was salient enough, and

2. that it propose the most salient of that object's properties.

The resulting rule is quite simple, and is shown in Figure 14. It could be paraphrased as saying "If the Current-item is very salient and has any properties at all in the domain data base, propose saying the most salient of those properties."

The first requirement (above) is fulfilled by the precondition on the salience of the Current-item (a value of .9 represents a very high threshold). The second requirement is fulfilled automatically by using the fact that part of the Current-item register is the Current-properties list, which lists in order of decreasing salience the properties of the Current-item; the first of these is thus the most salient property. This proposal is made at a fixed priority of .35: low enough that it only wins when all of the fairly important elements are already in the r-spec, but high enough to prevent it being skipped over. Notice that the salience of the property itself is not at issue in this rule – the fact that the house is "white" or "New England style" is

```
($prop-sal-obj
    preconditions
        (greater-than (get-salience current-item) .9)
        (not (null current-properties))
    actions
        (propose
            (create-rspec-elmt 'property
                               (first current-properties)
                               current-item
                               'unique)
         at .35)
```

The function "greater-than" is true if the first argument is greater than the second one. The function "first" returns the most salient property of the Current-item, because "current-properties" is ordered by decreasing salience.

Figure 14: The rule $prop-sal-obj.

hardly salient. However, prominent *objects* receive "elaboration" through this simple technique. When this rule's proposal wins, the following element is added to the r-spec:

```
(ELMT2 PROPERTY  color-of (house-1 white-1)
    (house-1 IBID)
    (white-1 (SUPERC white))))
```

(again, see Figure 8).

Before leaving the subject of writing and coordinating rhetorical rules there are two global variables, *shift-factor and *level-of-detail, which deserve elaboration. The purpose of these variables, which are called *rule parameters*, is to coordinate the rules at a global level.[25] For example, the rules in the Shift-topic packet use *shift-factor to coordinate the salience at which they will propose actions leading to a new Current-item. Likewise, *level-of-detail is used by the rules in the Elaborate packet to coordinate the salience required for details to be proposed. Both rule parameters are multiplicative factors and have a standard value of 1.0 (hence by default they have no influence on the system's behavior).

In this section I have tried to provide the flavor of composing rhetorical rules for GENARO. In the next and final section of this chapter a fairly detailed example of the generation of a short description will be presented.

3.4 An Example of Generating a Description

The purpose of this section is to provide the reader with a more concrete sense of the way the system works, especially with respect to the interactions between the rhetorical rules, and the tension between making decisions on the rhetorical (GENARO) versus the grammatical (MUMBLE) side of the fence. The picture being described will be the house scene shown in Figure 1 (page 4), and the input domain data base will be the hand-built KL-ONE network representing this picture shown in Chapter 4. The following account discusses the operation of GENARO through the construction of the four r-specs which lead to the following target description:

---

[25] This is not coordination in the sense of inter-rule communication, since the rules cannot *set* these parameters, only obey them.

> This is a picture of a two story house with a fence in front of it. The house has a red door and the fence has a red gate. Next to the house is a driveway. It is a cloudy day.

The reader may wish to refer to the basic algorithm in Figure 9 (on page 34) in reading the following account. This account does not explain the process of realizing the deep-generated r-spec's in MUMBLE – that is left for Chapter 4.

3.4.1 The First R-spec.

The program starts with "initialize-system", which simply does the initial bookkeeping for the control system. Next the objects in the domain data base are ordered by decreasing salience and the USOL is set to this list.[26] Let us suppose that this process yields the USOL shown in Figure 15.

The next action is that the top object on this list (House-1) is "popped" into the Current-item register; an automatic part of this process is that the two "sub-registers", Current-properties and Current-relations, are set to the lists of the

USOL object	*Salience*
House-1	1.0
Fence-1	.9
Door-1	.8
Gate-1	.7
Driveway-1	.6
Mailbox-1	.5
Porch-1	.5
etc.	

The objects on the USOL for this example and their respective salience values. (These salience values were adjusted to simplify this exposition of the operation of the program, and are not the empirically derived values.)

Figure 15: The USOL for this example.

[26] One implementation detail is that all of the individuated objects in the network (i.e. those representing actual objects in the world) share a common superconcept called "**individuated-objects". This concept provides quick access to the objects in the scene – it only remains to access the salience value for each of these and then sort this list of objects on the basis of their respective salience values.

properties and relations, respectively, of the Current-item (and these lists are also in order of decreasing salience). In this example Current-item would now be as shown in Figure 16. Finally, the R-spec register is set to the skeleton:

(RSPEC NO1 BODY nil OPTIONAL nil)

where the "nils" are empty lists (see Figure 7).

The program is now ready to begin building the first r-spec. The function "build-rspec" does this job by repetitively having all of the rules make their proposals and then performing the action specified by the proposal with the highest priority. During the proposing phase the proposals are kept on the "proposed-rspec-elmt-list", which is maintained in order of decreasing priority. As shown in Table 3, in this example six of the 8 rules in the system make proposals. Each of these proposals is described below in some detail.[27] (While this level of detail is not practical for the whole example, it is necessary in the beginning.)

Current-item: House-1
Current-properties: (two-story-building-1
white-1
new-england-house-1)
Current-relations: (in-front-of-1 {Fence-1}
part-of-8 {Door-1}
next-to-3 {Driveway-1}
part-of-9 {Porch-1}
etc.)

The Current-item register and its subregisters. The lists in each of the subregisters are ordered by decreasing salience (although the salience values are not shown here). The objects with which the Current-item has the various relations shown in the Current-relations subregister are shown in {}'s.

Figure 16: The first Current-item.

---

[27] The name of a proposal is based on the name of the themeobj embedded in the proposal: a "$" is added to the front of the themeobj name, and another number – "-j" – is added to the end. E.g. the themeobj "$white-1" gets proposed the first time as "$$white-1-1", the second time as "$$white-1-2", etc. Note that this is an extrapolation of the naming of themeobjs – the themeobj "$white-1" got its name through the addition of a "$" to the front of the domain object – "white-1" – from which it was derived. To summarize, "<item>-i" -> "$<item>-i" -> "$$<item>-i-j".

The proposed-rspec-elmt-list (no. 1 wins)

No.	*Proposal*	*Priority*	*Rule*
1	$$introduce-1-1	2.00	$intro
2	$$in-front-of-1-1	.85	$reln-salience
3	$$two-story-building-1-1	.40	$prop-salience
4	$$two-story-building-1-2	.38	$prop-sal-obj
5	$$newcuritem-1-1	.35	$newitem
6	$$white-1-1	.20	$prop-color

This table shows the six rules which fire in the first round of proposing (in column 4) and their respective proposals and priorities (in columns 2 and 3). The priority values are determined by their respective rules, as explained in the text.

Table 3: First r-spec: First round of proposals.

The proposal with the lowest priority is made by "$prop-color" -- its purpose is to specifically mention the *color* of the current-item, based on the observation that people use that property often in their descriptions of visual scenes. This rule says "If the Current-item has a color and the r-spec is not already too heavy, propose saying that color." The precise form of the rule is shown in Figure 17. The "let" statement is explained in Figure 18. In this case a temporary assignment is made to the variables "color" and "cut-off-weight". "Color" is bound to the first property on the Current-properties list that is a color. "Cut-off-weight" is bound to a threshold weight which is calculated to be somewhat smaller (.8) than the product of the global parameters for the optimum r-spec weight and the level of detail (the value .8 is a guess).

The second precondition, "(less-than (weigh-rspec) cut-off-weight)", then determines if the r-spec is already too "heavy" to support an elaboration of the Current-item with its color. The function "(weigh-rspec)" simply sums the weights (see above, page 27) of all of the elements in the r-spec at the time of the proposal. Indeed, as the r-spec describing "House-1" grows it will reach a weight great enough to cause this second precondition to fail.

The first precondition is simply that the object *has* a color in the domain data base at all. In this case both preconditions were met (with "White-1" being the color of "House-1") and the proposal called "$$white-1-1" is made at priority 0.2 -- that is, that potential r-spec element is inserted into the proposed-rspec-elmt-list (see Table 3). As with many of the thresholds, the value of 0.2 is a first guess at a priority value that will be low enough to succeed only when there's little else to say, yet high enough that it does occasionally succeed.

```
($prop-color
   (let (color (get-themeobj-of-type 'color current-props)
     cut-off-weight (times *minimum-rspec-weight
                           *level-of-detail
                           0.8))
      preconditions
         (not (null color))
         (less-than (weigh-rspec) cut-off-weight)
      actions
         (propose
            (create-rspec-elmt 'property
                               color
                               current-item
                               'unique)
          at .2)))
```

This rule proposes to mention the color of the Current-item. The "let" statement creates three local variables for use in the preconditions and actions parts. Specifically, the "cut-off-weight" is a threshold r-spec weight above which this rule will not even bother to make its proposal -- its formula is discussed in the text.

Figure 17: The rule $prop-color.

The "let" statement is a LISP programming technique for making the computation either more efficient or perspicacious. The general form is

```
(LET   (variable-1   expression-1
        variable-2   expression-2
        ...                        )

       <body>        )
```

The parenthesized expression after the "LET" consists of any number of <variable>/<expression> pairs, such that the <expression> is evaluated and the <variable> is assigned (or "bound to" in LISP terminology) that value. These bindings are *temporary* and *local* -- they only hold for the " of the LET, and are lost after the LET statement. In GENARO this is important in preventing accidental interactions between rules.

Figure 18: The LET statement.

The next higher priority proposal is made by the rule "$newitem". This rule runs whenever the r-spec is empty. It says essentially "If the r-spec is empty (i.e. it is a brand new r-spec), and no other better proposals are being made, throw out the Current-item and get the next one from the USOL."[28] This is, in fact, the standard mechanism by which the next object is gotten from the top of the USOL to the Current-item register: in the somewhat sparse domain data bases which have been used to date it usually happens that when the last r-spec has just been sent to MUMBLE (so that the new r-spec is empty), there is nothing more "of interest" (i.e. of high enough priority) to say about the old Current-item. Then $newitem flushes the old Current-item and gets the next one from the USOL. This rule will be discussed further at the end of the construction of the current r-spec about "House-1" (see below).

The proposal of "$prop-sal-obj" is the next highest priority (this rule was described above, page 42). In this case, the Current-item House-1 is counted as a prominent object -- with a salience greater than .9 -- and its most salient property, "two-story-building-1", is proposed by this rule.

This property is also proposed by rule "$prop-salience", at a slightly higher priority. This rule could be paraphrased as saying "If there is a property of the Current-item that is salient enough, propose saying it." The purpose of this rule (which is shown in Figure 19) is to see to it that, if a property of the Current-item is highly salient, it gets mentioned. (Recall that the standard value of *level-of-detail is 1.0.)[29] This rule is different from the previous ones in that it bases the priority of its proposal directly on the salience of the thing being proposed: the more salient the property, the more likely it is to find its way into the r-spec.

The proposal that was next to highest in priority in this first round of proposing was made by "$reln-salience". This rule does for the relations of the Current-item what "$prop-salience" does for its properties: it says "If the most salient relation of the Current-item is salient enough, propose it."[30] The coded form of the

---

[28] This rule embodies the implicit claim that if there is no salient elaboration of an object then we do not describe the object at all, since the only way that an object can be described is by virtue of some properity or relationship.

[29] In fact this rule only looks at the *most salient* of the properties of the current item: if this property is salient enough, it is proposed for mention; the other properties, being of lower salience, are not of interest to this rule. This is yet another form of the basic claim of this thesis: if a property other than the most salient one is to be mentioned it will be by some specific rule looking for some specific *rhetorical* feature.

[30] It is worth noting in passing that the separation, maintained within the input data base and within the rules, between properties and relations makes a rather strong claim that these entities enjoy distinct perceptual *and* linguistic roles. See Chapter 4.1 for a discussion of the source of this distinction.

```
($prop-salience
   (let (priority (times *level-of-detail
                        (get-salience
                            (first current-properties))))
      preconditions
         (greater-than priority .3)
      actions
         (propose
            (create-rspec-elmt 'property
                               (first current-properties)
                               current-item
                               'unique)
          at priority)))
```

This rule is interesting because the *priority* of the proposal is based on the *salience* of the Current-item's most salient property. Note also that while the "priority" appears in the precondition, this is just a short-hand way of checking the property's salience.

Figure 19: The rule $prop-salience.

rule is shown in Figure 20. In this case, the relation "in-front-of-1", between Fence-1 and House-1, is proposed at the priority of .85.

```
($reln-salience
    (let (priority (times *level-of-detail
                        (get-salience
                            (first current-relations))))
      preconditions
          (greater-than priority .3)
      actions
          (propose
             (create-rspec-elmt 'relation
                               (first current-relations)
                               current-item
                               'unique)
           at priority)))
```

In this rule (as with $prop-salience in Figure 19) the priority is calculated at the beginning of the rule, based on the salience of the most salient relation.

Figure 20: The rule $reln-salience.

Finally, at the highest priority, the rule "$intro" proposes to introduce House-1. This rule was discussed in detail above (page 41); its proposal, "$$introduce-1-1", wins and is placed in the r-spec. Thus at the end of the first round of proposing the r-spec looks like this:

```
(RSPEC NO1
   BODY
      (ELMT1 RHETOREME introduce (house-1)
         (house-1 (SUPERC house)
                  NEWITEM))
   OPTIONAL nil)
```

The qualifier "NEWITEM", incidentally, is added automatically to r-spec elements when the object being qualified was marked as unmentioned. Once that item is inserted into the r-spec, however, it gets marked as mentioned. This information is maintained in the item's *themeobj*. In fact, the main reason for having the level of representation of themeobjs is so that such facts as whether or not an item has been mentioned can be kept around, without having to modify the domain data base. In this case, the themeobj $house-1 gets marked as mentioned – specifically, as mentioned in the *first* r-spec.

If this r-spec were sent to MUMBLE as it is, it would be realized as something like "This is a picture of a house". The program judges this to be too little semantic content, based on the weight of the r-spec (that is, the weight of the single rhetoreme element, which is 2.0 – see page 27 for details), and so the predicate "rspec-complete?" fails, leading to another round of proposing.

At this point it may seem to the reader computationally inefficient that a whole round of proposing produces only one r-spec element. Many of the proposals made on this round are likely to made again on the next round, by the same rules. Nevertheless, because the r-spec itself is one of the data structures available to the rules for both examination and modification, this repetition is necessary to assure that rules are always basing their proposals on the most current information. Besides, if the rules are thought of as independent machines operating in parallel,[31] such extravagance is a natural and desirable part of the control stucture (even if it is slower on a serial machine).

As in the last round, the first thing to happen is that the Proposed-rspec-elmts-list is cleared. At the end of proposing only four of the six rules from the first round have fired again, and no new rules have fired (see Table 4).

---

[31] Note that while r-spec construction is inherently linear and serial, deciding on the next element for the r-spec can be done in parallel. Note also that generating from a *dynamic* data base would make the parallel aspect even more important, since selection deliberations would need to proceed quickly and without data base changes *during* the element selection process.

$Intro does not fire this time because its r-spec element is marked as "unique", causing the control structure to prevent the rule from firing again during this r-spec. The other rule missing in this round, $newitem, required an empty r-spec as one of its preconditions -- this is clearly no longer the case. The other four rules from the last round all fire again on identical grounds, so last time's "runner-up", $reln-salience, wins on this round. Its r-spec element is:

```
(ELMT2 RELATION in-front-of-1 (fence-1 house-1)
    (fence-1 (SUPERC fence)
             NEWITEM)
    (house-1 IBID))
```

and the whole r-spec would be realised as "This is a picture of a house with a fence in front of it.", but the weight of this r-spec (3.0) is still too small to send to MUMBLE. This assessment is made in a number of ways (discussed below), but all of them use a global parameter called "*minimum-rspec-weight", which specifies the minimum weight allowed for an r-spec, and the value of which is 3.8 for the example in this chapter.

In the third round of proposing nothing has changed except that $reln-salience is no longer making its proposal. Recall that once a themeobj is inserted into the r-spec it is marked as mentioned, and that this blocks that themeobj being proposed by any rule. This opens the way for $prop-salience's proposal, "$$two-story-building-1-5" at priority .40, to (finally) be the highest priority proposal when the dust settles on this third round of proposing. The r-spec element which gets added is straightforward, and the whole r-spec is shown in Figure 21. MUMBLE would realize this r-spec as "This is a picture of a two story house with a fence in front of it.", and in fact this *is* the r-spec that is sent to MUMBLE in this example. This is *not* because this r-spec has gone over the *minimum-rspec-weight, but because there is "nothing left to say".

The proposed-rspec-elmt-list (no. 1 wins)

No.	*Proposal*	*Priority*	*Rule*
1	$$in-front-of-1-2	.85	$reln-salience
2	$$two-story-building-1-3	.40	$prop-salience
3	$$two-story-building-1-4	.38	$prop-sal-obj
4	$$white-1-2	.20	$prop-color

The four rules which fire in the second round of proposing. (Note that the names of the proposals indicate how many times their themeobj has been proposed, e.g. $in-front-of-1 is being proposed for the second time in "$$in-front-of-1-2".)

Table 4: The first r-spec: The second round of proposals.

```
(RSPEC NO1
   BODY
      (ELMT1 RHETOREME introduce (house-1)
         (house-1 (SUPERC house)
                  NEWITEM))
      (ELMT2 RELATION in-front-of-1 (fence-1 house-1)
         (fence-1 (SUPERC fence)
                  NEWITEM)
         (house-1 IBID))
      (ELMT3 PROPERTY two-story-building-1 (house-1)
         (house-1 IBID))
   OPTIONAL nil)
```

Figure 21: The first complete r-spec.

To understand this, recall that the proposing process continues until the predicate "rspec-complete?" is true.

```
BUILD-RSPEC
      (repeat
               (initialize-for-proposing)
               (propose-rspec-elmts)
               (setq rspec-elmt (pick-best-proposal))
               (insert-in-rspec rspec-elmt)
       until (rspec-complete?))
```

One of the conditions of this predicate is that the Proposed-rspec-elmts-list is empty. That is, at the end of proposing, no rule had made a proposal, and this is what happens in this case. Of the three rules which ran in the last round, $prop-salience, $prop-sal-obj, and $prop-color, the first two are blocked because their proposed r-spec element, two-story-building-1, is already in the r-spec. $prop-color, on the other hand, has withdrawn its proposal to mention the color of House-1 (i.e. White-1) because its precondition on the weight of the r-spec has failed.

Specifically, the weight of this three element r-spec is 3.5, but the "cut-off-weight" threshold computed by $prop-color is only 3.04. This cut-off-weight is the product of three factors: the *minimum-rspec-weight (which is 3.8), "0.8", and the *level-of-detail (whose value is 1.0). The first two factors reflect my desire to have this rule only propose the color of the Current-item if the r-spec was less than 80% of its optimum weight. The third factor, *level-of-detail, is a global parameter which has a fairly straightforward meaning -- if a more or less detailed description is desired this parameter can be adjusted accordingly. In this case, if *level-of-detail had been slightly higher (i.e. 1.25 or greater), the cut-off-weight computed by $prop-color would have been *above* the r-spec weight. Thus the rule would have fired, proposing

"White-1" at priority .20, and this time it would have won since there would have been no competition from other rules. In this case the final r-spec would be realised as "This is a picture of a white, two story house with a fence in front of it."[32]

3.4.2 The Second R-spec.

Once the r-spec is sent to MUMBLE, GENARO (specifically the function "end-description?") checks whether this is the end of the description. This depends chiefly on the next object on the USOL -- if its salience is below a threshold the Conclude packet is turned on. In this example the next item on the USOL is Door-1, whose salience of .8 is well above the threshold, so the function "build-rspec" is called again.

The Proposed-rspec-elmts-list and R-spec are cleared and another round of proposing occurs. However, Current-item is still House-1 -- nothing has occured to change it, and in fact it is certainly possible that there was still a great deal to say about it. In this case, for example, $prop-color proposes once again (still at a priority of 0.2) that the color of the house be mentioned. But since the r-spec is empty $newitem also makes its proposal -- at a priority of .35. Recall that $newitem also ran at the very first round of proposing for the first r-spec; at that time, however, there were 4 other rules that had proposals with higher priorities.

$Newitem is one of the rules whose action is not to insert an element into the r-spec, but rather is a function which gets evaluated, performing an action. The action, in English, is "If the object that will be the next Current-item is salient enough, pop it from the USOL into Current-item, otherwise signal the end of the description by clearing the input registers". The actual rule is shown in Figure 22. In this case (next-curitem-salient-enough?) is true, so the proposed action -- which is also executed -- is to pop the next object on the USOL, which is Door-1, into Current-item.[33] This is the standard procedure for getting a new Current-item. It provides the flexibility of allowing several r-specs to be built about the same Current-item if there are enough salient things to say. In all of the runs studied so far, however, after each r-spec is sent to MUMBLE one round of proposing is spent with $newitem getting the next Current-item.

---

[32] See Chapter 6 for a way of using a *rhetorical rule* to signal the completion of building an r-spec.

[33] Note that the object Fence-1, which was above Door-1 on the USOL, was mentioned in the last R-spec, and so is no longer "unmentioned". This does not mean that nothing more can be said about Fence-1, but it does pose problems -- see the discussion in hypothesis I.b in Chapter 5.

```
($newitem
    preconditions
       (null r-spec)
    actions
       (propose
            (build-function
                 (if    (next-curitem-salient-enough?)
                  then (set-curitem (pop-usol))
                  else (setq usol       nil
                             cur-item nil)))
        at .35))
```

One of the rhetorical rules whose proposal is an action, not an r-spec element. Note the distinction between the *propose* action, which is executed (if the precondition succeeds) when the rule is run, and the *constructed* action, which is built by (build-function) but which is only executed if the proposal succeeds.

Figure 22: The rule $newitem.

Figure 23 shows the new Current-item, Door-1. All of the rules which make proposals in the first round of proposing with this new Current-item have already been explained. When the proposing is over, the Proposed-rspec-elmts-list contains four proposals, as shown Table 5.

Briefly, $prop-color is proposing mentioning that the door is red on the basis that (a) the door has a color and (b) the r-spec is still quite "light". $Newitem's proposal is based on the R-spec being empty at the moment. $Prop-salience is proposing the color of the door because this property is quite salient: in the data base the salience of property Red-1 is .80. Finally, $reln-salience has the highest priority proposal, which is to mention that Door-1 is a part of House-1, and this rhetorical element is duly inserted into the r-spec.

In the next round the proposals are as shown in Table 6; the two "red" proposals from the last round persist this time, and a new rule, $condense_prop, has made a proposal at a very low priority. With the winner of the second round of

Current-item: Door-1
Current-properties: (red-1)
Current-relations: (part-of-2) {House-1}

Figure 23: The second Current-item.

The proposed-rspec-elmt-list (no. 1 wins)

No.	*Proposal*	*Priority*	*Rule*
1	$$part-of-2-1	0.81	$reln-salience
2	$$red-1-1	0.80	$prop-salience
3	$$newcuritem-3-1	0.35	$newitem
4	$$red-1-2	0.20	$prop-color

This table shows the four rules which fire in the first round of proposing for the Current-item Door-1.

Table 5: The second r-spec: The first round of proposals.

The proposed-rspec-elmt-list (no. 1 wins)

No.	*Proposal*	*Priority*	*Rule*
1	$$red-1-3	0.80	$prop-salience
2	$$red-1-4	0.20	$prop-color
3	$$condense-prop-1-1	0.05	$condense-prop

The three proposals made in the second round of building the r-spec for Door-1.

Table 6: Second r-spec: The second round of proposals.

proposing inserted, the r-spec is as shown in Figure 24. If this r-spec were sent to MUMBLE it would be expressed as "The house has a red door".[34] However, GENARO has more to say.

In the next round the only proposal is by $condense-prop. This rule spots objects of lower salience than the Current-item (objects on the USOL) which share with the Current-item a property, and condenses the description of such objects into the description of the Current-item. Underlying this rule is a claim that two objects in a picture which have the same property, e.g. that they are both red or broken or in the foreground, can be described more concisely (following Grice's dictum for economy of expression) by being described together. I have termed such items "rhetorically parallel", because the shared perceptual element between two items invites specialized treatment at the rhetorical planning level. Specifically, $condense-prop says "Look down the USOL for a 'parallel object' (one which shares a *property* with the

[34] Or as one of the transformational variants of this sentence. See Chapter 4.

```
(RSPEC NO2
   BODY
      (ELMT1 RELATION part-of-2 (door-1 house-1)
            (door-1 (SUPERC door)
                  NEWITEM)
            (house-1 IBID))
      (ELMT2 PROPERTY red-1 (door-1)
            (door-1 IBID))
   OPTIONAL nil)
```

The r-spec for Current-item Door-1 after two rounds of proposing.

Figure 24: The first part of the second r-spec.

Current-item in this case)[35] ; if the parallel is strong enough (see below) and the r-spec is not already too heavy, then construct an action whose effect (if executed) is (1) to remove the parallel object from the USOL and make it the Current-item and (2) to put a 'condensation marker' in the r-spec to signal what has happened to MUMBLE." The precise form of the rule is shown in Figure 25. In this example, the Current-item Door-1 is red, and this rule detects that Gate-1 (the gate of the fence), which happens to be the object on the top of the USOL, is also red. Since this is the sole proposal for this round, Gate-1 is made the new Current-item.

The first thing that $condense-prop does (see Figure 25) is to do a search down the USOL, starting at the top and checking each object to see if it and the Current-item have any properties which have the same superconcept, e.g. Red-1 and Red-2 both have the superconcept Red. If such an object is found, certain facts about the object, such as its name, its position on the USOL, and the kind of similarity (e.g. "property: red"), are packed into "parallel-object". This temporary variable is then referenced by the preconditions and actions of the rule.

There are three preconditions:

1. that there is such an object;

---

[35] Note that the first parallel object found will also be the most salient such object on the USOL.

```
($condense-prop
   (let (parallel-object
            (find-parallel-object 'property))
      preconditions
         (not (null parallel-object))
         (parallel-enough?  parallel-object)
         (less-than
            (weigh-rspec)
            (times 0.7 *minimum-rspec-weight))
      actions
        (let (temp
                 (create-rspec-elmt 'rhetoreme
                                    'condense-prop
                                    '(Current-item
                                    parallel-object)
                                    'unique))
          (propose
             (build-function
                (set-curitem-to-parallel-object
                           parallel-object)
                (add-to-rspec temp))
           at .05))))
```

The rhetorical rule $condense-prop. The first "let" gets a vector of descriptors describing the first USOL object which is rhetorically parallel to Current-item and puts that vector in "parallel-object". The first two preconditions make sure that there is such an object and that it is salient enough. The action does two things: (a) builds an r-spec element describing the proposed condensation, and (b) builds a proposed action which pulls the parallel object out of the USOL and makes it the Current-item and then adds the r-spec element constructed in (a) to the r-spec. Full details are in the text.

Figure 25: The rule $condense-prop.

---

2. that it is parallel enough – this involves: (a) it not being too deep into the USOL, and (b) both properties (of the Current-item and the parallel object)

being salient enough[36] ; and

3. that the r-spec is not too heavy to accomodate a condensation -- since describing a second object in the same r-spec can lead to an excessively heavy r-spec, condensation is not started if the r-spec is already moderately heavy.

If all three of these preconditions are true, a proposal is constructed in the following manner:

1. an r-spec element is created which signals that a condensation has occured in the r-spec and points to the parallel elements; this element is set aside in a temporary variable called "temp";

2. the first action of the proposal is the removal of the parallel object from the USOL and the setting of the Current-item to this object; and

3. the second action is the insertion into the r-spec of the element constructed in step (1) and set aside in "temp".

(Note that the r-spec is *not* sent on to MUMBLE as part of this process.)

As stated above, in this example $condense-prop does indeed find a parallel object, and in fact wins with its proposal, making the Current-item as shown in Figure 26 and adding to the r-spec a rhetorical element of type "rhetoreme":

```
(ELMT3 RHETOREME condense-prop (door-1 gate-1)
    (door-1 IBID)
    (gate-1 (SUPERC gate)
            NEWITEM)))
```

Current-item: Gate-1
Current-properties: (Red-2)
Current-relations: (Part-of-3) {Fence-1}

Figure 26: The third Current-item.

---

[36] Note that this condition blocks condensing on just any two objects which share some property -- certainly one does not want to condense the description of a green house with the green trees around it. But in such cases the salience of at least one of the properties will be below threshold -- green is a very low salience property of a tree.

The major functions of this element are (a) to have a specific marker in the r-spec which indicates to MUMBLE that a condensation has taken place, and (b) to point to the pair of condensed objects. MUMBLE has an entry for "condense-prop" (as discussed in Chapter 5), and this entry specifies the linguistic constructions (e.g. conjunction, ellipses, verb phrase reduction) which can be used to realize the condensation.

In the next round of proposing the rules are working with a new Current-item, Gate-1. This object has the shared attribute (i.e. red) and a single relation (i.e. part-of). When both the property and the spatial relation of Gate-1 are inserted into the r-spec there is nothing left to say, and the r-spec, as shown in Figure 27, is sent on to MUMBLE. There are several ways that such an r-spec could be realized; my personal preference is "The house has a red door and the fence has a red gate", though one could also say "Both the door of the house and the gate of the fence are red". The dictionary entry for condense-prop would contain a specification for the structure of both of these constructions, as well as any others that appealed to the author of the dictionary.

```
(RSPEC NO2
   BODY
      (ELMT1 RELATION part-of-2 (door-1 house-1)
             (door-1 (SUPERC door)
                     NEWITEM)
             (house-1 IBID))
      (ELMT2 PROPERTY red-1 (door-1)
             (door-1 IBID))
      (ELMT3 RHETOREME condense-prop (door-1 gate-1)
             (door-1 IBID)
             (gate-1 (SUPERC gate)
                     NEWITEM)))
      (ELMT4 RELATION part-of-2 (gate-1 fence-1)
             (gate-1 IBID)
             (fence-1 IBID))
      (ELMT5 PROPERTY red-2 (gate-1)
             (gate-1 IBID))
   OPTIONAL nil)
```

The r-spec based on two Current-items, Door-1 and Gate-1, condensed into one r-spec by $condense-prop.

Figure 27: The second r-spec.

