Tempus:

A Task-Oriented System for Temporal Analysis and Reasoning

Thomas P. O'Hara

14 August 1992

A thesis submitted in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science in the Graduate School of
East Stroudsburg University

Contents

List of Figures

List of Tables

Acknowledgments

Three people have contributed much to the completion of this thesis:

Richard D. Amori
My thesis advisor, who provided valuable guidance

Cyril & Ann Marie O'Hara
My parents, who provided support throughout the years

Several others have made noteworthy contributions . . .

Allowing me to use *his* 486:
Al Semon

Assisting with MCAP and related matters:
John Peters

Clearing up programming problems:
Rich Scheller

Not revoking any of my network privileges:
Pete Zwetkof

Proofreading the document:
Don O'Hara Mike O'Hara Prof. Prince

Providing other support:
Rich O'Hara the members of Sigma Tau Gamma

To the memory of Eileen Harrity

Dedicated mother to Billy, Mark, and Brad

Dear friend to the Regulars

1 Introduction

1.1 A Simple Example

Temporal analysis and reasoning is common in many of our activities, even though we perhaps do not realize it. As an illustration, consider the following hypothetical work situation:

> O'Hara, get in here!
> I want you to analyze the Acme Rockets account.
> First, you will check out their accounts receivables.
> Then, when you calculate the assets and liabilities,
> evaluate the Los Alamos real estate market.
> Otherwise your figures might be way off.
> As usual, submit the report before 5pm on Friday.
> However, before you do any of this,
> get a shave--you look like hell!

Given this request, a human would have no trouble in determining what to do and when to do it (most likely using the tool shown in Figure 1).

In this scenario, temporal analysis would be used to determine the relations between the individual tasks, such as get-shave *precedes* analyze-Acme-account. Thus it is a combination of grammatical analysis and common-sense reasoning. Grammatical constructs are first analyzed to determine the basic relations between the events (e.g., evaluate-market *part-of* calculate-assets&liabilities). Then common-sense knowledge is used to infer the specific relations (e.g., evaluate-market *begins* calculate-assets&liability).

Next, temporal reasoning would be used to infer the relations among all the tasks and then to select an ordering that is consistent with the request. Thus it is simple deductive reasoning, followed by basic planning. For example, the following deduction is applicable:

check-accounts-receivables *precedes* calculate-assets&liabilities
calculate-assets&liabilities *started-by* evaluate-market
∴ check-accounts-receivables *precedes* evaluate-market

Similar deductions are applied until no more useful information can be derived (i.e., until all of the temporal constraints have been propagated). The result will be a partial ordering for the entire set of tasks, since the exact relation of the report preparation was left unspecified. Therefore planning would be necessary to determine when the report preparation should occur (e.g., it might be interleaved with the other tasks if O'Hara flies to Los Alamos for the evaluation).

The end result of the temporal analysis and reasoning is a list of tasks to be performed. Therefore, O'Hara's agenda might be as follows: get a shave, analyze the accounts receivable, begin report preparation, evaluate the Los Alamos real estate market, calculate the assets and liabilities, finish report preparation, and submit the report before Friday at 5pm.

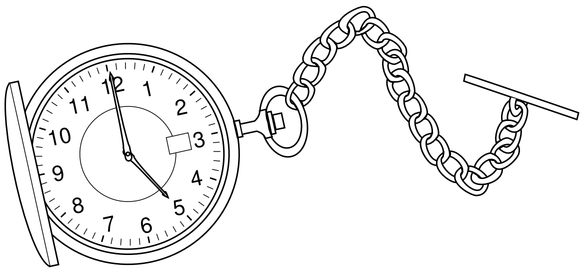


Figure 1. Traditional Temporal Reasoning Tool[1]

1 Rick Barker Studio Advertising Art

1.2 AI Perspective

Recently, temporal reasoning has received a great deal of attention in artificial intelligence (AI). James Allen developed a popular approach for reasoning about time intervals, using simple relations such as *before*, *during*, and *finishes* [Allen 83, 91]. For each pair of relations (or constraints) concerning a common interval, there is a deduction determining a relation between the other intervals. Allen's algorithm repeatedly applies these deductions until the constraints are propagated throughout the set of intervals. Johannes Koomen incorporated this approach in the TimeLogic system and later extended it for recurrent events with the Tempos system [Koomen 89]. Although these systems are research-oriented, the approach is sufficiently developed. Therefore, it is suitable for practical applications.

In natural language processing (NLP), temporal analysis refers to the extraction of the temporal relations from discourse and text. Verb tense has received the most attention, with much work following the philosopher Hans Reichenbach's method of distinguishing the reference time from event time [Reichenbach 47]. The event time is the time of occurrence, while the reference time indicates the perspective from which the event is viewed. In comparisons between the two times, it is common practice to use *ET* (or *E*) for the event time and *RT* (or *R*) for the reference time. For example, the past perfect tense references a point after the actual event, unlike the past tense:

1. The robot selected a new screw. $ET_1 = RT_1$
2. It had dropped the previous one. $ET_2 < RT_2$

In the first sentence, the *selection* event (ET_1) and its reference (RT_1) coincide. However, in the second sentence, the *dropping* event (ET_2) is referenced at a later time (RT_1).

Besides modeling tense, many researchers have used Reichenbach's analysis for sequencing events mentioned in discourse. If the reference times for two events are equivalent, then it is often possible to infer a specific relation between the event times. Since this condition holds in the example above ($RT_1 \approx RT_2$), the selection event is assumed to occur after the dropping event ($ET_1 > ET_2$). Fei Song [1991] used a similar analysis to monitor the temporal focus, following the work of Bonnie Webber [1987, 1988]. Also, Michael Brent [1989] determined constraints on the compatibility of tenses with various temporal connectives (subordinating conjunctions).

Various event structures (representations) have been proposed to facilitate the temporal analysis of events. For example, in the event structure developed by Marc Moens and Mark Steedman [1987, 1988], there are distinct constituents for the various phases of the event, such as the preparation for the action. This decomposition facilitates the interpretation of similar temporal connectives. Michael Almeida [1987] developed an alternative representation using semantic networks, which are graphs in which the nodes represent entities (objects of interest) and the links are relations between entities. He uses this method to provide a detailed treatment of temporal adverbials.

Also, several representations based on predicate calculus have been suggested, including the broad analysis by the philosopher Terence Parsons [1990]. Since his analysis is compatible with case theory as used in NLP, it is applicable to many approaches for temporal analysis. In fact, most of the event structures used in temporal analysis are compatible, especially since Reichenbachian tense structures are commonly incorporated.

1.3 ESU Perspective

At East Stroudsburg University (ESU), there is an ongoing project for the coordination of intelligent robots called the Multiple Cooperating Agents Project (**MCAP**) [Amori 88]. Currently there are several distinct components, including the recent **C-barp** system (Committee-based Architecture for Planning and Re-Planning [Resuta 91]), which uses a contract-based protocol for cooperative problem solving among autonomous agents. It also has a natural language interface called **TODUS** (Task-Oriented Discourse Understanding System [Percey & Taylor 86; Amori 90]). The tasks involved center around a workbench with positions for fasteners and holders for the tools (collectively termed the workbench cell). Since it is sufficiently constrained yet realistic, the domain provides a good environment for developing AI applications.

In order to extend the capabilities of the system, the **Tempus** system was developed for both temporal analysis and temporal reasoning. To accomplish this, the current work in temporal analysis was extended to handle task-oriented situations and constraints. In Tempus, more emphasis is placed on the interpretation of connectives and adverbials than on tense. To a lesser extent, there is an emphasis away from verbal aspect, which indicates an event's viewpoint. In addition, existing temporal reasoning approaches were extended to take advantage of constraints inherent in task-oriented problem-solving as well as the domain-specific constraints from the workbench environment.

For example, Tempus permits the following types of user utterances (a question mark is used to mark sentences that are grammatically questionable):

-- While holding the ruler,
   pound three nails into the first row of the blue bench.
   Then insert three in the second row.

-- ? Hammer a red for two minutes.
   ? Also screw a screw into the blue.

-- Fill the blue bench before 2pm.
   But first measure the bench

The system performs the same linguistic analysis performed by TODUS, such as the resolution of pronoun references, the handling of ellipsis, and the interpretation of ungrammatical constructs. In addition, for each of the user's task requests, Tempus determines its specific temporal information and its relation to other requests in the group. For example, in the first group above, the ruler-holding task temporally contains the nail-pounding task, both of which precede the screw-inserting task.

Before submitting the requests to the planner, Tempus needs to determine the temporal relationships among all of the requests, not just between pairs of tasks as usually specified by the user. Therefore it propagates each temporal constraint throughout a graph for the set of tasks. This propagation also serves to detect inconsistencies in the task requests, as in this set:

Screw in the screw before you hammer a nail	T_1 *precedes* T_2.
And, after hammering the nail, fill the red bench.	T_2 *precedes* T_3
But fill the bench before you screw the screw.	T_3 *precedes* T_1

The propagation algorithm will deduce T_1 *precedes* T_3 from the first two utterances. However, since this contradicts the third utterance, the specifications are inconsistent.

After all the utterances are processed, Tempus will generate a linear order for the planner requests consistent with the temporal constraints. Then, after submitting them to the planner, Tempus will give a description of the results to the user, as done by TODUS.

1.4 Goals for Research and Development

The following goals were established when the Tempus system was proposed. They motivated the research conducted for this approach to task-oriented temporal analysis and reasoning. The goals are still applicable, although the emphasis for some has been changed. During the discussion of the research and development, they will be reviewed.

- To extend temporal analysis to task-oriented domains, specifically the MCAP workbench cell.
- To provide a broad treatment of temporal adverbials for the specification of tasks.
- To develop domain-specific heuristics to facilitate temporal analysis.
- To determine a suitable representation of events for tasks in the domain.
- To extend temporal reasoning to the workbench domain with emphasis on restricting the possible relations among tasks (by task classifications).
- To investigate methods of expediting the search of temporal orderings.
- To determine if it is feasible to include hypothetical temporal reasoning as part of the temporal analysis.
- To explore the use of other mechanisms to increase the reasoning capability of the Tempus system.
- To show how this approach can be applied to other natural language processing applications, such as the interpretation of recipes.
- To conduct experiments to provide quantitative assessments of the performance of Tempus (e.g., scalability) in temporal analysis & reasoning.

1.5 Thesis Overview

This thesis describes the research into temporal analysis and reasoning for task-oriented domains. Specifically, it describes Tempus, ESU's solution to the problem, and its application to the MCAP workcell domain. After this introductory section, there is a lengthy background section that surveys the literature that influenced our approach. It serves mainly as a tutorial for the important concepts underlying the design of Tempus. It also describes the MCAP project and workcell domain.

Next, we discuss temporal analysis and reasoning for task-oriented dialogs. This is an important contribution since most other work deals with general discourse, such as narratives. The discussion leads to the description of the Tempus architecture. First, the important data structures are detailed. The system components are described next.

Then there is an analysis of the approach. This includes implementation and experimentation data that together show the effectiveness of the system. The following section places Tempus in perspective. Our approach is compared with that of others, and its strengths and weaknesses are presented. The section also discusses the research goals and provides suggestions for future work. The conclusion follows next, in which the thesis is summarized and the major contributions detailed.

In addition, the appendices cover detailed information on the implementation of Tempus. First, there is a collection of representative dialogs with TODUS. The system lexicon is listed next, illustrating the use of the lexical tags. Then there is an extended trace of several examples. Finally, a glossary is included that defines most of the terms used in the thesis, along with examples.

2 Background

2.1 Introduction

In this section, we will discuss the literature influential in the design of Tempus. First, the work based on James Allen's popular approach to temporal reasoning is described. Other approaches were not needed since his method proved sufficient for Tempus. This work is described first, since many subsequent approaches to temporal analysis use the interval relations that Allen incorporated. Most of the concepts in this section are described in detail since they are helpful for understanding the architecture of Tempus. Therefore the section is more like a tutorial than a survey.

2.2 Influential Work in Temporal Reasoning

2.2.1 James Allen

The recent advancement in temporal reasoning owes much to the work of James Allen, who developed the standard constraint propagation algorithm and also developed a temporal logic for AI [Allen 83, 84]. He has recently combined this related work into a general approach to temporal planning that removes many of the traditional difficulties [Allen 91].

Allen chose an interval-based representation for temporal relations over a point-based one, since intervals provide a better match for the requirements of natural language processing. Primarily, interval comparisons capture the relative nature of temporal specifications. In addition, interval comparisons are more intuitive. Figure 2 shows the thirteen relations that are basic to this representation. For each relation, the figure also indicates the symbol commonly used and the inverse relation. For instance, the inverse of *during* (*d*) is *contains* (*di*).

x Relation *y*		Inverse
After	>	<
Before	<	>
Contains	*di*	d
During	*d*	di
Equals	=	=
Finishes	*f*	fi
Finished-by	*fi*	f
Meets	*m*	mi
Met-by	*mi*	m
Overlaps	*o*	oi
Overlapped-by	*oi*	o
Starts	*s*	si
Started-by	*si*	s

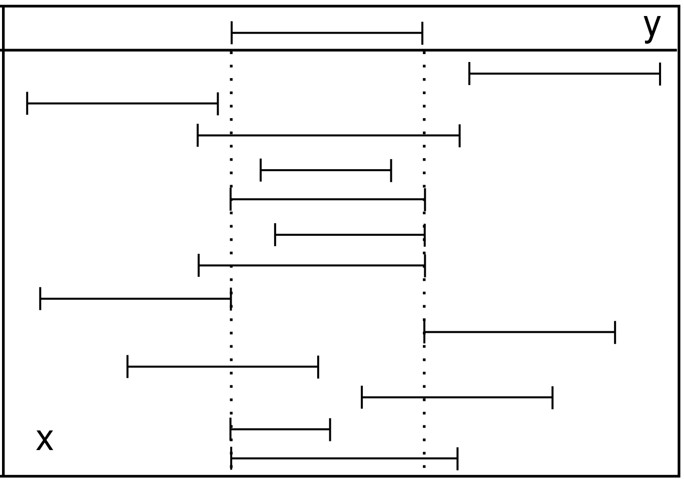


Figure 2. The Basic Interval Relations[1]

These basic relations can be combined to produce compound relations that allow for several possibilities. This is done by constructing a disjunction that specifies the alternatives, as in the definition of *includes*:

y includes x: x *during* y or x *starts* y or x *finishes* y or x *equals* y
equivalently: x {*during*, *starts, finish*es, *equals*} y
abbreviations: x *{d s f =}* y
$x \subseteq y$

For temporal reasoning using intervals, the problem is to determine the complete set of temporal relations for a group of intervals, given several of the pair-wise relations that exist. Allen's approach is based on the transitive closure of a relation in a graph, during which the transitive property of a relation is used to determine the relationships between a node and all other nodes that are accessible but are not directly connected. For instance, if a R b, b R c, and c R d, then the transitive closure will yield a R c, a R d, and b R d, as well. For the basic temporal interval relations, Table 1 lists the entire set of the transitivity properties. It has 169 entries since there is a transitive relation between any two of the thirteen basic relations.

