AUTOMATICALLY GENERATING NATURAL LANGUAGE REPORTS IN AN OFFICE ENVIRONMENT

Jugal Kalita and Sunil Shende
Department of Computer and Information Sciences
University of Pennsylvania, Philadelphia, PA 19104

ABSTRACT

In this paper, we describe a system which models a set of concurrent processes that are encountered in a typical office environment, using a body of explicitly sequenced production rules. The system employs an interval-based temporal network for storing historical information. A text planning module traverses this network to search for events which need to be mentioned in a coherent report describing the current status of the system. In addition, the planner also combines similar information for succinct presentation whenever applicable. Finally, we elaborate on how we adapt an available generation module to produce well-structured textual report for our chosen domain.

1. INTRODUCTION

This paper describes the implementation of a text generating system which produces natural language reports on the status of a system of inter-related processes at various stages of progress. The motivations behind this project were to model a system of concurrent processes and to successfully generate well-formed text about their temporal behavior. We view a process as an ordered sequence of events over time where an event refers to an *atomic* action by one of the participating agents. In many AI applications, monitoring the state of a system of processes is deemed essential. The ability of a system in such an environment to describe its actions in natural language will be certainly very useful.

As our sample domain, we chose a scenario where the system is used to assist the secretary for an academic journal by keeping track of a paper submitted for publication. The process being modeled is that of paper-submission with the usual participating agents being the author of the paper, the journal editor, the reviewers assigned to evaluate the paper and the journal secretary.

In the rest of the paper we present methods to represent the knowledge of the chosen domain, to model historical knowledge base of events and processes, to appropriately order the contents of the historical information store for presentation based on temporal relationships and other relevant factors, and to produce coherent English text describing the events in the domain. The system as described herein has been fully implemented, and currently avenues for further improvement are actively being pursued.

Section 2 of this paper gives a brief description of the domain being modeled along with an overview of the system components. Section 3 describes the representation of the processes in terms of a network of rules. Section 4 elaborates the nature of the historical information store. Section 5 deals with text planning which is crucial for selecting information to be presented in a coherent manner. Section 6 details the actual process of text generation. Finally, we conclude with a discussion about enhancements to the existing system which can contribute towards a more general implementation with the added power of expected future inference and and the ability to reason about viewpoints of agents external to the system. Further details about the system's design and implementation, and other relevant discussions can be found in [Kalita 86].

2. OVERVIEW OF THE SYSTEM

The system consists of three main modules as shown in figure 1. They are

- an augmented production system for domain knowledge representation,
- a text planner module (strategic component),
- a text generation module (tactical component).

The system of processes is modeled in terms of a production system with an explicit control struc-

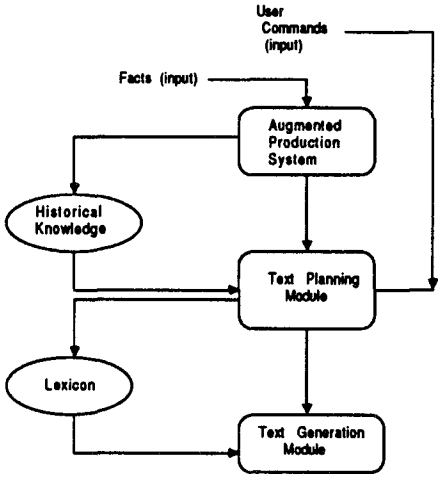

Figure 1: Overview of the system

ture for rule sequencing; this representation is due to [Zisman 78]. The execution of these rules builds a history network of events and processes. This network is modeled loosely on the notion of intervals proposed in [Allen 83,Allen 84].

The creation of a report in natural language entails that the actual description be easy to understand. The report must omit inessential information, be cogent, and give the user a sense of temporal structure. Towards this end, planning at the representational level is required of the system. Processes can be looked at as being composed of sub-processes; clearly the description of the history must be able to account for this compositional structure.

The text generator used in this project is MUMBLE [McDonald 83], [McDonald 85], which is a collection of morphology and grammar modules driven by user created objects called *realization specifications* containing information about the high-level specifics of the text.

3. REPRESENTATION OF DOMAIN KNOWLEDGE

The domain described above has been modeled using a production system whose rules are sequenced by imposing a Petri net structure on them. The decision to employ a production system has been influenced by the event-driven nature of the processes in the domain allowing declarative encoding of relevant knowledge.