But suppose that Gate-1 had not had Part-of-2 (with Fence-1) as its most salient relation, but rather, say, that there was a person standing next to the gate, i.e. Next-to(Gate-1, Person-1). Then Next-to would have been in the r-spec in place of Part-of, and the resulting sentence would have been (at best) something like "The house has a red door, and there is a red gate with a person next to it." While this is certainly acceptable, it is less compact and more awkward than the more parallel constructions above. This demonstrates that there are degrees of rhetorical parallelism, and that the criteria for condensation are probably more complex than are captured in $condense-prop as it is shown.

The operation of this rule also illustrates a potential problem that stems from the weakness of GENARO's control machinery -- once an object has been the Current-item and has been replaced in that role, there is no way for it to be described any further, *except* in relation to other objects which have become the Current-item. Thus it would be possible to design the system's rules in a way that prematurely threw away a Current-item, i.e. while there still was more salient material to mention. The solution is to specify rule parameters in such a way that rules which might replace the Current-item do not run until those doing elaboration of the Current-item are done -- in GENARO this is done by having the priorites of rules in the Elaborate packet be generally higher than those in the Shift-topic packet. Also the global parameter *shift-factor can be used to "turn down" (or up) the Shift-topic rules as a group.

3.4.3 The Third R-spec.

At the beginning of the third r-spec the r-spec register is cleared, the packet driver is checked (but no packets are switched on or off), and, with the Current-item still Gate-1, the first round of proposing yields only the proposal from $newitem to get the next object from the USOL. This is Driveway-1, and Figure 28 shows what it looks like once inserted into the Current-item.

Current-item: Driveway-1
Current-properties: nil
Current-relations: (next-to-3 {House-1}
next-to-6) {Bush-1}

The Current-item Driveway-1; note that it has no properties in the domain data base.

Figure 28: The fourth Current-item.

The first round of proposing in the construction of this r-spec yields two proposals, as shown in Table 7, the best of which is to mention the relation between Driveway-1 and House-1.

In the next round no proposals are made. This is because there is not much about Driveway-1 in the data base (i.e. it has no properties), and also because there is nothing in the remaining USOL with which it can be condensed. The resulting r-spec contains only one element (see Figure 29), and would be realized simply as "There is a driveway next to the house.", or "Next to the house is a driveway."

It may strike the reader that this sentence states a fact that is so ordinary, or predictable, that it would not normally be included in a description. However, whether or not this is the case is left, in this system, as a *perceptual* issue -- the more ordinary a fact is, the lower its salience, on the same grounds that its salience is *higher* the more *unexpected* it is. Thus a vision system whose world knowledge indicated that houses *always* had driveways next to them would have accorded the driveway very little salience in its perception of this scene, and GENARO would have accordingly been very unlikely to mention it.

Another issue brought up by this r-spec is its small size. Recall that this system assumes a one-to-one correspondence between r-specs and sentences: MUMBLE must produce exactly one sentence for each r-spec sent it. Furthermore, there is a

The proposed-rspec-elmt-list (no. 1 wins)

No.	*Proposal*	*Priority*	*Rule*
1	$$next-to-3-1	0.72	$reln-salience
2	$$newcuritem-5-1	0.35	$newitem

Table 7: Third r-spec: First round of proposals.

```
(RSPEC NO3
   BODY
      (ELMT1 RELATION next-to-3 (driveway-1 house-1)
            (driveway-1 (SUPERC driveway)
                        NEWITEM)
            (house-1 IBID))
   OPTIONAL nil)
```

Figure 29: The third r-spec.

correspondence between the weight of an r-spec and the complexity of the sentence resulting from it, i.e. light r-specs cannot be realized as complex sentences. This is a strong claim, but one that, after much experimentation with the system, there was little reason not to make. Certainly it is easier to consider the interactions between the rhetorical rules in GENARO and the dictionary entries in MUMBLE having made this assumption. And there is no compelling reason as yet to complicate the r-spec/sentence correspondence.

Although this r-spec ("NO3") is very "light", and its realization is, accordingly, quite brief, such sentences were quite common in the paragraphs written by subjects in the experimental studies, and seem to indicate places where there is a fairly isolated piece of nonetheless salient information. The style of people's descriptions, and of the texts generated by this system, thus reflect the "style" of the information being communicated.

However, the subjects also seemed willing to conjoin two completely unrelated clauses, as in "Next to the house is a driveway, and in the foreground of the picture is a mailbox". This kind of construction leads to an interesting question: Where in the planning process did the decision to conjoin these two unrelated objects happen? Did the subjects plan it from the beginning, or did they plan each clause separately and decide (during realization) to conjoin them because they were so short? Or, as a third possibility, did they plan the first clause and then, seeing that it was short, decide to talk about the second item without closing the sentence? That is, is the conjunction decision made before the planning of the first clause, after the planning of both clauses, or between clauses?

The proper psycholinguistic study could probably answer this question, but in this case GENARO makes some claims. The first proposal requires that the planner look ahead and decide to conjoin based on a guess that it will not have much to say about either object[37] , and thus that the surface size of the descriptions of both objects will be small. The second proposal asks nothing of GENARO and leaves the decision to conjoin for MUMBLE to make; although MUMBLE is capable of making this decision, it, like GENARO, will have to base its decision on an estimate of the eventual size of the second clause. However, this scheme violates the one sentence per r-spec convention adopted for this study, since it would take two r-specs to make the conjoined sentence. And although it would be fruitful to investigate the reasons for making this decision in GENARO versus in MUMBLE, this line of questioning will not be pursued here.

---

[37] Such a "look ahead" is legal for GENARO to make. A rule could be made that made this proposal just based on a few cheaply observable parameters of the Current-item and the object on top of the USOL plus a guess that there would not be much to say about either of these objects. However, if its "guess" were thorough enough to spot, say, a condensation that would happen on the USOL object, then it would be doing real look ahead, and this would be illegal.

The third proposal also requires GENARO to guess, but only about the size of the description of the second object. Since the decision to package the second element (about the mailbox) with the first element (about the driveway) must be made before computing the second element, this decision would be partially blind, and would be wrong sometimes. For example, with suitable modifications (see Chapter 6) GENARO could easily run $newitem (thus getting a new Current-item object from the USOL) without sending the current r-spec and clearing that register -- in this case making Mailbox-1 the Current-item. (In so doing it could also insert some kind of marker into the r-spec indicating to MUMBLE that this had happened.) It could then go on to describe the mailbox in the same r-spec with the driveway, and MUMBLE would be able to produce from this compound r-spec the kind of conjoined sentence at the beginning of this paragraph. Without using lookahead, however, GENARO runs the risk of finding itself with a full r-spec *and* still many salient things to say about the new (the second) Current-item (see the discussion of Hypothesis I.a in Chapter 5). There are three options at this point: 1) plan a large r-spec that is likely to result in a long run-on sentence; 2) stop planning (and saying?) the current sentence and replan the second conjunct with some of its material transferred into a subsequent r-spec; or 3) arbitrarily drop the over-loading material from the r-spec (and do not say it). This issue is discussed more fully in Chapter 5.

3.4.4 The Last R-spec.

The last r-spec in this example description leads to the sentence "It is a cloudy day.", yet the USOL contains no object "day", "clouds", or even "sky". Where does this r-spec come from?

The answer starts with my observation that subjects often ended their description of a picture with some kind of global comment on the whole scene, as a way of "wrapping it up". Some typical comments were "The landscaping is nice.", "Its a cloudy day in winter.", or "This is not an interesting picture.". The job of the rules in the Conclude packet is to generate the r-specs underlying such comments. But the trigger for drawing the description to an end is salience: when the new Current-item is below a certain salience value, the Conclude packet is turned on, some concluding remark is (or remarks are) made, and the whole system stops.

In this case, the next item on the USOL, Mailbox-1,[38] has a salience below the threshold. This decision is made by the action of the rule $newitem: recall that $newitem (Figure 22) uses the predicate "Next-curitem-salient-enough?" to decide between popping the next USOL item into the Current-item and signalling the end of the description. This function, shown in Figure 30, simply compares the salience of

[38] Actually the next two items on the USOL, Mailbox-1 and Porch-1, both have the same salience (0.5). In cases like this the program makes an essentially random choice between the equal-salience items.

```
(defun next-curitem-salient-enough? nil
    (greater-than (get-salience (top usol))
                  (times *theta *level-of-detail)))
```

The function that determines when to end the description. "*Theta" is a global variable for the minimum salience threshold used by this rule. "*Level-of-detail" is factored into this function so that when a higher level of detail is specified (by increasing this global parameter) the program includes less salient objects in the description, as well as saying more detailed (i.e. less salient) things about each object.

Figure 30: The function Next-curitem-salient-enough?.

---

the item on the top of the USOL with the threshold for minimum object salience ("*theta").

Since Mailbox-1 is below this threshold the action proposed by $newitem is to wipe the Current-item clear. This action is equivalent to setting an "end-the-description" flag, but it has the added advantage of blocking proposals which are extraneous to the Conclude packet. The Paragraph-driver detects that the Current-item is empty and responds by turning on the Conclude packet. To date this packet contains only one rule: "$light". The purpose of this rule is to comment on the illumination in the picture. The rule looks into the USOL specifically for "house-scene-1", which is the high-level frame in which all of the picture-level information is kept. Specifically, this concept has properties ("slots" in frames terminology) for "sky cover", "time of day" (e.g. day, twilight, or night), and "season of the year"[39] , and these are available to the Conclude rules via the Current-properties subregister. On the basis of this information $light offers the r-spec shown in Figure 31. MUMBLE's dictionary has an entry for "cloudy-day" which specifies the clause "It is ...".

The paragraph which is typed out by MUMBLE,

---

[39] In Chapters 4 and 5 I argue that these roles are naturally computed by a computer vision system in the process of understanding what is in the image – that claim will be assumed here. Also, in Chapter 6 there is discussion of the possibility of having the "properties" Sky-cover etc. be concepts which appear on the USOL as "objects" in their own right, thus assuring that they get mentioned in the paragraph in accordance with their salience.

```
(RSPEC NO4
   BODY
      (ELMT1 PROPERTY cloudy-day-1 (house-scene-1)
            (house-scene-1 (SUPERC outdoor-scene)
                           newitem)
   OPTIONAL nil)
```

Figure 31: The final r-spec in the example.

> This is a picture of a two story house with a fence in front of it. The house has a red door and the fence has a red gate. Next to the house is a driveway. It is a cloudy day.

is a very short example of the style and content of text produced by this system, but has served here to illustrate the operation of GENARO and some of the rhetorical rules. Since this discussion focused on the operation of GENARO, which is only one part of the larger system, many important issues about the computational context have not been addressed. In the next chapter both the content of the VISIONS data base and the operation of MUMBLE will be taken up. In the chapter after that the strengths and limitations of this system will be discussed.

3.5 Summary

In this chapter I have attempted to explain the operation of GENARO as a model of deep generation without getting too deeply into the many theoretical and controversial issues which the design of this very simple program raises. These will be thoroughly discussed in Chapter 5.

To summarize the important points about GENARO, then:

- The machinery of this program is very *weak* considering the intuitive difficulty and complexity of rhetorical planning. The restrictions on the preconditions and actions available to the rhetorical rules, plus the lack of any backtracking or lookahead facility, makes this an extremely *localized* (and even "myopic") planning device.

- The rhetorical rules represent an empirical theory about the rhetorical conventions required to generate scene descriptions. (These conventions, while probably *culturally* dependent, are not specific to English, and may in fact be linguistic universals.)

- The interface to MUMBLE, and the combination of these two indelible programs, makes a series of psycholinguistic claims which can be verified or disproved empirically. For example, GENARO can overflow an r-spec and be forced to dispatch it to MUMBLE before it is complete, and r-specs can be formed by GENARO which cannot be realized by MUMBLE.

- Because of its simplicity, GENARO is a very *fast* rhetorical planner.[40] Nonetheless, because of its reliance on perceptual *salience*, the quality of its output is high.

[40] A description such as the one presented in this chapter took between 6 and 70 seconds of CPU time to plan (on a VAX 11/780), depending on whether certain program options were turned on. Compiling the code, especially the KL-ONE interpreter, would result in a significant speed increase. See Appendix 2 for more details.

CHAPTER IV

THE SETTING FOR THIS PLANNER

One view of the A.I. program GENARO presented here is that it links two other A.I. programs: one that "knows about" the visual world and one that "knows about" English language. This chapter presents these two other programs, SALIENCE and MUMBLE, and discusses some of the issues that arose in designing and debugging the respective interfaces.

The SALIENCE system is a hypothetical computer vision system (patterned after the VISIONS system, [Parma 1980, Hanson 1978]) capable of constructing a complete three dimensional model of the contents of a natural scene. In this thesis I focus on the three dimensional model itself, and not the vision system which constructs it, although enough attention is paid to the system to justify the claims made about its representation. It is this representation which acts as the input to GENARO.

At the output end, the operation and data structures of the realization component, MUMBLE, will be presented, with emphasis on the issues that arise in the division of labor between deep and surface generation.

4.1 The input perceptual representation

Years ago, when I was looking for a Master's Project topic, I hit upon the idea of building a natural language interface for the computer vision system under development here at the University of Massachusetts (called VISIONS, see [Parma, 1980]). It soon became clear that building both a parser and a generator that were useful was an unrealistic project. However, in the process of examining the problem of giving a voice to the VISIONS system, I discovered that a perceptual representation offers several advantages to the study of language. By it nature, a perceptual representation is a *non-linguistic* structure which describes the objects in the image and their attributes and relations. By being non-linguistic the representation prevents cheating (for example, precompiling linguistic decisions into the perceptual representation). Additionally, since the job of a computer vision system is to map color images into 3-D representations of the parts of the image, there is a natural motivation for and check on the semantics of sentences generated from these representations (i.e. Is the person really "in front of the house" spatially, as the description says?). Finally, the domain of pictures of suburban house scenes, which has been the domain of input images for the VISIONS system, is quite suitable for gathering empirical data on the correspondence between natural scenes and people's textual descriptions of them.

The problem with using VISIONS, or *any* current computer vision system, is that the state of the art simply is not yet to the point of building, from raw visual data of an actual outdoors color photograph, a complete and consistent three-dimensional model of what is in the scene. Yet it was just such a representation that was needed in order to study the generation of natural language texts from a non-linguistic data base. I could have solved this by simply building the representation by hand without being concerned with the machinery which produced it -- but this would mean that *all* elements of the representation were essentially unmotivated and arbitrary. On the other hand, I could have decided to wait until the computer vision problem was solved and a working system built: the representation it used would certainly be well-motivated! The compromise I settled upon was to "design" my own "computer vision system", one which did do a full analysis of a complex picture, if only on paper. This system, which I have named "SALIENCE", is described below. Its design is based on what I understand the philosophy of the builders of the VISIONS system to be, based on years of interaction with this group.[1] The only part of the SALIENCE system that has actually been coded into the computer is a hand-built mockup of the final representation that one might expect from a fully functional visual analysis. To make this clear, the following section provides a brief explanation of how the SALIENCE system is designed. Following that, the hand-built representation will be presented, including a discussion of where the salience annotation in the representation came from.

4.1.1 How the SALIENCE system works.

Broadly speaking, the process of perception can be viewed as a process of building an internal model of some external world based on sensory data from that world and generic knowledge about it. In visual perception the problem is to

> "interpret two-dimensional ... images of complex scenes, such as house and road scenes. The interpretation process involves constructing a set of consistent models where each model contains an object-labeling of regions in the image and their location in three-dimensional space. Construction of these types of models is critically dependent on an ability to interpret typically imperfect information with the context of domain- and world-knowledge, goals, and current assumptions." (Wesley and Hanson [1982] describing the goal of the VISIONS system.)

It is difficult to appreciate the vast amount of subtle analysis that goes into reducing a visual image consisting of millions of points of light into a model which specifies a three dimensional reconstruction of the scene represented in the image. The approach

---

[1] In addition to several classes on the VISIONS system, I served as a Research Assistant with that group early in my graduate student career. My association with them has continued to the present, including many long discussions on the hard problems in mechanical scene analysis.

of the SALIENCE system, following the original VISIONS system design, is to combine bottom-up analysis of the raw visual image with top-down frame-based hypotheses about the subject matter of the scene. The bottom-up analysis is akin to the preprocessing that occurs in animal vision systems: converting a huge number of minute data points into a much smaller number of edges and regions. Further processing merges the edge and region information, as well as merging multiple edges into single edges, and multiple regions into single regions.

But such low-level processing is "blind" -- there are many places in pictures of actual scenes where the human visual system "fills in" missing, noisy, and erroneous low-level data. Such filling in requires knowledge of what is *supposed* to be there in the image, and for this people have, and computer vision systems need, a wealth of knowledge about objects in the world, the range of sizes, shapes, and orientations that they can be found in, the kinds of colors and textures they take on, and their normal relations with other objects in the world. What is more, this information must be accessible in a way that allows the bottom-up processes to ask "Here's a possibly related collection of regions and edges -- what object might they represent?"

The design of the SALIENCE system is that this high-level knowledge about the world is organized into a hierarchy of 3-D frames, each of which contains information about some object (or collection of objects) in the "real world", and which specifies the obligatory, optional, and illegal "fillers" which that frame may have on its "slots". For example, once the system has enough edge and region data to suggest that there may be a house in the image it will apply its generic knowledge about what parts houses typically have and how they are typically spatially arrayed to attempt to impose identifications on regions which would otherwise be ambiguous.

Along with each schema being considered the system maintains a "confidence value" in that schema -- the more slots which are filled in a schema X, and the higher the confidence values in each of those filler schemas, the higher the confidence value in schema X. An image is considered to be "seen and understood" when there is a scene-level frame (i.e. House-scene), or a collection of them, which has a confidence value that is some threshold amount *greater than* any other schema's confidence value.[2]

---

[2] There is a difference between human and computer vision in the following respect: people seem to have only one interpretation of the data at a time, i.e. they tend to "snap" onto a high-level interpretation of a scene that is exclusive of all others (viz. the Necker Cube phenomenon), whereas a computer can easily have two high-level schemas with equal confidence values, and without specific machinery for forcing the choice the program will remain in an interpretation "limbo".

4.1.2 The simulated perceptual representation.

Ultimately the goal of computer vision analysis is to have an internally consistent assemblage of schemas[3] such that each point in the image is accounted for by some frame, and each frame has for all of its obligatory slots either a filler or some "explanation" for the absence of the filler.

A local implementation [Woolf, 1981] of the knowledge representation language KL-ONE [Brachman, 1979] was used for these simulated data bases. The network for the picture in Figure 1 (page 4) is presented in Figure 32, and a detail from that network is shown here in Figure 33.

KL-ONE is a highly structured language in which to express semantic networks. It features a very limited set of inter-node *arc* types (such as "Superconcept" and "role") which can serve only a syntactic role in the network -- arcs cannot carry domain information (e.g. "is-in-front-of" arcs are disallowed). *Nodes* can be of two main types: concepts and roles. All of the domain information is carried in these concept and role nodes, and in the structure of their interconnection.

Several points can be made concerning the representation and its underlying ontology.

- The basic unit of the KL-ONE representation language is the *concept* -- in this application concepts function to represent *objects*, *properties*, *relations*, and *gestalts* in the domain (i.e. in the "world"); each of these will be elaborated below.

- A concept can either be "*generic*", representing a general description of an item, or "*individuated*", representing a specific item in the world. "Superconcept" links are used to join these two levels, as well as to describe the vertical conceptual hierarchy within each level. For example, the superconcept of "House-1" in Figure 33 is "House".

- "Horizontal" relationships between concepts are captured using "roles": these specialized nodes attach to a single concept and are used to describe that concept, much as "slots" are part of a frame and serve to describe that frame. For example, the "Agent" role in the figure belongs to the "In-front-of-1" concept, and expresses its relationship to the concept "Fence-1". In the local dialect of KL-ONE the roles attached to object concepts have a further differentiation into "attribute" or "subpart" roles: the former links the object to concepts functioning as properties of that object, while the latter links the object to objects which are subparts of it (usually structurally).

---

[3] For a discussion of the notion of *frame systems* and *schema assemblages* see Arbib [1977].

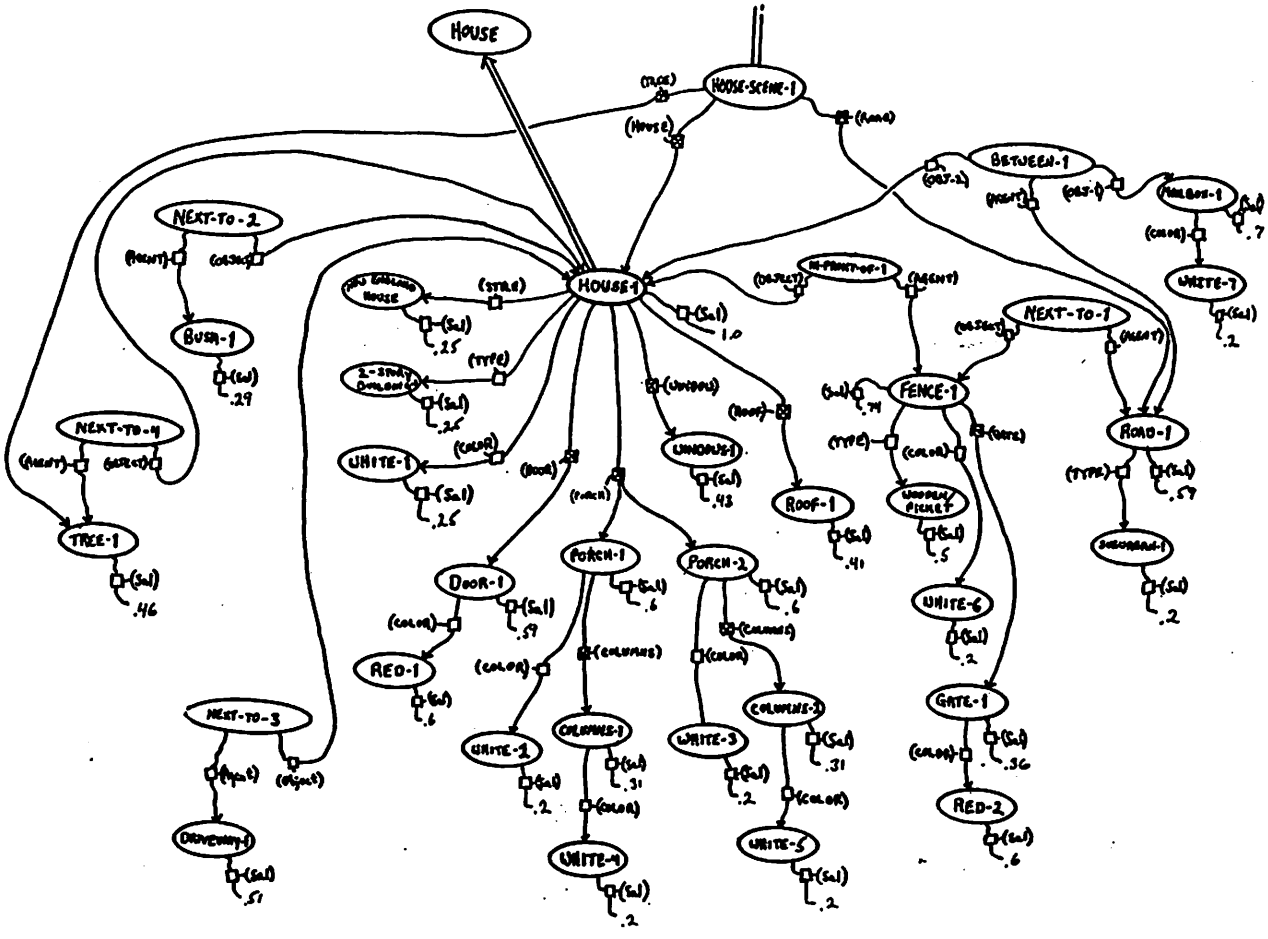


Figure 32: KL-ONE representation of the winter house scene.

The KL-ONE network representing the "perception" of the SALIENCE program, of the winter house scene (page 4). This figure actually shows only the instantiated concept nodes (see below) -- the full (implemented) network, including the generic level concepts, contains 73 concept nodes. The salience values shown represent the averages of the empirically derived values (see [Conklin, Ehrlich, and McDonald 1983]).

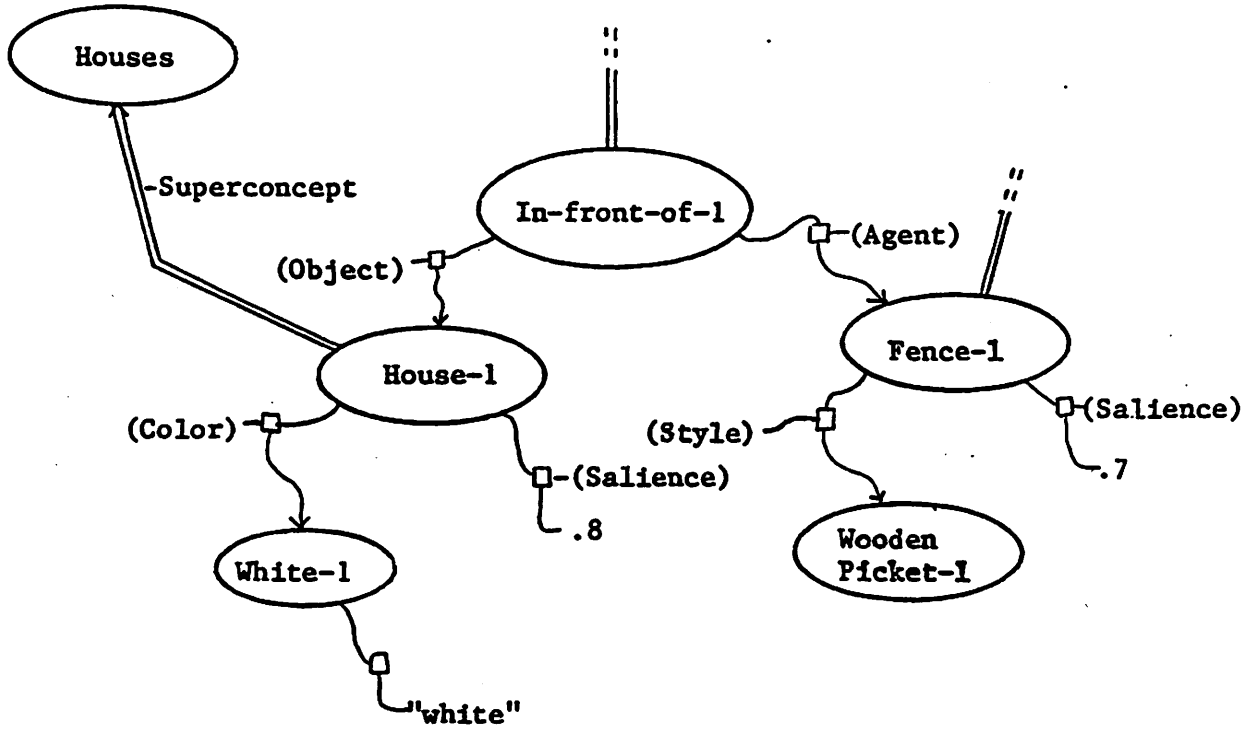


Figure 33: Detail of a KL-ONE Network.

A simple KL-ONE perceptual network. It contains facts that would be expressed in English as "In front of a white house is a wooden picket fence." Concept nodes, which are ellipses, can represent objects, properties of objects (e.g. "white-1"), relationships (e.g. "in-front-of-1"), and gestalts (none shown). Role nodes, which are the small squares, act as "slots" for the concepts which own them, specifying the role played by another concept to their owner.

The internal model of the visual scene is built from concepts functioning in one of the following categories:[4]

---

[4] This epistemology is not adequate to represent *actions* involving more than one object (e.g. Drive(Person, Car)), nor *beliefs* (e.g. Believes(Listener,Red(Door))). Actions involving a single object, however, can be treated as properties of that object (e.g. Running(Person), or even Standing(Person)).

1. *"objects"*: the fundamental entities in the domain, these represent actual objects in the world. Thus, these concepts do not modify or relate entities but are themselves modified and related. For example, the concept for the generic object House describes houses in general (including their structural subparts), whereas the concept for House-1 describes a specific house in a specific scene, and points to specific subpart concepts, specific properties, and specific relations.

2. *"properties"* (or "attributes"): concepts which function to modify, elaborate, or specify the objects in the domain (e.g. the color of an object – Red(Door)).

3. *"relations"*: concepts which express a relationship between two objects (e.g. In-front-of(Fence, House)).

4. *"gestalts"*: concepts which express complex relationships or properties among many of the domain concepts (often not expressible as a satisfactory predicate of any small number of arguments). For example, the House-Scene concept, the Season-of-the-Year concept, and such aspects of the image as the landscaping, whether or not the picture is in focus, and even the similarity of this picture to another are in this category.

Thus, these four categories cover everything that can be perceived by the SALIENCE system.

The Salience Annotation.

This section discusses what salience is (see also [Conklin, Ehrlich, and McDonald 1983], how it is calculated as a parameter of the perceptual process, and how it is represented in the KL-ONE visual representation.

In the KL-ONE data base which is "generated" by the SALIENCE system all four classes of entities have salience values. Recall that in KL-ONE a concept is "modified" by its role nodes. Thus, one of the attribute roles that all concepts have is the Salience role, whose value is a number between 0.0 and 1.0. In the simulated representations these salience values were taken from the experimental data. The average salience rating provided by the subjects for each object in a given scene was converted from its 0 to 7 value by dividing by 7, and the result was used in the network.[5]

---

[5] Actually, this is not entirely true. The experiments only gathered salience values for *objects*. Therefore the salience values shown for properties and relationships represent my own estimation of the relative saliences of these entities.

This is fine for the hand-built representation, but it must also be shown that the SALIENCE system would generate these values and place them in *its* data base. I claim that SALIENCE, or any computer vision system of comparable ability, not only *could* compute salience values comparable to those that were experimentally derived, but that it would do it *for free*, without any special computational effort. This amounts to the claim that salience is an aspect of perceptual, not rhetorical, processing. I therefore further claim that it is only possible to compute salience while performing the perceptual analysis.[6]

The model that results from this perceptual analysis is a subset of the body of world knowledge – in an important sense "understanding" a picture is identifying a cohesive subset of what is known about the world ("finding the right frames") with the elements of the perceptual input. One of the most difficult problems in vision research is the efficient selection of the elements of world knowledge which provide the best (i.e. most complete) account of the raw visual data [cf Selfridge 1982]. In this light the components of salience can be described more abstractly. Since salience has several components, and these are different for each of the categories objects, properties, and relations, each of these must be discussed separately.

