Planning in AI and Text Planning in Natural Language Generation

Jong-Gyun Lim
Columbia University
Department of Computer Science
New York, NY 10027

12 July 1992

CUCS-038-92

Abstract

Researchers in natural language generation are becoming increasingly aware of the similarities between the task of planning the content and structure of the natural language text and that of other AI planning tasks. The problem of text planning and other AI planning problems have been studied separately from each other, and while many overlapping results have been found, they have not been shared as often. The purpose of this paper is to explore the similarities between the two areas of research by reviewing the problems and solutions of each in order to gain a better insight into how needs in one area can benefit from results in another.

1. Introduction

Intelligent agents do not act by instinct or stimulus-response alone. They have goals, and before acting, they usually form a set of actions (or plans) that they follow in order to achieve their goals. Designing systems which can automate the generation of plans for a given goal has been the purpose of the long standing field of artificial intelligence (AI) known as *planning*. Over 30 years of research on planning, a large number of techniques have been introduced for progressively more challenging planning problems.

Although the domain of planning problems has evolved from marginally interesting toy domains to a lot more realistic domains involving complex real life actions and objects, yet another dimension of problems are introduced when communicative acts are involved. Undeniably, communication plays an important role in real life activities, and intelligent agents do plan for the communicative acts just as they do for other domain acts. However, researchers in planning have sidelined the problems involving communicative acts, perhaps because they did not want to deal with specialized knowledge and techniques required to handle them.

On the other hand, it is the purpose of another field of AI known as *natural language generation* to design systems which can automate the generation of communicative acts. *Text planning* is a task of natural language generation that plans for ‘‘what to say’’ (as opposed to ‘‘how to say it’’). Therefore, text planning is exactly analogous to AI planning except that it plans for communicative acts rather than domain acts. In addition, a natural language generation system is ultimately an integral part of an intelligent system that constructs a natural language text in order to meet specified communicative goals. Therefore, the process of language generation should be governed by the same principles that govern the behavior of the intelligent system that embodies it. Thus, we believe that there exists a close relationship between AI planning and text planning for natural language generation. Not surprisingly, techniques similar to those used by the AI planning systems are also found in text planning systems despite the fact that text planning researchers have worked separately from AI planning researchers.

There is, however, one major difference between the text planning problem and a typical AI planning problem. The difference is that a text planning problem is large and ill-defined whereas an AI planning problem is typically small and well-defined. Of course, there are problems in AI planning that are defined in a complex and underspecified domain. However, the difference is more inherent in the nature of the text planning problem itself. First of all, virtually all the usual elements of planning in a typical AI planning problem is at best vaguely defined for the text planning problem. Different text planning systems assume different definitions of goals, plans, operators, and the state of the world. One might attribute the reason to the fact that researchers in natural language generation did not view text planning as a kind of AI planning task and thus did not formalize the problem from the perspective of AI planning.

Because the text planning problem is large and ill-defined, it is not clear exactly what the task of text planner is. It might sound sufficient to define it as the task of deciding ‘‘what to say’’. For example, in TEXT, one of the earliest natural language generation system ([McKeown 82]), McKeown distinguishes the *strategic component* from the *tactical component* such that the former (the text planner) is responsible for determining the content of the text described in an internal representation, and the latter is responsible for converting that representation into English sentences. Similarly, a majority of other natural language generation researchers classify the task of text planner to be deciding ‘‘what to say’’, and that of the *linguistic realization* component to be deciding ‘‘how to say it’’.

However, depending on one’s perspective, much of the task of ‘‘how to say it’’ can also be categorized as the task of ‘‘what to say’’. In order for a text planner to decide ‘‘what to say’’, it needs to plan for various other tasks. To see this, let’s say that text planner has decided to communicate the following message to the hearer:

(REQUEST *hearer* (TRANSFER (block *A*) (from-loc *x*) (to-loc *y*)))

which means that the speaker wants the hearer to move block *A* from location *x* to another location *y*. Now, all of the following English sentences achieve the same purpose of communicating the same message but with different effects.

- Sentence 1: Move block *A* from *x* to *y*.
- Sentence 2: Could you move block *A* from *x* to *y*?
- Sentence 3: Would you please move block *A* from *x* to *y*?
- Sentence 4: Move the red block onto the blue block.
- Sentence 5: Place the red block onto the blue block.
- Sentence 6: Onto the blue block place the red block.

Different decisions give raise to different sentences and thus different effects. For example, the decision to use an interrogative sentence instead of an imperative sentence results in a more gentle request to the hearer (Sentences 1 vs 2); the decision to choose the word ‘would’ over ‘could’ and adding the word ‘please’ makes it a more formal request (Sentences 2 vs 3); the decision to ellaborate the block *A* as the ‘‘red block’’ and the location *y* as ‘‘onto the blue block’’ helps the hearer to easily find the location *y* (Sentences 3 vs 4); the decision to choose the verb ‘place’ over ‘move’ makes more sense when location *x* is, for instance, in the hand of the hearer (Sentences 1 vs 5); the decision to choose the constituency ordering in Sentence 6 gives the effect of emphasizing the location *y* rather than the block *A*.

Therefore, there are researchers in natural language generation who include as part of the task of text planning the task of syntactic selection, constituent ordering, and lexical selection in addition to the content delimitation and text structuring. Appelt, for example, considers ‘‘what to say’’ and ‘‘how to say it’’ to be two sides of the same problem, and he employs planning at all stages of the generation process in KAMP ([Appelt 85]).

Traditional AI planning problems, on the other hand, had very well defined tasks. Besides having only limited domain actions, their model of the world was often grossly simplified for the purpose of testing various planning techniques developed for their planning tasks. As a result, researchers in AI planning were successful in developing many good planning techniques and paradigms for domain problem solving. Although they have sidelined the problems of planning for communicative acts, recently, they have begun to explore the problems typically found in multi-agent planning settings where agents in the world collaborate together through communication. Such work is useful for work in natural language communication and thus it is encouraging. In fact, researchers in natural language understanding, in particular those who work on plan recognition (e.g. [Grosz & Sidner 90], [Pollack 91]) are investigating the possibility of extending results from that line of AI planning research.

It is the purpose of this paper to identify the components of text planning and view them from the perspective of AI planning research using AI planning terminology. Our aim is to gain better understanding of the similarities between the two areas of research and find opportunities to take advantage of results in one area and apply it in another.

The remaining sections of the paper are organized as follows. Section 2 briefly covers the major planning techniques developed in AI planning research. Section 3 introduces the place of text planning within natural language generation and outlines the elements of planning in text planning. In section 4, we examine the use of planning in various natural language generation systems while comparing the goals, plans, and planning techniques used in those systems with those used in AI planning systems. We attempt to clarify the task of text planning and to highlight the potential problems and notable results in the text planning research. Finally in section 5, we discuss recent work on collaborative planning in AI planning and natural language understanding research in order to predict future directions for text planning research.

2. Planning in Artificial Intelligence

In this section we review work on planning in the field of artificial intelligence (AI) in order to establish the ground for comparison with the work on planning in the field of natural language generation in latter sections.

Over the past 30 years of research in AI planning, many problem solving techniques have been developed and improved, and the scope and scale of the planning problem have been growing in proportion. So much work has been done in the field of AI planning that it is impossible to summarize all the results in this section, not to mention comparing them all with the planning work in natural language generation. Therefore, we select highlights of the planning techniques that have been the focus of AI planning research, and give a flavor of what each of the techniques are, how they are used to solve planning problems, and what kind of planning problems they are appropriate for. Before we do that, we first introduce the components of a typical AI planning task briefly so that they can be compared with the components of planning in the task of text planning in natural language generation. (See Section 3.)

Whenever appropriate, we includes references to text planning systems that use comparable or related planning paradigms, in particular those systems that are discussed in later sections. However, more detailed analogies are left to be explored in later sections where the text planning systems are discussed in detail also.

2.1 Components of Planning

The task of a typical AI planner is to generate an ordered set of actions for agent(s) such that the agent(s) can execute the actions to change the state of the world from an initial state to the goal state. Input to a planner is an initial world state, a set of actions called *operators* that are applicable to the world, and the desired world state called the *goal* state. The output of the planner is an organized collection (often a directed acyclic graph) of operators called a *plan*.

The model of world states and actions are typically well defined. They have been simplified and include predictable features in order to control the potentially enormous search space of possible plans. For example, the block-world as shown in Figure 2-1 typifies the sample domain for much research in traditional AI planning.

Another noticeable characteristic of domains for planning systems is that, like the block-world, the model of the world is almost always a physical model of the world. This is due to the nature of the application task of traditional planning systems which are usually robot control or navigation problems. Therefore, when planning is applied to the domain of natural language processing, where the problems involve communicating abstract thoughts, beliefs, and intentions, the first step taken is the design of formalism that covers both physical and communicative acts uniformly.

2.2 Planning Techniques

The process of connecting the input and output as described in the previous section is called plan *generation*, plan *synthesis*, or plan *construction*. A system or algorithm that carries out this process is a *planner* and the techniques used by a planner to achieve the goal (or to solve the problem) are *planning techniques*.

As early as 1963, a problem solving program called GPS [Newell & Simon 63] is introduced that was able to select operators needed to make the transition of state from the initial state to the goal state. Since then, various kinds of planners came into existence that introduced new architectures and techniques to remedy the limitations of each proceeding planning system. We review each of those architectures and techniques and discuss the issues that they address.

2.2.1 Non-hierarchical Planning

Systems like STRIP [Fikes & Nilsson 71], HACKER [Sussman 75], and INTERPLAN [Tate 75] are the earliest planning systems. They use techniques similar to those used in GPS. Basically this involves finding a sequence of operators (*i.e.* a plan) such that when applied in sequence to the initial state, the goal state results.

For example, in order to achieve the goal state in the Block-world (figure 2-1), a non-hierarchical planner repeatedly picks an applicable operator from the set of operators (there are four in the example) until the sequence at the bottom of figure 2-1 is found. Usually some heuristics are used in selecting an operator so that the choice contributes to achieving the desired goal state. However, as the problem becomes more complex, the search space becomes exponentially large, and it becomes

State of the World

A state of the world is represented by objects and predicates describing the state of the objects. A typical block-world consists of objects Block A, Block B, Block C, and a Robot-Arm, and predicates like the following:

- ON(A, B) — Block A is on block B.
- ONTABLE(A) — Block A is on the table.
- CLEAR(A) — There is nothing on top of block A.
- HOLDING(A) — The arm is holding block A.
- ARMEMPTY — The arm is holding nothing.

A sample world and its description using the predicates looks like the following:

```
    _____
   |     |                                      ON(C, B) &
   |  C  |                                      ONTABLE(B) &
   |_____|        _____                         ONTABLE(A) &
   |     |       |     |                        CLEAR(C) &
   |  B  |       |  A  |                        CLEAR(A) &
___|_____|_______|_____|____                    ARMEMPTY
```

Operators

A typical block-world consists of the following set of operators that are actions that can be performed by a Robot-Arm to the block(s):

- UNSTACK(A, B) Pick up block A from its current position on Block B.
- STACK(A, B) Place block A on block B.
- PICKUP(A) Pick up block A from the table and hold it.
- PUTDOWN(A) Put block A down on the table.

Goals

A goal state is an instance of the possible world states achieved by applying a sequence of operators. For instance, here is a state of the world that represents a goal state:

```
            _____
           |     |
           |  A  |
           |_____|                              ON(A, B) &
           |     |                              ON(B, C) &
           |  B  |                              ONTABLE(C) &
           |_____|                              CLEAR(A) &
           |     |                              ARMEMPTY
           |  C  |
___________|_____|_________
```

Plan

The following sequence of operators is considered a plan that changes the state of the world from the sample world to the goal world above:

```
UNSTACK(C, B) --> PICKUP(B) --> STACK(B, C)
--> PICKUP(A) --> STACK(A, B)
```

Figure 2-1: Components of Planning in AI: The Block-world Example prohibitively expensive to search for the desired plan even with the best heuristics. Planning for real world problems like, *going to a conference* is, thus, practically impossible; imagine planning with operators like open-the-door-to-get-out-of-the-room, pay-the-cab-driver, or check-the-seat-number-on-the-plane. The problem is that unnecessarily large number of operators should be searched when only a small subset of them are relevant at a given point.

2.2.2 Hierarchical Planning

In order to remedy the problems associated with the non-hierarchical planning, NOAH (Nets Of Action Hierarchies) [Sacerdoti 77] and ABSTRIPS (Abstraction-Based STRIPS) [Sacerdoti 74] planning systems represent a plan as a hierarchy of plans where a higher level plan is an abstraction of the plans at the lower level. Lower level plans describe the detailed action steps needed to achieve the higher level goal. In addition to this plan decomposition hierarchy, some systems also represent the actions and objects involved in a plan using subsumption relations in order to achieve the effect of having an even finer degree of granularity among the actions.

For the *going to a conference* example mentioned above, a hierarchical planner would represent an abstract plan like make-a-trip-to-a-destination that subsumes other abstract plans like take-a-plane-to-a-destination, drive-to-a-destination, etc. The plan take-a-plane-to-a-destination could be further specified into a sequence of abstract plans like get-out-of-house, take-a-cab, and take-a-plane. Likewise, any other plan on the top or intermediary levels would be fleshed out either by less abstract plans or by a sequence of plans that achieves it.

Given this kind of plan decomposition hierarchy, a hierarchical planner generates a plan by first constructing an ‘‘outline’’ of a plan using only those abstract plans on the higher levels of the plan hierarchy. For example, the following is a plan that achieves the goal of going to a conference:

1. Register.
2. Make travel reservations.
3. Make a trip to the conference.
4. Attend the conference.
5. Make a return trip.

Although this kind of abstract plan solves the problem completely, each step in the plan might be too abstract to be performed by the agent who will execute the plan. Therefore, a hierarchical planner fleshes out the details of each abstract plan with a set of lower level plans until all the abstract plans are expressed in terms of the concrete plan operators that can be carried out by the intended agent. The actual process of finding a sequence of appropriate abstract plans at each stage of the planning cycle is similar to the planning process used in nonlinear planning, but the search space is greatly reduced because rather than dealing with all the lower level plan operators, a hierarchical planner deals with only a limited set of abstract plans at a time.

2.2.3 Skeleton Planning

Skeleton planners use slightly different methods to reduce the search space. Like hierarchical planners, skeleton planners construct a plan by first constructing the skeleton of the plan and filling in the details with lower level plans. However, a skeleton planner does not construct a plan by generating them from a hierarchy of plans, but it produces one by *selecting* and *instantiating* one of the *pre-stored* plans. A pre-stored skeleton plan consists of the outline of solutions for many different kinds of problems and it is refined by instantiation of problem solving operators containing information for solving a specific or common problem in detail; these operators are found by the skeleton planner from the pool of domain specific knowledge.

For example, a skeleton planner would solve the problem of going to the ACL-91 conference held at Berkeley by searching for its library of plans and finding a pre-stored plan labeled, going-to-a-conference, say, which looks like the following:

{Pre-register} → Reserve-air-transportations → Reserve-a-hotel / Reserve-a-dormitory → Take-a-cab-to-the-airport / Take-a-bus-to-the-airport → Take-the-plane → Check-in → {Register} → . . .

where the curly brackets are used to indicate optionality, and the slash (‘‘/’’) is used to indicate choice. Next, the skeleton planner would instantiate this abstract plan with domain specific operators like Take-Olympia-shuttle-bus-to-Newark or Reserve-Continental-Airline-to-San-Francisco.

A system that adopted this skeleton planning technique is Stefik’s MORGEN system [Stefik 80] which is a knowledge-base program that assists molecular geneticists in planning experiments. In MORGEN many plans with various levels of abstractions and details are stored as *skeleton plans*, also called *strategies*, for solving problems.

2.2.4 Least-commitment Planning

The instantiation process during skeleton planning involves searching through vast amounts of domain specific knowledge until the right operator is found. This itself can be a big problem especially when there are conflicts among the instantiated operators. For example, buying the ticket for the Olympia shuttle bus to Newark would be a silly thing to do if the flight to San Francisco is scheduled to leave very early in the morning when there is no scheduled shuttle bus. Similarly, planning to go to Newark airport would be inappropriate if the plane departs from LaGuardia airport.

Stefik recognizes this problem as a planning problem in its own right and introduces the concept of *meta-planning* that inserts an additional process between the skeleton plan selection process and the operator instantiation phase called the *design phase*[1] During the design phase, MORGEN makes decisions about how the abstract steps in the skeleton plans are instantiated by which operators. One of the techniques it uses in selecting the operators is called *least-commitment planning* which makes sure that those operators that require least constraints are instantiated first. As a result it rarely commits itself to a decision that it must undo later. For example, since buying the ticket for the shuttle bus requires more constraints than making a plane reservation, because the time and destination of the shuttle bus depends on the plane schedule, the incorrect decision to buy the ticket for the shuttle bus heading to a wrong destination or leaving at off hours can be avoided. This planning technique is also known as *progressive refinement planning* [Stefik 81].

2.2.5 Opportunistic Planning

Opportunistic planning [Hayes-Roth 85] takes an approach that is totally different from the goal-driven planning approaches taken by other planning techniques discussed so far. A typical opportunistic planner constructs a plan by alternating through two distinctive phases. The first phase is an observation phase where different problem solving components communicate with each other via a *blackboard* to identify the constraints and operations that need to be performed. The second phase is a decision making phase where an operator is selected from the set of operators. The selected operator relaxes the constraints or contributes to the operations identified by the observatory phase. After an operator is selected, the observation phase takes over and each of the problem solving components re-evaluates the situation to post another constrained operation. This two phase

[1]Actually, he names them strategy space, design space, and instantiation space.

process continues until there are no more operations that need to be performed. The sequence of decisions made through this process is a plan that achieves the goal of the planner.

A plan constructed by the opportunistic planner may follow an orderly path and produce a neat top-down expansion. However, it is possible that during the planning process, some of the initial sequence of actions may have imposed limitations on the latter actions, and thus it is possible that a less than ideal plan is generated.

2.2.6 Reactive planning

Both hierarchical planners and skeleton planners discussed above as well as all the other planners mentioned up to now can be categorized as *deliberative planning systems*. In a deliberative planning system, a plan for completing an entire task is constructed prior to the execution of any action. However, for certain tasks, planning and execution of a plan must be interleaved because of the following characteristics of the real life planning problems:

- A planning system with limited resources must decide when to start planning, when to stop planning, and when to act; if a rock is falling over one's head, he would rather take any direction to get out of the way than stand there until he finishes planning for the next move he will make once he gets out of the way.
- The state of the world after execution of an action necessitates new goals or eliminates execution of already planned actions.
- The order of execution of actions can be altered due to the changed environment; some situations require immediate attention and rapid action.
- During the course of execution of actions, the initial specification of the goal may be changed requiring replanning of actions; for example, in our going to a conference example, if suddenly the Olympia shuttle bus stops running on the planned date due to their bankruptcy, then it would be more desirable to make an alternative plan, like taking a cab, than to declare goal failure and not attend the conference at all.

In order to deal with these problems, *reactive planning* systems (*e.g.*, [Brooks 86], [Agre & Chapman 87], [Kaebling 87]) take a drastically different approach to planning. They choose actions one at a time and execute an action as soon as they plan it. The choice of an action is purely reactive and it is based on the knowledge base that contains information about what actions to take under what circumstances.

For example, the information in the knowledge base of Brook's mobile robot contains the ordered set of ‘‘desired class of behaviors’’ (See figure 2-2) that guides the robot to behave in certain ways in whatever environments it will encounter:

1. Avoid contact with objects (whether the objects move or are stationary).
2. Wander aimlessly around without hitting things.
3. Explore the world by seeing places in the distance which look reachable and heading for them.
4. Build a map of the environment and plan routes from one place to another.
5. Notice changes in the static environment.
6. Reason about the world in terms of identifiable objects and perform tasks related to certain objects.
7. Formulate and execute plans which involve changing the state of the world in some desirable way.
8. Reason about the behavior of objects in the world and modify plans accordingly.

Figure 2-2: Desired Class of Behaviors [Brooks 86]

Brook's mobile robot is constantly aware of these prioritized set of guidelines so that it can react to the environment changes while satisfying possibly conflicting multiple goals. Being able to *constantly* follow guidelines and goals is an important requirement in planning for natural language generation as well. Hovy [Hovy 88a] observes that unlike other goals in traditional AI planning that can be forgotten upon satisfaction, many communicative goals are persistent goals, such as ‘‘be friendly’’. It is hard to imagine how to satisfy such a goal using non-hierarchical or even hierarchical planners without a reactive planning capability.