In the domain under consideration, there are several concurrent, independent processes. All activities in the domain can be viewed as constituting a single process — the overall process of *journal editing*. The overall process comprises a collection of time-ordered atomic actions and/or subprocesses. In addition to several atomic actions, the journal editing process contains three subprocesses: *referee review processes* — one for each referee designated by the editor. In turn, each referee review process contains a number of atomic actions. Communication and coordination is required among these processes. Additionally, communication is also necessary among the atomic actions constituting these processes for achieving proper temporal ordering.

In our implementation, the system of actions and processes is represented as a set of production rules. Since in a *pure* production system, it is difficult to achieve substantial interrule communication, a more acceptable control structure can be devised where the current state of the system in conjunction with the history of prior rule execution determines what needs to be done next. Such a control structure, first proposed in the context of an augmented Petri net model in [Zisman 78], has been adopted in our system. In our implementation, a state in the Petri net represents a state of the journal secretary (since our representation is from the viewpoint of the secretary). A transition between two states denotes an interaction between an external agent (such as the author, the editor or one of the referees) and the secretary. Each interaction is represented by a production rule which resides on a transition in the augmented Petri net.

The Petri net structure of the rules is implemented by maintaining a list of *active rules* as suggested in [Zisman 78]. States in the Petri net structure are not explicitly represented in the implementation. The set of active rules (in other words, the set of enabled transitions) at a given point in time determines the current state of the system. When a transition is enabled, its firing is determined by the rule *residing* at the transition. All rules at all enabled transitions constitute the *active rule set*. In summary, the production system models individual actions, whereas the execution of the Petri net models the dynamic relationships among the various processes.

In the augmented Petri net representation of our domain, the system (representing the editor's secretary) can exist in several states such as,

- waiting for the paper to arrive;
- waiting for the editor to designate referees;
- waiting for a referee to respond;
- waiting for a referee to submit his report;
- waiting for the editor to make a a decision.

Several interactions can take place between the secretary and the various external agents. These are illustrated in figures 2 and 3 which show the augmented Petri net representation of our domain from the point of view of the secretary. There are two networks —

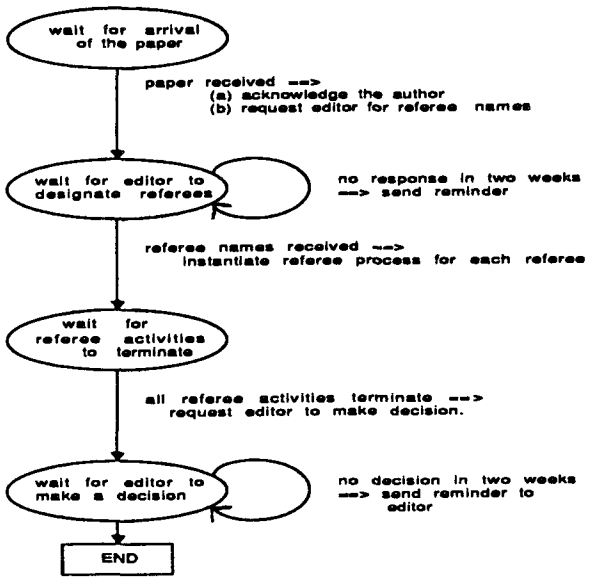

Figure 2: Petri net corresponding to the journal editing process

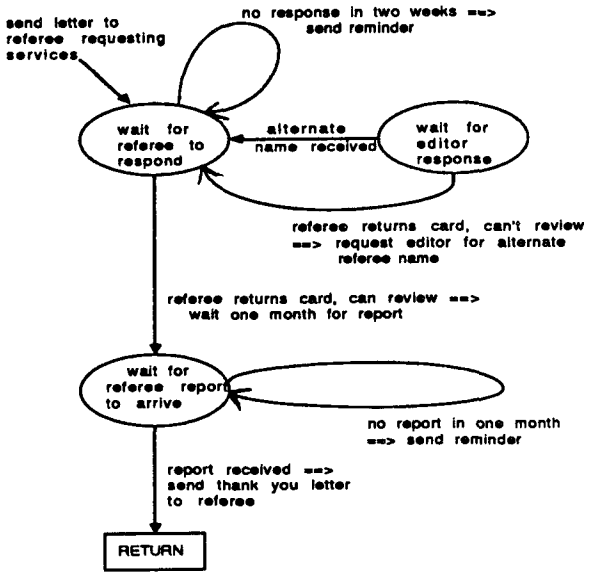

Figure 3: Petri net corresponding to a referee review process

---

- **Rule-1**: If a paper is received,
  - acknowledge the author of its receipt, and
  - request the editor to designate names of referees.
- **Rule-5**: If the author withdraws the paper, instantiate termination procedure.
- **Rule-6**: If the editor does not respond within two weeks, send him a reminder letter.

