Apt:
A System to Direct and Control Natural Language Generation

Penelope Sibun
Department of Computer and Information Science
University of Massachusetts at Amherst

COINS Technical Report 87-42

This research was supported in part by Contract N00014-87-K-0238 from the Office of Naval Research.

Apt: A System to Direct and Control Natural Language Generation

Penelope Sibun

Department of Computer and Information Science
University of Massachusetts at Amherst

Abstract

A key area in current research in natural language processing is the production of language for a purpose and in a context. This requires a system capable of more than the generation of isolated sentences. The system must mediate between a knowledge source—which has something to say, and a linguistic component—which has the mechanisms for saying it. Apt is one such system, which directs the construction of a description from a knowledge base of apartment layouts, and controls the description's expression by a linguistic component. This paper first discusses the domain of apartment layouts, then the structure of Noumena, the underlying knowledge base, and the nature of Apt's interface to both Noumena and the natural language generator Mumble. Next, some of the goals of Apt itself, and work being done to meet them, are explored. Finally, further theoretical issues, including lexical selection and the representation of deictic information, are touched upon.

1 The Model of Apt

What does directing and controlling natural language generation mean? It is taking some underlying knowledge base and directing the expression of this knowledge in a complete and meaningful way by a linguistic generation component of a natural language system. The generator possesses the capability of producing language from specifications, but its expertise is local to units roughly the size of a phrase. The controller directs the arrangement of phrases into clauses, clauses into sentences, and sentences into larger units; the controller is responsible for the structure of the discourse as a whole. The controller, based on any number of considerations, decides what subset of the knowledge base to talk about, and in what sequence. It keeps track of long-distance phenomena like focus and subsequent reference. The controlling and directing system decides what to say and, to a large extent, how to say it, while the generator does the actual saying.

1.1 Directing and Controlling for Mumble

Apt directs Mumble's language generation, though we would like the principles which guide the design of Apt to be as generally applicable as possible. In this paper, references to generation will be specifically to Mumble, unless noted otherwise. Mumble's representation is claimed to be necessary and sufficient for linguistic information, while Apt will be encoding information which is for the most part nonlinguistic, and will not want or need to confine itself to Mumble's representation.

Because Mumble is finely tailored to language, and Apt is not, we will not have a single knowledge representation throughout all levels of our system, though many other researchers have argued that this is desirable. [Bienkowski 86] [Appelt 85] [Paris 86] But we have not lost any power by having divergent representations. We are confident that the interface between Apt and Mumble is a sufficiently rich and versatile channel of communication to allow Apt to tell Mumble everything Apt would like to convey.

1.2 Apt As Go-between

Apt is the mediator between the underlying program (eg, expert system) or knowledge base and Mumble. The knowledge base and Mumble are completely separate entities and know nothing of each other. Apt, however, must know about both: it must know how to find information in the knowledge base it needs to express, and how to integrate what it finds; it must be able to tell Mumble how it wants this knowledge expressed.

I have been speaking as though the motivation for saying something comes from Apt, that is, as though in some sense the generation-directing component is the intelligent agent of the system. While this is in fact the case here, it is an artifact of working with a partial natural language system. A full-fledged natural language system, as we fondly conceive of it, would consist of a linguistic input component (a parser), a linguistic output component (a generator), a knowledge base or expert system—a knowledge base with an inference engine, and a large glob of glue holding these three together. In Counselor [McDonald 86c], whose first three components were, respectively, PLUM, Mumble, and Hypo (a legal reasoner), this glob of glue was Cicero. Cicero, loosely refered to as the discourse component, was responsible for mediating the communication among the other components. A desirable aspect of this mediation was a common representation; ideally the parser would parse into the same sort of structures used for controlling generation. This consideration should be kept in the back of the mind while designing Apt, so that someday we might integrate Apt into a larger system.

Cicero would also mediate the discourse, so that Counselor would behave like an intelligent agent as it interacted with a person. This requires expertise in carrying on a conversation: remembering what has been said; recognizing the nature of discourse moves—eg, there has just been a question, so an answer is in order; and expressing what needs to be said in an understandable manner. This last, of course, is the job of the component directing generation. I expect scepticism on this point, but I claim that intelligent agency is diffuse throughout the discourse component: its collective goals collude to bring about intelligent behavior.

1.3 Layout of this Paper

This paper lays out the preliminary specifications for a system that will describe the layout of a house, as a person would if asked `Could you please describe for me the layout of this house?` I first discuss the domain of layout descriptions. In the following three sections, I describe the knowledge base, our generator Mumble, and the function of the component, Apt, which lies in between. In the **Theses**, Section 7, I present some claims about Apt's functionality, and in the final section I consider lexical selection.

1.4 Terminology

It is exciting that the area of directing natural language generation is so new, but there is the attendant need to be careful about what to call things. It is desirable to steer clear of terminology which carries around baggage we may not want to buy into. *Coherence* is an example of such a term. Then there are terms, like *discourse unit*, for which it is clear that everyone has her own definition, but no one really knows what they mean. Finally, we want to avoid vapid terminology, such as *focus*, that is so worn as to lack any precise meaning.

The terminology for the actual processes that Apt engages in has proved to be problematic. Traditionally, Apt would have been referred to as a *planner*, since it *plans* what the generator will produce. However, I have found this term to be highly confusing. In the AI community, the notion of a planner as something that builds a plan to accomplish a goal and then executes it is too deeply entrenched. While certainly, Apt can be said *in some sense* to operate in this way, the notion does not capture very much of what Apt does, and can be downright misleading. Much of Apt's processing, as described in this paper, involves linearizing a knowledge base, constructing descriptions, and deciding on the form of the generator's expression of the descriptions.

Because of these considerations, I prefer not to use the term *planner* for Apt. Unfortunately, a concise and accurate alternative has not yet come to mind, so I have been making do with "a system that directs and controls the generator's execution." While this is rather clumsy, I feel that *direct* and *control* impart a better flavor of Apt's activities than *plan*.