[1] A modification to the version from [Koomen 89]

B r_2 C / A r_1 B	**<**	**>**	**d**	**di**	**o**	**oi**	**m**	**mi**	**s**	**si**	**f**	**fi**	**=**
before **<**	<	no info	< o m d s	<	<	< o m d s	<	< o m d s	<	<	< o m d d	<	<
after **>**	no info	>	> oi mi d f	>	> oi mi d f	>	> oi mi d f	>	> oi mi d f	>	>	>	>
during **d**	<	>	d	no info	< o m d s	> oi mi d f	<	>	d	> oi mi d f	d	< oi m d s	d
contains **di**	< o m di fi	> oi mi di si	o oi dur con =	di	o di fi	oi di si	o di fi	oi di si	o di fi	di	oi di si	di	di
overlaps **o**	<	> oi mi di si	o d s	< o m di fi	< o m	o oi dur con =	<	oi di si	o	o di fi	o d s	< o m	o
overlapped-by **oi**	< o m di fi	>	oi d f	> oi mi di si	o oi dur con =	> oi mi	o di fi	>	oi d f	> oi mi	oi	oi di si	oi
meets **m**	<	> oi di mi s	o d s	<	<	o d s	<	f fi =	m	m	o d s	<	m
met-by **mi**	< o m di fi	>	oi d f	>	oi d f	>	s si =	>	oi d f	>	mi	mi	mi
starts **s**	<	>	d	< o m di fi	< o m	oi d f	<	mi	s	s si =	d	< o m	s
started-by **si**	< o m di fi	>	oi d f	di	o di fi	oi	o di fi	mi	s si =	si	oi	di	si
finishes **f**	<	>	d	> oi mi di si	o d s	> oi mi	m	>	d	> oi mi	f	f fi =	f
finished-by **fi**	<	> oi mi di si	o d s	di	o	oi di si	m	oi di si	o	di	f fi =	fi	fi
equals **=**	<	>	d	di	o	oi	m	mi	s	si	f	fi	=

Table 1. Transitivity Table[2]

2 Taken from [Allen 83]. Note: dur = {d s f}, con = {di si fi}, and no info = {< > d di ... f fi =}.

The transitivity of compound relations (or their composition) is only complicated at first sight. The idea is that since only one of the basic relations in a compound actually holds (i.e., they are mutually exclusive), the composition must at least account for the transitive relation of the two basic relations that actually define the intervals. However, these two relations will not be known in advance and might not even be obtainable from the data. Therefore, the composition consists of the union of the transitive relations for all pairs of the basic relations, one from each of the compound relations. For example,

```
Composition({starts, during, finishes, equals}, {meets, overlaps}) =
   {before}                                       -- Trans(starts, meets)
   ∪ {before, overlaps, meets}                    -- Trans(starts, overlaps)
   ∪ {before}                                     -- Trans(during, meets)
   ∪ {before, overlaps, meets, during, starts}    -- Trans(during, overlaps)
   ∪ {meets}                                      -- Trans(finishes, meets)
   ∪ {during, overlaps, starts}                   -- Trans(finishes, overlaps)
   ∪ {meets}                                      -- Trans(equals, meets)
   ∪ {overlaps}                                   -- Trans(equals, overlaps)
= {before, during, meets, overlaps, starts}
```

In Allen's solution to the problem, each new relation is propagated throughout the graph of intervals using these transitive properties. Figure 3 shows the algorithm that he developed. The algorithm maintains a queue of relations to propagate. When an entry is removed from the queue, it is used to replace the existing network relation. Then the combination of this relation with any applicable relation is checked to see if the result is more restrictive than the corresponding network relation. If so the relation is added to the queue.

The test for applicable relations uses the Comparable function, which checks whether comparison between two nodes makes sense. For the basic propagation technique, this simply is true whenever the nodes are not equal. However, Allen realized it might be desirable to cluster the nodes for the intervals into related groups based on reference nodes; in this case, the Comparable function will be true only if the two nodes belong to the same cluster.

```
Composition (R1, R2)
    C ← ε;
    For each r1 in R1
        For each r2 in R2
            C ← C ∪ TransitivityTable(r1, r2);
    Return C;

Add (R(i, j))
    R(i, j) ← R(i, j) ∩ N(i, j)
    Add <i, j> to queue ToDo;
    while ToDo is not empty do
            Get next <i, j> from queue ToDo;
            N(i, j) ← R(i, j);
            For each node k such that Comparable(k, j) do
                R(k, j) ← N(k, j) ∩ Composition(N(k, i), R(i, j))
                If R(k, j) ⊂ N(k, j) then
                    add <k, j> to ToDo;
            For each node k such that Comparable(i, k) do
                R(i, k) ← N(i, k) ∩ Composition(R(i, j), N(j, k))
                If R(j, k) ⊂ N(j, k) then
                    add <j, k> to ToDo;
```

Figure 3. Allen's Constraint Propagation Algorithm[3]

For an illustration, consider the adaptation of Allen's example shown in the following figure. Nothing gets propagated after the first relation is asserted. But after the second, the transitive combination of the first two relations determines an additional relation (using Composition({oi mi}, {< > m mi}). The inverse of the first relation (arc 0-1) was required in this composition, which determines the deduction for arc 1-2 based on arcs 1-0 and 0-2. Thus the inverse of a compound relation is just the disjunction of the inverse of each constituent relation. The inverses are also used in the graphical representation of the resulting network, in order to show the natural interpretation for the example.

[3] Taken from [Allen 83] (including minor syntactic changes)

INPUT:

John was not in the room when I touched the light switch to turn it on.
However he was there later while the light went out.
Incidentally, I had deliberated somewhat before turning the switch.

TRACE:

Example of a simple task network (Allen[83])
ADD Relation 0-1: { o m }
0-1: { o m }

ADD Relation 0-2: { < > m mi }
0-1: { o m } 0-2: { < > m mi } 1-2: { < > di o m s si fi = }

ADD Relation 1-2: { d o s }
0-1: { o m } 0-2: { < m } 1-2: { o s }

ADD Relation 3-0: { d }
0-1: { o m } 0-2: { < m } 0-3: { di }
1-2: { o s } 1-3: { > di oi mi si } 2-3: { > }

Network is valid

RESULT:

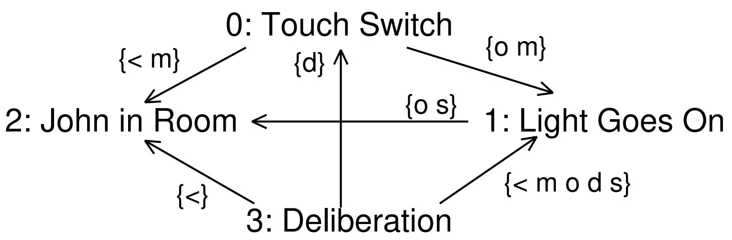


Figure 4. Example of Constraint Propagation

Allen estimated that the worst case performance of this algorithm is $O(n^2)$ for both time and space requirements[4]. This is based on the number of arcs in a fully-connected graph, since the algorithm updates each relation (or arc) at most a constant number of times. At first each relation is the uncertainty relation (the set of all the basic relations). Each time a relation is propagated, one or more of the basic relations will be removed from the network entry. Thus, at most thirteen updates are performed to each arc.

4 This is shown to be an underestimate in [Vilain and Kautz 86]. See the next section.

Allen also showed that this approach is sound, since it will not introduce invalid inferences. However, he admits that it is not complete since it only verifies local consistency. To detect global consistency, several relations have to be analyzed together (e.g., three at a time, four at a time, etc.). But this would lead to an exponential time complexity, which is unacceptable except for small inputs. Therefore, he makes the trade-off in favor of computational tractability, since the algorithm will produce acceptable results in most cases.

2.2.2 Marc Vilain, Henry Kautz, and Peter van Beek

After performing a finer-grain analysis on the performance of Allen's algorithm than the one indicated above, Marc Vilain and Henry Kautz [1986] present a case for using the point algebra as an alternative to the interval algebra, a formalization of Allen's method. They have two reasons for this preference: the interval approach is NP-hard[5], and the operations of the point algebra are much simpler than those for intervals.

They first assert that Allen's method actually has a time complexity of $O(n^3)$ instead of $O(n^2)$, since the number of interval operations is a better estimate than the number interval updates. For each entry taken from the queue, the algorithm performs O(n) interval operations: two transitivity computations and two intersections for each of the $n - 1$ other intervals in the graph. Since there will be $O(n^2)$ entries removed from the queue (i.e., the number of interval updates), the overall time complexity becomes $O(n^3)$. They next show that completeness for the interval algebra is NP-hard, which is not surprising, considering the exponential complexity of the consistent extension to Allen's algorithm.

The point algebra is based on the three relations <, =, and >. The same algorithm is applicable, with the substitution of the appropriate point operations. However, since there are only eight possible compound relations, the process of determining the transitive closure is much simplified--it could easily be implemented by a lookup table with 64 entries. In addition, they indicate that the algorithm is complete in this case, basically because the relations are more specific. Therefore applications using this approach would be more efficient than the interval-based approach, and the reasoning is con-

[5] That is, it is at least NP-complete, the set of problems for which the only known solutions are nondeterministic ones having polynomial complexity.

sistent. However, the completeness analysis was shown to be incorrect in the revised version of the paper, which includes related work by Peter van Beek [Vilain, Kautz & van Beek 89]. In addition, a new algorithm with complexity $O(n^4)$ is presented and proven to be complete.

However, they admit that there are some applications for which the interval algebra is still preferable. One case is the natural language processing of temporal specifications. There, the role of the temporal reasoner is more on recording and verifying the relations between tasks than on performing complicated inferences (although it still must infer implicit relations). In addition, the semantics of natural language is often too vague to be encoded as precise relations between points. For example, although the *starts&ends-before* relation between two intervals can be translated into relations concerning the end points, this is not the case for the *disjoint-from* relation:

x *starts&ends-before* y:
  x *{before, meets, overlaps}* y
  start(x) < start(y) & end(x) < end(y)

x *disjoint-from* y:
  x *{before, after}* y
  start(x) ? start(y) & end(x) ? end(y)

2.2.3 Johannes Koomen

The reference hierarchy that Allen suggested requires that the reference nodes be explicitly specified. However, Johannes Koomen [1989] shows that this is likely to lead to the construction of inappropriate hierarchies. Basically, it is difficult to predetermine the temporal structure for a series of intervals (especially for dynamic domains). Instead, he developed an automatic referencing algorithm that uses the relations asserted between intervals to determine the structure of the hierarchy.

The technique first computes the full closure for a set of relations, and then it removes relations that can be inferred using a path through reference nodes. This will sharply reduce the branching factor of most networks, which he claims tend to full connectivity when the basic constraint algorithm is applied. The algorithm subsumes the part of Allen's algorithm that actually adds the relation to the network. It also determines if the relation represents a special link and proceeds accordingly: for a relation $\mathsf{R} \subseteq \{\mathsf{s\ d\ f}\}$, it will attempt to assert a *reference link*; and for a relation across reference node boundaries (a *sidelink*), it will check if the link could safely be removed.

When a reference link is added, there are several cases to check. First, if the candidate reference node is already contained in the reference list, then nothing needs to be done. Or, if it is contained in the list of an ancestor for the node, the newly added relation can be deleted since it is implicit in the hierarchy. Otherwise, the candidate node is added to the reference list. In addition, the current node and its children are checked to ensure that inferable reference links are removed, as is done for the sidelinks.

When a sidelink is added, a check is made to see if the link can be inferred from a path through any of the from-node's reference nodes. If so, it will be deleted as long as both nodes for the sidelink do not share a reference node. This provision is to guard against the loss of information in the hierarchy.

When reference intervals are used, the query time is O(log n) and the assertion time is $O(\rho^2 n)$, with ρ the average size of a cluster divided by the maximum. Since ρ to expected be small and somewhat constant, this is a big improvement over the $O(n^3)$ approach without reference nodes. Unfortunately no assertion time complexity is provided for the automatic referencing technique, which requires path searching. But Koomen shows the number of compositions required with and without reference nodes for tests producing much constraint propagation.

For one of the tests, a 101 node network is created and random relations are assigned until each link has a singular relation, that is, a compound relation having only one basic relation (e.g., *{meets}*). To achieve the final state, a link is repeatedly chosen at random. If it is unassigned, it is replaced by a random singular relation. Or, if the previous relation is not singular, one of its basic relations will be chosen at random. Otherwise it is ignored since it is already singular. Then the new relation is propagated through the network. Furthermore, if an inconsistency is detected, the inconsistent link will be nullified. In line with the $O(n^3)$ time complexity of the algorithm, the following results were obtained:

Number of invocations to Composition (for 101 node test):

without reference nodes:	1,105,346
with automatic referencing:	475,585

Unfortunately, the run time of the test was not reported. However, the test does show the effectiveness of this automatic referencing technique.

2.3 Influential Work in Temporal Analysis

2.3.1 Hans Reichenbach

As mentioned earlier, much work in temporal analysis derives from the work of Hans Reichenbach [1947]. He developed an analysis for tense in natural language based on the relationship of three time points: event time (E), reference time (R), and speech time (S). The event and speech times correspond to the intuitive notions implied by the terms. But the reference time was introduced to make explicit the manner in which the event is viewed.

For instance, with the present perfect tense, an event's consequence is viewed:

The robot has dropped the fastener. [E < R = S]

In contrast, with the past tense, the event itself is viewed:

The robot dropped the fastener. [E = R < S]

The following table shows the tense structures for the six basic tenses, plus three nonstandard ones (which he calls posterior).

Tense	Tense Structure
Present	S = R = E
Past	E = R < S
Future	S < R = E
Present Perfect	E < S = R
Past Perfect	E < R < S
Future Perfect	S < R > E
Posterior Present	S = R < E
Posterior Past	S > R < E
Posterior Future	S < R < E

Table 2. Reichenbach's Tense Structures[6]

For example, a sentence in the simple present usually references the current time, which coincides with the event time:

The screw is in the left-hand corner. [S = R = E]

6 Adapted from [Reichenbach 47], which used a different notation for the comparisons.

In the past tense the reference time also coincides with the event time, but both precede the speech time:

The blue hammer was unavailable. [E = R < S]

This technique nicely handles the past perfect tense, in which the reference is to a time after the event but before the speech time:

The screw had been in the lower corner. [E < R < S]

The next example illustrates the posterior past, in which the reference time precedes both the event time and the speech time (which are unordered):

The robot was going to fasten the screw. [E > R < S]

Two specific relations underlie the tense structures [Passonneau 88]. First, the present, past, or future distinction is determined by the relation of the reference time to the speech time. For instance, all of the past tenses have the reference time prior to the speech time. Similarly, the simple, perfect, or posterior distinction is based on the relation of the reference time to the event time. For example, in the perfect tenses, the reference time follows the event time, whereas, in the posterior tenses, the reference time precedes the event time. This last correspondence is the motivation for treating the posterior tense constructions as having equal status as the six basic tenses, even though traditional English grammar is biased against this. In summary, these distinctions follow:

Time of occurrence:

present	past	future
R=S	R<S	R>S

Viewpoint of occurrence:

simple	perfect	posterior
R=E	R>E	R<E

2.3.2 Bonnie Webber

To facilitate the processing of temporal expressions, Bonnie Webber [1987] developed a mechanism to track the temporal focus of a dialog. Her system maintained a stack of focus points used to provide the context for the reference time. In addition, four heuristics for the management of the temporal focus were defined as shown in Figure 5 below. These are to be applied in parallel, and the most plausible interpretation will be accepted.