Object salience. First, visual processing relies on the conventions of *centrality* and *size* to direct its attention so that its first analyses are of those parts of the photograph which are most likely to yield a potent model for identifying the rest of the scene.[7] Larger regions toward the center of the photograph would thus make the best candidates for initial investigation. Second, elements of the image which are *unexpected* (i.e. which do not have a good "fit" with their slot in the hypothesized frame) are important to the efficient allocation of resources. Thus, frames would need to be annotated with some measure of their "goodness of fit" into the frame assemblage, for use in the evaluation of the quality of the "explanation" offered by the frame assemblage. (Recall that frame assemblages are *competing* with each other for the best account of the data.) Finally, information about the *intrinsic importance* of various items in the scene signals the need for the allocation of additional resources towards the confirmation of their indentification (e.g., if the system is told, as part of its world knowledge, that people are intrinsically important, it would require higher confidence values on instantiations of the "people schema").

---

[6] Note that this is weaker than the claim that salience itself is *used by* the perceptual analysis process. It may be that the elements of salience are merely by-products of the analysis.

[7] One might object that perception always happens in a *context*, and that the kind of perception that occurs depends on the goals of the viewer. This is certainly true – viewers will assign salience in a scene depending on what they think the point of looking at the picture is. However (as pointed out in Chapter COINS TR 83-14) there is an ambient context for viewing pictures to which the following analysis appeals: The picture is a communication act, and it is generally used to show or tell the viewer something. This issue is further discussed at the end of this section.

The three factors listed above are all computed or are easily available within the normal course of internal model building for the SALIENCE system. They are also precisely the three major components of *object salience* as they were described in Chapter COINS TR 83-14: size and centrality, unexpectedness, and intrinsic salience.[8]

Property Salience. Calculating the salience of properties and relationships is a bit more subtle, since these are *about* objects and are therefore dependent on the object's salience for their own salience. Properties derive their salience from 1) their unexpectedness (the color of a red house – in New England – would be quite salient), 2) their intrinsic salience (the color of a fire engine is also salient, but only because red is a bright, attention-attracting color), and 3) (perhaps) the salience of the object to which they are attached.[9]

Relationship salience. Likewise, the salience of a specific *relationship* depends on: 1) its unexpectedness (e.g. "the car on-top-of the house"); 2) the physical distance between the objects being related (e.g. if all else is the same, In-front-of(Object1, Object2) is more salient than In-front-of(Object3, Object2) if Object2 is closer to Object1 than Object3 is); 3) (perhaps) its intrinsic salience (some relationships, e.g."in-front-of", may be more important in general than others, e.g. "in-back-of"); and 4) (perhaps) the salience of the objects being related.[10]

Regarding the salience of properties and relations the following caveat should be observed: these salience values are meant to be relative to each other *with respect to some object*, and not "between" objects. That is, it may well be meaningless to talk about the relationship between the salience of the color of the *gate* and the salience of the color of the *door*.

---

[8] It is conceivable that there are other components of salience which are not computed by the SALIENCE system as described here. See the discussion of Hypothesis IV in Chapter 5 for a refutation of this objection.

[9] This part of the claim would predict, for example, that given two cars of the same color in a picture in which one of the cars is more salient because it is lying on its roof and has people standing around it, the color of the upside down car would be more salient. This claim is quite ammenable to empirical investigation, of the same type described in Chapter COINS TR 83-14.

[10] Note that the claim here is that the salience of objects, properties, and relationships are all mutually interdependent, i.e. the salience of a property depends in part on the salience of the object which it modifies, and this salience comes in part from the salience of the object's properties and relations. Should this turn out under empirical study not to be the case it would in no way weaken the claim that salience is computed for free by the visual analysis process.

Gestalt Salience. I do not offer here an account of how the salience of gestalts might be calculated, though it would surely have at least the components of unexpectedness and intrinisic salience (Out-of-focus is intrinsically salient; the gestalt Snow-covered-ground in a picture of a palm tree on a tropical beach would be salient due to its unexpectedness[11]).

In any case, what is important is that all of the above-cited factors that function as components of visual salience are readily available as parameters in the visual analysis processing, supporting the claim made above that salience is a byproduct of perceptual processing, and not an extra computational expense required by the design of GENARO.

Text generation systems in the past (see Chapter 2) have often used a data base containing, essentially, "and then ..." links. It should also be clear from this discussion that salience is not such a "pre-wiring" of a data base with the order in which items should be mentioned. Such systems side-step the hard issues involved in doing *selection* by building the solution -- the order in which items are to be mentioned -- into the data base.

However, the charge could be made that the hand-built representations used in this system had effectively prespecified the order of mention, via the ordering over items implicit in their salience values. That is, in one sense there is only an implementational difference between on the one hand specifying "A <and-then> B" and on the other mentioning items in order of decreasing "value" and valuing A higher than B. However, it must be clear that since salience is in principle *computed* and independent of the rhetorical planning process, the design of this system allows it to be automatically sensitive to changes in the visual input, which would not be possible if the order of selection were actually prespecified. For example, in the visual represention of a robot navigating through an environment the salience values of the perceived objects would be constantly shifting, and the kind of description the robot's GENARO/MUMBLE system generated would be changing as it moved.

The role of context.

As mentioned above, context has a powerful influence on how a scene is perceived, and especially on how salience is assigned to objects, properties, etc. in the scene. It could be claimed, in fact, that any discussion of salience without a precise specification of the conditions of the viewing act, including the beliefs and values of the viewer, was specious. The various statistical analyses in Chapter COINS TR 83-14 showed, however, that, while context and the viewer's purpose do play a role, these

---

Of course, there is the issue of precisely *what* is unexpected in a "troubled" scene: in a picture of a bush growing in the middle of a highway is it the bush or the highway that is out of place? With gestalts this problem becomes significant, because the gestalt can apply to a considerable portion of the image.

are sufficiently shared among members of the same culture that a discussion of the "normal" perception of and salience assignment in a scene is reasonable.

This shared context can be identified: pictures are communication acts, and as such the viewer can use in their analysis the context that the picture was taken in order to communicate something, specifically to show or tell the viewer something. The rules of using this medium effectively are the subject of courses in photography and art, and will not be discussed here. However, it is interesting to note how some of the parameters of salience that were discussed above follow from the communicative context.

The *size* of an object's image is largely a function of how far the photographer was from the object, and the *centrality* is a function of how the camera was pointed. These sources of salience, then, reflect the decisions that went into the photographer's choice of viewpoint, and therefore the communicative intention of the photographer.

Of course, there are more specific and constraining contexts than the communicative one described above. The same house picture will get different reactions from a real estate broker, an architect, an energy specialist, and a burglar. If the viewer is asked to find "what's wrong with this picture" it will affect their perception process. These factors will bias the perception process by shifting the intrinsic saliences associated with items in the viewer's world knowledge, as well as affecting how resources are allocated for the analysis (i.e. whether fine detail is important, or action, or mood, etc.). In conclusion, much important work remains to be done in this area, but it is the work of perceptual psychologists and builders of computer vision systems, and is not crucial to the understanding and use of salience in computational linguistics.

4.2 MUMBLE and the MUMBLE/GENARO interface

Once GENARO has planned and built an r-spec, that specification must be *realized*, i.e. a grammatical English utterance must be found which conveys the message(s) in the specification. This is the part of the generation process that is truly language specific, since it is at this stage that the rules of the grammar and of lexicalization are used to find an utterance in the language which fulfills the specification.

McDonald's MUMBLE system [McDonald 1980, 1981a, 1983a, 1983b] fulfills this requirement superbly. Not only is the system designed to be a flexible tool of linguistic research (i.e. the grammar and lexicon are both specified as data bases used by the control system), it also embodies some strong claims about processing during realization (e.g. left-to-right top-down indelible construction of the text). In this latter half of the chapter some details of the operation of MUMBLE will be discussed in

section 4.2.1; its dictionary and grammar will be discussed in sections 4.2.2 and 4.2.3; 4.2.4 will present an example of the process of realizing one of the r-spec's from the extended deep generation example in Chapter 3; the issue of lexicalization will be touched on in 4.2.5; and in the last part the difference between GENARO and MUMBLE will be discussed.

The combination of GENARO and MUMBLE represents a complete model of the generation process, one that uses relatively "weak" subsystems embodying interesting claims about needed processing power. One of the most exciting aspects of this model is the opportunity it affords to investigate the *interaction* between deep and surface generation. For example, what amount of detail an r-spec should contain, whether or not the surface processor should be able to ask "questions" of the deep processor (or vice versa), and which subsystem should contain some of the specific processes needed around the interface.[12]

Another interesting aspect of this combination of programs is that both programs are designed to be essentially *real time* in their generation. This is a large part of what the "weakened mechanisms" mentioned above buys this system. That is, at the cost of being fallible this system is able to produce natural-sounding text from a non-linguistic representation at a rapid rate. In fact, it is precisely its fallibility that makes this system psycholinguistically interesting (see Chapter 5).

4.2.1 How MUMBLE works.

MUMBLE is the first program of its kind to be specifically designed for use with source programs that use different representational systems. It embodies several psycholinguistically plausible limitations to its computational power: strictly left to right production and refinement of text, linguistically motivated limitations on the examinable buffer, indelible decisions (no backtracking), and a structural distinction in the treatment of function words versus content words. In addition, the system is driven by the "message" to be expressed (the r-spec), rather than by the grammar. Finally, the linguistic structure of the text being produced is explicitly represented, thus allowing flexibility, generality, and perspicuity of the grammar rules.

MUMBLE can be thought of as a pair of transducers.[13] The first one takes

---

[12] Unfortunately, many of these issues require extended research with the whole system that has not been done to date, due to difficulties in getting MUMBLE to run on the VAX computer at the University of Massachusetts, as well as the enormous processing demands of seriously investigating these empirical systems questions.

[13] It is *not* actually a pair of transducers, however, since the functionality of the two "virtual" transducers has been folded into a single control structure. The justification for this implementation is too detailed for presentation here (but see [McDonald 1981c]).

the elements of the r-spec (in the order in which they occur) and builds the surface phrase structure tree, using the dictionary entries for the respective r-spec elements to direct the process. The second transducer then walks through the tree, using the grammar to produce output text at the leaf nodes. There are four psycholinguistically interesting aspects to this operation:

- The transducers operate *"on-line"*, i.e. the output from the first transducer must be completely consumed by the second transducer before the first one can go on to the next r-spec element at the same level.

- The output of both processes is *indelible*: both the surface structure tree and (more obviously) the output text cannot be changed once they are produced by their respective processes. Thus, as with GENARO, neither of MUMBLE's transducers has any provision for backtracking or lookahead.

- Decisions are made based on *local* information only, that is, contextual information that is local at the position in the tree to the node at which the decision is being made. The tree cannot be scanned for information – any global information needed for a decision must have been anticipated at the point where it was local and deliberately set aside at that point.

- Finally, the overall process must operate in *quasi-real time*; i.e. the number of operations between the consumption of two r-spec elements or the output of two words must be no greater than some fixed maximum which is unrelated to the size of the input or output streams. (This is stronger than the usual time bound of linear time.)

The "dictionary" is the data structure which is used by the first transducer. It specifies the vocabulary of the r-specs, by associating elements from an r-spec with potential realizing phrases: for each r-spec element there is a dictionary entry which specifies how that element may be expressed. This specification is in terms of the linguistic vocabulary established by the grammar. The grammar is then used by the second transducer to: 1) interpret the tree into text, and 2) enforce the constraints and conventions specified in the grammar. The user of MUMBLE does not actually need to know a great deal about the details of its control mechanisms, since the user's job is simply to use and/or extend the grammar to provide the range of English needed for the specific application, and, more importantly, to write dictionary entries for each of the terms in the user's data base, specifying how those terms are to be realized when they are encountered in an r-spec.

4.2.2 MUMBLE's dictionary.

The input interface to MUMBLE is defined by the dictionary. Each possible r-spec element has an entry in the dictionary specifying a realization, or often a range of realizations, for that element. Thus, the process of writing GENARO's rhetorical rules cannot take place without considering how the r-spec elements they produce will be realized by their respective dictionary entries.

Specifically, the realization is specified in terms of parse tree substructure which is to be placed into the tree. When tailoring MUMBLE to a new domain (a new "speaker" in McDonald's terms), most of what one does is write a dictionary entry for each term in the domain data base that might find its way into an r-spec. Entries must be written in a "dictionary entry language", which constrains and specifies what the writer must provide and what can be done by the entry. When the program is loaded, the dictionary entries are compiled into a computationally fast tabular form to enhance the program's run-time performance.

The grammar for dictionary entries is presented in Figure 34. For those familiar with LISP the syntax of an entry will be familiar as a specialized form of a LISP function, with essentially a name, parameters, and a body. The body of an entry consists of a series of "decisions" which are evaluated sequentially. An example of a dictionary entry is shown in Figure 35.

A *decision* is a series of production-rule-like condition/action pairs: if the condition is true, the action is invoked – but in decisions the actions are always "choice-applications". Each choice-application is like a *function call* to a "choice"

```
<entry> :== (DEFINE-ENTRY entry-name
                          <parameter-list>
                          [<decision-form>])
<parameter-list> :== ({local-variable})
<decision-form> :== (decision-name {DEFAULT
                                    <choice-application>}
                                    [[<predicate>]
                                    <choice-application>])
<choice-application> :== (choice-name {<parameter-list>})
```

Figure 34: MUMBLE grammar of dictionary entries.

Terms in <>'s are non-terminals in the grammar; ()'s and terms in upper case are constants; expressions separated by | are alternates of which exactly one must be chosen; expressions in {}'s may occur *zero* or more times; and expressions in []'s may occur *one* or more times.

```
(define-entry in-front-of (agent object)
   default
     (x-is-reln-y agent '#$in-front-of object))
```

Figure 35: An example dictionary entry.

The MUMBLE dictionary entry for the domain relation "in-front-of". It has a single decision, which is composed of a single choice-application, invoking the "x-is-reln-y" choice.

("choices" are also "functions" and are described immediately below). Thus, an entry is evaluated by taking each decision in turn and evaluating the series of choices that compose the decision. The entry in Figure 35 has a single choice, which is to express the relationship in the form "x is relation y". For example, if the r-spec element were "(ELMT1 RELATION in-front-of-1 (fence-1 house-1) ...)", the entry for "in-front-of" would be invoked with the parameter "agent" bound to "fence" and "object" bound to "house", and would simply pass "fence" and "house" through to its single choice. In more elaborate entries, which had more than a single default choice, each choice would have a predicate: when the entry was being "run" the first choice found to have a true predicate (one that evaluated to true) would be the one that was invoked (see "<decision-form>" in the grammar).

A choice, then, is a LISP-like function whose action adds a piece of English syntactic substructure to the surface structure tree under construction. The grammar of choices is shown in Figure 36.

There are two major parts to a choice, one that provides linguistic structure and one that maps semantic elements into that structure. The "<structure>" part specifies the piece of linguistic substructure to add to the phrase marker, using the vocabulary

```
<choice> :== (DEFINE-CHOICE choice-name
                            <parameter-list>
                            {<phrase-part>}
                            {<map-part>})
<parameter-list> :== ({local-variable})
<phrase-part> :== PHRASE <structure>
<map-part> :== MAP <map>
```

Figure 36: The grammar of choices.

The "<structure>" and "<map>" elements are explained in the text.

of the grammar. This structure will have "slots", into which the parameterized values in the r-spec element are inserted. The "<map>" part specifies where each such value in the r-spec element is to be inserted into the structure being built.

Referring to the choice for "x-is-reln-y" in Figure 37, the phrase part uses the terms defined in MUMBLE's grammar to build the tree shown in Figure 38, where the underlined terms are "slots" to be filled by the arguments of the rule. The map part of the x-is-reln-y choice then specifies that the value of "x" (which was the "agent" in the "in-front-of" entry and which has the value "fence") be inserted as a child of "subject" in the tree. Likewise, "in-front-of" and "house" are directed into the "prep" and "prep-obj" slots, respectively. Thus the example r-spec element given above would be realized in this case as "A fence is in front of the house". (The determiner slots are filled by a dictionary decision which is shared among all entries which have a noun phrase realization, based on information found in the r-spec specifying properties of Fence-1 and House-1.)[14]

4.2.3 The grammar for scene descriptions.

In MUMBLE the grammar is represented as a collection of specialized functions which constitute the vocabulary used by the dictionary for specifying the linguistic structures to build. MUMBLE's realization process is strictly constrained by the grammar -- it is nearly impossible to have "canned phrases" lacking in linguistic

```
(define-choice x-is-reln-y (x r y)
   phrase (basic-clause ()
            predicate (vp-pred-adj ()
                        pred-adj (prepp ()
                                        )))
   map ((x . (subject))
        (r . (predicate pred-adj prep))
        (y . (predicate pred-adj prep-obj))))
```

Figure 37: An example of a choice.

The "phrase" part of this choice specifies a piece of tree structure to be built, using terms from the grammar, and the "map" part specifies how to insert the values of the three parameters of the choice, "x", "r", and "y", into the tree structure.

[14] If neither "Fence-1" nor "House-1" were marked as a "NEWITEM", then both would by default be realized using the definite article in the determiner slot.

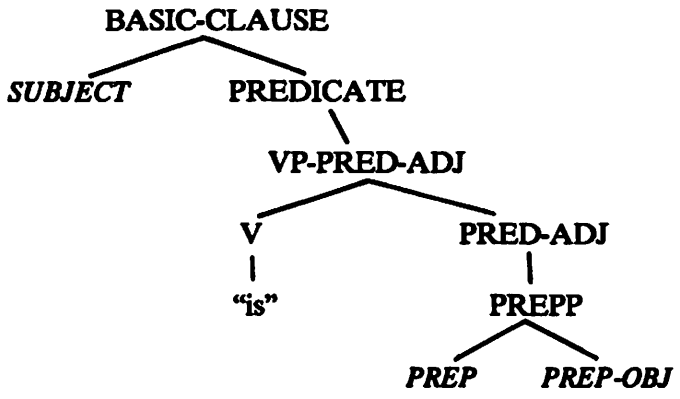


Figure 38: An example partial parse tree.

This tree is constructed by the Choice shown in Figure 37, in conjunction with the grammar entries for each of the terms in the tree ("BASIC-CLAUSE", "VP-PRED-ADJ", etc).

---

structure.[15]

Unlike the dictionary, MUMBLE's grammar is relatively stable between speakers. Applying MUMBLE in a new domain requires writing a whole new dictionary, as the interface to the new speaker, but the grammar only needs to have any new English constructions added which are necessary to express the meanings of the new speaker.

One result of the analysis of the salience experiments was a large corpus of paragraph-length scene descriptions. The data showed that scene descriptions can cover the whole range of English, but that there was a certain stylized subset that was widely used. It was this subset of English that I adopted as the target for the GENARO/MUMBLE system.

The four relationship constructions.

The vast majority of clauses in scene descriptions are for describing spatial relationships. I have catalogued the syntactic forms available for this function, and it turns out that there are just four common ones (see Figure 39). The figure shows each of the four forms in a template notation (although such templates are not used

---

[15] Any phrase which was to be left without linguistic structure would have to be specified as a word, as in "kick-the-bucket". Such a treatment would prohibit any morphological changes to elements of the canned phrase (e.g. "kick*ed*-the-bucket").

Example relationship: In-front-of(Fence, House)

Template notation: Relation(Agent *, Object)

Form SIMPLE
Template: <Agent> is <Relation> <Object>
Example: A fence is in front of the house.
Description: The basic form. It has somewhat more stress on the <Agent> than the other slots.

Form THERE
Template: There is <Agent> <Relation> <Object>
Example: There is a fence in front of the house.
Description: A simple variation on Form SIMPLE. More interesting than Form SIMPLE, it more strongly stresses the <Agent>.

Form REL-FIRST
Template: <Relation> <Object> is <Agent>
Example: In front of the house is a fence.
Description: This form fronts the <Relation>, stressing it slightly. It sounds more interesting than Form SIMPLE. It is sometimes used to *break* the flow of the text.

Form HAS
Template: <Object> has <Agent> <Relation> it
Example: The house has a fence in front of it.
Description: By fronting the <Object>, this form serves to stress it.

Figure 39: The syntactic forms for spatial relations.

* The new item being introduced is always in the <Agent> slot (see text). In the template notation used here, "Agent" simply indicates the first argument of the Relation, while "Object" is the second argument.

---

in GENARO or MUMBLE), and gives an example and a brief description of the use of each form.

Introducing a new item

In the average paragraph description of a picture the main item or theme is introduced in the first sentence. Successive sentences then relate a new item to some previously mentioned item. This pattern repeats until all of the most important objects in the scene have been mentioned. Each clause in such paragraphs, then, describes two objects, an "Old-item" and a "New-item", and provides the relationship that holds between them. Figure 40 illustrates this kind of "chaining" structure. Notice that the Old-item does not need to be mentioned explicitly -- part of our world knowledge is that where there is a house there is almost always a yard. It is also interesting that almost all of the spatial relations in this paragraph are expressed

---

i) This is a *picture* of a large white wooden *house*.
<New-item> <New-item>

ii) In front of the *house* is a white *fence*.
<Old-item> <New-item>

iii) In the *yard* is a *tree*.
<Old-item> <New-item>

iv) Next to the *house* is a *driveway*, *which* is
<Old-item> <New-item><Old-item>
mostly shaded by a large *tree*.
<New-item>

v) In front of the *house* is a *street* and *sidewalk*.
<Old-item> <New-item> <New-item>

vi) Across the *top of the picture* are *power cables*,
<Old-item> <New-item>
and in the lower *left* is a white *mailbox* on a
<Old-item> <New-item>
brown *beam*.
<New-item>

vii) It is late *afternoon*.
<New-item>

Figure 40: New and old items in an actual description.

This paragraph was written by a subject in the experimental studies.

as prepositions -- in this discussion I will only discuss such relations.[16]

As each new item is introduced in the chaining it does not go arbitrarily into either the <Agent> or <Object> slots in the various forms. Rather, as mentioned in Figure 40, the Agent slot normally is filled by the New-item. This is not to say that the Agent/New-item position is always focal -- Form HAS stresses the Object/Old-item. This issue is discussed further below.

It is also apparent from the paragraph in Figure 40 that sentences in actual descriptions are more complex and ornate than the simple templates listed in Figure 39. Several clauses can be conjoined together, or several noun phrases (representing several New-items). And beside the normal elaboration of objects' descriptions, using pre- and post-nominal modifiers, people elaborate the relationships, often with a participle, e.g. "There is a fence *running* in front of the house". Nonetheless, the basic unit of the descriptive paragraph is the relation, expressed in either its simple form, or one of the transformational derivatives.[17]

The uses of the forms

Each of the forms has a different rhetorical force (see Figure 39). Form SIMPLE is the basic "vanilla" form, and was used infrequently by the subjects in our experiments. Part of its limitation is that it appears to stress all three elements (Agent, Object, and Relation) equally. Form THERE is a more "flavorful" variation, and also serves to highlight the Agent. Form REL-FIRST stresses the Agent and, secondarily, the Relation. It has the advantage of leaving the Agent/New-item in sentence-final position, where it may be arbitrarily elaborated. It also seems to have more of a "breaking" force in the flow of the text than any of the other forms; this may account for its popularity in short paragraphs (e.g., it was used almost exclusively in the sample paragraph above.) Finally, Form HAS places more stress on the Object/Old-item, which it fronts; it is used infrequently, and is at least useful for providing syntactic variety.

---

[16] Although spatial relations are expressed in other ways, non-prepositional forms are almost exclusively used in expressing relations from the little used *viewer centered* frame of reference, as discussed below. For example, "The tree obscures the roof of the house" is another way of saying "The tree is in front of the roof" but which stresses the viewer centered frame of reference.

[17] The treatment of these syntactic constructions as a simple form plus three transformational derivatives stems from the way they are generated in MUMBLE. No claim is being made here about the "correct" linguistic analysis.

Some informal experimental results

One of the first questions about these forms (especially in terms of generating them) is whether there are some combinations of forms that are better than others. The evidence, in fact, is that all of these forms can be used successively (i.e. can form a pair of clauses). I performed an informal experiment in which subjects were asked to rate each of the 16 possible permutations of sentences expressing two spatial relations (i.e. "A white fence is in front of the house, and a sidewalk is in front of the fence." is a (SIMPLE,SIMPLE) pair).

The results were essentially negative. With only three experimental subjects, there was not one combination which was unanimously disapproved of, nor any that were unanimously approved of. This supported my observation, based on familiarity with the textual data from the description experiments, that almost any transition between forms could be found, although there were forms which flowed together more melifluously than others.[18] Also, subjects' descriptions illustrated a wide variety of uses and combinations of the syntactic forms. A few paragraphs used the REL-FIRST form to the exclusion of all others, while some used only Form THERE. Thus, the limited data from these informal experiments precludes drawing any conclusion except that preferences for combinations are personal and widely varied.

Use of the forms in the system.

Adding these four forms to the grammar of MUMBLE is a straightforward task. Form SIMPLE is derived directly using the phrase structure component of MUMBLE's grammar, and the other three forms are derived by writing three new entries for the transformation component.

A more difficult issue is: Should MUMBLE or GENARO decide which form to use? This amounts to the following two questions. What factors are involved in choosing the syntactic form? Which component has more ready access to these factors?

To answer the first question requires a theory of focus and stress -- although each of the forms stresses some part of the semantic predicate more than others, it is not clear what makes one item more "stress-worthy" than the others. Stress (both acoustic and syntactic) interacts with quantifier scope, already-backgrounded material, etc. (cf. Grosz, Reichman, Hobbs).

---

[18] This is part of the problem with research on rhetorical conventions – it is very difficult to get hard data, to show that "this rhetorical combination is always wrong". Incidentally, though there was very little agreement about it, subjects seemed to prefer the combinations (SIMPLE, REL-FIRST) and (THERE, REL-FIRST) the most.

The data from the above simple experiment do suggest that one property of the interface between deep and surface generation is that deep generation has few direct restrictions on or requirements for syntax used by the surface component, only weak constraints. This has lead us to the view that, for the moment, GENARO should generally leave the decision about which syntactic construction to use to MUMBLE.[19] Thus, when the implementation of the generation system is complete, MUMBLE's decisions about which form to use will initially be random, followed by the addition of constraints and interactions to the grammatical rules based on observed stylistic "failures" in the generated text.

4.2.4 An example realization.

In Chapter 3 we saw how GENARO operated to construct a series of r-specs that composed a scene description. In this section the operation of MUMBLE in realizing one of the r-specs in that description is presented. The sentence that will be explained is "The door of the house is red, and so is the gate of the fence." (The operation of MUMBLE is presented in greater detail in [McDonald 1981c], and this specific example is reviewed, from the perspective of MUMBLE, in [McDonald 1982b].)

The r-spec underlying this sentence is shown again in Figure 41 (the original appeared on page 55).

When MUMBLE receives this r-spec it first scans it for rhetoremes, making note of the "condense-prop" element, and then begins immediately with the top-down realization process. R-specs are processed from top to bottom, but in two passes -- in the first pass the first rhetoreme element encountered is used to create the root node of the tree; in the second pass, all elements are processed.[20]

---

[19] One possible exception to this is that since (a) the Form REL-FIRST has the rhetorical force of a break, and (b) breaks occur when $newitem runs and pops the next arbitrary object off of the USOL, it seems that it would be useful (and cheap) for GENARO to post advice to MUMBLE to use this fronted-PP form whenever $newitem had run.

[20] This is a relaxation on the stipulation that MUMBLE process the r-spec in strictly top to bottom linear order. If this stipulation is to be observed GENARO will be required to put its elements in an order that MUMBLE can process them in linearly, e.g. with the condense-prop element first in this r-spec. This, on the other hand, would require that GENARO's rules have the power to specify where in the r-spec a new element is to be inserted; currently, *GENARO* builds the r-spec in strictly linear order. This tension illustrates one of the tradeoffs between deep and surface generation that can be explored empirically with this system. I.e. in which program is the added machinery for non-sequential r-spec processing the most parsimonious? This question is not answered in this thesis.

```
(RSPEC NO2
   BODY
        (ELMT1 RELATION part-of-2 (door-1 house-1)
            (door-1 (SUPERC door)
                    NEWITEM)
            (house-1 IBID))
        (ELMT2 PROPERTY red-1 (door-1)
            (door-1 IBID))
        (ELMT3 RHETOREME condense-prop (door-1 gate-1)
            (door-1 IBID)
            (gate-1 (SUPERC gate)
                    NEWITEM)))
        (ELMT4 RELATION part-of-2 (gate-1 fence-1)
            (gate-1 IBID)
            (fence-1 IBID))
        (ELMT5 PROPERTY red-2 (gate-1)
            (gate-1 IBID)))
```

Figure 41: The second r-spec from the example description.

The r-spec based on two Current-items, Door-1 and Gate-1, condensed into one r-spec by $condense-prop.

The dictionary entry for "condense-prop" has two choices: one is to merge the subjects of the relations (e.g. "Both the X and the Y are Prop."), and the second is to use some form of VP-deletion (e.g. "The X is Prop, and so is the Y."). In this case the second of these forms is chosen on the basis of complexity in the r-spec: the specifications of the two halves of this r-spec (the parts before and after the condense-prop rhetoreme) are complex enough (i.e. weigh enough) that the first choice would likely lead to an awkward sentence (e.g. "Both the door of the house and the gate of the fence are red."). This criterion is built into the condense-prop dictionary entry. The surface structure that is built by this choice is shown in Figure 42.