There are constructs in Apt which will be realized in Mumble as function words, phrases, and other linguistic objects. "Content" words are not in Mumble per se, they are in the lexicon. Apt will have a representation for these, too. To avoid any confusion and to leave questions of implementation open, I will use the following typographical conventions: for the Mumble construct *restrictive-relative-clause*, Apt will have Apt::restrictive-relative-clause; for the lexical entry `left`, Apt will have Apt::left.

2 The Domain

The domain we have chosen is based on Charlotte Linde's work with apartment layouts. [Linde 74] Her dissertation constitutes a seminal work in the study of discourse—the structure of language over units larger than a sentence. of her reasons for choosing people's descriptions of apartment layouts as a locus of study mirror our considerations, as computational linguists, in choosing a tractable domain for our research.

Apartment descriptions are single-speaker, so we don't have to worry about the turn-taking strategies and discourse moves of multiple-speaker dialogs. The topic is self-contained: people know when they are done. The task is something everyone seems to be able to do, but it is not such an everyday endeavor that is has become formulaic: it is likely that people are actively constructing what they are saying. The thing being described is part of the physical world, so we can validate what speakers have said by comparing their descriptions to "reality." For our purposes in building a computer program, this last is important. We have some hope of constructing a knowledge base that will capture most of the elements we find in people's descriptions; both the affective and abstract content of this sort of task are minimal, and, at a first approximation, seem easily factored out. Finally, within the constraints of the task, we still find a rich interspeaker variety of expression.

2.1 The Data

Linde gathered 72 descriptions of apartment layouts from New York City residents, who were unaware that she was doing research in sociolinguistics. For about half the descriptions, she also obtained a sketch by the speaker of the layout.

For our own corpus of data, I interviewed seven people who were familiar with my one-level house. All of them had spent several days and at least one night there. These data are in the process of being analyzed, and while some of Linde's predictions hold up nicely, we have found some apparent contradictions to her rules for apartment descriptions. Some of the variance can certainly be explained by differences in the two studies: Linde's subjects were talking about New York City apartments while mine were talking about a house in a rural area; Linde did not know her subjects while I knew mine well; and I did not have available to me a complete specification of Linde's methods when I collected my own protocols. None of these can be factored out at this time, instead we must bear them in mind when noting differences between the two corpora.

2.2 Components of Descriptions

Linde conceived of an underlying knowledge model of apartment layouts as "a system of spatial relations involving points and positions." [Linde 74] (p 93) Being a linguist, she felt that the organization of the system should be formalized as a phrase structure network. The points in her network are rooms, and she feels that the unmarked (default) strategy for a layout description is to build up the apartment room by room, starting at the front door and traversing the rooms as though one were conducting a tour. Some rooms, *one-room branches*, which have a single entrance, may be visited by the tour guide "standing in the doorway" and indicating them. *Multi-room branches* must be entered, so that each room may be visited.

Linde found that the space in an apartment was divided into two types: storage space, exemplified by closets, and living space. People divided the living space into three classes : *hallways*; *major rooms*; and *minor rooms*. *Major rooms* include living room, kitchen, and bedroom. *Minor rooms* include dining room, study, library, and bathroom, that is, rooms which are either not obligatory or which have some special social status (ie, bathroom). Major rooms are those whose existence can be presupposed. Linde claims that this has the interesting consequence that a major room can be introduced with the, even if it has not been previously mentioned, while a minor room cannot.

As the apartment is described, two pieces of information must be mentioned about each room: its *existence* and its *location*. Generally, these pieces of information are realized as a noun phrase and a locative phrase. Several types of sentences result from possible combinations.

```
(1) To the left is the bedroom.

(2) You walk left into the bedroom.

(3) You go left and there's the bedroom.
```

As can be seen, the last example is a conjunction of two sentences, one of which mentions the room's location and the other its existence. Linde suggests that this production of sentences with diluted semantic content is most often found when the speaker is having trouble constructing what to say next.

2.3 Levels of Planning

Based on her analysis of the protocols, Linde suggests four levels of planning a layout description. First, the speaker chooses a strategy, either a map—describing the apartment as though one were looking at an architectural drawing of it, or a tour—describing the apartment as though one were leading someone through the rooms, in some hypothetical sense.

Second,[1] the speaker must decide the order in which to visit the rooms of the apartment, that is, how to conduct the tour. The order is largely constrained, of course, by the actual relationships among the rooms, but there are choice points in almost every layout, and the speaker must decide which branch to follow. Linde claims that a tour is composed of optional introductory and closing summaries, and an establishment of a starting point followed by the actual tour. Clearly, establishing a starting point is an important step; it constrains the course of the tour and

[1]Since the bulk of Linde's protocols used the tour strategy, her theory tends to assume it. Though she is not explicit on this point, I suspect that most of her claims have been validated only for descriptions using the tour strategy.

provides an anchor for deictic terms, such as `left` and `right`, which speakers use to explain spatial relationships (nobody uses, for example, compass points in describing their apartment). While my own data contradict Linde's assertion that tours always start at the front door, it is unavoidable that the selection of a place to start the tour must be made.

Third in Linde's heirarchy of planning decisions is the placement of sentence boundaries. This is a matter of "chunking" the information perspicuously. Only so much can go into one sentence; conversely, some items of information and positions in the tour require the beginning of a new sentence. For example, Linde claims that sentence boundaries are obligatory after a multiroom branch and before a branch with a major room, while they are optional before a minor room.

The fourth stage of planning involves every choice made between two sentence boundaries. These choices include tense, subject, and definite and indefinite articles.

While my own data contain several descriptions that do not seem to fit the tour strategy, I am confident that the same levels of planning are involved. Indeed, only the second level, executing the strategy, if any, need differ: the first, choosing a strategy, and the third and fourth, forming sentence boundaries and constructing the sentences, would be the same regardless of strategy.