Figure 4: Examples of production rules

one corresponding to the overall journal editing process and the other corresponding to a generic referee review process. Some of the rules in the production system are shown in figure 4.

4. THE NETWORK OF INTERVALS: HISTORICAL KNOWLEDGE

The production system described in the previous section models the domain knowledge. In order to generate a report about a particular paper, the production system must build an historical record of event instances as they take place over time. The representation must adequately capture temporal relationships among events, and also minimize the amount of computation necessary in order to achieve reasonable levels of efficiency.

Most generally accepted approaches to modeling time such as the state space approach [Fikes 71] and situation calculus [McCarthy 69] are inadequate for the representation of concurrency, and are based on the notion of time points with accompanying semantic difficulties. [Allen 83,Allen 84] propose a model of time based on *intervals*. The model allows for representing the relationships between temporal intervals in a hierarchical manner using a constraint propagation technique. It also facilitates structuring knowledge in a way convenient for typical reasoning tasks, and also is intuitively more appealing due to the observation that notwithstanding the instantaneous appearance of many real-world events, they can be decomposed recursively for closer examination, if desired.

Intervals in our system are of two types — *simple intervals* (designated henceforth as intervals) and *reference intervals*. Intuitively, an interval corresponds to *the time between two successive interactions* between agents in the system. In contrast, a reference interval corresponds to the time during which a whole series of interactions take place (i.e the temporal duration of a process or sub-process). Reference intervals, based on the suggestion in [Allen 83], allow us to conveniently group together clusters of simple intervals. Each interval is identified further by a description of some of the events which occurred during the two interactions whereas each reference interval is identified by the intervals which comprise it.

An interval (or reference interval) is said to be instantiated if it becomes part of the history network. In particular, every instantiated interval has slots which contain its starting event, its ending event and a link to its predecessor interval in the history network. In addition to the starting and ending events, an interval may also contain *side* events that neither start nor end the interval but which occur in the domain process as a matter of course between the starting and ending events and do not have any further consequences.

An interval is, designated *open* if it has been started by some event but which is yet to be completed by another event. A completed interval is called a *closed* interval. Since the representation is from the journal secretary's point of view, it can be seen that interactions are almost always initiated by agents other than the secretary but are terminated by some action taken by the secretary. Thus, in our representation, an instantiated open interval has as its starting event, an action performed by the secretary, whereas input from the external agents determines the ending event of some hitherto open interval.

In effect, the relationships among intervals are maintained in the network where nodes represent individual intervals, and each arc's label indicates the possible relationships between the two intervals represented by its end nodes. Since the network built by our production system is historic in nature, there is no temporal uncertainty, and therefore, each arc has only one label which indicates strict temporal precedence among the intervals.

Although our system is basically input-driven at the user-system interface, there are many situations where the script of the domain process demands some temporal deadlines or monitors. For instance, in the event that the editor fails to respond (by way of input) to the request for names of reviewers, the system has to generate, perforce, another request as a reminder. To enable the system to faithfully retrieve such timeout activity from the network, we employ *demons* to explicitly monitor open intervals pending response. On timeout, the system simply creates another open interval with the same starting event as the previous one timed out by its associated demon.

