Time
and
Natural Language Generation

David R. Forster

COINS Technical Report 89–01

Abstract

Until now, the problem of generating natural language texts which contain temporal expressions has been largely ignored, with treatments extending at best only to tense. In particular, temporal adverbials have not been incorporated, even though they represent the second most common way of expressing temporal relations in natural language. This work proposes a treatment of temporal adverbials, concentrating on prepositional phrases, which integrates well with schemes for generating tense, and is based on four assertions: 1. Planning temporal prepositional phrases requires reference to common-sense knowledge. 2. Generation of natural language is best achieved through the use of hierarchical constraint propagation. 3. Lexical selection of a single item requires the use of multiple sources of constraints. 4. In contradistinction to other proposals, it is asserted that tense and multiple temporal adverbials describing the same event are derived from a common data source.

This treatment will not only ensure the principled generation of correct temporal prepositional phrases, but will enable the simultaneous specification of tense and multiple temporal adverbials. In addition, it will establish the origins of different temporal expressions in relation to client data, define the correspondence of temporal data to their expression in language terms, and define the duties of the semantic component of a natural language generator and the duties of the program requesting the generation (the client). Furthermore, the generation of temporal expressions will serve as an additional test for current temporal representation schemes, one which is expected to indicate inadequacies in them and point the way to better mechanisms.

Contents

List of Figures

List of Tables

1 Introduction

In a world in which computer programs tackle increasingly complex tasks, it is becoming important for them to express results in more sophisticated ways. Simple output statements, graphics, and tables are all too inflexible to satisfactorily express complex relationships, to effectively reach different audiences, or to adapt content to changing states of the world. What is needed is a means for producing text in a natural language (such as English) to describe the situations formally modelled in the program, in short: a natural language generator.[1] Such generators already exist, though they are still experimental, and in many ways unrefined. One particular shortcoming is the lack of an adequate mechanism for the expression of time and temporal relations. This work aims to remedy this situation.

Programs (hereinafter called **clients**, or **client programs**) which require the services of a natural language generator often need to express temporal relations between events and states. Until now, only tense[2] has been treated in a principled way computationally; mechanisms for expressing more specific temporal information have been largely ad hoc. As a result, the necessary information either is not being expressed or is being expressed without an appreciation of the details.

Consider the example of a scheduling program, which schedules meetings on a calendar: It records the dates and times of appointments, conferences, holidays, and vacations, for the user and for subordinates and associates. Typically, such a program would need to produce sentences such as:

1. *You will meet John Smith at 2:00 on Tuesday to discuss widgets.*
2. *You cannot meet Smith on the 25th. You will be out of town at AAAI on that day.*
3. *You won't need to see Smith again today. You already saw him at ten [today].*

Now, without a detailed model of the linguistic properties of the data, there is a danger that the text produced will be confusing, or inconsistent ("*#he will see you last Tuesday*"), or bad in some way (e.g. "**at Tuesday*").[3] There are three reasons for this:

1. Temporal relations may be expressed in more than one way, and aspects of the same relation may appear in more than one part of the same sentence (thus the inconsistency of "*will*" and "*last Tuesday*" above).
2. The exact form of the realisation may be dependent on the object being described (e.g., "*at 9:00*" and "*at Christmas*", but not "**at Tuesday*").
3. The form of realisation of a temporal relation can be varied (e.g., said on the 24th, "*tomorrow*" vs. "*on the 25th*").

[1] For an excellent discussion of the need for natural language generators, see Mann and Moore (1981).

[2] Tense will be used in a technical sense, referring jointly to tense and aspect, as the terms are traditionally used. Tense traditionally refers to the time of action or state, as shown by the form of the verb (e.g., past, present, future). Aspect traditionally refers to the duration or completion of an action or state, as shown by the form of the verb (e.g., perfect, imperfect). Thus, "*John had eaten*" is in the past perfect (past tense, perfect aspect). Our usage of tense is similar to that of Matthiessen (1984), who refers to primary, secondary, and further tenses; for him, primary tense corresponds to traditional tense, and secondary and higher tenses correspond to traditional aspect. Collectively, these are referred to as relative tenses.

[3] We will follow the linguist's convention of marking syntactically incorrect sentences with a star (*), questionable sentences with a question mark (?), semantically odd sentences with a number sign (#), and sentences acceptable only to speakers of certain dialects with a per cent sign (%).

From this we can see that a good generator has to be able to determine the correct form for expressing temporal relations, choose among several such forms (when necessary), and handle interactions between any forms actually realised.[4] Related to this is the requirement that the temporal data used by the client must be communicated to the linguistic component, and translated to a form which the linguistic component can use, taking into account the fact that the two systems probably view the world in different ways (perhaps substantially).[5]

No generator, past or present, would be able to deal with all aspects of the generation of the above example sentences. BABEL (Goldman 1975) could generate tensed sentences, but required the tense and aspect to be given as parameters. PENMAN (Mann 1983), given the correct grammar and choosers, would be able to generate the example sentences, but the choice of the prepositions would be complex, and related to tense, etc. in non-obvious ways. While both MUMBLE (McDonald 1983) and KAMP (Appelt 1985) could be made to generate the examples, neither of them has any model of the semantics of temporal expressions (i.e., the realisations of temporal data in natural language), since neither were designed to deal with the issues of concern here. MUMBLE deals with questions related to syntax and style, and KAMP determines what the appropriate content is. DIOGENES (Nirenburg et al. 1988) would be capable of producing the examples, but there would appear to be no co-ordination between tense and temporal adverbials.

Before we can generate temporal expressions, we must have a representation for the times involved, and we must have some understanding of the syntax and semantics of temporal expressions. Much work has been done on both, which will be briefly outlined here.

Various authors (Allen (1983), Bruce (1972), Kahn and Gorry (1977), Ladkin (1986), and McDermott (1982), to name a few) have produced representations for time. Since these temporal representation systems have not generally been used in natural language processing, it is not clear that the representational needs of language are served by the systems so far available. Of these, only Bruce (1972) describes a system which actually uses temporal expressions appearing in text, and his method for combining tense and adverbials[6] is not intuitive.

Many linguists[7] have suggested models for tense. Not all of them considered the interaction of their model with temporal adverbials. All gloss over some details of the representation. One of the more complete works (on which we will rely a great deal) is that of Dowty (1979), which presents a framework and definitions for a substantial fragment of English, and which discusses temporal adverbials and interactions with tense in some detail. Some of the work in theoretical linguistics has been taken up by computational linguists and AI researchers: Matthiessen (1984) is an excellent survey of work on the use of tense to express the temporal relations underlying an utterance; included in his work is a brief foray into temporal adjuncts and conjuncts. Pustejovsky (1987) has recently presented a novel theory of aspect which allows the internal event structure of a verb to be modelled.

A complete model of tense and temporal adverbials is lacking: AI temporal representations may not provide the right data for linguistic processing; linguistic models lack a connection to time units (i.e., recognized divisions of time, e.g., minutes, hours, days) and common-sense knowledge in general, and also lack an acceptable mechanism for co-ordinating the processing of tense and temporal adverbials. The goal of this work is to develop a theory for the semantics of time units

[4] **Realise** is used here in the same sense as in Quirk et al. p. 42, i.e., selecting some language form to express a notion.

[5] Unless otherwise noted, **temporal datum** will refer to the data structure (not the type) used by a client program to represent a time and made available by the client to the text generator for processing.

[6] The term **adverbial** refers to a functional category, like "subject" and "object," which has the function of an adverb. It can be realised by most formal linguistic constituents, such as adverbs, noun phrases, or prepositional phrases.

[7] See section 5.5 for references.

and temporal relations in natural language, and to develop a theory of processing this type of semantic information. These theories will be implemented as part of the semantic component for a text generation system.[8] It will then be possible to examine the impact of the requirements of the semantic processing on the temporal representation scheme used by the client, and to determine how current temporal representation schemes can be augmented to facilitate processing of this sort.

For reasons to be given shortly, I will not deal with all possible realisations of temporal relations, but concentrate on prepositional phrases serving as temporal adverbials, in particular those using the prepositions *in*, *at*, and *on*. For convenience I will use the term Temporal Prepositional Phrase (**TPP**) to refer to these. Unless otherwise noted, all references to TPP's will be to those using *in*, *at*, or *on*.

We restrict the study in this way in order to make manageable the amount of work to be accomplished while retaining an interesting and useful body of data. Prepositional phrases account for about 40% of all adverbials (temporal or other), and thereby make up the largest class of adverbials.[9] More to the point, they are the expressions a scheduling program is most likely to need when describing appointments. Despite the narrow range, there are interesting problems present. Strong interactions between prepositions and prepositional objects may be observed, as well as interactions between the TPP's and other syntactic constituents. These interactions in turn appear to depend strongly on common-sense knowledge. Interactions of this sort are wide-spread in language and an understanding of them would contribute greatly to natural language generation. Therefore, while the problem is manageably small, it has all the essential elements of a larger problem with wide-ranging ramifications. (It should be emphasized that restricting the study to a few TPP's does not imply that the approach to be used is inadequate for dealing with a wider variety of expressions. On the contrary, the approach lends itself very well to a general treatment of language data.)

Some aspects of the generation of temporal expressions will not be treated in this work (or treated only as they impinge on TPP's), either because the questions they raise are not primarily semantic ones, or because a treatment of them would dilute the focus of this work. Therefore, I will not deal with variation in realisations, the implicit communication of temporal relations,[10] the interaction of modals and semi-modals with time,[11] temporal connectives, negation, frequency adverbials, or deixis.

In section 2 we will examine the problem more closely. Data will be presented, together with a summary of earlier attempts at a solution, and a discussion of their weaknesses. In section 3 we will examine a coherent solution to the problem, and in section 4 we will discuss some of the details of an implementation of that solution. In section 5 we will review related work, especially in light of the solution proposed here. Section 6 presents a summary of the work.

2 The Problem

We have seen that the generation of TPP's is necessary to sophisticated text, but that their generation is difficult. Traditional views of the structure of TPP's have not taken into account common-sense attributes of the data, nor have they taken into account interactions between sentence constituents. As a result, only a partial explanation of the data has been achieved, requiring many

[8] This represents a stance with respect to the organization of a generation system, i.e., that such a system may be constructed using separate semantic and syntactic components. See section 4.1 for more details.

[9] Quirk et al. (p. 489), states that 10981 adverbials were found in a sample of ca. 75000 words, and of these, 4456 were realised as prepositional phrases. All other realisations were less frequently used.

[10] For example, if we hear "*John used to hide in the barn,*" and know that the barn burned down five years ago, then we know that the hiding must have taken place before that time.

[11] For example, "*I hope to get rich*" means that the speaker is not yet rich, but would like to be so in the future.

observations to be treated simply as curiosities. By exploiting common-sense knowledge and interactions, temporal data may be treated in a much more useful fashion.

2.1 Definitions

Before discussing the problem, we need to establish some terminology, reviewing some terms presented in the introduction, and introducing others. The term **adverbial** refers to a word or a group of words which together function as an adverb; it is a functional category, like "subject" or "object." Adverbials can be realised by adverbs, noun phrases, or prepositional phrases, among other constituents. By extension, **temporal adverbials** are adverbials which refer to time.

In this subsection we will refer to **tense** in its traditional sense, i.e. referring to the time of action or state as shown by the form of the verb (e.g., past, present, future). Here, as elsewhere, **aspect** refers to the duration or completion of an action or state as shown by the form of the verb (e.g., perfect, imperfect). A widely accepted (often with minor variations) analysis of tense and aspect due to Reichenbach (1947) assumes that utterances establish or use references to three times: speech time (T_s, the time at which the utterance is made, generally "now"), event time (T_e, the time at which the event named by the verb occurs), and reference time (T_r, a time in relation to which an event may be described). Thus, "*The project will have been completed by Tuesday*" establishes T_r as some time on *Tuesday*, in the future relative to T_s, and T_e as some time between T_s and T_r ($T_s < T_e < T_r$).

We will find it necessary to discuss the grammarians' view of temporal relations (note that this term is used differently in the temporal representation literature, and elsewhere in this report). For this purpose, I will give here a short list of such relations, as well as a list of the grammatical forms which may be used to realise temporal expressions, and of the functional forms in which temporal expressions are used.

Constituent Type	Position	Duration	Frequency	Relationship
Prepositional Phrase	*at ten*	*for two hours*	*from time to time*	*for the first time*
Noun Phrase	*today*	*three hours*	*three times*	*the first time*
Adverb Phrase	*soon*	*long*	*often*	*before*
Verb	*shall*	*last*	*drill [the troops]*	*regurgitate*
Adjective	*recent*	*old*	*frequent*	*Elizabethan*
Derivational Affix	*ex-husband*	*long-term*	*oft-quoted*	*post-war*
Inflectional Affix	*added*			

Table 1: Examples of Temporal Expressions

At least the following four temporal relations find expression in natural language:[12]

position — the time *when* an action occurred, or to which a state applies

duration — the length of time; called **span** when linked to a time position. Span is further subdivided into **forward span** (from some point in time forward to a relative future) and **backward span** (from some point in time backward to a relative past)

frequency — how often an action occurs, or a state applies

relationship — the relationship between two time positions.

[12] Quirk et al., pp. 481-482 and 526-555.