Focus Maintenance Heuristic:
$TF_{new} \leftarrow TF_{current}$

Embedded Discourse Heuristic #1:
If $E_{current}$ <> $R_{current}$ then
push(TF_Stack, $TF_{current}$)
$TF_{new} \leftarrow E_{current}$

Embedded Discourse Heuristic #2:
if begins_embedded_narrative($clause_{next}$) then
push(TF_Stack, $TF_{current}$)
$TF_{new} \leftarrow E_{next}$

Focus Resumption Heuristic:
if ends_embedded_narrative($clause_{current}$) then
$TF_{new} \leftarrow$ pop(TF_Stack)

Figure 5. Temporal Focus Maintenance Heuristics[7]

The focus maintenance heuristic simply retains the current focus, whereas the resumption heuristic resumes a previous one, which was stacked using one of the embedded discourse heuristics. The main difference between the two embedded heuristics is that the first is keyed on the tense structure of the current clause, but the second uses other linguistics clues from the next clause.

7 These are Pascal versions of the descriptions presented in [Webber 87].

She illustrates the heuristics with the following example:

1. John went over to Mary's house [E = R < S]
2. On the way, he had stopped by the flower shop for some roses. [E < R < S]
3. He picked out 5 red ones, 3 white ones and one pale pink. [E = R < S]
4. Unfortunately the roses failed to cheer her up [E = R < S]

At the beginning of the analysis, a new focus structure is created. Then after sentence 1, the focus maintenance heuristic retains the same focus. Next after sentence 2, the first embedded discourse heuristic is applied. This stacks the focus (E1), before replacing it with E2. Therefore the picking event is interpreted in relation to the flower shop event. Finally, since the last sentence does not make sense with regard to the florist, the focus resumption heuristic has to be applied. This pops the previous focus (E1) and uses it to order the last clause.

Now consider replacing the second sentence with

2′. But first he stopped by the flower shop.

In this case, there is no tense shift to provide a clue. So the second embedded discourse would be used instead of the first, although with the same effect. Note that the two heuristics could be merged, since tense shift is just another linguistic clue suggesting an embedded discourse. However, Webber doesn't make note of this possibility.

2.3.3 Fie Song

Fei Song [1991] developed a tense hierarchy to guide the focus monitoring using an efficient deterministic algorithm. Webber's version, which depends on the nondeterministic evaluation of the four heuristics, could lead to inefficient backtracking. Furthermore, he extended the temporal focus structure to include the entire tense structure, not just the event time. Therefore the repetition of a nonsimple tense structure can be treated as focus maintenance.

The tense hierarchy (see Figure 6) defines the allowable sequences of tenses, starting from the simple present. Note that the prospective tenses correspond to Reichenbach's posterior ones. During discourse analysis, tense transitions are considered coherent if the hierarchy contains a link to the new tense structure from the focused tense structure or from one of the stacked structures. Focus creation is suggested by the thin lines in the diagram, whereas focus maintenance is suggested by the thick ones (as does the repetition of the current tense).

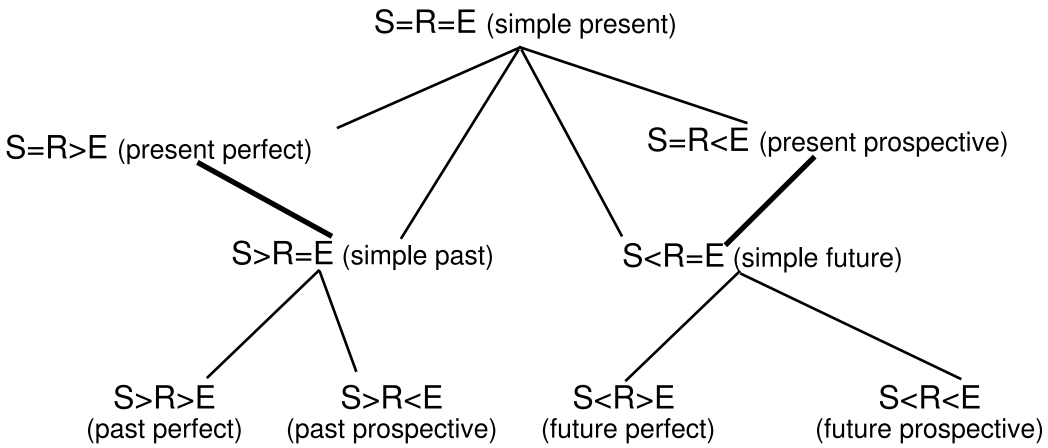


Figure 6. Song's Tense Hierarchy[8]

[8] Adapted from [Song 91]

The following example shows how this approach helps in the case when there are successive occurrences of a tense in which the reference time and event time differ (i.e., the repetition of a nonsimple tense):

1. John went to a hospital. $[S_1 > R_1 = E_1]$
2. He had fallen on a patch of ice, $[S_2 > R_2 > E_2]$
3. and had twisted his ankle. $[S_3 > R_3 > E_3]$

With Webber's approach, R_3 will be interpreted against the temporal focus of E_2, and the third clause is treated as beginning a second embedded discourse. Instead it should be treated as $R_3 = R_2$ with the clause continuing the current embedded discourse.

With Song's method, the analysis starts by putting the default tense structure $[S_0 = R_0 = E_0]$ on the focus stack. When the first clause is analyzed, a new focus structure is created and placed on the stack since there is a creation link (thin line) in the hierarchy from the present tense to the past. Likewise, the second clause causes a new focus structure to be stacked (because of the creation link from the past to the past perfect). However, the third clause matches the focus structure, which therefore is simply maintained.

As an example of the use of the maintenance links, consider a variation of the florist trip example:

1. John has come back from a florist shop. $[S_1 = R_1 > E_1]$
2. He picked out three roses, ... $[S_2 > R_2 = E_2]$

When clause 2 is encountered, the top of the focus stack has the structure for the present perfect. Since the link from the present perfect to the past indicates maintenance, the current focus is retained. Also, the relation $E_2 \le E_1$ is inferred (E_1 is part of the temporal focus). This models the use of the present perfect tense to introduce a topic and then the past tense to elaborate on the details.

2.3.4 Marc Moens and Mark Steedman

Various representations for event structure have been proposed. Moens and Steedman [1988] have shown that it is useful to decompose an event into components representing the preparatory phase, the event's culmination, and the consequent state:

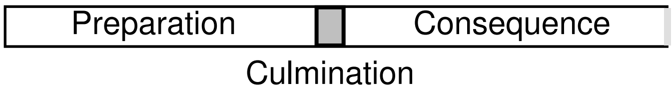


Figure 7. Nucleus Event Structure[9]

The culmination is the event that marks the transition from the preparation to the consequence. Although normally instantaneous, the culmination can be decomposed using the nucleus structure. Likewise, the other components are usually atomic but can be decomposed. This structure facilitates the modeling of verbal aspect and the handling of similar adverbs, like *when* and *while*.

Moens and Steedman hold that *when* can only appropriately be used to express a contingent relationship between two clauses, such as causality or enablement, not just temporal coincidence as with *while*. To show this, they give an interpretation of *when* in terms of the nucleus event structure. The adjunct (subordinate) clause defines a core event for interpreting the expression. Once the core event has been identified, either it will be decomposed into the three constituents above, or it will be treated as the culmination in some broader event, requiring the addition of preparatory and consequent components. Both cases require contextual information and world knowledge to provide clues and missing information. For example, consider

[9] Adapted from [Moens & Steedman 88].

When they built the 39th Street bridge . . .
a. . . . a local architect drew up the plans.
b. . . . they used the best materials.
c. . . . they solved most of the traffic problems.

In (b), the adjunct is decomposed as <building; complete building; have completed> and the main clause is associated with the preparatory process of building. However, in both (a) and (c), the adjunct is treated as the culmination in <preparing to build; build; have built>: in (a) the main clause is associated with the preparation, but in (c) the main clause is associated with the consequence.

The use of *when* in these examples is considered appropriate since there is a definite contingent relationship between the two clauses. However, this does not hold for the following:

When my car broke down, the sun set.

This sentence is considered inappropriate since there is no real relationship between the two events. In terms of the nucleus structure, it is hard to associate the sunset event with either the preparation for or the consequence of the breakdown event. Thus a connective not involving contingency (e.g., *just after*, *just before*, or *at approximately the same time that*) should be used.

Furthermore, they developed a classification for events that is becoming popular. In addition to the usual *state* and *process* classes, there are three new classes: a *culminated process* is a process with a specific goal; a *culmination* is similar except that the transition is emphasized; and a *point* is an event viewed as simultaneous:

state	the bench is full
process	hold the ruler
culminated process	screw a screw
culmination	fasten a screw into the middle slot
point	drop the screw

Their classification is used throughout this thesis. For the sake of clarity, the discussion of related work reflects this classification.

2.3.5 Michael Almeida

An alternative representation for events is given by Michael Almeida [1987] using semantic networks. His approach is compatible with those following Reichenbach and also incorporates Allen's interval relations (except *overlaps*) in the representation. He provides an in-depth analysis of temporal adverbials and connectives in the context of narrative discourse, including the interaction of these constructs with tense and aspect. Much of this will carry over to other discourse areas, except for some of the heuristics that are applicable only to narratives.

For instance, Figure 8 illustrates his representation of two temporally-related events. The event descriptions are represented by the circle-triangle combination (for brevity), and the event time is given by the node immediately above (indicated by the time link). Since both events are culminations, the final subinterval of each is being referenced. This is represented by the reference-time node being in the final-sub-interval relation to the event-time node (which is therefore super-interval to the other). Furthermore, the adjunct's reference time is asserted to be just before the main clause's reference time, since the relation is before and has a very short duration (epsilon or zero). Note that the narrative time line is modeled by the change in the points labeled now_1 and now_2. Thus a simple focus mechanism is supported.

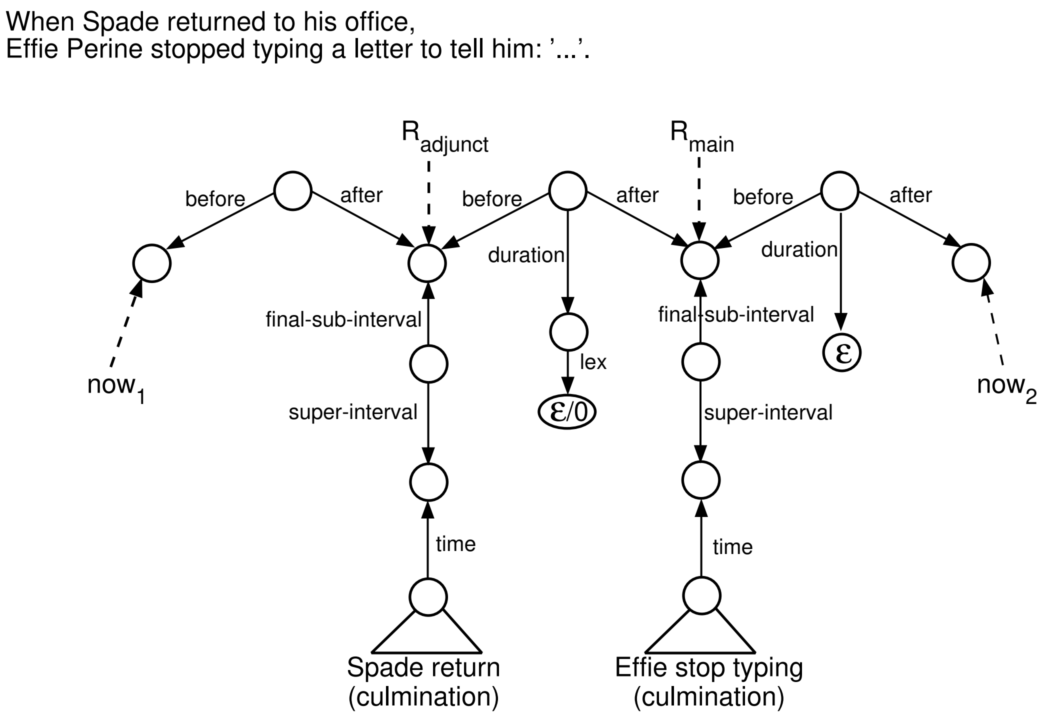


Figure 8. Semantic Network Event Representation[10]

This figure is also an example of the first of four cases in his analysis of *when*. Since both clauses are perfective, the reference time of the main clause is asserted to be shortly (or immediately) after the reference time of the adjunct. This covers the natural interpretation that Spade's return causes Effie to stop typing, in line with the interpretation using the nucleus event structure. However, there is no restriction that the two clauses are in a contingent relationship, as Moens and Steedman require.

The four cases for the interpretation of *when* follow (based on the aspects of the main and adjunct clauses):

perfective/perfective:	R_{main} {> mi} $R_{adjunct}$
perfective/imperfective	$R_{main} \subseteq E_{adjunct}$
imperfective/perfective:	$E_{main} \supseteq E_{adjunct}$
imperfective/imperfective:	R_{main} {=} $R_{adjunct}$

10 Adapted from [Almeida 87]

When one of the clauses is imperfective, while the other is perfective, the imperfective event contains the other (or they are equal). The only other case is when both clauses are imperfective. In general, the reference times are equivalent, and any subpart of adjunct event can be referenced. But, for narratives specifically, the start of the adjunct event is being referenced.

The interpretation of *while* is much simpler. Regardless of the aspect of either clause, the reference times of both are equivalent. Almeida says this is preferable over the standard analysis of *while*, which has the adjunct event during the main event, since in some cases the main event is being stretched out to fill the time of the adjunct, as in the following:

He walked up and down the office while the girl used the telephone.

This analysis then makes most of the cases of *while* subsumed by those for *when*. Therefore his treatment of both connectives is less specific than that of Moens and Steedman, although it is also less restrictive.

Almeida also presents a detailed analysis of temporal adverbials, as outlined in Table 3. Most of these are frame adverbials, which bound the temporal location of events. There are two basic types: interval frame adverbials for relatively large intervals of time and point frame adverbials for relatively small ones. For the non-narrative adverbials, the classification is based on the relation of the interval to the speech time: fixed present adverbials contain it, fixed past ones are prior to it, fixed future ones are after it, but variable times have no restriction. Also, a distinction is made between covering intervals, which have no gaps between successive intervals, such as a week, and non-covering ones such as Tuesday.