Reactive planning and execution monitoring are two very active areas of planning research today. While many researchers are busy formalizing the theory of action, knowledge, and the temporal relations among the activities to capture and even to predict the changing environment ([Agre & Chapman 87], [Kabanza 90], [Pednault 87]), there is at least one planner that allows the agents to modify the environment in order to relax constraints in planning, [Safavi 91].

Safavi's NEGOPRO system is an automatic scheduler in the domain of software manufacturing projects that allocates resources and assigns tasks to the human developers of a project. When there is a need to reschedule, NEGOPRO initiates its *reactive schedule revision cycle*. (Figure 2-3) What is interesting about this is that during the revision cycle, a human scheduler is allowed to interact with NEGOPRO and to modify the constraints by altering the available resource capacity (or capacity constraints) or the requirement constraints of the goal product.

Figure 2-3: Reactive Scheduling [Safavi 91]

2.2.7 Increasingly Reactive Planner

As demonstrated above, reactive planners have the advantage that they can behave robustly in dynamically changing world without having to model the worlds in advance. Also they are extremely responsible because they do not need to do the combinatorially explosive search that slows down other search-based deliberative planners. However, pure reactive planners have limitations also. Since they do not maintain any model of the world or explicit goal structures, they cannot be used for solving tasks that require intensive domain knowledge. That is the reason why although pure reactive systems are often used for real time tasks like driving and walking, they are not suited for playing expert chess [Rich & Night 91].

For this reason, the *increasingly reactive planner* [Mitchell 90] is a deliberative planner which incorporates both a reactive component and a search-based planning component such that an agent in his planner, called Theo-Agent, reacts when it can, plans when it must, and learns using an explanation-based learning mechanism by augmenting its reactive component with newly formulated stimulus-response rules whenever it is forced to plan. Therefore, what starts out as a deliberative planner with a minimum reactive component becomes increasingly reactive as its learning component acquires new stimulus-response rules that eliminate the need for planning in similar subsequent situations.

Going back to our example, planning for the task of going to a conference using pure deliberative planner would be extremely difficult, if not impossible, if we were to construct a robust plan that considers all the unexpected situation that might occur (for example, the Olympus shuttle bus company going out of business). On the other hand, we would not go far with the pure reactive planner either because without the detailed domain knowledge about the way airline and hotel reservation systems work, a pure reactive planner would not generate a plan that includes a plan to make an advanced flight reservation, and by the time it finds out that there are no seats available, it will be too late to find an alternative ‘‘desired behavior’’ to achieve the goal of going to the conference. Using an increasingly reactive planner, however, we can generate a plan that takes advantage of both deliberative planning and reactive planning. Basically, the plan will be identical to the one generated by either a hierarchical or skeleton planner except that the execution of the plan will be monitored and, whenever necessary, stimulus-response rules will be triggered to deal with the unexpected events. As the increasingly reactive planner becomes more and more robust, the stimulus-response rules will contain more complex actions; at the end when it ‘‘learns’’ to become completely reactive, it is possible that the entire plan that solves the task of going to a conference will be encoded in a single stimulus-response rule so that when a need to go to a conference arises, execution of necessary actions can take place without wasting any time to plan for that task.

At the time of this survey, there is no natural language generation system that adopts this increasingly reactive planning mechanism (except for suggestions made in [Paris 91]; also see section 4.6.1 of this paper.), but it is not hard to imagine why an efficient generation system should also adopt this mechanism to be able to plan the content and surface structure of the output text when necessary, but use some sort of stimulus-response rules to retrieve pre-stored structures or even some previously generated piece of texts when the situation is similar (aka ‘‘similarity-based generation’’ to go with similarity-based learning).

2.2.8 Cooperative Distributed Planning

For a large task with more than one sentient agent collaborating together towards the completion of the task, it is necessary to coordinate the efforts of the agents in order to achieve the task while taking advantage of *distributed* information, resources, and expertise.

With the emergence of the new field in AI called *parallel and distributed AI* much research has been conducted to solve the problem of coordinating the actions of the individual agents in order for them to work together effectively. (See [Barr *et. al.* 89] for comprehensive coverage of the work up to 1989.) The variety of approaches can be grouped into the following three categories based on the nature of the distribution of the agents:

1. Centralized Multi-agent Planning: This is the least distributed kind. A single agent (master agent) makes a plan and distributes pieces of the plan to other agents (slave agents) that carry out the task and report back to the master agent. Since this kind of planner is built to avoid inconsistent conflicting actions, it is typically used in a domain where unexpected conflicts for a resource can be costly; for example, scheduling the use of airspace or machining tools. However, the master agent is required to have a substantial amount of information about the actions and agents. A lot of communication resources are consumed during the process of acquiring and updating such information.

2. Contract Net Planning: [Davis & Smith 83] This is more distributed than centralized multi-agent planning. A single agent (*manager*) decomposes the goal into subgoals and negotiates with the agents to decide what agents (*contractors*) will take responsibility for which subgoals. A contractor who has received a contract for a subgoal can in turn be a manager and further decompose the subgoal into even smaller subgoals and distribute them to other contractors through negotiations. At each level of the decompositions and negotiations, the distribution of the subgoals is based on information about the capabilities of the contractors making the bids for the subgoals.

Since it is assumed that the allocated subgoals are independent of each other, managers who decompose them should know the relations among the subgoals and also the way to combine the result when they are completed. Therefore, contract-net framework is still most relevant for tasks with well-defined task hierarchies that can be decomposed into essentially independent subtasks.

3. Distributed Multi-agent Planning: This is the most distributed planning situation of the three discussed here. In distributed multi-agent planning, there is no centralized controller. Individual agents would like to think that other agents are working towards the common goal and interest; however, since no agent has a global view of the activities, there is no way to be sure. While an agent is working towards a goal, it is possible that some other agent is working on a goal which may be competitive or orthogonal to its goal. Therefore, in order to collaborate the efforts of distributed agents towards a converging goal, it is necessary to exchange information among the agents.

One natural solution is to allow agents to communicate with other agents so that an agent can gather enough information to create a plan. By assuming that all the agents are rational and thus building plans that contribute to the solution of the problem, eventually the effort converges to the solution.

However, sometimes direct communication among two arbitrary agents is not possible; for example, an agent doesn't know who other agents are. In this case, each agent can broadcast to a shared memory structure (called blackboard [Erman *et. al.* 80]), which other agents can be counted on to read.

2.2.9 Summary

Several planning techniques in AI have been surveyed. The non-hierarchical planner is a first generation planner that uses a planning algorithm that is too inefficient to be applied to domains other than the primitive blocks-world domain. The hierarchical planner improves the planning algorithm of non-linear planner by reducing the search space. It is most appropriate for a domain in which actions can be decomposed into smaller sub-actions among which minimum constraints exist. The skeleton planner is another improvement over the non-linear planner, but since it needs a pre-stored plan to solve a problem, it is only useful in a domain where it is possible to predict the event scenarios. The least-commitment planner plans for goals that are persistent over time: for example, save-money, hurry-up, etc. The opportunistic planner is good for planning for multiple interacting actions that impose constraints on one another. The reactive planner can handle unexpected situations, but it is not useful for solving problems that require intensive domain modeling. Thus, an increasingly reactive planner is introduced that has the capability to solve problems and to react to the changing environment. The increasingly reactive planner is also able to remember the plans it has constructed, so that it becomes increasingly robust in the domain. Finally, planners in multi-agents environments were reviewed.

This review is by no means an exhaustive review of all the planning techniques.[2] However, it touches on enough bases of the AI planning techniques to be useful as a quick reference to the terms and concepts that will be referenced later when reviewing the planning techniques used in the content planning of natural language generation systems.

---

[2]See [Georgeff 90] and [Tate *et. al.* 90] for more comprehensive review of AI planning techniques.

3. Elements of Planning in Natural Language Generation

In the previous section we reviewed the techniques of planning in AI, and in the following section we will look at the usage of AI planning techniques in natural language generation systems. The purpose of this section is to bridge the two sections by overviewing the field of natural language generation from the perspective of the AI planning and identify the elements of planning in natural language generation in AI planning terms.

In order to view the aspects of planning in natural language generation, we first enumerate the kinds of tasks in natural language generation. Then we discuss the widely varying views of different researchers in drawing the line between those of the planning tasks and of non-planning tasks. Finally, we summarize the components of text planning and introduce some of the vocabularies that will be used in discussing specific systems in the later sections.

3.1 Tasks of Natural Language Generation

3.1.1 Evolution of the Natural Language Generation Tasks

Almost 20 years ago, when research in natural language generation started, the goal of research was as primitive as to generate a random sequence of sentences that are grammatically correct. For example, [Yngve 62] describes a system that uses generative context-free grammar and a random-number generator to generate grammatical sentences.

Later on, efforts were made to achieve more general goal of building a component of machine translation (*e.g.,* [Wilks 73]) or paraphrasing system (*e.g.,* [Klein 65], [Klein & Simmons 63]) that takes some internal representation of the ‘‘meaning’’ of a text and converting it into a grammatically correct realization of the text.

Thus, the focus of these earlier natural language generation researches was on the mechanics of constructing phrases and sentences using linguistic resources, and the task of natural language generation was more or less clearly defined as converting the pre-selected set of contents into a unit of text that conveys the intended meaning in the given representation of the contents. Such a task was assumed to be relatively ‘‘easy’’ compared to the task of natural language understanding because it was believed that while understanding of a text requires the ability to infer the intended meaning of the text using domain knowledge and context of the discourse, generating a text only requires linguistic knowledge to find a mapping from the specification of the content to the matching linguistic elements.

However, the definition of the natural language generation tasks became less clear as the representation of the content and the level of details in specification of the content varied widely from one application to another. For example, when a text generation system was designed to be used as a component of a machine translation system, it was designed under the assumption that the input specification would include all the details about the syntactic structure and lexical entries of the sources language such that the task of text generation would be straight forward application of the linguistic knowledge to find the matching syntactic structure and lexical entries of the target language. In reality, however, it is not possible to find and apply such linguistic knowledge because of the presence of ambiguity which can only be resolved by the semantic and pragmatic knowledge sources. Thus, it became apparent that the task of text generation is more complex and involves more than simple application of linguistic knowledge.

At the turn of the decade, research in natural language generation became very active with the introduction of new results by [McKeown 82], [Cohen & Perrault 79], and [Appelt 85], and since then, the scope of natural language generation task has increased drastically. Most notably, the task of determining the content of the generated text was included as a legitimate part of natural language generation task. Therefore, rather than starting from the representation of the message content, a natural language generation system was designed to take as input a discourse goal like ‘‘describe <object>’’ or ‘‘explain <event>’’ and to use the underlying application system to construct the content of the message and then to transform it into a surface linguistic form. The process of content formation is called *deep generation* and that of transforming the content into text is called *surface generation*, and the part of the generation system that is responsible for the task of deep generation is called *strategic component* and that for the surface generation is called *tactical component*. Appelt also acknowledged the task of both deep generation and surface generation as part of natural language generation task. However, he disputed the validity of the two stage architecture of strategic and tactical components. Instead, he adopted the work of Cohen and Perrault on *speech acts* and implemented a system called KAMP (Knowledge And Modalities Planner) in which an integrated planning module *plans* the content of the output sentence, and realizes it in short sentences which are served as illocutionary acts to influence the hearer to perform an action.

3.1.2 Variety of Natural Language Generation Tasks

The task of today's natural language generation system is tied together with the task of the application systems that underlie them. For example, the task of TEXT system ([McKeown 82]) is producing paragraph length answers to the data base queries, the task of XPLAIN ([Swartout 73]) is generating expert system explanation by summarizing the reasoning trace in the expert system environment, and the task of GUIDON ([Clancey 83]) is generating intelligent computer aided instructions.

Natural language generation systems are adaptive to the discourse participants and the conversational settings. In a dialogue situation, a generation system is able to take turns, tailor the responses according to the expertise of the conversational participants ([Paris 87]), respond to the misconceptions ([McCoy 88]), and deal with the unexpected reactions of the hearer ([Moore & Paris 89], [Moore & Swartout 91]).

Some systems are also conscious about the the nature of the media, and the presentation environment. For example, KAMP ([Appelt 85]) generates text that refers to the body gesture of the speaker, PAULINE ([Hovy 87]) generates text with varying length and details depending on the amount of presentation time available to the speaker, and COMET ([McKeown *et. al.* 90], [Feiner & McKeown 90]), generates text that make references to the graphical display of objects that is presented on the same computer screen alongside the text.

In addition, there are systems that specializes in non-interactive applications. The task of ANA ([Kukich 83]) is generating stock market report, and the task of UNIVERSE ([Lebowitz 85]) is generating melodrama stories.

Many of the generation systems mentioned so far assumes the presence of separate surface realization component that takes care of the task of realizing the selected content into text. While the overall task of natural language generation has become more diverse being involved with more linguistic and non-linguistic criterias, the task of surface realization has remained more or less unchanged except for the fact that more of the decisions in the realization tasks — for example, ordering and structuring of the constituents, and choice of words or phrases — are influenced by the other component of the generation system.

3.2 Text Planning Task

3.2.1 Why Planning

Before attempting to answer the question ‘‘which of the natural language generation tasks enumerated in the previous section are we referring to as the text planning task?’’, we believe it is worthwhile to consider the question ‘‘why is planning used in natural language generation?’’.

An intuitive answer might be because we, the human natural language generators, plan our utterances before saying them. As a matter of fact, the instructor of the Public Speaking course offered at Columbia University encourages the students to *plan* for the speech before delivering it ([Williams 87]). He explains to his students that the goal of planning is for clarity and coherence of the speech and advices his students to follow the following guideline when preparing for the speech:

1. Decide on your subject
2. Decide the purpose of your speech
3. Analyze your audience and the occasion
4. Gather materials for you speech
5. Decide on a subject sentence
6. Plan the main points of your speech
7. Plan your supporting materials
8. Plan your speech in outline form
9. Plan your introduction
10. Plan your conclusion
11. Prepare speaker’s notes
12. Practice orally for delivery

which is remarkable similar to the steps followed by the natural language generation systems discussed in the previous section — except for the last step.

Perhaps the most influential work that introduced the AI notion of planning to natural language generation is the line of research based on the *speech act theory* ([Austin 62], [Searle 69]) that views the linguistic acts as a particular kind of physical acts. [Cohen & Perrault 79] describes a planning system that produces a plan containing a sequence of speech acts that would accomplish the goal of the planning system. Appelt’s KAMP is the first language generation system in which planning module treats speech acts uniformly with other domain acts and generates plans consisting of both physical acts and speech acts; however, most of the other generation systems share the same view and uses planning to generate plans containing speech acts.

3.2.2 What Is the Text Planning Task

The term *text planning* is one of the most overloaded terms used in the natural language generation research community. In fact, one of the main purposes of the 1988 AAAI Workshop on *Text Planning and Realization* was to clarify the meaning of that term. Even today, different researchers use the term to refer to different set of tasks. Therefore, rather than trying to give a vague general definition of the term, we list a number of distinguished views on the scope of the text planning task.

First View: the task of text planning includes only the task of *content selection* and *text structuring*, which is precisely the task of deep generator described above — *i.e.,* given a discourse goal, generate ordered propositions representing the content of the text to be realized by the surface generator. The strategic component in TEXT or TAILOR ([Paris 87]) system is one example of such text planning task, although [McKeown 82] uses the term *content planning* to refer to them. Another example of the task of text planning in GENIE ([Wolz 90]) in which additional process of revising and restructuring the text structure is allowed. Wolz also calls the text planning in GENIE the task of content planning.

Second View: [Meteer 89] views text planning as three interleaved activities of:

- *selection*: choosing the information to be included in the utterance, which is comparable to the task of content selection;
- *orchestration*: composing and organizing the selected units, which is comparable to the task of text structuring;
- *resource choice*: choosing the linguistic resources for expressing each unit, which is viewed as a task of surface generation by the first view of text planning mentioned above.

Thus, Meteer has a broader definition of text planning.

Third View: This is the view shared by the members of the 1988 AAAI Workshop on *Text Planning and Realization* ([Hovy *et. al.* 88]), and one that most closely resembles the task of text planning in Hovy's PAULINE ([Hovy 87]). In this view, the task of text planning includes the following:

- determining which elements to say;
- structuring the input elements into coherent order;
- building noun phrases (including pronominalization) from a set of attributes;
- determining the appropriate level of detail to use
- controlling the slant and style of the text.

Thus, this view classifies even more of the surface generation tasks under text planning than does the second view; in this view word choice is included as part of the text planning task whereas in the second view only the choice of abstract linguistic resource is included.

Forth View: Finally, the task of text planning in DIOGENES ([Nirenburg *et. al.* 89]) includes almost all of the surface generation tasks such that given the output of the text planner, practically all that the realization component has to do is to replace the syntactic representation by a surface string. In summary, the task of text planner consists of:

- *text structuring*: determining the order of propositions and the boundaries of sentences;
- *lexical selection*: selecting lexical units;
- *syntactic selection*: selecting syntactic structure;
- *coreference treatment*: introducing anaphora, deixis, and ellipsis phenomena when appropriate;
- *constituent ordering*: establishing the order of syntactic constituents in a sentence.

Interestingly, the task of *content delimitation* is not categorized as a text planning task. Nirenburg believes that it is the task of the underlying application system to select propositions that are relevant to the propositional goal of the system; given that DIOGENES is language generation subsystem of a machine translation system, it is understandable.

3.3 Components of Text Planning

What are the components of the text planning? We believe that identifying the components of text planning helps us to better understand the nature of the text planning tasks in natural language generation and also to facilitate the comparison with the task of planning in AI.

Compiling the list of the components of a plan for text planning, however, is difficult because, as we described in the previous section, it is not even clear what the text planning task is. But then again, the picture of text planning task will not be clearer unless we identify the components of the plan. Therefore, we have compiled a list of the components of a plan for text planning in figure 3-1 as we did with the components of planning in AI in the domain of block-world (see figure 2-1).

In the following section we review the text planning task of each of the selected language generation systems by identifying the assumptions they make about 1. the state of the world: how is it defined and represented; 2. the goal of the planning task; since ‘‘planning’’ implies solving a problem, the text planning must have a problem that it is trying to solve; 3. the operators available to the speaker to achieve to goal; and 4. the plans and planning techniques used by the planner in order to achieve the goal.

State of the world

- Objects (same as those in AI planning)
- *Intelligent* Agents (discourse participants)
- Predicates describing the state of objects
- Predicates describing the mental-state of the agents
    - Intention
    - Belief: e.g. H believes that S wants z. H believes that S wants to inform z.
    - Mutual belief[3]
- State of the discourse; attentional state, focus space

Operators

- Domain actions
- Communicative actions: A speaker says something with the intention that the hearer will recognize the intentions underlying the utterance.
    - Speech act[4]
        - Inform
        - Request
        - Promise
    - Indirect speech act; conversational postulates

Goals

- Domain goals
    - Change the state of objects.
- Communicative goals (or discourse goals)
    - Change the state of beliefs and intentions of agents.
    - Initiate or participate on a conversation.
    - Change the state of discourse; change the current focus of discussion (attentional state)
- Rhetorical goals (See figure 4-10 for the list by [Hovy 88b].)
- Social goals

Figure 3-1: Components of a Plan in NL Generation

4. Text Planning in NL Generation

In this section we review a number of text planning systems and unimplemented proposals for text planning that employ AI planning techniques discussed in section 2.2. Up to this point in the paper, we outlined planning research in the field of AI, and analized the components of planning in the text planning task in general. Now, we look at the specific use of planning techniques in actual natural language generation systems.

A total of nine text planning systems plus two proposals have been selected for review out of an order of magnitude more of other systems and a much larger number of proposals. The nine systems included here are selected because they are the landmark natural language generation systems that also use one form of planning or another for the generation of the text. Although the authors of these systems all claim that they use planning for the generation of the text, the specific planning task, the techniques used for the planning, and even the definition of planning varies widely from one system to another.

Common to all these systems and proposals is the fact that they all employ one or more of the planning techniques *similar* to the ones highlighted in section 2.2. Therefore, we categorize them under the name of the planning techniques that resemble those most prominently used in that system. However, it should be noted that even though identical names are used to refer to them (for the convenience), there are often subtle discrepancies between the named AI planning techniques and those used in systems that are categorized under the same name.

For each one of the implemented text planning systems, we try to find a precise definition of the system's planning task by identifying the following elements from planning: state of the world, operators, goals, and plans. Then, we look at the planning techniques used in building a plan.