Prepositional phrases (**PPs**), noun phrases (**NPs**), adverb phrases (**AdvPs**), verbs (**Vs**), adjectives (**Adjs**), and affixes[13] (both derivational and inflectional) can all be used to express these relations. Table 1 shows examples of their use in expressing these relations. Note that any constituent type, excepting inflectional affixes,[14] can be used to express any temporal relation, within limits.[15]

Temporal expressions often appear in adverbials. There are three forms of temporal adverbials:

adverbials of number and frequency (*twice, often*) indicate the number of times a generic event takes place.

durational adverbials (*for an hour*) indicate the length of time a specific event takes.

frame adverbials (*from 8 to 10, at 10*) indicate an interval within which (or point in time at which) a specific event occurs. It may be split up further into **intervals** and **points**.

Notice that adverbials may be realised as PPs (*at 10, from 8 to 10, in the morning*, etc.).

2.2 The Traditional Analysis

Quirk et al. (1985:526-555) presents a traditional analysis, exemplified by the texts in (1a-3e). *At* is used to refer to points in time (1a, 1b), or to intervals which may be treated as points (1c, 1d). *On* is used only to refer to days (2a), or, together with a specific day, to major parts of days (2b, 2c). *In* is used otherwise (3a–3e).

1. (a) *at ten o'clock*
   (b) *at daybreak*
   (c) *at Christmas*
   (d) *at breakfast*
2. (a) *on Tuesday*
   (b) *on Tuesday morning*
   (c) *on the morning of the 17th*
3. (a) *in January*
   (b) *in the summer*
   (c) *in 1987*
   (d) *in the eighteenth century*
   (e) *in the evening*

Currently most models for the semantics of these prepositional phrases depend on precedence[16] or on containment relations, for example, that "*on the 25th*" means that the time of a particular

[13] Affix is the general term for prefixes, infixes, and suffixes. A derivational affix is an affix which creates a new word from an old one, such that the meaning of the new is derived from the old. An inflectional affix is an affix which merely changes the form of a word by inflection (e.g. for a past tense).

[14] This is probably due to the fact that English is not a highly inflected language.

[15] This is even true of verbs: If the action or state denoted by a verb necessarily takes place at some time, or has a duration, or is repetitious, or is related to some prior action or state, then the verb has a temporal component. Thus, *shall* refers to a future action, *last* refers to a state holding over a period of time, *drill* refers to a repeated action, and *regurgitate* (figurative) refers to a speech action following a phase of memorizing.

[16] Precedence simply means that one time is before another, i.e. one precedes another.

event was in the set of times contained by the 25th. Similar definitions are then given for phrases involving *at* and *in*. No further detail is attempted in these models.

There are several problems with the conventional analysis of TPPs. First, it is too vague; for example, no rules are given governing which intervals may be treated as points and which may not. Second, it is both too powerful and too weak (i.e., it excludes things it should allow, and allows things it should exclude) — witness examples (4–7) below. Third, it doesn't even touch on the interactions of TPP's with other temporal expressions, like those exhibited by examples (8–14) below. Fourth, it does not link the objects being described with the descriptions in any useful way.[17]

Obviously, relations of precedence and containment are not sufficient for explaining the following observations:

4. (a) *?at coffee break*
   (b) *%at the weekend* (British English)
   (c) *I worked hard all through the year/when I was in his course. At the time I thought it was a good idea.*
   (d) *At night I usually have the window open.*
   (e) **at Canada Day/Independence Day*
   (f) **at summer*
5. (a) *?on the early morning of the 17th*
   (b) *%on the weekend* (American English)
   (c) *on the following evening/morning*
6. (a) **in the early morning of the 17th*
   (b) *in the early morning*
7. (a) *at 9 on January 17th*
   (b) *#at 9 in January*

We must draw on our common-sense knowledge of how these events differ to explain these observations.

2.3 Interactions in Language

In addition to the curious behaviour of TPP's noted above, we must deal with the fact that linguistic entities interact with each other. We are all familiar with these interactions through the agreement required between subject and verb number. Similarly, different mechanisms for expressing temporal relations interact, as we can see in the examples (8-14) below.

8. *I saw him yesterday.* (tense)
9. *#I saw him tomorrow.* (tense)
10. *#At ten I had seen him at ten.* (aspect)
11. *?It will last forever at ten.* (verb)

[17] By this I mean that nothing is specified about the mapping from client data to linguistic data or about acquiring the data from the client or even about what sort of data is required from the client. Insofar as the conventional analysis is purely linguistic, this function may be judged to be beyond its scope. For our purposes, however, this link must also be established.

12. *Today at ten I will see you.* (more specific adverbial)
13. *?Today on the first I will see you.* (extra adverbial)
14. *?At ten I will see you soon.* (adverb)

The most obvious way in which temporal expressions interact is shown by the agreement required between tense and adverbials (8, 9). Tense requires T_t to be related to T_τ in a certain way. If a different relation is reflected in the adverbial, the sentence is bad, as in (9). Aspect, on the other hand, requires T_e to be related to T_τ in a certain way. The adverbials in (10) indicate that $T_e = T_\tau$, and not $T_e < T_\tau$, as required for perfect aspect, making the sentence bad. Some verbs assert that a proposition is true over a period of time; (11) is at best questionable, because "*last*" is one of these verbs while the adverbial indicates that that period should in fact be a point in time. Sentences (12) and (13) show that large intervals may be combined with smaller intervals or points without difficulty, whereas intervals of equal size may not be so combined. Finally, (14) is odd because "*soon*" is relative to the present moment, whereas "*at ten*" picks out a different time.[18] Subtler forms of interaction are also known.[19]

2.4 Common-Sense Roots of Interaction

Whereas earlier models have paid attention to the duration of events and precedence relations between events, it seems that little use has been made of our common-sense knowledge of these events. References to coffee breaks and breakfasts differ, for example, and the reason for this must lie in our view of the world.[20] By examining differences between events which figure in examples of acceptable and unacceptable text, we should be able to determine the common-sense knowledge required to answer questions like the following, and therefore to correctly generate temporal expressions.[21]

(1d vs. 4a) How do coffee breaks and breakfasts differ, such that one can be treated as a point and another cannot?

(4b, 5b) What is there about a British weekend which allows it to be referred to as a point, and what is there about an American weekend which allows it to be referred to like a day?

(4c) How is it that arbitrary intervals may be introduced and referred to deictically as points?

(4d) Why can the night be treated as a point, and not other times of day?

(1c vs. 4e, 4f) Why can certain holidays (or seasons, depending on your view) like Christmas and Easter be treated like points, but not others?

[18] In fact, the situation is probably more complex than this, since (14) is odd even if the present moment is the same as that picked out by the adverbial. Similarly, we would expect "*#I have seen him at ten,*" to be bad because the adverbial picks out a time other than the present moment, yet this sentence is bad even if that time and the present moment are identical.

[19] For example, "*John polished every boot*" appears to require all the boots to be polished simultaneously, unless the semantics of *every* includes references to time (Cresswell, 1985).

[20] It should be noted that expressions like these may be idiomatic, and a reliance on common-sense knowledge to explain such expressions should be argued for. An argument for such an approach would consist of finding expressions used similarly and for which usage a consistent common-sense explanation is applicable. The argument is strengthened if invented words used in the same fashion are judged by native speakers to refer to things with the same common-sense attributes as the things referred to by the observed expressions. This argument will necessarily follow from the work to be completed.

[21] The questions given are initial suggestions only. As work progresses, the questions may be rephrased or replaced by others.

(5a, 6a, 6b) Why is "*the morning of the 17th*" treated like a day, but not "*the early morning of the 17th*", and furthermore, if it can't be treated like a day-interval, then why can't it be treated like another kind of interval?

(5c) Why does the introduction of a deictic element force us to treat parts of days like days?

(7a, 7b) Why is it odd if we skip time units?

3 A Coherent Treatment of Time

We have seen that one of the principal problems in natural language generation is the need to select exactly the right phrase and the right words in that phrase to express some concept. We have also seen that this selection process depends both on the concept and on the syntactic environment of the expression. In particular, we have described several problems with the generation of prepositional phrases as temporal adverbials

1. Certain temporal units require the use of specific prepositions. Where variations on these units are allowed (e.g. *Christmas* vs. *Christmas Day*), the prepositions allowed for the variant usually differ from those allowed for the "root" form.
2. The agreement of tense and temporal adverbials must be ensured in some way.
3. Time intervals can sometimes be treated as points in time.
4. Prepositional phrases detailing different levels of time units describing the same event may be combined, but levels may not be skipped, unless they are clear from context. Thus, "*at 9 on Thursday*" is acceptable, but not "**at 9 in January.*"[22]

We have examined current models for TPPs and determined that they are based on precedence and containment only, and that, for our purposes, they are both too powerful and too weak, do not explain interactions with other functional classes, are not linked to real-world objects, and are in general too vague.

Any new approach to generating temporal expressions must address the problems with generation and the weaknesses of the current models. The solution I propose has four parts:

1. Associate with each word and phrase constraints that describe how the word or phrase can be used or what it refers to (e.g., *at* can be used to pick out a point in time, *Tuesday* refers to an interval of one day's length).
2. Base the constraints on common-sense knowledge.
3. Allow the constraints to propagate in the syntax tree for the sentence.
4. Require the constraints which meet at any node to be consistent.

In addition to this, some work must be done by the client. It must be required to make some of the decisions which can be expressed in non-linguistic terms, and to bundle together some of its information. The interface to the client must be designed in such a way that the client can be queried in non-linguistic terms about desired variations in expressions. For example, the client

[22] This example can be made acceptable by putting it in context, as in: A: "*When do you start work in the morning?*" B: "*At 9 in January, but at 10 the rest of the year.*"

should decide whether it wants to refer to the Christmas holiday or to the day on which Christmas occurs, and the client should bundle with that reference an indication of the level of detail it requires for the description. If insufficient information is available in the initial generation request made by the client, then it should be possible to query the client about its needs or preferences.[23]

The proposed solution remedies the problems with generation by

1. forcing the use of the correct prepositions and the agreement of tense and temporal adverbials by propagating constraints and enforcing constraint satisfaction
2. enabling the use of variations in phrasing, and enabling the treatment of intervals as points, by basing the constraints on the additional common-sense attributes of described objects.

The solution remedies weaknesses of the current theoretical model by

1. modelling interactions explicitly
2. defining links between client objects (representing real-world objects) and linguistic objects
3. attending to detail to a degree not practicable for a theoretical system, but required for the implementation of a processing model

The charge of being "too powerful and too weak" must be levelled against any system which disallows things it should allow while allowing other things which it should not. Whether or not the system proposed here is subject to this criticism must be determined empirically, but I claim that whatever discrepancies result from the use of this system will ultimately be remediable by refining treatments of objects.

We will examine the solution in the subsections following, starting with the minimum demands (with regard to temporal information) which will be made of the client in interfacing to the system. Next, we will examine the nature of the additional attributes which will be used with the constraints. After that, we will examine the mechanism of hierarchical constraint propagation to be used in this system. Finally, we will outline some of the experiments with which the system in general, and the constraints in particular are to be tested.

3.1 Basic Temporal Data

We must now examine basic aspects of the temporal data: the source and the basic attributes. Clearly, the source of the raw temporal data must be the client, since only the client knows which time is relevant. Furthermore, since variations on temporally identical expressions do not all share the same attributes, the selection of precisely which view of the temporal data to use must also come from the client program. For example, *Christmas*, *Christmas Day*, and *December 25th* can all denote the same thing, namely the twenty-fifth day of the twelfth month of any year, but they each carry a different intension — one is a season of the church year, another is a particular day within that season, and the third is the date of that day.[24]

[23] It could be argued that the client should include all the information it wants expressed in the initial generation request, since queries made later merely represent a delay in collecting needed information. This argument ignores the fact that the mechanism for bundling the information may well assume things about either the surface structure or the deep structure of sentences, and therefore force the client to have knowledge of language.

[24] While it could be argued that part of the decision to use a particular form could be made by some style component instead of the client, the main thing is that the decision is external to the grammar, and therefore outside the realm of this work.

We will define the basic attributes to be those of precedence and containment: The client should be capable of determining whether or not a given client temporal structure represents a time which precedes or contains any other client temporal structure. The next subsection examines necessary extensions to these basic attributes.

3.2 Time and Common-Sense Knowledge

In light of the data, it is obvious that the choice of a preposition must be in accord with the nature of the temporal datum being described, and equally obvious that more than just the 'size' of that datum, or precedence and containment relations with other data are important. At least the following attributes appear necessary:

Regularity with which an event occurs: breakfast is such a regular event it may be considered to be a useful reference point, thus allowing its treatment as a point in "*at breakfast.*"

Significance of the event: Christmas is such an important event in both the church and secular calendars that we treat it as a point, allowing "*at Christmas.*"

Duration of an interval: the duration of a weekend has the same order of magnitude as a day, thus possibly allowing its treatment as a day in *on weekends.*

Perception of time units: days have clearly defined boundaries, determined by a natural external source (the rising and setting of the sun), thus allowing us to conceive of them as somehow fixed or solid, and by extension, as things "on" which events can in some sense be "put," in analogy to spatial perception.