It is often the case that processes (sub-processes) are generic in nature; i.e. they can be *repeatedly instantiated* for different agents. For example, our domain calls for a certain sequence of interactions between the secretary and a referee. This sequence defines a generic process and our system needs to keep track of three such processes, one for each referee assigned to review a paper. The second type of interval in our system, the reference interval which corresponds to processes and sub-processes, help the text planner combine multiple

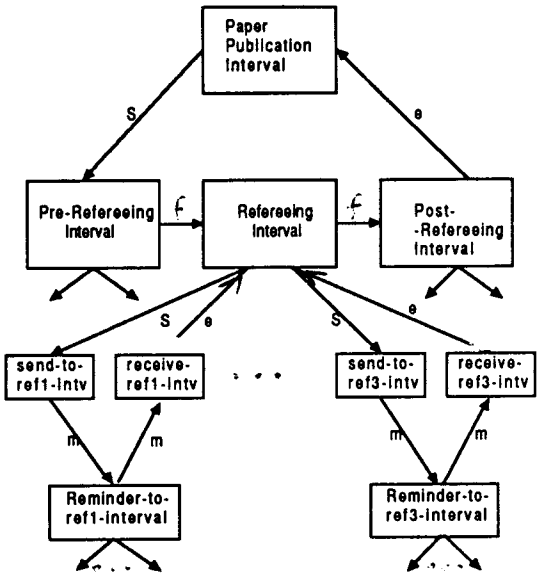

Figure 5: The network configuration (partial)

instances of *events* into succinct specifications without having to repeat descriptions for each individual instance. It is also desirable in an application like ours to annotate the text describing a process with *summaries* or paraphrases of key sub-processes that occur naturally in the domain. Since it is clear that an open reference interval captures our intuitions about an *ongoing process*, the planner can generate appropriate temporal cues from the interval representation to determine text parameters like tense and aspect for a papraphrase.

An instance of the network configuration built up by the production system is shown in figure 5. However, not all details are shown in the figure.

5. TEXT PLANNING

Our computational goal, as stated earlier, was to describe or report on a set of processes in a compact manner, without disturbing the temporal nature of the process description. The text planning module thus needs to ensure that the text plan is ordered in time and that multiple event instances in processes be encapsulated, if possible, into concise descriptions.

The former requirement entails that the events we set out to describe be ordered in time. From the description of the system network in the previous section, we notice that most of the sequencing is avoided by simply working one's way backward along the predecessor links in an interval data structure. However, the conciseness requirement forces the planner, at

some points, to combine information which is common to more than one interaction sequence (sub-process) without ruining the nature of the then current plan. Our representation of reference intervals associated with ordinary intervals gives the extra information necessary to guide the planner in such cases.

Briefly, the planner goes through three phases: it determines the content of the overall plan by way of a controlled search over the network of intervals and then produces a sequence of specifications for the actual text generator in the order in which they are to be realized into text along with information regarding discourse focus. Since our intention is to provide as informed a description (report) of the system's activity as possible, we have chosen to restrict the content of a report to the interactions occurring from the time the *previous* request was issued to the time of the current request. In this respect, the interval data structure provides us with some useful information pertaining to what happened *just before* the previous report was requested and what is *expected* to occur *just after* the current one. This information is contained in the open intervals at the time of the previous request, and at the present moment respectively. The current version of the system does not utilize the expectation information contained in the intervals - the report could be extended appropriately to reflect it in future. In fact, the same information could be used to detect and report the occurrence of *unexpected* events occurring in the system.

The first phase, then, combines a search over the required part of the interval network starting from the intervals that are currently *open* with an ordering over the events which occurred in the intervals in the search space. Since reports can be generated at any pont in time during the life of a paper, and one can ask for repeated reports at later times, *only events that have not been reported in an earlier report* need to be included in it. However, to ensure that the current report establishes a relationship with the immediately preceding report, one or more events that have been reported earlier may be included. This will be clear from the example presented later. The ordering ranks events tagged with their associated intervals (called "situations") temporally, while simultaneously ensuring that *similar* events corresponding to different instances of the same generic process are clubbed together. It should be noted that during this phase when similar events with one or more different participants are combined together for the purposes of generation, the system's decision *need not be totally based on temporal sequentiality.* Combinations may be performed for similar events although they occurred widely separated on a linear time scale. For example, suppose a referee sends his review quite early (say, at a time t_1), whereas a second referee does so with considerable amount of delay (at a time t_2), and several other things happen during the interval (t_1, t_2). While reporting, the system will combine the two reviewing events into a single sentence such as *The two referees A and B sent their reviews by date* t_1, and report the events that took place in the interval (t_1, t_2) either before this sentence or after it depending on other considerations.