4.1 Skeleton Planning

A variety of special-purpose generators use script-like topic collection and ordering strategies; for example, [Kukich 83] describes the generation of stock market reports, [Rosner 87] translates Japanese reports and descriptions into German, and [Woolf 84] describe a program that tutors students, by planning which schema to select and when to change it, based on its representation of their knowledge and misconceptions. In this section, we review two works that generate responses to queries to database objects; McKeown's TEXT system and Paris's TAILOR system. Both of them use pre-stored plans called schema in generating the content and the structure of the responses, which is the most widely used technique in natural language generation today.

4.1.1 McKeown's TEXT

McKeown's TEXT is a paragraph-length response generation system for questions about database structure [McKeown 82]. TEXT generates text in response to a question based on the two stage model of language production: the *strategic component* which determines the content and structure of discourse and the *tactical component* that uses grammar to translate the message into English. We focus on the strategic component of TEXT for the purpose of reviewing the planning task of TEXT.

Given a discourse goal, the strategic component of TEXT uses text *schemas* to select information from a relevant knowledge pool and organizes the information into a rhetorical structure appropriate for the discourse goal. In planning terms, schemas correspond to scripts and the *rhetorical predicates* that constitute a schema correspond to plan operators such that by executing the rhetorical predicates in the order prescribed within a selected schema, the text planner collects information and organizes it into an appropriate rhetorical structure.

We now look at the components of planning and the skeleton planning process used in TEXT in more detail.

State of the World

A state of the world in TEXT can be defined by a set of *propositions* that are derived from the relevant knowledge pool, where a proposition consists of the description of an object in the underlying database. A more abstract way of defining a state in TEXT would be to define it to be the state in ATN (Argumentative Transitive Network) that is used in TEXT to represent schemas, analogous to pre-stored plans in the skeleton planning terminology. (See Figure 4-2 for an example of an ATN.) ATN is a non-deterministic finite automata consisting of states and arcs where one state is reached from another by tracing through the arcs that connect them. Tracing an arc in an ATN corresponds to ‘‘executing’’ a plan operator, and as will be discussed later, execution of a plan operator produces a new set of propositions. Therefore, each state in an ATN implies a different set of propositions, and thus different state of the world. In this sense, a state in ATN can be viewed as the state of the world in TEXT.

Since the goal of the planner is a discourse goal, the definition of the state should also include propositions about the intentional model of the agents (e.g. beliefs and intentions), but TEXT makes no explicit represent of the intentional state of the agents. Consequently, TEXT lacks the ability to reason about the effect of the generated text or to handle dialogue with follow-up questions. (See Section 4.5.1.)

Operators

The plan operators in TEXT are the *rhetorical predicates* because execution of a rhetorical predicate changes one state of the world to another by adding more propositions to the set of propositions that are collected up until the previous state. There is one rhetorical predicate associated with an arc in the ATN, and for each one of the rhetorical predicates, there is a predicate function associated with it that accesses the relevant knowledge pool and retrieves a set of propositions that match the predicate. The set of rhetorical predicates McKeown uses is listed in Figure 4-1 explained by a brief description or an example sentence that can be realized from the resulting proposition.

Goals

The goal in the planning task of TEXT is to answer the user's current question. There are three types of questions handled by the system:

1. define <concept>
2. describe information available about <concept>
3. compare <concept1> <concept2>

In order to really ‘‘answer’’ a user's question, it is necessary to have a user model (or some equivalent construct) that can be used to verify whether the set of propositions planned for generation are sufficient, comprehensive, or relevant for the user. TEXT, however, does not use a user model; therefore, the text planner assumes that a goal is reached when a selected schema is fully instantiated with the propositions.

Plans

Attributive	"Mary has a pink coat."
Equivalent	State the equivalence of two mentioned items.
Specification	Specification of general fact.
Explanation	Reasoning behind an inference drawn.
Evidence	Evidence for a given fact.
Analogy	Make analogy.
Representative	Item representative of a set.
Constituency	Presentation of sub-parts or sub-classes.
Covariance	Antecedent, consequent statement.
Alternatives	"We can visit the Empire State Building or call it a day."
Cause-effect	State the cause-effect relation.
Adversative	"It was a case of sink or swim."
Inference	"So people form a low self-image of themselves."
Topic	Introduces the topic.
Amplication	Expands on the topic.
General Illustration	Universal statement supporting claim.
Particular Illustration	Specific statements supporting claim.
Comparison	Noting features in common between two entities.
Contrasting	Noting feature differences.
Conclusion	Drawing conclusion.
Identification	Identifying an entity as belonging to a specific class.
Renaming	Provide alternative names for an entity.
Positing	Introduce an entity into the text.

Figure 4-1: Rhetorical Predicates in TEXT [McKeown 82]

A plan in TEXT is a schema. A *schema* is a script-like pattern of rhetorical predicates used to direct the topic collection process and to order the selected topics into a coherent paragraph. After analyzing naturally occurring text, McKeown identifies the following four schemas listed with the request(s) that each of them are intended to be used to generate a response for:

1. Identification Schema
    - requests for definitions
2. Attributive Schema
    - requests about available information
3. Constituency Schema
    - requests for definitions
    - requests for available information
4. Compare and Contrast Schema
    - requests about differences between objects

For example, the constituency schema defined in Figure 4-2[5] can be used to characterize the following text from an encyclopedia that describes the steam and electric torpedoes. (Curly brackets are used to indicate the name of the rhetorical predicate used to generate the sentence(s) following them.):

> {**Constituency**} Modern torpedoes are of 2 general types. {**Depth-identification (attributive)**} Steam-propelled models have speeds of 27 to 45 knots and ranges of 4000 to 25,000 yds. (4,367 - 27,350 meters). {**Comparison**} The electric powered models are similar {**Depth-identification (attributive)**} but do not leave the telltale wake created by the exhaust of a steam torpedo. [McKeown 82]

Although a schema looks and works like a stored-plan of a typical skeleton planner, schemas are more general and flexible than stored-plans. A typical stored-plan is represented by a linear sequence of actions with the only allowed variation being that an action in the sequence might be marked as optional or it could be a list of options from which one is selected. In contrast, a schema is represented in full augmented transition network. Therefore, even though it appears that there is only a limited number of schemas used in TEXT, many variety of plans can be generated on the fly from a schema based on the availability of the knowledge in the relevant knowledge pool.

Building a Plan

The plan construction process in TEXT consists of two steps: selecting a schema and filling the selected schema. The process of selecting a schema in TEXT corresponds to the process of searching for a pre-stored plan in the skeleton planning. Likewise, the process of filling the selected schema corresponds to the instantiation of the selected plan in the skeleton planning. Therefore, the overall process of building a plan is basically the same as that of skeleton planning.

Basically, the schema selection process can be summarized as following: depending on the question-type, find a set of schemas that may be used, and on the basis of the information availability, select a single schema. For example, if a request is made for the definition of or for information about an object, then the constituency schema is selected except when there is not enough information in the

---

[5] ‘‘{}’’ indicates optionality, ‘‘/’’ indicates alternatives, ‘‘+’’ indicates that the item may appear 1-n times, ‘‘*’’ indicates that the item may appear 0-n times.

```
Constituency
Cause-effect*/Attributive*/
    { Depth-identification/Depth-attributive
        {Particular-illustration/evidence}
        {Comparison; analogy}  }+
{ Amplification/Explanation/Attributive/Analogy }
```

<a>

<b>

Figure 4-2: The Constituency Schema (<a>)
and Its Representation in ATN (<b>) [McKeown 82]

knowledge base. Then the identification schema is used for definition type questions and the attributive schema is used for information type questions.

On the other hand, the process of filling the selected schema involves more complex procedures. As mentioned above, a schema is a pattern of rhetorical predicates (see Figure 4-1) that outlines the structure of the output paragraph. To *fill a schema* means to traverse the schema and to *instantiate* the rhetorical predicates in the schema with propositions in the relevant knowledge pool by executing the predicate functions associated with the rhetorical predicates. The complexity in the schema filling process is in part introduced by the following three cases:

1. More than one proposition can be instantiated for a single rhetorical predicate.
2. While traversing the schema, a point is encountered where a choice among alternative rhetorical predicates is needed.
3. While traversing the schema, an optional rhetorical predicate is encountered.

In order to deal with these cases, McKeown introduces the use of *focus constraints* as criteria for selecting the appropriate propositions. The role of focusing in natural language understanding has been studied before. In particular, Grosz ([Grosz 77]) identified the role of *global focus* and *immediate focus* in the interpretation of referring expressions in dialogue. In TEXT, McKeown uses global focus to separate out the relevant knowledge pool from the knowledge base and immediate focus to single out a proposition from the set of possible propositions. More specifically, of the three cases listed above, the first case is resolved by selecting one proposition from the set of instantiated proposition that is most closely related to the current immediate focus. The second case is resolved by producing all the propositions for all alternative rhetorical predicates and then selecting the proposition (and thus the associated rhetorical predicate) that is most closely related to the immediate focus. Finally, the third case is resolved by instantiating the rhetorical predicate only if the instantiatable proposition is in the current immediate focus.

Given that the goal of the planning task in TEXT is to answer user’s question, immediate focus may be viewed as a heuristic to guide the search through the relevant knowledge pool to find the proposition that will become part of the plan that achieve the goal, and in fact it is a good heuristic because a user is more likely to be satisfied by an answer that is focused and thus more coherent.

Conclusion

There are only a few schemas to choose from, and each of them is very general. Therefore, the process of instantiating a schema accounts for most of the planning task while very little effort is required of the schema selection process; the opposite is true in AI skeleton planning where the most difficult task is to select the relevant pre-stored plan and instantiating that plan is relatively trivial task.

One of the reasons for having only a few and general schemas is because it is difficult and time consuming to develop domain specific schemas for every new domain. Therefore, McKeown proposes the concept of *schema library*, which is analogous to the plan library in skeleton planning, in which both general and specific schemas are defined in a schema abstraction hierarchy ([McKeown *et. al.* 90]). Naturally, increased number of schemas will require more sophisticated schema selection criterias, and thus it will result in a more balanced work load between the schema selection and schema instantiation processes; also, it will make the planning technique resemble the AI skeleton planning more closely.

Being a kind of skeleton planner, TEXT inherits the advantages and disadvantages of the AI skeleton planner. Due to the readily available ‘‘pre-stored’’ schemas, the need to construct a plan is saved and thus the planning process is expedited. However, since schemas are pre-written (although partially generated on the fly) and provide almost no information about the explanatory structure, TEXT is not capable of reasoning about (or explaining) the text it has generated.

4.1.2 Paris’s TAILOR

Paris’s TAILOR ([Paris 87]) is an improvement over the TEXT system with the added capability to tailor responses to users of varying expertises. Paris makes the observation that user’s domain knowledge affects not only the amount of detail to be included in the responses but also the *kind* of information to be presented; *i.e.*, when an expert asks for the description of an object, he is expecting description around object parts and their properties whereas when an a novice asks the same question, he is expecting description of the object in terms of functional information of parts. Therefore, Paris introduces *process trace*, a strategy to traverse the knowledge base, to be used in conjunction with McKeown’s constituency schema in generating responses tailored to the users. (See Figure 4-3 for the two different types of responses generated by TAILOR.)

Description of the microphone for an expert user
The microphone is a device that changes soundwaves into current. The microphone has an aluminium disc-shaped diaphragm, and a doubly-resonant system to broaden the response. The diaphragm is clamped at its edges. The system has a button, and a cavity.
Description of the microphone for a naive user
A microphone is a device that changes soundwaves into a current. Because a person speaks into the microphone the soundwaves hit the diaphragm of the microphone. This causes the diaphragm to vibrate.

Figure 4-3: Two Types of Responses Generated by TAILOR [Paris 87]

The planning technique in TAILOR most closely resembles skeleton planning. The components of planning and the planning process resembles those of TEXT system. Therefore, we review planning in TAILOR by comparing it with that in TEXT.

State of the World

A state in TAILOR, like that in TEXT can be defined by a set of propositions that are derived from the knowledge base. In TAILOR, however, a state includes propositions about the domain knowledge of the user.

The portion of the knowledge base that contains a user's domain knowledge is called the *user model*. A user model is constructed progressively during the course of discourse generation such that once a description of an object is generated to a user, the user is assumed to have learned the object. (The current implementation of this perfect learner model is relaxed.) At the beginning, a user is assumed to be a *naive* user in the domain. As the user learns more and more about the basic concepts and parts description of objects in the domain, the user increasingly becomes an *expert* user.

Operators

In addition to the rhetorical predicates defined and used in TEXT, Paris adds *directives* to be used as the operators that collect information from the knowledge base by tracing a process in it. The definitions and use of those directives are shown below under *Plans*.

Goals

The goal in the planning task of TAILOR system is similar to that in TEXT except that the one kind of question handled by TAILOR is:

- describe information available about <concept>.

Paris mentions the importance of user's domain goals in deciding what to include in an answer, but she does not address the issue any further in her thesis.

Plans

A plan in TAILOR is called a *strategy* and it may be in one of the three forms: constituency schema, process trace, or some mixture of constituency schema and the process trace. While the constituency schema alone was sufficient in TEXT as a plan to generate a response for the description of an object, Paris found out that for the kind of text appearing in a junior encyclopedia, where the intended audience is a novice reader, descriptive strategies like constituency schema cannot sufficiently guide the generation process. Instead, she found that using directives that trace the knowledge base process results in a better plan because such a plan generates a text like those that appear in a junior encyclopedia.

The process trace is very similar to a schema except that instead of rhetorical predicates, each arc in the process trace is one of the following action operators (called directives):

- *Identification*: This is identical to the identification predicate in the constituency schema.
- *Long-side-chain1*: If there is an enabling (or causal) condition for an event on the *main path*, then diverge from that main path to a *side link* to describe the enabling condition before returning to the main path.
- *Next-main-link*: The main path consists of several event-links. This directive indicates to take the next link on the main path.
- *Long-side-chain2*: Same as *long-side-chain1* except that the diverging side link is unrelated to the event preceding the side link.
- *Short-side-link*: Include additional explanation for the event being described on the main path.
- *Attributive*: This arc is similar to the attributive predicate of the constituency schema except that in retrieving attributive information from the knowledge base, retrieve only the properties about the object, not structural relations.
- *Substeps*: If the event just mentioned is decomposable, then mention the substeps.

Based on her study of various texts, her intuition about what makes a text coherent, and experimentation with TAILOR, Paris designed the process trace using the directives listed above. Figure 4-4 shows partial implementation of the process trace with subnets for *substeps* and *long-side-chain* omitted.

Since both process trace and constituency schema are implemented using the same mechanism, the ATN, mixing the two strategies is facilitated. Choosing the strategy and mixing the two strategies is discussed next.

Building a Plan

As in TEXT where the plan construction process is a two stage process of selecting a schema and

Figure 4-4: ATN corresponding to the Process Trace [Paris 87]

filling the selected schema, the plan construction process in TAILOR consists of two stages: selecting an appropriate strategy and filling in the strategy with propositions in the knowledge base. The major difference between the plan construction process of the two systems is in the first stage. Since the second stage is not much different and it is already discussed in the previous section, we concentrate on the strategy selection process here for the TAILOR system.

At first it seems trivial to select a strategy since there are only two strategies: constituency schema and process trace. However, because the system can switch from one strategy to another during the construction of a description, it is necessary to plan for choosing an initial strategy and then combining the strategies based on the user model.

Once the initial strategy is selected, based on the knowledge base and the user model, the system should then be prepared to switch between the two strategies in order to generate a description aimed at users with intermediate levels of expertise. Two things that need to be specified are: 1) *when* is it possible to switch from one strategy to the other 2) *the conditions* under which it is possible to switch. Basically switching between the strategies takes place anytime an object or its parts are introduced, and the conditions for which a switch can take place is whether or not the user knows the object.

Conclusion

Under the same skeleton planning formalism, Paris increases the flexibility of using the pre-stored plans in TEXT by modifying the constituency schema and the process trace to have switch points in them such that a variety of combination of the two strategies can be used. Also, choice of a plan is guided not by the knowledge base alone but by the combination of the knowledge base and the information in the user model.

As a result the TAILOR system can use a *tailored* plan in generating a response that can best satisfy a user whether he is an expert user, a naive user, or anything in between.

4.2 Hierarchical Planning

Hierarchical Planning is the most "popular" AI planning technique used in natural language generation. Researchers who use hierarchical planning in their generation system believe most strongly that they are indeed using an established AI planning technique, and they often cite NOAH [Sacerdoti 77] as their reference. Almost unmodified AI hierarchical planning is used in many text planning systems because the task of collecting topics for generation is very similar to that of searching for operators in AI problem solving. In fact, on many occasions, speech acts are treated on par with domain actions, and the task of topic collection is handled by the same hierarchical planner that solves the domain problem.

We have selected two systems for review in this section. The first is DISCOURSE PLANNER, the text planner module in Dale's EPICURE system. DISCOURSE PLANNER uses hierarchical planning to generate a cooking recipe in much the same way that a AI hierarchical planner would generate an action decomposition hierarchy for a problem solving. The only difference is that DISCOURSE PLANNER goes through additional steps to traverse the recipe to convert the domain actions into *request* speech acts. The second system is Appelt's KAMP which constructs an action decomposition hierarchy consisting of both domain actions *and* speech acts in a single planning process.

Some of the other systems reviewed later use hierarchical planning, also: e.g. PAULINE, STRUCTURER, and EXPLANATION GENERATOR. But they use other other planning techniques in conjunction, and thus for those systems we do not discuss the use of hierarchical planning in as much detail as we do here.

4.2.1 DISCOURSE PLANNER in Dale's EPICURE

EPICURE is a system that generates connected discourse from specifications of plans of action [Dale 88]. DISCOURSE PLANNER is a module within EPICURE that corresponds to the content planner of the system (See Figure 4-5). It determines ‘‘what to say’’ by specifying the particular actions to be described.

Figure 4-5: The Overall Structure of EPICURE [Dale 88]

The output of DISCOURSE PLANNER is a tree-structured *discourse specification* that corresponds to the hierarchical decomposition of the plan it describes. The process of building a plan resembles that in NOAH [Sacerdoti 77] because it starts out with a top level goal, and recursively expands it to more detailed subgoals using the PLAN LIBRARY, where the problem solving information is stored, and the HEARER MODEL, where the capability of the hearer is stored.

A major difference between planners like NOAH and the DISCOURSE PLANNER is that the actions in the plan are *request* speech acts. The EPICURE system does not have the capability of carrying out any physical actions except for making requests to someone else. Therefore, in order to achieve a goal, EPICURE generates a plan that specifies the requesting actions to be carried out.

In [Dale 88], EPICURE is applied to the domain of cooking, and the particular plans considered are cookery recipes. We look more closely into the components of the plan and the process of building a plan with examples in this domain.

State of the World

A state of the world is defined by objects with their constant properties and temporal properties. *Constant properties* of an object are those properties of an object that do not change throughout the life of the object. For example, an object *x* that represents *1-2 tablespoons of honey* is said to have the following constant properties:

substance(*x*, honey)

quantity(*x*, range(<1, tablespoon>, <2, tablespoon>))

meaning that the object *x* consists of honey and its quantity is somewhere between 1 to 2 tablespoons.

Temporal properties of an object are those properties of an object that are only true at certain time (or state). For example, if an object *x* is hot at state *s*, then the following proposition *P* is true:

$P=\mathrm{holds}(s, \mathrm{hot}(x))$

Therefore, a state *s* can be characterized by the following formula:

$s = \mathrm{holds}(\mathrm{s}, P_1) \vee \mathrm{holds}(\mathrm{s}, P_2) \vee ... \vee \mathrm{holds}(\mathrm{s}, P_n)$

where P_{i} represents a proposition about an object.

Operators

Operators are, as we discussed in section 2, actions that change the state of the world. Since the only act that EPICURE is capable of carrying out is the *request* speech act, the task of DISCOURSE PLANNER that plans text for EPICURE should really be deciding "what to request", so that by making proper sequence of requests, the hearer who is capable of carrying out the requested acts can follow the requests and change the state of the world. For example, DISCOURSE PLANNER may decide that it is necessary to execute the action *chop(potato)*. Then, the actual operator that changes the state of the world is *Request(chop(potato))*.

Operators that change the state of the world directly are called *eventualities* in EPICURE. The term eventuality is from the taxonomy of eventuality introduced by Bach ([Bach 86]) and it refers to any collection of objects and the state or event in which the objects participate. In the recipe domain, Dale simplifies the implementation by limiting the eventuality to a particular kind of eventuality called *events*.

An eventuality (really an event) is represented by the subsumption relations it has on the taxonomic hierarchy, the period during which the eventuality takes place, and a finite set of possible participants (agents or objects) identified by the role it plays in the eventuality.