Narrative Frame Adverbials		
interval frame adverbials		
	(on/in) $NP_{interval}$	on the weekend
	(on) $N_{day\text{-}of\text{-}week}$	Wednesday
	(in) the $N_{day\text{-}sub\text{-}interval}$	in the morning
	(on) the following $N_{interval}$	the following evening
	(on) the previous $N_{interval}$	on the previous afternoon
	that $N_{interval}$	that night
	(the) next $N_{interval}$	next week
point frame adverbials		
	at $NP_{point\text{-}of\text{-}time}$	at half past five
Non-narrative Frame Adverbials		
fixed present		
	today	today
	this $N_{covering\text{-}interval}$	this week
fixed past		
	yesterday	yesterday
	last N	last night
	the other day	the other day
fixed future		
	tomorrow	tomorrow
	next N	next year
variable time		
	this $N_{non\text{-}covering\text{-}interval}$	this Tuesday
Durational Adverbials		
for		for ten minutes
Relational Adverbials		
until		until three o'clock
then		then ...
before		before seven
after		after tomorrow
later		fifteen minutes later
Temporal Connectives		
before		... before ...
after		... after ...
until		... until ...
when		... when ...
while		... while ...

Table 3. Temporal Adverbial Categories[11]

11 This is a compilation of the categories presented in [Almeida 87], with the exception that the category TemporalConnectives is introduced for clarity. Almeida had *before*, *after*, and *until* only under RelationAdverbials, and *when* and *while* under MiscellaneousAdverbials.

2.3.6 Michael Brent

Michael Brent [1989, 1990] developed an attractive approach for analyzing the compatibility of temporal connectives with various tenses. First, the reference and speech times of the tense structures for the two clauses occurring with the connective are aligned to determine the relationship of the event points. If no relationship is determinable then the tense combination is considered invalid. Otherwise, three possibilities by the columns in Table 4. This table classifies the valid combinations using the relationships between the event times of the clauses.

	$E_{mat} < E_{adj}$	$E_{adj} < E_{mat}$	$E_{adj} = E_{mat}$
matrix adjunct	past perfect past	past past perfect	past past
matrix adjunct	present perfect present	present present perfect	present present
matrix adjunct	future perfect present	future present perfect	future present

Table 4. Legal Tense Combinations[12]

Afterwards the connective is checked to see if it is compatible with the determined relationship, as in the following examples:

	$E_{mat} < E_{adj}$	$E_{adj} < E_{mat}$	$E_{adj} = E_{mat}$
before			X
by the time	X	X	X
once		X	X
when			X

However, for this approach to be applicable, the interpretation of $E_{adj} = E_{mat}$ is not just simple temporal coincidence: it can indicate either causality or that no relevant intervening events occur. Anyway, the constraints he presents are generally stricter than common usage.

[12] Taken from [Brent 89].

2.3.7 Rebecca Passonneau et al. (Unisys)

At Unisys, several researchers have developed a general text processing system, called PUNDIT, that includes support for temporal analysis [Passonneau 87, 88; Dahl et al. 87]. The temporal analysis component consists of three modules. The first filters out utterances that don't refer to an actual time (e.g., an expression that indicates a habitual action). The second determines the event structure for each distinct event referenced. The last one produces a Reichenbachian tense analysis.

A novel feature of the system is the handling of nominalizations. These will be temporally processed in three contexts: as the prepositional object of a temporal connective ("after engagement"), as an argument to a situation-describing verb ("failure occurred"), or as an argument to a verb that is semantically distinct ("investigation revealed contamination"). Since the nominalizations lack tense, the corresponding information will be derived from the governing constituent, such as the verb in the last two cases above.

Situations are classified as states, processes, and transition events. The last class covers both of the culmination and culminated process classes from Moens and Steedman's classification. Next, there are two features that describe the intervals associated with the situations. Kinesis refers to the internal structure of the interval; stative intervals do not involve significant change, unlike active intervals. Situations having stative kinesis cannot be modified by rate adverbials:

? The pressure was quickly low.	[stative kinesis]
The pressure quickly dropped.	[active kinesis]

Boundedness refers to the relation of the interval to other intervals (including the moment for the event time). An unbounded interval extends indefinitely on both sides of the event. But a bounded interval meets a subsequent interval at the event time. Also, an unspecified interval can contain the event time within it or at either end point.

Incidentally, the structure of a bounded interval matches the nucleus event structure developed by Moens and Steedman. This can be seen from the representation for "The pump failed":

```
Lexical Aspect = transition event;   Progressive = no
Situation1
    event(fail1, become(inoperative(patient(pump1))), moment(fail1))
Situation2:
    event(fail2, inoperative(patient(pump1)), period(fail2))
```

The first situation is the transition event from the state of being operative, corresponding to the culmination component of the nucleus. The second is the resulting state of being inoperative, corresponding to the consequence. Furthermore, the first situation has active kinesis, while the second is stative.

The analysis of temporal connectives is based entirely on the relation of the reference times for the two clauses. Therefore, for *before*, the reference time of the main clause is asserted prior than that of the adjunct clause. The treatment of *after* is analogous: the main reference time is asserted after the adjunct reference time. Also, for *when* the reference times are equated. Although this approach is not as comprehensive as Almeida's, it does produce intuitive results for the different types of temporal structures:

The pump failed when the engine was rotating.
    transition of failure *during* period of rotation

The pump failed when the engine rotated.
    transition of failure *during or at one endpoint of* period of rotation

The engine rotated when the pump failed.
    same as the previous

2.3.8 Other Influences

In his book *Events In The Semantics Of English*, Terence Parsons [1990] presents support for his subatomic semantics, a predicate logic approach using quantification over events and times. Several issues related to temporal analysis are discussed in detail, including event classification, tense, and temporal adverbials. The following example illustrates his representation for events:

Yesterday at noon, Brutus stabbed Caesar

$$
\begin{aligned}
&(\exists I)\ [I < \mathrm{now}\ \ \&\ \ I \subset \mathrm{Yesterday}\ \ \&\ \ I \subset \mathrm{Noons} \\
&\quad \&\ \ (\exists e)(\exists t)[t \in I\ \ \&\ \ \mathrm{Stabbing}(e)\ \ \&\ \ \mathrm{Agent}(e,\mathrm{Brutus}) \\
&\qquad\qquad\qquad\qquad \&\ \ \mathrm{Theme}(e,\mathrm{Caesar})\ \ \&\ \ \mathrm{Culminates}(e,t)]]
\end{aligned}
$$

In the expression, the last term indicates that the event culminates at the given time, which is in an interval prior to the current time and is also contained in the sets Yesterday and Noons. This integration of the various sources of temporal information makes the representation compact and intuitive.

For the domain of cooking, Robin Karlin [1988] has developed a natural language interface to an animated simulation. Since she found that verbal modifiers are essential for the expressive power of recipes, she places an emphasis on their interpretation, in particular on the temporal modifiers for repetition and duration. Repetition is mainly specified by adverbials giving cardinal counts or frequency. In addition, it is indicated by plural nouns if the action normally cannot be done at once. Duration can be given by any of the following: explicit time units, state change, gradable terms, or co-extension with another event. The implementation uses a frame-based representation and includes domain knowledge about the cooking tasks, including default values for the actions modeled.

The grammar of the English language is a complex topic to master, even for native speakers of the language. Therefore, grammar books of various sorts are available. However, one of the most scholarly of these is the encyclopedic *A Comprehensive Grammar of the English Language* by Randolph Quirk, Sidney Greenbaum, Geoffrey Leech and Jan Svartvik, whose research spanned twenty years [Quirk et al. 85]. This work was consulted frequently in order to clarify the concepts required for the temporal analysis in Tempus.

2.4 Multiple Cooperating Agents Project (MCAP)

2.4.1 Overview

At East Stroudsburg University, MCAP is a long-term project centered around the coordination of four intelligent robots, currently using children's workbenches for the tasks. The layout of a workbench follows:

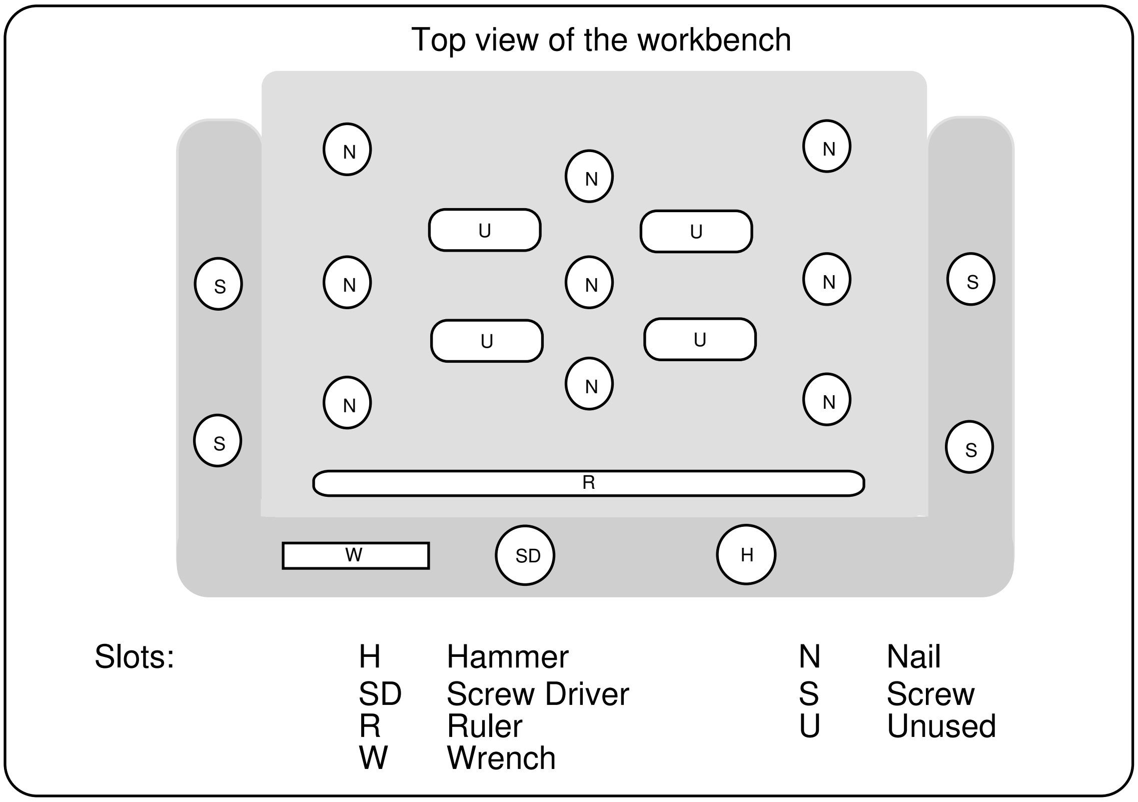


Figure 9. MCAP Workbench

Note that although screws could fit into the nail slots, the robot manager does not allow for this action since it makes little sense. Other cases of mis-match are also treated as illegal. Thus the workbench is treated as a manu-facturing environment in which strict differentiation is required.

There is a robot arm on each side of the workbench area, and at each corner there is a work area accessible only by the two nearest robots. This often makes robot coordination a significant part of the solution to a user's task request. In fact, much of the research that went into MCAP has been in the area of planning for multiple agents.

Figure 10 shows the overall architecture for MCAP. There are five main components: the Central Controller initializes the system and then coordinates the component communication; the Robot Manager governs the actual robot operation; FUSE performs sensory fusion to maintain the world model; C-barp plans the tasks; and TODUS analyzes the user's the natural language requests. (The CP components are the control packages required for low-level operations.) Of the main components, TODUS and C-barp will be considered in more depth. TODUS defines the basic discourse analysis in Tempus; C-barp determines the high-level robot actions available.

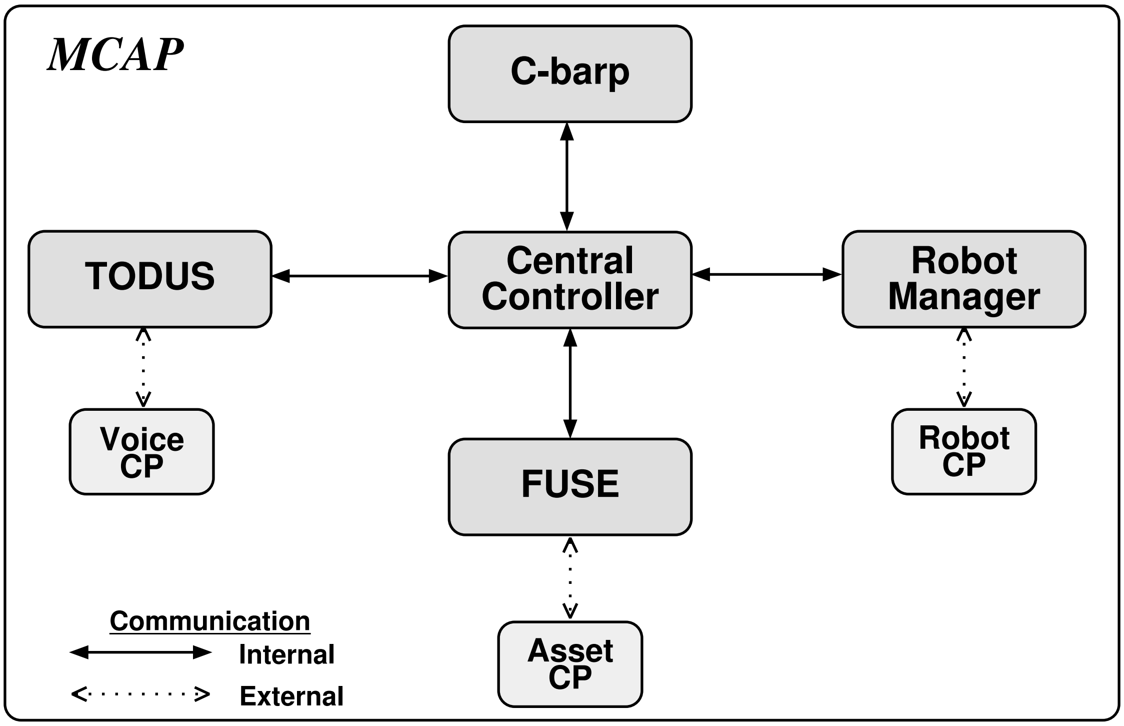


Figure 10. MCAP Architecture[13]

[13] Adapted from [Peters 1989]

2.4.2 TODUS

To direct the robots to perform a task, the user enters a series of natural language requests to TODUS, which determines if the utterances are consistent with the state of the workbench. The following examples illustrate the form of acceptable requests (see Appendix A for samples of complete dialogs):

> Use the red hammer to insert a green nail into the blue bench. With the blue screwdriver, screw a green screw. Into the red bench. Upper right corner.
>
> Hammer a green nail into the red bench. Now another one into the blue one.
>
> Hammer a red. Screw a screw into the blue.
>
> Hammer a nail into either bench. Hammer a nail into the red bench, anywhere. Screw a screw somewhere in the upper row of the blue bench.
>
> Fill the second column of the red bench with nails.
>
> Is the green nail in the blue bench? Is there a red screw in the bench?
>
> Where is the blue screw?
>
> How many nails are there in the blue bench? How many in the red one?