The second phase processes this stream of ranked situations to create the appropriate specifications for the text generator. This includes information about the sentence content (derived directly from the event in the situation) along with various features of the sentence like tense and indications to conjoin noun phrases etc. For the present, our tense assignment is rather naive and only relies on the network to provide temporal indicators based on the relationship of the time at which an event occurred in the context of the overall time interval covered by the plan. Thus we only assign present, present perfect, past and past perfect tenses to sentences. A more sophisticated tense assignment, perhaps along the lines of [Matthiesen 84], would be necessary for future extensions of this project.

Thus, during the first two phases, temporal relationships and event similarities are used to order the events into an appropriate sequence for presentation. However, in order to determine how exactly each sentence is going to be constructed, during the third phase, the planner annotates the text specifications with information regarding *discourse focus.* Generating the actual text involves several decisions such as choice of sentence structure, and sentential subject, etc. Assuming, each sentence is of the *svo*[1] form, focus considerations can be used to choose between active and passive voices, and to decide which case slot's (e.g., direct-object, indirect-object, source, destination, etc.) filler to be used to fill the position of the sentential subject.

Choice of subjects for generated sentences can be done considering the movement of focus of discourse from sentence to sentence as in the TEXT system [McKeown 85] following the analysis of Sidner [Sidner 83]. Focus information also facilitates pronominalization decisions. There are two types of foci: *global* and *immediate.* Global focus is constant, but the immediate focus may vary from sentence to sentence. It is updated using techniques similar to those used by McKeown. To track immediate focus, we keep track of three variables as done by McKeown: *CF* — current focus, *PFL* — potential focus list, and *FS* — past immediate focus or focus stack. CF is that focus of the current sentence whose initial value is the global focus. PFL consists of the items referred to in the current sentence. FS is updated each time the focus changes. When the text changes to a member of

[1] That is, it has a subject, a verb and an object phrase with optional following prepositional phrases

the PFL, the old focus is pushed onto the stack. When conversation returns to an item previously discussed, the stack is popped to yield that item. The focus rules used are as follows (in order): shift focus to a member of previous PFL, i.e. CF(new sentence) $\in$ PFL(last sentence), maintain the same focus, i.e., CF(new sentence) = CF(last sentence), and return to a previous focus, i.e., CF(new sentence) $\in$ FS.

In summary, the text planner traverses, orders and translates the intervals desired to be reported into a stream of specifications which can be realized directly by the text generator.

6. THE TACTICAL COMPONENT

The tactical component used by the system to generate actual text is MUMBLE [McDonald 83,McDonald 85]. The decision to employ MUMBLE was primarily influenced by its success in generating a large variety of grammatical constructs in several text generation applications as reported in [Karlin 85],[McDonald 83],[McDonald 85],[Rubinoff 86], the flexibility in the design of its input as well as its ready availability. Its design is based on the assumption that division between planning the content of speech and its verbalization is unambiguously defined. This partitioning of the generative decisions into levels marks one aspect of difference between MUMBLE and Appelt's generation system [Appelt 83] which emphasizes the homogeneity of various decisional procedures.[2]

The input to MUMBLE consists of realization specifications (r-specs) representing the system's communicative goal, and produced by the planner (or the strategic component). Given the input r-spec, MUMBLE assigns (or, attaches) it to an appropriate position in an incipient surface structure tree. A depth-first traversal of the tree accompanied by recursive phrasal realization of unprocessed embedded elements results in the production of well-defined English text.

It must be stressed at this point that the tactical component plays a strong supporting role to the strategic component in the pursuit of coherence in the generated text. One method for enhancing coherence is by appropriate lexical substitution for referring expressions allowing previously generated sentences to exert influence on the realization of the current sentence. The system keeps track of the objects which have been referred to so far and how this reference has been made. This along with focus information enables the system to refer to objects by incomplete descriptions and to introduce pronominalization. This avoids unnecessary repetition leading to succinctness in the text. More importantly, it enables the hearer to distinguish new information from old so that comprehension is not hampered. Otherwise, it may lead to misunderstanding on the part of the hearer; examples of this phenomenon can be seen in [Granville 84].