In presenting her four levels of planning, Linde does not claim that a speaker makes all the choices necessary at one level before proceding to the next, more finely grained, one. The levels analysis is not a description of a heirarchy of planning, with an enforced sequence of decision-making. Instead, the levels represent a static description of the processes which convert knowledge to speech: the earlier levels contain decisions which are less language-dependent, while in the later levels the decisions are closer to language. There is, of course, some necessary ordering: it seems reasonable to assume that a strategy is selected before actual sentences are planned (though the strategy may subsequently be switched). However, it is also quite plausible that availability of certain lexical and syntactic constructions will motivate the choice of strategy. This will be discussed in Section 8.

It is clear that the levels of planning are interleaved. Linde's analysis has shown that hesitations—pauses in the production of language utterances—are most frequent just before difficult parts of the description, as well as at the beginning of the description itself. People are also likely to produce sentences with less dense semantic content and simpler syntactic form in these places. It would seem that speakers temporize by "thinning out" their linguistic output while they are planning the difficult bits to follow.

3 Previous Work

This section contains brief descriptions of systems which have addressed the issues of directing and controlling natural language generation.

3.1 McKeown

Kathleen McKeown, with her TEXT system [McKeown 85], accomplished early work in planning for generation of text. Her domain is a knowledge base containing descriptions of such things as ships and missiles. Her system answers questions about this knowledge base (eg, `what is a ship?`). TEXT is divided into two components, the *strategic* and the *tactical*, which decide, respectively, what to say and how to say it. [2]

The strategic component constructs a *relevant knowledge pool*, a subset of the knowledge base (KB), selects a strategy which encodes *rhetorical techniques*, and employs an *immediate focus* method, based on Sidner's [Sidner 79] model of focus. McKeown's analysis of paragraph-length texts written by people indicated to her that rhetorical techniques used to achieve a particular communicative goal (such as definition or comparison) tended to cooccur. She reasoned that an effective method for having a generator produce coherent text would be to aggregate these techniques into *schemata*. Selectiong a strategy for answering a question becomes a matter of selecting an appropriate schema and instantiating it. This schema instantiation can be done recursively: instantiating a field of a schema may require the instantiation of another schema.

Schemata interact with the focussing mechanism to make suggestions for how the tactical component realizes is output. For instance, the current focus may suggest the use of a passive verb rather than an active one.

Schemata seem to work well in constrained generation tasks. However, they are not very generalizable: the methods for organizing the underlying representation are to a large extent precompiled and prepackaged.

3.2 Paris and McKeown

More recently, McKeown and Cecile Paris [Paris 86] have been working in the context of a system with more inherent flexibility. The domain of this system, RESEARCHER, involves understanding descriptions of complex physical objects, found in patent abstracts and encyclopedia articles.

Paris and McKeown distinguish *declarative* and *process* strategies for getting information from the KB. The former strategy structures knowledge in ways determined by the strategy, while the latter strategy produces text which reflects structure in the knowledge itself. The system should be able to switch between strategies as appropriate.

The process strategy follows causal links in order to trace the process of the object described. For instance, if the object is a telephone, the process might be the mechanisms which transmit sound to and from the receiver. The strategy selects goal and start states of the process and traverses the causal links between them, only going off the direct path if the focus warrants the digression in order to provide an analogy or illustrate an important side effect.

[2]Currently, the tactical component for TEXT is Mumble.

Paris and McKeown point out the similarity between their concept of a process strategy and Linde's tour strategy. Indeed, both strategies attend to the interaction of some animate force and the static objects in the KB. The concept of a process strategy opens up McKeown's system to allow production of more varied sorts of text.

3.3 Bienkowski

Marie Bienkowski's Extemper [Bienkowski 86] is the latest in a family of AI programs whose domain is giving directions in a city (she also handles explanations of requirements for an undergraduate major in computer science.) Clearly, navigating through a city involves many of the same issues, such as physical relationships among objects, as does describing apartments. Bienkowski's system relates text planning to traditional AI planning, in this case, planning how to get from point A to point B.

Bienkowski claims there are several ways to linearize the underlying KB for expression in language: exploiting an ordering already in the KB; deriving an ordering; and imposing an ordering. A derived ordering is one that can be obtained from an intermediate state of an underlying AI planner. Bienkowski gives the apartment tour strategy as an example of an imposed ordering.

3.4 Conklin

The closest ancestor of the work with Apt is undoubtedly Jeff Conklin's GENARO [Conklin 83]. GENARO directed descriptions of photographs of outdoor scenes, particularly houses, for generation by Mumble. After examining people's paragraph-length descriptions of such scenes, Conklin constructed a production system which, given a represenation of a scene, used local rhetorical principles and the notion of *salience* to produce a description. Conklin felt that the sorts of physical objects found in outdoor scenes have inherent salience, which determines when they are mentioned or whether they are mentioned at all.

Unmentioned salient objects are stored on a list in order of salience, from highest to lowest. GENARO iteratively proposes its rhetorical rules for constructing a paragraph. When one rule fires (its conditions are satisfied), it is implemented. The unmentioned objects are used in order unless they are extracted from within the list by virtue of a salient relationship with the object currently being described. When this happens, the object is included in the realization specification (rspec) of which the current object is the head. When an rspec reaches a certain weight, it is sent off to Mumble to be realized, and a new one is started.

The concept of salience is GENARO's important contribution to the field. However, in the more complex domain of apartment layouts, in which more than static scenes are described, I feel that salience will not be a powerful enough criterion for choosing what to say next.

4 The Underlying Knowledge Base

We are writing a system for directing generation, not a knowledge representation. With this in mind, we want to be judicious in our expenditure of effort on the underlying representation. Thus, we want to make it as easy on ourselves as possible to construct a versatile system, while not grossly violating our intuitions about the nature of knowledge of apartment layouts. We are not in any position to make claims as to the psychological validity of our representation. As a research group, we do not have the tools available to do a thorough analysis of our subjects' protocols; furthermore, the state of psychology is such that we still would have no clear idea about what people were thinking.