TODUS does not reject ungrammatical sentences. Instead it just prints warning messages. But it does reject sentences that do not make sense or that are unintelligible. In fact, since the domain is well-constrained, TODUS is adept at determining the user's requests, handling a variety of linguistic constructs (e.g., ones using ellipsis or containing pronominal references).

It supports two types of task requests: simple and composite, which imply several simple ones. It also allows three types of questions: yes/no, location, and count. In addition, when multiple tasks are specified together, the ordering is based on the sequence of expressions. But there are no provisions for explicit temporal directives. Tempus removes this restriction.

2.4.3 C-barp

After analyzing the utterances, TODUS generates internal requests to C-barp, which interleaves plan generation and execution using a distributed contract-bid approach [Resuta 91]. A novel feature of C-barp is its committee-based control structure. The first committee consists of the four individual robot planners that compete for the contracts. The second committee assesses the bids and also maintains contingency plans. The other committee determines if the actual results differ from the expectations. A strategic agent monitors the actions of all the committees.

The planner uses levels of abstraction to simplify plan construction and to facilitate replanning. The highest level is the job set, which corresponds to a series of related requests from TODUS. At the next level there is the job, which covers a single request. The middle level is for chores, or parts of the job that can be done by different robots. Chores are broken down into elemental chores, corresponding to logically distinct actions. The lowest level is for the elemental move, which covers the low-level robot motions. Examples of each of these follow:

Job Set	fasten screw at <2,3>; fasten nail at <3,3>
Job	fasten screw at <2,3>
Chore	pass screwdriver
Elemental Chore	pick up screwdriver
Elemental Move	open gripper

These levels determine where checkpoints are inserted. After requesting the robot manager to perform the set of elemental moves for an elemental chore, C-barp requests a synopsis of the work area from FUSE. If this is not as expected, failure correction is initiated. If the failure is recoverable then replanning is performed for the elemental chore. Otherwise the user will be alerted to the problem.

3 Task-Oriented Temporal Analysis & Reasoning

3.1 Introduction

In this section, the application of temporal analysis and reasoning to task-oriented domains will be discussed. For the most part, the discussion will apply to any task-oriented domain, but some of it will only be applicable to domains similar to the MCAP workbench domain, which will be used for most of the illustrations. However, examples will also be drawn from the area of command-level interfaces to show the generality of the approach.

Temporal analysis is treated first. Emphasis is on the interpretation of temporal adverbials and connectives, which are most important in this area of discourse. Verbal aspect will also be discussed since it is useful to express finer distinctions in the task specifications. Tense will be briefly covered, since it is used the least to make distinctions. Also, the use of focus maintenance in task-oriented domains will be discussed.

Next, temporal reasoning will be considered. The basic approach will not need modification. Although it is a general constraint propagation technique, it cannot be simplified without jeopardizing its effectiveness. But after all the relations have been propagated, the tasks might still be partially ordered. Therefore the last section discusses a method for determining an order for task execution consistent with the constraints.

3.2 Temporal Analysis

3.2.1 Overview

Most natural language expressions for task specifications consist of imperative commands using the simple present tense (with perfective aspect):

> *Measure* the green bench using the ruler.
> *Fill* the red bench with screws.
> *Compile* the C source file.

Therefore, since tense variation is not frequent, the temporal specifications mainly consist of temporal adverbials or connectives:

> Hold the ruler *for two minutes*. *Then* fasten the nail.
> *In ten minutes*, run the sort program.
> Print the error log *at 5pm.*
> Screw the red screw, *before* you hammer a blue one.

When finer distinctions need to be made concerning the tasks mentioned in a complex sentence, imperfective aspect can be used (instead of the usual perfective):

> *When* you hammer the red nail,
> also screw in the green screw.
>
> *When* you hammer the red nail,
> make sure you are also screwing in the green screw.
>
> *When* hammering the red nail, screw in the green screw.
>
> *When* hammering the red nail,
> make sure you are also screwing in the green screw.

Imperfective aspect can also be used with other temporal adverbials, but the perfective counterpart is more common:

> Make sure you *are testing* the program *on Monday*.
> *Test* the program *on Monday*.

Note that in the second example the testing is considered a complete process that will be performed entirely on Monday, where the first can be spread out over a few days.

In the following sections, a simple yet effective analysis is given for temporal adverbials and connectives including the influence of tense and aspect. The approach concentrates on the interpretation of the constructs without concern for grammatical niceties. Thus instead of reporting that some usage is ungrammatical or questionable, a system based on this model would still interpret the user's intentions, except when the utterances are unintelligible. This is appropriate for most natural language interfaces, since the user directs the interface and probably does not want to be corrected on his or her English, no matter how ungrammatical or infelicitous.

Therefore, to simplify the analysis, the model concentrates on the similarities rather than the differences between constructs. For example, all of the frame adverbials are analyzed by placing the reference time of the event within the temporal period specified. Likewise, most of the tenses will be treated in the same way, since the distinction between present and future is blurred.

In task-oriented domains, this will pose no significant problems for several reasons. First, most people are not strict about their usage when speaking, so the interface does not have to be written for the special-case grammarian. Narratives, on the other hand, usually have been prepared by skilled writers, so a text processing program needs to determine subtle differences in expressions. In addition, the interface has the option of requesting further information if it cannot adequately determine a task specification. For written narrative, this obviously is not possible. Similarly, the interface can include a mandatory verification of the task to be performed (e.g., if the task performance cannot easily be undone).

3.2.2 Tense and Aspect

Tense analysis is not as important as the correct interpretation of the temporal adverbials, since there is little variety of tenses in task-oriented dialogs. This is a major distinction from other discourse areas, such as narratives, where tense plays a crucial role. But there are still certain uses for some of the tenses. Therefore, a modification of the Reichenbachian framework will be used, as explained below.

Since an interval-based representation is being used for events, the three reference points in a tense structure will be treated as intervals. Thus for the common tenses of this application area, the reference interval coincides with the event interval, as in both the main and adjunct (subordinate) clauses of this complex sentence:

Before you *insert* the nail, pick up the ruler.
$S_{adjunct}$ {< =} $R_{adjunct}$ {=} $E_{adjunct}$ S_{main} {< =} R_{main} {=} E_{main}

Note that the speech time can either be coincident with the reference time, as in a request for immediate action, or prior to the reference time, as in a request for a later action which is common in task-oriented dialogs. This models the ambiguity that the present tense has in regard to the time of action. Since this is the default situation, the tense structures will omit the speech time interval from now on, except to show when the tense indicates a different relationship or to highlight a contrast.

The present perfect tense is sometimes useful since it changes the emphasis to the consequence of a task:

Before you *have inserted* the nail, pick up the ruler.
$R_{adjunct}$ {mi} $E_{adjunct}$ R_{main} {=} E_{main}

Here the reference interval immediately follows the event interval (i.e., the event interval meets the reference interval). This gives a more intuitive interpretation than R {>} E, a straight translation of the traditional tense structure E < R = S, which allows a gap between the intervals.

The past tenses are uncommon since the time of the actions is always the present or future. But they can be used to describe reasons for a requested action or to provide commentary:

> Since you *dropped* the only red screw, fasten a green one.
> $S_{adjunct}$ {>} $R_{adjunct}$ {=} $E_{adjunct}$ R_{main} {=} E_{main}
> Before you *had dropped* the screw, you *were doing* well.
> $S_{adjunct}$ {>} $R_{adjunct}$ {mi} $E_{adjunct}$ S_{main} {>} R_{main} {= f} E_{main}

For most applications, these uses are not necessary. Therefore, the past tenses can be ignored for the sake of simplicity. But, for applications incorporating machine learning, these types of expressions can provide valuable feedback.

For the most part, the future tense can be interchanged with the present tense (since the actions indeed refer to the future), but the simple present is usually preferred:

> *Screw* in two screws. Then *hammer* a nail.
> S_1 {< =} R_1 {=} E_1 S_2 {< =} R_2 {=} E_2
> You *will screw* in two screws. Then you *will hammer* a nail.
> S_1 {<} R_1 {=} E_1 S_2 {<} R_2 {=} E_2

In addition, the use of the future tense in a subordinate clause with a temporal connective seems odd, if not inappropriate:

> ? You *will screw* in two screws, before you *will hammer* a nail.
>
> ? *Screw* in two screws, before you *will hammer* a nail.
>
> You *will screw* in two screws, before you *hammer* a nail.
>
> *Screw* in two screws, before you *hammer* a nail.

Finally, the prospective tenses are often used to provide previews of the actions to be taken or to set the background for the actual actions:

> First *you are going to work* with the red bench. S_1 {< =} R_1 {<} E_1
> Screw in three screws.... S_2 {=} R_2 {=} E_2

Then *you are going to use* the blue bench.	S_3 {< =} R_3 {<} E_3
Fill the second column with nails....	S_4 {=} R_4 {f =} E_4
You *will be going to be working* with the red bench.	S_1 {<} R_1 {<} E_1
First you will screw in three screws....	S_2 {<} R_2 {=} E_2
. . .	

The first example shows how the prospective is used to segment the utterance. This can be exploited during (temporal) focus maintenance, as we will explain later. The second example shows that there is little difference between the simple and future versions of the first two sentences, except for the former allowing actions in the immediate future. In addition, the analysis of the prospective tenses tends to get complicated since they are also used in modal constructions (expressing requirements, intentions, etc.):

You *are going to document* all the source files,
or you will not get the raise you have asked for.

These uses are beyond the simple model being developed here. Finally, in each of these examples, the prospective tenses can be replaced by the simple future (similar to the way the past tense can be used for the present perfect).

Verbal aspect indicates whether an action is viewed as being complete (perfective or nonprogressive aspect) or as being ongoing (imperfective or progressive). It is used to override the default viewpoint of events associated with the event's category. The default viewpoints are shown below:

state:	$RT \subseteq ET$
process:	$RT \subseteq ET$
culminated process:	$RT = ET$
culmination:	RT {finishes, equals} ET
point:	$RT = ET$

As mentioned earlier, imperfective aspect allows for finer distinctions in the ordering of events. With the usual case of perfective aspect, the event is treated in its entirety. But with imperfective aspect, the event is viewed from within. Therefore, the reference time will be contained in the event time ($RT \subseteq ET$).

3.2.3 Temporal Adverbials

The analysis of temporal adverbials will follow Almeida [1987] by using the specified time to order the event's reference time, instead of ordering the actual event time. This nicely handles the interaction between the adverbials and other temporal constructs:

> Fill the first row with screws *before 7 pm*.
> Make sure you are filling the first row with screws *before 7 pm*.

In both cases, the reference time is prior to 7 pm. However, in the first sentence, the event time precedes the specified time, whereas, in the second sentence, it could possibly extend past the specified time.

General Framework

Given any temporal adverbial specification, the main practical concern is the determination of a reference time for the event such that the temporal constraints are met. This time is usually a time interval but it can also be a time point. However, the determination usually cannot be performed in isolation, since the specifications are often vague and since there might be several tasks possible for the same time frame:

> *In the afternoon* run the graphics demos *for an hour*.
> And compile the expert system *before 3 pm*.
> Also, purge the hard drive *this week*.

Thus these decisions have to wait until all the tasks are processed. Then, domain knowledge can be used to determine a satisfactory order. However, certain parameters could be inferred to facilitate the later decisions:

<start time, end time>	acceptable reference time range
RT relation	relation of RT to the specified interval
duration	length of the event occurrence
extent	length for an (the) actual task
repetitions	number of task occurrences

For instance, the three tasks might be analyzed as follows:

	RT relation	time interval	duration	extent	repetitions
compile:	{<}	<24 Aug: 1500, 24 Aug: 1505>	60m	60m	1
purge:	$\subseteq$	<24 Aug: 1200, 28 Aug: 1700>	30m	30m	1
demo:	$\subseteq$	<24 Aug: 0800, 24 Aug: 1700>	60m	15m	3

As used here, the extent is the typical length for a task of the given type, but the duration is the length of the event for the performance of the task. The interaction between these two parameters determine if the actual event is a repetition (duration > extent), an exact rendition (duration = extent), or a partial completion (duration < extent) of the event type. For the case of repetitions, the intervals between each individual task are assumed to be equal (following Karlin [1988]), as in the five-minute gap between successive demos.

Most of the categories of temporal adverbials that Almeida presented for narratives (see Table 3) are still applicable to task-oriented situations. However, since the fixed-past frame adverbials can only be used with the past tenses, they are restricted to commentary. Likewise, adverbials using *previous* or *preceding* have limited use. The remainder of this section will provide an analysis for most of the adverbials applicable to this area.

The general format for temporal adverbials follows:

$$\text{Temporal Adverb} \quad \rightarrow \quad \begin{array}{ll} \text{(prep)} & \text{NP}_{\text{interval-type}} \\ \text{adj} & \text{N}_{\text{interval-type}} \\ \text{adverb} & \end{array}$$

This shows that most of the adverbials are built around noun phrases, preceded by an optional preposition. For these cases, the actual noun is referred to as $N_{interval\text{-}type}$ and the noun phrase as $NP_{interval\text{-}type}$ (the interval type indicates the time span). This distinction is useful since some of the adverbials require specific adjectives, such as *next* (e.g., "next month"). Although the second case could be considered as a special case of the first, it is clearer to keep them separate. Furthermore, there are several different types of intervals referred to by the adverbials. The most general interval type covers an arbitrary stretch of time, as in the adverbial *this* $NP_{interval}$:

Do it *this minute*.
Just do it sometime *this century*!

But some of the interval types have a restricted range, as in the adverbial *at* NP_{time}:

Fill the bench *at 2pm*.
Fill the bench *at half past seven*.

Frame Adverbials

Two common adverbials are *(on)* $NP_{interval}$ and *in* $NP_{interval}$. Here the reference time is taken to be within the interval specified, and it is also taken as any subinterval of the event time (except for a special case of *in*). Next, either the date range or the time range is filled using the specified interval, and the component not modified will use the default value. The parameters for all adverbials using $NP_{interval}$ and $N_{interval}$ are handled in the same manner. Also, *during* $NP_{interval}$ is treated like *on* and *in*, since there is negligible semantic difference between the three adverbials. For example, the slight contrast is illustrated below:

? Fill the red bench *in the weekend*.
Fill the red bench *on the weekend*.
Fill the red bench *during the weekend*.

Run the constraint propagation sieve *in March*.
? Run the constraint propagation sieve *on March*.
Run the constraint propagation sieve *during March*.

Next there are adverbs that can be considered as translations of equivalent forms using *in*, *on*, or *during* as given above:

today ≈ (on) this day RT $\subseteq$ <0800, 2000>

tomorrow ≈ (on) the next day RT ⊆ <current-date + 1, current-date + 1>
tonight ≈ (during) this night RT ⊆ <1700, 2400>

The determination of these endpoints is based both on context and domain knowledge:

Run the MCAP demo *today*.
Test the MCAP demo *today*.

In the first case, it is assumed that the demonstration will be during the normal operation hours, whereas in the second, it is only required that the testing occur before midnight.