Incomplete description of referring noun phrases in our system include usage of phrases such as "the author" or "the paper" in subsequent references instead of the complete phrases "the author D.D. McDonald" or "the paper entitled 'Generation Using MUMBLE'" which are used for introduction. Additionally, when a person introduced earlier by name is referred to subsequently at a point where pronominalization is deemed inappropriate, the person is referred to by his/her name. Similarly a group of people may be introduced by a phrase such as "the three referees, viz., B.L. Webber, A.K. Joshi and T. Finin". Subsequently, the first time one of these persons is referred to alone, we refer to him/her by name. If the same person is referred to again, only then pronominalization is resorted to provided it does not lead to ambiguity. Incomplete description also enables us to use phrases such as "on the same day", "yesterday", and "today" instead of always producing the complete phrase such as "on January 18, 1986" at all times. Also note that, the first time the system specifies a date, it specifies the year. However, in subsequent specifications, the year is not mentioned unless it is different from that of the immediately preceding date mentioned.

Adapting MUMBLE to our system required substantial additional extensions for handling of cardinal numbers, proper names, various tenses, etc. A number of new structures had to be added to support the desired features in the final text. Some of these are simple, others complex. Some are general while others are domain specific. At this point, we feel that it is pertinent to corroborate the observations in [Karlin 85] that it is difficult to create new structures that capture language generalizations due to the total absence of constraints on their nature.

Finally, we conclude this section by presenting examples of text produced by the system (assuming that the secretary is producing the report). The whole system has been designed such that we can perform simulation of events such as arrival of the paper, arrival of the reviews by the secretary. At any point during this simulation, the system may be asked to generate a report. Below, we reproduce two such reports — the first one was produced at an arbitray point during the life of the paper. The second one was produced for the same paper after the paper was processed completely.

[2] It is relevant to note at this point that Hovy, in his proposed approach to generation [Hovy 85], assumes a well-defined boundary between planning and actual generation, but allows the generational component to seek advice from the planner at limited decision points.

I received the paper entitled "Generation Using MUMBLE" from the author D.M. McDonald on January 1, 1986. He was informed of the receipt of the paper on the same day. I requested the names of three referees for the paper from the editor A.K. Joshi on the same day. He sent the names, viz., T. Finin, N. Badler and B. Webber a week later. I have sent a postcard inquiring availability to review the paper to each of them today.

Assuming we continue the simulation performed to the end and ask the system to produce a report again, the system generates the following text.

I had sent a postcard inquiring availability to review the paper entitled "Generation Using MUMBLE" to each of the referees, T. Finin, N. Badler, and B. Webber on January 15, 1986. Each sent a letter expressing ability to review the paper to me by January 30. I sent a copy of the paper to each of them. Each sent a review of the paper to me by February 15. I requested the editor to make a decision regarding publication of the paper on the same day. He sent a positive decision regarding publication of the paper to me yesterday. I informed the author of the decision yesterday.

7. CONCLUSIONS

Our goal here was to generate a natural language text report on a set of concurrent processes. A full-fledged activity report should comprise three distinct segments. The preamble of the report should contain a succinct description of events which had occurred prior to the point in time when the last report was generated (in cases where multiple reports are requested). This should be followed by the main body of the report stating the events which have occurred between the time of the last report and the current time. Finally, there should be a brief mention of events which are expected to occur in the immediate future.

Although attempts have been made in the current project to provide information about the past, the approach taken is simplistic. Improving this section of the report will involve reasoning about saliency of events in order to select events from past history for reporting. In addition, investigation regarding the formulation of the actual text for summarization must also be carried out.

We believe that representation of status should allow the system, in principle, to draw inferences about expected future events. Our implementation does not address the issue of expectation, but can be a starting point toward the goal of including inferred knowledge about the immediate future with the description of the system's history.

Another direction in which we intend to pursue further research is regarding presentation of unexpected events. This will necessitate incorporation of the ability to make inferences about what is expected and what is not. Additionally, appropriate text to report such events has to generated.

Currently, in our system, no explicit *textual links* are established between the contents of a sentence and that of its predecessor(s). This seems satisfactory due to the general characteristics of narrative text where temporal succession as well as simultaneity is indicated implicitly by simple sequencing of sentences. However, further improvement in the naturalness of the text can be achieved by inclusion of appropriate clue words which function as explicit inter-sentential links. The PROTEUS sentence planner in [Ritchie 84] may be able to provide helpful insights in this attempt. Furthermore, the quality of text produced can be improved substantially by incorporating a selection of commonly used temporal expressions in English. A rich compendium of such expressions is available in [Smith 78].