With this root part of the surface stucture in place, MUMBLE begins its traversal. Starting at node "[c1]" each node is expanded according to its dictionary entry and the grammatical forms invoked in that entry. For example, [c1] is expanded as a basic clause, which consists of just a subject and a predicate. The nodes are expanded in a depth-first manner, yielding the partial surface structure shown in Figure 43. At this point there are two strong sources of constraint on the realization of "elmt-5": the fact that it is in a conjunction, and the fact that it has been marked to undergo VP-deletion. MUMBLE knows that any parallel decisions made in the second (or succeeding) conjuncts should be made using the same choices that were made in the first conjunct. For example, since a "red" property was realized in the first conjunct, and another such element is about to be expanded, this

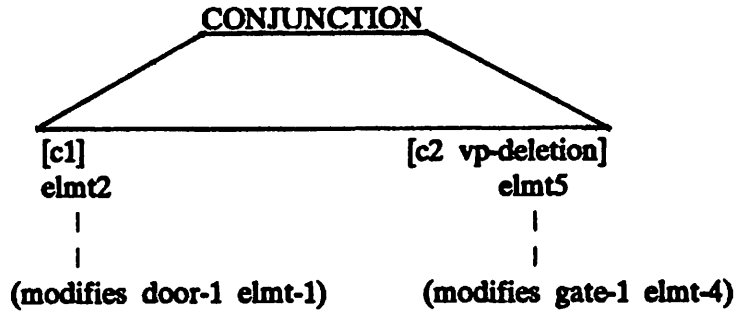


Figure 42: The initial surface structure.

The root and first two nodes of the surface structure. Note that the structure uses the element names to represent what elements have yet to be realized and where they fit into the structure. The condense-prop entry knows that it is the property elements in the r-spec which are the target of the parallel being drawn, so that all other elements are subordinated to these.

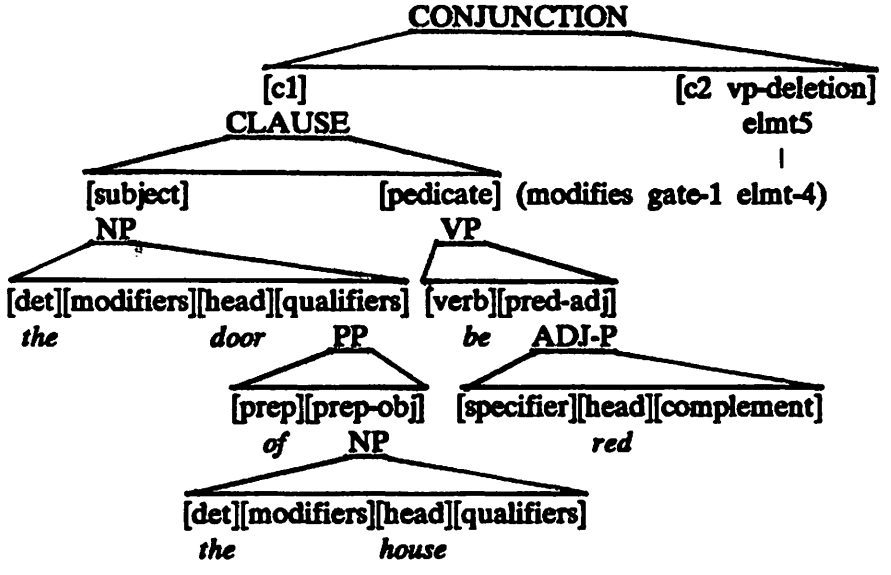


Figure 43: An intermediate surface structure.

The phrase structure tree after the first clause has been realized. So far MUMBLE has output "The door of the house is red, and ...".

constraint will dictate that the same predicate adjective form that was selected before be used again.

The force of the directive to perform VP-deletion is to either transform the selected predicate adjective construction into a "predicate-preposed" form (where the repeated predicate can be pronominalized as "so is") or to leave the word order the same and add an adverb such as "too" or "also". The second choice is reserved for very light elements (in order to avoid stranding the adverb at the end of a complex clause), so the first form is chosen.

The rest of the traversal proceeds in the same way as for the first conjunct, resulting in the second clause being "... and so is the gate of the fence". No deliberations over the realization decisions are needed since the constraint to make the same choice as in the last conjunct dominates the action.

4.2.5 Lexicalization.

In generation the process of lexicalization involves the selection of the word (or phrase) which best realizes the meaning of a semantic term. The "lexicon" is the data structure which provides the mapping from "meanings" to words. The difficulty of doing the mapping depends on the entities in the domain data base -- if there is a one-to-one mapping from entities in the data base to words in the lexicon[21] then the lexicalization process is largely trivial.

One of the reasons for choosing a visual representation as the input data base to GENARO was that there was a natural check on the temptation to load up the representation with linguistic assumptions. For example, if the data base were to contain an entity called Fence-in-front-of-house, and the lexical entry for this entity specified that it always be realized with the canned text "There is a fence in front of the house", then much of the interesting hard work of the generation process would have been hard-wired into the "perceptual" representation.

On the other hand, a computer vision system that actually "understood" the image it was working on -- in the sense of having an interpretation of the scene which included identification of the objects in it and their three dimensional locations -- would also provide high-level knowledge about the objects in the scene and their relationships. Once you have done the hard work of identifying a collection of regions with the frame for "house" in the world knowledge base, then anything that is known about houses in the world (including their lexicalization as "house") is available. The distinction between such high-level conceptual information and

---

[21] In fact, "canned text" is nothing more than a trivial mapping from domain entities to words, phrases, sentences, and in some cases whole paragraphs -- each of which still exists as a single "lexical entry".

linguistic information is not crisp: there is a gray area in which it is very hard to distinguish whether a given fact is linguistic (i.e. including rhetorical and stylistic) or not.

Linguistic versus non-linguistic facts.

The practical problems with working in this gray area are well illustrated by the problem of describing objects which form a class or cluster but which are also separate objects. If several trees are visible in the front yard in the picture, they may be best described as a single entity: the "trees in the front yard". Other examples are the "clouds in the sky", the "path to the front door" (which consists of separate stones or tiles laid roughly in a row), and the "bikes in the yard" (where there are two bicycles lying in front of a house).

The fundamental issue in each case is whether the clustering itself provides an important key to the visual identification of the objects or the scene as a whole. That is, if the process of doing the visual analysis would be well-served by knowing about the possibility of a particular kind of clustering (as is certainly the case for the "clouds in the sky"), then that concept should be in the world knowledge of the vision system. On the other hand, there are certainly cases where objects are mentioned together in a description for rhetorical reasons, and not because they form a perceptual entity. The "bikes in the yard" is probably such an example, especially if they are lying at opposite ends of the yard in the picture.

Thus it would appear that some object clusterings are perceptual objects and some are rhetorical objects, and the distinction has to do largely with the goals of the *visual perception* component and the personal preferences of its designers. This will be true in any A.I. domain where a computer is being used to construct in internal model of a complex environment. In fact, when building the linguistic interface to such systems the status of a given fact as linguistic or not can be treated quite pragmatically, depending on whether the fact is most efficiently and flexibly encoded in the linguistic or prelinguistic components. (The problem of object clusterings is further discussed in Section 5.2.)

As mentioned above, when hand-building the perceptual representation I made an effort to avoid encoding it with linguistic information. However, since there was no working computer vision system available as a "referee", decisions about what was "cheating" were based on my understanding of the needs and abilities of a "SALIENCE" system (see page 68). Subsequent use of this representation has shown that some of those decisions did indeed allow linguistic facts into the "non-linguistic" representation, most noticeably in terms of lexicalization issues that were side-stepped.[22]

[22] Some interesting issues in lexical decomposition and generative semantics were also revealed, but these will not be discussed here.

Lexicalization and the domain data base.

Each of the three classes of perceptual entities in the visual representation – objects, properties, and relationships – have their own lexicalization requirements. Basically the issue with each class is the extent to which "solutions" to the hard lexicalization problems were implicit in the design of the representation of that class. Each of these classes will be reviewed below, along with a discussion of their lexicalization.

Objects – These represented the objects in the scene. Since the SALIENCE system would have a token for each object it does not matter whether that token were named "House-1" or "G258" – there is still usually a simple correspondence between object tokens in the data base and words in the lexicon. And this correspondence can be implemented as a simple table look-up mechanism between object concepts and the word that realizes them. Of course there will be perceptual entities which do not have a specific name (e.g. the regions where the Sky is visible through Foliage), and there will be entities which have several names (e.g. "House", "Cottage", "Building", "Home", etc.). However, such problems are beyond the scope of this thesis.

Properties – These one-place predicates were generally represented in the KL-ONE data base as concepts which were pointed to by (i.e. were fillers for) the role nodes of objects. Again, their names were very suggestive – Red-1 was meant to represent a specific color (in some general color representation scheme) that the system would treat as "Red". That is, it does not matter what the property "Red" is labeled in the data base, but if the system has potentially identified a region as a Firetruck and does not link it to Red-1 (or whatever it is called), it should weaken the Firetruck hypothesis; conversely, if what is thought to be a Road does have that concept as a filler on its Color role then the Road hypothesis should be weakened.

In an actual computer vision system, however, there would be no need (except for the convenience of the system designers) to have the concept that represented the color red be labeled "Red". What *would* be important would be a formula associated with this property concept which specified the range of (Red, Green, Blue) color vectors that would be allowed as instantiations of that node. This property concept could then be specified by object concepts (e.g. "Firetruck") as a constraint on the acceptable fillers of their "color-of" role.

As with object concepts, lexicalization of such property concepts would then be a matter of table look-up of the correct word.

Relations. The lexicalization problem becomes quite complex for relation concepts. In the KL-ONE data base these concepts were given names like "in-front-of-1", meaning simply the first instance in the data base of an "in front of" relationship between two objects. This of course assumes that the SALIENCE system used "in-front-of" as a spatial primitive, which further assumes that that particular

abstraction was a useful device in performing the perceptual analysis. Whether or not this is the case is open to debate, and depends on the extent to which the builders of the computer vision system found the in-front-of relation to be a potent constraint in specifying spatial inter-relations between objects that were allowed, disallowed, and preferred. As with all of the locative prepositions, "in front of" may seem like a basic perceptual entity, but examination reveals it to have a rather rich semantics (see [Cooper 1968]). That is to say, it is only fair to presume that a term that is a *linguistic* primitive (e.g. a locative preposition) is also a *perceptual* primitive if it can be argued that the term is doing real work in the processing of the perceptual analysis.

What are the criteria for perceptual primitives? As stated above, perceptual processing can be powerfully viewed as determining what subset of the system's world knowledge -- what schema assemblage -- can best account for the perceptual data, which in turn leads to the need to specify, in the world knowledge, the distinctions and constraints which actually distinguish between items in the world. In the case of spatial relations the question can be stated as "What is the most powerful set of primitive relations which can be made for stating constraints on objects' relationships?" There are many choices. Since the system is constructing a three-dimensional model it needs a coordinate system, such as a three axis Cartesion system, in which to specify the locations (and sizes) of the objects in the scene. Such a system provides an absolute frame of reference in which spatial positions may be specified as a three place vector. It is awkward, however, for the expression of relationships *between* objects -- for these relations one would prefer an *object-centered* frame of reference, especially for objects which distinguish between their sides (i.e. "front", "back", etc).

In designing the visual data base for this system I considered the following system. Part of the description of each object is a list of all of its relations with other scene objects.[23] Since the descriptions are object-centered, these relations are not commutative.

The relation that object A has to object B *could* be expressed, for object A, in a polar coordinate system: Relation (A, B) would be expressed as the "range" (a term from gunnery), in feet, and the "bearing", in degrees. The zero degree bearing would define "straight ahead", based on a judgment of which side was that object's front side, which would be based in turn on world knowledge about the object. Some objects, e.g. bushes and trees, have no front side, so relations of these objects would be expressed as a single "range" value.

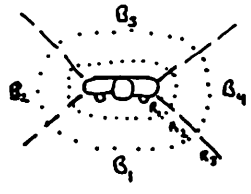
I felt, however, that a coordinate system that used feet (and fractions thereof) and degrees provided too much detail, even for high-level visual processing, and that a representation that was coarser, or more abstract, was needed. Such a system is

---

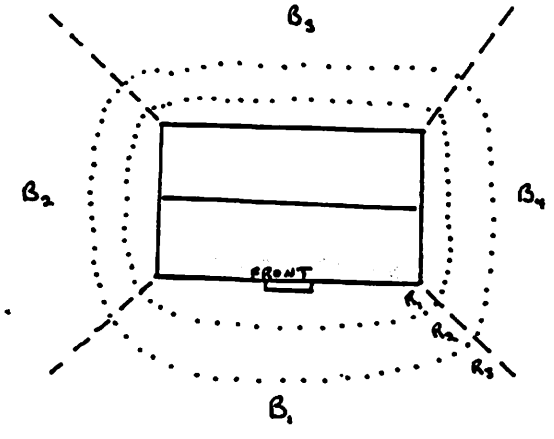
[23] As a practical matter one would not want or need to specify *all* inter-object relations, only those that were some combination of near enough and salient enough.

illustrated in Figure 44. The range is limited to three values, corresponding roughly to "next-to", "near", and "far". The bearing is likewise limited to four values, corresponding to "in-front-of", "on-the-right-side-of", "behind", and "on-the-left-side-of".

The space around Person:



The space around House:



The overlapping of these spaces, showing that
Relation(Person, House) = (R2, B3)
but
Relation(House, Person) = (R2, B1):

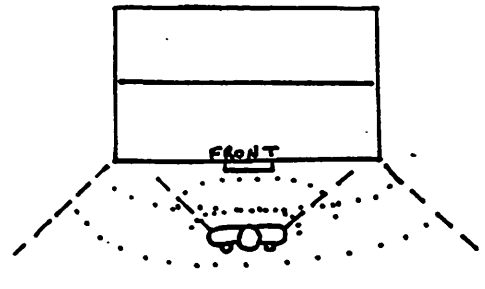


Figure 44: An example of object-centered relations.

In the diagram the dotted lines define the boundaries between "ranges", so that objects within the first dotted line are at range R1, those between that line and the next are at range R2, etc. The dashed lines define the boundaries for "bearings", so that (for objects which have a front and back) the "front side" is bearing B1, the right side is bearing B2, etc.

My claim is that such a system, by greatly reducing the amount of detail in the description of spatial relations, facilitates the specification and use of relational data in the high-level part of a computer vision system, as well as simplifying the lexicalization problem for relations. Whereas a full polar coordinate system using feet and degrees would require the use of complex formulas to determine if something were "in front of" an object, the proposed system divides up the space around an object into a finite number of regions, and the area that is "in front of" that object – that area in which other objects are "in front of" it – can be specified simply as a subset of these regions.

In summary, the purpose of this discussion has been to illustrate some of the problems in the representation and lexicalization of spatial relations. Domain relationships must be expressed in the terms that are useful to the system which builds and uses the domain data base. In this domain, the use of such linguistic primitives as "in-front-of" in the data base turns out to have been a theoretically questionable choice. It was beyond the scope of this thesis, however, to extend the already considerable work that has been done in exploring the exact semantics of the various locative prepositions (see [Cooper 1968], [Bogess 1978], [Waltz 1981], [Herskovits 1982]).

Gestalts. Finally, there is the issue of representing and lexicalizing gestalts in the domain data base. Recall that gestalts are domain entities that express complex interrelationships between multiple domain objects, such as the fact that a scene is a House-scene, or that the Time-of-year is winter. Such entities, although they can be compactly represented as concepts within KL-ONE, do not always find such compact expression in English. Sometimes they are realized as phrases, and even clauses, and thus are not lexicalized per se.[24] Of course, there are also single word realizations for many gestalts, e.g. "landscaping", and these could be lexicalized in the same way as *object* concepts.

There is more than a theoretical problem with allowing into the domain data base primitives that are more linguistic than perceptual. It is possible to combine a non-symmetrical linguistic relation (e.g. In-Front-Of, Behind)[25] and a specification of which of its arguments is to be the New-item, and get an r-spec which is realizable only using Form SIMPLE. For example, suppose that a picture was being described in which a fence was large and central, with a house visible behind it in the background; the Fence has already been mentioned, and now the r-spec contains

---

[24] Those gestalts that have standard phrasal realizations, e.g. "the time of year", could well have a single lexical entry.

[25] Note that there are many examples of such non-symetrical pairs in English, e.g. "above" and "below", "tall" and "short" ("How tall is he?" is not the opposite of "How short is he?"). The members of these pairs, although superficially simple opposites, have different presuppositions.

In-Front-Of(Fence, House) and New-item(House). Now, one might say "The fence is in front of a house" to introduce the House (though in my dialect this is at best weak). But all of the following constructions are bad:

THERE: *There is the fence in front of a house
REL-FIRST: *In front of a house is the fence.
HAS: *A house has the fence in front of it.

If the Fence were the New-item (expressed using Form SIMPLE as "A fence is in front of the house"), as was the case in previous examples, or if the relation were *Behind*(House,Fence) (e.g "There is a house behind the fence"), then this problem would not have arisen.

THERE: There is a house behind the fence.
REL-FIRST: Behind the fence is a house.
HAS: The fence has a house behind it.

What this indicates is that I have gotten "caught" for having linguistic information in the "non-linguistic" data base. If the data base had used "pure" perceptual (non-linguistic) relational primitives, and if the system were really set up to do the hard work of lexicalization, then the above problem would not have occured. The choice of whether to use "In front of" or "Behind" would be left until after the New-item relations had been specified (by GENARO) and the choice would be straightforward.

To correct for this problem in the current implementation two things would have to be done. First, the relationships in the perceptual data base would have to be rewritten along the lines of the perceptual primitives outlined above. This would not affect the operation of GENARO, except to add the complication of handling the cases when two relationships held between two objects (see Figure 44), and choosing which one to use. Second, the dictionary would need to be altered: rather than separate entries for "in-front-of", "next-to", etc., there would be a single entry, "Spatial-Relation", which would compute the correct lexicalization from the range and bearing arguments passed to it.

4.2.6 Why Deep Generation?.

What is the value added by the process of deep generation? I.e. what essential work is it doing that CANNOT be done by machinery designed for surface generation, or, at least, that can only be done much more expensively by that machinery?

In this discussion I will brashly equate GENARO with deep generation and MUMBLE with surface generation. Thus the most straightforward way to answer the question is to show why MUMBLE could not simply look into the VISIONS data

base and generate well-formed paragraphs. Briefly the argument goes like this: MUMBLE's control structure is designed for repeatedly 1) making a choice among alternative forms for realizing a r-spec element based on a fixed dictionary entry, and then 2) indelibly building a piece of surface structure as a direct result of that choice. In order to build a paragraph the r-spec element would have to be something like "Describe-picture" – with its corresponding dictionary entry containing alternate *templates* for the construction of paragraphs.

One of the problems with having MUMBLE do its own selection is that such template schemes are typically inflexible, and require a level of description of the domain data that is detailed enough to allow delicate decisions yet "coarse" enough that the options at decision points can be inspected and evaluated cheaply. (This is not to say that such a template could not be designed, but I hope to show below that doing so would be like forcing a square peg into a round hole.) In the most general case, for example, all possible rhetorical constructions would have to be anticipated by the builder of the template, as well as all potential interactions between template slots.

For example, the first slot would be "Introduce-Main-Item", whose ultimate effect would be to produce a sentence like "This is a picture of a ...". At this point, *GENARO* uses a whole set of demon-like production rules, each looking into the domain data base for the raw materials necessary to propose its specific rhetorical effect. For MUMBLE to perform a similar operation would require

1. that the r-spec "contained" the entire portion of the domain data base that was eligible for mention in the text; and

2. that the dictionary entries were written directly in the language of the domain data base, and not (necessarily) in a language that would support (or even allow) rhetorical deliberations.

It is GENARO's free competition among separate and independent "knowledge sources" that makes its architecture ideally suited for high-level linguistic planning. And it is precisely this style of loosely-coupled rule interaction that is so difficult to achieve with MUMBLE. That is, MUMBLE's design does not allow it to freely examine the contents of its input and organize what to say and what not to say based on desired rhetorical effects. This is not a failure of MUMBLE: McDonald specifically claims (McDonald 1981) that MUMBLE was built within the major A.I. paradigm for generation of an utterance, which establishes two distinct and loosely coupled processes. The first process, to which McDonald gives the generic name "the speaker", speaks the language of the domain data base and is responsible for determining what the desired rhetorical effects are and selecting what specific elements of domain knowledge to include in the utterance under construction. (This includes choice of the main items – the "topic(s)" – and specification of the level of detail.) The product of this process is a specification of the content and style of the utterance (the r-spec) which is used by the second process, "realization".

The job of realization, then, is to find a linguistic expression of the specification which is consistent with the grammar of the language. MUMBLE is such a device: for generality, it uses a user-definable "dictionary" which specifies the language in which the r-specs may be written. The r-spec is "expanded" in a top-down fashion, with the psychologically interesting constraint that decisions are "indelible" -- thus the realization of an input r-spec occurs as an indelible, depth-first process of successively elaborating nodes in a linguistic surface structure tree (from which the output words are taken directly) until all of the leaf nodes are lexical items.

GENARO, on the other hand, is an attempt to build a generalized "speaker" -- a system which embodies domain-independent (as well as domain-dependent) rhetorical conventions in a control structure which allows planning (if via a weak mechanism) of the r-spec. GENARO is also designed to explore the limits of the use of salience (which is taken to be a natural metric contained in the domain data base) as a heuristic in the economical planning of rhetorically effective paragraphs. The choice of a production rule control structure for GENARO allows free and flexible mixing of diverse rhetorical and stylistic conventions. But this freedom is limited in GENARO to rule *interaction* -- once the rule's element has been added to the r-spec there is no going back. That is, like MUMBLE, GENARO uses the indelibility constraint. While this is not as free an algorithm as full scale planning, in which elements might be removed from the r-spec as well as being added, it appears to be powerful enough to

CHAPTER V

IMPLICATIONS OF THE MODEL

In this chapter the more theoretical aspects of the work presented in this thesis is discussed. In the first section, 5.1, the psycholinguistic implications of the system are discussed, while the second section, 5.2, discusses the use of GENARO as a tool in doing cognitive science research.

5.1 The claims of this thesis

In this section the underlying claims about the power of the machinery in the model are discussed, drawing on the details of the salience experiments, GENARO, and MUMBLE as revealed in the preceeding chapters. The spirit of those claims, and of this chapter, is to make strong claims for the implications of the design and implementation of this generation system, and then to discuss strengths and weaknesses of these claims openly.

The fundamental claim being made here is that

> *deep generation can be done quickly and effectively using a data-driven, indelible planning phase, IF the domain data base is annotated with salience.*

To reiterate the intended meanings of these terms, by "deep generation" I mean the process of reasoning about conceptual and rhetorical facts, as opposed to the narrowly linguistic reasoning that takes place during realization. By "effectively" I mean that mechanically generated descriptions for a picture will be indistinguishable from those generated (specifically, *spoken*) by people. And by "data-driven, indelible planning" I mean a style of control in which the input data directly specifies the planner's process of applying its knowledge, and which involves neither lookahead or backtracking.

I will not argue the above claim directly; instead, I have broken it down into a series of subclaims, each of which takes a specific feature of GENARO and examines its necessity and sufficiency in terms of a model of human performance in speech[1] generation. Some of the most interesting and daring claims have to do with the

---

[1] The model being presented here is aimed at accounting for the kind of real-time rhetorical planning that people do during *speech*, not the more laborious and methodical process of writing.

limited power of the machinery in the model, and whether the performance failures of the model demand more powerful machinery or more clever use of the mechanisms that are there. Ideally, such failures will demand nothing – if they coincide in a principled way with human generation failures.

Part of what is at issue here is the human/superhuman fallacy -- a system which generated text without ever hesitating or making a false start, even if possible, would not make a very interesting model of human generation performance, since humans make a wide variety of errors during generation. If, however, the system failed under certain situations and succeeded under others, it has at least the potential of constituting a theory of human generation performance. And the more the Input/Output behavior of the system corresponds to that of people -- the more the circumstances that cause humans trouble are shared by the system -- then the more evidence there is for the theory.

Thus, from a Cognitive Science standpoint, the interesting question is not "Does the system make errors in generation?", but rather "How do the errors of this system coincide with human performance failures?" and "How do the costs of specific operations in the system compare with measurements of speed and memory load in human subjects performing similar tasks?"

In each of the following sections a claim is presented, the means for testing that claim are discussed, and where available the actual results from experiments with subjects and with the model are presented. Many of the discussions of the testability of a claim turn on the problem of evaluating whether some mechanism in the program has "*worked*" or not. This devolves to the problem of evaluating whether a text paragraph is stylistically and rhetorically acceptable. The reader is cautioned in advance, therefore, that "proofs" of a mechanism's adequacy in this area are generally soft. (Psycholinguistic means for testing such claims are mentioned where I have worked them out.)

5.1.1 Descriptions require salience.

The first claim is that

Hypothesis 1

> Some annotation of salience in the domain data base is necessary to organize descriptions based on that data base.

This is another way of stating the commonsense notion that an understanding of what is important, and what is unimportant, is essential to a sensible discussion about anything. The hypothesis can best be tested by proving the falsity of the contrapositive: that perfectly adequate descriptions can be produced without any information about salience in the domain data base.

Testing this claim.

Intuitively, it is hard to imagine how one could describe a picture of a house, say, using a visual data base in which no information about relative salience was available, i.e. in which the house was represented in the same way, and with the same detail, as the fence in the foreground or the small flower at the base of the fence. On what basis would the topic of the introductory sentence "This is a picture of a ..." be selected? If one did not use salience itself as the criteria, one would end up having to use the *components* of salience (as defined here) anyway, such as size, centrality, and expectedness.

The only reasonable alternative to selection based on salience factors is the use of strategies dealing with the form of the data base itself. For example, it might be proposed that the house could be identified as the main item in the scene because its concept node had the most role nodes, or had the largest number of subsumed subconcepts. But these factors are very dependent on the nature and scope of the perception system's generic knowledge about the objects in the scene, and are therefore at best only secondarily related to the specifics in any given scene. That is, if the system happened to "know" a great deal about cars, then the above strategy would find any car in any picture to be the main item in that scene, regardless of its placement in the image.

Some empirical evidence in support of this claim is described below in the discussion of testing Hypothesis I.a.

5.1.2 Salience is the primary strategy.

The second claim is that

Hypothesis I.a

> Natural-sounding descriptions can be generated using, as the primary selection strategy: Mention the most salient things first. A second strategy, Mention items that are directly related (in the domain data base) to the previous item, is of secondary importance in the selection of what to say next.

This claim is a refinement of Hypothesis I; it specifies that there are two major selection strategies, and that the salience-based one has considerably more influence than the relation-based one. This claim takes on greater significance when these strategies are contrasted with other possible selection strategies. It asserts that the

primary strategy is *not* based on (directly)[2] the arrangement of objects in the scene, nor on functional or structural relationships among the objects in the scene. That is, one might imagine that it was necessary to scan through the objects in the scene (i.e. center of the picture outwards) to get an appropriate order in which to mention those objects. But this hypothesis claims that *selection* is primarily a function of salience in conjunction with domain relationships.

Testing this claim.

There are several ways of testing this claim. If the operation of the system actually is primarily salience-based then the quality and range of the descriptions it produces will be an empirical measure of the truth of this claim. There may well be rhetorical constructions that are simply out of reach of such a simple design. For example, a description that discussed the background objects as a group in great detail and then moved on to discuss the foreground objects would have a paragraph structure which required more global awareness during deep generation than GENARO provides to its rules.[3]

One source of potential confusion in this discussion is the distinction between what the content of the text is and how it got there. This claim is *not* that other selection strategies cannot lead to well-formed descriptions. In fact it is clear that people use a multitude of strategies in planning descriptions. In the sentence "This is a picture of a house with a fence in front of it" the mention of "fence" could be due to the salience of the fence, but it also could be due to the speaker's desire to mention something more foregrounded than the house, or a belief that good picture description style demands that fences be mentioned in the same sentence with the object they surround.[4]

Rather, the intent here is to establish the minimal conditions for generation of a well-formed paragraph-length scene description. Additional strategies, more complex conventions, more powerful planning machinery, and feedback between the surface and

---

[2] Certainly these other factors enter into the salience-based strategy, but *indirectly* -- the calculation of salience, as described here, subsumes the other perceptual factors.

[3] However, GENARO could "accidentally" produce such a structure if the saliences of the objects happened to be distributed in the right way, i.e. with all of the background objects having higher salience than the foreground objects, and no strong relations between background and foreground objects.

[4] Note that these are both rhetorical motivations for mentioning the fence. If I had listed the speaker's possible preoccupation with white fences (or whatever), that would count as the speaker's use of the salience strategy, since "preoccupation" is just a form of intrinsic salience.

deep generation processes all may contribute to the richness of human language. The intent here is to discover the minimal machinery and linguistic knowledge to achieve acceptable texts, since on this framework a solid theory of the richer aspects of language generation could be constructed.

For Hypothesis I.a to be true at least 2 conditions must hold:

- that the system is indeed primarily salience driven, and
- that the descriptions it generates are accurate and natural sounding.

The first condition is somewhat open to interpretation. GENARO's architecture is heavily structured to support the salience-based strategy. The USOL is salience-ordered, the Current-item is nothing more than the most salient unmentioned object, and both the Current-properties and Current-relations subregisters of the Current-item are salience ordered. However, GENARO's arhitecture supports two kinds of rules: those which use these structures in the "intended" way (as stacks whose only visible member is the item "on top") and those that override the salience ordering and look *inside* of the various lists. As Figure 45 illustrates, of the 8 rules in the system, only $prop-color and $condense-prop have the ability to reach into the salience-ordered lists and remove an item. Furthermore, these rules generally run at low priorities, so their contribution to the average r-spec is considerably less than the salience-based rules. On the condition that this is a complete set of deep generation rules for this domain, the rule set itself would be evidence for the primacy of the salience-based strategy. This set of rules is still quite preliminary: it is quite likely that many specialized or idiomatic constructions exist for English scene descriptions which have not been captured here. Considerable experimentation and development remains to be done before any claims can be founded on the make-up of this rule

Rule	Relation-based?
$intro	
$prop-salience	
$prop-sal-obj	
$prop-color	yes
$reln-salience	
$newitem	
$condense-prop	yes
$lighting	

Figure 45: Salience-based versus relation-based rules.

"Relation-based" means that the rule has the ability to propose an item which is not the Current-item or its most salient property or relation.

set.