The *at* NP_{time} adverbial behaves like those in the *on/in* family except that an actual time rather than an interval is specified. But, as Almeida shows, the time should be treated as a small interval surrounding the time point in order to accommodate the differences shown by variations of the adverbial:

Fasten the red screw *at 3pm*.
Fasten the red screw *at precisely 3pm*.
Fasten the red screw *at around 3pm*.

Then, *now* can be handled as if it were defined in terms of *at* as follows:

now ≈ (at) this time RT ⊆ <current-time, current-time + x>

Again, context and domain knowledge will be required to determine the size of the interval implied. This analysis shows that many frame adverbials can be treated in the same general fashion, so long as minor semantic differences are ignored.

Durational Adverbials

The special case of *in* involves the case when the interval is a time period (e.g., "in three hours"). The *in* $NP_{time\text{-}period}$ adverbial is ambiguous between the interpretation that has the event within the time period and the one that has the event's extent given by the period. In the second case, the event time interval will be set to the specified length, and the reference time will also be set to the same interval (i.e., RT = ET):

> You are going to fill the bench *in nine minutes*.
> a. ET {finished-by, equals} RT ⊆ <now + 9m, now + 9m + typical extent>
> b. RT = ET = <x, x + 9m>; extent = 9m

This interpretation is compatible with the analysis in [Moens & Steedman 88], in which the *in* adverbial transforms a culmination *(fill)* into a culminated process (*fill in nine minutes*). To select the appropriate interpretation, the default extent for the task is compared to the specified time. If the times are comparable then the extent-specification reading is assumed, otherwise the start-time reading will be used. Therefore, if it usually takes ten minutes to fill a bench, the previous sentence will be taken as a command to speed up a little bit. In order to avoid this problem, the speaker can use an adverbial that explicitly indicates the interpretation:

> You are going to fill the bench *within nine minutes*.
> You are going to fill the bench *in under nine minutes*.

The *for* $NP_{interval}$ temporal adverbial is used to specify the duration for the event time. This is modeled by setting the duration parameter and by again equating the reference and event times. As with *in* $NP_{time\text{-}period}$ the expression is transformed into a culminated process:

> Fasten the blue screw.
> Then wait *for two minutes*.
> RT = ET = <x, x + 2m>; duration = 2m (extent = undefined)
> Next . . .

Moens and Steedman claim that the *for* adverbial is only appropriate with process events (where the default extent parameter is undefined) as the above. But they also show that just about any occurrence can be given an interpretation since an event of a particular type can usually be transformed into another type. As an example, consider the contrast in the following:

> Screw the blue screw in row 1 column 1 *for five minutes*.
> RT = ET = <x, x + 5m>; duration = 5m (extent = 1m)
>
> Fasten the blue screw in row 1 column 1 *for five minutes*.
> RT = ET = <x, x + 5m>; extent = duration = 5m

Since the time to fasten a screw is about a minute, both of the events specify durations longer than the typical task extent. The first one is taken as a request for a repeated rotation of the screw (to make it well-fastened), since the culminated process to screw a nail in the corner can readily be transformed to a process of simply screwing the screw in the same position. However, the second expression seems odd, since, like many culminations, *fasten the screw* is generally taken as a nonrepeatable action. The task is usually extended to fill the duration of the interval, indicating that the work is to be done slowly. But the repeated occurrence interpretation can be forced if more plausible (e.g., a request to repeatedly do and undo the action).

Relational Adverbials

Whereas the frame adverbials place the event or reference within the given time interval, the next group of adverbials, the relational adverbials of Table 3, place the event or reference outside the interval (i.e., either before or after it). The following shows how *before* $NP_{interval}$ is handled:

> Screw in a red screw *before 3pm*.
> ET = RT {before} <3pm, 3pm + ε>

The adverbial *after* $NP_{interval}$ has the opposite effect, with the reference after the interval. Also, *by* $NP_{interval}$ is treated similar to *before* $NP_{interval}$, except that the event time can meet the specified interval.

> Screw in a red screw *by 3pm*.
> ET = RT {before, meets} <3pm, 3pm + ε>

The $NP_{interval}$ *earlier* and $NP_{interval}$ *later* adverbials behave like the *before/after* pair, having opposite effects on the ordering of the reference time against the specified interval. However, with this pair, the interval is determined based on the current focus point, which will be discussed later. The treatment for these two is illustrated by this example:

> Run the graphics demo *tomorrow at 2pm*.
> ET = RT $\subseteq$ <current-date + 1 : 1400 - ε, current-date + 1 : 1400 + ε>
> But test the program *fifteen minutes earlier*.
> ET = RT $\subseteq$ <current-date + 1 : 1345 - ε, current-date + 1 : 1345 + ε>
> And run the MCAP demo *a half an hour later*.
> ET = RT $\subseteq$ <current-date + 1 : 1430 - ε, current-date + 1 : 1430 + ε>

Here the interaction between adverbials within a clause is handled intuitively. Note that the temporal focus is updated by the first sentence but not by the others since they are defined relative to the first.

Next, *until* $NP_{interval}$ produces an effect similar to that of *for*, since the event's duration is being specified, although in this case the interval's end point is given. Therefore the same considerations must be taken into account to determine whether the intended event is a repeated occurrence of the task, a slowed-down rendition of it, or a partial performance of it:

> Screw the blue screw in row 1 column 1 *until five o'clock.*
> RT = ET = <x, x + 5m>; duration = (5pm - focus-time); (extent = 1m)
>
> Fasten the blue screw in row 1 column 1 *until five o'clock*.
> RT = ET = <x, x + 5m>; extent = duration = (5pm - focus-time)

The last two relational adverbials to be considered, *then* and *next*, are quite similar. Both ensure that the current clause is interpreted as being at a subsequent time to the previous clause, instead of possibly overlapping it:

> Screw a green screw in the red bench
> a. And hammer a blue nail in the yellow bench
> ET = RT $\subseteq$ <focus-time + ε, focus-time + ε + extent>
>
> b. And *then* hammer a blue nail in the yellow bench
> c. And hammer a blue nail in the yellow bench *next*.
> (both) ET = RT $\subseteq$ <focus-time + x, focus-time + x + extent>

In case (a), the event could be taken to occur at the same time as the screw task, if the domain supports simultaneous actions and the actions don't interfere with each other; otherwise, the hammer event has to be subsequent. This second interpretation is explicitly specified in both (b) and (c). Note that *next* can be substituted for *then* in case (b), but the opposite doesn't hold for case (c).

Time-Frequency Adverbials

The last group to be considered consists of adverbials that specify the frequency of the action. This group is quite diverse since there are various ways to indicate the number of times an action is to be repeated. There are two broad classes [Quirk et al. 85]: definite frequency adverbials give an explicit count ("twice") or the time interval between occurrences ("weekly"); and indefinite frequency adverbials consist of a qualitative indication (e.g., "continually", "repeatedly", and "occasionally").

The definite frequency type is simpler to model since the frequency is not dependent upon context. If a count is specified, this will be used for the repetitions parameter. Otherwise the value is determined by dividing the overall duration by specified interval. Examples of the two cases of definite frequency adverbials follow:

> Run the MCAP demo *twice this week.*
> RT ⊆ <start-of-week, end-of-week>; repetitions = 2
>
> During this month, run the graphics demos *weekly.*
> RT ⊆ <start-of-month, end-of-month>; repetitions = 4

With the indefinite frequency type, domain knowledge is required to determine a quantitative value for the frequency specification. This is required since an adverb like *occasionally* modifies the general frequency for the type of action rather than for a generic action:

Fill the blue bench with screws,
a. and *occasionally* hammer a nail in the red bench. repetitions = 5
b. and empty a row in the other bench *occasionally*. repetitions = 2

The first case is given a higher repetition value since hammering is a frequent action in the domain. However, emptying a row is infrequent, so it is given a relatively low count. Note that some indefinite frequency adverbials cannot be handled with this approach, since they reflect habitual or customary actions. These will not be modeled, as explained below.

Adverbials Not Modeled

There are several types of adverbials that will not be considered in this thesis, because they are not sufficiently applicable to task-oriented domains to justify the added complexity required to model them. These include the following: adverbials involving negation ("not on Friday"); the indefinite frequency adverbials expressing usual occurrence ("by and large") or universal frequency ("perpetually"); and miscellaneous time-relationship adverbs ("yet") [Quirk et al. 85].

Summary

Table 5 summarizes temporal adverbial analysis in Tempus. The first two entries cover the many time-frequency adverbials available. In addition, the $NP_{interval}$ entry covers the cases having the preposition *in* or *during* omitted.

temporal adverbial	R *relation* interval	R *relation* E	parameters set
Adv_{count}	{s d f =}	{s d f =}	repetitions
$Adv_{frequency}$	{s d f =}	{s d f =}	repetitions
after $NP_{interval}$	{>}	{s d f =}	date or time
at NP_{time}	{s d f =}	{s d f =}	time
before $NP_{interval}$	{<}	{s d f =}	date or time
by $NP_{interval}$	{< m}	{s d f =}	date or time
during $NP_{interval}$	{s d f =}	{s d f =}	date or time
$NP_{interval}$ ***earlier***	{mi =}	{s =}	date and/or time
for *NP*	{=}	{=}	duration
in *(Adv)* $NP_{interval}$	{f =}	{=}	extent
in $NP_{interval}$	{s d f =}	{s d f =}	date or time
$NP_{interval}$	{s d f =}	{s d f =}	date or time
$NP_{interval}$ ***later***	{mi =}	{s =}	date and/or time
next	{s =}	{s =}	date or time
now	{s d f =}	{s =}	time
on $NP_{interval}$	{s d f =}	{s d f =}	date
over $NP_{interval}$	{s d f =}	{s d f =}	date
then ...	{s =}	{s =}	date or time
today	{s d f =}	{s d f =}	date
tomorrow	{s d f =}	{s d f =}	date
tonight	{s d f =}	{s d f =}	date and time
until $NP_{interval}$	{m}	{f =}	date and time
within $NP_{day\text{-}subinterval}$	{s d f =}	{s d f =}	time

Table 5. Temporal Adverbial Definitions[1]

[1] This table was derived using information mainly from [Almeida 87]. But [Karlin 88], [Moens & Steedman 88], [Quirk et al. 85], and [Song 91] were also consulted. In addition, see Figure 2 for definitions of the temporal relations (with their abbreviations).

3.2.4 Temporal Connectives

Temporal connectives are subordinating conjunctions that indicate the temporal order between the clauses in a complex sentence. Although they are a subgroup of temporal adverbials, their importance warrants treating them as a separate group. For instance, it is more convenient to specify the relative ordering of tasks than to use common reference points:

a.	Shortly before 3pm, clear the bench.	R_1 {f =} E_1
	Then insert a red screw at 3pm.	R_2 {=} E_2
	Next hammer a nail.	R_3 {=} E_3
	Also, measure the size of the bench.	R_4 {=} E_4
b.	Clear the bench	R_1 {f =} E_1
	before you insert a red screw at 3pm.	R_2 {=} E_2
	Then, *when* you are hammering the red nail,	R_3 {d s f =} E_3
	measure the length of the bench.	R_4 {=} E_4

As with temporal adverbials, the reference point of the main event is being specified using the interval given by the adverbial (connective). However, that interval is actually the reference time for another event, described by the subordinate clause. At first sight, it seems that the actual event times are ordered, but the effects of tense and aspect suggest otherwise:

b′.	Clear the bench	R_1 {f =} E_1
	before you have inserted a red screw at 3pm.	R_2 {mi} E_2
	Then, *when* you hammer the red nail,	R_3 {=} E_3
	be measuring the length of the bench.	R_4 {d s f =} E_4

Here, the relations of the reference times are the same, except that the events orderings differ. The bench clearing could possibly overlap the screw insertion, instead of just preceding it. But the bench measurement temporally contains the hammering, instead of the opposite:

b.	E_1 {= fi} R_1 {<} R_2 {=} E_2	$\Rightarrow$	E_1 {<} E_2
	E_3 {= fi si di} R_3 {=} R_4 {=} E_4	$\Rightarrow$	E_3 {= fi si di} E_4
b′.	E_1 {= fi} R_1 {<} R_2 {mi} E_2	$\Rightarrow$	E_1 {< d o m s} E_2
	E_3 {=} R_3 {=} R_4 {d s f =} E_4	$\Rightarrow$	E_3 {d s f =} E_4

Before and After

The connectives *before* and *after* are nearly opposite in nature. Note that Almeida [1987] asserts the difference that *before* doesn't require the occurrence of the subordinate event, whereas *after* requires both event occurrences. The following is a counterexample to that claim:

> Test the MCAP programs, *before* you show Ed the demo.
> Show Ed the MCAP demo, *after* you test the programs.

If Ed leaves town before the programs have been tested, he might never see the demonstration. Therefore, both connectives do not require that the presumed temporally-subsequent task ever occurs. However, if it does occur, it must indeed be subsequent.

The interaction with aspect shows that the two connectives are not inverses of each other [Harper and Charniak 86]:

Test the MCAP programs, *before* you are showing Ed the demo.

$E_1 = R_1 \; \{<\} \; R_2 \; \{s =\} \; E_2 \quad \Rightarrow \quad E_1 \{<\} E_2$

Be showing Ed the MCAP demo, *after* you test the programs.

$E_1 \; \{= fi\ si\ di\} \; R_1 \; \{>\} \; R_2 \; \{=\} \; E_2 \quad \Rightarrow \quad E_1 \{> di\ oi\ mi\ si\} E_2$

Be testing the MCAP programs, *before* you show Ed the demo.

$E_1 \; \{= fi\ si\ di\} \; R_1 \; \{<\} \; R_2 \; \{=\} \; E_2 \quad \Rightarrow \quad E_1 \{< di\ o\ m\ fi\} E_2$

Show Ed the MCAP demo, *after* you are testing the programs.

$E_1 = R_1 \; \{>\} \; R_2 \; \{d\ s\ f\ =\} \; E_2 \quad \Rightarrow \quad E_1 \{> d\ oi\ mi\ f\} E_2$

Although these sentences are odd, they show that *before* has an additional restriction that the start of the subordinate event is being referenced. This holds by default for perfective events since the entire event is being referenced. But, since the default reference for imperfective events is any subinterval, the restriction produces the effects noted above.

Therefore, the analysis can be summarized as

Task_{main} *before* Task_{sub}: $\quad R_{main} \{<\} R_{sub}$ and $R_{sub} \{s =\} E_{sub}$

Task_{main} *after* Task_{sub}: $\quad R_{main} \{>\} R_{sub}$

Once and Until

The connectives *once* and *until* are used to specify one of the boundaries of the main event, using the subordinate event. Thus they have opposite effects and are practically converses. For *once*, the reference time of the main clause immediately follows the reference time of the subordinate clause (i.e., the end of the subordinate reference equals the beginning of the main reference). But for *until* the reference time of the main clause immediately precedes that of the subordinate clause (i.e., the beginning of the subordinate reference equals the end of the main reference). For example,

Screw in a red screw, *once* you measure the bench.
E_1 {=} R_1 {mi} R_2 {=} E_2 $\Rightarrow$ E_1 {mi} E_2
Measure the bench, *until* you screw in a red screw.
E_1 {=} R_1 {m} R_2 {=} E_2 $\Rightarrow$ E_1 {m} E_2

There is an important difference in the two previous examples. In the second, the duration of the measurement task can be extended to ensure that the two events meet. Therefore, the sentence seems a little odd in isolation since the measurement task is not fixed in time. But in a proper context the sentence would seem all right:

At 2pm, pick up the ruler.
Measure the bench, *until* you screw in a red screw.
But don't start screwing before 2:10pm.