4.1 The Representation

The knowledge base (KB) will be simple from Apt's point of view. The easiest situation for Apt is for everything it wants to know about to be a first-class object. A first-class object is anything that can be returned by a function, that is, a data structure that can be manipulated as an individual, independent of its context. Thus, a first-class object is directly accessible. A Lisp *defstruct* is a first-class object, but its slots are not. In our Lisp implementation, we use *flavors* to construct the KB. [3]

4.2 The Represented

The KB will consist of everything (in the ideal limit) which could be said about the house or apartment in question. All data in the KB—physical *objects*, *properties*, and *relations*—is explicitly represented, and will be refered to here as *noumena*.[4] This tripartite classification is similar to that used by Conklin in GENARO [Conklin 83]. The KB itself is called Noumena.

At a first approximation, we will represent: the house; the rooms; the rooms' relations to the house; the rooms' relations to each other; rooms' properties—name, function, hallway/major/minor; objects in the rooms and their appropriate relations; walls, corners, doors, windows, closets; positions in rooms; viewpoints in rooms; and paths through rooms.

We will not incorporate an explicit heirarchy, such as house dominating rooms dominating objects. I suspect such a structure may fall out of the model. If it does, this will suggest that the structure is inherent; if we build in the structure a priori, we most likely will never know whether the structure is only there because we put it there.

We have been very careful only to represent as much as is necessary to replicate our protocols. To add any further structure to Noumena would be making an implicit prediction about cognitive organization that we are not in a position to make.

[3]The implementation is in Symbolics Common Lisp and runs on Symbolics 3600s under Genera 7.0.

[4]*Noumenon*: a thing-in-itself, independent of sensuous or intellectual perception of it.

Viewpoints and paths are both structured aggregates of noumena, what we term *second-order noumena*. A *viewpoint* is composed of a *position* plus *points of interest*—noumena which can be seen or in some other sense reached from the position. Viewpoints are related to each other through either the relations of their positions or the relations of their points of interest.

For example, one viewpoint, **looking-into-the-kitchen-from-the-outside-entrance**, may have **outside-entrance-to-kitchen** as its position and **kitchen**, **stove**, **refrigerator**, and **sink** as its points of interest. Sharing a proximity relation with **outside-entrance-to-kitchen is next-to-kitchen-sink**. This position is part of the viewpoint **looking-out-over-kitchen-sink**, which includes as points of interest **kitchen-window**, which has property **picture-window-ness**. Properties, as first-class objects, will be shared; that is, there will be only one instance of, eg, **picture-window-ness**, and this property will be connected to all objects which have it. Sharing the property **picture-window-ness is living-room-window**, the position of the viewpoint **looking-out-through-living-room-window**.

Thus, a section of Noumena can be traversed by these three viewpoints, the first step through shared properties of the positions of **looking-into-the-kitchen-from-the-outside-entrance** and **looking-out-over-kitchen-sink**, the next step through a shared property of points of interest of **looking-out-over-kitchen-sink** and **looking-out-through-living-room-window**. An alternative traversal can go directly from **looking-out-through-living-room-window** to **looking-out-over-kitchen-sink** through the property **picture-window-ness**.

Another aggregate of noumena is a *path*. A path consists of two or more positions and their relations. A path may possibly include other things, most likely physical properties of rooms or house. A tour description of a layout will incorporate strategies using paths as a means of getting from one place to the next.

5 Mumble from Apt's Point of View

For Apt's purposes, there are three important features of Mumble: Mumble's indelible, "real-time," left-to-right execution; Mumble's expected input, the stream of *specifications*; and, in particular, the *context characteristics* of these specifications. Apt understands specifications and their assembly from various parts, because that is the language in which it speaks to Mumble. Much of this actual assembly knowledge is mechanical and doesn't involve interesting work on Apt's part. Apt's real job is choosing what will go into the specifications.

5.1 Left-to-right Execution

Mumble builds a surface-structure tree and traverses it left-to-right to produce output text. But we need not feel constrained to give Apt a similar structure. Intuitively, while actual speaking seems an indelible "real-time" process, deciding what to say seems to happen more in chunks: we feel ourselves periodically stopping or slowing down in order to think of what comes next. If indeed

Apt's processing and Mumble's generating are asynchronous, then it little matters whether they operate under differing forms of control.

If Mumble has nothing to do because Apt is still working, this is entirely appropriate, and would constitute a real pause in the discourse. Apt's operation is in some sense indelible and left-to-right, though in a way that is slightly different from Mumble's. Apt can delay as long as it likes before handing off a chunk to Mumble, and it can iterate over this aggregation of information as much as it likes, being free to change its mind about any aspect. But in handing over the chunk, it hands over control and that set of decisions has become indelible: Apt cannot interrupt Mumble to make a change.

5.2 Mumble's Input

Mumble expects its input to be a stream of specifications for linguistic output. Normally, in a discourse context, this will be a stream of rather complex objects called *bundle specifications* (bundles). A bundle is composed of a head specification which can be either another bundle or a kernel, and any number of further specifications. The head specification is in a favored position because it is around this that the information in the bundle is built into a clause or a phrase. It is unclear whether there is any theoretical distinction to this head, or whether it is just a pragmatic allowance for the fact that Mumble has to start building the sentence somewhere. Mumble uses a formalism similar to the TAG (Tree Adjoining Grammar) formalism [McDonald 85b]. Thus, as the surface-structure tree is built, *attachment points*—places where subtrees can be knit in—are activated. It is at these attachment points that the structures for the further specifications can be placed. It is through the mechanism of attachment, then, that we can generate the following paradigm of alternatives from one bundle containing the same set of specifications.

```
(4) A bedroom is on the left.

(5) There is a bedroom on the left.

(6) A bedroom, which is on the left....
```

5.3 Context Characteristics

A *realization class* (rclass), consists of a list of *parameters* and a list of *choices*. Choices within an rclass are possible realizations which are syntactically different but, by virtue of being in the same rclass, are taken to "mean" the same thing. The arguments of a specification are mapped to the parameters of an rclass. Mumble makes a choice based on the *characteristics*—*grammatical*, *argument*, and *context*—annotating that choice.