Granularity of time units:[25] the use of adverbs, such as *just*, is clearly a sign that there is an interval within which events are considered to be recent. Granularity is the size of this interval. Further research may show that a more complex notion, allowing different classes of recency, is required.[26]

3.3 Hierarchical Constraint Propagation

Stefik (1981) outlines a method for planning based on constraint satisfaction in a hierarchy, where a rough hierarchical plan is refined by using constraints to determine valid variable values in plan steps, and to reason out further constraints, which are propagated throughout the hierarchy of the plan. Stefik identifies formulation, propagation, and satisfaction as the three operations possible on constraints, and notes that constraint propagation is useful only when the problem involves loosely

[25] This is distinct from Hobbs' (1985) use of the term granularity.

[26] Three such classes which suggest themselves are: remote, recent, and immediate, corresponding to sentences like:

1. "*Bill left a long time ago.*"
2. "*John left today.*"
3. "*Tom just left.*"

It is not clear whether the size of objects in one class is a function of the size of an object in another class, or whether the sizes are unrelated. If these sentences were spoken in this order in close succession, we could imagine a situation in which the departure in (1) was a few hours prior to the time of speech, and the departure in (3) was just seconds prior. Thus, we could have the following correspondences: immediate: $\Delta t \leq$ *minutes*, recent: $\Delta t \leq$ *hours*, and remote: $\Delta t >$ *hours*. By the same token, we could imagine a situation in which the departure in (1) was many years prior to the time of speech, while at the same time maintaining that in (3) was just seconds prior. The given correspondences would remain the same, except for remote being $\Delta t >$ *years*.

coupled subsystems. He notes further that constraints represent a communication medium within a hierarchy.

We can fruitfully apply this to the domain of natural language generation, since a widely accepted model for the structure of sentences, the syntax tree, is hierarchical in nature. Given that choice of phrasing must be made within the hierarchy established by a syntax tree, four arrangements are possible for making decisions, and therefore for the directions in which constraints should be allowed to propagate.

1. Forced from above
2. Forced by a single element from below
3. Cooperative decision from below
4. Cooperative decision from above and below

(1) is ruled out because it requires low level detail to be used to make high-level decisions before the details are apparent. Both (2) and (3) are ruled out because they do not allow proper treatment of **C-command** and **government** mechanisms widely used in linguistics to describe the influence of one node in the syntax tree on its 'cousins'.[27] (4) avoids both difficulties, while providing an effective model for choice of phrasing.

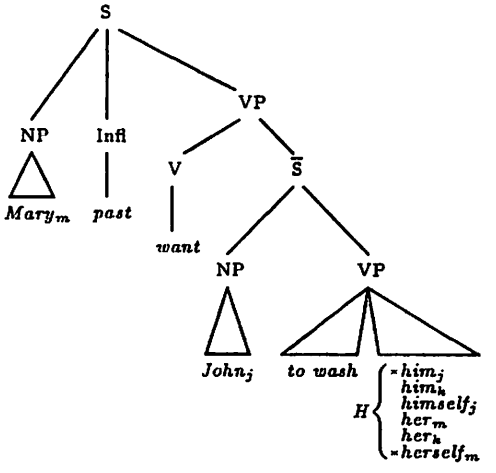

Figure 1: Reflexives and Government

Another way of stating (4) is to say that phrases mutually constrain each other. Figure 1 illustrates this situation, showing a syntax tree for the sentence "*Mary wanted John to wash H*," where H can be $himself_j$, him_k, her_k, or her_m, depending on whether H refers to *John*, or to another male, or to any other female, including *Mary*.[28] Now, *John* governs H, but *Mary* does not. Since a pronoun is constrained to be reflexive when co-referent with a governing node, H must be reflexive if it co-refers with *John*, but may not be reflexive if it co-refers with *Mary* instead.

[27] More technically, α c-conmands β if the first branching node dominating α also dominates β, and neither α nor β dominate each other. α governs β if the α is the verb, adverb, noun, preposition, or tense marker closest to β, and c-commanding it, and no sentence boundary intervenes. Government and c-command can be used to explain why, for example, reflexive pronouns are sometimes required and sometimes not allowed (see below). The reader is invited to consult the technical literature for a more precise definition (see Radford (1981) and references cited therein). For our purposes, this definition will suffice without clouding other issues.

[28] Subscripts in the figure indicate co-reference, i.e., that two names denote the same object.

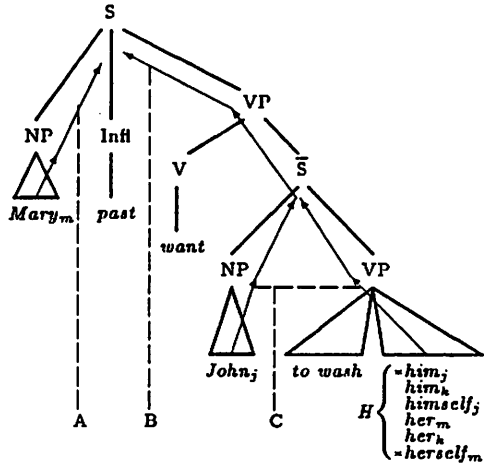

Figure 2: Reflexives and hierarchical constraint propagation

Figure 2 shows the same syntax tree from the point of view of hierarchical constraint propagation, focussing on the constraints on reflexives. The identity of the subject is given as part of constraint A. Any reference to *Mary* in a phrase governed by the subject must now be reflexive, and any references to others must be non-reflexive. Thus, constraint B must contain either no references to *Mary*, or reflexive references only. The constraints marked C are similar: If *H* refers to *John*, the constraints only allow the reflexive form, thus $himself_j$ is acceptable but $*him_j$ is not. If *H* refers to another male, the constraints only allow the non-reflexive form, thus him_k is acceptable, but $*himself_k$ is not. If *H* refers to *Mary*, then only the non-reflexive form is allowed, thus her_m is acceptable, but $*herself_m$ is not. Note that the constraint from A forcing the reflexive form for co-referents of *Mary* does not apply, since *H* is not governed by *Mary* but by *John*. Finally, if *H* refers to another female, then the constraints apply as for another male, thus her_k is acceptable but $*herself_k$ is not.

Figure 3 shows another example of hierarchical constraint propagation, this time concentrating on temporal expressions. Let us examine in turn the generation of each temporal expression. The generation of "*at nine*" proceeds by satisfying the point requirement of *at* (C in the figure) with the point nature of the hour *nine* (D). Similarly, "*on the seventeenth*" is generated by satisfying the day-interval requirement of *on* (F) with the nature of *the seventeenth* (G). The time expressed by each of these PP's is propagated (E) to 'texpr' and found there to be consistent.[29] These constraints are combined and propagated ultimately to the S-node (B), to be combined with the constraint derived from the tense to be used (A).

To underscore this example, let us consider some other sentences in which phrases like this appear. (We assume the sentences are all said after the seventeenth.)

15. **John saw Mary at the seventeenth.*
16. *?John will see Mary on the seventeenth.*
17. *John saw Mary on the seventeenth.*

The difficulty with (15) lies with combining a preposition with a particular noun phrase, and not with combining TPP's in a sentence. The difficulty with (16) lies with agreement between the tense

[29] Note that the propagated values are not 'nine' or 'the seventeenth', but rather the client structure from which both of these was derived.

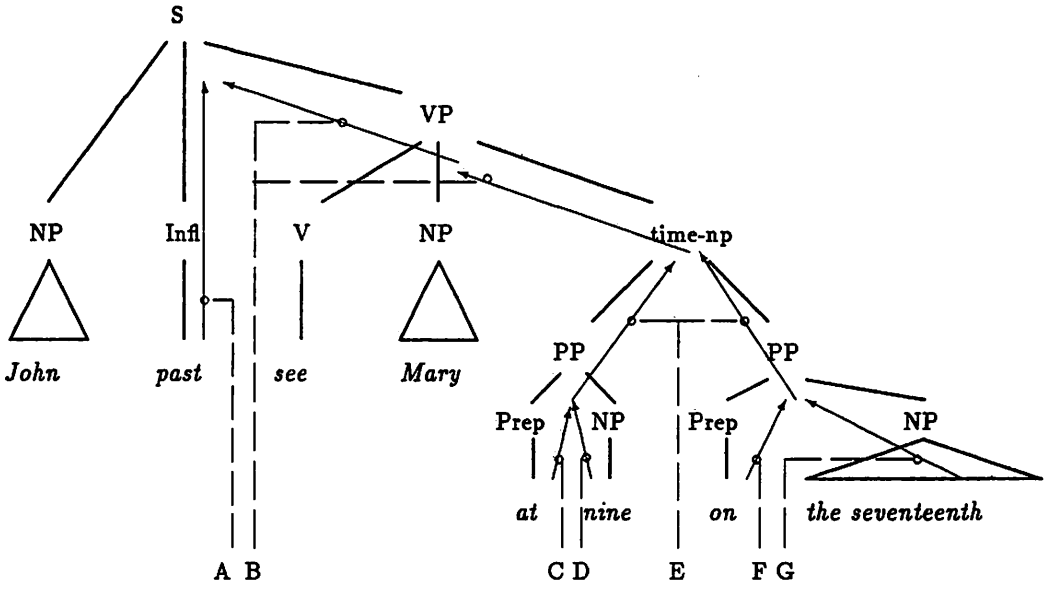

Figure 3: Hierarchical constraint propagation and temporal expressions

and the time expressed by the TPP. In contrast to these two, (17) is perfectly acceptable, because all of the constraints are in harmony.

From the above, we observe that semantic processing for natural language generation is a classic candidate for a solution of the kind proposed in Stefik (1981). The hierarchy is provided by the syntactic structure of the sentence. The constraints are provided by word meanings or transformations on inherited constraints. Realisations for words and phrases are selected on the basis of satisfied constraints, and the selection process for one word or phrase is loosely coupled with the process for another, as required for this kind of solution.

GNOMON, a semantic component for text generation being developed to test these theories, uses hierarchical constraint propagation as the basis of its processing. The syntax tree is constructed with instantiations of syntactic elements, known as **possible realisations**. These elements may be underspecified forms, forms lacking a parameter value, or forms referencing an unexplored optional class of phrases. If several possible instantiations offer themselves as expansions for a given parameter or class, then constraints are applied to narrow the range. If this does not narrow the selection satisfactorily, a filter (a kind of global constraint) may be applied. Alternately, it may be deemed suitable to concentrate on another part of the problem and wait until constraints are propagated back to the original problem, in the expectation that the selection will be reduced after application of the constraint. Just which operation is to be applied is determined by the application of rules.

3.4 Experiments

Previous sections left open the details of how constraints will be manipulated, and what constraints will be used. This subsection suggests some computational experiments to determine what constraints to use, in particular in the generation of TPP's. The section following this deals with how the constraints are to be used.

The experiments will address the nature of the knowledge required for the generation of

correct text.[30] Their primary aim will be to determine what constraints are useful and necessary, to determine what the form of those constraints should be, and to suggest one or more constraint sets to be used. A by-product of this will be to provide guidelines for judging constraints and a set of judgements for the constraints studied.

Before starting on the experiments, we need to collect a set of sentences which represents a wide range of possible temporal expressions of interest, both valid and invalid.[31] The temporal expressions will be used in two ways. First, they will serve as starting points in our search for useable common-sense constraints. By comparing pairs of acceptable and unacceptable sentences with minimal differences, we can get a good idea of the common-sense knowledge being used, giving special consideration to the attributes noted in section 3.2. Second, the sentences will serve as test cases by which to judge the correctness of the output. From the sentences, we must devise a number of scenarios in which their utterance would be reasonable, and then translate these scenarios into client data objects.

The next step is to formulate a relevant constraint for each attribute noted above, and to run the generator with various combinations of constraints on all scenarios, noting when it overgenerates or undergenerates. If they overgenerate or undergenerate to some degree, we must choose that combination which appears to differ least from a perfect match, vary the formulation of some constraint, and reiterate. Our work is complete when one combination generates all the correct sentences and no others.

4 An Implementation

Much of the work described above has already received implementation, which I describe in this section.

4.1 System Model

The overall system includes:

- a client program (a program which requires text to be generated)
- a semantic component (GNOMON)
- a syntactic component.

Figure 4 illustrates this configuration.

From the point of view of this work, the heart of the system is GNOMON, the semantic component, which we will examine in detail in this section. The other components will be treated as black boxes, except for a brief description here to provide a concrete basis for the examples to follow.

The client program chosen for experimentation is a program to schedule appointments, called CALENDAR. CALENDAR can be used to record upcoming appointments, deadlines, etc., in a convenient form, using the graphics interface of a lisp machine. It warns users before an event approaches

[30] They will not deal with what may be considered primarily design issues, such as the format of the interface language between client and generator. Though some of the design issues might usefully be the subject of experiments, for the time being a fixed configuration will be determined and used for the duration of the experiments (see section 4).

[31] Ideally, this set would be exhaustive, but this is clearly impossible.