The eventuality operators can be categorized into three different kinds based on the effects they have on the objects to which they are applied. First, there are operators that only change the state of the object like those operators in the block world (See Section 2-1). For example, the PEEL operator takes an ingredient x and changes the state of x from *peeled*(x, +) to *peeled*(x, -). Then there are operators that change the structure of an object. For example, the *grate* operator takes an ingredient x and changes its structure from a bounded individual to one which is perceived as a mass. Finally, there are operators that may create new objects either by combining two or more existing objects or by breaking an existing object into parts. Examples of such operators in the cooking domain are *add(x, y)* and *half(x)* operators.

Goals

A goal state of EPICURE in the domain of cooking tasks is the preparation of food in some manner. For example, BUTTER-BEAN-SOUP, is the goal state that can be reached by executing a plan generated by the planner. It should be emphasised that the planner in EPICURE is DISCOURSE PLANNER the task of which is not to build a plan of actions that results in the goal state directly, but rather to build a sequence of *directives* that guides an agent to achieve the goal state by executing the directives. In that regard, the goal state BUTTER-BEAN-SOUP should really be described as something like *Direct(agent, make(BUTTER-BEAN-SOUP))*. However, a directive can be viewed as an action (alias a speech act) itself that contributes indirectly towards achieving the goal state.

Plans

A plan in EPICURE is a prescribed sequence of eventualities that must take place in order for an initial state to reach the goal state. More precisely, it is a hierarchical decomposition of eventualities, called a discourse specification, that is to be instructed to the agent to execute. All the leaf nodes of the plan consists of eventualities that the agent is physically capable of executing. For example, the plan that results in the goal state BUTTER-BEAN-SOUP is shown in Figure 4-6.[6]

Figure 4-6: Butter Bean Soup Plan [Dale 88]

[6]Note that each node in the plan represents an eventuality rather than the discourse act that directs the agent to execute the eventuality. This is for the simplification only. In the actual plan generated by the DISCOURSE PLANNER, the eventualities at the leaf nodes of the plan is replaced by the directive discourse acts that represents the REQUEST speech act that requests the agent to execute those eventualities.

In this plan, the eventuality e_3 is expanded to three other eventualities, e_7,e_8,e_9, because the agent who has requested for the plan is not capable of *preparing the beans* according to the USER MODEL of the system. On the other hand, the three eventualities are not further expanded because the system believes that the agent is capable of executing them.

Building a Plan

The DISCOURSE PLANNER builds a plan using hierarchical planning technique; it selects one of the higher level plans that guarantees a solution, and then expands one or more of the plans as it is deemed necessary until all the plans are expanded as needed. The need for expanding a plan is decided by the *planning axioms* that determine whether or not the agent carrying out the action is capable of the action.

If the agent, say *A*, is physically capable of a task *T*, then the action *Request(A, T)* is the terminal node in the action decomposition hierarchy of the plan that DISCOURSE PLANNER generates. Otherwise, decompose the task *T* into subtasks T_i, and recursively apply the testing until the task is decomposed into only those subtasks that the agent is capable of achieving. When the entire plan is composed of such tasks, the discourse planning algorithm terminates.

From the implementation of the DISCOURSE PLANNER in the domain of the cookery recipes, one might think that the technique used in generating plans is closer to skeleton planning than to hierarchical planning because rather than working out a plan to perform the transition from the initial state to the goal state, DISCOURSE PLANNER simply accesses the PLAN LIBRARY to pick out a plan. The appropriate decomposition of the plan is already specified in the library. However, Dale explains that the particular implementation of EPICURE in his thesis lacks the planner that can generate the decomposition automatically. His design of the fully implemented system includes a planner that is capable of generating the decomposition of a task when one is not already available from the PLAN LIBRARY.

Conclusion

In this section we looked at an example of the most direct use of hierarchical planning for the task of text planning. Aside from making decisions for what actions to request, DISCOURSE PLANNER does not have much planning to do in terms of constructing and organizing the content of the text. This is because EPICURE does not use a representation of rhetorical relations, and thus, it has no rhetorical goals to achieve. Furthermore, any request for an action generated by EPICURE is assumed to be understood and executed perfectly by the hearer without a dispute, as long as s/he has knowledge to execute it. Therefore, it is not the concern of the DISCOURSE PLANNER to change the intentions of (or to persuade) the hearer. With these simplifications, the task of content selection becomes much like a straight forward problem solving task in AI, and thus it is enough to use the simplest form of hierarchical planning.

4.2.2 KAMP

Appelt's KAMP (Knowledge And Modalities Planner) [Appelt 85] is a planning system that is capable of planning physical and illocutionary acts to influence another agent's knowledge and intentions to perform an action. KAMP is properly named a ‘‘planner’’ because KAMP generates a sentence by using a uniform planning mechanism to perform all tasks from the content selection to the selection of specific words and syntactic structures that realize the selected content as an English sentence.

Unlike many theories and systems that use the *conduit metaphor* in which strategic and tactical components are considered as two independent modules, KAMP treats the two components as closely interacting components in the overall planning process. As a result, KAMP simultaneously considers serveral goals and different kinds of knowledge ranging from domain knowledge to the linguistic knowledge as shown in Figure 4-7.

State of the World

The world in KAMP, like that in a block-world planning domain, is divided into discrete states, each of which is characterized by a set of logical formulas that describe what is true in that state; for example, $P_1 \wedge \ldots \wedge P_n$, where each P_i represents a logical formula that holds true in that state. Unlike a block-world, however, the state description in KAMP includes a wide variety of *intensional concepts* that includes knowledge per se, mutual knowledge, belief, and intention of agents in the world. More specifically, a state description in KAMP incorporates information about **Know**, **Believe**, **Mutually-Know**, **Knowref** (knowing the denotation of a description), **Intend** (intending to perform a particular action). For example, as an effect of executing the action, *Do(Speaker*,

Figure 4-7: Overview of KAMP [Appelt 85]

Inform(Hearer, P)) {*i.e.*, the act of speaker's informing the hearer about P}, the following new information will be added as a part of the description of the new state of the world:

Know(*Hearer*, **Intend**(*Speaker*, **Believe**(*Hearer*, *P*))).

{*I.e.*, the hearer knows that the speaker intends the hearer to believe P.}

Within the possible-worlds semantics approach that is adopted in KAMP to represent knowledge, introduction of these mental state descriptors presents problems that have not been faced by previously implemented planning systems. In the possible-worlds formalism, the mental state of an agent is not an explicit object that can be manipulated as a goal or subgoal. Instead, mental states are characterized implicitly by the worlds that are consistent with an agent's beliefs or preferences. Therefore, achieving a state in which *A* knows *P* requires making *P* true in every world that is consistent with what *A* knows. This is a problem because the possible worlds that involve *A*'s knowledge are potentially infinite.

There is another problem. Unlike other physical objects in the world, agents are capable of reasoning about their knowledge. Therefore, when planning to affect an agent's knowledge, the planner needs to reason about what the agent can deduce when new information is added to his knowledge.

In order to deal with these problems, KAMP uses *action summaries* that summarize the common stereotypical effects of actions on knowledge and the physical world. Using action summaries as a heuristic guide, KAMP reduces the large search space and makes a reasonably correct plan that can be verified in an actual situation.

Operators

Operators in KAMP can be largely classified as physical actions that affect the state of the physical objects in the world and illocutionary acts that affect the knowledge-state of the agents in the world.

Example of *illocutionary acts* are *inform* and *request* whereby the knowledge and intentions of the speaker are transferred to the hearer. In order to achieve these high level illocutionary acts, *surface speech acts* like *command*, *ask*, and *assert* are used that are abstract representations of physical utterances. A surface speech act performed by a speaker is meaningless unless the hearer recognizes the propositional content of the utterance. Therefore, there is another layer of actions between the surface speech acts and the physical utterances called *concept activation actions* that ensure that the concepts or objects to be communicated are at the current common focus of attention. A concept activation action does not have to be realized by pure linguistic actions; it may include gestures and other physical actions with communicative intent. For example, if a speaker generates a sentence ‘‘Use the wheelpuller to remove the flywheel.’’ while pointing to a wheelpuller on the table with the intent to communicate to the hearer that a wheelpuller is the object on the table currently being pointed to, then the pointing gesture is a concept activation action.

The actual utterances themselves are called *utterance acts*. They realize the descriptors chosen for the concept activation actions with specific words and syntactic structures. Aside from the fact that it helps to achieve the goal of the concept activation actions, utterance acts are not much like physical ‘‘actions’’; however, by being treated as actions, they fit into the uniform planning formalism of KAMP where higher-level goals and actions are influenced by the expansion of low-level linguistic actions in the hierarchical design of its utterance-planning mechanism.

Goals

A goal state in KAMP is a state of the world that includes the intended set of physical and mental states. For example, in the police crime laboratory example given by Appelt, an officer brings for analysis substances found at the scene of a crime. In the example, the goal of the system is to identify the unknown substance. The goal state, that will be achieved by executing the plan of actions constructed by KAMP, contains modified set of logical formulas that describe the changed mental state of the agent performing the actions of analyzing the mystery substance; that is, the agent knows what the substance is.

Plans

A plan is represented in KAMP by a data structure called a *procedural network* borrowed from the data structure in Sacerdoti's NOAH planner [Sacerdoti 77]. Plans are represented as partially ordered sequences of actions; linear ordering of actions are imposed only when sufficient information has been gathered which gives the advantage that one can avoid committing oneself to an incorrect linear ordering that would subsequently have to be discarded.

A procedural network can be thought of as a two-dimensional data structure where the horizontal dimension is a temporal one reflecting the partial ordering of the actions, and the vertical dimension is one of abstraction, along which goals and abstract actions (illocutionary acts, surface speech acts, concept activation actions, and utterance acts) are refined into sequences of low-level executable actions.

An example of the procedural network that represents the utterance plan for the utterance ''Remove the pump with the wrench in the tool box'' is shown in Figure 4-8.

Building a Plan

The process of building a plan is very similar to that of NOAH planner, the basic control cycle of which is described below:

1. Goals are expanded at the current level of abstraction.
2. Critics are applied to detect interactions and modify the plan.
3. If the entire plan has been expanded to the lowest-level actions, the process is terminated; otherwise it is repeated and actions on the next lower level of abstraction are considered.

When KAMP is given an initial goal, it starts by creating a procedural network consisting of a single plan step that contains the goal. Then it expands the goal(s) at the current level of abstraction in a top-down, context-free manner, using the action summaries. When all the goals are expanded, the critique cycle begins. As in NOAH, critics in KAMP detect interactions among the plans in order to fix context sensitive interactions and to reduce the complexity of the plans.

Figure 4-8: A Procedural Network for an Utterance Plan [Appelt 85]

After the critique cycle has finished and all goals and high-level actions have been fully expanded to the next level, the planner uses deductive theorem proving to verify that the proposed sequence of actions actually achieves the goal. If the proof fails due to the inadequacy of the action summaries (thus, there is no heuristic guide to reduce the search space), then the planner has to resort to brute force search to reduce the large search space. In that case, finding an alternative plan becomes very inefficient, if not infeasible.

The planner iterates the process of expansion and criticism until the entire plan has been expanded to the current level in the abstraction hierarchy. Next, the entire process is repreated for actions at the next lower level of abstraction. If the planner is already at the lowest level and all critics have been applied to the resulting plan, then a complete, executable plan has been found.

Conclusion

KAMP is another example of a natural language generation system that makes direct use of an AI planning technique known as hierarchical planning. KAMP demonstrates the use of planning in natural language generation where the planning takes place at all stages of the generation process from the selection of the content to the determination of syntactic structure and lexical items of the text. By allowing a backtracking mechanism to retract decisions across all levels of the planning, KAMP manages to avoid the conduit model of generation system. However, this planning approach of KAMP results in inefficiency of the generation process. Even for the generation of a single sentence, a great deal of planning and backtracking is necessitated. While KAMP is capable of successfully mixing various kinds of knowledge and achieving multiple goals simultaneously, it does not include the knowledge about how to combine individual speech acts into larger piece of coherent text (i.e. rhetorical knowledge). Therefore it is limited to generating 2-3 sentences at a time.

4.3 Least-commitment Planning (Restrictive Planning)

There aren't many natural language generation systems that use least-commitment (also known as restrictive or limited-commitment) planning techniques, but two of the more well known recent systems have adopted this planning technique for different purposes. First, Hovy's PAULINE is an ambitious effort to build a system that generates text in an environment as complex as the real world. In his attempt to devise a planning algorithm that produces plans that satisfy complex set of pragmatic goals, Hovy is faced with the limitations of the conventional hierarchical or skeleton planning, and finds the least-commitment planning technique most adequate for certain tasks. So, he uses interleaving of least-commitment planning and hierarchical planning for his planning task.

The other system that is reviewed in this section is Meteer's SPOKESMAN natural language generation system. The task of text planning in SPOKESMAN is not as complex as that in PAULINE; nevertheless, the task becomes a challenging one when Meteer tries to find an efficient text planning algorithm that progressively refines given abstraction of the utterances into linguistic terms in one continuous process without making a mistake of including content that cannot be expressed in natural language. Again, the most appropriate choice for her text planning needs turns out to be the least-commitment planning technique.

There is another system currently being built by Suthers which uses least-commitment planning techniques. That system uses a combination of hierarchical planning, least-commitment planning, and opportunistic planning, and it is reviewed in section 4.4.3 of this paper.

4.3.1 Hovy's PAULINE

PAULINE (Planning And Uttering Language In Natural Environments) plans and realizes natural language paragraphs that satisfy pragmatic goals set by the user ([Hovy 87], [Hovy 88b], [Hovy 88c]). The task of planning in PAULINE is distinguished from that in other systems reviewed so far in this paper because PAULINE attempts to satisfy a large number of pragmatic goals by means of satisfying rhetorical goals that vary the style and slant of the text.

In the course of devising a planning paradigm to generate a plan that satisfies the wide variety of goals, Hovy recognizes limitations of using traditional top-down planning alone — be it hierarchical or skeleton planning. Top-down planning is *prescriptive* planning, he says, that is well suited for generating a plan for tasks like topic collection or topic ordering; however, it cannot be used to generate a plan that satisfies rhetorical or pragmatic goals. For example, goals like ‘‘be formal’’, ‘‘be friendly’’, or ‘‘impress the listener’’ cannot be achieved by a plan generated by a prescriptive planner because these goals cannot be achieved, flushed from the goal list, and forgotten. Furthermore, these goals cannot be achieved fully. They are goals to be ‘‘kept in mind’’ throughout the course of text generation.

Therefore, Hovy realizes that PAULINE needs to use a *limited-commitment* (or least-commitment) planning paradigm where prescriptive planning (hierarchical planning) is used to make plans for those tasks that are easily performed in top-down fashion, and restrictive planning is used for other tasks that cannot be achieved by a fixed-plan. These tasks, typically pragmatic, can be accomplished gradually by restricting the problem and by limiting the number of open alternatives. We now look at this interleaved planning paradigm more closely by identifying the planning components and the interleaved planning process itself.

State of the World

PAULINE has a very rich representation of pragmatic information about 1) interlocutor's personal characteristics, 2) speaker's goals with respect to the hearer, and 3) the conversational atmosphere all of which constitute together a state of the world.

More specifically, Hovy includes factual knowledge, opinions, emotional states, and interpersonal relationships of the interlocutors in the representation of a state in PAULINE. He also includes the speaker's goal with respect to the hearer in terms of the intended effect of the speaker on the future behavior, opinion, and relative status of the hearer. Finally, he considers the physical and social setting in which a conversation takes place. The summary of pragmatic information representing a state in PAULINE is shown below:

- Interlocutor's Personal Characteristics:
  - knowledge of the topic (speaker and hearer) — *expert, student, novice*
  - interest in the topic (speaker and hearer) — *high, normal, low*
  - opinions of the topic (speaker and hearer) — *good, neutral, bad*
  - language ability (hearer) — *high, normal, low*
  - emotional state (speaker and hearer) — *happy, angry, calm*
- Speaker-Hearer Relationship:
  - depth of acquaintance — *friends, acquaintances, strangers*
  - relative social status — *dominant, equal, subordinate*
  - emotion — *like, neutral, dislike*
- Conversational Atmosphere (setting):
  - time — *much, some, little*
  - tone — *formal, informal, festive*
  - conditions — *good, normal, noisy*

Operators

Operators in PAULINE are as rich as the representation of the state of the world in PAULINE, naturally. Basically, an operator in PAULINE is an act of selecting from a variety of choices that will eventually affect the style of the output text. For example, at the stage of topic collection, selecting a topic that the hearer is ignorant of will increase the knowledge of the hearer, and at the stage of word choice, selecting informal words or using slang will set the tone of the conversation as more informal or festive. The types of decisions that PAULINE makes are:

- **topic collection**: Collect aspects of the topic and related items as sentence topics. In addition to inclusion/exclusion of a topic, PAULINE uses the following two strategies in collecting a topic:
  - *evasion*: Indirectly refer to the topic so that the hearer will infer the topic by himself. This strategy is useful when generating a text that satisfies the goal of delivering a sensitive message while not offending the hearer.
  - *selectivity*: Say only part of the topic in order to emphasize or de-emphasize certain aspect of the topic or to give a biased version of the story.
- **topic organization**: Find appropriate groupings and interpretations of candidate topics; find appropriate ways to juxtapose topics in phrases. The order of topic presentation may give significantly varying effects on the hearer; examples are found in [Hovy 88b].
- **sentence organization**: For each topic, select appropriate subject, pre-sentence clauses, verb, predicate clauses, etc., and order them. The process of sentence organization can be guided by a criteria like *focus* in [Mckeown 85].

- **clause content and organization**: Determine and order the contents of clauses and noun groups. For example, choice of adverbs play a large role in communicating affect.
- **word choice**: Select appropriate words and phrases. Choosing one lexical item over another can vary the meaning of the sentence significantly.

The variety of choices enables PAULINE to generate many different types of text with different slant and style such that each type of text satisfies different goal(s) that the speaker has intended; however, because of the size of the operator search space, the text planning task becomes complex in PAULINE. In order to reduce the task to a more manageable size, Hovy introduces an intermediate layer between pragmatic goals and these operators, namely the rhetorical goals; they will be discussed in what follows.

Goals

The description of goals in PAULINE is perhaps best captured by the following quotation from [Hovy 88b]:

> ... any identifiably distinct collection of information that is activated to guide the behavior of the system toward a desired specific final state can be called a goal; in practice, we dignify those collections that we consider somehow ''natural'' starting points by calling them goals, and the rest we simply call strategies or plans. p.26

In other words, just about any action in PAULINE can be considered a goal, and indeed, there are a large number of goals in PAULINE which can be categorized as pragmatic goals and rhetorical goals. The *Pragmatic goals* (in particular, interpersonal goals) are listed in 4-9, and these goals are supposed to be achieved by application of the operators described above under ***Operators***. However, the wide range of decisions from syntactic choices to topic collections are too complex and numerous to be applied directly in satisfying speaker's pragmatic goals. Therefore, Hovy introduces *rhetorical goals* that play the role of achieving the pragmatic goals indirectly; they satisfy the speaker's intended pragmatic goals by creating various stylistic effects on the text. Figure 4-10 show the list of rhetorical goals in PAULINE.

Plans

A plan in PAULINE is either a prescriptive plan or a restrictive plan. A *prescriptive plan* is a set of

1. goal: increase knowledge (i.e., instruct, describe, relate)
2. goal: access knowledge (i.e., query)
3. goal: reorganize knowledge (i.e., reinterpret, explain)
4. goal: ‘‘decrease’’ knowledge (i.e., confuse, obfuscate)
5. goal: make the topic seem good or bad, contrary to the hearer’s opinion
6. goal: make the topic seem good or bad, bolstering the hearer’s opinion
7. goal: activate or deactivate a specific goal in the hearer
8. goal: make the hearer like or dislike the speaker
9. goal: make the hearer respect (or, perhaps, disrespect) the speaker
10. goal: make the hearer feel closer to or more distant from the speaker
11. goal: make the hearer feel socially inferior to, equal with, or dominant over the speaker
12. goal: make the tone formal, informal, or intimate
13. goal: be hasty, normal, or effusive

Figure 4-9: Pragmatic Goals in PAULINE [Hovy 88b]

1. **RG:formality** (highfalutin, normal, colloquial): Highfalutin language is used for speeches and toasts
2. **RG:simplicity** (simple, normal, complex): Simple text has short sentences and easy words
3. **RG:timidity** (timid, normal, reckless): Willingness to include opinions
4. **RG:partiality** (impartial, implicit, explicit): How explicitly opinions are stated
5. **RG:detail** (details only, interpretations, both): Too many details can be boring to non-experts
6. **RG:haste** (pressured, unplanned, somewhat planned, planned): When there’s little time, you speak fast...
7. **RG:force** (forceful, normal, quiet): Forceful text is energetic and driving
8. **RG:floridity** (dry, neutral, flowery): Flowery text contains unusual words
9. **RG:color** (facts only, with color): Colorful text includes examples and idioms
10. **RG:personal** reference (much, normal, none — two ranges, for speaker and hearer): Amount of direct reference to the interlocutors
11. **RG:openmindedness** (narrow-minded, open-minded): Willingness to consider new topics
12. **RG:respect** (arrogant, respectful, neutral, cajoling): Communicating relative social status

Figure 4-10: Rhetorical Goals in PAULINE [Hovy 88b]

plan steps that direct actions to be performed. Such a plan is constructed by a prescriptive planner just as in hierarchical expansion planning; plans generated by KAMP or EPICURE fall into this category.