Apt assembles each bundle of specifications for Mumble. Along with bundling together the specifications of its choice and choosing the head specification, Apt has a third medium through which to direct Mumble's generation: the context characteristics. The context characteristics allow Apt to suggest a particular construction for the phrase. The context characteristics are essentially consistency tests which check parameters set by Apt. Apt can set these parameters to direct Mumble's choice in an rclass: if Apt prefers a Apt::restrictive-attribute to be a *restrictive-relative-clause* rather than a *prenominal-adjective*, then it will indicate this through the context characteristics.

In her work, Linde posits a scale of sentencehood: sentence, wh-relative, that-relative, and reduced relative (such as prepositional phrase). Linde makes the strong claim that when introducing two rooms of equal importance, speakers never place them in constituents of unequal rank. Thus, though the following is grammatical, it is not used because living room and bedroom are both major rooms.

```
(7) As you come in the front door, there's a living room which leads to
the bedroom.
```

[Linde 74], p 124.

Apt will use such knowledge of rhetorical impact in directing Mumble's attachment. I hypothesize that here, as elsewhere, Apt's strong interest in how things are said will result in it directing the vast majority of the attachment decisions, leaving very little "free variation" up to Mumble.

Apt's involvement in sentence structure will have another interesting result. Its overeagerness to express something just so may force Mumble to make some mistakes. The nature of these mistakes is discussed in Section 7.2.

6 Apt as Linearizer

Language linearizes knowledge, which is inherently nonlinear. In picking and choosing what to say from the knowledge base, Apt knows it must arrange the pieces in a sequence. The goal is to communicate the information in a sensible manner, and Apt needs to impose an organization that will achieve that goal. Regardless, then, of the underlying organization of the knowledge base (for example, heirarchical or network), the planner must thread a linear path through it in order to talk about it. Many strategies have been considered for this task, for example, McKeown's work with schemas discussed earlier [McKeown 85].

6.1 Linearizing from the Knowledge Base

Schemas are a straightforward linearization strategy. The path taken through the KB is prethreaded, by virture of the slots of the schema, and all a systems needs is to know how to fill in the slots from the KB. Schemas are a good strategy for discourse which is repeated often, like explanations and stories.

General linearization tools used by people often implicitly or explicitly rely on time. Frequently a knowledge structure is mapped into a temporal paradigm though there is no inherent temporal ordering.[5] For instance, when communicating a set of instructions, if there are two steps, A and B, it needs to be conveyed whether these are consecutive or concurrent. Apt can encode this information by using Apt::and-then or Apt::at-the-same-time. This can be generalized to narratives in general and applied by analogy to static descriptions.

Imposing procedural ordering on knowledge which is not inherently procedural is another linearizing tool. People, being the intelligent agents that they are, often describe objects in terms of how they interact with them. A prime example of this method is people's descriptions of apartment layouts. The majority of layout descriptions take the form of a tour through the rooms: people use the procedure of moving through the apartment to linearize their knowledge of its structure.

6.2 Linearizing for Mumble

Within each unit it hands to Mumble, Apt has a sense of left-to-rightness. It knows, for instance, that sentences are usually most felicitously constructed with the previously given information followed by the new. Apt also has more subtle knowledge of the rhetorical effects of arranging sentence elements to provide emphasis and facilitate implications. This ordering of these elements must be done left-to-right because of the linearity of language. However, before a unit is completed, Apt's decisions are not indelible. For the sake of computational efficiency, we don't want Apt to spend a lot of effort doing and redoing, but there is nothing to prevent our leaving this option open. The important thing to bear in mind is that this reworking would occur without Mumble's knowledge or input; once Apt has given the reins to Mumble, they are very much out of Apt's hands.

6.3 Linearizing in the Domain

In analyzing our respective data, both we and Linde have found that people use a variety of strategies to organize their knowledge of layouts. Linde decided that all apparent methods were either one of two strategies: tour or map. We feel that her analysis is somewhat flawed: rather than insisting upon a bifurcation, it would be more fruitful to find finer classifications of strategies.

My data support Linde's assertion that the goal of a description (in Linde's terminology, the *minimal description*) is to mention all the major rooms and their spatial relationships to each other. But my subjects either recalcitrantly refused to pick one of Linde's strategies, or if they did, would not stick with it. What we seemingly have found instead, is that a speaker will choose a

[5]In a similar vein, it is said that many of our rhetorical connectives stem from Greek orators' use of the method of loci to memorize their orations. The method of loci involves mentally traversing a well-known building and visualizing cues to the text being memorized, placing these cues in particular positions scattered on a path through the building. Thus, we have one point "on one hand," and the next "on the other hand," while other points are "further."

strategy, stick with it until it is no longer useful, and then try another one. Here is a very clear example of a strategy switch, from tour-like to map-like. The ellipses (....) indicate pauses in the description:

```
(A) Um, and then the walkway goes up to the main entrance, which.....is
like in a miniature, small hallway between the living room and the
kitchen.  And the kitchen has sliding glass doors....and....then
there's--there's kind of a big central....room-thing, I mean, like, when
you come in....  um....um....which leads to the stairs down to the
cellar....so there's a big central thing which must be about....10 feet
across or whatever, with closets in it, that kind of separates the whole
house.  Then there's a living room on one side and a kitchen on the
other side....And then, to one side of the kitchen is your
bedroom....and then....I guess the house is in the shape....of an....L,
where your bedroom is on one of the inside legs and Sabine's bedroom's
on the other.  And the third bedroom is on the outside leg facing the
road.  And then the living room is on that leg....and then the kitchen
and the hallway and the living room make up the other....side.
```

Note that the description becomes much more fluent after the switch in strategy. It seems that the speaker was having difficulty with his original strategy, but after switching was much more comfortable. In a follow-up discussion with the subject, this impression was confirmed.