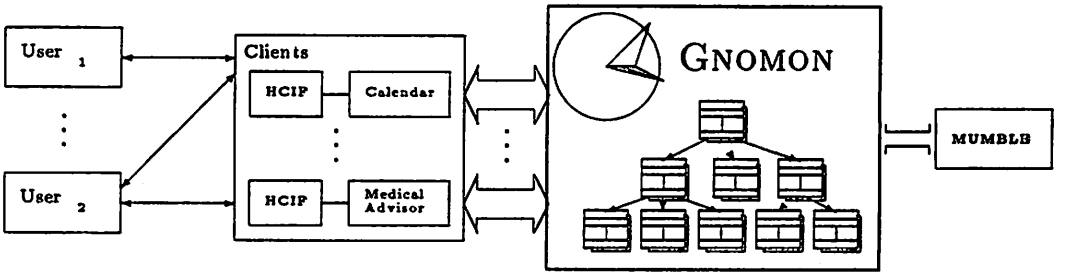

Figure 4: The system model

and reminds them again when the event arrives. Currently, CALENDAR uses a prototype of GNOMON to generate descriptions of events and to generate error messages.

To interface to GNOMON, CALENDAR uses a special language being developed as part of this work. The language provides access to each of three communications routes. The first route is via the actual request to generate, built from messages consisting of message type and values, loosely based on thematic roles. The second route is via the global context record, which records largely non-linguistic facts about a dialogue, such as the identity of the speaker or the audience, etc.[32] The third route is via queries about common-sense attributes of client data objects which GNOMON can direct at the client. The client must provide functions which can answer these queries. This in particular provides two-way communication between the client program and GNOMON.

The syntactic component is MUMBLE, a text generator developed by McDonald (1983). MUMBLE generates text by generating and then traversing a **surface structure**, a structure like a syntax-tree. Leaf nodes are words, and interior nodes are choice structures, i.e., structures representing different syntactic realisations of some linguistic content (e.g., "*John eats sushi.*" vs. "*For John to eat sushi...*"). When traversing the structure, word nodes are "said," and choice structures are expanded to generate further surface structure. Note that the choice is made immediately upon expansion, based on grammatical validity. In this regard, GNOMON and MUMBLE stand at opposite ends of the spectrum. GNOMON delays details, while MUMBLE fills them in immediately.[33] GNOMON uses the interface language defined as part of MUMBLE to interface to it.[34]

4.2 The Algorithm

GNOMON uses hierarchical constraint propagation to choose applicable realisations and lexical items based on constraints derived from the realisations themselves and from context. The hierarchy is based on a syntax tree for an utterance.[35]

Conceptually, there is a set of possible realisations at each node of the syntax tree, explored or unexplored. Taken together, these represent all possible ways of expressing the information contained in the messages to GNOMON. It is GNOMON's task to search this tree for an appropriate utterance. A possible realisation is represented (recursively) by a form consisting of uninstantiated

[32] GNOMON cannot know when these identities change, but the client may, and therefore we give it the opportunity to convey these changes to GNOMON.

[33] McDonald claims that this makes Mumble psychologically valid, since it seems that people start to speak before sentences are fully thought out.

[34] Although this language does not now have two-way communication built in, GNOMON is designed to allow two-way communication with the syntactic component. Note that there is nothing in MUMBLE's design philosophy to prevent extending the language in this way.

[35] In fact, so much syntactic information is available from this that future design changes may obviate the need for a separate syntactic component.

forms and directives for the syntactic component. Uninstantiated forms are realisable only by further possible realisations. Figure 5 illustrates this. One set of possible realisations is shown explicitly by the boxes marked "parameter-1, possibility 1" and "parameter-1, possibility 2". Elsewhere in the diagram they are indicated by the use of "stacked" boxes.)

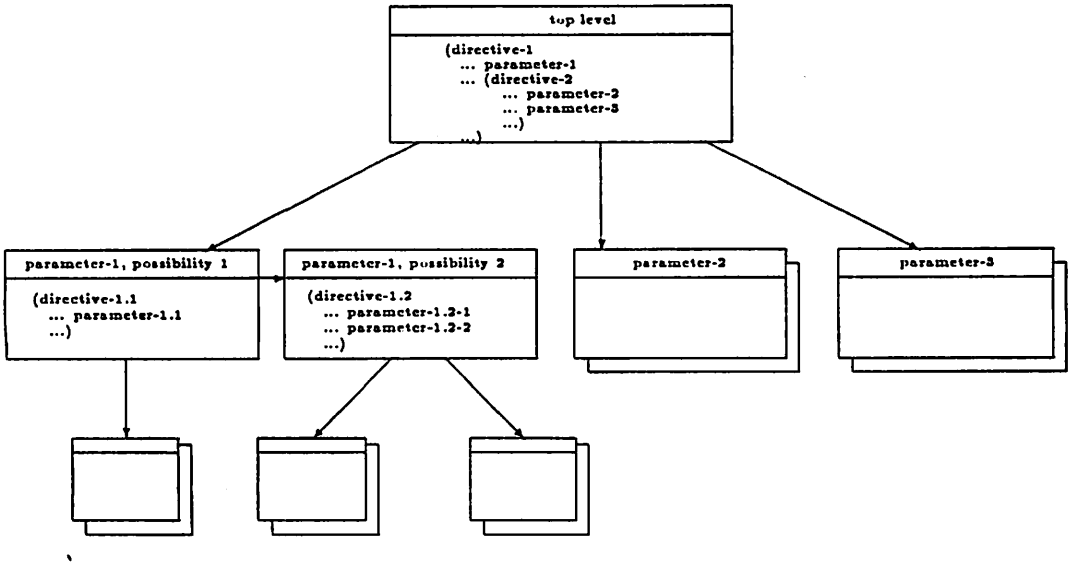

Figure 5: A possible realisation tree

4.2.1 Context Specifications

Possible realisations are derived from c-specs, or context specifications, which provide information about the syntactic form to be used to realise a particular phrase, the constraints on its use, and the effect the form has on context. The format of a c-spec definition is given below. See section 4.3 for examples of its use.

(`define-context-specification` *name class vars initial-conditions uninstantiated-classes syntactic-form target context-effects constraints*)

Name names the c-spec for later reference

Class the class of uninstantiated form which the syntactic form contained in this c-spec can instantiate

Vars variables to be used within the body of the c-spec

Initial-conditions conditions which have to be met if the c-spec is to be considered for instantiation[36]

Uninstantiated-classes the classes of forms within the syntactic form contained in this c-spec that are uninstantiated

Syntactic-form a message to be passed to the syntactic component to realise a desired phrase

Target where the syntactic form should go in the containing syntactic form; in general, this is a specific slot related to the class type, but a target may be specified which instructs the syntactic component to append the form to an utterance

[36] This may well be subject to replacement by reliance entirely on the constraints or on conditions derived automatically from the constraints.

Context-effects expected changes to the current context forced by the use of this c-spec in an utterance, e.g. by making an object available for later anaphoric reference
Constraints constraints on the use of this c-spec

4.2.2 Design Issues

A number of design issues had to be decided before GNOMON could be built, including:

1. What does the interface language between client and GNOMON look like? In particular, what is its format, and what are the primitives?
2. How are parameters made available to the c-specs? What are the c-specs allowed to do with the parameters? Are the parameters available globally, or locally? If locally, is there a protocol for accessing non-local variables? If local, how are parameters passed, and if global, how do we ensure that non-local data does not adversely affect the text generated?
3. Under what conditions should filters be applied?
4. Under what conditions should uninstantiated forms be expanded?
5. What kinds of transformations should be allowed? Just those based on some form of reasoning (as in Stefik), or any transformations generated by the application of some function?
6. Should nodes be opaque or transparent to constraints, i.e., should they normally transmit a constraint, or should it be necessary to specify that the constraint be transmitted at a given node?
7. Where is constraint satisfaction tested? How is it tested, i.e., do constraints have to be equal, or just consistent? If they need only be consistent, how is consistency measured? Are all constraints equally important?

Some of these questions might also qualify as subjects for experimentation, but for the purposes of this work, we must delay experimentation of this sort. The design to be used for the duration of this work is described below. Decisions have been made as conservatively as possible, i.e., so that as much control as possible can be exercised over the generation process.

1. Interface language: GNOMON receives its parameters in a list consisting of pairs of parameter type and parameter value. The parameter types include `agent`, `experiencer`, different eventuality types, etc. The types can be extended or modified as necessary.
2. Parameters and c-specs: c-specs see only the parameters provided them by the interpreter. At the top level, the parameters given to GNOMON are simply passed on to the c-specs. At lower levels, c-specs might specify what parameters should be passed on to subordinate uninstantiated forms. We will allow constraints to be formulated by the application of arbitrary functions. Information will be available both from the global context record and from the parameters.
3. Filters: Filters are applied when the application of advisory rules suggests it.
4. Expansion of uninstantiated forms: The forms are expanded when the application of advisory rules suggests it.
5. Allowed constraint transformations: We follow Stefik's lead in allowing only those transformations based on some form of reasoning.

6. Opacity vs. transparency: In order to have more control over the propagation of constraints, we require nodes to be normally opaque.

7. Testing constraint satisfaction: Constraint satisfaction is required wherever constraints meet. Satisfaction is achieved when constraints are found to be consistent, where consistency is determined on the basis of the constraint types. For the time being, all constraints are considered to be equally important.

4.2.3 Processing

GNOMON first creates a top-level underspecified form consisting of the single expression ((:parameter :utterance)) and queues up an expansion operation (see below) on this parameter. It then processes the queue, applying processing rules to determine which operation to execute next, terminating when the queue is empty. The results are then passed on to the syntactic component. Finally any effects on global context are noted.

One of four operations can be enqueued, each of which may subsequently involve queuing up other operations:

Expansion — one or more valid realisations may be found for an uninstantiated form (denoted by (:parameter :<parameter-class-name>)) and entered as possible realisations of that form.[37] A narrowing operation on the form is enqueued, as are expand and propagate operations on all the children.

Propagation — constraints resulting from the current expansion may be propagated to the parent node, and from there to other parts of the realisation tree. If new constraints are introduced, a propagate operation on the parent node is enqueued.

Narrowing — the constraints on an uninstantiated form may be evaluated for consistency, and inconsistent realisations rejected. If only one possible realisation remains afterwards, it is instantiated. If the application of constraints resulted in a change, then a propagate operation on this node is enqueued, otherwise a filter operation is enqueued.

Filtering — a set of realisations may be filtered based on non-grammatical criteria. This happens only if more than one possible realisation would still be allowable for some node after a quantity of processing.[38] If the filter operation results in a change, then a narrow operation on this node is enqueued, otherwise another filter operation (specifying a different filter) is applied.

The exact action to be taken depends on advice from a set of higher level constraints. These constraints are intended to reflect conventions in speech which lie outside of semantics and syntax, such as Gricean maxims.[39] Thus, in obeying the maxim of brevity, it may be advisable to prefer anaphoric references, and filter out other realisations when an anaphoric realisation is available.

[37] Valid realisations are those realisations for which the initial conditions for use are satisfied. The initial conditions usually depend on message data, but may also depend on the context. For example, "*you*" may only be used when referring to the audience. In GNOMON, context is recorded in a fairly simplistic way, allowing reference to objects recently introduced in conversation, or to participants in the conversation, or to recent events. The context attributes recorded are based on work by Lewis (1972), and include :audience, :speaker, :location, and :time.

[38] For example, both "*you*" and "*the person I am talking to*" are valid ways of referring to the audience, but *you* is preferable because of its brevity. Note that filtering is not intended to be a last resort only, since it should be possible to invoke it when the speaker is pressed for time.

[39] Grice (1975) suggests that in normal speech, certain conventions are respected. For example, people tend to stick to the topic in conversation, tend to be succinct, polite, etc. None of these things are governed by syntax or semantics, yet they are part of the speech generation process.

Since these conventions are different in different cultures, and therefore independent of the actual process of generation, it is inadvisable to build them into the generator itself, but rather to express them as governing constraints.

Though the actual rules to be used are in a state of flux, we can assume for the time being that they include:

1. items are propagated bottom-up (prefer nodes lower in the tree)
2. propagate from child nodes before you narrow the parent
3. narrow bottom-up
4. narrow only when constraints are available
5. narrow before you expand
6. filter when you can't narrow
7. expand left-to-right if possible

An abridged form of the algorithm in pseudocode is given in appendix A. for reference.

4.3 Detailed Examples

Let us now follow the generation of the text

"*You will meet Philip on Tuesday.*"

from its request in the client program, through its processing by GNOMON, to its delivery to the syntactic component.

In its normal operation, the client program (a scheduler) will need to report when certain events are scheduled, as well as who is involved, and possibly where they will take place. To request the generation of the above sentence, the client sends the following message to GNOMON:

```
((protracted-event #<EVENTUALITY meet>)
 (agent           #<PERSON David>)
 (experiencer     #<PERSON Philip>)
 (reference-time #<TIME Tuesday, 19-APR-1988 14:00>)
 (event-time      #<TIME Tuesday, 19-APR-1988 14:00> :day)
 (speech-time     #<TIME Monday, 18-APR-1988 10:00>))
```

Here, `protracted-event` names the type of the eventuality to be described, in this case an event with some duration.[40] `Agent` introduces the entity which carries out an action or is in some state, while `experiencer` introduces the entity which is acted upon. `Reference-time`, `event-time`, and `speech-time` introduce the times relative to which the eventuality is described. For present purposes, it is unnecessary (and possibly misleading) to give details on the implementation of the structures `EVENTUALITY`, `PERSON`, and `TIME`. Suffice it to say that they are client structures which may be translated into lexical items by routines defined by the client.