A *restrictive plan*, on the other hand, does not have the kind of structured format that is found in a prescriptive plan. In fact, a restrictive plan is not something that is constructed out of a planning process, but it is a collection of restrictions that persist. Whereas a prescriptive plan directs actions to be performed, a restrictive plan provides information that influences the decision making process when alternatives are encountered while executing the actions prescribed in the prescriptive plans.

Building a Plan

Two kinds of planning techniques are used in PAULINE — prescriptive planning that generates prescriptive plans and restrictive planning that ‘‘generates’’ restrictive plans. For the wide variety of decisions spread around the generation process in PAULINE, Hovy recognizes the limitations in using traditional top-down prescriptive planning alone to control the entire decision making process.

For example, although prescriptive planning is sufficient for topic collection and topic ordering, it is not adequate to be used for open ended alternatives encountered later on in the generation process such as choice of lexical terms or styles.

On the other hand, restrictive planning is useful for planning to achieve those goals that cannot be achieved by a specifiable set of actions but rather by an accumulation of decisions that collectively affect the satisfaction of the goals during the course of text generation — i.e. precisely those rhetorical goals listed in Figure 4-10.

Therefore, Hovy uses restrictive planning, in conjunction with prescriptive planning to control the planning process in PAULINE. In order to discuss the use of restrictive planning and the interleaving of the two planning methods, we include the architecture of PAULINE in Figure 4-11.

Figure 4-11: PAULINE's Architecture

As shown in the figure, PAULINE is provided with input topics and a set of pragmatic goals that activate a number of rhetorical goals. During the topic collection phase and most of the topic organization phase, prescriptive planning is used exclusively. Planning at these phases is similar to prescriptive planning as used in TEXT ([McKeown 82]). For some of the topic organization tasks and all of the realization tasks, either restrictive planning or a combination of prescriptive and restrictive planning is used. More specifically, for new topic inclusion, topic juxtaposition, and intrasentential slant and style, restrictive planning is used, and for topic interpretation, a combination of the two techniques are used.

As the arrow in the figure going from Realization box to two preceding boxes shows, there is two way control among each phase of generation. In particular, unlike systems like TEXT and TAILOR where strategic components and realization components are clearly separated with one way flow of control going from the former to the latter, PAULINE allows interleaving of the two processes. For example, during the topic collection phase, the prescriptive planner can consult the output of the realization process and decide whether to continue collecting additional topics (in the case where RG:detail is yet to be satisfied) or to to move on to the topic organization phase with portions of the message are underspecified, postponing some decisions until more linguistic context is available.

Conclusion

PAULINE attacks a wide range of problems in language generation and attempts to provide solutions to generating pragmatically appropriate text, a problem that has not been addressed by other generation researchers. In order to generate pragmatically appropriate text, Hovy realizes that it is not enough to plan for the content of the text alone, but it is also necessary to plan for the appropriate style and slant of the text. Thus, his text planning task includes not only "what to say" but also "how to say it" aspect of text generation. Consequently, the planning problem becomes large and complex. Therefore, Hovy uses two types of planning, prescriptive and restrictive planning, and interleaves text planning with realization. However, it is still a very large task to plan for both the content and the style of text, and thus Hovy leaves out much of his good suggestions in the implementation of PAULINE.

4.3.2 Meteer’s SPOKESMAN

SPOKESMAN is a natural language generation system ([Meteer 90a], [Meteer 89], [Meteer 90b]) that consists of two components: a text planner that selects and ‘‘orchestrates’’ information to be generated and MUMBLE ([McDonald 84]), a system that realizes the output of the text planner into English sentences. The text planner in SPOKESMAN is unusually concerned with the expressibility of the content it has planned. As a result, SPOKESMAN generates only content that is guaranteed to have matching linguistic counterparts in English. Furthermore, SPOKESMAN is concerned with the efficiency of the generation system. Therefore, unlike other NL generation systems that sacrifice efficiency by allowing feedback from the realization component to the text planner in order to re-plan for content that cannot be expressed, the text planner in SPOKESMAN uses a *least-commitment planning* strategy to ensure the expressibility of each newly added content. In fact, SPOKESMAN goes as far as providing no feedback loop such that absolutely no unexpressible content is allowed to be generated.

In order to use the least-commitment planning strategy, Meteer introduces multiple levels of representation into what used to be a two layer representation of text plan and realized text. Therefore, the task of planning in SPOKESMAN can be viewed as generation of a text plan that is guaranteed to be realized into English text through progressive refinement of the plan.

State of the World

The representation of the state of the world in SPOKESMAN is almost Utopian when it comes to representing beliefs, knowledge, and interpersonal relationships of speaker and hearer — only *correct* knowledge is represented, and once it is delivered from the speaker to the hearer in the form of grammatically correct natural language, the hearer understands it exactly as the speaker has intended.

Because of this simplification, the text planner in SPOKESMAN is free from the preoccupation with any unintended changes that might happen to the state of mind of the hearer, and thus SPOKESMAN omits a lot of information about the state of the world that other generation systems (e.g. PAULINE) have to deal with. As a result, SPOKESMAN, like TEXT, lacks the ability to handle dialogue and follow-up questions.

Operators

The plan operators in SPOKESMAN are *abstract linguistic resources* and the constraints on the composition of those resources. Abstract linguistic resources are projections of concrete linguistic resources — the lexical resources (nouns, verbs, and adjectives) that carry the content and the grammatical resources (morphological markings and phrase structure) that carry the structural information. By representing its plans in terms of abstract linguistic resources, the text planner is guaranteed to generate a plan that can be realized into grammatically correct natural language sentences.

Meteer claims that a single *vocabulary* of abstract linguistic resources may correspond to a single word or a paragraph length text, and thus while language is comparable in power to RST (see Section 4.4.1) in that it provides vocabulary for the organization of units of clause-size and larger, it is more powerful because it allows the text planner to address the internal composition of clause-size elements.

Although her approach seems appealing, Meteer has yet to come up with a set of abstract linguistic resources that meet her claims. She has found linguistic work that provides vocabulary for lexical resources, but her current representation of vocabulary for grammatical resources is much too vague and weak to be comparable to RST, I think.

Goals

The ultimate goal of SPOKESMAN is the communicative goal of transferring knowledge or instruction of the speaker to the hearer. However, as mentioned before, SPOKESMAN assumes a model of the world in which the hearer understands the message of the speaker exactly as intended as long as the message is in the form of grammatically correct natural language.

Therefore, the goal of text planner in SPOKESMAN is reduced to that of selecting and including into the plan only an information that can be realized by the linguistic components. Meteer does not mention what to do about information that cannot be expressed by the abstract linguistic resources, which, I think, should be considered a case of a failed goal.

Plans

A plan in SPOKESMAN, called *Text Structure*, represents the utterance being planned in terms of abstract linguistic terms. During each least-commitment planning cycle, Text Structure accumulates the commitments made so far in the utterance such that those commitments constrain the choice of other resources that will be added in the latter cycles. Text Structure is implemented by a tree-like data structure, and when the planning is over, it contains the following information:

- **Constituency**: Each node in the tree has a ‘‘contents’’ which is an object representing the information expressed in that constituent. The information contained in each node may correspond to a constituent ranging in size from a single word to a whole paragraph.

- **Structural relations among constituents**: Each node is marked by its functional relations to their parents, children, and siblings, for example as ‘‘coordinate’’, ‘‘head’’, ‘‘argument’’, or ‘‘adjunct’’.
- **Semantic category the constituent express**: In addition to the name of the constituent, each node contains the lexical heading and the semantic category the constituent expresses.

Once Text Structure is built, it is traversed depth first by the linguistic realization component, and at each node, the abstract constituent is mapped into corresponding concrete linguistic terms.

Building a Plan

Meteer explains that Text Structure is built by two interleaved but distinct activities. One is the *selection* activity that chooses information to be included in the utterance, and the other is the *orchestration* activity that composes and organizes the selected units. However, the selection activity is a completely independent process and its only contribution to the construction of a Text Structure is the selection of a pool of information to be included. After that, everything is done by the orchestration process which creates the tree and expands the tree with the vocabulary of abstract linguistic resources that will map the given pool of information into the linguistic terms. More specifically, the orchestration process extends the Text Structure by traversing it depth first and by applying one of the following two actions at the node encountered:

1. expanding nodes whose contents are composite objects by using predefined templates associated with the object type (such as expanding an ‘‘event’’ object by making its arguments subnodes)
2. adding units into the structure at new nodes. The units may come from an already positioned composite unit or they may be individuals handed to the orchestrator by an independently driven selection process.

Conclusion

In this section we reviewed the use of the least-commitment planning technique in SPOKESMAN. Meteer introduces an added layer between text plan and linguistic terms called abstract linguistic resources which is a projection of concrete linguistic resources. The task of text planning thus becomes a progressive refinement of the abstract representation of the utterances until they are represented entirely in terms of the abstract linguistic resources.

Although the use of least-commitment planning and the elimination of the feedback loop from the realization component to the text planning component processes an efficient algorithm for text generation, there is one question that needs to be answered: can we really find a ‘‘good’’ set of vocabulary that represents the expressiveness of natural language and completely ensures the expressibility of the generator’s text plan? In AI planning research, it is recognized (and Meteers herself admits) that least-commitment planning without the feedback loop is inadequate for general purpose planning ([Cohn & Feigenbaum 82]), and thus most least-commitment planners usually do allow a feedback loop — e.g. [Stefik 80]. Meteer argues that with text planning, unlike with general problem solving, it is possible to predict all the possible situations since we know a great deal about the structure of the language, but she leaves the task of finding such a set of vocabulary as a ‘‘long range challenge’’ of her work.

4.4 Opportunistic Planning

Planning for natural language generation is a complex task. In the review of Hovy’s PAULINE system, in section 4.3.1, we showed that considering the pragmatic context within which language generation takes place complicates the process of text planning so as to necessitate the use of an additional planning technique, restrictive planning. In this section, we review another system by Hovy that uses opportunistic planning to control the decisions to include additional or supporting material and to exclude undesirable or untrustworthy material based on pragmatic context.

Also, in this section, we discuss the work of Nirenburg which uses opportunistic control in text planning. In his system, Nirenburg uses opportunistic control not only to decide whether to include information in or exclude information from the pre-selected pool of information, but also to decide how to present the selected information, which is, as he points out, another way of deciding *what* to present. These decisions are based on discourse, referential, lexical, and syntactic provisions in addition to the pragmatic context discussed in Hovy’s system.

Finally, we comment on the ideas suggested by Suthers[7] in [Suthers 90]. In his writing, Suthers suggests the combined use of prescriptive (see Section 4.3.1), restrictive, and opportunistic planning for the text planning task where opportunistic control is used to modify the knowledge pool, generated by two other planning techniques, that contains propositions to be realized as text.

---

[7] The implementation of his ideas was not available by the time of this writing.

4.4.1 RST Based Planning

RST (Rhetorical Structure Theory) is a theory of text organization, developed by Mann and Thompson ([Mann & Thompson 87a]), that characterizes the structure of a text in terms of the relations that hold between two successive pieces of text. Although RST was originally intended to be used as a theory that *explains* the relations among coherent sentences, the relations in RST are formalized as plan operators to be used in *planning* coherent paragraphs by [Hovy 88d], [Moore & Swartout 91], and [Moore & Paris 89].

In this section, we review RST briefly, and then see how Hovy formalizes and applies it in hierarchical planning of natural language generation. Later on in section 4.5.1 we look at the formalization of RST by Moore, Paris, and Swartout and their application of RST in reactive planning of natural language generation.

4.4.1.1 RST

After studying hundreds of paragraphs, Mann and Thompson proposed a set of approximately 25 rhetorical relations that normally occur between portions of a coherent English text. Rhetorical Structure Theory (RST) postulates that a paragraph is coherent if all of its parts can be related by these relations.

RST has four kinds of objects: relations, schemas, schema applications, and structures. RST *relations* hold between two non-overlapping text spans called *nucleus* and *satellite* such that a nucleus (N) represents that item in the pair which is most essential to the writer's purpose, and a satellite (S) represents the other item in the pair that is related to the text in nucleus in one of the ways defined by RST relations. The names of all the RST relations are listed in Figure 4-12.

Each of the rhetorical relations are defined in terms of the following four fields:

1. Constraints on the Nucleus
2. Constraints on the Satellite
3. Constraints on the combination of Nucleus and Satellite
4. The Effect (and the locus of the effect)

For example, the PURPOSE relation defined in Figure 4-13 accounts for the relationship between the pair of clauses 1 & 2 below:

1. To see which Syncom diskette will replace the ones you're using now,

- Circumstance
- Solutionhood
- Elaboration
- Background
- Enablement and Motivation
  - Enablement
  - Motivation
- Evidence and Justify
  - Evidence
  - Justify
- Relations of Cause
  - Volitional Cause
  - Non-Volitional Cause
  - Volitional Result
  - Non-Volitional Result
  - Purpose
- Antithesis and Concession
  - Antithesis
  - Concession
- Condition and Otherwise
  - Condition
  - Otherwise
- Interpretation and Evaluation
  - Interpretation
  - Evaluation
- Restatement and Summary
  - Restatement
  - Summary
- Other Relations
  - Sequence
  - Contrast

Figure 4-12: RST Relations [Mann & Thompson 87a]

2. send for our free ‘‘Flexi-Finder’’ selection guide and the name of the supplier nearest you.

In this example, the first clause is a satellite of the second clause, and the relation between the two is a PURPOSE relation because the first clause presents the activity through which the situation in the first clause is realized.

Relation name:	PURPOSE
Constraints on N:	presents an activity
Constraints on S:	presents a situation that is unrealized
Constraints on N+S:	S presents a situation to be realized through the activity in N
The effect:	Reader recognized that the activity in N is initiated in order to realize S
Locus of the effect:	N and S

Figure 4-13: Definition of RST Relation PURPOSE [Mann & Thompson 87a]

There are five primitive patterns called *schema*, defined in terms of the RST relations, that specify how spans of text can co-occur. Each of the five schemas can be varied and combined with other schemas under the set of rules called *schema applications*. The result of an RST based text analizer, called *text structurer*, consists of text spans structured in terms of these schemas and schema compositions.

Although Mann and Thompson mention about *Constructive RST* and its use in text planning ([Mann

& Thompson 87b]), they only provide a draft procedure that is entirely a human executed procedure. Therefore, Hovy's text structurer ([Hovy 88d], [Hovy 91]) is the first implementation that uses RST in construction of text.

4.4.1.2 Hovy's STRUCTURER

The planning task of Hovy's text structurer — henceforth, STRUCTURER — is slightly different than that of a content planner (or text planner) we've seen so far. The problem that the STRUCTURER is concerned with is how to organize and present a set of clause-sized chunks of input elements into a coherent paragraph. Therefore, Hovy makes an implicit assumption that there is a device that identifies the input to the STRUCTURER in advance. (Some researchers question this assumption because they believe that the tasks of choosing what to say and choosing a strategy for saying it cannot be divided [Moore & Swartout 91] [Paris 91]).

Given input propositions, the STRUCTURER uses a formalized version of the RST relations (shown below) as plan operators to search the permutation space of possible paragraphs that can be formulated from the propositions. The output of the STRUCTURER is a structure that covers all the propositions in a coherent paragraph described in terms of the RST relations.

Using RST relations instead of McKeown's schemas (see Section 4.1.1) as plan operators has an advantage because the paragraph trees constructed in terms of the RST relations contain information about the rhetorical relations between any two parts of the text. Also the relations control spans of text as small as two clauses, which is more fine grained than what schemas can control. These advantages do not come for free, however. Opportunistic assembly of an RST paragraph tree from a set of independent RST relations is a lot more costly than instantiating and traversing a schema because at each cycle of the growth of the paragraph, each RST relation has to be tested for its applicability.

In an attempt to find a middle ground while taking advantage of the selectivity of schemas and the dependencies of RST relation, Hovy develops the notion of growth points that restrict the number of applicable RST relations at each cycle of the growth of the paragraph by allowing growth of only those relations whose goals appear in the growth point fields of the currently selected RST relations. Although this approach has added the computational advantage of schemas to the STRUCTURER, at the same time it brought their inflexibility. In fact, as Paris points out [Paris 91], the revised RST relations resemble McKeown's schemas with *rhetorical predicates* replaced by *rhetorical relations*.

In order to circumvent the added problem of inflexibility, Hovy introduces varied treatment of growth points. Depending on additional criteria such as the availability of time or space (which are somewhat *ad hoc*), growth points may be treated only as *suggestions*. The process of deciding whether to treat them as mere suggestion or strong requirement or something in between is opportunistically guided by rhetorical goals like RG:haste or RG:verbosity (see Figure 4-10) and thus flexibility is regained.

In Hovy's terms, *top-down planning* is straightforward hierarchical planning with strongly enforced growth points, and *open-ended planning* is opportunistic planning with growth points treated as optional suggestions. We now look at these two planning techniques as well as the components of planning used in the STRUCTURER.

State of the World

A state of the world in STRUCTURER, as in KAMP, is represented by the combination of the state of the domain objects and mental state of the agents in the world. However, the state of domain objects is paid very little attention since the task of STRUCTURER is exclusively about the speaker's discourse plans rather than domain plans, as in NOAH, or even a combination of the two, as in KAMP. Also, even agent's mental states are not influential because the focus of the STRUCTURER is not on changing the mental state of the agents. The focus is on organizing pre-selected units of messages into a coherent paragraph according to a set of rules embedded in the plan operators that are pre-defined and unaffected by the changes in hearer's mental states.

Operators

Plan operators in STRUCTURER are exactly those RST relations defined by Mann and Thompson except that the constraints on *nucleus*, *satellite*, and *nucleus + satellite* are viewed as requirements that must be met by some piece of text before it can be incorporated coherently into the preceding text. Likewise, the effect field of the RST relation is viewed as the characterization of the goal(s) that the operator achieves.

In formalizing the RST relations to be used as plan operators, Hovy represents the constraints as communicative intent of the speaker in terms of the formal theory of rational interaction developed by, among others, [Cohen & Levesque 85]. For example, the PURPOSE relation shown in Figure 4-13 is formulated as the PURPOSE plan operator shown in Figure 4-14. The RST relation in Figure 4-13 is a descriptive relation, and it is used only to *described* a text. However, the RST based plan operator in 4-14 is an action generator, and thus, it actually changes the state of the world; the PURPOSE operator affects the state of mind of the hearer such that the hearer shares the belief of the speaker (BMB) that there is a PURPOSE relation between an act, *?act-1*, and a state, *?state-1*. Like other plan operators, an RST plan operator is executed under some preconditions. For the case of PURPOSE operator, the preconditions are those listed under *Nucleus requirements/subgoals* and *Satellite requirements/subgoals*. Paraphrased in English, they mean that both speaker and the hearer should believe 1) that an agent, *?agent-1* has the goal to achieve a state, *?state-1*, 2) that the result of an action, *?act-1*, is another action, *?act-2*, and 3) that the object of the second action, *?act-2* is to achieve the state, *?state-1*.

Name: PURPOSE

Result: 1. (BMB SPEAKER HEARER (BEL ?agent-1 (RESULT ?act-1 ?state-1)))
2. (BMB SPEAKER HEARER (PURPOSE ?act-1 ?state-1))

Nucleus requirements/subgoals:
1. (BMB SPEAKER HEARER (ACTION ?act-1))
2. (BMB SPEAKER HEARER (ACTOR ?act-1 ?agent-1))

Nucleus growth points:
. . . {omitted}

Satellite requirements/subgoals:
1. (BMB SPEAKER HEARER (STATE ?state-1))
2. (BMB SPEAKER HEARER (GOAL ?agent-1 ?state-1))
3. (BMB SPEAKER HEARER (RESULT ?act-1 ?act-2))
4. (BMB SPEAKER HEARER (OBJ ?act-2 ?state-1))

Satellite growth points:
. . . {omitted}

Figure 4-14: PURPOSE Plan Operator [Hovy 91]

Goals

Goals in STRUCTURER are communicative and rhetorical goals. A communicative goal is to transfer speaker's belief or intention to the hearer. For example, ''the purpose of turning off a switch is to darken the room.'' could be the belief of a speaker that the speaker wants to transfer to the hearer.