A better test of this hypothesis is to generate paragraphs with a "lesioned" model: disabling alternately those parts of the system which are salience-based and then those which are relation-based will give some measure of the relative importance of those two aspects of the model. To implement a lesion of the relation-based operations is as simple as removing the two relation-based rules: $condense-prop and $prop-color. To implement a lesion of the salience-based operations is also simple: just scramble all three of the salience-ordered lists (USOL, Current-properties, and Current-relations) into random order, thus disposing of the salience information that was encoded on them.

The results of this experiment show graphically the importance (and primacy) of salience in scene descriptions. Figure 46 shows three paragraphs: the first is a "baseline" run of the system on the representation for the picture in Chapter 1 (a thorough trace of this run is provided in Appendix 2); the second paragraph was generated by turning off the relation-based rules; the third was generated by scrambling the salience-order lists in the system. (It should be remembered, however, that, because GENARO and MUMBLE had not, at the time of this writing, been actually connected together, the text shown is a hand-simulated realization of r-specs actually generated by GENARO. However, MUMBLE is clearly capable of producing text of this complexity and grammatical range.)

This is a picture of a white, two story house with a fence in front of it and a driveway next to it. This New England house also has a bush and a tree next to it. Both the door of the house and the gate of the fence are red. The house has a white porch. Next to the fence is a road, and there is a mailbox in front of the road.

This is a picture of a white, two story house with a fence in front of it and a driveway next to it. This New England house also has a bush and a tree next to it. The door of the house is red. The gate of the fence is red. The house has a porch. Next to the fence is a road, and there is a mailbox in front of the road.

This is a picture of some columns. A house has a red door and a fence has a red gate. Next to the house is a driveway. The house has a white porch.

Figure 46: Paragraphs from the "lesioned" model.

A "normal" [5] baseline description of the picture in Figure 1 (page 4), a description of the same picture with the relation-based operations "lesioned" out, and another description with the salience-based operations "lesioned". (The Conclude packet was kept turned off for these runs, preventing the rule $light from running, to simplify the exposition.)

In the second paragraph all that has been lost is the condensation of the descriptions of the Door and the Gate into the same sentence, and the modification of the Porch with its property White. (In the first paragraph these elements were contributed by $condense-prop and $prop-color, respectively.) The third paragraph, besides its rhetorical awkwardness, is simply wrong. It demonstrates that without a notion of what is perceptually important in a picture attempts to describe the scene are doomed. Without the annotation of salience (or its raw materials: size, centrality, and so on) the representation lacks crucial perceptual information that is specific to this picture. Indeed, it *could have* been a picture of some columns -- it is possible to imagine a picture for which the salience-lesioned paragraph is appropriate (if still somewhat awkward).

The second condition on the truth of Hypothesis I.a, that the descriptions generated by the system be accurate and natural sounding, is amenable to a Turing-style test: take a picture and mix descriptions of it that were generated by people and by the system, and test how well human subjects can differentiate between them. If subjects can guess the source of the descriptions with statistical reliability, then the claim has been disproved. This experiment has not been performed, due to the present technical difficulty of getting more than a very few mechanically-generated descriptions -- see Chapter 6.

Finally, there is another source that is helpful in determining the sufficiency of the salience strategy. In [Conklin, Ehrlich, and McDonald 1983]it is claimed that there was only a fair correlation (about .52) between the salience rating data and the textual data. One way this might be interpreted is as a measure of the importance of the salience strategy in subjects' descriptions. If the subjects were using only the salience strategy a correlation of 1.0 between the rating and textual data would be expected. If, on the other hand, the salience of an object had nothing to do with when it was mentioned one would expect a 0.0 correlation.

The correlation of 0.52 could be thought of as meaning that roughly 50% of selection and the resultant object ordering was due to the use of salience-based selection. The other 50%, presumably, was due to other strategies, e.g. the relation-based strategy. However, it would mistaken to give full weight to these data, however, because of the procedural difficulties and dangers inherent in reducing peoples' notions of salience and their written paragraphs to a few statistics based on simple quantification procedures (these procedures are described in detail in [Conklin, Ehrlich, and McDonald 1983]). However, the 50% figure is high enough to be at least weakly confirming of Hypothesis I.a.

5.1.3 Stepping down the USOL is sufficient.

The third claim, and another subclaim of Hypothesis I, is that

Hypothesis I.b

> I define the *Locality Constraint* to be the following limitation on the power of a deep generator: each domain item is made the Current-item once, and the system rules describe only the Current-item. The result is that an item is described at only one point in the text; the claim is that this is sufficient to cover a broad range of descriptive texts.

For example, before it becomes the Current-item the Fence might be mentioned by virtue of being related to the Current-item (i.e. "... a house with a fence in front of it."), but it will not be described; and once Fence has had its chance as the Current-item, it will only be mentioned again by being related to something further down the USOL (i.e. Sidewalk). This claim actually has two parts:

1. that each object gets only one chance to be the Current-item, and

2. that only the Current-item gets described.

I distinguish here between "mentioning" and "describing". An item is mentioned by having an explicit reference to it in the text (e.g. in "The fence is green." both the object Fence and its property Green are *mentioned*). An item is only described when there is some elaboration to the reference which gives additional information about the item (e.g. only the Fence is *described* in "The fence is green"). In GENARO's architecture only objects are described (by becoming the Current-item; see Hypothesis V below), and description occurs through the proposing of the items from the Current-properties and Current-relations subregisters.

Testing this claim.

Some possible interactions that are barred by this claim are

- A stack underlying the Current-item, so that items described could be pushed onto this stack and popped off for further description later on (see also hypothesis I.c);

- Giving rules the ability to propose properties or relations of objects other than the Current-item.

Much of the force of this claim hangs on what may seem to be an implementation detail: whether or not to remove an object from the Unmentioned Salient Object List when it is mentioned (i.e. mentioned by virtue of being related to

the Current-item). In the current implementation such objects are removed from the USOL at the moment they are mentioned. For example, in the example in Chapter 3, part of describing the House (when it was the Current-item) was mentioning its relationship to the Fence. But when Fence is removed from the USOL on the basis of being mentioned in this r-spec -- because it is indeed "mentioned" at that point -- then the potential for an interesting problem is created. Suppose the USOL is (House Fence ... Road ... Sidewalk ...); Fence gets mentioned in the r-spec with House, removing it from the USOL. But suppose that the relation Next-to(Fence, Sidewalk) is very salient; because only the Current-item gets described, this relation will not get mentioned until Sidewalk becomes the Current-item. However, if while Road is the Current-item Sidewalk gets mentioned, i.e. in the relation Next-to(Road, Sidewalk), then Sidewalk will be removed from the USOL for the same reason Fence was. Hence neither Sidewalk nor Fence ever becomes the Current-item, and the relation between them is never mentioned. In fact, neither one is ever properly described -- only mentioned briefly.

This is a nice example of a failure of the simple control structure and the Locality Constraint it embodies. It indicates that the machinery is, in places, too weak to generate acceptable texts. This is methodologically very important: the intent of the experiments with this system have been to determine what the *minimal* machinery and linguistic knowledge are to produce "acceptable" scene descriptions. And while there is certainly some latitude in the definition of "acceptable", it is important for the claim of minimal power that the program sometimes falls below this minimum (otherwise, how would you know that it was minimal?). Another point for the methodology used here is that it is generally easier and clearer to add power to control mechanisms than to take it away.

Of course, it is not crucial to GENARO, or even to the Locality Constraint, that objects which are mentioned via a relation with a Current-item get removed from the USOL, and the example above suggests that it is wrong to do this. But there is a complication associated with the obvious alternate scheme of requiring that *all* USOL objects become the Current-item on their way from the USOL. Suppose, in the above example, that items which are only mentioned, e.g. Fence in the first sentence, are not marked "mentioned", and are left on the USOL.[6] In this case, after the initial sentence "This is a picture of a house with a fence in front of it", whenever Fence became the Current-item it would lead to some sentence like "The fence is white and has a gate." This would describe the most salient property and relation of Fence, but the bridge between this sentence and the first mention of Fence would be awkward at best. That the Fence is White would perhaps be better mentioned in the first sentence, e.g. "This is a picture of a two story house with a *white* fence in front of it." (The best place for stating that "The fence has a gate"

[6] This is equivalent to the more precise modification to the program of making the USOL into an "Un*described* Salient Object List", and having the system mark objects as "described" instead of "mentioned".

would depend on the salience of Fence and what other relationships it had to objects in the scene.)

But such constructions, in which an object is *described* (not just mentioned) in two different sentences, are *impossible* under the Locality Constraint, the basis of this Hypothesis. An object either gets completely described when its turn as the Current-item comes, or it does not get described at all (only mentioned).[7] How much this actually impairs the range of descriptive texts generable by GENARO is an open question, requiring further experimentation with the system to answer.

There are at least three independent enhancements to the machinery of this model that would allow it to generate these more natural r-specs (i.e. r-specs in which two non-parallel objects were described). However, these enhancements do not seem to observe the Locality Constraint it its strict sense (i.e. that each domain item is made the Current-item once, and the system rules describe only the Current-item). The following description of these enhancements relies on a distinction between the three major components which comprise GENARO. These components are the *rules*, the *algorithm*, and the *data structures*. Previously the distinction was made between the control structure (the "machinery" of the program) and the rhetorical rules. Here I am further distinguishing within the control structure between its data structures (e.g. the USOL) and the algorithm, which specifies both how information can move between data stuctures and how and when the rules can run. (This is a powerful distinction that can be made when analyzing any AI program.)

Each of these components could be enhanced to provide the additional power needed to avoid the holes in the planning described above.

- Making the action of switching Current-items during r-spec construction more commonplace, by adding to the *rules* the ability to discern when to perform this powerful action.

- Adding to the *algorithm* the ability to TEMPORARILY set the Current-item to some other object which is being mentioned (e.g. Fence, in the example above), so that if the new object had any highly salient properties or relations they could be expressed.

- Adding to the *data structures* a "Related-item" register, which would be set when any relations of the Current-item were being put into the r-spec; the Related-item register would combine some of the properties of the last two options: the rules would have to know how to use it, and it would have

---

[7] An object's description may extend over more than one adjacent sentence -- this would be done by modifying the rules which control the actions of ending r-spec construction and sending the r-spec on to MUMBLE so that they allowed dispatching the old r-spec without bringing in a new Current-item (see Chapter 6).

higher salience thresholds than the Current-item register has.

These are presented to give the reader a sense of how one might go about extending this deep generation model, and will not be explored here any further (but see Chapter 6).

[Give an example of some of the complications that arise from using any of these schemes to have two levels of focal-ness. This demands a full experiment. 1) Select an enhancement (eg #3); 2) Make up a data base/picture in which everything is either salient or not; 3) Describe in detail how the system can plan itself into a corner; 4) thus, claim this enhancement requires much more global control to use effectively, so that 5) it is not a simple patch to extend the power of GENARO, and finally 6) present full scale planning as a probably-too-powerful solution to these complications. Refer to Hypothesis V. (indelibility)]

5.1.4 Rhetorical structure is not recursive.

The fourth claim is that

Hypothesis I.c

> Descriptive paragraphs are not generated by a recursive (i.e. stack) mechanism.

This amounts to the claim that the machinery for parsing and generating paragraph structure of the order of a finite state automaton (FSA) in power. In general, stack mechanisms are more powerful, because they can generate/parse (hereafter simply reffered to as "generate") structures which are indefinitely embedded. However, any specific paragraph has a finite (and usually very small) amount of embedding, and whatever that amount is, an FSA could be made which had enough states to be able to generate it. The key difference is that if just one more level of embedding were added to the paragraph structure, the stack machine could still deal with it, but the FSA could not.

The difficulty with claims about the nature of the underlying structure, as this one is, is that they must argue from behavior to internal mechanisms, since the internal mechanisms of language are themselves inaccessible for direct study. To prove this claim wrong would require a demonstration that, no matter how deeply embedded the structure of a paragraph was, a person could generate it. On the other hand, it cannot be proved correct, since there is no behavior of which an FSA is capable but a stack mechanism is not.

However, it is nonetheless worthwhile to examine paragraph structure in terms of the structure of the text itself. In these terms the problem is whether the structure of text is basically recursive, or whether there simply is a certain amount of

returning to a previously mentioned object to further describe it. Another issue is how the power of GENARO relates to that of an FSA with respect to paragraph structure.

One consequence of the last hypothesis (#I.b) is that once introduced, objects are only returned to by way of some related object being described. This is in contrast to the notion that we describe a certain item to a point, then *set it aside* while elaborating something relating to it, and then *return to* describing it further.

The notion that discourse structure is recursive is proposed, among others, by Barbara Grosz [Grosz, 1980], and is also quite intuitive. However, since the claim of recursiveness requires a stack mechanism (if only of limited depth), and since this extra machinery adds considerably to the power of the current program, it is therefore undesirable unless justified by the data.

Testing this claim.

Again, this Hypothesis cannot be "proved", though it might be weakened by a well-formed paragraph which had a *necessarily* recursive structure. Figure 47 presents what may be such a paragraph. A "recursive structure" analysis of this paragraph would claim that it starts out being about the scene itself and its main object, the HOUSE, then "pushes the stack" to elaborate the PORCH (4), then pushes the stack

---

(1) This is a picture of a large old plantation home.
(2) The house is surrounded by large trees, and
(3) a field extends beyond it in the background, with
some mountains in the distance.
(4) There is a porch in the front of the house, and
(5) 4 large columns at the front of the porch.
(6) There are chairs on the porch, along with a table, and
(7) some of the chairs look like rocking chairs that face
off towards the mountains.
(8) The porch is white, as is the whole house, except for
a black trim around the windows.
(etc.)

Figure 47: A possibly recursively-structured paragraph.

This paragraph was written by the author to make a point. While it would be easier to evaluate with a copy of the picture it claims to describe, the point here is simply that it is a "good" paragraph -- that it is smooth and natural sounding.

again to elaborate some things on the PORCH ((5) and (6)). Then it pops back to discussing the PORCH itself ((8)); presumably it would eventually pop back to further description of the house.

However, the deep generation model offered here can nearly generate this description, without any enhancements to the machinery of the model. According to the current hypothesis, this paragraph consists of (at least) four simple shifts to the next most salient object: HOUSE, PORCH (4), CHAIRS on the PORCH (6), and color of the PORCH (8). The weak link in this claim is that in this paragraph that last shift was to a *property*, rather than an object. Evidently, in order to produce *this* paragraph we must either give up the current hypothesis or the claim that descriptions are "object-driven" (Hypothesis V), since there is no way to produce sentence (8) in that position using the current architecture (i.e. without somehow making the Current-Item PORCH once again). Strictly speaking, then, this hypothesis is false.

However, there is one last thing to say for non-recursive machinery: that while it cannot produce *all* legal paragraphs, it can produce a subset of them which *expresses* all of the semantic content of the full set. Hence, if sentence (8) were to come immediately after sentence (4), this paragraph would no longer be recursive, and in fact, with appropriate use of pronouns, is also smoother, at least in my dialect.[8] In any case the rearranged paragraph is not inferior to the one which had recursive structure, and the intent of Hypothesis I.b is to claim that there are no situations or their descriptions which *demand* recursive structure, i.e. no expressive power is lost by foresaking recursively structured paragraphs for those structured locally using salience.

5.1.5 Iterative proposing is necessary and sufficient.

The fifth claim is that

Hypothesis II

> No more or less than the power of Iterative Proposing is required to effectively use rhetorical conventions when they are expressed as production rules.

---

[8] The paragraph in that case would be: "(1) This is a picture of a large old plantation home. (2) The house is surrounded by large trees, and (3) a field extends beyond it in the background, with some mountains in the distance. (4) There is a porch in the front of the house. (8) It is white, as is the whole house, except for a black trim around the windows. (5) There are 4 large columns at the front of the porch. (6) There are chairs on the porch, along with a table, and (7) some of the chairs look like rocking chairs that face off towards the mountains."

Iterative Proposing (I.P.) is the control structure used by GENARO; it follows the traditional production rule paradigm, with the additional provision of a system of "*priorities*" which are posted by the competing rules and which determine which of the active rules actually achieves its particular action. Thus the rule's knowledge is applied by successive rounds in which two things happen: all active rules make a proposal, and the best of these is "run".

The claim here is that this control structure is necessary and sufficient for efficiently coordinating GENARO's rhetorical rules.[9]

Testing this claim.

The *sufficiency* of I.P., as with all other aspects of the system, rests on the judgment that the overall behavior of the system is sufficient: if the system is simply the sum of its parts, and the system is sufficient, then the parts are necessarily sufficient. Thus the sufficiency of I.P. ultimately rests on the judgment of the sufficiency of GENARO as a deep generation component.

Perhaps, however, the system is more than sufficient -- perhaps it has more than enough computational power to generate scene descriptions. In this case some part would be unnecessary (where "part" is taken very loosely), and the claim of *necessity* would be false. Specifically, given the production rule framework, are iteration and proposing both necessary? Recall that *proposing* entails that, in a given round, only *one* rule wins and gets its way, as opposed to all the rules which are eligible running and getting their way all on the same round. *Iteration* is the natural complement to proposing -- only one rule fires for each round, so the program iterates through many rounds of proposing to give the rule set a larger opportunity to "express" its knowledge.

Given the one-rule-per-round aspect of proposing, iteration is clearly necessary -- almost no r-spec would be complete with only one element in it.[10] But is *proposing* necessary, i.e. why not let all the rules that have a proposal win and have their way?

---

[9] This claim may be circular in a sense, i.e. the rhetorical rules described here are to some extent written so that they demand exactly the power of Iterative Proposing. I argue below, however, that more power is not necessary, and that less would cause a serious loss of performance.

[10] Of course, the notion of "r-spec *element*" could be expanded to be a semantic packet large enough to specify a complex sentence, or even a paragraph. However, to do so would be equivalent to claiming one sentence of output per rule proposal (since a 1:1 correspondence between r-specs and sentences has already been imposed -- see page 21 and Hypothesis VI.b, below), which would place a demand on the rules that lead to a severe loss in generality and flexibility.

The example trace of the generation of a description in Section 3.3 contained several instances in which a series of elements were inserted into the r-spec in an order-dependent manner -- latter elements were proposed based on the insertion of previous elements into the r-spec. This amounts to a very limited form of inter-rule communication: since the only actions available to rules are additions to the r-spec or changes in the Current-item, and since both of these data structures are also available for inspection by the rules' preconditions, a one-way narrow-bandwidth channel exists between a rule being processed and the winning rules that preceeded it in the current r-spec. In particular, any of the rules that check the r-spec in their preconditions clearly will require more than one round of r-spec building in order for that check to be at all useful.

For example, all of the condensing rules base their actions both on the existence of a rhetorically parallel object somewhere in the USOL and on the size of the r-spec not being too large. Without multiple rounds of successive r-spec building (i.e., if all rules which fired achieved their action), several condensing rules could propose condensations that would catastrophically overload the r-spec any time the Current-item had several potential parallels. Although such collisions could be programmed around, it would be at the cost of adding considerable power to the rhetorical rule language and to the complexity of the rules themselves.

Not only the proposals but also their *priorities* must be recalculated after each element is inserted in the r-spec, since rules can base the priority of their proposal on the weight of the r-spec.

In summary, I.P. offers a weak form of inter-rule communication that is still strong enough for efficient coordination of rhetorical effects in the course of the construction of an r-spec. I.P. is necessary, since loss of either the proposing or the iterative aspects would critically damage the power of the system.

5.1.6 Rhetorical planning can be done indelibly.

The sixth claim is that

Hypothesis III

> The planning of r-specs can be effectively managed indelibly -- backtracking is not needed, because the domain of rhetorical planning is resilient enough that it is in fact difficult (at this level) to "paint yourself into a corner".

Testing this claim.

The planning of a descriptive text is done as a pair of iterative processes: the construction of successive r-specs describing objects of lower and lower salience in the scene, and the insertion of successive rhetorical elements into each r-spec as it is built. These processes have a very simple control structure -- they are both indelible, since there is no look-ahead or backtracking. While this is an unusual design for a planner, it makes a strong claim about the complexity of the process, which is, if deep generation can be done well deterministically without extravagant costs elsewhere in the process (e.g. backup hidden by being hard-wired into the rules) then this planning process can be said to be inherently deterministic.[11]

There is at least one specific situation in which GENARO fails to plan adequately (described next), and which either counts as evidence that GENARO, due to its indelibility and/or its extreme locality, is not a *sufficient* model of deep generation, or is an illustration of the system successfully modeling a human performance failure.

This failure occurs as a result of the inability of the system to look ahead even one r-spec element.[12] Suppose that the r-spec is filled up to 75% of its optimum weight and there are two competing proposals: one is a condensation, the other is Finish-building-rspec. The later proposal is being made at a low priority (slightly lower priority than the condensation proposal), to avoid the r-spec being too light. Thus the condensation succeeds, pulling in a new Current-item and starting a fresh round of proposing. Since the Current-item is new, there will be several high priority proposals based on it. But as the r-spec gets heavier, Finish-building-rspec is making the priority of its proposal higher and higher. At some point the r-spec is overweight enough that Finish-building-rspec wins, and the r-spec is sent to MUMBLE for realization. However, since the cutoff point was determined without respect to either r-spec content or "goals" of the condensation rule[13] , and since r-spec construction was cut off when proposals were still being made at a high priority, there is a good chance that the resulting r-spec will be awkward, or perhaps even impossible, to realize grammatically.

---

[11] That is, the rule of parsimony dictates that, even if a strong, nondeterministic mechanism can account for a phenomenon, if a weaker, deterministic mechanism can account as well for the same phenomenon then the process itself can be said to be of the type of the weaker mechanism, i.e. deterministic.

[12] The difference between lookahead and backtracking is largely implementational. Both serve to give the control structure a gap between when a choice is contemplated and when it must actually be committed to.

[13] Condensation rules do not have goals per se, but they count on the property or relation upon which the condensation was based being (independently) inserted into the r-spec.

For example, suppose that just before the condensation the r-spec would have been realized as "The door of the house is red ...", and that the condensation made "gate" the Current-item, and that "gate" had the salient properties "red" and "broken" and the salient relation "part-of(fence)". Assuming the same dictionary entries as were presumed in Section 4.2.4, and following that same processing, MUMBLE would next choose to realize the second clause under the constraints that it is a second conjunct and that it is marked for VP-deletion. If MUMBLE chose to follow the same realization as was described in that example, it would complete the realization of the second conjunct as "... and so is the gate of the fence ..." without ever having encountered a suitable node in the surface structure at which to expand the "broken" element in the r-spec.

This could just be a failure in MUMBLE: the grammar rule which allowed VP deletion could detect conditions that would cause it to fail and would force some other (less complex) realization. This could also be a failure of rhetorical planning: GENARO's specification of condensation, and the resulting choice in MUMBLE to realize the r-spec as a pair of conjoined clauses, lead the system down a blind alley. If GENARO had "known" that it was going to be adding several salient properties of the parallel object to the object's description, it could have signalled that to MUMBLE early in the r-spec, possibly leading to the realization "The house has a red door, and the fence has a broken red gate that is on fire."

The general issue here is that GENARO talks to MUMBLE in a language that is both extremely large (perhaps infinite) and only loosely defined. From GENARO's point of view the range of possible r-spec's is limited only by the number of rules and the contraints provided by the packet-switching mechanism. MUMBLE, on the other hand, has dictionary entries that are written with specific r-spec elements, or combinations thereof, in mind. In any domain GENARO will surely create r-spec's which the dictionary writer has not anticipated. One's hope is that MUMBLE's dictionary is written with enough flexibility and redundance that even unanticipated r-spec's will be realized correctly, due to the linguistic generalizations that are captured in its dictionary and grammar. However, GENARO's control structure offers so few limitations to the possible patterns of rule firings and interactions that there will always be the likelihood of breakdowns, i.e., r-spec's which from GENARO's standpoint are perfectly well-formed but which MUMBLE's dictionary is not capable of realizing.

One of the most likely modes of failure will be r-spec's which are too long, as described above. The psycholinguistically interesting question is, Is there evidence of such behavior in people? Clearly people stop in mid-sentence, and start their utterance over with a different construction. But is this because they inadvertantly tried to pack too much into their "internal r-spec", only to find that they have to

either say something ungrammatical or start over?[14] Unfortunately, the kind of studies necessary to resolve these psycholinguistic issues are beyond the scope of this thesis. The force of this claim, however, is that it will be found that there are clear analogs between the errors of the GENARO/MUMBLE system and those of people.

"One of the most likely modes of failure will be r-spec's which are too long, as described above. The psycholinguistically interesting question is, Is there evidence of such behavior in people? Clearly people stop in mid-sentence, and start their utterance over with a different construction. But is this because they inadvertantly tried to pack too much into their "internal r-spec", only to find that they have to either say something ungrammatical or start over?[15] Unfortunately, the kind of studies necessary to resolve these psycholinguistic issues are beyond the scope of this thesis. The force of this claim, however, is that it will be found that there are clear analogs between the errors of the GENARO/MUMBLE system and those of people."

COMMENT: It should be possible, for example, to collect a corpus of errors made by subjects describing pictures out loud for an experimenter and tape recorder. This would be huge project, however. Not only would we have to accumulate enough data to get some errors, but we would then have to look hard at the grammar for MUMBLE and see how we would account for specific empirical errors. Too much work. Later.

COORDINATING SEVERAL CONDENSATIONS

Another rhetorical construction that is problematic for GENARO is the coordination of several condensations. For example, suppose that in the Winter House Scene of Figure 1 the door of the house, the gate of the fence, and the box of the mailbox were all the color red, and were on the USOL in that order. One possible description of this situation would be an utterance like "There are three red things in the picture: the house door, the fence gate, and the mailbox", or at least "The house door, the fence gate, and the mailbox are all red." However, the current set of rhetorical rules could not put all three objects into the same r-spec, because the r-spec would be too heavy, and it is difficult for the rules to coordinate several (even two) condensations. Specifically, $condense-prop checks the weight of the r-spec before suggesting a condensation, and any reasonable cut-off weight for general r-specs would

---

[14] There is a methodological danger with such questions: while it is fairly straightforward (if tedious) to gather data on people's performance failures, one must be very careful about attributing these to a particular condition in the generation machinery (e.g. "r-spec overload").

[15] There is a methodological danger with such questions: while it is fairly straightforward (if tedious) to gather data on people's performance failures, one must be very careful about attributing these to a particular condition in the generation machinery (e.g. "r-spec overload").

be too low to allow a second condensation within one r-spec.[16] Unless the condense rules can be rewritten within the current control framework this limitation stands as a real shortcoming of simple I.P., one that would surely be remedied by doing "real" planning of rhetorical effects as part of the r-spec building process. That is, one might imagine that the same "short-sighted" rules could be orchestrated by a more powerful control mechanism, one which, say, collected all possible condensations and then allowed a "meta-condensation" rule to look at the collection, looking for further condensations. Such a rule could notice the proposals to condense the house door with the fence gate and the fence gate with the mailbox, and could propose a single, more encompassing condensation.[17]

5.1.7 Salience is perceptual.

The seventh claim is that

Hypothesis IV

> Salience is perceptual, not linguistic. The components of visual salience are computed as a by-product of constructing an internal model of the scene in a picture, so that selection based on salience is a scheme in which order of mention is distinctly *not* prespecified in the visual representation.

In other words, the annotation of an object's visual salience can be provided as a natural part of the perceptual analysis of that object in the image, and it is this analysis which determines when the object should be mentioned.

---

[16] It may in fact be possible to alter $condense-prop so that multiple-condensation constructions could be made. In fact, it could be done just using a more elaborate scheme of assigning and checking weights of r-spec elements. The key might be: (1) changing the weight assigned to condense elements (the more there were of the same type, the more NEGATIVE their weight); or (2), what is the same as (1), letting $condense-prop raise the *optimum-rspec-weight global used by $Finish-building-rspec; or (3) again the same kind of thing, create a type of r-spec element that has negative weight and let the rules "barter" about sending the r-spec in the currency of its weight -- each time $condense-prop added another condense rhetoreme, thus setting up the "goal" of some amount of r-spec building, it could deposit a certain amount of these "lifting elements" to finance the weight of the new elements.

[17] [Another way out: look into the visual representation for situations like <several objects with the same salient color> and do the condensation (as reification will have to be done) at the perceptual level. Why not? Clearly the difference between perceptual and rhetorical objects and concerns is blurred throughout this project -- is reification a perceptual or conceptual or rhetorical phenomenon? See Part Two of this (implications) chapter!]

The significance of this claim is that, unlike previous efforts at generating text from a large data base (e.g. [Goldman 1974], [Swartout 1977], Davey [1979], [Mann and Moore 1981]), the input data base to this system is not "pre-wired" for the order in which items in it are mentioned. Furthermore, because the salience annotation is assigned as an integral part of the system using the data base (SALIENCE), the selection process and the resultant ordering of objects in the text are directly responsive pre-linguistic forces. This application for studying the selection problem is much more realistic than in domains in which the order of mention of objects is prespecified.

Testing this claim.

The justification of this claim appeared in the previous chapter (page 74). It was argued there that there are three necessary components of visual salience, and each one is computed as the by-product of a necessary subprocess of the perception process. The current claim might be shown wrong in two ways. It might be that one or any of these specific subprocesses are actually unnecessary for machine perception. In that case the corresponding component(s) of salience would not be computed "for free", but rather would involve extra processing. Whether or not this is the case is unfortunately an academic issue until a full-blown system for doing visual perception is operational, and it can be determined what the necessary and sufficient subprocesses for perception are – hence, this subcase is untestable at present.

The other way that this claim could be wrong is that there may be other components of visual salience besides those listed above, and these may have nothing to do with perception.[18] In other words, it may be that there is a fourth component of visual salience that would need to be computed by a non-perceptual process. Now, it is in fact clear that at least one such component exists: it is the component of salience that reflects the internal state of the cognitive system (e.g. emotional state), only one part of which is perceptual. It is such factors that I labeled as "noise" when doing the statistical analysis of the experiments with salience ratings (see [Conklin, Ehrlich, and McDonald 1983]). The question is, do such factors have a place in this model? Ultimately, they do, but in taking the two slices of cognition called "perception" and "deep generation" and studying how they are linked, which is, broadly speaking, what this thesis is about, one comes perilously close to needing to include such things as "awareness" and "attending" in the theory, and this is simply far beyond the scope of the state of the art in A.I. Hence this subcase is also untestable.