Prior to sending the message to GNOMON, the client program updates the recorded context to reflect its knowledge of the identity of the speaker, the listener, the current time, etc. In this

[40] Although *meet* is used here in the sense of a protracted event (e.g., an appointment), note that it is often used ambiguously to refer also to a momentaneous event (an introduction or brief encounter). Note also that the protracted event contains the momentaneous event. (For more details on eventualities, see Dowty (1979), Bach (1981), or Pustejovsky (1987).)

case, it notes that the current listener is #<PERSON David>, the speaker is the machine, and that the current time is #<TIME Monday...>.

GNOMON applies the algorithm presented in the previous section. It selects the underspecified form ((:parameter :class)) for realisation, enqueuing an expand operation on it. The resulting form is:[41]

```
(discourse-unit
  :head (general-clause
          :head (:parameter :class)
          :accessories (:unmarked)
          :further-specifications nil))
```

In addition to the uninstantiated :class form visible here, several other parameters require expansion, including :tense and :event-time, which may be appended to the form as :accessories or :further-specifications. Expanding the :class form yields:

```
(discourse-unit
  :head (general-clause
          :head (transitive-verb_two-explicit-args
                  (:parameter :verb)
                  (:parameter :agent)
                  (:parameter :experiencer))
          :accessories (:unmarked)
          :further-specifications nil))
```

Note that a form with three uninstantiated forms (:experiencer, :agent, :verb) has taken its place. These forms are each expanded in turn. Several different realisations for a given form may be considered. For example, among the realisations considered for the :agent form are "*David*" and the one eventually chosen, "*you*." The specification for "*you*" follows:

```
(define-context-specification
   2nd-person :agent                ; name and class
   ((agent (find-message 'agent))) ; put any 'agent' message in variable 'agent'
   (and agent                       ; if agent message is present, ...
        (eq agent                   ; ...and is same as audience,...
            (get-context-attribute :audience))) ; ...then can use this spec
   ()                               ; no further uninstantiated forms
   'second-person-singular          ; syntactic form to be used
   :agent                           ; use to replace form ':agent'
   nil                              ; no effect on context
   nil)                             ; no constraints to satisfy
```

This specification may only be used if an :agent message was given to GNOMON, and the entity named in the message is the audience.[42] If used, the form second-person-singular is inserted in place of the form :agent in the final r-spec. A specification may call for other classes to be processed in order to define uninstantiated forms within the syntactic form to be used. Similarly, a specification may note that its use will have some effect on context (e.g., that a particular event was introduced into the discourse), or that other linguistic constraints on its use must be satisfied. This specification requires no further processing, has no effect on context, and requires no further constraints to be satisfied.

The projected specifications for TPP's show more of the use of constraints in GNOMON, as can be seen below and in figure 6.

[41] The forms used here to represent the input to the generator are actually Mumble r-specs.

[42] The :agent message, if present, is located by the routine find-message. The identity of the audience is maintained as part of the context (cf. Lewis, 1972), and is found by using the routine get-context-attribute.

```
(define-context-specification
   temporal-pp :time
   ((time (find-message 'event-time)))
   time
   (:time-np :prep)
   '(:specification       (prepositional-phrase
                             (:parameter :prep)
                             (:parameter :ttime-np))
     :attachment-function vp-prep-complement)
   :further-specifications
   '(:time ,time)
   '((:parameter :time-np time)))


(define-context-specification
   on :prep
   ()  ; no vars
   t  ; always useable
   ()  ; no additional classes
   "on"  ; word 'on'
   :prep ; sub for uninstantiated form 'prep'
   ()  ; no effect on context
   '((time-nature :day-interval))) ; require related time to be point


(define-context-specification
   temporal-np :time-np
   ((time (find-message 'event-time)))
   time
   ()
   '(general-np :head (np-proper-name ,(noun-for time))
                :accessories (:number singular
                              :determiner-policy no-determiner
                              :gender neuter
                              :person third))
   :time-np
   '(:time ,time)
   '((time-nature ,(cond ((hour? time) :point)
                         ((day? time)  :day-interval))
                         (t :non-day-interval)))
     (time ,time)))
```

The `temporal-pp :time` would be applicable whenever an `event-time` message was present, providing it was consistent with the constraints. This c-spec would generate a PP to be added to the end of the containing phrase (due to the specification of the `:further-specifications` target). The c-spec inherits the `time` constraint from the `:time-np` child.[43] The on `:prep` *c*-spec requires the nature of the time to be a day-length interval. The `temporal-np :time-np` *c*-spec also requires a day-length interval, but also constrains the time to be consistent with the time expressed by it.

Consulting figure 6, we can see the constraints on time nature being propagated up to the `:time` node. A narrowing operation on the :prep node results in the choice of "*on*."

Processing continues until the execution queue is empty, after which the *r*-spec below emerges (the root of each word appearing in the final text appears on the right-hand side). The *r*-spec is sent to MUMBLE, which then produces the desired text. GNOMON finally records the changes to context made by the sentence (e.g., the time most recently mentioned would now be *Tuesday*, which could be referred to anaphorically as *then*, under the right circumstances).

[43] The exact form to be used to specify constraints derived from the constraints imposed by subordinate nodes is still under development. For details, see the appendix.

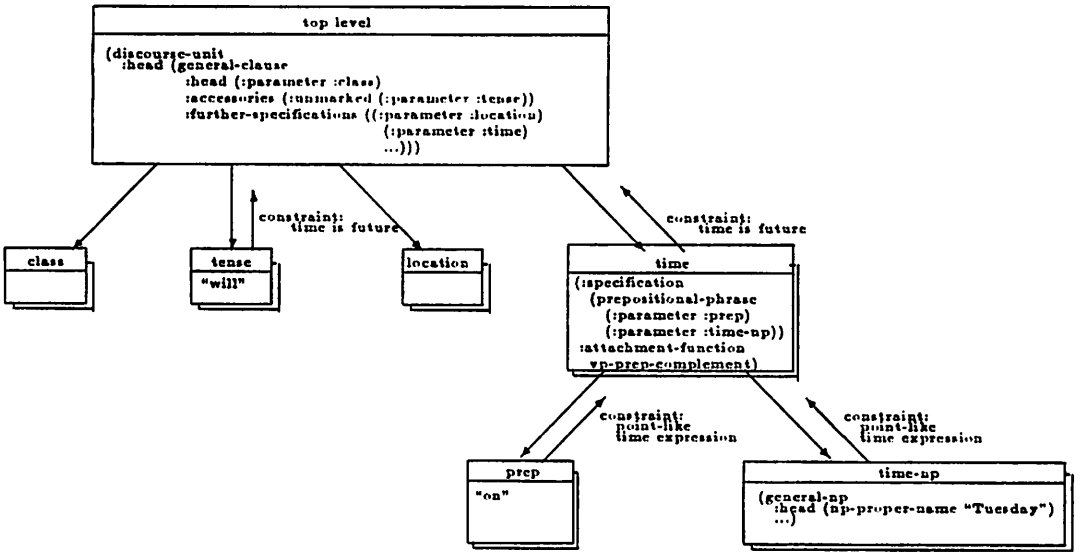

Figure 6: Constraints in example

```
(discourse-unit
  :head
    (general-clause
      :head
        (transitive-verb_two-explicit-args
          "meet"                                    ; meet
          second-person-singular                    ; you
          (general-np                               ; Philip
            :accessories (:number singular
                          :determiner-policy no-determiner)))
      :further-specifications
        (:specification                             ; on Tuesday
           (prepositional-phrase
             "on"
             (general-np
               :head (np-proper-name "Tuesday")
               :accessories (:number singular
                             :determiner-policy no-determiner)))
         :attachment-function vp-prep-complement))
  :accessories
   (:tense-modal "will"))                           ; will
```

Figure 7 shows the possible realisation tree for this example.

To better understand the functioning of the system, let us follow the generation of a second example:

"*You will have already met Philip at two.*"

We will assume that the client is reacting to a suggestion by the user that the user and Philip meet at some later time; the user is presumably unaware of the meeting already scheduled. In this case, the message to GNOMON could be as follows:

```
((protracted-event #<EVENTUALITY meet>)
 (agent           #<PERSON David>)
 (experiencer     #<PERSON Philip>)
 (reference-time #<TIME Tuesday, 19-APR-1988 18:00>)
 (event-time      #<TIME Tuesday, 19-APR-1988 14:00> :hour)
 (speech-time     #<TIME Monday, 18-APR-1988 10:00>))
```

We will assume for purposes of exposition that the sentence from the first example has not been

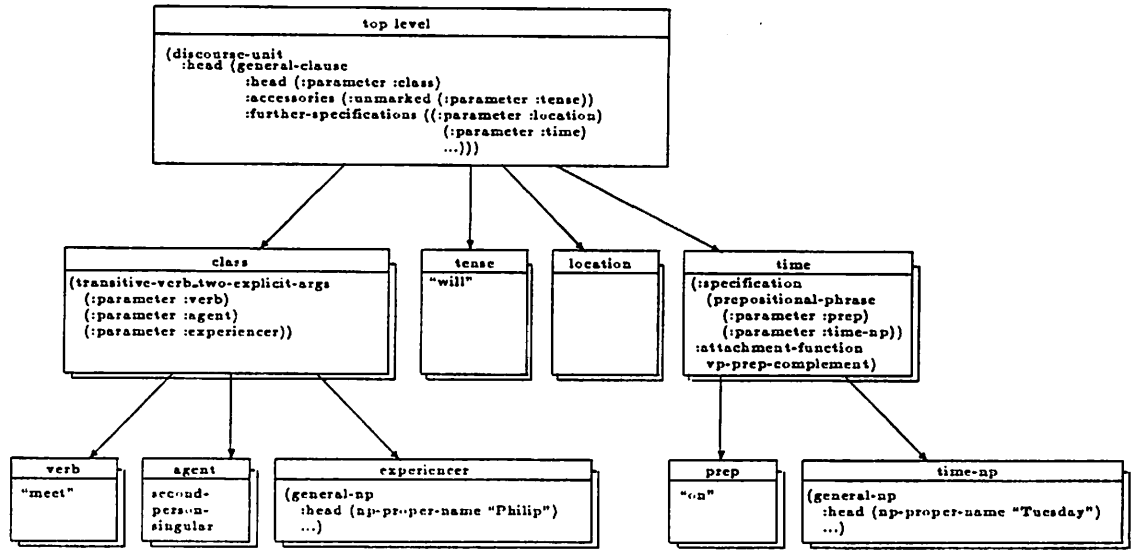

Figure 7: Possible realisation tree for "*You will meet Philip on Tuesday.*"

generated prior to this.[44]

Generation proceeds as before, with the recorded context being updated by the client to reflect the current context known to it. GNOMON selects the same underspecified sentential form for realisation as before, and performs the same expansions, with a few slight differences: the prepositional phrase used is "*at two*" instead of "*on Tuesday,*" the tense is future perfect instead of simple future, and the adverb "*already*" is inserted in the *r*-spec at the appropriate place. The result is shown below:

```
(discourse-unit
  :head
    (general-clause
      :head
        (transitive-verb_two-explicit-args
          "meet"                                        ; meet
          second-person-singular                        ; you
          (general-np                                   ; Philip
            :head (np-proper-name "Philip")
            :accessories (:number singular
                          :determiner-policy no-determiner)))
      :further-specifications
        ((:specification                                ; at two
            (prepositional-phrase
               "at"
               (general-np
                 :head (np-proper-name "two")
                 :accessories (:number singular
                               :determiner-policy no-determiner)))
          :attachment-function vp-prep-complement)
         (:specification                                ; already
            (adverbial "already")
          :attachment-function adverbial))
  :accessories
   (:tense-modal "will" :perfect))                      ; will have
```

The reason for these differences is simple: In this example, the conditions for using perfect aspect are satisfied, contrary to the previous example. Similarly, whereas in the previous example the

[44] If the sentence from the first example had been generated before this one, there would be some interaction between the two, due to the influence of recorded context, resulting in the generation of "*You will have met him at two.*"

conditions for using the preposition "*on*" are satisfied and the conditions for "*at*" are not, here the situation is reversed. Finally, because of the salience and the timing of the prior meeting, the conditions for using "*already*" are satisfied.

The justification for inserting the adverb "*already*" is somewhat more complex. As has been noted before, the semantics of *already* may involve the violation or contradiction of an expectation of one of the discussants.[45] Information concerning expectations could be communicated to GNOMON either directly, via the messages, or indirectly, by consulting the context or, when all else fails, querying the client during planning. For the purposes of this example, we have elected the indirect route.[46] We will assume that some record of the proposed later meeting between the user and Philip is part of the context record, most reasonably in the category of "previous discourse." By consulting this record, and comparing the time of the proposed meeting with that of the scheduled meeting, we can determine that the conditions for "*already*" are fulfilled.

5 Related Work

The present work is in part a synthesis of work from many different disciplines within AI, as well as work from allied fields. From linguistics and philosophy come the data which must be accounted for, as well as theories of language processing and meaning, and a methodology for investigating certain kinds of problems. From AI come mechanisms for representing time, for representing and processing natural language, and for manipulating constraints. The following sections will discuss major issues in these fields as they relate to the present work.