So the measurement task is given an implicit duration of ten minutes, which is longer than usual. This can be taken as a command for the robot to keep itself busy for a few minutes.

Thus these two connectives are interpreted as follows:

T_{main} *once* T_{sub}: R_{main} {mi} R_{sub}
T_{main} *until* T_{sub}: R_{main} {m} R_{sub}; $duration_{main} = start(T_{main}) - start(T_{sub})$

When and While

The connectives *when* and *while* are very close in meaning, although there are some important differences. For instance, *when* often is used to suggest causality as noted by Moens and Steedman (see section 2.3.4), and it can be used in narratives to introduce climatic information, as in the following (based on an example from [Quirk et al. 85]):

I was editing the document, *when* suddenly the power went out.

On the other hand, *while* often has the effect of stretching the main event to fill the time specified by the subordinate clause [Almeida 87]:

Review the documentation, *while* the server is down.

Some of these distinctions can be maintained by treating the reference times as equal, and for the case of *while*, ensuring that each reference time spans the event (or events):

Hold the red bench, *when* you pound in the nail.

E_1 {= fi si di} R_1 {=} R_2 {=} E_2 $\Rightarrow$ E_1 {= fi si di} E_2

Hold the red bench, *while* you pound in the nail.

E_1 {=} R_1 {=} R_2 {=} E_2 $\Rightarrow$ E_1 {=} E_2

In the first case the main event represents a process that temporally contains the subordinate event, a culminated process. In the second, all the intervals are equal, which is acceptable.

Unfortunately, the analysis for *when* is too general when the subordinate clause has imperfective aspect:

Hold the red bench, *when* you are pounding in the nail.

E_1 {= fi si di} R_1 {=} R_2 {d s f =} E_2 $\Rightarrow$ E_1 {d di o oi s si f fi =} E_2

Hold the red bench, *while* you are pounding in the nail.

E_1 {=} R_1 {=} R_2 {=} E_2 $\Rightarrow$ E_1 {=} E_2

Both clauses now represent processes since the imperfective aspect transforms the culminated process *pound a nail* into the process of *pounding the nail*. Here the simple approach of equating the reference times leads to the broad interpretation that the two events just somehow overlap.

As shown earlier in section 2.3.5, Almeida handles the *when* connective using four special cases based on the aspects of both clauses. In addition to increasing the complexity of the analysis, his approach still leaves open the interaction with the present perfect tense, which he didn't consider:

Hold the red bench, *when* you have pounded in the nail.
E_1 {= fi si di} R_1 {=} R_2 {mi >} E_2 ⇒ E_1 {> di oi si fi} E_2

Hold the red bench, *while* you have pounded in the nail.
E_1 {=} R_1 {=} R_2 {mi >} E_2 ⇒ E_1 {mi >} E_2

Compile the program for optimization, *when* you have debugged it.
E_1 {= fi} R_1 {=} R_2 {mi >} E_2 ⇒ E_1 {> di oi si fi} E_2

Put down the ruler, *when* you have been pounding in the nail.
E_1 {= fi} R_1 {=} R_2 {oi} E_2 ⇒ E_1 {di oi si} E_2

The first sentence doesn't make much sense for the domain but is included to show the contrast produced by the present perfect. Next, the second sentence shows that this interaction makes little sense for the connective *while*. The third case shows that the combination of *when* and the present perfect is like *after* with the simple present. The last sentence illustrates the effect of the present perfect progressive (i.e., with imperfective aspect).

In order to retain the simplicity of the approach and yet produce results more approximate than the above, domain knowledge will be used to filter the possible relations. Since this is compatible with the processing required for nucleus event structures, it is an attractive alternative. First, the contingency relation between the two events is checked. If there is no applicable relation, then no filtering will be performed. Next, the contingency relation will be translated into a set of compatible interval relations, which is then intersected with the relation previously determined to produce the new relation between the two events. The following tables show how this information might be defined for MCAP.

event$_1$	***event$_2$***	***contingency relation***
fasten	fill	constituent of
hammer	insert-fastener	equivalent to
hold-bench	insert-fastener	expedient for
insert-fastener	fasten	preparation of
measure	fasten	preparation of
pick-up	insert-fastener	preparation of
place	insert-fastener	preparation of
pound	insert-fastener	equivalent to
screw	insert-fastener	equivalent to

Table 6. Contingency Relations for MCAP Events

x ***contingency relation*** y	E_x ***interval relation*** E_y
consequent of	{mi}
constituent of	{s d f}
equivalent to	{=}
expedient for	{di o}
preparation of	{< m}

Table 7. Interpretation of Contingency Relations

For example, again consider

Hold the red bench, *when* you are pounding in the nail.

E_1 {= fi si di} R_1 {=} R_2 {d s f =} E_2 $\Rightarrow$ E_1 {d di o oi s si f fi =} E_2

After the initial analysis is performed as shown above, the contingency relation between the events hold-bench and pound-nail (i.e., insert-fastener) is determined to be expediency. This is compatible with {di o si =}, since it should overlap the beginning somehow. Since the retrieved relation is a subset of the previous relation, it becomes the new relation between the events.

3.2.5 Temporal Focus Maintenance

The need for temporal focus has already been illustrated. In the section on tense it was shown that the prospective tenses are often used to segment a dialog (in effect, by using independent time frames). Also, in the section on temporal adverbials, it was shown that certain adverbials are interpreted with respect to the current time focus (e.g. "ten minutes later").

The following set of requests illustrates the processing generally required:

E_1	Screw in a red screw,	$F=E_0$
E_2	before screwing in a blue one.	$F=E_1$
E_3	Then hammer in a green nail	$F=E_2$
E_4	and also a red one.	$F=E_3$
	But, before picking up the red screw,	$F=E_4$
E_5	measure the slot.	$F=E_1$
E_6	Also measure the screw width.	$F=E_5$
	Anyway,	$F=E_6$
E_7	next hammer a yellow nail.	$F=E_4$

The focus is first initialized to a time prior to the first event. Then when each of the first four events is processed, the focus is simply set to the given event. But then a reference to the first event (an example of a nontrivial anaphoric reference) is used to explicitly set the focus for the fifth event. This also implicitly sets up an embedded discourse. After the fifth event is processed, it becomes the next focus. The sixth event is similarly processed. Next, the adverb *anyway* is used to signal a return to the previous focus (event four).

Both the anaphoric reference and the context shift adverb are examples of conversational moves, which indicate shifts in the dialog. Rachel Reichman [1985] developed a model for processing dialogs that maintained a context space using clues from conversational moves. Since her model applies to general discourse (with an emphasis on argumentative-style discourse), some of the conversational moves will not be applicable here, such as the indication of a counterpoint to the opponent's argument. In addition, others will only be required in advanced situations where the computer plays an active role in the problem solving (e.g., if it needs to explain how it determined a complicated plan). In short, only a few types of the moves need to be modeled for simple task-oriented systems.

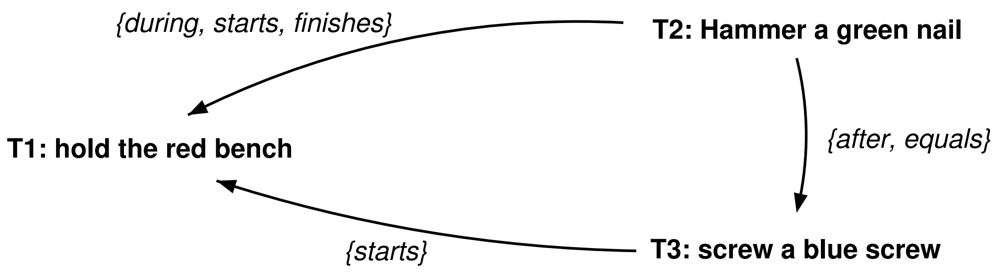
The mechanism for maintaining focus is straightforward when considered at a high-level, as can be seen from the previous example. However, certain of the required operations pose difficulties (especially for implementations of focus maintenance). The major concern is the resolution of anaphoric event references. For instance, since there is no explicit mention of a picking-up event prior to the reference, a simple match will not resolve the reference. To solve the problem, the contingency relations (shown in Table 6) can be incorporated into the matching process. Thus the first event will be matched since picking up a screw is part of the preparation for the actual insertion.

Since only a small number of tenses will be used, Song's approach to focus maintenance will not be as applicable as Webber's. In addition, Webber's approach has been reworked using a deterministic process (which Song emphasized as a design feature of his model), because the backtracking otherwise required would lead to inefficient processing.

3.3 Temporal Reasoning

Allen's algorithm is a simple yet powerful technique for temporal reasoning. However, since it runs in $O(n^3)$ time in the number of events represented, it becomes very inefficient when large numbers of events are modeled. But, in task-oriented dialogs, the number of events discussed is typically limited to several dozen at most. This contrasts with extended narratives, which might describe hundreds of events. Therefore, the algorithm is suitable for most purposes in this application area.

After the constraint propagations have occurred, the tasks might still be partially ordered, due to the ambiguity of the specification. To order the network, domain information (e.g., the contingency relations of Table 6) will be used to select an acceptable interpretation for relations with more than one possibility. Thus, the situation might be as follows:



Using the contingency relations, the *during* and *after* relations would be selected to resolve the ambiguity. In the first case, holding the ruler facilitates hammering the nail and therefore should overlap at least the beginning of it. In the other case, an explicit event order is preferred over equality since the fastener-insertion tasks might interfere with each other. These choices yield the execution sequence (T1, T3, T2) for the tasks.

4 Tempus Architecture

4.1 High-Level System Architecture

Tempus comprises six components (see Figure 11). *System Interface* initializes the system and directs the execution flow. *Discourse Analysis* parses each of the user's utterances and applies semantic analysis with the aid of *Temporal Analysis* and *Temporal Reasoning*. Next, *Response Generation* gets the task sequence from the temporal reasoning component. It communicates with the task planner using *Planner Interface* and then describes the results to the user.

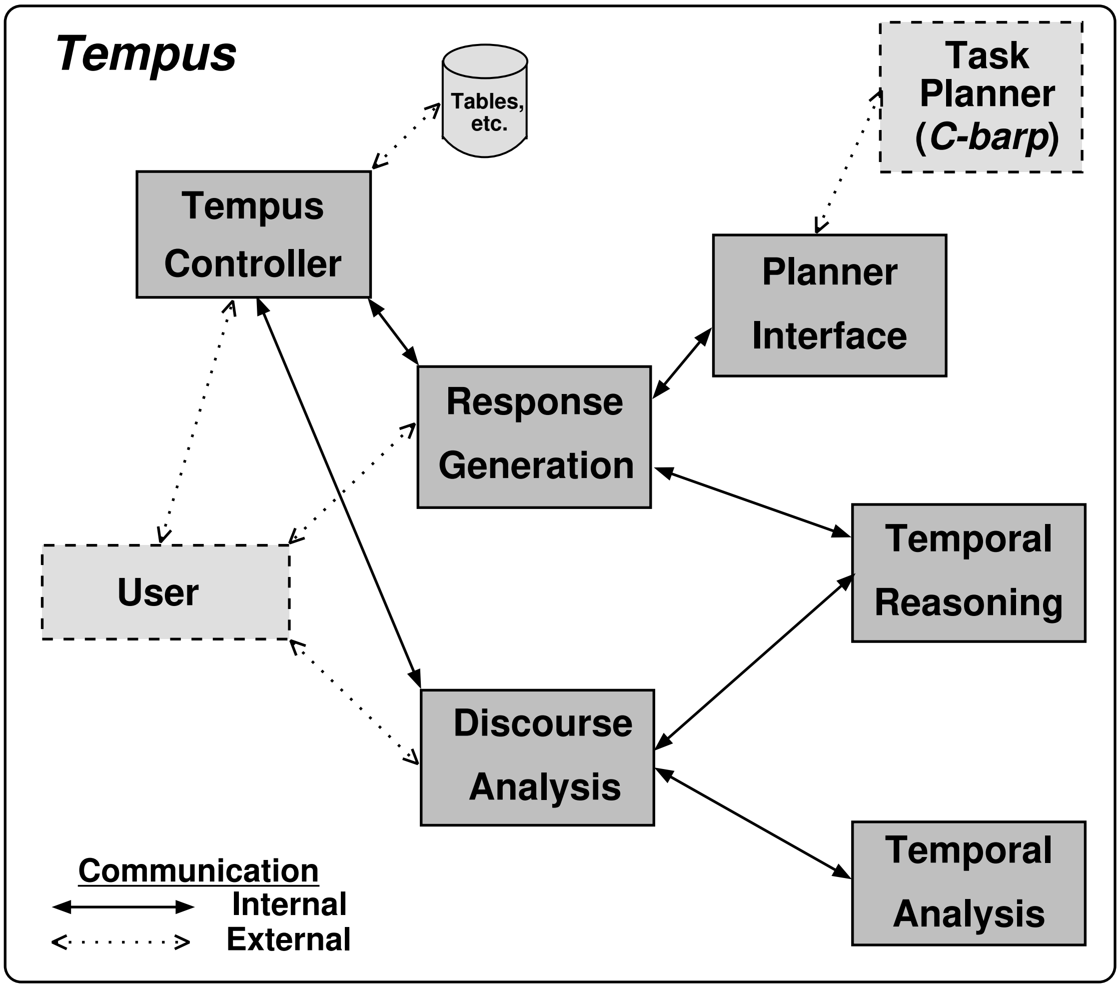


Figure 11. Tempus Architecture

4.2 Data Structures

Several types of data structures are used by Tempus. Figure 12 shows those specific to temporal analysis and reasoning. For instance, *IntervalRelation* defines a 13-element boolean array, indicating the inclusion of the basic relations. *TransitivityTable* is a 13 × 13 matrix of these relations. *TaskNetwork* is also a matrix of the relations, but it is based on the maximal number of tasks. For example, it is usually a 100 × 100 matrix.

EventStructure consists of several parameters: *start_time*, *end_time*, *int_relation*, *er_rt_relation, duration, extent*, and *repetitions*. The first two are instances of a system-specific type that indicates the date and time. This combination determines the interval of the temporal specification for the event[1]. Next, *RT_interval_rel* relates the reference point of the event to this interval. More importantly, *ET_RT_rel* relates the event time to the reference point, as determined by the combination of tense, aspect, and the event type. The last three parameters are simple integer values describing the actual event interval and its atomic action in the case of composite tasks.