This happens to be the goal that can be achieved directly by the PURPOSE plan operator in Figure 4-14. (Instantiate *?act-1* with ‘‘turning off’’ *?state-1* with ‘‘darkened room’’.)

Preconditions of a plan operators are communicative goals (or sub-goals) themselves. To continue our example, the definition of the PURPOSE plan operator requires the following beliefs of the speaker to be conveyed to the hearer: 1) ‘‘Turning off the switch results in turning off the light.’’ and 2) ‘‘Turning off the light darkens the room.". Unless the model of the hearer indicates that the hearer already knows about them, these beliefs should be communicated to the hearer, and thus they become communicative goals themselves.

In addition to the communicative goals, STRUCTURER takes the rhetorical goals into consideration when making decisions about whether to include or exclude a plan operator at an optional growth point. At the time of this writing the criteria for the inclusion/exclusion is not fully developed, but Hovy indicates the possibility of using many of the rhetorical goals he has identified in his thesis, [Hovy 87] (also see Section 3), notably, RG:simplicity, RG:verbosity, RG:detail, and RG:haste.

Plans

A plan in STRUCTURER is a *paragraph structure tree* within which each and all of the input units are related with others by one or more of the RST plan operators. Hovy does not justify the advantage of representing the content of a paragraph in a tree form, and it is not obvious whether tree representation of a paragraph buys much for text generation, especially when it is true that the relations among the contents of a text may require cyclic representation. However, it is a convenient (if not logical) representation of the paragraph considering that the children of a node represents the plan operator that achieves the sub-goals (preconditions) of the plan operator at the parent node. An example of a plan is shown in Figure 4-15 with the text generated from it.

It is possible that more than one paragraph tree structure is produced. In that case, the planner selects the best tree according to the criteria embedded in the planner such as ‘‘select one with the least nodes.’’ Since the operators in the plan are RST operators, executing the plan corresponds to generating English sentences from the plan. This is done by traversing the selected best tree in left-to-right order and sending the leaves as input to Penman to be generated in English and the relations at branch points to be generated as interclausal relation words or phrases.

```
                           SEQUENCE
                           /      \
                          n        s
                         /          \
           CIRCUMSTANCE               SEQUENCE
               /  \                       /  \
              n     s                    n     s
             /       \                  /       \
    ELAB-ATTRIB     ELAB-ATTRIB   ARRIVE11400     E107   {Knox will
      /  \                  /  \    {Knox will         load for
     n     s               n     s    arrive on 4/24}     4 days}
    /       \             /       \
  E105  READN2211408  POSTN11410    HEADING11416  {Knox is heading SSW}
   :       {Knox is C4}  {Knox is at 18N 79E}
{Knox is en route to Sasebo}
```

Knox, which is C4, is en route to Sasebo. It is at 18N 79E, heading SSW. It will arrive on 4/24. It will load for 4 days.

Figure 4-15: Paragraph Structure Tree and Corresponding Text [Hovy 91]

Building a Plan

As mentioned above, there are two ways of building a plan depending on the treatment of the growth points: top-down planning and open-ended planning.

The top-down planner, like KAMP and EPICURE, is a hierarchical planner. Given one or more communicative goals, it finds RST plan operators whose *result* field matches some or all of these goals while satisfying all the requirements in the *Nucleus requirements/subgoals:* and *Satellite requirements/subgoals:* fields. If all the growth points in the so found operator(s) are fulfilled by the input features, then the planning process terminates. If, on the other hand, one or more growth points remain unfulfilled, then they are put on the agenda as subgoals to be satisfied. For each of the subgoals in the agenda, the same hierarchical planning procedure is recursively applied until either all the subgoals are satisfied or all input elements have been used and unsatisfied subgoals remain. It is not clear how the remaining subgoals are satisfied; in [Hovy 88d], Hovy briefly mentions that the STRUCTURER could request more input with desired properties from the encapsulating system, but he doesn't talk about the resolution of the possible conflict that might arise when expanding the content already delimited by the encapsulating system.

The open-ended planner is an opportunistic planner. Like the top-down planner, it first finds the RST plan operator that achieves the given communicative goal. However, when it comes to expanding the growth points of the so found plan operator, the open-ended planner adapts interactively to the environment and interlocutors. Depending on the circumstances and the rhetorical goals that need to be satisfied, the open-ended planner decides whether or not to expand the growth points. Work is in progress in developing the criteria that control the inclusion and priorization of candidate growth points. For example, the following criteria may play a role in determining the inclusion or exclusion of the growth points:

Reasons for inclusion

1. Explicit goal to communicate the extra material
2. Rhetorical-stylistic considerations: balance/parallelism of text
3. Hearer’s lack of knowledge
4. Hearer’s inability to understand without extra information
5. Hearer asks for more information

Reasons for exclusion

1. Lack of time or space
2. Unpleasant/undesirable effects of extra material
3. Lack of known detail
4. Untrustworthiness of material

As criteria for ordering the growth points, Hovy considers using the focus theory described in [McCoy & Cheng 91].

Conclusion

Hovy’s STRUCTURER is the first implementation that uses RST relations in text planning. Given a communicative goal and input representing the content of the text to be generated, STRUCTURER uses two different kinds of planning, top-down planning and open-ended planning. The top-down planning is used to structure input as a hierarchical plan within which each unit is related to others by one or more of the RST plan operators. When realized into a paragraph, they achieve the given communicative goal. The open-ended planning approach enables the STRUCTURER to generate more flexible and varied paragraphs by opportunistically incorporating various rhetorical goals and other circumstantial criterias. In our opinion, however, the current implementation of the open-ended planning lacks criteria that determine inclusion/exclusion of a growth point, and thus, the opportunistic choice of growth points is driven by *ad hoc* decisions.

STRUCTURER is criticized by a number of researchers for two reasons ([Moore & Swartout 91], [Paris 91], [Meteer 88], [Suthers 90], and section 4.4.3 of this paper). First, the presumptions that STRUCTURER makes about the input clause units imply the existence of an underlying program that can ‘‘select’’ the message to be included. Moore and Swartout believe that any process which could identify such a message would have to do much of the work of a STRUCTURER anyway, and thus the task of choosing what to say and choosing a strategy for saying it cannot be divided. Second, the RST plan operators indicate, in the growth points fields, *what* description can be said, but they do not include information about *why* this information can or should be included at a particular point. Therefore, the STRUCTURER is not capable of identifying the *intentional goal* of the speaker for choosing a particular plan operator. This is a problem when it comes to answering follow-up questions. A solution for these problems are addressed by [Moore & Swartout 91]. (Also, see Section 4.5.1.)

4.4.2 Nirenburg's DIOGENES

DIOGENES (DIstributed, Opportunistic GENEration System) is a natural language generation system built by Nirenburg and his group at the center for machine translation at Carnegie Mellon University ([Nirenburg *et. al.* 89]). Because it is being developed within the context of a machine translation system, the planning task in DIOGENES is devoid of the usual problem of deciding *what to say*; it is assumed that an interpretation system delineates the content from analysis of the source language.

However, as we discussed in section 4.3.2, knowing what to say is quite different from being able to say it. In section 4.3.2 we saw that Meteer introduced an additional layer of abstract linguistic resources to ensure the expressibility of the content being generated. Unlike Meteer, Nirenburg uses concrete resources (*e.g.* lexical entries and other linguistic constructs) directly in planning the realization of selected content. Also, Nirenburg emphasizes the importance of the presentation of selected content, and makes the point that not only the decisions for *what to say*, but also those for *how to say it* may achieve the intentional goal of changing the state of the mind of the discourse consumer.[8] Therefore, his planner makes decisions about the inclusion or exclusion of pre-selected information based not only on their lexical and syntactic expressibility, but also on semantic and pragmatic appropriateness of the resulting expression.

---

[8] *Consumer* is how Nirenburg refers to what is normally called ‘‘hearer’’. In Nirenburg's terms, ‘‘speaker’’ is referred to as *producer*.

For such a planning task, Nirenburg finds the conduit model of planning architecture[9] inappropriate (even the modified architecture with the feedback loop among the components), and adopts a single component planning architecture where various knowledge sources are triggered and applied opportunistically to achieve the task.

State of the World

A state of the world in DIOGENES represents domain facts and objects as well as the state of the mind of the consumer. Nirenburg believes that in order to treat rhetorical plans using the same mechanism as domain plans, the representation of discourse state should include the following model of the discourse producer that he implements in DIOGENES:

- producer's knowledge about the objects, actions and states in a domain (domain model);
- a model of the consumer including the consumer's domain model;
- producer's knowledge about mental objects, actions, and states that allows him to create coherent text of a chosen style (discourse model);
- a text creation record needed for introduction of pronominal anaphora in the generated text or for reference to background knowledge produced during the course of conversation, among other uses.
- a processing agenda containing active instances of both domain and rhetorical goals and plans;
- knowledge about lexis and syntax of the target language.

Operators

The operators that change the state of the world in DIOGENES are knowledge sources (KS's) that posts decisions on the blackboard. Under the opportunistic control of the two ''meta'' knowledge sources, *obviation* and *retraction*, various knowledge sources are triggered to add/remove decisions to/from the blackboard. There are thirteen knowledge sources and they make decisions about the structure of the text, coreference links, lexical selection, syntactic selection, and constituent ordering. We categorize these knowledge sources as the plan operators of DIOGENES because they are precisely the means by which decisions about how to say it are made; as Nirenburg mentions, deciding *how to say it* is as important as deciding *what to say* when it comes to influencing the state of the mind of the discourse consumers.

Goals

---

[9]Decisions about text structuring, lexical selection, syntactic selection, and other decisions are made sequentially in a conduit model of planning architecture.

Goals in DIOGENES can be categorized into two kinds, communicative goals (or intentional goals) and rhetorical goals. Communicative goals are those that the producer may want to achieve in order to alter the state of mind of the discourse consumer. For example, modifying the consumer's semantic memory is a goal that the producer may achieve by giving a definition of a missing semantic entry, and modifying the episodic memory of the consumer is a goal that can be achieved by stating a fact or describing an object related to an event. Similarly modifying the emotional state, agenda, and other state of mind of the consumer are all examples of the communicative goals.

The rhetorical goals considered in DIOGENES are not as interesting as those considered in Hovy's PAULINE, for example, because the scope of the rhetorical goals in DIOGENES is limited to generating discourse that are cohesive in terms of causal and temporal links among the clauses (e.g. ‘‘provide a justification for an action’’); the rich relations that exists between rhetorical goals and communicative goals in PAULINE are missing in DIOGENES.

A goal in DIOGENES is represented as a part of the producer discourse model and it is implemented as a structure with four fields: a goal *type* which is either intentional or rhetorical, *preconditions* that must be met for the goal to be instantiated, a set of *plans* that are known to achieve the goal, and local *evaluation functions* that determine the best plan.

The evaluation functions in the representation of a goal do consider pragmatic and stylistic factors similar to a specific rhetorical goals of PAULINE, RG:haste and RG:force in Figure 4-10. However, they are not considered as goal themselves, but only as an optional parameters to the evaluation functions.

Plans

A plan can be either a simple primitive action such as an *inform* or *ask* speech act, or a complex action. A primitive action results directly in actual realization; since DIOGENES uses Lexical Functional Grammar for realization, a primitive action is represented using logical forms, where constituents are paired with their functional category ([Kay 79]). A complex action is represented as a decomposition of primitive actions.

Both primitive or complex plans are represented in a frame with slots for the plan name, preconditions, effects, and a pointer to goals that the plan can achieve.

Building a Plan

As in other AI opportunistic planning systems (e.g. [Hayes-Roth 85]), a plan in DIOGENES is constructed by opportunistic application of the plan operators. Each plan operator is applied when there is a task in the *agenda* that can be achieved by the operator and conditions posted on the blackboard that meet the preconditions for that operator. As a result of the application of the operator, the agenda is updated with a new set of tasks to be achieved. When the agenda is cleared, the planning task is completed.

As mentioned in section 2.2.5, however, earlier commitment to an undesirable sequence of operators may pose some constraints on the selection of a desirable sequence of operators later on. Consequently, less than desirable plans might be constructed or the entire planning process might be jeopardized. This problem is only more aggravated when the planning task is as complex as text generation.

As one solution to deal with this problem, Nirenburg introduces a special *control blackboard* and two control knowledge sources that operate on this blackboard called *obviation KS* and *retraction KS*. The knowledge sources use heuristic rules and cost analysis in order to evaluate the application of each operator and to make necessary removal and blocking of tasks on the agenda.

Nirenburg also simplifies the opportunistic planning process by encoding some implicit ordering of the operator application sequence. For example, the output of the *Discourse-Cohesion-KS* operator is a text structure skeleton that resembles McKeown's schema, and the rest of the operators are then opportunistically applied to fill in the slots in that skeleton.

Conclusion

In this section we looked at an example of a natural language generation system that uses opportunistic planning for its planning task of determining *how to say* a pre-selected set of information. Whereas in Hovy's STRUCTURER (see last section), the use of opportunistic planning is limited to inclusion/exclusion of optional information based on the pragmatic settings, in DIOGENES, opportunistic planning is used exclusively for the entire planning process. Unfortunately, however, the task of content delimitation is excluded from the planning process.

Presently, DIOGENES is still in its early stage of development, and thus, there is no available data to evaluate its performance. However, it is my opinion that the system will run very slow even with a large body of control heuristics. The problem is that each of the thirteen knowledge sources is large and vague. Allowing them to be triggered anytime during the planning process and still controlling them to converge to a successful text plan is an enormous challenge. I believe that a lot can be gained if some combination of planning techniques is used rather than sorely relying on opportunistic planning. In this regard, the *Discourse-Cohesioin-KS* that provides some skeleton of the text structure is an encouraging first step towards using more efficient hybrid planning techniques (i.e. combined skeleton and opportunistic planning).

4.4.3 Suthers' Suggestion

Suthers suggests a text planner model that uses a combination of prescriptive (see Section 4.3.1), restrictive (or least-commitment), and opportunistic planning techniques. Although an implementation is not available at the time of this writing, we will, nevertheless, review his suggested planning architecture [Suthers 90], and comment on his suggestions because the coordination of different planning techniques is of interest to us and has relevance to this section.

The motivation for Suthers' suggestion is the problem associated with using RST as the means for structuring a selected pool of information. In section 4.4.1.2 we saw that Hovy's STRUCTURER assembles an RST based paragraph structure tree by pattern matching RST operators to the input pool of clause-sized chunks of content. Although Suthers acknowledges the benefits of using RST over Schemas despite the cost of processing time overhead, Suthers questions the validity of using rhetorical abstraction altogether. Suthers argues that without knowledge about the inter-relations among the selected pieces of information, there is no way that generic plan operators like RST can capture the rhetorical relations among them. For example, if a concept and some background information about that concept are placed together in the pool of selected information, how would an RST operator that is ignorant of such relations among them be able to place the background information at the satellite slot of the BACKGROUND RST operator?

The answer Suthers provides is a mixed architecture where the processes of content selection and text structuring are combined such that during each *refine* $\rightarrow$ *retrieve* $\rightarrow$ *modification* cycle, selection and retrieval of coherent packages of knowledge are followed by a modification process that places ordering links among the selected propositions. Upon completion of the *modification* phase, the

Ordering phase is initiated and ordering of propositions is completed by further imposition of structure based on rhetorical and linguistic considerations.

It is during the modification phase that opportunistic planning is used. Details of the opportunistic planning method are not mentioned in the paper, but the need for opportunistic planning is justified by the kinds of tasks that is required of the modification phase. In particular, the process of filtering and augmenting the pool of propositions is guided by the control called *critics* which applies itself opportunistically whenever there is a need to remove or add propositions from/to the selected propositions based on the knowledge sources from the user model.

During the *refine* → *retrieve* → *modification* cycle, both prescriptive planning and restrictive planning techniques are used in order to satisfy the two kinds of goals mentioned in section 4.3.1 — *informative goals* are handled by prescriptive planning while *persistent goals* that remain active throughout a span of discourse are handled by restrictive planning.

The approach suggested by Suthers is interesting, and it could well be the solution to remedy the problems associated with relying on rhetorical abstractions like RST for the task of text structuring, but he maintains the conduit model of text planning architecture where the task of lexical and syntactic choices are pushed into the realization component which is completely separated from the rest of the planning tasks.

4.5 Reactive Planning (Execution Monitoring)

4.5.1 Moore's EXPLANATION GENERATOR

Moore's EXPLANATION GENERATOR is part of the Program Enhancement Advisor (PEA), which is a prototype expert system intended to aid users in improving their Common Lisp programs by recommending transformations that enhance their code ([Moore & Swartout 91], [Moore & Swartout 89], [Moore & Swartout 88], [Moore & Paris 89]). The planning task of the EXPLANATION GENERATOR is very similar to that of Hovy's STRUCTURER (see Section 4.4.1.2.) Given a discourse goal, Explanation Generator constructs the utterance plan in terms of RST plan operators (which are a little different from those in the STRUCTURER) using simple hierarchical planning. However, Moore *et. al.* realize that it is not possible to generate a single most appropriate utterance plan for any user and circumstances. Therefore, they adopt a *reactive planning* approach by providing a feedback loop from the user to the EXPLANATION GENERATOR as shown in Figure 4-16.

Figure 4-16: Architecture of EXPLANATION GENERATOR [Moore & Swartout 91]

After the *text planning* module in the diagram generates an utterance plan using hierarchical planning, the Penman text generation system realizes it into natural language text for the user. Upon receiving the text, the user may ask follow-up questions requesting for clarification or additional information, which is then fed back into the EXPLANATION GENERATOR for the planning of additional or modified utterances.

This reactive planning approach for explanation generation is not only more flexible and adaptive to the users, but it is also advantageous from the point of view of efficiency; rather than planning for a one-shot, all inclusive utterance plan, the planner can produce a minimal plan with only the information that is readily available to it.

In order for the system to deal with the follow-up questions, Moore and Swartout incorporate the following capabilities in their system design:

- to plan responses such that the intentional and rhetorical structure of the responses is explicit and can be reasoned about,
- to keep track of conversational context by remembering not only what the user asks, but also the planning process that led to an explanation,

- to taxonomize the types of (follow-up) questions that are asked and their relationship to the current context, and
- to provide flexible explanation strategies with many and varied plans for achieving a given discourse goal.

We examine how these capabilities are realized while reviewing the elements and process of planning.

Operators

The plan operators in the EXPLANATION GENERATOR are based on RST relations like the plan operators in Hovy's STRUCTURER. However, Moore and Swartout's formalization and modifications make their RST plan operators superior to those of Hovy's in two respects. First, the plan operators in the EXPLANATION GENERATOR do not require that all of the topics to be discussed are given to the planner as input. Second, the plan operators in the EXPLANATION GENERATOR incorporate the intentional goals that are omitted in Hovy's plan operators. This enables the EXPLANATION GENERATOR to plan the answers to follow-up questions and to recover when the hearer does not understand the response produced. Therefore, the plan operators in the Explanation Generator are used not only in identifying relations among portions of text, but they are used also to explain the reason *why* a portion of text appears in the response.

Looking more closely at the plan operators, each operator consists of the following four fields:

- *an effect:* a characterization of goal(s) this operator can achieve. An effect may be either an intentional goal or rhetorical relation; for example,
    - *intentional goal:* Achieve the state where the hearer is persuaded to do act. (See Figure 4-17<a>.)
    - *rhetorical relation:* Provide motivation for an act. (See Figure 4-17<b>.)
- *a constraint list:* a list of conditions that *must* be true before the operator can be applied. Constraints may refer to facts in the system's knowledge base or in the user model, whereas in Hovy's STRUCTURER, all the facts had to be from the pre-selected pool of input clause-units).
- *a nucleus:* the main topic to be expressed (required).
- *satellites:* additional information needed to achieve the effect (may be optional).

At the time of this writing, Moore *et. al.* have implemented approximately 75 operators. The reason

NAME: PERSUADE-USER-TO-DO-ACT
EFFECT: (PERSUADED ?hearer (GOAL ?hearer (DO ?hearer ?act)))
CONSTRAINTS: (AND (GOAL ?speaker ?goal)
(STEP ?act ?goal)
(GOAL ?hearer ?goal)
NUCLEUS: (FORALL ?goal (MOTIVATION ?act ?goal))
SATELLITES: nil

In English it means: To achieve the state in which the hearer is persuaded to do an *act*, IF the *act* is a step in achieving some *goal(s)*, that the hearer shares, THEN motivate the *act* in terms of those *goals(s)*.