5.1 Formalisms for the Representation of Time

A number of researchers have wrestled with the problem of representing time, in particular, for purposes of natural language processing and planning. Bertram Bruce (1972) published an early paper on time and natural language processing, which represented time simply as intervals with endpoints defined symbolically. Kahn and Gorry (1977) later experimented with a "time specialist," a program which could reason using several different temporal representation schemes, but each of

[45] We can extend the findings of Hoepelman and Rohrer (1981) to English, using their model for the semantics of "durative" *schon* for the pertinent interpretation of *already*, with one slight alteration: They noted that the semantics of "durative" *schon* depended on the violation or contradiction of an expectation of the speaker (cf. section 5.5). Sentences like "*You already saw Philip at two,*" however, may involve a perceived misconception of the listener which the speaker is attempting to rectify, as in the following, said at six in the afternoon:

A: I want to see Philip now.
B: But you already saw him at two.

Hence, we should properly talk about the violation or contradiction of an expectation of the speaker *or* listener.

[46] There are other reasons for choosing the indirect route. The violated expectation of concern here is that two so similar events (the meetings) do not normally occur so closely together. Consider that the event in question has either appeared recently in the history of the discourse, or is prominent in the mind of the speaker. In the former case, the event is part of the context common to both participants. In the latter case, the event (hypothesized or real) is part of the speaker's (i.e., the client's) private context. Either way, the event is part of a context which can be made available to GNOMON by the client, and therefore communicable by the indirect route.
Alternately, if we choose the direct route, then the violated expectation must be signalled by part of the message. This can be done by introducing it as a modifier of an element already present, or as an element in its own right. If it is to modify something already present, it would seem natural (because of the analysis given by Hoepelman & Rohrer) for that thing to be the event time, something like: `(event-time time-1 :unexpected)`. Two problems with this arise: It is unclear how to specify whose expectations are being violated, should this be necessary later. It is unclear what modifiers of this sort would signify when applied to the message elements for speech time and reference time.

which treated events as if they were durationless. Neither of these works had any philosophical foundation.

One of the best-known proponents of interval-based temporal representations is James Allen (1984). Viewing intervals as primitive units in the system yields consistent results which may be easily expressed using one or more of thirteen primitive relations (`before`, `after`, `during`, `contains`, `overlaps`, `overlapped-by`, `meets`, `met-by`, `starts`, `started-by`, `finishes`, and `finished-by`).[47] Relations between two intervals may be expressed as any combination of these primitive relations; thus "*T on or before the 25th*" would be expressed as:

$$T \xrightarrow{\text{(or before during overlaps meets starts finishes finished-by)}} the\ 25^{th}$$

Networks are constructed from these intervals; constraint propagation is used to determine more closely how the intervals in the network are related. Later work (Allen and Kautz, 1985) required the introduction of world histories. Still missing from the system is the ability to handle recurrent behaviour, such as going to work on weekdays.[48]

One point-based representation for time is presented in McDermott (1982). This scheme treats time as dense, using closed, convex, point-based intervals and a branching-time model. McDermott's treatment of time is in a sense opposite to that of Allen: McDermott builds his representation around points, since he is interested in representing instantaneous states of affairs. Points are used to construct intervals, and intervals to construct events.[49]

Neither of these schemes would be able to represent adequately a sentence like: "*John went to the beach every Sunday for 3 years.*" For this, non-convex intervals are needed; other solutions are artificial or obscure. Ladkin (1986a, 1986b) presents a framework for the representation and manipulation of non-convex intervals. He extends Allen's relations on convex intervals to unions of convex intervals, and to arbitrary sets of points. He discusses the reasons for using both point-based and interval-based representations: A point-based representation is useful for describing system states, whereas intervals provide a more direct representation for many natural-language expressions (e.g. "*John walked for three hours.*"). In addition, intervals are used widely, have a defined duration, are extensible, and have a less ad hoc treatment of end points.

5.2 Formalisms for the Representation of Meaning

There are three points in this system at which meaning must be considered: input, processing, and output. In the first case we are asking about the client representation, in particular with respect to language. In the second case we are concerned with an efficient internal representation for processing. In the third case we are interested in ensuring that the text generated have the "appropriate" meaning.

We will assume nothing about the representation used by the client beyond the stipulation that the client define suitable functions to translate client objects into lexical items, and suitable predicates which we can apply to client objects to determine if certain constraints apply. It is unreasonable and counterproductive to assume that the client has any understanding of language

[47] Allen chooses to represent time intervals rather than time points, since using the latter can lead to paradoxical behaviour. Vilain (1982) extended Allen's work on temporal intervals to points, although it is not clear how points may be constructed from intervals, or vice versa, in Vilain's system.

[48] Hans Koomen, one of Allen's students, is currently working on this aspect.

[49] Since his intervals are point based, he is committed to taking a stand on whether they are closed or open on either side. This is a question he would rather leave unanswered, but assumes for present purposes that they are closed on both sides. This should give us some cause for concern, since it means that in the long run we will be faced with contradictions. McDermott considers this to be a minor problem, that "... *it doesn't seem very important for most events whether they include two extra instants or not.*" (McDermott, 1982, p. 110.)

beyond being able to name the things it works with. We can assume only that it knows about eventualities, objects, and modifiers (both of objects and of eventualities). Furthermore, since GNOMON is intended to be a general tool, it is unreasonable to impose on its clients a particular representation formalism.

For system-internal representations, I will take the point of view here that the four mechanisms used generally for encoding meaning (logic, frames, procedural semantics, and semantic nets (which includes conceptual dependency)) are ultimately capable of representing the same things, though each with a different degree of facility. Therefore the deciding factors in adopting one for internal processing will be ease of use, efficiency, and speed.

Semantic nets have functioned as the method for organizing information in several generation systems (Simmons & Slocum (1972), Goldman (1975), Mann & Moore (1981)). This sort of mechanism has proved to be unwieldy in my experience. Taking SNePS as an example, two major problems with its use for natural language generation are the apparent need for the network to represent the literal meaning to be conveyed before that meaning has been derived from input structures, and the fact that it is not compositional, a feature which is essential for the proper functioning of this system.

Winograd's SHRDLU (1973) uses procedural semantics to encode language knowledge for parsing. Both syntactic rules and word meanings are formulated as procedures which in turn generate programs to carry out some function, such as querying a database, or moving objects in a world of blocks. While this mechanism is useful for handling small problems, it is not always perspicuous, and should be used with care.[50] To some extent, GNOMON relies on procedural semantics by allowing the client to use predicates and functions to provide constraints and words.

Moore (1981) presents an example of the use of logic to represent meaning. His representation is guided by the nature of the mechanisms perceived to exist in natural language for expressing meaning, including eventualities, time and space, collective entities and substances, and propositional attitudes and modalities. As with other logics, his is compositional, for manageability. Notable differences from other logics include the use of generalised quantifiers, temporal relational operators (at and during), and tense operators (past and future).

Logic is very attractive as a representation for meaning, in part because of its wide-spread use in linguistics and philosophy (see section 5.5). There are difficulties with processing information encoded this way when fragments of the whole appear redundantly in different parts of the structure without being annotated as such (cf. especially the difficulties with Montague Semantics in dealing simultaneously with tense and temporal adverbials, noted in section 5.5). Nonetheless, logic serves as a useful framework within which constraints in GNOMON can be evaluated.

GNOMON has been influenced by all of the representation techniques mentioned here, and as such represents a synthesis of them, ultimately allowing data from different sources to cooperatively determine the content of an utterance. GNOMON's nets do represent a new way of dealing with parameters to the generation process, which is significant in and of itself.

Finally we may ask what GNOMON's utterances "mean." If GNOMON works correctly, the utterances should correspond to the "meaning" of GNOMON's hierarchy. By applying the appropriate logic operators to the net, forms in predicate calculus can be produced.

[50] The experience of compiler writers is quite relevant here. Parsers which operate by recursive descent and encode knowledge of the programming language in procedural terms are easy to write for simple languages, but can easily get out of hand for more complex ones.

5.3 Natural Language Generation

Work in generation has come a long way from its early beginnings, when generators either produced random sentences as tests for grammatical formalisms (ignoring higher-level issues), or produced meaningful text only for an extremely limited domain, with high level interactions worked out in advance by the programmer. Here I will only have room to discuss a few systems related to my own work in one way or another (see Mann et al, 1982, for a good overview of generation work up to the early part of this decade).

Simmons and Slocum (1972) carried out some of the first work on generating text to communicate encoded information. Their program traversed a semantic net using an augmented transition network. Tense, aspect, and other features of the sentence were given as input data, however. High-level concerns, such as the goals of the speaker, etc., were not considered.

Goldman (1975) built on Simmons' and Slocum's work to investigate the translation of a Conceptual Dependency representation into an utterance, focussing on lexical choice. His program BABEL made word choice via pattern matching with discrimination nets, selected on the basis of the primitive act involved. Relations for tense and aspect, among other things, were determined from the underlying CD, using an admittedly simple model for time (fixed points on a time line were associated with all events, states and state-changes).

In contrast to BABEL, GNOMON makes no assumptions about being given the precise meaning to be conveyed, but rather assumes that details can be arrived at via a dialogue with the client. Similarly, GNOMON assumes nothing about the representation of time used by the client, although it does assume that certain basic relationships between times may be determined. Finally, because of the way in which time is introduced to the sentence in BABEL, it would appear likely that temporal expressions would be repeated in a sentence, although this may change the meaning unintentionally. Needless to say, this is not the case in GNOMON, since far greater control over the introduction of temporal expressions is possible.

Appelt (1985) built on the work of Moore (1981) and Sacerdoti (1977) and others to investigate discourse as an integral part of human activity, i.e. discourse combined with non-verbal actions or substituted for them to achieve some goal. His program KAMP (Knowledge And Modalities Planner) is a hierarchical planner à la NOAH. KAMP uses STRIPS-like formalisms, called **action summaries** to aid in proposing plans, and then verifies the consistency of the plans with axiomatisations of linguistic (and other) knowledge, using a first-order logic theorem prover. Although one of the design goals of KAMP was to have a single process decide what to say and how to say it, the fact that the planning of actual utterances was considered trivial and relegated to a unification grammar indicates that KAMP failed in this respect.

Both GNOMON and KAMP are hierarchical, and both draw on linguistic and other knowledge to determine what is to be said. KAMP, however, concentrates on the content of the utterance, whereas GNOMON concentrates on the meaning. While KAMP uses its logic after the planning to verify the consistency of the plans, GNOMON maintains consistency throughout the generation of the utterance.

Like Appelt, Mann and Moore (1981) took the planning approach to generation in their system KDS, although the planning was not assumed to start as far back conceptually as in KAMP. Before KDS, data structures internal to an application were often designed specifically with text processing in mind, and relied on to include all the relevant material for a sentence, but no excess. To obviate the need for such reliance, Mann and Moore introduced the **fragment-and-compose** paradigm to generation, where the input data was **fragmented** into proto-sentences and then **composed** into sentences of the right form.

Mann (1983) implemented PENMAN, a program intended to generate high-quality domain-

independent text using a systemic grammar (Halliday, 1975). Its design calls for it to acquire relevant information from the environment (based on some goal) and iteratively plan text output, generating successive drafts of sentences, analysing them for possible improvements and replanning until no more substantial improvements can be achieved. Of the four modules in the system (acquisition, text planning, sentence generation, and improvement), only sentence generation is implemented. The generation module explores the use of **chooser** mechanisms in directing text generation on the basis of non-linguistic decisions. The choosers determine which features in the systemic grammar will be used (e.g., past, present or future tense) and may make their choices based on the results of formalized inquiries sent to the environment.

KDS was intended to make it easier for a program without linguistic expertise to arrange for text to be generated. PENMAN was intended to do this and to be portable and domain-independent as well. GNOMON is also intended to fulfil these goals. Like PENMAN, it queries the environment (in GNOMON's case the client) in non-linguistic terms. There the analogy breaks down, however, since the information acquired by choosers in PENMAN is not retained for future application of consistency checks, nor are any interactions specifically modelled, as they are in GNOMON.

McDonald (1983) describes a goal-directed incremental generator (MUMBLE) for which psychological validity is claimed.[51] The generator assumes that decisions about what to talk about, how much to say, the relative importance of any given item, as well as the overall organization of the material have been made prior to calling the generator, and are reflected in the *r*-specs received by the generator.[52] MUMBLE uses **description directed control**, a control technique in which the surface structure of an utterance is used to determine how to expand the *r*-specs into useful phrases, and when a phrase is complete. Both local and long-distance relations are allowed to influence choices, although the long-distance relations are few in number and completely under the control of the grammar writer.[53] Only forward dependencies are allowed.

MUMBLE and GNOMON place similar requirements on the client with regard to their input, although GNOMON takes away some of the burden of deciding how much to say, allowing the client to delay specification of some of the input until it is actually required. Like MUMBLE, GNOMON uses expressions very much like the surface structure to determine how to expand a given phrase. Unlike MUMBLE, however, GNOMON only allows local constraints to affect generated structure; long-distance relations are used only insofar as local constraints may be propagated some distance. This is at the same time both more restrictive and more flexible: grammar writers are only free to say how a structure can constrain a containing phrase, but they may arrange for that constraint to be propagated quite some distance in the structure. MUMBLE also only allows forward dependencies, whereas GNOMON places no such restriction on constraint propagation. Finally, GNOMON is designed to allow meanings to interact and in part to direct the generation, whereas MUMBLE has no such appreciation of word meaning.