---

[18] There may also be components of salience which were not specified above but which are, when analyzed, still found to be derived from elements of the basic perception process. Such a component would actually serve to lend credence to this Hypothesis, and in any case would not upset it.

5.1.8 Descriptions are object-driven.

The eight claim is that

Hypothesis V

> Perceptual descriptions are oriented to the *objects* in the domain data base, while properties and relationships are secondary.

This claim is implemented directly in the model, in the form of the USOL (Unused Salient *Object* List) and the Current-item register. Since the USOL contains only objects, and is the sole source of data for the Current-item register, the only entities that the system focuses on describing are domain *objects*. Another way to say this is that GENARO is "object-driven".

Testing this claim.

This claim is supported by the fact that, as a strategy in the approach described here, the system uses it and produces well-formed descriptions. The claim would be discredited by a situation in which a property or a relation needed to be mentioned early, even though the property's object or the relations' objects were not salient. For example, suppose that one were describing a house scene in which the car was parked, not in the driveway, but on the peak of the roof of the house, somewhat precariously. Suppose further that both the house and the car were perfectly normal in all respects, except of course for their unusual and improbable relationship. Clearly this relationship is highly salient, and would be mentioned very early in the description. Does not this situation contradict this Hypothesis?

The answer is that for two objects to have a highly salient relationship *they must also be salient themselves.* In the example, the house, and certainly the car, would be highly salient objects. The reason for this lies in the processing of the SALIENCE system. I claim that objects derive their salience in part from their properties and relations, so that it cannot happen that a low salience object has a high salience property or relationship.[19]

---

[19] It could in fact be argued that the identification of *objects* is the primary task of the perception process -- that objects are the primary entity of perception, and that properties and relations are perceived as being "about" objects. In any case, this hypothesis does not rest on this stronger claim: even if objects are not the primary perceptual entity, they are certainly on an equal footing epistemologically with properties and relations, and that is all that is needed for this hypothesis to hold.

Since the identification of an object cannot in any way be separated from the identification of its properties and relations, any property or relation which is unexpected or intrinsically salient lends its salience to the object which it is "about". For example, a blue house is odd enough to demand a bit of extra processing to confirm the identification of the regions in question as a house, thus increasing both the salience of the house and of the property blue. However, the property blue applied to a tree could be cause for discarding the tree hypothesis altogether, and thus would make the tree itself (in the event that the tree hypothesis was still the best one available for that region) and its property blue extremely salient.

And because objects are defined to derive a portion of their visual salience from the salience of their properties and relations to other objects, it is impossible for a low salience object to have highly salient properties or relations. Basically, properties and relations are "about" objects in a way which is not reciprocated.[20] Thus, it is enough to drive rhetorical planning from the salience of the *objects* in the data base, and to have the system organized to move only the objects around within the machinery (i.e. the USOL and the Current-item register), knowing that the other classes of entities (properties and relations) will come with them.[21]

5.1.9 No feedback from surface to deep generation.

The ninth claim is that

Hypothesis VI.a

Feedback from surface to deep generation is expensive and unnecessary.

Feedback is expensive because it requires between these processes a whole other "channel" of communication -- SG (surface generation) must be able to formulate precise diagnostics about its failure in processing the received r-spec, and DG (deep generation) must be able to interpret these "messages" and use them to modify the sent r-spec to correct for the error. Feedback is unnecessary because it is cheaper to

---

[20] When this system and SALIENCE are extended to handle moving pictures and action, this theory will have to be expanded to include the preeminence of *action* in the perception process. More important than the identity of an object hurtling at one is its speed and trajectory, and the simple epistemolgy offered here would need to be extended to include such (property-like) descriptors.

[21] By the same argument, gestalts (the fourth class of perceptual entities) have a status more like that of objects: they are not *about* any specific object, although they are based on many things in the scene at once (e.g. time of year, time of day, picture focus, etc.). Gestalts should thus be included in the USOL, and be described as the Current-item according to their salience. See Chapter 6.

allow the system to fail occasionally – people do (in the form of production errors – see [Garrett 1975]). Furthermore, people are able to detect that the utterance they are producing or have just produced is ambiguous, yet the generation process is a poor place to have to do this[22] (except of course for constructions that are canned and which can therefore be marked in the grammar as potentially ambiguous). That is, one can imagine a production system in which the realization component was able to detect ambiguity in its output and to signal this condition to the planning component. However, in a full language system capable of input and output, concurrent parsing of the text being produced offers an inexpensive way to detect ambiguities in the output material, especially under the assumption that the parser has "nothing else to do" when the generator is running.

Testing this hypothesis.

What would qualify as a disproof of this claim? Feedback might be required if MUMBLE (the SG) needed some information that was not provided in the r-spec. However, such situations could also be remedied by having GENARO (the DG) anticipate the questions and provide the answer in advance. It is possible that such extra work on the part of GENARO would be expensive, or that the r-spec would grow unmanageably large with information anticipating all such questions. But neither of these conditions "proves" the need for feedback.

The other possibility is that MUMBLE might need to signal that it has been sent down a "garden path" by GENARO's r-spec, and that it needs GENARO to replan it (avoiding the part of the specification which caused the problem). This would be a very different kind of feedback, however, because by the time MUMBLE has detected that it has taken a dead-end path, it is too late. Some of the surface structure already completed would have to be thrown away to take advantage of the new "corrected" r-spec from GENARO, but MUMBLE's processing is left-to-right and *indelible*. This kind of "feedback" is indeed necessary for a fully functioning generation system, in order for it to detect and correct its own errors. But at this level the feedback could as well come from a parallel on-line parsing process – it is not the kind of fine-tuning feedback from SG to which this claim is meant to refer.

---

[22] Generally speaking, an ambiguous structure is one to which more than one translation of an *input* can be given. It is natural (almost unavoidable) to detect ambiguity during the process of analyzing (parsing) speech or text – it is signaled (in the simplest case) by having more than one grammar rule which could apply. Likewise, production is sensitive to ambiguity in its (logical/semantic) input. However, in neither process is it natural to detect ambiguities in the process's *output*, because the rules are not (without explicitly adding it) "looking forward" at the structure that they are creating.

Thus this claim is almost impossible to disprove, since it rests heavily on the design of the DG and SG components. In general any specific failure or awkwardness of the system which appeared to require feedback from SG to DG could be remedied by either giving the SG better facility for handling the problem or, more likely to succeed, altering the DG so that it did not leave the resolution of the ambiguity unspecified.

The claim about the expense of feedback is likewise difficult to weigh -- which is more expensive, signalling specific errors during generation, or designing the system to avoid most of them and putting up with the failures that remain? This is an empirical question that will have to be answered through experience with using the system.

5.1.10 One r-spec per sentence.

The tenth claim is that

Hypothesis VI.b

> It is adequate to the interface between deep and surface generation to have the basic unit of planning at the deep level correspond to a single sentence at the surface level.

This is not to say that the r-spec must be complete before MUMBLE starts to realize it -- MUMBLE could have started on the realization of an r-spec before GENARO has finished with it.[23] Rather, the basic "packet" of information at the rhetorical level is the r-spec, and making the assumption that it will get realized as a single sentence greatly simplifies the design of the rhetorical rules and their dictionary entry counterparts in MUMBLE.

This claim specifically prevents designing MUMBLE's dictionary and/or grammar so that it was able to successfully realize in two sentences an r-spec that was too big for one. Such flexibility within MUMBLE, if it were possible, would undermine the principle of locality which lead to the failure described above (hypothesis III, page 117). That is, this claim simply states a restriction necessary to keep the GENARO/MUMBLE system interestingly weak. (Note that it also prevents MUMBLE from accumulating more than one r-spec before starting to realize them as a compound r-spec.)

---

[23] There are, however, serious ramifications for where in the realization the latter parts of the r-spec can be put.

5.1.11 Summary.

In this section GENARO was presented as an initial theory of human language generation. That is, it was claimed that the external behavior of the system approximated that of humans generating text, and further, some claims were made about the explanatory adequacy of the internal mechanisms of the system. The following list presents a short "scorecard" summary of the results of this chapter. (The underlined parts listed here are merely short phrases meant to be suggestive of the full claims -- they are *not* the claims themselves.)

1. *Descriptions require salience*: well established; almost a logical necessity.

2. *Salience is the primary strategy*: well established, though the relation-based strategy runs a close second.

3. *Stepping down the USOL is sufficient*: probably false, but can be fixed with simple extension.

4. *Rhetorical structure is not recursive*: no simple conclusion.

5. *Iterative proposing is necessary and sufficient*: well established; depends on judgements of the quality of the system's output.

6. *Rhetorical planning can be done indelibly*: no simple conclusion.

7. *Salience is perceptual*: true.

8. *Descriptions are object-driven*: well-established.

9. *No feedback from surface to deep generation*: a useful constraint; fairly well established.

10. *One r-spec per sentence*: a useful constraint; quite implementation dependent.

5.2 GENARO as a tool for linguistic research

In this section I present GENARO in another light: as a tool for investigating rhetorical and stylistic conventions.

For example, the notion that one should "say what is relevant or salient" is clearly a maxim of everyday conversation. If this system only offered evidence for the efficacy of this rhetorical maxim in producing scene descriptions it would not be a significant contribution. Part of the value of the system is that it provides a computational framework in which one may study rhetorical and stylistic conventions in a more detailed and specific way.

The domain of style and rhetoric has not yielded easily to linguistic investigation; this has been due in part to the lack of any technical language or precise paradigm in which potential rhetorical facts could be suggested and tested. Grice commented on this in his famous paper on conversational maxims [Grice, 1975]. He differentiates his "Cooperative Principle", that participants in a dialogue cooperate to move the dialogue forward according to certain rules, into submaxims falling into four categories: Quantity, Quality, Relation, and Manner. After discussing some submaxims in the first two categories, he comments on the category of Relation:

> Under the category of *Relation* I place a single maxim, namely, "Be relevant." Though the maxim itself is terse, its formulation conceals a number of problems which exercise me a good deal; questions about what different kinds and foci of relevance there may be, how these shift in the course of talk exchange [sic], how to allow for the fact that subjects of conversation are legitimately changed, and so on. I find the treatment of such questions exceedingly difficult, and I hope to revert to them in a later lecture.

Although salience and relevance are distinct, they have enough in common that Grice's comments can be taken to illustrate the complexity and difficulty confronting a non-computational approach to the study of the rhetorical issues of what to say and how to say it.

In this section I will examine a single rhetorical convention and the process of writing a rule which captures it.

5.2.1 Reifying object clusters.

This section discusses the problem of identifying a set of objects which should be treated rhetorically as a single object. I use the term "reification" for this process, to emphasize that a new "object" has been brought into existence, if abstractly. The

general problem was also discussed in Chapter 4 (page 92) -- the following two paragraphs are repeated from that discussion.

If several trees are visible in the front yard in the picture, they may be best described as a single entity: the "trees in the front yard". Other examples are the "clouds in the sky", the "path to the front door" (which consists of separate stones or tiles laid roughly in a row), and the "bikes in the yard" (where there are two bicycles lying in front of a house).

The fundamental issue in each case is whether the clustering itself provides an important key to the visual identification of the objects or the scene as a whole. That is, if the process of doing the visual analysis would be well-served by knowing about the possibility of a particular kind of clustering (as is certainly the case for the "clouds in the sky"), then that concept should be in the world knowledge of the vision system, and would be represented as a single entity in its interpretation of the scene. On the other hand, there are certainly cases where objects are mentioned together in a description for rhetorical reasons, and not because they form a perceptual entity. The "bikes in the yard" is probably such an example, especially if they are lying at opposite ends of the yard in the picture.

In this section the key issue is the design of a "$condense-reify" rule for GENARO. That is, what would be involved in writing the rule that captured the rhetorically-motivated reifications in scene descriptions. Here are the key issues involved:

1. Some reifications seem to be perceptually based, others are generated rhetorically, and some fall in a gray area in between.[24] How can this be accounted for?

2. Several instances of the same kind of object in a scene do not always lead to their being described as an object cluster. What are the criteria by which reification occurs?

3. How are these perceptual and rhetorical object clusters represented in the domain data base?

4. In what sense is reification a condensation of the description of several parallel objects?

Each of these questions is discussed below.

---

[24] A related problem is the linguistic distinction between count nouns and mass nouns, but this has more to do with how a collective concept is lexicalized.

Perceptual vs. rhetorical clusters.

Here are some other examples of object clusters that might be mentioned in a typical house scene:

1. the trees in the yard
2. the leaves in front of the fence
3. a flock of birds
4. the tools by the car
5. the cars in the driveway
6. the UFO's in the scene

Here are some collections of objects which would probably not be clustered (unless there were some unusual shared aspect to all of the objects):

7. the bushes in the picture
8. the architectural structures in the scene
   (e.g. the house and garage)
9. the leaves on the left half of the tree
10. some houses (where there is one large and central house in the scene and another one just visible in the distance)

Note that some context is imaginable for each of the clusters in items 1. through 6. that would make it a bad cluster (e.g. the trees are too distant and widely separated to be clustered). Likewise, one can imagine circumstances which could cause any of 7. through 10. to be considered a good cluster. Thus context and/or the purpose or goals of the viewer have a considerable impact on the reification process. However, this will be dealt with minimally below – the context that pictures function to show or tell something[25] will be considered adequate to allow a meaningful discussion.

The first list above omitted one class of objects which are *always* "clustered": the parts that make up an object! The parts are themselves objects (which are themselves composed of objects), so that the structural subparts of an object are trivially clustered together -- into the object itself. A fence is a collection of pickets and runners, structurally connected. This is reification in the purest sense. In fact, a fence would never be described as "a group of pickets connected together", because it would have existed in the perceptual representation as an object on *perceptual* grounds, and thus clearly would be available for free to the generation system.

---

[25] See the discussion on the effects of context on salience, Section 4.1.2, "The role of context".

A more interesting case of perceptual clustering is exemplified by clouds in a partially cloudy sky. Here the clouds are not structurally connected, although it seems reasonable that they would be clustered perceptually. That is, in addition to each cloud being represented as an object concept, the collection of clouds would be represented as the object concept "the-clouds-in-the-sky". Likewise, the "flock of birds", the "cars in the driveway" and the "UFO's in the scene" all are tightly enough bound together within the cluster that it is reasonable to argue that they are *perceived* as separate objects *and* as a collective entity.

In another case, several flat stones lying unconnected on the ground are a perceptual object (a "path") if they are linearly arranged. In this case, I claim, a *functional* feature ties these objects into a cluster object: the stones are used to walk on. Such objects would probably also be computed in the perceptual representation, although the case for this claim is weaker, since function is a very high-level aspect of world knowledge.

On the other hand item 1., "the trees in the yard", might only be an object for rhetorical reasons: the trees can be widely separated, of different varieties, sharing only proximity to some house, or containment in a fence. In such cases it would be dubious to claim that they are *perceived* as a group. However, when describing the scene we have a choice between not mentioning them, enumerating them, or alluding to them as a group. Especially with the goal of brief description, things which do not have common groupings and corresponding group names -- that is, which are not perceptual objects -- may be lumped together as a *rhetorical* expedient. The key requirement for clustering objects, be it on rhetorical or perceptual grounds, is that they share the same description to some extent.

The criteria for clustering.

Object clusters thus seem to exist on a continuum from structurally-based to rhetorically-based. However, I propose that the criteria for reification are similar throughout this continuum. Clearly some collections do not get reified. Why not?

The list of good and bad clusters above suggests some criteria:

1. Widely separated objects are harder to cluster, whereas those in close *proximity* are easier.

2. Likewise, objects which share some kind of *containment*, as in by a fence, a yard, or the sky, are more easily clustered.

3. The objects must be very *similar*: two bicycles will cluster more easily than a bicycle and a tricycle, or even a good and a broken bicycle. This is especially true for rhetorical clusters, since the point is that they share the same description.

4. They should be of roughly *equal salience*: two houses are less likely to be clustered if one of them is on fire.

5. Finally, the *number* of objects is important: the more there are, the easier it is for them to be clustered. If there are only two, not much economy is gained by clustering them (at least rhetorically).

These criteria (and there may be others) are fairly orthogonal, so that the decision to cluster can be thought of as based on a boolean function which has at least five input arguments.

Representing object clusters.

KL-ONE offers a natural way to represent clustered objects.[26] (The following discussion is specifically with respect to *perceptually-based* clusters, though it will be claimed later that all clusters are represented the same way.) Whether or not they are clustered, a group of objects of the same type will be represented as a set of individuated object concepts sharing the same generic concept. Figure 48, part A, shows a segment of a KL-ONE network representing three trees which have not been clustered. As part B shows, clustering is represented by adding a new concept to the network. It has a single role for its members, and the fillers of that role are the objects that compose the object cluster. In this scene the two oak trees have been clustered.

Note that, like all other concepts, the cluster object concept has a salience value. This is natural: this additional object is a perceptual entity, and has the same sources of salience (size and location, intrinsic salience, and unexpectedness) as any other objects. Furthermore, the salience value can serve to position the cluster object within the USOL, so that it gets described in accordance with its salience.

$condense-reify.

To account for rhetorically-based clustering within GENARO's framework will require a rule which condenses objects of the same type. Its operation would be similar to $condense-prop: when a new object became the Current-item $condense-reify would look down the USOL[27] checking each object to see if it was the same kind of object as the Current-item. This could be done by simply checking

---

[26] For a review of KL-ONE syntax, see page 70.

[27] It might actually be more effective for this rule to look directly into the domain data base, depending on whether or not *all* of the objects in the representation are included on the USOL or not.

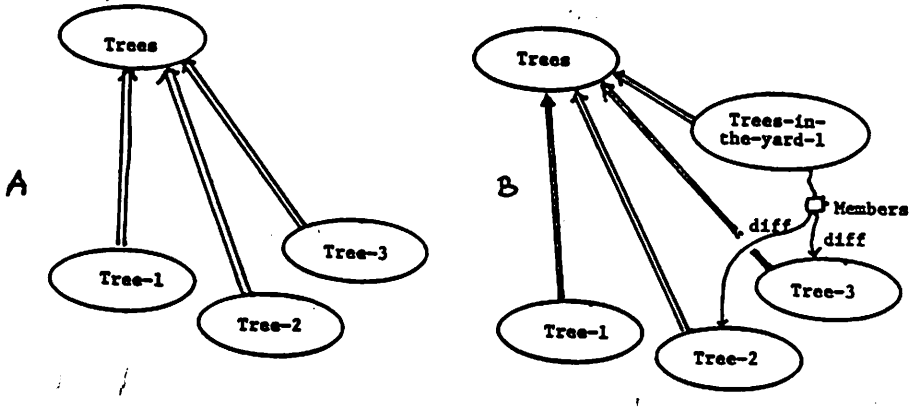


Figure 48: Clustering objects in KL-ONE.

The "diff" arcs *differentiate* the "member" node of the cluster object into its various fillers.

---

if the USOL object shared an immediate superconcept with the Current-item. All such objects would be gathered together in a local buffer within the rule and checked to see which ones, if any, were "clusterable" with the Current-item. The criteria for clustering were listed above, and these would be captured in a function which measured "clusterability" among objects. Let us call this function "cluster?", and let us further suppose that this function determines the salience of the cluster object it creates.

The intriguing aspect of this rule is that it could create the cluster object in either of two ways: as a themeobj (as all other rhetorical rules operate), or *as a perceptual object*, in the same manner as described immediately above. That is, there is nothing wrong, in principle, with allowing a rhetorical rule to make changes in the domain data base. Such actions have been avoided to this point, since they reduce the independence of GENARO from its input data base, but this is largely an aesthetic matter. And the advantage is that doing so makes two interesting claims:

1. That *all* cluster objects are represented in the same way, regardless of their source; and

2. That *perceptual objects can be created as a result of the language process* – describing several objects as being clustered results in seeing them as a cluster.

The second claim accords nicely with the intuitive observation that perception, even of a static picture, is not a static representation, and that it responds to other cognitive processes.

The action performed by $condense-prop, if its proposal won, was to remove an object from the USOL and make it the Current-item, setting it up to be described next. I am proposing here that $condense-reify actually creates the object (as described above) in the KL-ONE data base, and that it removes from the USOL all of the objects subsumed by the new cluster object, and that it makes the cluster object the Current-item.

One aspect of this mode of reifying object clusters is that rhetorically-based cluster objects should replace their most salient member object on the USOL, and thus be treated as if their salience were equal to this most salient member, regardless of the salience which is actually assigned by the function (condense?). It might be objected that this is undesirable: it is possible that the salience of the cluster object will be considerably greater than that of any of its members. In a city street scene, for example, in which all the men are wearing bowlers (derby hats), each bowler will be low in salience, but collectively they might be unexpected enough (at least in the U.S.) that their cluster object would have a high salience.

It is debatable, however, that in such situations the cluster object would be rhetorically-based. Is it not as likely that such an unusual situation (e.g. the bowler example) would be *perceptually* odd (and salient) enough to have triggered clustering during the perceptual analysis? If so, then the mode of operation of $condense-reify makes an additional claim: that there will only be small gaps between the salience of an object cluster and its most salient member, since situations leading to large gaps will have already been clustered perceptually. And since this salience gap is (at most) small, there is no loss of performance in mentioning the rhetorically-based cluster object at the point in the description when its most salient member would have been mentioned.

Summary of reification.

This section has illustrated using GENARO to do essentially cognitive science research. Starting with an observed rhetorical phenomenon (object clustering), we studied when it happened, the way it would be represented, and how this representation would be manipulated by a rhetorical rule which captured the phenomenon. A rule with some rather surprising properties emerged, and its operation made it possible to speculate, in a general way, about the relationship between perception and language.

5.3 Summary

This chapter has covered the most theoretical aspects of the design and use of GENARO. The first section elaborated the various claims that are embedded in the design of the program (though it was seen that some of these claims are weak or untestable). The second section provided an detailed example of the use of GENARO (and the framework in which it operates) as a tool for doing cognitive research.

CHAPTER VI

CONCLUSIONS

This thesis describes a body of cognitive science research which featured (1) psychological experiments studying the perceptual phenomenon of salience, and (2) construction of an A.I. program which demonstrated that a very fast and very localized kind of rhetorical planning was possible using salience as a heuristic. This chapter reviews the features and limitations of this program (section 6.1), as well as examining some exciting extensions which could be made to it (section 6.2).

6.1 Features and Limitations

6.1.1 Salience.

Since the basis of the rhetorical planner GENARO is the existence in its input data base of a salience annotation, it is important to determine how broadly applicable the notion of salience is. What kinds of processing besides perceptual analysis are likely to compute salience in their data base?[1] Is the aspect of internal model building crucial, or are there other processing goals which have salience (or its components) as a byproduct? Would it be useful to hand-code salience values into an otherwise hand-coded data base?

The key feature of salience that makes it promising for other applications is that it is *a simple numerical encoding of the relative importances of different facts in a data base*. Whether it is computed by the "expert system" using the data base or by a separate process which knows about user goals, and even if it is coded into the data base by hand, it has the effect of providing another dimension -- the dimension of salience -- to a data base. Metaphorically speaking, if all of the facts in a data base are of equal value the data base can be thought of as *flat*. Adding salience gives the data *depth*: instead of having to work with the entire data base at one time, a user or a system can view it in "slices" of diminishing salience. Clearly this has application beyond generating descriptions of the data, since in any large data base (that is not inherently "flat") queries, reasoning, and search could all take advantage of this dimension.

---

[1] I take if for granted that any computer system doing perceptual analysis (in any perceptual modality) will be computing the elements of salience. For example, I claim that a computer vision system capable of perceiving objects and actions in a moving picture will also be computing salience.

Another advantage of salience is that it implicitly encodes a measure of "obviousness". One of the most important things to know about the relationship between your listener and what you have to tell them is what is obvious to them, so that you can avoid saying those things [Grice 1975]. As Mann et al. [1981] point out, this information is best thought of as part of a "model of the reader", i.e. a data structure which represents at least what is obvious to the reader and what they have already been told, and perhaps also what the reader believes and what their current attention is on. In GENARO the latter two factors are ignored. However, thematic-objects which have been mentioned in some way in an r-spec are marked as "mentioned", thus keeping track of what the reader has been told. On the assumption that what is obvious to the reader is also obvious to the speaker, this system also (indirectly) encodes the obviousness of its visual information -- as salience. That is, whatever it is that might make an object salient -- that is was unexpected, or large and central, or a rare or interesting object in itself -- would also make it unobvious to a reader who could not also see the picture being described. In short, GENARO has a simple model of its audience.

The notion of salience will prove to be a weak heuristic, however, in data bases in which all facts are equally salient. For example, if none of the major components of visual salience (i.e. size and centrality, unexpectedness, and world-knowledge-based) have analogs in the domain covered by the data base, I suspect that that domain is inherently flat (with respect to salience). Also, salience will be of limited value where the system is highly sensitive to context and the computation of those components of salience which are responsive to that context is expensive.

6.1.2 GENARO.

The other important contribution of this thesis is the demonstration that text planning (i.e. deep generation) can proceed in a limited form without the computational expense of weighing alternative plans, reasoning backwards from desired rhetorical goals, or devising a maximally dense parcel which achieves many rhetorical goals. Certainly people are capable of these things, but there are domains of discourse in which even people forego complex planning. Furthermore, if we are to provide computers with real-time language generation abilities we will have to settle (temporarily) for less than Shakespearean prose.

GENARO is able to rapidly select what to talk about and package it according to some simple rhetorical conventions. The resulting text (via MUMBLE) is of good quality and, because the system is highly data-driven, closely resembles in structure and content the input data. Because of its speed the system can be made to accomodate a dynamic data base -- one in which the facts are changing during the

generation process.[2] Also, the program is designed to support development and refinement of the rhetorical conventions it uses.

For what applications is GENARO ill-suited? If no salience annotation is available in the input data base, the salience-driven approach is certainly at a loss. Furthermore, to the extent that the salience annotation leaves large "slices", i.e. large subsets of the data base with the same salience values, the additional power (and cost) of the traditional planning mechanisms will be needed.[3] Some possible applications of the system are discussed below in section 6.3.

6.2 Some immediate extensions to the system

Although GENARO is by design a weak and localized planning device, there are some shortcomings that were described in this thesis which had little or no theoretical merit. In this section some of these shortcomings are examined, along with the specific changes that are being implemented concurrent with this writing.

6.2.1 Ending r-spec construction.

Although most of the production rules in GENARO operate by adding elements to the r-spec, there are a few (e.g. $newitem) which accomplish rhetorical effects via performing a control action. One control action which is a candidate for such encoding is the process of determining when an r-spec is the right size (or weight) to be sent to MUMBLE. The current mechanism is a function in the control structure which monitors the weight of the r-spec and signals the proposing loop to stop when that weight passes a certain threshold (see page 52 for details). Since the size of the r-spec is a rhetorical parameter, however, it would be better if that parameter were controlled by a rule. In the near future I will be implementing such a rule, $finish-building-rspec, which will have as its action signalling the proposing loop to stop.[4] The intriguing aspect of this rule is that it can make its proposal at a priority which is a function of the weight of the r-spec: the bigger the r-spec gets, the

---

[2] For example, the output of a computer vision system which was analyzing a dynamic scene, e.g. a movie.

[3] In such circumstances GENARO is forced to make arbitrary choices among the equal salience items. It remains to be seen how often this leads to poorly structured text -- the model makes the implicit psychological claim that the order of mention among equal salience items is non-critical.

[4] Use of this rule in an actual run of GENARO is shown in Appendix 2.

higher the priority at which this rule is proposing to "finish building the rspec".

This will be a more flexible scheme for controlling r-spec size. It will allow the rule body to build large r-specs when there are many salient things to say about an item; likewise, when there is little to say, $finish-building-rspec's proposal will win easily while the r-spec is still small.

6.2.2 Less salient properties and relations.

The reader may have noted that, in the very simple set of rhetorical rules described here, there was no mechanism for a salient property or relationship of the Current-item to be expressed unless it happened to be the *most salient* property or relation. The Current-properties and Current-relations subregisters were ordered in order of decreasing salience, but the rules $prop-salience and $reln-salience only knew to propose the top item on each of these lists. Furthermore, the lists did not change during the construction of an r-spec for a Current-item.

The next enhancement is to modify these subregisters so that they behave in the same manner as the USOL: when the top (most salient) item is mentioned, remove it from the list, exposing the next most salient item as the top.[5] In this way, $prop-salience and $reln-salience would continue to propose properties and relations of the Current-item as long as the top item continued to be salient enough. Possibly these rules would also need to be modified so that the priority at which they posted their proposals was a function of the salience of the item which they were proposing.

6.2.3 The Current-item as a stack.

In Chapter 5 Hypothesis I.b (page 109) asserted that objects only appeared as the Current-item once (and that this was adequate). Hypothesis I.c (page hypic_pn) went on to specifically assert that descriptive paragraphs are not generated by a stack mechanism. Nonetheless, it is interesting to speculate on the advantages and problems with providing a stack for the machinery of GENARO.

There are two immediate modifications which this additional power would allow.

- Two objects could be described in the same r-spec, since the first Current-item object could -- by being pushed "into" the stack -- be "set aside" temporarily, while the second object was the Current-item. Presumably the stack would be popped at the end of the r-spec to restore the first (and more salient) object as the Current-item.

---

[5] Use of this convention is shown in an actual GENARO run in Appendix 2.

- On a larger scale, objects could be stacked up indefinitely in the Current-item stack, spanning many r-specs. This style of use of the stack would allow recursively structured texts.

In general, the problem with this scheme is its power: pushing items into a stack is a nice way to set them aside temporarily, but the decision of when to *pop* the stack presents major complications which the simpler machinery completely avoids. In the first option above the problem is what to do with the temporary Current-item object when it gets popped. Should it go back into the USOL, to be made the Current-item again later, or should it be thrown away (as is usual with Current-item objects)?

The second option presents this same difficulty on a larger scale. It establishes two alternative actions that might be taken when a Current-item is thoroughly described: taking the next object from the USOL, or popping the Current-item stack. The proper use of this additional alternative is an interesting application for future research with the system.