<a> Plan Operator for Persuading User to Do an Act

NAME: MOTIVATION
EFFECT: (MOTIVATION ?act ?goal)
CONSTRAINTS: (AND (GOAL ?speaker ?goal)
(STEP ?act ?goal)
(GOAL ?hearer ?goal)
NUCLEUS: (BEL ?hearer (STEP ?act ?goal))
SATELLITES: none

In English it means: To motivate an *act* in terms of a *goal*, IF the *act* is a step in achieving the *goal*, and the hearer shares the *goal*, THEN make the hearer believe that the *act* is a step in achieving the *goal*.

<b> Plan Operator for Motivation

NAME: RECOMMEND-ENABLE-MOTIVATE
EFFECT: (GOAL ?hearer (DO ?hearer ?act))
CONSTRAINTS: none
NUCLEUS: (RECOMMEND ?speaker ?hearer ?act)
SATELLITES: (((COMPETENT ?hearer (DONE ?hearer ?act)) *optional*)
(PERSUADED ?hearer (GOAL ?hearer (DO ?hearer ?act)) *optional*))

In English it means: To enable the hearer to have a *goal* of doing an *act*, recommend the hearer to do the *act*. When necessary, make the hearer competent to do the *act* and persuade him to have the goal of doing the *act*.

<c> High-level Plan Operator for Recommending an Act

Figure 4-17: Plan Operators of EXPLANATION GENERATOR [Moore & Swartout 91]

why there are more plan operators than there are RST relations (about 25) is because some of the plan operators are *high-level plan* operators like the one shown in Figure 4-17<c>. A high-level plan operator is a type of ‘‘schema’’ that is described in terms of either several speech acts like RECOMMEND, or a goal describing intended effects on the hearer which must eventually be satisfied by primitive RST plan operators such as MOTIVATION in Figure 4-17<b>. For a comprehensive list of the RST plan operators, see [Moore 89].

Goals

Goals in the EXPLANATION GENERATOR are discourse goals represented in terms of the effects that the speaker intends his utterance to have on the hearer. As a result of a query from the user or a result of reasoning in the encompassing PEA expert system, a discourse goal is posted to the EXPLANATION GENERATOR in order to interact with the user. The following is an example of a discourse goal that says that the speaker wishes to achieve the state in which the hearer (i.e., the user) has adopted the goal of doing the replacement. (Figure 4-18 and 4-19 show the complete text plans that achieve this goal.):
(GOAL USER (DO USER REPLACE-1)) where

```
REPLACE-1 = (REPLACE   (actor USER)
                       (object CAR-FUNCTION)
                       (generalized-means FIRST-FUNCTION))
```

From the point of view of the expert system, often a discourse goal is used as a way of achieving its domain goal(s). For instance, the domain goal behind the discourse goal above is to transform the CAR function in the user's code into the FIRST function. In many instances, however, the system is capable of performing the transformation by itself without having to persuade the user to do it! In such cases, the expert system still persues the discourse goal in order to get approval from the user before making the transformation.

Plans

A completed *text plan* in the EXPLANATION GENERATOR is a lot more than a simple tree-like structure of plan operators (which is the case with the paragraph structure tree in Hovy's STRUCTURER shown in Figure 4-15). In addition to providing an explicit representation of the explanation produced by the system indicating how parts of the plan are related and the purpose that they serve, a completed text plan also contains information about assumptions made and alternative strategies that could have been selected at various points during the planning process. Global and local focus points are indicated in the text plan such that during the reactive planning phase this information can be used in responding to a follow-up question from of a user.

To illustrate completed text plans in more detail, we include two sample text plans that are constructed for the discourse goal:

(GOAL USER (DO USER REPLACE-1)).

The completed text plan in Figure 4-18 resulted from selecting the plan operator in Figure 4-17<c> with neither of the optional satellites expanded. (By default the planner does not expand the satellites.)

Then, if there is a feedback from the user, ‘‘Why?’’, the reactive planner examines the dialogue history and the current focus point in the text plan, *local-context*, and decides to expand one of the optional satellites:

(PERSUADED USER (GOAL USER (DO USER REPLACE-1)))

resulting in the new text plan shown in Figure 4-19.

Figure 4-18: Completed Text Plan for Recommendation [Moore & Swartout 91]

Building a Plan

Unlike Hovy’s STRUCTURER, EXPLANATION GENERATOR must plan an utterance by choosing what to say from the knowledge base of the expert system, the execution trace of the expert system’s reasoning, the user model, and the dialogue history containing past utterances, *and* then organize it into a coherent structure.

Given a discourse goal, the EXPLANATION GENERATOR finds all the operators from the plan library whose effect field matches the goal to be refined. From the set of candidate plan operators,

Figure 4-19: Completed Text Plan for Persuading the CAR-to-FIRST Transformation [Moore & Swartout 91]

EXPLANATION GENERATOR finds one operator based on the following heuristics[10]:

- Don't tell the user things he already knows (based on the user model).
- Don't repeat things that have been said already (based on dialogue history).
- Select an operator that is more detailed.
- Select an operator that makes no (or little) assumptions in finding variable bindings from the expert system's knowledge base and/or user model which satisfy the constraints in the constraint list of the candidate operator.

Once a plan operator has been selected, the planner is now faced with one more decision to make; i.e., in the process of instantiating the selected operator, should the optional satellites be expanded? For example, should the first satellites of the plan operator in Figure 4-17<c> be expanded? A simple hierarchical planner could either blindly expand all such subgoals until it cannot expand any

[10]More detailed selection process is discussed in [Moore 89].

further or make the decision based on the user model (i.e., if the user is not competent to perform the act, then tell him what he needs to know to do so.) The problem with the former method is that the planner will often end up generating much more information than is necessary. In the latter case, the planner fails if the user model is incomplete or incorrect which is often the case in the real life.

EXPLANATION GENERATOR solves this problem by using the reactive planning approach. When the planner does not know whether the expansion of a subgoal is beneficial or not (i.e. there is no relevant information in the user model), then it simply does not expand the subgoal. After the generation of the advisory text if the listener respond with ‘‘why?’’ or ‘‘huh?’’, then the planner considers expanding the subgoal(s).

In addition to expanding the unused optional subgoals, the reactive planner uses a number of other recovery heuristics including the following:

- If another plan exists for achieving a goal, try it.
- If alternative bindings exist for plan variables, try alternative bindings.
- If any assumptions were made in planning the last explanation, plan responses to make these assumptions true.
- If goal is to describe an entity or compare 2 entities, try giving example(s).
- If goal is to describe an entity and there is an analogous entity the user may be familiar with (from user model or dialogue history), draw an analogy to the familiar concept.

Conclusion

Moore *et. al.* apply the reactive planning approach to text planning in which the planner can employ feedback from the user in guiding its process of planning. In order to be capable of responding to the feedback, the planner not only must maintain the recorded history of the text plans used in producing responses, but it also must keep track of *why* and *how* it came up with those plans. Moore *et. al.* show how to use RST based plan representation in EXPLANATION GENERATOR to achieve both. As a result, EXPLANATION GENERATOR is a more efficient and robust text planner that is capable of generating more useful responses in an environment where user model is not complete or correct.

However, the robustness of the system is bounded by the coverage of the RST operators. First of all, there are only so many RST operators that can be applied (about 75), and it is not obvious that they cover the complex and various discourse structures of natural language text. Next, each RST plan operator is pre-written. Therefore, a follow-up question can only be answered properly if there is a pre-written entry in the *satellite* slot of RST that is specifically designed to handle that particular follow-up question. This is a rather significant limitation, and it is a limitation that is inherent in other purely reactive AI planners as well.

4.6 Increasingly Reactive Planning

4.6.1 Paris' Suggestion

Although the reactive planning approach shown in section 4.5.1 enables the EXPLANATION GENERATOR to plan adaptive utterances to the user, the adaptive capabilities of the EXPLANATION GENERATOR is limited by the rigidity of the RST plan operators. The 75 or so defined RST plan operators defined in the EXPLANATION GENERATOR are static operators that are pre-written by the designers of the text planner and unchanged throughout the course of a dialogue. This is undesirable considering that human beings are able to make modifications to their discourse strategies as well as the content of the utterances while involved in a dialogue.

Furthermore, it is very hard, if not impossible, to construct a complete and correct library of plan operators that covers every possible situation that might occur in a dialogue between a user and the system. Therefore, it is desirable for a text planner to have the capability of learning new plan operators and modify the existing ones adaptively to the users and situations.

Paris recognizes this need in the Explanation Generator and proposes a text planning facility that is capable of acquiring and changing its plan operators based on information presented to the user and the follow-up questions asked ([Moore & Paris 89], [Paris 91]). For example, consider the dialogue situation below:

System:	I recommend act1.
User:	Why?
System:	Because it is a step towards achieving goal1.

If this dialogue situation is repeated over and over again, the planner may learn a new plan operator by adding a step to its original basic plan so that with the new plan operator, the system generates the following utterance instead:

System:	I recommend act1 because it is a step towards achieving goal1.

In addition to adding a step to a plan operator, Paris proposes a number of other ways of modifying/creating plan operators including the following:

- starting with a set of basic operators and grouping them to form larger schema,
- changing a satellite from being optional to required,
- replacing a step of a plan operator, and
- adding or changing a constraint.

Paris does not call this new planning approach of hers the *increasingly reactive planning* approach, and actually she is only at the stage of experimenting with this approach; however, in several respects her planning approach resembles that of the increasingly reactive planning approach described in section 2.2.7 that is relatively new in the AI planning research as well.

5. Collaborative Planning: An Unexplored Area of Planning

Text planning in natural language generation has focused primarily on the task of planning a paragraph length text in response to a communicative goal on system user. There are a handful of generation systems that carry on a limited form of dialogue (*e.g.* [Moore & Paris 89], [Moore & Swartout 91]) and some that take a user model into account (*e.g.* [Paris 87], [McCoy 88], [Wolz 90]). However, the use of text planning is not found in a dialogue situation which requires that the generation system carries out a negotiation or argumentation with the user(s). Therefore, although there are many systems that *passively* engage in a dialogue in order to provide answers to the queries about the underlying data base or expert systems, there is no existing generation system that is capable of *actively* engaging in a dialogue, participating in a collaborative task environment where multiple agents involved in the task communicate with each other to achieve their common goals. This is curious considering the fact that a lot of work in AI planning has been concerned with modeling an intelligent agent that is capable of generating plans and communicating with other agents in multi-agent task domain. Intuition tells us that this is one of the most natural areas where an automated language generation system should be used. However, the main focus of generation research has been on the task of one-shot descriptions or responses rather than that of carrying out extended dialogue.

In order to provide insights and directions for the collaborative planning research in natural language generation, we observe the nature of collaborative planning and mention some of the related work in the field of AI planning. Then we briefly review recent efforts in the field of natural language understanding that models dialogue in a collaborative planning environment.

5.1 Collaborative Planning Situations

We have already discussed the situation for *cooperative distributed planning* in section 2.2.8. However, there are a greater variety of collaborative planning situations. For example, a number of recent works model non-cooperative multi-agent task environments in which compromises are reached by argumentation and negotiation among the conflicting agents ([Sycara 88], [Sycara 89], [Zlotkin & Rosenschein 88], [Safavi 91]). Some AI researchers in planning realize that the new tasks introduced by the collaborative planning make the planning paradigm used for single agent inadequate for multi-agent planning tasks, and they describe a new planning paradigm for the multi-agent planning domain ([Lansky 86], [Georgeff 86], [Pednault 87]).

Rather than discussing the details of these and other works in collaborative planning, we briefly summarize what a collaborative planning situation consists of in order to give a flavor of the new challenges faced by the AI planning researchers:

- Number of agents: Multiple agents.
- Goals
    - *Conjunctive*: Each agent persues its own set of goals. Thus, the goal to be satisfied is the conjunction of all the goals of the agents.
    - *Ill-specified*: Agents may persue conflicting goals which should be detected and resolved.
    - *Non-enumerable*: It might not be possible to enumerate all the goals of the agents because during the course of an interaction, new goals may be generated; *e.g.* agents change their mind.
- Nature of collaboration: A collaboration may not be always cooperative. In fact, it can take any of the following forms:
    1. *Cooperative*: Agents collaborate constructively in order to achieve their goals (conflicting or non-conflicting).
    2. *Non-cooperative*: Agents pursue their goals (conflicting or non-conflicting) without the intention to cooperate with others. An agent might even interfere with the activities of others intentionally in order to satisfy his own goals.
    3. *Independent*: Each agent works independently of one another. Their activitives may or may not interfere with one another.
    4. *Master-slave*: The goal of a master agent is persued by one or more slave agents.
- Planning Tasks: In addition to the usual problem solving tasks, the task of communication is necessitated.
    - *Synchronization*: Ordering of activities and simultaneous actions (*e.g.*, lifting a piano) require communication among the agents involved.
    - *Negotiation*: When a goal conflict takes place, it can be resolved by finding a compromise by negotiation. This limits the usefulness of traditional planning techniques which assumes that goals must be satisfied completely [Sycara 88].
    - *Argumentation*: An agent can affect other agent's plan by argumentation.

5.2 Approaches from Natural Language Understanding

Early research on planning for natural language understanding has been active in the representation of plans for task-oriented dialogues ([Allen and Perrault 80], [Cohen & Perrault 79], [Litman & Allen 87], [Pollack 86a]). Although a typical dialogue situation involves multiple agents (speaker and hearer), only one set of beliefs and intentions, those of the speaker, is represented in early works. The task of the hearer is then to infer the beliefs and intentions of the speaker and to react to them. Thus, in effect, the hearer is not treated as a sentient agent, and their model of plan representation is inadequate for collaborative planning.

Perhaps the first work in natural language understanding that has relevance to collaborative planning is Pollack's model of plan inference [Pollack 86b]. If the model of a plan assumes the beliefs of the speaker only, then there is no way that a hearer can recognize an invalid plan. For example, if the speaker asks the hearer to replace a lightbulb, but the hearer has broken wrists, then the speaker's plan should be invalidated. However, without the means to represent the discrepancies between the beliefs of the speaker (that the hearer is capable of replacing the lightbulb) and those of the hearer, plan invalidation cannot be modelled. Pollack recognizes this problem and includes a separate representation of beliefs and intentions for each agent in her model of plan inference. Furthermore, Pollack's representation of a plan, called *SimplePlan*, contains the mental state of the owner of the plan, rather than a data structure that consists of a sequence of actions to be executed by a random agent. Therefore, Pollack's representation of a plan is an adequate first step towards a model of a collaborative plan. More precisely, the SimplePlan of the single agent consists of four entries.

1. the agent's beliefs about the executability of his actions,
2. the agent's beliefs about the relationships between actions,
3. intentions of the agent to do actions, and
4. intentions of the agent to do actions to contribute toward the achievement of the SimplePlan itself.

Is it sufficient to model a collaborative plan by the collection of all the SimplePlans of the agents? Grosz and Sidner do not think so. They point out in [Grosz & Sidner 90] that (cooperative) collaboration among multiple agents may involve simultaneous, conjoined, or sequential execution of an action. Thus, it is necessary for every agent participating in the execution of an action to have a *mutual belief* about the intentions and abilities of one another and about the ways in which those intentions and actions support the achievement of the overall goal. The representation of the plan that contains this information is called a *SharedPlan*, and it consists of the following six entries:

1. the agents' mutual belief about the executability of the shared actions,
2. the agents' mutual belief about the way in which to perform the shared actions based on the relationships between the shared actions,
3. the agents' mutual belief that they intend to take a part in carrying out the shared actions,
4. the agents' mutual belief that their intentions for taking parts in the shared actions is to contribute toward the achievement of their SharedPlan itself.
5. intentions of the agents to do the shared actions,
6. intentions of the agents to do the shared actions to contribute toward the achievement of their SharedPlan itself.

It is possible that multiple agents may have agreed to work toward having a SharedPlan but have not achieved one yet. During that period, the agents are said to have a partial plan, called *SharedPlan**. A SharedPlan* may contain only partial collection of mutual beliefs and intentions. Also, some of the beliefs may be partially specified. In [Lochbaum *et al* 90], Lochbaum *et. al.* claims that the correct model of communication should include the representation of a SharedPlan* and the algorithm for augmentation of the SharedPlan. After all, the purpose of the communication is to exchange beliefs and intentions in order to reach a mutual ground (i.e. the SharedPlan). Lochbaum *et. al.* details the algorithm for augmenting a SharedPlan* that models interleaving of communication and execution. Their algorithm also models the situation where an agent either assents or dissents to the course of action suggested by other agents. However, it does not include the model of negotiation or argumentation which is necessary in the case of conflicting or non-coorperative situations.

5.3 Conclusion

The line of work on SharedPlan is the first of its kind in the field of natural language processing, intended to model the collaborative planning situation. The work is encouraging and motivating because we believe that the elements of collaboration are an intrinsic aspect of natural language dialogue, and thus we believe that a model of dialogue should embed a model of collaboration. As reviewed in this section, however, the model of collaboration should involve more than simple dissent when the plans of the other agents are disagreeable. Agents need to affect other agent's plans by argumentation in cases where they believe the other holds incorrect beliefs. Agents also use persuasion and negotiation in order to alter the intentions of others. A model that covers all this should also include text generation because negotiation and argumentation are more likely to be the task of language generation than that of understanding. To this date, however, there has been no work in text planning that addresses collaborative planning in AI.

6. Conclusion

In this paper, we have listed a variety of text planning tasks and attempted to match the techniques used in text planning tasks against those used in AI planning tasks by reviewing nine of the landmark natural language generation systems and two suggestions for better text planning ideas. Figure 6-1 shows a summary of the reviews compiled under three categories: under *overall tasks* is a description of the general tasks that the natural language generation systems are designed for; under *text planning tasks* is the definition/scope of the text planning tasks within these systems; and finally, under *planning techniques* is the summary of comparable AI planning techniques used in these systems.

The table shows, as has been pointed out several times in the paper, that the scope of the text planning task varies from one system to another. In particular, it is apparent from the table that the text planners can be divided into two groups — those that treat linguistic realization of content as a text planning task and those that do not. We discussed that the systems in the first group are either very inefficient (KAMP), only partially implemented (DIOGENES), or not very robust (PAULINE). On the other hand, we admitted that the reasons for using planning for linguistic realization are valid. Among other reasons, we assented to Hovy's argument that ''how to say it'' is part of ''what to say'', and thus the same reasoning that is used in selecting the content of a text should be used in selecting the tone and style of how the selected content is going to be represented. Curiously, none of the three systems use either reactive or increasingly reactive planning. We believe that using increasingly reactive planning to gradually ''automate'' planning for the surface realization tasks could be one way to compensate for the inefficiencies.

The AI planning techniques have been applied to limited and small domains. Text planning, on the other hand, is a much more complex task and requires more sophisticated planning solutions. Having reviewed various text planning systems, it seems to me that text planning researchers seek solutions from using hybrid planning techniques. Hovy, for example, uses a hybrid of prescriptive and restrictive planning in his PAULINE and a hybrid of hierarchical and opportunistic planning in his STRUCTURER. Suthers proposes a hybrid planning system that combines hierarchical, restrictive, and opportunistic planning. Also, Paris suggests a hybrid of reactive and skeleton planning which happens to bear a lot of resemblance with a new AI planning technique known as the increasingly reactive planning. It is hard to compare the pros and cons of one hybrid approach to another because

	TEXT	TAILOR	DISCOURSE PLANNER	KAMP	PAULINE	SPOKESMAN	STRUCTURER	DIOGENES	Suthers' Suggestion	EXPLANATION GEN	Paris' Suggestion
Overall Task											
a. generating paragraph length responses	+	+	.	.	+	.	.	.	.	+	+
b. generating single sentence responses	.	.	.	+	.	.	.	.	.	.	.
c. taking user models into account	.	+	.	+	+	.	.	.	.	+	+
d. extended interactive question/answering	.	.	.	.	.	.	.	.	.	+	+
e. producing slanted text	.	.	.	.	+	.	.	.	.	.	.
f. providing interface to data base system	+	.	.	.	.	.	.	.	.	.	.
h. providing interface to expert system	.	.	.	.	.	+	.	.	.	+	+
i. determining what to say	.	.	+	.	.	.	.	.	.	.	.
j. problem solving using illocutionary acts	.	.	.	+	.	.	.	.	.	.	.
k. structuring clause-sized chunks of input elements into coherent paragraph	.	.	.	.	.	.	+	.	+	.	.
l. machine translation	.	.	.	.	.	.	.	+	.	.	.
Text Planning Tasks											
a. *content selection*	+	+	+	+	+	.	.	+	.	+	+
b. *content delimitation*	+	+	.	.	+	+	+	+	+	+	+
c. *text structuring*	+	+	.	.	+	.	+	+	+	+	+
d. *content orchestration*	.	.	.	.	.	+	.	.	.	.	.
e. *resource choice*	.	.	.	.	.	+	.	.	.	.	.
f. *lexical selection*	.	.	.	+	+	.	.	+	.	.	.
g. *syntactic selection*	.	.	.	+	+	.	.	+	.	.	.
h. *coreference treatment*	.	.	.	.	+	.	.	+	.	.	.
i. *constituent ordering*	.	.	.	+	+	.	.	+	.	.	.
Planning Techniques											
a. *skeleton planning*	+	+	.	.	.	.	.	.	.	.	.
b. *hierarchical planning*	.	.	+	+	+	.	+	.	+	+	+
c. *least-commitment planning* or *restrictive planning*	.	.	.	.	+	+	.	.	+	.	.
d. *opportunistic planning*	.	.	.	.	.	.	+	+	+	.	.
e. *reactive planning*	.	.	.	.	.	.	.	.	.	+	.
f. *increasingly reactive planning*	.	.	.	.	.	.	.	.	.	.	+

Figure 6-1: Summary of the System Reviews

every approach is applied to a text planner with different goals and assumptions. However, this paper describes in detail why hybrid of AI planning techniques are useful in solving challenging tasks of text planning.