Kay's FUG (1984) typifies work on unification grammar (see also Shieber (1986) for a more detailed introduction). He uses the formalism for translating text from one language to another. He emphasizes uniformity both in representation and in processing. A single frame-like notation is used

[51] The claim for psychological validity rests on four characteristics of the generator, said to have analogues in people: 1. text is produced incrementally, 2. decisions are indelible, 3. look-ahead is limited, and 4. grammaticality is enforced.

[52] At the time, McDonald was non-committal about lexical choice, purposely using only those terms with simple translations into words. The approach now taken is to assume that these, too, are determined external to the generator, i.e. the onus for lexical choice is on the program which calls the generator.

[53] Note that these are not global relations in the sense that they would always be available at all levels after having been established. For example, one long distance relationship is that of "current subject." Within a subordinate clause, the current subject is the subject of the subordinate clause, but on leaving the clause, the current subject of the containing clause is restored.

to describe both grammar rules and subcategorisation frames.[54] A single processing technique is used throughout, i.e. unification. One drawback to FUG is the exponential computational complexity.

According to the analysis in Shieber (1986), GNOMON and FUG belong to the same class of grammars, i.e., unification grammars. FUG, however, uses only simple features, and processes them with straight unification. GNOMON's constraints are more general than features, and do not have to be unifiable, but merely consistent. Furthermore, FUG's feature structures appear to retain their structure throughout the generation process, whereas GNOMON's constraints may be rearranged at any point.[55] Although GNOMON and FUG have the same computational complexity, this may prove to be less of a problem with GNOMON if judicious use is made of its filtering capabilities.

DIOGENES (Nirenburg et al, 1988) is the generation component of a machine translation system. It is a knowledge-based generator using blackboards as a processing paradigm. Its input includes an interlingua text. Data flows in a single direction, relative to the blackboards (from the input to the lexical selection to the syntactic blackboards). Lexical selection is emphasized, with selection based on collocation information and on weighted matches with the encodings of target language phrase meanings. This allows some flexibility in translating words in the source language which have no exact match in the target language. Control in DIOGENES is also handled with a blackboard, which is used very much like a queue.

There are three things worth special attention here: First, the fact that the system is part of a machine translation system is significant. As a result, the precision of the incoming information is limited by the sophistication of the parser, the flexibility of the interlingua, and by characteristics of the source language. Should DIOGENES need to be used outside of a machine translation context, the client which uses it will need to be able to produce interlingua text forms, which will in turn require significant linguistic knowledge on the client's part. In contrast, the input to GNOMON is dominated by structures native to the client program for which text is to be generated. Furthermore, a machine translation system cannot in general consult the author of the source text to determine if one translation is more appropriate than another, whereas GNOMON can direct questions at the client concerning the input data. Ultimately, the precision of GNOMON's input data depends on the sophistication of the client's model of its own data, and by the client's ability to reason with its data.

Second, the restrictions on collocation are based on words, which is too coarse a measure. While this records the phenomenon, it does not explain it. Furthermore, it results in large lexica, due to the potentially very large number of collocation entries. A better solution, implemented in GNOMON, is to note instead the characteristics of the words such that they do or do not appear together. This will allow us to come closer to an explanation, and potentially allow the lexicon to be much smaller.

Third, according to Stefik (1981), one major reason for taking a hierarchical approach is the ability it gives to defer decisions on detail until the last reasonable moment. While DIOGENES is in some sense hierarchical, it does not use the hierarchy to defer these decisions. Instead, it selects lexical items early and defers syntactic decisions. GNOMON takes the opposite tack, using the syntax as a hierarchy and deferring lexical selection until the choices are sufficiently constrained.

---

[54] A subcategorisation frame specifies the context in which a word may be used, and what arguments it may require. For example, "*gargle*" may be declared to be an intransitive verb requiring an animate subject.

[55] To give an example, the structures `[U: [V: W]]` and `[V: X]` may be inconsistent, though this is not detectable in FUG. GNOMON could arrange to propagate the constraints on `V` up to the point where the structures would have to be proven consistent to be allowed.

5.4 NLP and Time

Little work has been done on temporal expressions in natural language processing. Simmons and Slocum (1972) simply assumed that tense and aspect would be specified as input data to their text generator and Goldman's (1975) extension to their work did little to improve the situation. Bruce's CHRONOS (1972) accepted input in stylized English and answered questions about the relations between events. It used point-based intervals to represent time. Although CHRONOS could model the tense systems of Reichenbach (1947) and others, there were definite problems with it: The semantics were underdeveloped. Temporal adverbials were treated as an element of tense, thus "*He left*" and "*He left at six*" did not have the same tense.[56] Ungrammatical tense combinations were not accounted for. Modality was not well modelled.

Matthiessen (1984) presents an excellent catalogue of conditions under which various tenses may be chosen for generation. Like others whose work has been examined here, Matthiessen views tense and aspect as derivative from the relationships holding between members of a set of times. Matthiessen concentrates almost entirely on tense and aspect, only briefly discussing temporal adverbials. He indicates that adverbials may partially replace or supplement expressions of aspect, but does not give a satisfactory account of how the adverbials would be treated in the same framework. It would appear that Matthiessen at any rate does not take the rigid view taken by Bruce, that the introduction of an adverbial requires the introduction of a new element of tense, even when that relation is already reflected in the tense.

5.5 Linguistic Models of Temporal Expressions

Many linguists (Jespersen, Prior, Reichenbach, Montague, Bennett and Partee, and Johnson, to name a few) have suggested models for tense; some of these also considered the interaction of their model with temporal adverbials. Reichenbach (1947) describes tense as derivative from the relations between three temporal parameters: the time of the event (T_e), a reference time (T_r), and the time of speech (T_s).[57] His work forms the basis of much later work on tense and time, including that of Johnson (1981), who extends this work to a language with multiple past and future tenses. Montague (1974) presents a framework and grammar by which sentences may be constructed with entirely compositional meanings. Tense is modelled using Priorean tense operators; adverbials are modelled as the application of the function denoted by the adverbial to the denotation of the verb. The interaction between tense and temporal adverbials is not given any special consideration. Bennett and Partee (1978) point out some of the difficulties with Montague's model for tense, and discuss some of the interactions with temporal adverbials. Their focus, relative to temporal adverbials, is on adverbials of frequency.

Some linguists have concentrated their investigations on temporal adverbials. Hoepelman and Rohrer (1981) provide an analysis of the German adverbs *noch* (*still*) and *schon* (*already*). For example, their analysis of what they call "durative" *schon* corresponds closely to one English use of *already*, as in "*It's only ten o'clock, and already it's boiling hot outside!*" They note that its semantics depend on the violation or contradiction of an expectation of the speaker, expressing the meaning model-theoretically as:

$$
\begin{array}{lll}
[\![\text{schon}\phi]\!]^{M',i,j,g} = 1 \iff & & \\
\quad \exists x, y & (y < j < x \ \& \ [\![\phi]\!]^{M,i,y,g} = 0 & \\
\quad \& & \forall z(y < z \leq x \ \rightarrow \ [\![\phi]\!]^{M',e,z,g} = 0) & \\
\quad \& & \exists t(x < t \ \& \ \forall w(x < w < t \ \rightarrow \ [\![\phi]\!]^{M',e,w,g} = 1)) & \\
\quad \& & \exists s(j < s < x \ \& \ \forall u(y < u < s \ \rightarrow \ [\![\phi]\!]^{M,i,u,g} = 1))) &
\end{array}
$$

[56] Bruce uses tense in a technical sense: it is the composite of tense (in the traditional sense), order, duration, frequency, and aspect (technical sense: phase of an action — the beginning, middle, or end of an action).

[57] Also referred to as "time of utterance."

where M is the "real" model, M' is the model of the speaker, e is the speaker's world of expectations, i is the world in which schon ϕ is to be evaluated, j is the time at which schonϕ is to be evaluated, and g is an assignment function. Here, $[\phi]$ represents the denotation of ϕ with respect to the model. Loosely paraphrased, schon(ϕ) is true in the real world at time t iff ϕ is not true in the expectation world of the speaker/listener at time t, but is true in that world at time t', $t' > t$, and is true in the real world at time t, but is not true in the real world at time t'', $t'' < t$.
Note that this model makes no reference to tense.

The most significant work to date (for our purposes) is that done by Dowty (1979). In a sense, the work proposed here refines the definitions of adverbials in that work, and extends it to a computational framework. In particular, Dowty did not provide definitions for *at*, or *in*.[58]

Dowty goes to some trouble to exclude the expression "$*$*on today*" while allowing "*Thursday*" and "*on Thursday*". He eventually achieves this by making *today* be of one category (TmAV), and *Thursday* be of another (Tm), allowing Tm expressions to become TmAV expressions by the introduction of a preposition, such as *on*, or simply with no changes. Note that in order to achieve this, Dowty had to introduce two categories to his grammar, instead of one, solely for the purpose of restricting the applicability of a grammar rule. He does not go into the details of temporal prepositional phrases such as we are discussing, though he does note their complexity.[59] I believe that, had he included them in his grammar, he would have been forced either to introduce temporal prepositions via a rule,[60] or by introducing a new category, consisting solely of prepositions like *in*, *at*, and *on*.

Dowty discusses several possible mechanisms for composing tense and temporal adverbials:

1. Allow tense and temporal adverbials to be separate, with tense added to a sentential form after adverbials,
2. Allow tense and temporal adverbials to be separate, with adverbials added after tense,
3. Use double-indexing, and
4. Introduce them into the sentence at the same time (essentially pre-composed).

He notes that the first two schemes are unacceptable: tense cannot simply be made subordinate to temporal adverbials, nor vice versa, without misrepresenting the meaning of a text. The third scheme offers a possible way around this: double-indexing requires two indices, one representing the time of the event, and the other the time of speech; the temporal adverbial must be satisfied by one, and the tense must agree with the relation between the two. Dowty does not adopt this "*for simplicity's sake and because of certain problems.*"[61] The alternative he does adopt (the fourth mechanism) is to introduce tense and temporal adverbials in a single rule. This mechanism requires three variants of the same rule form, one for each tense. This route presents difficulties in the use of more than one or no adverbials.

In concluding this section, we should note that temporal expressions are as dependent on context as any other linguistic form. Thus, for example, a reference to June 25th, 1988 may only

[58] Dowty provides a definition for a different sense of *in*, but one which introduces a duration, or forward span of time (e.g., *in six weeks*). We require a definition which introduces a point or interval within a period of time (e.g., *in 1987*).

[59] In Dowty (1979:371n), he also notes the incompleteness of the solution: It allows "*on Thursday*" and excludes "$*$*on today*." If it also allows "*until Thursday*," then it must either allow both "*until today*" and "$*$*until on Thursday*," or disallow them both. Either solution is unacceptable.

[60] That is, within the body of the rule, rather like Montague's treatment of *he* and *him* (Montague, 1974:198n.). This is, of course, inconsistent with the treatment of other prepositions, which are drawn from the lexicon.

[61] Dowty (1979:329).

take the form *"today"* if the utterance is made at some time on June 25th, 1988. Although the use of context is not an important issue for this work, it does figure in some choices of phrasing, and must be addressed, however briefly. For present purposes we will merely assume that a context as discussed in Lewis (1972) provides an adequate model.

5.6 Constraint Propagation

Natural language generation involves loosely coupled subsystems. Stefik (1981) points out that constraint propagation is a good technique for solving problems involving such systems. Consequently, it is worthwhile to examine past work involving constraint propagation, in particular that by Stefik (1981) and by Steele (1980).

Stefik's (1981) MOLGEN is a system which combines hierarchical planning with constraint satisfaction to arrive at a powerful technique for solving problems involving loosely coupled subsystems. In MOLGEN, a plan is first generated in a goal-directed fashion, with many details purposely left out. This initial plan forms the basis for the hierarchy. Constraints (propositions which must be true for the plan to succeed) relevant to the missing details are then operated on to resolve the missing details. Stefik identifies three operations on constraints: **formulation, propagation,** and **satisfaction.** Constraints are formulated directly on the basis of a plan step. Constraints may be combined with the aid of a reasoning mechanism upon propagation up the levels of the hierarchy. Constraints are satisfied when variables appearing in them can be instantiated such that the proposition represented by the constraint is true. By using constraints on values, rather than using the possible values directly, Stefik can take advantage of a least commitment strategy, yet eliminate impossible plan instances.

There are interesting similarities and differences between GNOMON and MOLGEN. Natural language generation involves loosely coupled subsystems, as is the case for MOLGEN's domain. MOLGEN uses a hierarchy based on plans, while GNOMON bases its hierarchy on syntactic structures.

Steele (1980) discusses the design of a constraint-based programming language, a language in which constraints and constraint propagation have the status of a general language facility, like garbage collection in LISP. The paradigm is based on the concept of a *"program as a network of devices connected by wires."* Devices have "pins" to which the wires are connected. Constraints are arbitrary functions relating the values on these pins. The values may be variable, constant (e.g., either an input value or a constant in some equation), one of a set of allowed values, or have a default value. Changing the value on a pin 'awakens' a device, requiring it to recompute values for each of its pins, and then propagate the values to devices connected to those pins.[62] He discusses the changes needed to the system for the introduction of hierarchical devices, presenting one design which is essentially based on macros, and noting that a proper implementation (one allowing recursion) would require a design more like a procedure.