While our analysis is not yet complete, we feel that the best approach to implementing all the descriptions we have found is as follows. The knowledge base, as described above, will consist entirely of first-class objects. A strategy will consist of a traversal of part of the KB. whenever a major room has been mentioned, it will be marked. What it means to mention a room may differ from strategy to strategy. Does mentioning an aspect of a room constitute a mention, or must some critical subset be refered to?

Strategies will be defined by the objects they follow. For example, a tour strategy will try to move along and between paths. A visual description strategy will move between viewpoints. A salience strategy will move between objects which share certain features, such as the **picture-window** in the kitchen and in the living room as described above in Section 3. A prototype strategy will move between objects that are usually strongly associated, like **refrigerator**, **stove**, **sink**, and **dishwasher**.

One striking aspect of the raw data is the preponderance of disfluencies. Disfluencies may be characterized as pauses, hesitations, stutters, repetitions, filler words (eg, um), and extra introductory words (eg, and, that, so). I speculate that many of the disfluencies in descriptions are the result of the planning process pausing to switch strategies. It would seem that a fluency unit can be no larger than the run of a strategy.

7 Theses

I would like to make some specific claims about the character of Apt. [6] I believe that any planner we build must adhere to these principles. The only way, of course, to see if these principles actually hold up is to use them in implementing a planner. If we build a planner for a particular domain, we can claim that our principles work in the restricted case, and we can then investigate their global nature.

1. Apt controls Mumble; there is no two-way communication or control.
2. Many "grammatical" errors will be Apt's fault.
3. Apt will give Mumble only complete fluency units.

7.1 Apt Controls Mumble; There Is No Two-way Communication or Control

Apt should tell Mumble what to do in such a way that no querying is necessary and no argument is tolerated. This is appealing for a model which is both theoretically coherent and computationally efficient. There is an important converse of this iron control: Apt will never have to ask Mumble what it did, since Apt will have completely specified Mumble's actions (completely to the extent to which it cares, by definition).

Part of the reason that Apt should be in control is that Mumble has no memory of what it has done. At any point in its execution, Mumble's state encodes the current status of its data and what it has left to do. Once Mumble has produced output, it no longer cares about what is in the past. However, for fluency over stretches of output larger than is encompassed in a bundle, and to take care of phenomena like subsequent reference, someone has to keep track of what has been expressed. This is Apt.

Since it is Apt that has the longer attention span and larger number of things to worry about, it is a good candidate for running the show. Further, since Apt is capable of telling Mumble what to do, and Apt doesn't need Mumble to tell it anything, one-way control seems a natural choice.

One-way control is theoretically appealing for several reasons. The literature today is in strong agreement that a system capable of natural language interaction must consist of fully-integrated components. Here, of course, the agreement ends. One view is that this integration depends on common representation throughout the system, or at least as far as is at all practicable [Bienkowski 86] [Appelt 85] [Paris 86] [Jacobs 86]. Other views espouse complicated control structures, such as blackboards, demons, specialists, and other set-ups which involve a number of

[6]These claims are a modified version of the PSDA Theses composed by Penelope Sibun and Scott D. Anderson, July 1986. Many of the original Theses are no longer stated as such, but instead have permeated the work as a whole.

processes chatting about the best way to proceed. [7] One-way control helps smooth the integration of components which have different representations.

One-way control is advantageous from an engineering point of view, it provides a well-defined interface. Additionally, although this places a large burden on Apt's output to Mumble, it alleviates the need to institute a complicated mechanism of error recovery when Mumble is unable to deal with the output from Apt.

However, there are some interesting arguments for two-way control between director and generator. The major ones seem to be, first, that the generation director may inadvertently or by design underspecify its input to the generator, and second, that it may want to monitor the generator's output. In either case, Apt did not or could not specify Mumble's action to the extent to which Apt cared.

If Apt gives Mumble underspecified input, this input has gaps in it: Apt has left out crucial elements, or not attended to important considerations such as sentence length. In a case of underspecified input, Mumble would have to realize there is a problem and ask Apt to supply the missing information. While I do not feel that there is good theoretical motivation for underspecified input, the arguments for the generation controller monitoring the generator's output are more appealing.

There are some situations in which the controller may want to monitor the generator's output. Assuming that some lexical selection is left to the generator (more on lexical selection below), there may be times when Mumble is given free rein to choose between items Apt considers semantically and rhetorically equivalent, for example, **since** and **because**, which are alternates in an rclass known to the planner as **Apt::since/because**. Normally, Apt doesn't care which of these is used when. But depending on context and how careful the speech is meant to be, the system may care whether the output has any unusual, unintended properties, such as rhyming, alliteration, and assonance. People generally ignore the potential for such in their speech, but will from time to time notice they have said something like "such self-sorry-feeling stuff." In these situations, people may choose to deliberately continue the pattern (perhaps search for more adjectives beginning with a certain sound, even if this produces more adjectives than originally intended), or may immediately try to break the pattern. It is unclear who in the system should notice and take action (or decide not to). It seems unreasonable for Apt to plan everything down to the phoneme level as the overhead would be enormous. On the other hand, Mumble does not have the necessary memory. It appears that the solution in this case is some form of parsing the output, similiar to editing.

More consequential than alliteration is what is known as "opportunistic planning." Opportunistic planning is deliberately underspecifying the input to Mumble because the generation director wants to wait until Mumble reaches a certain point in its realization of the surface structure before the director makes a choice, presumably because there will be information available after Mumble's partial execution that the controller did not have before.

[7] Frankly, I harbor the suspicion that as much as from theoretically-motivated control issues, these baroque interfaces result in large part from several researchers or research groups attempting to integrate disparate systems and not being able to completely agree (the Counselor Project stumbled seriously at just such a point).