Finally, we want to investigate the issues involved in generating reports from the points of view of various participants. This will involve selecting events relevant to the person for whom the report is being prepared, and generating text in appropriate English. Choice of events will be dictated by various factors such as direct or indirect involvement of the reportee, his/her goals and responsibilities, and limitations regarding what he/she is allowed to know. Appropriate textual generation, i.e., choices of voice, sentence and clause structures, subjects and objects for the sentences as well as those of words will be governed by the nature of the listener's involvement with the events being described among other factors to be investigated. This will involve making a distinction between *real events* as stored in the historic knowledge base and *virtual events* characterizing how real events are reported from various perspectives.

ACKNOWLEDGEMENTS

The second author was partially supported by NSF grants MCS-8219116-CER, MCS-82-07294 and DCR-

84-10413 during the course of his work on this project.

We would like to thank Dr. Bonnie Webber for her painstaking reviewing of several drafts of the technical report on which this paper is based. We also thank Robin Karlin and Michael Niv for going through earlier drafts of this paper and making useful comments.

References

[Appelt 83] Appelt, D.E., Planning Natural Language Utterances, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, 1983, pp. 59-62.

[Allen 83] Allen, J.E., Maintaining Knowledge about Temporal Intervals, *Communications of ACM*, Volume 26, November 1983, pp. 832-843.

[Allen 84] Allen, J.E., Towards a General Theory of Action and Time, *Artificial Intelligence*, Volume 23, 1984, pp. 123-154.

[Fikes 71] Fikes, R.E., and Nilsson, N.J., STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving, *Artificial Intelligence*, Volume 2, 1971, pp. 189-205.

[Granville 84] Granville, R., Controlling Lexical Substitution in Text Generation, *Proceedings of the Annual Conference on Computational Linguistics*, 1984, pp. 381-384.

[Hovy 85] Hovy, E.H., Integrating Text Planning and production in Generation, *Proceeding of the Ninth International Joint Conference in Artificial Intelligence*, Volume 2, 1985, pp. 848-851.

[Karlin 85] Karlin, R.F., *ROMPER Mumbles*, Technical Report MS-CIS-85-41, Department of Computer and Information Sciences, University of Pennsylvania, Philadelphia, 1985.

[Kalita 86] Kalita, J.K., and S. Shende, *Generation of Natural Language Text Describing a System of Asynchronous, Concurrent Processes*, Technical Report MS-CIS-86-66, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, 1986.

[Matthiesen 84] Matthiesen, C., *Choosing Tense in English* , ISI Research Report ISI/RR-84-143, University of Southern California, November, 1984.

[McCarthy 69] McCarthy, J., and Hayes, P.J., Some Philosophical Problems from the Standpoint of Artificial Intelligence, *Machine Intelligence*, Volume 4, Edinburgh University Press, 1969.

[McDonald 83] McDonald, D.D., Description Directed Control: Its Implications for Natural Language Generation, *Computer and Mathematics with Applications*, Volume 9, No. 1, 1983, pp. 111-129.

[McDonald 85] McDonald D.D., and Pustejovsky, J.D., Description Directed Natural Language Generation, *Proceedings of Ninth International Joint Conference on Artificial Intelligence*, Volume 2, 1985, pp. 799-805.

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

[Ritchie 84] Ritchie, G., A Rational Reconstruction of the PROTEUS Sentence Planner, *Proceedings of the Annual Conference on Computational Linguistics*, 1984, pp.327-329.

[Rubinoff 86] Rubinoff, R., Adapting Mumble: Experience with Natural Language Generation, Proceedings of AAAI, 1986.

[Sidner 83] Sidner, C., Focusing in the Comprehension of Definite Anaphora, in *Computational Models of Discourse*, ed. Brady, M., MIT Press, 1983, pp. 209-265.

[Smith 78] Smith, C.S., The Syntax and Interpretation of Temporal Expressions in English, *Linguistics and Philosophy*, Volume 2, No. 1, 1978, pp. 43-100.

[Zisman 78] Zisman, M.D., Use of Production Systems for Modeling Asynchronous, Concurrent Processes, in *Pattern-Directed Inference Systems*, ed. Waterman, D.A., and Hayes-Roth, F., Academic Press, 1978, pp. 53-68.