Steele does not have recursive 'devices', and works primarily with a static network, although it is possible to reconfigure the network in mid-computation and still have things work. In contrast to this, GNOMON has recursive 'devices', and a constantly changing network (that of a sentence). This is one indication of the fact that Steele's system and GNOMON are really solving different problems. Steele's system is finding values for parameters in a circuit, whereas GNOMON is designing the circuit.

[62] A more efficient implementation uses a number of queues, instead of awakening the devices directly.

6 Conclusions

We have explored the problem of generating temporal expressions and shown that it is amenable to solution by the application of AI techniques. We have proposed a specific solution based on hierarchical constraint propagation and outlined a course of experiments to determine specifics of the solution and to verify its feasibility. We have reviewed related work, and noted that it is generally consistent with the direction of this research.

In general terms, the benefits of the work proposed here include the proper handling of *in*, *at*, and *on*, the modelling of non-linguistic interactions within the grammar, a closer integration of common-sense data with 'meaning', a better understanding of the interface and underlying knowledge needed by client programs to take advantage of natural language generation programs, and a clearer understanding of the role of perspective on temporal expressions in particular and natural language generation in general.

In the above sections we have asserted that:

1. syntactic templates provide hierarchical structure for the generation process
2. constraints are propagated over this structure, possibly with transformations
3. constraints derive either directly or indirectly (via transformations) from data provided by the client. External knowledge sources are not required.
4. operations on the structure may be controlled by higher-order constraints
5. constraints arriving at any node must be consistent
6. only words within the client's vocabulary can be used.
7. word groups are selected by the client (or the interface, on its behalf)
8. government (as defined in linguistics) applies in the structure

As a consequence:

1. the simultaneous specification of tense and multiple temporal adverbials is enabled
2. global effects may be achieved only by propagating local constraints
3. the resulting sentence depends only on non-linguistic client data (which includes context and user models), words from the client vocabulary, and the language rules. No mechanism for ad hoc inclusion of language data is allowed.
4. multiple sources of constraints must be used to achieve lexical selection of a single item
5. Gricean maxims (and rules of this ilk) can be incorporated easily by an obvious utilisation of the existing control mechanism

Future research directions involve extending the grammar for more complete coverage of the language, as well as testing different scenarios which will provide more severe tests for the generation system.

A. Pseudocode for GNOMON

The following fragments of pseudocode represent a high-level description of the processing carried out by GNOMON. Invoke-gnomon is the function which the client calls to request text generation by GNOMON. It begins by selecting a root possible realisation and enqueuing an expand operation on this node. It then processes the queue, selecting operations in the order suggested by advice-giving rules. The operations are carried out by the functions expand, narrow, propagate, and filter. The syntactic component is then called on to actually perform the utterance. Finally, we note what was said, so that it can be used later to define the context. In the following, p-r is an abbreviation for possible-realisation.

```
invoke-gnomon(messages):  /* input messages from client */
begin
    p-r = new root p-r;
    enqueue expand operations on slots of root;
    until execution queue is empty:
      apply processing rules to choose next op;
      /* expand, propagate, instantiate, or filter */
      select
        expand:          expand(class,p-r);
        propagate:    propagate(p-r);
        narrow:          narrow(class,p-r);
        filter:          filter(filter,class,p-r);
      endselect;
    enduntil;
    forward r-specs to generator;
    record context changes;
end;
```

```
expand(class, p-r):  /* class of p-r to be expanded */
begin
    /* find p-r's and context specs for class */
    for each c-spec in class:
      if conditions for use are satisfied
      then
        add to queue of p-r's for that class;
        enqueue propagate and narrow operations on new nodes;
      endif;
    endfor;
    enqueue narrow operation on self;
    record constraints common to all p-r's
end;
```

```
propagate(p-r):              /* p-r which is source of constraints */
begin
    transform constraints as required by the c-spec template;
    find parents of p-r and attach transformed constraints to it;
    enqueue propagate operation on parent;
end;
```

Constraints are transformed by incorporating them in new forms. The following forms can be used to refer to the untransformed constraints and parts of constraints:

`:children`	all the constraints from the children, as-is
`(:parameter` *class*`)`	all constraints from the *class* child
`(:parameter` *class functor*`)`	the *functor* constraint of the *class* child
`(:parameter` *class functor arg*`)`	the *arg* argument of the *functor* constraint of the *class* child (*arg* may be a number or the keyword `:args`)

```
narrow(class,p-r):        /* class and p-r to be narrowed */
begin
    /* build sets of core constraints (must be satisfied) */
    /* and constraints to be tested (could be pruned) */
    for current constraints:
      if singleton constraint:
      then record as core constraint;
      else record as constraint to be tested;
      endif
    endfor
    for all constraints to be tested:
      if constraint is inconsistent with core:
      then remove p-r from which it comes;
      endif
    endfor
    if only one p-r remains:
    then instantiate it;
    else
       if some constraints were inconsistent:
       then /* maybe there's more to prune */
          narrow(class,p-r)
       endif
    endif
    if changes were made:
    then enqueue propagate operation on self;
    else enqueue filter operation on self;
    endif
end;
```

```
filter(filter,   /* type of filter to apply */
       class,    /* class of p-r to filter */
       p-r):
begin
    apply filter to p-r's in class of given p-r;
    if changed
    then enqueue narrow operation on this class;
    endif
end;
```

Bibliography

Allen 1983 Allen, James F., *Maintaining Knowledge about temporal intervals*, Communications of the ACM26(11), November 1983, pp. 832-843.

Allen 1984 Allen, James F., *Towards a general theory of action and time*, AI 23(2), July 1984, pp. 123-154.

Allen and Kautz 1985 Allen, James F., and Henry Kautz, *A Model of Naive Temporal Reasoning*, in Hobbs and Moore, Formal Theories of the Common-sense World, Ablex 1985, pp. 251-268.

Appelt 1985 Appelt, Douglas E., *Planning English Sentences*, Cambridge University Press, 1985.

Bach 1981 Bach, Emmon, *On Time, Tense, and Aspect: An Essay in English Metaphysics*, in Cole, P. (ed), Radical Pragmatics, Academic Press, 1981, pp. 63-81.

Bennett and Partee 1978 Bennett, Michael, and Barbara Hall Partee, *Toward the Logic of Tense and Aspect in English*, Indiana University Linguistics Club, Bloomington, Indiana, 1978.

Brachman and Levesque 1985 Brachman, Ronald, and Hector Levesque (eds.), *Readings in Knowledge Representation*, Kaufmann, 1985.

Bruce 1972 Bruce, Bertram C., *A model for temporal references and its application in a question answering program*, AI 3(1), 1972, pp. 1-25.

Cresswell 1985 Cresswell, M.J., *Interval Semantics and Logical Words*, ch. 3 of Cresswell, Adverbial Modification, Reidel, 1985, pp. 67-95.

Dowty 1979 Dowty, David, *Word Meaning and Montague Grammar*, Reidel, 1979.

Goldman 1975 Goldman, Neil M., *Sentence Paraphrasing from a Conceptual Base*, Communications of the ACM18(2), February 1975, pp. 96-106.

Grice 1975 Grice, H. Paul, *Logic and Conversation*, 1975, pp. 64-75.

Grosz et al. 1986 Grosz, Barbara, Karen Sparck Jones, and Bonnie L. Webber (eds.), *Readings in Natural Language Processing*, Kaufmann, 1986.

Halliday 1975 Halliday, Michael A.K., *Learning How to Mean*, Edward Arnold, 1975.

Hobbs 1985 Hobbs, Jerry R., *Granularity*, IJCAI-85, 1985, pp. 432-435.

Hoepelman and Rohrer 1981 Hoepelman, J., and Christian Rohrer, *Remarks on Noch and Schon in German*, in Tedeschi and Zaenen (eds.), 1981, pp. 103-126.

Johnson 1981 Johnson, Marion R., *A Unified Theory of Tense and Aspect*, in Tedeschi and Zaenen (eds.), 1981, pp. 145-176.

Kahn and Gorry 1977 Kahn, Kenneth, and Gorry, G.A., *Mechanizing Temporal Knowledge*, AI 9(2), 1977, pp. 87-108.

Kay 1984 Kay, Martin, *Functional Unification Grammar: A Formalism for Machine Translation*, COLING 84, pp. 75-78.

Ladkin 1986a Ladkin, Peter, *Primitives and Units for Time Specification*, AAAI-86, 1986, pp. 354-359.

Ladkin 1986b Ladkin, Peter, *Time Representation: A Taxonomy of Internal Relations*, AAAI-86, 1986, pp. 360-366.

Lewis 1972 Lewis, David, *General Semantics*, in Davidson and Harman (eds.), Semantics of Natural Language, pp. 169-218, Reidel, 1972.

Maida and Shapiro 1982 Maida, Anthony S., and Stuart C. Shapiro, *Intensional Concepts in Propositional Semantic Networks*, Cognitive Science 6(4), 1982, pp. 291-330, also in Brachman and Levesque (eds.), 1985169-189.

Mann 1983 Mann, William C., *An Overview of the Penman Text Generation System*, ISI/RR-83-114, April 1983

Mann and Moore 1981 Mann, William C., and James A. Moore, *Computer Generation of Multiparagraph English Text*, AJCL 7(1), January-March 1981, pp. 17-29.

Mann et al 1982 Mann, William, Madeleine Bates, Barbara Grosz, David D. McDonald, Kathleen R. McKeown, and William Swartout, *Text Generation: the State of the Art and Literature*, JACL 8(2), 1982.

Matthiessen 1984 Matthiessen, Christian, *Choosing Tense in English*, ISI/RR-84-143, November 1984.

McDermott 1982 McDermott, Drew, *A Temporal Logic for Reasoning about Actions and Plans*, Cognitive Science 6(2), April-June 1982, pp. 71-109.

McDonald 1983 McDonald, David D., *Description Directed Control*, Computers and Mathematics 9(1), 1983, pp. 111-130, also in Grosz, Sparck Jones, and Webber (eds.), 1986, pp. 519-537.

Montague 1974 Montague, Richard, *The Proper Treatment of Quantification in Ordinary English*, in Thomason (ed.), Formal Philosophy, Yale, 1974, pp. 247-270.

Moore 1981 Moore, Robert C., *Problems in Logical Form*, ACL-81, 1981, pp. 117-124, also in Grosz, Sparck Jones, and Webber (eds.), 1986285-292.

Nirenburg et al. 1988 Nirenburg, Sergei, Rita McCardell, Eric Nyberg, Philip Werner, Scott Huffman, Edward Kenschaft, and Irene Nirenburg, *DIOGENES-88*, CMU Technical Report CMU-CMT-88-107, June 11, 1988.

Pustejovsky 1987 Pustejovsky, James, *An Event Structure for Lexical Semantics*, TR Brandeis University Computer Science Department, 1987.

Quirk et al. 1985 Quirk, Randolph, Sidney Greenbaum, Geoffrey Leech, Jan Svartvik, *A Comprehensive Grammar of the English Language*, Longman, 1985.

Radford 1981 Radford, Andrew, *Transformational Syntax*, Cambridge University Press, 1981.

Reichenbach 1947 Reichenbach, Hans, *Elements of Symbolic Logic*, Macmillan 1947, republished by Dover, 1980.

Sacerdoti 1977 Sacerdoti, Earl D., *A Structure for Plans and Behavior*, Elsevier Scientific, 1977.

Schank 1975 Schank, Roger C., *Conceptual Information Processing*, American Elsevier, 1975.

Shapiro and Rapaport 1985 Shapiro, Stuart C., and William J. Rapaport, *SNePS Considered as a Fully Intensional Propositional Semantic Network*, SUNY at Buffalo, TR 85-15, October 1985.

Shieber 1986 Shieber, Stuart M., *An Introduction to Unification-Based Approaches to Grammar,* CSLI Lecture Notes Number 4, 1986.

Simmons and Slocum 1972 Simmons, R.F., and Jonathan Slocum, *Generating English Discourse from Semantic Networks,* Communications of the ACM15(10), October 1972, pp. 891-905.

Steele 1980 Steele, Guy L. Jr., *The Definition and Implementation of a Computer Programming Language Based on Constraints,* MIT Ph.D. Dissertation, AI-TR-595, Massachusetts Institute of Technology, 1980.

Stefik 1981 Stefik, Mark, *Planning with Constraints (MOLGEN: Part 1),* AI 16(2), 1981, pp. 111-140.

Tedeschi and Zaenen 1981 Tedeschi, Philip, and Annie Zaenen (eds.), *Tense and Aspect,* Syntax and Semantics, v. 14, Academic Press, 1981.

Vere 1983 Vere, Steven A., *Planning in Time: Windows and Durations for Activities and Goals,* IEEE Trans. on Pattern Analysis and Machine Intelligence 5(3), May 1983, pp. 246-267.

Vilain 1982 Vilain, Marc B., *A System for Reasoning about Time,* AAAI-82, 1982, pp. 197-201.

Winograd 1973 Winograd, Terry, *A Procedural Model of Language Understanding,* in Schank and Colby (eds.), Computer Models of Thought and Language, Freeman, 1973, pp. 152-186, also in Grosz, Sparck Jones, and Webber (eds.), 1986249-266.