One of the concerns of Apt is certainly focus. There are two aspects to focus: the one of given information versus new; and the focus that is motivated by the relative importance of of the information being conveyed. We want Apt to direct the realization of both sorts of focus. It is certainly Apt which decides what is more and less important of what is being said (for example, in apartment layouts, which are the major and minor rooms). It should also be Apt that remembers what is old information and what is in focus.

7.2 Many "Grammatical" Errors Will Be Apt's Fault

Mumble should not make any errors by breaking on Apt's input. Further, we do not want Mumble to say anything a person would not say because Apt had not done its job for Mumble.

Having said that Apt will not cause Mumble to make mistakes, what do we have to say about speech errors? Clearly, we want a generator complex and sensitive enough to make the same kinds of mistakes that people do. I feel that some classes of grammatical mistakes can be explained by putting Mumble in the position where the only choice that it can make is ungrammatical (but still makes sense, in most cases). An example of this sort of thing is:

`(8) RSVP is generally a phrase that nobody knows what means.`

The sense of this sentence is clearly something along the lines expressed by:

`(8') (Generally) nobody knows what the phrase RSVP means.`

The speaker wanted to say two things about 'RSVP', which we could consider the topic of the sentence: that is is a phrase and that nobody knows what it means. Because 'RSVP' was the most important element of the sentence (or because it was thought of first, though this may boil down to the same thing), it is expressed first. Only then was the rest of the sentence realized, but too late to make a "grammatical" structure. Note that the insertion of a resumptive `it` after `what` does not improve the grammaticality. [8]

This example makes a clear case for allowing the controller override the generator: sometimes the importance of sentence elements as expressed by their order takes precedence over grammaticality, which must make do as best it can. Above, in Section 4 on Mumble, I explained that Apt can influence choices in the rclass through the context characteristics. Normally, Mumble is constrained in its choices by the argument and grammatical characteristics, which together insure grammaticality of the output. Thus, the above sentence could not be generated. However, with a slight modification to the existing implementation of Mumble, Apt could take control of an rclass and force a choice, overriding any grammatical considerations.

[8]This example is drawn from speech error data collected by the author. This type of error is quite common, both with and without a resumptive pronoun. Interestingly, most examples involve a variation on "x doesn't know .... " in the embedded clause.

7.3 Apt Will Give Mumble Only Complete Fluency Units

This thesis was originally: "The planner will give Mumble only complete *discourse* units." Unfortunately, this phrasing did not really capture what we meant. A discourse unit, in speech, is most properly defined as a *turn* in a conversation, delimited at one end by a person starting to speak, and at the other by their coming to a stopping point by their own motivation; interruptions are disregarded (but may be considered, in their own right, turns of the interrupter). A discourse unit embodies a thought in that its endpoints are defined by the speaker. But this thought could be very large, even a long extemporaneous speech. In our domain, a discourse unit is usually an entire description. Clearly, it wouldn't be very interesting or realistic for Apt to plan the entire description and hand it in one lump to Mumble. The type of thought we want to capture, then, is of a somewhat smaller size.

Linde suggests that sentences are the proper unit by which to plan. "Sentence" needs to be loosely enough construed to allow for interrupted and trailed-off sentences. But even so, a sentence seems too formal a unit. As seen below, what I would like to call a *fluency unit* can be either larger or smaller than a sentence.

For a fluency unit, I would like to use a more pragmatic notion of something that is between a discourse unit and a sentence in size, or even something smaller than a clause. Listening to a layout description, one gets a clear sense of "starting and stopping" within a description. These disfluencies appear entirely appropriate markers for planning units, at a first approximation.

Following are two excerpts from descriptions.

```
(B) The....doorway on the lefthand side, as we're facing out towards the
street....is....leads to a very short hallway, in which we have....the
side door, and opposite that the door that goes down to the basement.
Then in the righthand....doorway, we have like a um....we have a
hall---large hallway that leads into the kitchen, and at right angles
the smaller hallway that leads to the bedrooms.  Then, in the
kitchen....there's a large window which faces the backyard, with two
smaller windows directly flanking it.  And....if we're facing....towards
the backyard now, on the righthand side is....a sliding glass door,
and....a few feet from that is a smaller window, towards the living
room.
```

```
(C) It's built around that, um, and so off to your right is the dining
room-kitchen,....and then, diagonally up to your right,....as you go
past the kitchen, is one of the bedrooms....ah, you go around the
chimney to your left and....in the front corner of the house is the
living room,....and, if you keep going to your---as you go around to t
he left if you go right there's a closet to your left, and on the right
is....ah, another bedroom....and on the left is another bedroom and
straight ahead is the bathroom.
```

8 Lexical Selection

A current area of research is lexical selection: specifically, where in the generation process it takes place. In our system, this question boils down to when and whether Apt decides to use a particular word, phrase, or syntactic construction. In this case, we are talking about an item as it would appear in the word stream (with possible morphological modification). This means that, for some reason, Apt has not selected Apt::picture-window but `picture window`. However, where there is only one possible lexical item, such as `left`, it is theoretically moot whether Apt chooses Apt::left or `left`.

Apt may want the ability to choose a lexical item early on in its process, while linearizing and creating units, rather than late, within the unit, when it has already chosen what part of the KB it is talking about. Why? If Apt is mindful of whether it has a word for a KB object, or, more interestingly, an aggregation of KB objects, this may affect the choice of strategy within that portion of Apt's direction.