6.2.4 Unrelated objects in one r-spec.

A related proposal is the simple modification of allowing the rule $newitem to get a new Current-item object during the construction of a single r-spec.[6] Currently $newitem only runs when the r-spec is empty, so that it serves the function of having each new r-spec start with a fresh Current-item. By allowing this rule to run whenever there were not much to say about a Current-item object, r-specs would get constructed which described two (probably) unrelated objects, resulting in sentences like "There is a driveway next to the house, and a mailbox in the foreground." Curiously, such constructions were fairly common in the experimental written descriptions. This simple proposal makes a claim about how such sentences come to be: the speaker finds they have said everything that is salient about one object, but feels that the sentence is still too short, so they "tack on" a short description of another object.

Actually, the proposal being made here makes a stronger claim, since in GENARO the second object is chosen from the USOL, and hence only on the basis of its salience. That the second object's description should also be brief is a kind of planning that GENARO does not currently do. However, since the thematic-objects on the USOL already have their Current-properties and Current-relations lists compiled, it would be inexpensive to extend the predicates available to the rhetorical rules to include a heuristic estimate of the r-spec weight that such USOL objects will require to be described.

---

[6] This proposal was also discussed in Chapter 3 on page 62.

6.2.5 Gestalts.

In Chapter 4 it was claimed that there were four classes of entities necessary to describe static scenes: objects, properties, relations, and gestalts. The last category, gestalts, was defined to cover "concepts which express complex relationships or properties among many of the domain concepts", for example, the landscaping, or the season of the year. Not much use was made of this class in the current implementation of GENARO, however, because I felt that the rhetorical phenomena that it accounted for was secondary to those of the other classes. In experimenting with the system, however, I discovered two specific applications of gestalts which allowed substantial improvements to the system.

The first application dealt with the way in which such topics as the season of the year (e.g. summer, winter, etc.), the time of day (e.g. day, night, twilight, etc), and the weather (e.g. sunny, cloudy, snowing, etc) were brought into the description. In the current implementation these items are properties of the scene-level schema, i.e. they are attributes of the top-level schema which organizes information about the whole scene (cf. page 64 in Chapter 3). And they are introduced into the description by rules in the Conclude packet, although only one rule, $light, has been used to date. This scheme has the disadvantage that it only allows these gestalts to be mentioned in the conclusion of the description.[7] If a gestalt were particularly salient (e.g. a winter scene in which the falling snow is visible), however, then it should be mentioned at a point in the paragraph consistent with its relative salience in the scene.

There is a simple mechanism for doing this: consider gestalts to have the same status as objects and sort them into the USOL. In this way gestalts would get described via being the Current-item exactly as objects do, at a point that reflected their salience in the picture. The observation that such items as lighting and time of the year are mentioned late in descriptions would be accounted for by according them a low salience in the visual representation, so that only if there were something unusual or unexpected about them would they be mentioned before the conclusion.

6.2.6 Exploiting the Paragraph Driver.

In GENARO the packets for the rhetorical rules are controlled by the "Paragraph Driver". This simple device simply performs some simple checks (e.g. Is this the very first thing being said?) at the beginning of each round of proposing and, based on the results, turns packets on or off.[8] Its purpose is to provide global

---

[7] This scheme was originally prompted by the informal observation that many subjects mentioned the season and/or the lighting at the *end* of their descriptions (if they mentioned them at all).

[8] Recall that there are four packets: Introduce, Shift-topic, Elaborate, and Conclude.

"shape" to the structure of a paragraph. It turned out, however, that scene descriptions have very little structure[9] besides the one generated locally by applying the salience-based and relationship-based rules (see the discussion of Hypothesis I.a in Chapter 5).

However, it is likely that paragraphs in other domains and multi-paragraph texts will have more complex global structures, and will require more than a purely local mechanism to generate them. For example, in her work on describing facts in a data base, McKeown [1982] found that there were several paragraph-level *schemas* which accounted for rhetorical structure over a large range of expository texts. These schemas were expressed as context free grammars, the primitives of which were descriptors such as "attributive", "amplification", and "restriction". Each descriptor specified the rhetorical function of a sentence or clause in the text as a generic speech act. A legal "sentence" in this paragraph grammar, then, was a description of a legal rhetorical structure for an expository paragraph.

If each descriptor in this more complex domain (e.g. general expository text) corresponded to a packet of GENARO-style rules, and the Paragraph Driver in GENARO were equipped with McKeown's "schema" grammar for a particular type of expository text (e.g. "identification"), then it quite possible that GENARO's machinery would support the generation of paragraphs with rich internal structure. This is an interesting experiment, and is the natural direction to take with the extension of the system to other domains, but the point here is simply that there is nothing in the design of GENARO which is *inconsistent* with this kind of high-level structure, as long as it does not require lookahead or backtracking to implement.[10] However, whether GENARO's machinery can actually support this kind of global planning, without radical change, is a issue that can be resolved only by trying it.

It is also interesting to note that the above extensions to the Paragraph Driver would involve adding at least a register for keeping track of the paragraph-level "Current-topic" (as well as mechanisms for accessing and changing it). The result architecture would have two primary registers driving the rules: the Current-topic and the Current-item. This accords nicely with the findings of Foss [1982], who showed that these two kinds of items were the two that were primed for in subjects reading of descriptive text. It has been pointed out above that there were several clear advantages to developing this system in the domain of generating scene descriptions,

---

[9] More precisely, the only global structure needed in scene description paragraphs is that the first sentence be introductory. No additional structure is needed for producing good quality descriptions.

[10] Such high-level structure is, however, inconsistent with the *spirit* of GENARO, which is to investigate the limits of an extremely localized and myopic style of planning, including having no representation, implicitly or explicitly, of a global structure or pattern for the text being generated.

including the naturalness of the salience annotation in the visual representation and the structural simplicity of the descriptive texts. The contribution of this work as an A.I. system, however, will be measured in terms of the range of applications in which it can serve as the generator part of a natural language interface.

APPENDIX A

LISTING OF GENARO TOP LEVEL

```
(defun GENARO nil
   (comment (top level routine of the planning system.))
   (init-system)
   (make-description))


(defun INIT-SYSTEM nil
   (comment (initializes all of the various global variables.
            called by genaro.))
   (get-envmt-file)
   (terpri)
   (writeln "This run is on "
            envmt-file
            ".  "
            (time)
            "  "
            (date))
   (writeln "Comments (if any) on this run:")
   (print-run-notes (setq run-notes (get-run-notes-from-user)))
   (cleanup-themeobjs)
   (comment (registers:))
   (init-parameters)
   (init-*rule-list *lesion)
   (setq usol          (make-usol)
         msg           nil
         msg-list      nil
         on-packets-list
                       (makeset '(intro shift-topic elaborate)))
   (set-cur-item (pop-usol))
   (setq main-item cur-item)
   (comment (counters:))
   (setq msg-count 0)
   (setplist '*counters nil)
   (comment (flags:))
   (setq *slow-run   t
         *again      nil
         *descr-done
                     nil
```

```
            *pop-props&relns
                        t))


(defun MAKE-DESCRIPTION nil
    (comment (repeatedly builds r-specs and sends them to
              MUMBLE.  called by genaro.))
    (repeat nil
              (init-for-msg)
              (build-msg)
              (mumble-msg)
     until  (end-descr?)))


(defun BUILD-MSG nil
    (comment (builds the msg by repeatedly proposing a set
              of msgelmts and picking the best one. called by
              make-description.))
    (init-build-msg)
    (repeat nil
              (init-proposing)
              (propose-msgelmts)
              (setq msgelmt (pick-best-msgelmt))
              (insert-in-msg msgelmt)
              (mark-mentioned msgelmt t)
     until  (msg-complete?)))


(defun INIT-PROPOSING nil
    (comment (initializes for a round of msgelmt proposing.
              clears the proposed-msgelmt-list from the last
              time. called by build-msg))
    (setq proposed-msgelmt-list nil)
    (if (null *quiet-mode) (writeln "**********************"))
    (packet-switching)
    (if (null *quiet-mode)
        (writeln)
        (writeln "Beginning proposing ...")))


(defun PROPOSE-MSGELMTS nil
    (comment (after checking the paragraph-level script this
              routine sets which rule packets should be on
              accordingly and then gives all of the rules a
              chance to fire. called by build-msg.))
    (mapcar *rule-list try-rule))
```

```
(defun PICK-BEST-MSGELMT nil
   (comment (since proposed-msgelmt-list is built to be in
            decreasing order the only thing to worry about is
            if there are more than one with highest priority.
            for now I'll just take the first one that was
            proposed. called by build-msg.))
   (car proposed-msgelmt-list))


(defun INSERT-IN-MSG (msgelmt)
   (comment (if the value of the themeobj that *msgelmt* points to
            is a lambda-expr it gets run. if not the themeobj gets
            added at the end of msg. called by build-msg.))
   (if (not
         (eq (setq *again (show-state
                           "Just before inserting the msgelmt ..."))
             'again))
       (let (temp      (eval (eval msgelmt))
             themeobj (eval msgelmt))
         (cond ((null themeobj))
               ((and (listp temp) (eq (car temp) 'lambda))
                (apply temp nil))
               (t (ncondconc msg (list msgelmt)))))))


(defun MSG-COMPLETE? nil
   (comment (decides when to stop building this message and send
            it to mumble. if the *again flag is set then just go
            around again without having "executed" or inserted
            any msgelmts. if *finish-rspec is set then
            $finish-building-rspec has won.))
   (comment (called by build-msg.))
   (cond ((eq *again 'again) nil) (t *finish-rspec)))


(defun MUMBLE-MSG nil
   (comment (records the msg. prints the message on the console
            & waits for operator approval if the *slow-run flag is
            on. will eventually actually start mumble as another
            process and send it the message as a message! called
            by make-description.))
   (if msg
       (setq msg-count (add1:i msg-count))
       (let (cur-msg (get 'msg 'cur-msg))
         (set cur-msg msg)
         (ncondconc msg-list (list cur-msg))
```

```
            (break:jc 'mumble-msg)
            (cond (*quiet-mode (show-message-brief cur-msg))
                  (t (terpri)
                     (show-message cur-msg)
                     (princ
"************************************************************")
                     (terpri))))))


(defun END-DESCR? nil
   (comment (decides when to end the description. based on length
            of description and salience of the last cur-item.
            called by make-description))
   (cond ((or *descr-done (and (null msg) (talked-out?))) t)
         (t (break:jc 'end-descr?))))


(defun TALKED-OUT? nil
   (comment (true if the message and the usol are both empty.
            called by packet-switching.))
   (and (null usol) (null cur-item)))
```

APPENDIX B

GENARO OUTPUT

THE FOLLOWING TEXT IS LITERAL OUTPUT FROM THE GENARO PROGRAM. SUBSEQUENT ANNOTATION OF THIS OUTPUT APPEARS IN UPPER CASE.

This run is on world43. 18:19:33.93 9-APR-1983

"WORLD43" IS THE REPRESENTATION OF THE WINTER HOUSE SCENE, AS SHOWN IN FIGURE 1.

Comments (if any) on this run:
(an empty line ends the comment)
(* uses the previously typed comment)

This is a base-line run with normal default settings for
the system parameters WORLD43 is the representation of the
winter house scene shown in Figure 1.

Initially the usol is:
(house-1 fence-1 door-1 gate-1 driveway-1 mailbox-1 porch-1 road-1
porch-2 columns-1 sidewalk-1 columns-2 bush-1 bush-2 roof-1 tree-1
windows-1 yard-1)

Cur-item just got reset to $house-1. THE TOP (MOST SALIENT) USOL ITEM IS POPPED INTO "CUR-ITEM".
The USOL now is: (fence-1 door-1 gate-1 driveway-1 mailbox-1 porch-1
road-1 porch-2 columns-1 sidewalk-1 columns-2 bush-1 bush-2 roof-1
tree-1 windows-1 yard-1)
****

THE FIRST ROUND OF PROPOSING – EACH RULE
Beginning proposing ... THAT IS IN AN ACTIVE PACKET IS "TRYED".
trying $prop-color ...
-> in $prop-color, need (lessp:r msg-wt [= 0.0] cut-off [= 3.04])
trying $condense-prop ...
trying $prop-salience ... THE ABOVE EXPRESSION SHOWS THE PARTICULARS
trying $prop-sal-obj ... OF THE BOOLEAN FUNCTION WHICH DECIDES
trying $reln-salience ... IF THE RULE $PROP-COLOR SHOULD RUN.
trying $light ... (MSG-WT AND CUT-OFF ARE VARIABLES)
trying $intro ...
trying $newitem ...
trying $finish-building-rspec ...

```
Just before inserting the msgelmt ...
      Cur-item and friends: $house-1
      ($2-story-building-1 $white-1 $new-england-house-1)
      ($in-front-of-1 $next-to-3 $next-to-2 $next-to-4)

       THE CURRENT ITEM, THE CURRENT-PROPERTIES LIST, AND
        THE CURRENT-RELATIONS LIST ARE SHOWN FOR EACH ROUND.

      The msg so far: nil

      The proposed-msgelmt-list (no. 1 goes to the msg):
      1      $$introduce-1-1              2.00     $intro
      2      $$in-front-of-1-1            0.57     $reln-salience
      3      $$2-story-building-1-1       0.56     $prop-salience
      4      $$newcuritem-1-1             0.40     $newitem
      5      $$2-story-building-1-2       0.38     $prop-sal-obj
      6      $$white-1-1                  0.20     $prop-color
**********

      THE RULE $INTRO WINS THIS FIRST ROUND. (THIS STYLE OF DISPLAY
       OF THE PROPOSED-MSGELMT-LIST IS DESCRIBED IN CHAPTER 3.)


Beginning proposing ...  (THE SECOND ROUND OF PROPOSING)
trying $prop-color ...
 -> in $prop-color, need (lesspr msg-wt [= 2.0]  cut-off [= 3.04])
trying $condense-prop ...
 -> in parallel-enough?$1, need (lessp usol-posn [= 1]  max-dist
       [= 4.0])
 -> in parallel-enough?$2, need (greaterpr cur-item-prop-sal [= 0.4]
       *parallel-prop [= 0.3])
 -> in parallel-enough?$3, need (greaterpr pobj-prop-sal [= 0.3]
       *parallel-prop [= 0.3])
trying $prop-salience ...
trying $prop-sal-obj ...
trying $reln-salience ...
trying $light ...
trying $intro ...
trying $newitem ...
trying $finish-building-rspec ...

Just before inserting the msgelmt ...
      Cur-item and friends: $house-1
      ($2-story-building-1 $white-1 $new-england-house-1)
      ($in-front-of-1 $next-to-3 $next-to-2 $next-to-4)
```

The msg so far: ($$introduce-1-1)

The proposed-msgelmt-list (no. 1 goes to the msg):

1	$$in-front-of-1-2	0.57	$reln-salience
2	$$2-story-building-1-3	0.56	$prop-salience
3	$$2-story-building-1-4	0.38	$prop-sal-obj
4	$$finish-rspec-1-1	0.22	$finish-building-rspec
5	$$white-1-2	0.20	$prop-color

****

$RELN-SALIENCE WINS WITH ITS PROPOSAL TO MENTION IN-FRONT-OF(FENCE-1, HOUSE1). NOTE ALSO THAT IN THIS ROUND THE RULE "$FINISH-BUILDING-RSPEC" MAKES A LOW PRIORITY (.22) PROPOSAL – THIS RULE PROPOSES TO STOP BUILDING THE CURRENT R-SPEC AND SEND IT TO MUMBLE, AND THE PRIORITY OF ITS PROPOSAI IS A FUNCTION OF THE SIZE (WEIGHT) OF THE R-SPEC. SEE THE DISCUSSION IN CHAPTER VII.

Beginning proposing ... (THE THIRD ROUND OF PROPOSING)
trying $prop-color ...
-> in $prop-color, need (lesspr msg-wt [= 3.0] cut-off [= 3.04])
trying $condense-prop ...
-> in parallel-enough?$1, need (lessp usol-posn [= 1] max-dist [= 4.0])
-> in parallel-enough?$2, need (greaterpr cur-item-prop-sal [= 0.4] *parallel-prop [= 0.3])
-> in parallel-enough?$3, need (greaterpr pobj-prop-sal [= 0.3] *parallel-prop [= 0.3])
trying $prop-salience ...
trying $prop-sal-obj ...
trying $reln-salience ...
trying $light ...
trying $intro ...
trying $newitem ...
trying $finish-building-rspec ...

Just before inserting the msgelmt ...
Cur-item and friends: $house-1
($2-story-building-1 $white-1 $new-england-house-1)
($next-to-3 $next-to-2 $next-to-4)

AS A RESULT OF THE MOST SALIENT RELATION OF HOUSE-1, IN-FRONT-OF-1, BEING INSERTED INTO THE R-SPEC, THAT RELATION IS AUTOMATICALLY POPPED OFF OF THE CURRENT-RELATIONS LIST, LEAVING THE NEXT MOST SALIENT RELATION, NEXT-TO-3, IN THE 'SPOTLIGHT' AT THE FRONT OF THAT LIST. THIS RELATION ACTUALLY

GETS INTO THE R-SPEC (BELOW).

The msg so far: ($$introduce-1-1 $$in-front-of-1-2)

The proposed-msgelmt-list (no. 1 goes to the msg):

1	$$2-story-building-1-5	0.56	$prop-salience
2	$$next-to-3-1	0.48	$reln-salience
3	$$2-story-building-1-6	0.38	$prop-sal-obj
4	$$finish-rspec-2-1	0.32	$finish-building-rspec
5	$$white-1-3	0.20	$prop-color

****

IN THIS ROUND THE PROPOSAL TO FINISH BUILDING THE R-SPEC HAS RISEN TO A .32 PRIORITY. IT WILL KEEP RISING AS THE R-SPEC GROWS, UNTIL IT WINS.

Beginning proposing ...
trying $prop-color ...
 -> in $prop-color, need (lesspr msg-wt [= 3.5] cut-off [= 3.04])
trying $condense-prop ...
 -> in parallel-enough?$1, need (lessp usol-posn [= 1] max-dist [= 4.0])
 -> in parallel-enough?$2, need (greaterpr cur-item-prop-sal [= 0.4] *parallel-prop [= 0.3])
 -> in parallel-enough?$3, need (greaterpr pobj-prop-sal [= 0.3] *parallel-prop [= 0.3])
trying $prop-salience ...
trying $prop-sal-obj ...
trying $reln-salience ...
trying $light ...
trying $intro ...
trying $newitem ...
trying $finish-building-rspec ...

Just before inserting the msgelmt ...
Cur-item and friends: $house-1
($white-1 $new-england-house-1)
($next-to-3 $next-to-2 $next-to-4)

The msg so far: ($$introduce-1-1 $$in-front-of-1-2 $$2-story-building-1-5)

THE R-spec AT THIS POINT WOULD BE REALIZED AS "THIS IS A PICTURE OF A TWO STORY HOUSE WITH A FENCE IN FRONT OF IT."

The proposed-msgelmt-list (no. 1 goes to the msg):

```
      1       $$white-1-4                    0.56     $prop-salience
      2       $$next-to-3-2                  0.48     $reln-salience
      3       $$white-1-5                    0.38     $prop-sal-obj
      4       $$finish-rspec-3-1             0.37     $finish-building-rspec
**********

Beginning proposing ...
trying $prop-color ...
trying $condense-prop ...
 -> in parallel-enough?$1, need (lessp usol-posn [= 1]  max-dist
       [= 4.0])
 -> in parallel-enough?$2, need (greaterp:r cur-item-prop-sal [= 0.4]
       *parallel-prop [= 0.3])
 -> in parallel-enough?$3, need (greaterp:r pobj-prop-sal [= 0.3]
       *parallel-prop [= 0.3])
trying $prop-salience ...
trying $prop-sal-obj ...
trying $reln-salience ...
trying $light ...
trying $intro ...
trying $newitem ...
trying $finish-building-rspec ...

Just before inserting the msgelmt ...
      Cur-item and friends: $house-1
      ($new-england-house-1)
      ($next-to-3 $next-to-2 $next-to-4)

      The msg so far: ($$introduce-1-1 $$in-front-of-1-2
      $$2-story-building-1-5 $$white-1-4)

      The proposed-msgelmt-list (no. 1 goes to the msg):
      1       $$next-to-3-3                  0.48     $reln-salience
      2       $$new-england-house-1-1        0.42     $prop-salience
      3       $$finish-rspec-4-1             0.42     $finish-building-rspec
      4       $$new-england-house-1-2        0.38     $prop-sal-obj
**********

Beginning proposing ...
trying $prop-color ...
trying $condense-prop ...
trying $prop-salience ...
trying $prop-sal-obj ...
trying $reln-salience ...
trying $light ...
trying $intro ...
```

```
trying $newitem ...
trying $finish-building-rspec ...

Just before inserting the msgelmt ...
      Cur-item and friends: $house-1
      ($new-england-house-1)
      ($next-to-2 $next-to-4)

      The msg so far: ($$introduce-1-1 $$in-front-of-1-2
      $$2-story-building-1-5  $$white-1-4 $$next-to-3-3)

      The proposed-msgelmt-list (no. 1 goes to the msg):
      1      $$finish-rspec-5-1          0.52     $finish-building-rspec
      2      $$new-england-house-1-3     0.42     $prop-salience
      3      $$new-england-house-1-4     0.38     $prop-sal-obj
      4      $$next-to-2-1               0.33     $reln-salience

        IN THIS ROUND $FINISH-BUILDING-RSPEC FINALLY WON WITH ITS
        PROPOSAL, TERMINATING CONSTRUCTION OF THIS R-SPEC. THE
        FOLLOWING R-SPEC IS THEN SENT TO MUMBLE.

R-spec msg-1 is:
----

    THIS R-SPEC WOULD BE REALIZED BY MUMBLE AS
     "THIS IS A PICTURE OF A WHITE, TWO-STORY HOUSE WITH A FENCE
      IN FRONT OF IT AND A DRIVEWAY NEXT TO IT."

msgelmt: $$introduce-1-1      ($introduce-1)
    $$priority      2.0
    $$proposed-by   $intro

    themeobj: $introduce-1     (nil)
          $mentioned          1
          $object             $house-1
          $type               rhetorical
-----
msgelmt: $$in-front-of-1-2     ($in-front-of-1)
    $$priority      0.57
    $$proposed-by   $reln-salience

    themeobj: $in-front-of-1     (in-front-of-1)
          $sal                0.95
          $object             $house-1
          $agent              $fence-1
          $type               reln
```

```
          $mentioned          1
-----
msgelmt: $$2-story-building-1-5          ($2-story-building-1)
     $$priority        0.56
     $$proposed-by    $prop-salience

     themeobj: $2-story-building-1      (2-story-building-1)
          $object              $house-1
          $type                 prop
          $sal                  0.4
          $mentioned            1
-----
msgelmt: $$white-1-4         ($white-1)
     $$priority        0.56
     $$proposed-by    $prop-salience

     themeobj: $white-1       (white-1)
          $object              $house-1
          $type                 prop
          $sal                  0.4
          $mentioned            1
-----
msgelmt: $$next-to-3-3       ($next-to-3)
     $$priority        0.48
     $$proposed-by    $reln-salience

     themeobj: $next-to-3       (next-to-3)
          $sal                  0.8
          $object              $house-1
          $agent                $driveway-1
          $type                 reln
          $mentioned            1
----- end of R-spec -----
**********************************************************
**********************

Packet intro is turned off.          THE INTRO PACKET IS ALWAYS TURNED
                                          OFF AT THE END OF THE FIRST R-SPEC.
Beginning proposing ...
trying $prop-color ...
trying $condense-prop ...
trying $prop-salience ...
trying $prop-sal-obj ...
trying $reln-salience ...
trying $light ...
trying $intro ...
```

```
trying $newitem ...
trying $finish-building-rspec ...

Just before inserting the msgelmt ...
      Cur-item and friends: $house-1
      ($new-england-house-1)
      ($next-to-2 $next-to-4)

ALTHOUGH THE RULE "$NEWITEM" IS CLOSE TO WINNING WITH ITS PROPOSAL
TO GET THE NEXT OBJECT FROM THE USOL, "$HOUSE-1" IS STILL THE
CUR-ITEM, AND OTHER RULES STILL HAVE MODERATE PRIORITY THINGS TO
SAY ABOUT THE HOUSE. ($NEWITEM WILL ONLY RUN WHEN THE R-SPEC IS
EMPTY, TO AVOID NON-DELIBERATELY SWITCHING TO A NEW CUR-ITEM IN
MID R-SPEC.)

      The msg so far: nil

      The proposed-msgelmt-list (no. 1 goes to the msg):
      1      $$new-england-house-1-5      0.42     $prop-salience
      2      $$newcuritem-2-1             0.40     $newitem
      3      $$next-to-2-2                0.33     $reln-salience
**********

Beginning proposing ...
trying $prop-color ...
trying $condense-prop ...
 -> in parallel-enough?$1, need (lessp usol-posn [= 1]  max-dist
       [= 4.0])
 -> in parallel-enough?$2, need (greaterp:r cur-item-prop-sal [= 0.4]
       *parallel-prop [= 0.3])
 -> in parallel-enough?$3, need (greaterp:r pobj-prop-sal [= 0.3]
       *parallel-prop [= 0.3])
trying $prop-salience ...
trying $prop-sal-obj ...
trying $reln-salience ...
trying $light ...
trying $intro ...
trying $newitem ...
trying $finish-building-rspec ...

Just before inserting the msgelmt ...
      Cur-item and friends: $house-1
      nil
      ($next-to-2 $next-to-4)

      The msg so far: ($$new-england-house-1-5)
```

```
      The proposed-msgelmt-list (no. 1 goes to the msg):
      1       $$next-to-2-3                  0.33     $reln-salience
      2       $$finish-rspec-6-1             0.07     $finish-building-rspec
**********

Beginning proposing ...
trying $prop-color ...
trying $condense-prop ...
 -> in parallel-enough?$1, need (lessp usol-posn [= 1]  max-dist
       [= 4.0])
 -> in parallel-enough?$2, need (greaterp:r cur-item-prop-sal [= 0.4]
       *parallel-prop [= 0.3])
 -> in parallel-enough?$3, need (greaterp:r pobj-prop-sal [= 0.3]
       *parallel-prop [= 0.3])
trying $prop-salience ...
trying $prop-sal-obj ...
trying $reln-salience ...
trying $light ...
trying $intro ...
trying $newitem ...
trying $finish-building-rspec ...

Just before inserting the msgelmt ...
      Cur-item and friends: $house-1
      nil
      ($next-to-4)

      The msg so far: ($$new-england-house-1-5 $$next-to-2-3)

      The proposed-msgelmt-list (no. 1 goes to the msg):
      1       $$next-to-4-1                  0.33     $reln-salience
      2       $$finish-rspec-7-1             0.17     $finish-building-rspec
**********

Beginning proposing ...
trying $prop-color ...
trying $condense-prop ...
 -> in parallel-enough?$1, need (lessp usol-posn [= 1]  max-dist
       [= 4.0])
 -> in parallel-enough?$2, need (greaterp:r cur-item-prop-sal [= 0.4]
       *parallel-prop [= 0.3])
 -> in parallel-enough?$3, need (greaterp:r pobj-prop-sal [= 0.3]
       *parallel-prop [= 0.3])
trying $prop-salience ...
trying $prop-sal-obj ...
trying $reln-salience ...
```

trying $light ...
trying $intro ...
trying $newitem ...
trying $finish-building-rspec ...

Just before inserting the msgelmt ...
Cur-item and friends: $house-1
nil
nil

IN THIS RUN THE SETTING OF THE RULE PARAMETERS IS SUCH THAT EVERYTHING THERE WAS TO SAY ABOUT THE HOUSE IN THE DATA BASE GOT MENTIONED (BOTH PROPERTY AND RELATION LISTS WERE EMPTIED).

The msg so far: ($$new-england-house-1-5 $$next-to-2-3 $$next-to-4-1)

The proposed-msgelmt-list (no. 1 goes to the msg):
1 $$finish-rspec-8-1 0.27 $finish-building-rspec

THIS R-SPEC IS SENT TO MUMBLE, WHERE IT IS REALIZED AS SOMETHING LIKE "THE NEW ENGLAND STYLE HOUSE HAS A BUSH AND A TREE NEXT TO IT."