Although the focus of this paper has been on the use of AI planning techniques for solving text planning problems, many constructs developed for text planning could be further studied and applied to benefit the AI planning problems. For example, one could think of using schemas as building blocks for the library of schemas to replace less appealing scripts. Also, one might adopt the use of growth points from RST, and use it to represent other plan operators. Certainly, AI planning will benefit from the experiences accumulated from text planning researches. In particular, in the domain of planning where mental states and communication play a role, the variety of planning formalisms and techniques discussed in this paper should be useful.

Recently, the task of planning in AI has become more complex by expanding to include tasks similar to text planning like multi-agent communication and collaborative planning as summarized in section 5. However, both AI planning and text planning research have more work to do in order to build autonomous systems which can cooperate with human agents in a collaborative work environment. AI planning techniques should be tried on domains where the planned course of actions can be affected by communicative acts. Also, their model of the world should include the state of the mind of the agents; e.g. intentions, beliefs, and knowledge. Text planning research should be headed towards planning for content and style that affects the particular hearer(s) or situation. Also, a text planner should be adaptive to the feedback of the hearer(s) and the changes in the state of the world. Furthermore, a text planner should be able to plan communicative acts for the speaker who participates actively in communication and can lead the conversation when negotiation or argumentation is needed to convey the intentions or beliefs of the speaker. We believe that by keeping a close watch on the development of each other’s problems and solutions, the two areas of research will have much to learn from each other.

Acknowledgement

We = I and Kathy McKeown. I = myself.

Thank Kathy. Thank Diane and Steve.

References

[Agre & Chapman 87]
Agre, P. E. and Chapman, D.
Pengi: An Implementation of a theory of activity.
In *Proceedings AAAI-87*, pages pp.268-272. 1987.

[Allen and Perrault 80]
Allen, J. F. and Perrault, C. R.
Analyzing Intention in Utterances.
Artificial Intelligence 15(1):143-178, 1980.
also in Readings in Natural Language Processing in pp. 441-458.

[Appelt 85] Appelt, D. E.
Planning Natural Language Utterances.
Cambridge University Press, Cambridge, England, 1985.

[Austin 62] Austin, J.L.
How to do Things with Words.
Harvard University Press, Cambridge, Mass., 1962.

[Bach 86] Bach, E.
The Algebra of Events.
Linguistics and Philosophy 9:p.5-15, 1986.

[Barr *et. al.* 89] Barr, A., Cohen P. R., and Feigenbaum, E. A.
The Handbook of Artificial Intelligence.
Addison-Wesley, 1989.

[Brooks 86] Brooks, R. A.
A Robust Layered Control System for a Mobile Robot.
IEEE Journal of Robotics and Automation RA-2(1), 1986.
Also a tech report, A.I. Memo 864, M.I.T. A.I. Lab, 1985.

[Clancey 83] Clancey, W. J.
Overview of GUIDON.
Journal of Computer-Based Instruction 10(1-2):pp.8-15, 1983.

[Cohen & Levesque 85]
Cohen, P. R. and Levesque, H. J.
Speech Acts and Rationality.
In *Proceedings of the 23rd Conference of the ACL*, pages 49-59. Association for Computational Linguistics, Stanford, CA, 1985.

[Cohen & Perrault 79]
Cohen, P. R. and Perrault, C. R.
Elements of a Plan-Based Theory of Speech Acts.
Cognitive Science 3:177-212, 1979.
also in Readings in Natural Language Processing pp.423-440.

[Dale 88] Dale, Robert.
Generating Referring Expressions in a Domain of Objects and Processes.
PhD thesis, University of Edinburgh, November, 1988.

[Davis & Smith 83]
Davis, R. and Smith, R. G.
Negotiation as a Metaphor for Distributed Problem Solving.
Artificial Intelligence 20(1), 1983.

[Erman *et. al.* 80] Erman, L. D., Hayes-Roth, F., Lesser, V. R., and Reddy, R. D.
The Hearsay-II Speech-understanding system: Integrating Knowledge to Resolve Uncertainty.
ACM Computing Surveys 12(2), 1980.

[Feiner & McKeown 90]
Feiner, S. and McKeown, K. R.
Coordinating Text and Graphics in Explanation Generation.
In *Proceedings of the Eleventh AAAI Conference*, pages pp.442-449. American Association of Artificial Intelligence, 1990.

[Fikes & Nilsson 71]
Fikes, R. E. and Nilsson, N. J.
STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving.
Artificial Intelligence 2:pp.189-208, 1971.

[Georgeff 86] Georgeff, M. P.
Actions, Processes, and Causality.
In Georgeff, M. P. and Lansky, A. (editor), *Reasoning about Actions & Plans: Proceedings of the 1986 Workshop*, pages pp.99-122. AAAI and Center for the Study of Language and Information, Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1986.

[Georgeff 90] Georgeff, M. P.
Planning.
The Morgan Kaufmann Series in Representation and Reasoning. Readings in Planning.
Morgan Kaufmann, San Mateo, CA, 1990, pages pp.5-25, Chapter 1.

[Grice 57] Grice, H. P.
Meaning.
Philosophical Review 66(3):pages 377-388, 1957.

[Grosz 77] Grosz, B. J.
The Representation and Use of Focus in Dialogue Understanding.
Technical Report Technical note 151, Stanford Research Institute, 1977.

[Grosz & Sidner 90]
Grosz, B. and Sidner, C.
Plans for Discourse.
Intentions in Communication.
MIT Press, Cambridge, MA, 1990, pages pp.417-444, Chapter 20.

[Hayes-Roth 85] Hayes-Roth, B.
A Blackboard Architecture for Control.
Artificial Intelligence 26(3):pp.251-321, 1985.

[Hovy 87] Eduard Hovy.
Generating natural language under pragmatic constraints.
PhD thesis, Yale University, 1987.

[Hovy 88a] Hovy, E. H.
On the Study of Text Planning and Realization.
In *Proceedings of the AAAI Workshop on Text Planning and Realization* , pages pp.17-29. American Association of Artificial Intelligence, St Paul, MN, August, 1988.

[Hovy 88b] Hovy, E. H.
Generating Natural Language Under Pragmatic Constraints.
Lawrence Erlbaum Associates, Inc., Hillsdale, NJ, 1988.

[Hovy 88c] Hovy, E. H.
Two Types of Planning In Language Generation.
In *Proceedings of the 26th ACL Conference*, pages 179-186. ACL, Buffalo, June, 1988.

[Hovy 88d] Hovy, E. H.
Planning Coherent Multisentential Text.
In *Proceedings of the 26th ACL Conference*, pages 163-169. ACL, Buffalo, June, 1988.

[Hovy 91] Hovy, E. H.
Approaches to the Planning of Coherent Text.
Natural Language Generation in Artificial Intelligence and Computational Linguistics.
Kluwer Academic Publishers, 1991, pages pp.83-102.

[Hovy *et. al.* 88] Hovy, E. H., McDonald, D. D., and Young, S. R.
Current Issues in Natural Language Generation: An Overview of the AAAI Workshop on Text Planning and Realization.
In *Proceedings of the AAAI Workshop on Text Planning and Realization* , pages pp.8-14. American Association of Artificial Intelligence, St Paul, MN, August, 1988.

[Kabanza 90] Kabanza, F.
Synthesis of Reactive Plans for Multi-Path Environments.
In *Proceedings AAAI-90*, pages pp.164-169. 1990.

[Kaebling 87] Kaebling, L. P.
An Architecture for Intelligent Reactive Systems.
In Georgeff, M. and Lansky, A. L. (editor), *Reasoning About Actions and Plans*, pages pp.395-410. Morgan Kaufmann, Palo Alto, CA, 1987.

[Kay 79] Kay, M.
Functional Grammar.
In *Proceedings of the 5th meeting of the Berkeley Linguistics Society*. Berkeley Linguistics Society, 1979.

[Klein 65] Klein, S.
Automatic Paraphrasing in Essay Format.
Mechanical Translation 8:pp.68-83, 1965.

[Klein & Simmons 63]
Klein, S. and Simmons, R. F.
Syntactic Dependence and the Computer Generation of Coherent Discourse.
Mechanical Translation 7:pp.50-61, 1963.

[Kukich 83] Kukich, K.
Knowledge-based Report Generation: a Knowledge-engineering Approach.
PhD thesis, University of Pittsburgh, 1983.

[Lansky 86] Lansky, A. L.
A Representation of Parallel Activity Based on Events, Structure, and Causality.
In Georgeff, M. P. and Lansky, A. (editor), *Reasoning about Actions & Plans: Proceedings of the 1986 Workshop*, pages pp.123-160. AAAI and Center for the Study of Language and Information, Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1986.

[Lebowitz 85] Lebowitz, M.
Story-telling as Planning and Learning.
14():pp.483-502, 1985.

[Litman & Allen 87]
Litman, D. J. and Allen, J. F.
A Plan Recognition Model for Subdialogues in Conversations.
Cognitive Science 11:pp.163-200, 1987.

[Lochbaum *et al* 90]
Lochbaum, K. E., Grosz, B. J, and Sidner C. L.
Models of Plans to Support Communication: An Initial Report.
In *Proceedings of the Eleventh AAAI Conference*, pages pp.485-490. American Association of Artificial Intelligence, 1990.

[Mann & Thompson 87a]
Mann, W.C. and S.A. Thompson.
Rhetorical Structure Theory: A Theory of Text Organization.
Discourse Structure.
Ablex, Norwood, NJ, 1987.

[Mann & Thompson 87b]
Mann, W. C. and Thompson, S. A.
Rhetorical Structure Theory: Description and Construction of Text Structures.
NATO ASI Series. Natural Language Generation; New Results in Artificial Intelligence, Psychology and Linguistics.
Martinus Nijhoff Publishers, 1987, pages pp.85-95, Chapter 7.

[McCoy 88] McCoy, K. F.
Reasoning on a Highlighted User Model to Respond to Misconceptions.
Computational Linguistics 14(3):pp.23-37, 1988.

[McCoy & Cheng 91]
McCoy, K. F. and Cheng, J.
Focus of Attention: Constraining What Can Be Said Next.
Natural Language Generation in Artificial Intelligence and Computational Linguistics.
Kluwer Academic Publishers, 1991, pages pp.103-124.

[McDonald 84] McDonald, D.
Description Directed Control.
Computers and Mathematics 9(1), 1984.
Reprinted in Grosz, *et. al.* (eds.) *Readings in Natural Language Processing*, Morgan Kaufman Publishers, California, 1986, pp.519-538.

[McKeown 82] McKeown, K. R.
Generating Natural Language Text in Response to Questions About Database Structure.
PhD thesis, University of Pennsylvania, May, 1982.
Also a Technical report, No MS-CIS-82-05, University of Pennsylvania, 1982.

[Mckeown 85] Mckeown, K. R.
Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text.
Cambridge University Press, Cambridge, England, 1985.

[McKeown *et. al.* 90]
McKeown, K., Elhadad, M., Fukumoto, Y., Lim, J., Lombardi, C., Robin, J. and Smadja, F.
Language Generation in COMET.
In Dale, R, Mellish, C., and Zock, M. (editor), *Current Research in Language Generation*, pages 103-140. Academic Press, London, UK, 1990.

[Meteer 88] Meteer, Marie W.
Defining a Vocabulary for Text Planning.
In *Proceedings of the AAAI Workshop on Text Planning and REalization, AAAI-88*. St Paul, MN, 1988.

[Meteer 89] Meteer, Marie W.
The Spokesman Natural Language Generation System.
Technical Report 7090, BBN Systems and Technologies Corporation, Cambridge, MA, July, 1989.

[Meteer 90a] Meteer, M. W.
The ''Generation Gap'': The Problem of Expressibility in Text Planning .
PhD thesis, BBN, 1990.
BBN Report No. 7347.

[Meteer 90b] Meteer, M. W.
Abstract Linguistic Resources for Text Planning.
In *Proceedings of the 5th International Natural Language Generation Workshop*, pages pp.62-69. AAAI, Dawson, PA, 1990.

[Mitchell 90] Mitchell, T. M.
Becoming Increasingly Reactive.
In *Proceedings AAAI-90*, pages pp.1051-1059. 1990.

[Moore 89] Moore, J. D.
A Reactive Approach to Explanation in Expert and Advice-Giving Systems.
PhD thesis, University of California, Los Angeles, 1989.

[Moore & Paris 89]
Moore, J.D. and C.L. Paris.
Planning Text for Advisory Dialogues.
In *Proceeding 27th ACL*, pages 203-211. ACL, Vancouver, BC, June, 1989.

[Moore & Swartout 88]
Moore, J. and Swartout, W. R.
Planning and Reacting.
In *Proceedings of the AAAI Workshop on Text Planning and Realization, AAAI-88*. St Paul, MN, 1988.

[Moore & Swartout 89]
Moore, J. D. and Swartout W. R.
Explanation in Expert Systems: A Survey.
Submitted to Artificial Intelligence Journal.
January, 1989

[Moore & Swartout 91]
Moore, J. D. and Swartout W. R.
A Reactive Approach to Explanation: Taking the User's Feedback into Account.
Natural Language Generation in Artificial Intelligence and Computational Linguistics.
Kluwer Academic Publisher, 1991.

[Newell & Simon 63]
Newell, A. and Simon, H. A.
GPS, a Program that Simulates Human Thought.
Computers and Thought.
McGraw-Hill, New York, NY, 1963, pages pp. 279-293.

[Nirenburg *et. al.* 89]
Nirenburg, S., Nyberg, E. H., 3rd, and Defrise C.
Text Planning with Opportunistic Control.
Technical Report CMU-CMT-89-113, CMU, July, 1989.

[Paris 87] Paris, C. L.
The Use of Explicit User Models in Text Generation: Tailoring to a User's Level of Expertise.
PhD thesis, Columbia University, 1987.

[Paris 91] Paris, C.
Generation and Explanation: Building an Explanation Facility for the Explainable Expert Systems Framework.
Natural Language Generation in Artificial Intelligence and Computational Linguistics.
Kluwer Academic Publishers, 1991, pages pp.48-82.

[Pednault 87] Pednault, E.
Formulating Multiagent, Dynamic-world Problems in the Classical Planning Framework.
In Georgeff, M. P. and Lansky, A. (editor), *Reasonning about Actions and Plans, Proceedings of the 1986 Workshop*, pages pp.47-82. Morgan Kaufmann, Timberline, OR, 1987.

[Pollack 86a] Pollack, M. E.
Inferring Domain Plans in Question Answering.
PhD thesis, University of Pennsylvania, 1986.
Also a Tech. Rep. No. MSCIS-86-40.

[Pollack 86b] Pollack, M. E.
A Model of Plan Inference That Distinguishes Between the Beliefs of Actors and Observers.
In Georgeff, M. P. and Lansky, A. (editor), *Reasoning about Actions & Plans: Proceedings of the 1986 Workshop*, pages pp.279-295. AAAI and Center for the Study of Language and Information, Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1986.

[Pollack 91] Pollack, M. E.
The Uses of Plans.
Script of presentation in *Computers and Thought Lecture* in *IJCAI-91*.
August, 1991

[Rich & Night 91] Shapiro, D. M and Murphy, J. F. (editor).
Artificial Intelligence.
McGraw-Hill, Inc., 1991.

[Rosner 87] Rosner, D.
The Automated News Agency SEMTEX — a Text Generator for German.
NATO ASI Series. Natural Language Generation; New Results in Artificial Intelligence, Psychology and Linguistics.
Martinus Nijhoff Publishers, 1987, pages pp.133-148.

[Sacerdoti 74] Sacerdoti, E. D.
Planning in a Hierarchy of Abstraction Spaces.
Artificial Intelligence 5:pp.115-135, 1974.

[Sacerdoti 77] Sacerdoti, E. D.
A Structure for Plans and Behavior.
Elsevier, New York, NY, 1977.

[Safavi 91] Safavi, A.
Planning and Scheduling of Software Manufacturing Projects.
PhD thesis, The Robotics Institute, Carnegie Mellon University, April, 1991.

[Searle 69] Searle, J.
Speech Acts: An Essay in the Philosophy of Language.
Cambridge University Press, Cambridge, England, 1969.

[Stefik 80] Stefik, M. J.
Planning with Constraints.
PhD thesis, Stanford University, 1980.
Rep. No. 80-784.

[Stefik 81] Stefik, Mark.
Planning with Constraints (MOLGEN: Part 1).
Artificial Intelligence 16:p.111-140, 1981.

[Sussman 75] Sussman, G. J.
A Computer Model of Skill Acquisition.
American Elsevier, New York, NY, 1975.

[Suthers 90] Suthers, D.
Reassessing Rhetorical Abstractions and Planning Mechanisms.
In *Proceedings of the 5th International Natural Language Generation Workshop*, pages pp.137-143. AAAI, Dawson, PA, 1990.

[Swartout 73] Swartout, W. R.
XPLAIN: A System for Creating and Explaining Expert Consulting Systems.
Artificial Intelligence Journal 21(3):pp.285-325, 1973.

[Sycara 88] Sycara, K. P.
Resloving Goal Conflicts via Negotiation.
In *Proceedings of the 7th National Conference on Artificial Intelligence*, pages pp.245-250. AAAI, St. Paul, MN, August, 1988.

[Sycara 89] Sycara, K. P.
Argumentation: Planning Other Agents' Plans.
In *Proceedings of the 11th International Conference on Artificial Intelligence*, pages pp.517-523. IJCAI, Detroit, MI, August, 1989.

[Tate 75] Tate, A.
Interacting Goals and Their Use.
In *Proceedings of the 4th IJCAI*, pages pp.215-218. International Joint Conferences on Artificial Intelligence, 1975.

[Tate *et. al.* 90] Tate, A., Hendler, J., and Drummond, M.
A Review of AI Planning Techniques.
The Morgan Kaufmann Series in Representation and Reasoning. Readings in Planning.
Morgan Kaufmann, San Mateo, CA, 1990, pages pp.26-49, Chapter 1.

[Wilks 73] Wilks, Y. A.
An Artificial Intelligence Approach to Machine Translation.
Computer Models of Thought and Language.
Freeman, San Francisco, CA, 1973, pages pp.114-151.

[Williams 87] Williams, R. N.
Planning for Public Speaking: An Outline Guide.
Textbook distributed in Publisc Speech course taught by the author at Columbia University.
1987

[Wolz 90] Wolz, U.
An Object Oriented Approach to Content Planning for Text Generation.
In *Proceedings of the 5th International Natural Language Generation Workshop*, pages pp.95-104. AAAI, Dawson, PA, 1990.

[Woolf 84] Woolf, B. P.
Context Dependent Planning in a Machine Tutor.
PhD thesis, University of Massachusetts, 1984.

[Yngve 62] Yngve, V.
Random Generation of English Sentences.
In *1961 International Conference on Machine Translation of Languages and Applied Language Analysis*, pages pp.66-80. Her Majesty's Stationaery Office, London, 1962.
(National Physiical Laboratory, Symposium No.13).

[Zlotkin & Rosenschein 88]
Zlotkin, G. and Rosenschein, J. S.
Negotiation and Conflict Resolution in Non-Cooperative Domains.
Automated Reasoning.
Prentice-Hall, Englewood Cliffs, NJ, 1988, pages pp.100-105.

Table of Contents

List of Figures