Specifically, a strategy Apt may employ, the possible-word strategy, would look at some interconnected subpart of the KB and see if there was a single lexical entry for the aggregate. In principle, if a single term can describe an aggregate of objects, then the fluency unit can be built around the term. Obviously, in this situation, Apt needs to know that a word is available, to have its fingers on the word, in order to proceed with such a strategy. Here is an apparent example of such a strategy at work in one of my protocols.

```
(D) Then, we go to that smaller hallway on the righthand side, to get to
the other two bedrooms and bathroom.  **Material deleted.** ....if we're
still facing in the hallway the same direction in which we came in, on
the righthand side we go into Sabine's room.  **Material deleted.** So,
we leave Sabine's room and we walk....across the hall, then, across the
T,....the top of the T, which would be....um, the small---a small
hallway from, well, let's see, how should we describe this ....The
hallway to the bedrooms....in relationship to that wide hallway between
the living room and kitchen is like a T.  There's a short bit....and
then there's a top of the T, and that's the---at one end of the T is
Sabine's room, at one end of the T is my room.
```

Here, the speaker suddenly hit upon the word T. We see her redo her description of that part of the layout. Since we see both (part of) a non-T and a T description, we can examine the effect of the lexical item. We can also notice, in this example, that the T metaphor was more of a hinderance to clarity than a help.

There is a inverse of the above situation: when *no* word can be found to adequately describe a collection of objects. The following excerpts from the protocols below describe the same "funny-shaped hall/room-thing."

```
(E) there's one thin hallway, leads from the kitchen into the living
room, then there's a wider one, that leads from....the kitchen to the
living room....about three times as wide.

(F) and we walk down that....wide hallway, which is almost a room in
itself.
```

People sometimes have to work harder to describe things for which there is no neat description. It is noteworthy that the other protocols avoid mentioning this area of the house.

9 Conclusion and Further Research

This paper has explored a broad range of issues and touched only briefly on many of them. I have presented a domain for controlling and directing generation and have discussed some of the considerations I will bring to bear in building Apt to create layout descriptions for Mumble.

Current work has been two-pronged. On one hand, we are trying to push as many fluency units through to Mumble as possible, to test Mumble's flexibility and, more importantly, the robustness of the interface between Apt and Mumble.

The other line of research I am pursuing involves theoretically complex issues of how people describe the world. *Deictic terms*, words like `left`, `right`, `behind`, `above`, and `below`, which have different reference depending on who the speaker is and her relationship to objects under discussion, are important in a situation that heavily involves describing physical objects and giving directions. These very common terms are very difficult to define and represent. Discovering an elegant and robust implementation of deixis would constitute a real contribution to the literature of the field of directing and controlling natural language generation.

10 Acknowledgements

I thank David McDonald for advising; Kee Hinckley and tanstaafl for pushing; Peri Tarr for pulling; Alison Huettner and John Brolio for discussing; Marie Vaughan for commenting; Brian Holt for staying up all night; Stephen Harrison for being logical; and, of course, the other half of PSDA.

References

[Appelt 85] D. Appelt, "Planning English Referring Expressions," *Artificial Intelligence* 26, 1985, 1-33.

[Bienkowski 86] M. Bienkowski, *A Computational Model for Extemporaneous Elaborations*. CSL Report 1, Cognitive Science Laboratory, Princeton University, 1986.

[Conklin 83] E. Conklin, *Data-Driven Indelible Planning of Discourse Generation Using Salience*. COINS Technical Report 83.13, University of Massachusetts, 1983.

[Jacobs 86] P. Jacobs, *Knowledge-Intensive Natural Language Generation*. Technical Report 86CRD026, GE Corporate Research and Development, 1986.

[Linde 74] C. Linde, *The Linguistic Encoding of Spatial Information*. Doctoral Dissertation, Columbia University, 1974.

[McDonald 82] D. McDonald, *Description Directed Control: Its implications for natural language generation*. Department of Computer and Information Science, Univeristy of Massachusetts, 1982.

[McDonald 85a] D. McDonald and J. Pustejovsky, "Description-Directed Natural Language Generation." In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence,* Los Angeles, 1985.

[McDonald 85b] D. McDonald and J. Pustejovsky, *TAG's as a Grammatical Formalism for Generation*. Counselor Project Technical Memo CPTM #5, Department of Computer and Information Science, University of Massachusetts, 1985.

[McDonald 86a] D. McDonald, *Natural Language Generation*. Counselor Project Technical Memo CPTM#12, University of Massachusetts, 1986.

[McDonald 86b] D. McDonald, M. Vaughan, and J. Pustejovsky, "Factors contributing to efficiency in natural language generation." In G. Kempen ed., *Natural Language Generation: Recent Advances in Artificial Intelligence, Psychology, and Linguistics,* Martinas Nijoff Publishers, Dordrecht, The Netherlands, in press, 1986.

[McDonald 86c] D. McDonald and J. Pustejovsky. "The COUNSELOR Project at the University of Massachusetts." In *The FINITE STRING Newsletter, Computational Linguistics* Vol 12 #2, April–June 1986.

[McDonald 86d] D. McDonald, *Natural Language Generation: complexities and techniques*. Counselor Project Technical Memo CPTM #14, Department of Computer and Information Science, University of Massachusetts, 1986.

[McKeown 85] K. McKeown, "Discourse Strategies for Generating Natural Language Text." *Artificial Intelligence* 27, 1985, 1-42.

[Levelt 82] W. Levelt, "Linearization in Describing Spatial Networks." In S. Peters and E. Saarined eds, *Processes, Beliefs, and Questions,* 199-220, 1982.

[Paris 86]	C. Paris and K. McKeown, *Discourse Strategies for Descriptions of Complex Physical Objects*. Department of Computer Science, Columbia University, 1986.
[Sibun 86a]	P. Sibun and S. D. Anderson, *The PSDA Theses*. Unpublished document, Department of Computer and Information Science, University of Massachusetts, 1986.
[Sibun 86b]	P. Sibun and S. D. Anderson,. *Definition of a Planner*. Unpublished document, Department of Computer and Information Science, University of Massachusetts, 1986.
[Sidner 79]	C. Sidner, *Towards a computational theory of definite anaphora comprehension in English discourse*. Ph. D. Dissertation, MIT, 1979.