R-spec msg-2 is:
---
msgelmt: $$new-england-house-1-5 ($new-england-house-1)
$$priority 0.42
$$proposed-by $prop-salience

themeobj: $new-england-house-1 (new-england-house-1)
$object $house-1
$type prop
$sal 0.3
$mentioned 2
---
msgelmt: $$next-to-2-3 ($next-to-2)
$$priority 0.33
$$proposed-by $reln-salience

themeobj: $next-to-2 (next-to-2)
$sal 0.55
$object $house-1
$agent $bush-1
$type reln
$mentioned 2
---
msgelmt: $$next-to-4-1 ($next-to-4)

```
      $$priority        0.33
      $$proposed-by   $reln-salience

      themeobj: $next-to-4      (next-to-4)
            $sal                 0.55
            $object             $house-1
            $agent               $tree-1
            $type                reln
            $mentioned           2
----- end of R-spec -----
**********************************************************
**********************

Beginning proposing ...
trying $prop-color ...
trying $condense-prop ...
trying $prop-salience ...
trying $prop-sal-obj ...
trying $reln-salience ...
trying $light ...
trying $intro ...
trying $newitem ...
trying $finish-building-rspec ...

Just before inserting the msgelmt ...
      Cur-item and friends: $house-1
      nil
      nil

THIS TIME AT THE BEGINNING OF A NEW R-SPEC $NEWITEM WINS ...

      The msg so far: nil

      The proposed-msgelmt-list (no. 1 goes to the msg):
      1       $$newcuritem-3-1                0.40      $newitem


Cur-item just got reset to $door-1.
The USOL now is: (gate-1 driveway-1 mailbox-1 porch-1 road-1 porch-2
columns-1 sidewalk-1 columns-2 bush-1 bush-2 roof-1 tree-1 windows-1
yard-1)
**********************

Beginning proposing ...
trying $prop-color ...
 -> in $prop-color, need (lessp:r msg-wt [= 0.0]  cut-off [= 3.04])
```

```
trying $condense-prop ...
trying $prop-salience ...
trying $prop-sal-obj ...
trying $reln-salience ...
trying $light ...
trying $intro ...
trying $newitem ...
trying $finish-building-rspec ...

Just before inserting the msgelmt ...
      Cur-item and friends: $door-1
      ($red-1)
      ($part-of-1)

      The msg so far: nil

      The proposed-msgelmt-list (no. 1 goes to the msg):
      1       $$red-1-2                    1.12     $prop-salience
      2       $$part-of-1-1                0.54     $reln-salience
      3       $$newcuritem-4-1             0.40     $newitem
      4       $$red-1-1                    0.20     $prop-color
**********

Beginning proposing ...
trying $prop-color ...
trying $condense-prop ...
 -> in parallel-enough?$1, need (lessp usol-posn [= 1]  max-dist
       [= 4.0])
 -> in parallel-enough?$2, need (greaterp:r cur-item-prop-sal [= 0.8
       ]*parallel-prop [= 0.3])
 -> in parallel-enough?$3, need (greaterp:r pobj-prop-sal [= 0.7]
       *parallel-prop [= 0.3])
trying $prop-salience ...
trying $prop-sal-obj ...
trying $reln-salience ...
trying $light ...
trying $intro ...
trying $newitem ...
trying $finish-building-rspec ...

Just before inserting the msgelmt ...
      Cur-item and friends: $door-1
      nil
      ($part-of-1)

      The msg so far: ($$red-1-2)
```

```
      The proposed-msgelmt-list (no. 1 goes to the msg):
      1       $$part-of-1-2                0.54     $reln-salience
      2       $$condense-prop-1-1          0.35     $condense-prop
      3       $$finish-rspec-9-1           0.07     $finish-building-rspec
**********************

Beginning proposing ...
trying $prop-color ...
trying $condense-prop ...
 -> in parallel-enough?$1, need (lessp usol-posn [= 1]  max-dist
       [= 4.0])
 -> in parallel-enough?$2, need (greaterp:r cur-item-prop-sal [= 0.8
       ]*parallel-prop [= 0.3])
 -> in parallel-enough?$3, need (greaterp:r pobj-prop-sal [= 0.7]
       *parallel-prop [= 0.3])
trying $prop-salience ...
trying $prop-sal-obj ...
trying $reln-salience ...
trying $light ...
trying $intro ...
trying $newitem ...
trying $finish-building-rspec ...

Just before inserting the msgelmt ...
      Cur-item and friends: $door-1
      nil
      nil
```

NOTHING ELSE WOULD BE SAID ABOUT THE DOOR, EXCEPT THAT THE RULE $CONDENSE-PROP HAS NOTICED THAT THE GATE OF THE FENCE IS ALSO RED, AND DECIDES THAT THIS SHARED PROPERTY IS SALIENT ENOUGH IN BOTH CASES TO WARRANT CONDENSING THE OBJECTS' DESCRIPTIONS.

```
      The msg so far: ($$red-1-2 $$part-of-1-2)

      The proposed-msgelmt-list (no. 1 goes to the msg):
      1       $$condense-prop-2-1          0.35     $condense-prop
      2       $$finish-rspec-10-1          0.17     $finish-building-rspec
```

THE GATE IS MADE THE CURRENT-ITEM AND IS DESCRIBED AS USUAL (NO PROVISION HAS BEEN MADE IN THE $CONDENSE-PROP SHOWN HERE TO GUARANTEE THAT "RED-2", THE BASIS FOR THE CONDENSATION, GETS MENTIONED). THIS HAS NOT BEEN A PROBLEM YET (SINCE PROPERTIES SALIENT ENOUGH TO BE CONDENSED ON HAVE ALWAYS GOTTEN MENTIONED) I DON'T WANT TO MAKE ANY CLAIMS ABOUT THE "ACCIDENTAL" CHARACTEI THIS OPERATION.

```
Cur-item just got reset to $gate-1.
The USOL now is: (driveway-1 mailbox-1 porch-1 road-1 porch-2
columns-1 sidewalk-1 columns-2 bush-1 bush-2 roof-1 tree-1 windows-1
yard-1)
**********

Beginning proposing ...
trying $prop-color ...
 -> in $prop-color, need (lesspr msg-wt [= 1.5]  cut-off [= 3.04])
trying $condense-prop ...
trying $prop-salience ...
trying $prop-sal-obj ...
trying $reln-salience ...
trying $light ...
trying $intro ...
trying $newitem ...
trying $finish-building-rspec ...

Just before inserting the msgelmt ...
      Cur-item and friends: $gate-1
      ($red-2)
      ($part-of-2)

      The msg so far: ($$red-1-2 $$part-of-1-2 $$condense-prop-2-1)

      The proposed-msgelmt-list (no. 1 goes to the msg):
      1      $$red-2-2                    0.98     $prop-salience
      2      $$part-of-2-1                0.48     $reln-salience
      3      $$red-2-1                    0.20     $prop-color
      4      $$finish-rspec-11-1          0.17     $finish-building-rspec
**********

Beginning proposing ...
trying $prop-color ...
trying $condense-prop ...
trying $prop-salience ...
trying $prop-sal-obj ...
trying $reln-salience ...
trying $light ...
trying $intro ...
trying $newitem ...
trying $finish-building-rspec ...

Just before inserting the msgelmt ...
      Cur-item and friends: $gate-1
      nil
```

($part-of-2)

The msg so far: ($$red-1-2 $$part-of-1-2 $$condense-prop-2-1 $$red-2-2)

The proposed-msgelmt-list (no. 1 goes to the msg):
1 $$part-of-2-2 0.48 $reln-salience
2 $$finish-rspec-12-1 0.22 $finish-building-rspec

****

Beginning proposing ...
trying $prop-color ...
trying $condense-prop ...
trying $prop-salience ...
trying $prop-sal-obj ...
trying $reln-salience ...
trying $light ...
trying $intro ...
trying $newitem ...
trying $finish-building-rspec ...

Just before inserting the msgelmt ...
Cur-item and friends: $gate-1
nil
nil

The msg so far: ($$red-1-2 $$part-of-1-2 $$condense-prop-2-1 $$red-2-2 $$part-of-2-2)

The proposed-msgelmt-list (no. 1 goes to the msg):
1 $$finish-rspec-13-1 0.32 $finish-building-rspec

THIS R-SPEC COULD BE REALIZED IN MANY WAYS: "BOTH THE DOOR OF THE HOUSE AND THE GATE OF THE FENCE ARE RED", "THE HOUSE DOOR IS RED, SO IS THE GATE OF THE FENCE", ETC. WORK IS CURRENTLY UNDERWAY TO EXPLORE THE RHETORICAL DEMANDS AND CONSEQUENCES OF THESE CONSTRUCTIONS.

R-spec msg-3 is:

---

msgelmt: $$red-1-2 ($red-1)
$$priority 1.12
$$proposed-by $prop-salience

themeobj: $red-1 (red-1)
$object $door-1

```
            $type                prop
            $sal                 0.8
            $mentioned           3
-----
msgelmt: $$part-of-1-2       ($part-of-1)
     $$priority       0.54
     $$proposed-by    $reln-salience

     themeobj: $part-of-1      (dec30-0037)
            $sal                 0.9
            $agent               $door-1
            $object              $house-1
            $type                reln
            $mentioned           3
-----
msgelmt: $$condense-prop-2-1       ($condense-prop-2)
     $$priority       0.3528
     $$proposed-by    $condense-prop

     themeobj: $condense-prop-2      (lambda)
            $simil                     (props (red))
            $mentioned           nil
            $object              ($door-1 $gate-1)
            $type                rhetorical
-----
msgelmt: $$red-2-2       ($red-2)
     $$priority       0.98
     $$proposed-by    $prop-salience

     themeobj: $red-2      (red-2)
            $object              $gate-1
            $type                prop
            $sal                 0.7
            $mentioned           3
-----
msgelmt: $$part-of-2-2       ($part-of-2)
     $$priority       0.48
     $$proposed-by    $reln-salience

     themeobj: $part-of-2      (dec30-0105)
            $sal                 0.8
            $agent               $gate-1
            $object              $fence-1
            $type                reln
            $mentioned           3
----- end of R-spec -----
```

```
**********************************************************
**********************

Beginning proposing ...
trying $prop-color ...
trying $condense-prop ...
trying $prop-salience ...
trying $prop-sal-obj ...
trying $reln-salience ...
trying $light ...
trying $intro ...
trying $newitem ...
trying $finish-building-rspec ...

Just before inserting the msgelmt ...
      Cur-item and friends: $gate-1
      nil
      nil

      The msg so far: nil

      The proposed-msgelmt-list (no. 1 goes to the msg):
      1       $$newcuritem-5-1                0.40      $newitem

Cur-item just got reset to $mailbox-1.
The USOL now is: (porch-1 road-1 porch-2 columns-1 sidewalk-1
columns-2 bush-1 bush-2 roof-1 tree-1 windows-1 yard-1)
**********************

Beginning proposing ...
trying $prop-color ...
 -> in $prop-color, need (lesspr msg-wt [= 0.0]  cut-off [= 3.04])
trying $condense-prop ...
trying $prop-salience ...
trying $prop-sal-obj ...
trying $reln-salience ...
trying $light ...
trying $intro ...
trying $newitem ...
trying $finish-building-rspec ...

Just before inserting the msgelmt ...
      Cur-item and friends: $mailbox-1
      ($white-7)
      ($in-front-of-3)
```

```
      The msg so far: nil

ALTHOUGH THE MAILBOX HAS A PROPERTY AND A RELATION IN THE DATA
BASE, THE RULES FIND THAT NEITHER OF THEM IS SALIENT ENOUGH,
AND THE WINNING PROPOSAL IN THIS FIRST ROUND OF PROPOSING
IS TO THROW OUT THE MAILBOX CURRENT-ITEM AND TRY THE NEXT
MOST SALIENT OBJECT, PORCH-1.

      The proposed-msgelmt-list (no. 1 goes to the msg):
      1     $$newcuritem-6-1          0.40    $newitem
      2     $$in-front-of-3-1         0.33    $reln-salience
      3     $$white-7-1               0.20    $prop-color

Cur-item just got reset to $porch-1.
The USOL now is: (road-1 porch-2 columns-1 sidewalk-1 columns-2
bush-1 bush-2 roof-1 tree-1 windows-1 yard-1)
**********

Beginning proposing ...
trying $prop-color ...
 -> in $prop-color, need (lessp:r msg-wt [= 0.0]  cut-off [= 3.04])
trying $condense-prop ...
trying $prop-salience ...
trying $prop-sal-obj ...
trying $reln-salience ...
trying $light ...
trying $intro ...
trying $newitem ...
trying $finish-building-rspec ...

Just before inserting the msgelmt ...
      Cur-item and friends: $porch-1
      ($white-2)
      ($part-of-3)

      The msg so far: nil

      The proposed-msgelmt-list (no. 1 goes to the msg):
      1     $$part-of-3-1             0.45    $reln-salience
      2     $$newcuritem-7-1          0.40    $newitem
      3     $$white-2-1               0.20    $prop-color
**********

Beginning proposing ...
trying $prop-color ...
 -> in $prop-color, need (lessp:r msg-wt [= 1.0]  cut-off [= 3.04])
```

```
trying $condense-prop ...
 -> in parallel-enough?$1, need (lessp usol-posn [= 2]  max-dist
       [= 4.0])
 -> in parallel-enough?$2, need (greaterp:r cur-item-prop-sal [= 0.2
       ]*parallel-prop [= 0.3])
 -> in parallel-enough?$3, need (greaterp:r pobj-prop-sal [= 0.2]
       *parallel-prop [= 0.3])
trying $prop-salience ...
trying $prop-sal-obj ...
trying $reln-salience ...
trying $light ...
trying $intro ...
trying $newitem ...
trying $finish-building-rspec ...

Just before inserting the msgelmt ...
      Cur-item and friends: $porch-1
      ($white-2)
      nil

      The msg so far: ($$part-of-3-1)

      The proposed-msgelmt-list (no. 1 goes to the msg):
      1       $$white-2-2                    0.20     $prop-color
      2       $$finish-rspec-14-1            0.12     $finish-building-rspec
**********

Beginning proposing ...
trying $prop-color ...
trying $condense-prop ...
 -> in parallel-enough?$1, need (lessp usol-posn [= 2]  max-dist
       [= 4.0])
 -> in parallel-enough?$2, need (greaterp:r cur-item-prop-sal [= 0.2
       ]*parallel-prop [= 0.3])
 -> in parallel-enough?$3, need (greaterp:r pobj-prop-sal [= 0.2]
       *parallel-prop [= 0.3])
trying $prop-salience ...
trying $prop-sal-obj ...
trying $reln-salience ...
trying $light ...
trying $intro ...
trying $newitem ...
trying $finish-building-rspec ...

Just before inserting the msgelmt ...
      Cur-item and friends: $porch-1
```

```
nil
nil

The msg so far: ($$part-of-3-1 $$white-2-2)

The proposed-msgelmt-list (no. 1 goes to the msg):
1       $$finish-rspec-15-1              0.17      $finish-building-rspec
```

THIS R-SPEC COMES OUT AS "THE HOUSE HAS A WHITE PORCH". (A RULE BEING DEVELOPED -- DESCRIBED IN CHAPTER 5 -- WOULD AT THIS POINT HAVE CONDENSED THE DESCRIPTIONS OF PORCH-1 AND PORCH-2, SINCE THEY ARE BOTH PART OF THE SAME HOUSE.)

```
R-spec msg-4 is:
-----
msgelmt: $$part-of-3-1        ($part-of-3)
     $$priority       0.45
     $$proposed-by   $reln-salience

     themeobj: $part-of-3      (dec30-0038)
          $sal                  0.75
          $agent                $porch-1
          $object               $house-1
          $type                 reln
          $mentioned            4
-----
msgelmt: $$white-2-2        ($white-2)
     $$priority       0.2
     $$proposed-by   $prop-color

     themeobj: $white-2      (white-2)
          $object              $porch-1
          $type                prop
          $sal                 0.2
          $mentioned           4
---- end of R-spec -----
************************************************************
**********

Beginning proposing ...
trying $prop-color ...
trying $condense-prop ...
trying $prop-salience ...
trying $prop-sal-obj ...
trying $reln-salience ...
trying $light ...
```

```
trying $intro ...
trying $newitem ...
trying $finish-building-rspec ...

Just before inserting the msgelmt ...
      Cur-item and friends: $porch-1
      nil
      nil

      The msg so far: nil

      The proposed-msgelmt-list (no. 1 goes to the msg):
      1       $$newcuritem-8-1              0.40      $newitem

Cur-item just got reset to $road-1.
The USOL now is: (porch-2 columns-1 sidewalk-1 columns-2 bush-1
bush-2 roof-1 tree-1 windows-1 yard-1)
**********

Beginning proposing ...
trying $prop-color ...
trying $condense-prop ...
trying $prop-salience ...
trying $prop-sal-obj ...
trying $reln-salience ...
trying $light ...
trying $intro ...
trying $newitem ...
trying $finish-building-rspec ...

Just before inserting the msgelmt ...
      Cur-item and friends: $road-1
      ($suburban-1)
      ($next-to-1 $in-front-of-3 $in-front-of-2)

      The msg so far: nil

      The proposed-msgelmt-list (no. 1 goes to the msg):
      1       $$next-to-1-1              0.42      $reln-salience
      2       $$newcuritem-9-1           0.40      $newitem
**********

Beginning proposing ...
trying $prop-color ...
trying $condense-prop ...
trying $prop-salience ...
```

```
trying $prop-sal-obj ...
trying $reln-salience ...
trying $light ...
trying $intro ...
trying $newitem ...
trying $finish-building-rspec ...

Just before inserting the msgelmt ...
      Cur-item and friends: $road-1
      ($suburban-1)
      ($in-front-of-3 $in-front-of-2)

      The msg so far: ($$next-to-1-1)

      The proposed-msgelmt-list (no. 1 goes to the msg):
      1       $$in-front-of-3-2          0.33     $reln-salience
      2       $$finish-rspec-16-1        0.12     $finish-building-rspec
**********

Beginning proposing ...
trying $prop-color ...
trying $condense-prop ...
trying $prop-salience ...
trying $prop-sal-obj ...
trying $reln-salience ...
trying $light ...
trying $intro ...
trying $newitem ...
trying $finish-building-rspec ...

Just before inserting the msgelmt ...
      Cur-item and friends: $road-1
      ($suburban-1)
      ($in-front-of-2)

      The msg so far: ($$next-to-1-1 $$in-front-of-3-2)

      The proposed-msgelmt-list (no. 1 goes to the msg):
      1       $$finish-rspec-17-1        0.22     $finish-building-rspec
```

THIS R-SPEC IS REALIZED AS "A ROAD RUNS IN FRONT OF THE FENCE, AND THERE IS A MAILBOX IN FRONT OF THE ROAD." NOTE THAT THE MAILBOX DID GET MENTIONED, BY VIRTUE OF ITS RELATIONSHIP WITH THE ROAD.

```
R-spec msg-5 is:
-----
```

```
msgelmt: $$next-to-1-1        ($next-to-1)
     $$priority        0.42
     $$proposed-by   $reln-salience

     themeobj: $next-to-1      (next-to-1)
          $sal                  0.7
          $object               $fence-1
          $agent                $road-1
          $type                 reln
          $mentioned            5
----
msgelmt: $$in-front-of-3-2       ($in-front-of-3)
     $$priority        0.33
     $$proposed-by   $reln-salience

     themeobj: $in-front-of-3     (in-front-of-3)
          $sal                  0.55
          $object               $road-1
          $agent                $mailbox-1
          $type                 reln
          $mentioned            5
----- end of R-spec -----
**********************************************************
**********************

Beginning proposing ...
trying $prop-color ...
trying $condense-prop ...
trying $prop-salience ...
trying $prop-sal-obj ...
trying $reln-salience ...
trying $light ...
trying $intro ...
trying $newitem ...
trying $finish-building-rspec ...

Just before inserting the msgelmt ...
      Cur-item and friends: $road-1
      ($suburban-1)
      ($in-front-of-2)

      The msg so far: nil

      The proposed-msgelmt-list (no. 1 goes to the msg):
      1       $$newcuritem-10-1              0.40      $newitem
```

AT THE BEGINNING OF THIS R-SPEC $NEWITEM DOES ITS FAMILIAR "FETCH" OF THE NEXT MOST SALIENT OBJECT. HOWEVER, AS HAPPENED TO MAILBOX, NOTHING ABOUT THE BACK PORCH IS SALIENT ENOUGH TO MENTION.

Cur-item just got reset to $porch-2.
The USOL now is: (columns-1 sidewalk-1 columns-2 bush-1 bush-2 roof-1
tree-1 windows-1 yard-1)
****

Beginning proposing ...
trying $prop-color ...
 -> in $prop-color, need (lessp:r msg-wt [= 0.0] cut-off [= 3.04])
trying $condense-prop ...
trying $prop-salience ...
trying $prop-sal-obj ...
trying $reln-salience ...
trying $light ...
trying $intro ...
trying $newitem ...
trying $finish-building-rspec ...

Just before inserting the msgelmt ...
Cur-item and friends: $porch-2
($white-3)
nil

The msg so far: nil

The proposed-msgelmt-list (no. 1 goes to the msg):

1	$$newcuritem-11-1	0.40	$newitem
2	$$white-3-1	0.20	$prop-color

WHEN $NEWITEM GOES TO POP THE NEXT OBJECT OFF OF THE USOL, HOWEVER, IT FINDS THAT THE OBJECT ITSELF IS BELOW THE SYSTEM'S LOWER THRESHOLD ON OBJECT SALIENCE, AND ITS ACTION IS TO SIGNAL THIS CONDITION.

** The salience of the new cur-item ($columns-1) is at or below 0.3
(it is 0.30), so the description is ended.
****

THIS TRIGGERS TURNING ON THE "CONCLUDE" PACKET, WHICH CONTAINS THE RULE "$LIGHT" ...

Packet shift-topic is turned off.

Packet elaborate is turned off.

```
Packet conclude is turned on.

Beginning proposing ...
trying $prop-color ...
trying $condense-prop ...
trying $prop-salience ...
trying $prop-sal-obj ...
trying $reln-salience ...
trying $light ...
trying $intro ...
trying $newitem ...
trying $finish-building-rspec ...

Just before inserting the msgelmt ...
      Cur-item and friends: nil
      ($white-3)
      nil

      The msg so far: nil


      The proposed-msgelmt-list (no. 1 goes to the msg):
      1      $$light-6-1                    0.80     $light
**********
```

```
Packet finish is turned off.

Beginning proposing ...
trying $prop-color ...
trying $condense-prop ...
trying $prop-salience ...
trying $prop-sal-obj ...
trying $reln-salience ...
trying $light ...
trying $intro ...
trying $newitem ...
trying $finish-building-rspec ...

Just before inserting the msgelmt ...
      Cur-item and friends: nil
      ($white-3)
      nil

      The msg so far: ($$light-6-1)

      The proposed-msgelmt-list (no. 1 goes to the msg):
```

1 $$finish-rspec-18-1 0.22 $finish-building-rspec

(E.G. DAY OR NIGHT, CLOUDY OR CLEAR, ...) IS MENTIONED. SINCE THE PRINTING OF THIS RUN THE DETAILS OF REPRESENTING AND PROCESSING THIS GESTALT HAVE BEEN REFINED CONSIDERABLY.

R-spec msg-6 is:

msgelmt: $$light-6-1 ($light-6)
$$priority 0.8
$$proposed-by $light

themeobj: $light-6 (nil)
$mentioned 6
$object nil
$type rhetorical

----- end of R-spec -----

****

REFERENCES

Appelt, Doug, *Planning Natural Language Utterances to Satisfy Multiple Goals*, Ph.D. Disseration, Stanford University (to appear as a technical report from SRI International), 1982.

Arbib, Michael A., "Parallelism, Slides, Schemas, and Frames", in *Systems: Approaches, Theories, Applications*, W. E. Hartnett (ed.), D. Reidel Publishing Company, Dordrecht-Holland, 1977.

Arnheim, Rudolf, *Art and Visual Perception: A Psychology of the Creative Eye*, University of California Press, Berkeley, 1974.

Boggess, Lois, *Computational Interpretation of English Spatial Prepositions*, Ph.D. thesis, University of Illinois, 1978.

Brachman, R., *A Structural Paradigm for Representing Knowledge*, Report 3605, Bolt, Beranek, and Newman, Cambridge, Mass., 1978.

Brown, Gretchen, "Deep Generation", an unpublished class paper at M.I.T., 1973.

——, *Some Problems in German to English Machine Translation*, Massachusetts Institute of Technology, Technical Report 142, December 1974. Project MAC.

Brush, Robert, "The Attractiveness of Woodlands: Perceptions of Forest Landowners in Massachusetts", *Forest Science*, Vol. 25, No. 3, pp. 495-506, 1979.

Clancey, W. (to appear) "The Epistemology of a Rule-Based Expert System: A Framework for Explanation", Journal of Artificial Intelligence; also available as Heuristic Programming Project Report 81-17, Stanford University, November 1981.

Cohen, P., *On Knowing What to Say: Planning Speech Acts*, University of Toronto, Technical Report 118, 1978.

Cooper, G. S., "A Semantic Analysis of English Locative Prepositions", Bolt, Beranek, and Newman report #1587, B.B.&N., Cambridge, 1968.

Conklin, Jeffrey "A Scene Description System: Preliminary Survey", unpublished Master's Thesis, Department of Computer and Information Science, University of Massachusetts, Amherst, Mass., 1979.

—— and D. McDonald "Salience: The Key to the Selection Problem in Natural Language Generation", in the Proceedings of the Association for Computational Linguistics, Toronto, Canada, 1982.

——, K. Ehrlich, and D. McDonald, "An Empirical Investigation of Visual Salience and its Role in Text Generation", in *Cognition and Brain Theory*, Vol. 6, No. 2, Spring 1983, or as COINS TR 83-14, COINS, University of Massachusetts at Amherst, 01003.

Davey, A., *Discourse Production*, Edinburgh University Press, Edinburgh, 1979.

Dehn, N., "Memory in story invention", in *Proceedings of the Third Annual Conference of the Cognitive Science Society*, University of California, Berkeley, August 1981.

Didday, R. L. and M. A. Arbib, "Eye Movements and Visual Perception: A 'Two Visual System' Model", in *Int. J. Man-Machine Studies*, Vol. 7, pp 547-569, 1975.

Firschein, O. and Fischler, M. A., "Describing and Abstracting Pictorial Structures", in *Pattern Recognition*, Vol. 3, pp. 421-443, 1971.

Friedman, J., "Directed random generation of sentences," *Communications of the ACM* 12, (6), 1969.

Garrett, M., "The Analysis of Sentence Production", in *Psychology of Learning and Motivation*, Academic Press: New York, Vol. 9, 1975.

Goldman, N. M., *Computer Generation of Natural Language from a Deep Conceptual Base*, Ph.D. thesis, Stanford University, 1974.

Grice, H. P. (1975) "Logic and Conversation", in P. Cole and J. L. Morgan (Eds.) *Syntax and Semantics: Speech Acts*, Vol. 3, Academic Press, N.Y.

Grosz, B. J., "Focusing and description in natural language dialogs," in A. Joshi, et al. (eds.), *Elements of Discourse Understanding: Proceedings of a Workship on computational Aspects of Linguistic Structure and Discourse Setting*, Cambridge University Press, Cambridge, 1980.

Hanson, A. R. and Riseman, E. M. "VISIONS: A Computer System for Interpreting Scenes", in *Computer Vision Systems*, Hanson, A. R. and Riseman, E. M. (Eds), Academic Press, New York, pp 449-510, 1978.

Herskovits, A., *The generation of French from Semantic Structure*, Stanford Artificial Intelligence Labratory, Technical Report 212, 1973.

Hooper, K., "Picture Recognition: A Consideration of Representational Media and Realism", unpublished paper, 1980.

Kempen, G., "Building a Psychologically Plausible Sentence Generator", presented a the Conference on Empirical and Methodological Foundations of Semantic Theories for Natural Language, Nijmegen, The Netherlands, 1977.

Loftus, G. R., "Models of Picture Recognition", in *Learning by Eye*, R. Wu and S. Chipman (Eds.), 1982.

Mann, W., Madeline Bates, Barbara Grosz, David McDonald, Kathleen McKeown, and William Swartout, "Text Generation: The State of the Art and the Literature", Information Sciences Institute technical report RR-81-101, Marina del Rey, California, 1981.

Mann, W. and Moore, J. "Computer Generation of Multiparagraph Text", *American Journal of Computational Linguistics*, 7:1, Jan-Mar 1981, pp 17-29.

Marcus, M. *A Theory of Syntactic Recognition for Natural Language*, MIT Press, Cambridge, Massachusetts, 1980.

McDonald, David D., *Language Production as a Process of Decision-Making under Constraints*, PhD. dissertation, MIT, 1980.

——, "Language Generation: the source of the dictionary", in the Proceedings of the Annual Conference of the Association for Computational Linguistics, Stanford University, June, 1981a.

——, "MUMBLE: A Flexible System for Language Production", in the Proceedings of the 7th IJCAI (Vol. II), Vancouver, B.C., Canada, 1981b.

—— and J. Conklin "Salience as a Simplifying Metaphor for Natural Language Generation", in the Proceedings of the Annual Conference of the American Association of Artificial Intelligence, 1982a.

——, "Natural Language Generation as a Computational Problem: an Introduction", in *Computational Models of Discourse*, M. Brady and R. Berwick (Eds), MIT Press, Cambridge, Mass, 1983a.

——, "Description Directed Control: Its implications for natural language generation", in *International Journal of Computers and Mathematics*, Vol. 9, No. 1, 1983b.

McKeown, K. R. *Generating Natural Language Text in Response to Questions about the Data Base Structure*, Ph.D. Dissertation, Moore School of Electrical Engineering, University of Pennsylvania, 1982.

Meehan, J. R., "Using planning structures to generate stories", *American Journal of Computational Linguistics*, Fiche 33, 1975.

——, "TALE-SPIN, an interactive program that writes stories", in *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, August 1977.

Parma, Cesare C., Hanson, A. R., and Riseman, E. M. "Experiments in Schema-Driven

Interpretation of a Natural Scene", in *Digital Image Processing*, Simon, J. C. and Haralick, R. M. (Eds), D. Reidel Publishing Co., Dordrecht, Holland, 1980 pp 303-334.

Searle, S. R., *Speech Acts: An Essay in the Philosophy of Language*, Cambridge University Press, Cambridge, England, 1969.

Selfridge, Oliver G., "Pandemonium, a Paradigm for Learning." In *Proceedings of the Symposium on Mechanisation of Thought Processes*. D. V. Blake and A. M. Uttley, eds. London: H. M. Stationery Office, 1959.

Selfridge, P. G., "Reasoning About Success and Failure in Aerial Image Understanding", TR 103, Computer Science Department, University of Rochester, N.Y., May 1982.

Simmons, R., and J. Slocum, "Generating English discourse from semantic networks." *Communications of the ACM* 15, (10) October 1972, 891-905.

Soloway, Elliot, Beverly Woolf, Eric Rubin, and Paul Barth "Meno-II: An Intelligent Tutoring System for Novice Programmers", Proceedings of International Joint Conference in Artificial Intelligence, Vancouver, British Columbia, 1981.

Swartout, W. *A Digitalis Therapy Advisor with Explanations*, Massachusetts Institute of Technology, Laboratory for Computer Science, Technical Report, February 1977.

Swartout, W. *Producing Explanations and Justifications of Expert Consulting Programs*, Technical Report 251, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1981.

Thompson, H., (1977) "Strategy and tactics: a model of language production", in *Papers from the 13th Regional Meeting, Chicago Linguistic Society*.

Waltz, David, and Lois Boggess, "Visual Analog Representations for Natural Language Understanding", in the Proceedings of IJCAI-79, pp. 926-934, 1979.

Waltz, David, "Generating and Understanding Scene Descriptions", in *Elements of discourse understanding*, Joshi, Webber, and Sag (Eds.), Cambridge University Press, 1981.

Wesley, Leonard P. and Hanson, Allen R., "The use of an Evidential-Based Model for Representing Knowledge and Reasoning about Images in the VISIONS System", in proceeding of the Workshop on Computer Vision, Ringe, New Hampshire, IEEE Computer Society Press, August 23-25, 1982.

Winograd, T., *Understanding Natural Language*, Academic Press, New York, 1972.

Woolf, Beverly, "The U. Mass. KL-ONE System", unpublished user's manual, Department

of Computer and Information Science, University of Massachusetts, Amherst, Mass., 1981.