Actions enumerates the tasks for the MCAP workbench domain. *Contingencies* gives the relationships possible between the actions, excluding simple temporal coincidence. *ContingencyTable* defines the particular contingency relations between pairs of actions. Furthermore, *ContingencyInterpretation* gives the interval relation compatible with the contingency.

The last group of entries covers types enumerating the possibilities for tense and aspect. The combination of the first two determines one of the nine tense structures. The last entry lists the two possible event viewpoints.

1 A popular implementation is a value giving the number of seconds since 1 Jan 1970.

```
Relations = {before, after, during, contains, overlaps, overlapped_by,
             meets, met_by, starts, started_by, finishes, finished_by, equals}

RelationAbbrevs = {b, a, d, di, o, oi, m, mi, s, si, f, fi, e}


IntervalRelation = Array [1..#Relations] of Boolean

TransitivityTable =  Array [1..#Relations, 1..#Relations] of IntervalRelation

TaskNetwork =  Array [1..MaxTasks, 1..MaxTasks] of IntervalRelation


EventClasses    =   {state, process, culmination, culminated_process, point}

EventStructure = Record
        start_time       :   DateTime
        end_time         :   DateTime
        RT_interval_rel  :   IntervalRelation
        ET_RT_rel        :   IntervalRelation
        duration         :   Integer
        extent           :   Integer
        repetitions      :   Integer


Actions = {fasten, fill, hammer, insert-fastener, measure, pick-up,
           place, pound, screw}

Contingencies = {consequent, constituent, equivalent, expedient, preparation}

ContingencyTable : Array[1..#Actions, 1..#Actions] of Contingencies;

ContingencyInterpretation : Array[1..#Contingencies] of IntervalRelation



TenseTimes = {present, past, future}

TenseViewpoints = {simple, perfect, prospective}

VerbAspects = {perfective, imperfective}
```

Figure 12. Data Structures New to Tempus

Figure 13 shows the data structures that are direct extensions to those from TODUS. The features enumerated by the *Tags* type are binary-valued. Most are based on traditional grammatical analysis, and the others are semantic features from the workbench domain. The last row of the list contains the new ones added for Tempus. Each *LexiconEntry* contains a field for the *word*, its *tags*, and a miscellaneous *code*. The *tags* field is a boolean array that indicates the set of the features that hold. The code value is particular to the word class. The code indicates the type of action for verbs; but it indicates the temporal relation for adverbials. This relation gives the default interpretation for the event's reference time against the time specified.

Next the figure shows that there are six *UtteranceClasses*: simple commands (tasks), composite tasks, anaphoric references, and three types of questions. *ClauseTypes* are the familiar grammatical categories. But *ConversationalMoves* are new ones that typify the common dialog shifts associated with task-oriented discourse. For instance, *advance* is the default case of dialog progression, whereas *enter_subdialog* and *exit_subdialog* are special cases for embedded discourse.

The last data structure shown is the complex frame structure that Tempus uses for modeling utterances. In most cases, the string parameters contain the actual words used in the utterance. However, some constructions are treated by using translations. In these cases, the parameters will have a generic value. The frame slots specifically for temporal analysis are the ones starting at *conversational_move*. The last three are very important: *event_structure* is an instance of the type previously discussed; *connective_rel* is the interpretation of the connective for a subordinate clause; and *other_task* identifies the main clause.

```
Tags = {adjective, color, auxiliary, determiner, noun, preposition,
        pronoun, verb, command, delimiter, plural, number, question,
        source, destination, hole, tool, fastener, column_position,
        column, row_position, row, one, filler, new_action,
        imperfective, date, time, temporal}

LexiconEntry = Record
    word    :  String
    tags    :  Array [1..#Tags] of Boolean
    code    :  Integer

UtteranceClasses =  {Command, CompositeTask, YesNoQuestion,
                     LocQuestion, NumQuestion, AnaphoricRef}

ClauseTypes = {simple, subordinate, main}

ConversationalMoves = {advance, enter_subdialog, exit_subdialog}

FrameSlots  =  Record
      error                :  Boolean
      status               :  String
      class                :  UtteranceClasses
      action               :  String
      fastener_num         :  String
      fastener_color       :  String
      fastener             :  String
      tool_color           :  String
      tool                 :  String
      source_color         :  String
      source               :  String
      row_pos              :  String
      row                  :  String
      column_pos           :  String
      column               :  String
      bench_num            :  String
      bench_color          :  String
      bench                :  String
      text                 :  String
      conversational_move  :  ConversationalMoves
      anaphoric            :  Boolean
      clause_type          :  ClauseTypes
      tense_time           :  TenseTimes
      tense_viewpoint      :  TenseViewpoints
      verb_aspect          :  VerbAspects
      event_class          :  EventClasses
      event_structure      :  EventStructure
      connective_rel       :  IntervalRelation
      other_task           :  Integer
```

Figure 13. Data Structures from TODUS (with extensions)

4.3 Component Description

4.3.1 Tempus Controller

The Tempus Controller component (Figure 14) receives control from the operating system and is the main driver for the system. First, it interprets the run-time options that were specified, as in this command-line invocation of the system:

tempus Start_Up 9 101 102 2001 tempus.log trace suite[2]

Next, it initializes Tempus for a new session by reading in system files (e.g., the lexicon) and setting up the tables required (e.g., temporal relations). Then, it repeatedly invokes the discourse analysis and response generation components until the user enters no more input or an unrecoverable error is encountered. After the session is complete, it terminates the system after performing some house-cleaning functions, such as updating the lexicon.

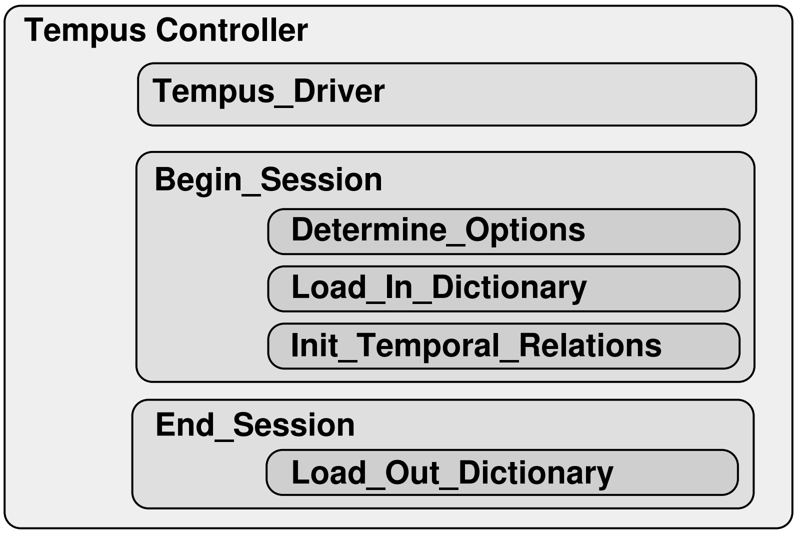


Figure 14. Tempus Controller Component

[2] This specifies that Tempus is to be started up for MCAP operation (nine arguments; message queue keys 101 and 102; central controller process ID 2001; and error log tempus.log), using the test suite for input and producing a trace.

4.3.2 Discourse Analysis

The Discourse Analysis component (Figure 15) performs the syntactic and semantic analysis of the set of utterances input by the user. Therefore it serves as a driver module for the bulk of the system. After a group of utterances is input by the user, it is preprocessed and partitioned into individual utterances. Each utterance is first parsed according to the following task-oriented grammar, which allows for ungrammatical constructs.

Utterance	→	Command | Question | Fragment
Command	→	Filler PP Filler (aux *you* Filler) VP Filler
Question	→	ques VP ques NP2 VP (PP)
Fragment	→	(Filler) (NP | PP) (PP) (Filler)
ErrorRecovery	→	(aux* Filler (NP | NP2 | VP | PP))*
VP	→	(AP) verb (NP PP) (AP)
NP	→	det NP2 PP
NP2	→	adj noun (PP)
PP	→	prep (NP | VP)
AP	→	adv | (prep) NP

Next, the temporal relations are extracted and added to the task network (which must retain consistency). Then, domain-specific semantic checks are made to ensure that the requested task can be carried out and also to fill in implicit information.

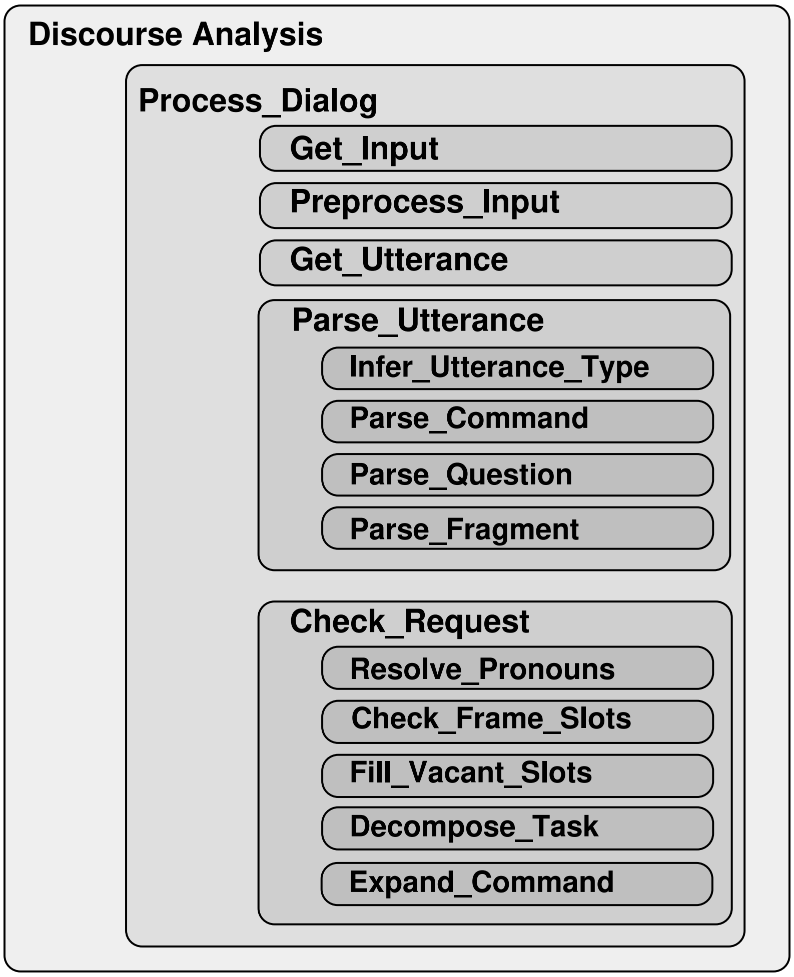


Figure 15. Discourse Analysis Component

4.3.3 Temporal Analysis

The Temporal Analysis component (Figure 16) analyzes each utterance to determine its event structure and its temporal relationship to others in the set of utterances. The event structure indicates the type of event and the parameter values determined during the parse. Domain knowledge is incorporated into the analysis when ambiguous situations are encountered.

First, the event structure is created, using linguistic information such as the verbal aspects of the adjunct and matrix clauses. Then when the second clause of a complex sentence is analyzed, the connective is analyzed to determine the relation between the two tasks. Afterwards, the focus of the dialog is maintained, using clues like the verb's tense.

Figure 17 shows the definitions for the modules of this component. The main purpose of *Create_Event_Structure* is to determine the value for *ET_RT_rel*. Later, *Analyze_Connective* composes the main ET-RT relation with the connective relation giving the intermediate ET1-RT2 relation. It next composes the intermediate result with the subordinate RT-ET relation to yield the relation between the two events (ET1-ET2). Then, it filters this relation with the contingency interpretation to produce the relation to be asserted. The other module, *Maintain_Focus*, is based on Webber's heuristics.

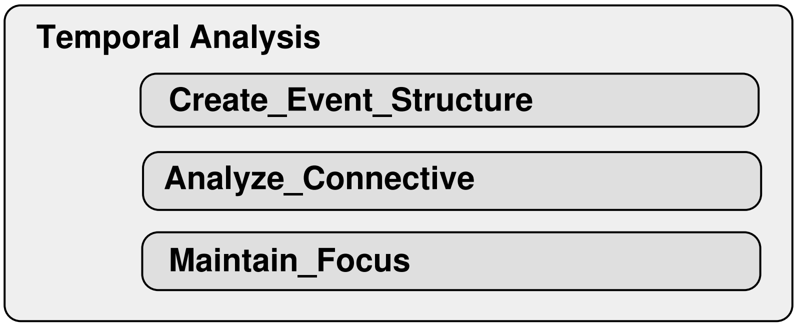


Figure 16. Temporal Analysis Component

```
Create_Event_Structure (inout request, in current)
    with request[current] do
        if question(class) then
            event_type ← state
        else if aspect = imperfective then
            event_type ← process
        else if action <> null then
            event_type ← verb_category(action)
        else
            event_type ← culminated_process

        if duration > 0 or extent > 0 then
            event_type ← culminated_process
            aspect ← perfective

        case event_type of
            state, process :                ET_RT_rel ← {si, di, fi, =}
            point, culminated_process :     ET_RT_rel ← {=}
            culmination :                   ET_RT_rel ← {fi, =}

Analyze_Connective (inout request, network, in main, sub)
    et1_rt2_rel = composition (ET_RT_rel[main], connective_rel[sub]);
    rt2_et2_rel = inverse (ET_RT_rel[sub]);
    et1_et2_rel = composition (et1_rt2_rel, rt2_et2_rel);

    default_rel = contingency (action[main], action[sub]);
    filtered_rel = intersection (et1_et2_rel, default_rel);
    if filtered_rel <> null_relation
        et1_et2_rel = filtered_rel;

    Add_Relation (et1_et2_rel:<main, sub>, network, ok);
    if not ok then
        error_display ("Inconsistent task specification");

Maintain_Focus (inout focus_stack, focus, in request, current)
    case request[current].conversational_move of
        exit_subdialog :
            if not empty(focus_stack) then
                focus := pop(focus_stack)
            enter_subdialog :
                push (focus_stack, focus)
        advance :
            if anaphoric then
                previous := focus
        while previous > 0 and not matched do
                if match(request[previous], request[current]) then
                    focus := previous
                while top(focus_stack) > focus
                    dummy := pop(focus_stack)
        else if tense_viewpoint <> simple then
            push(focus_stack, focus)
        else
            focus := current - 1
```

Figure 17. Temporal Analysis Module Definitions

4.3.4 Temporal Reasoning

The Temporal Reasoning component incrementally creates a task network representing the user's requests. This serves to verify the consistency of the task requests and to determine any relationships that were not specified in the input. Afterwards it will order the network to produce an execution sequence for the tasks to be performed. Figure 18 shows the component modules.

As already mentioned, Allen's constraint propagation algorithm is used for the temporal reasoning. It is invoked as each new relation is added to the network. However, any relation that is to replace an existing one must be more restrictive than the previous one (i.e., the method is monotonic). Another version of the algorithm is shown in Figure 19. This version makes explicit all the operations that are required for the algorithm without concern for reference intervals. For example, the inverses of each relation are stored in order to speed up the constraint propagation, since the extra storage is negligible except for very large networks.

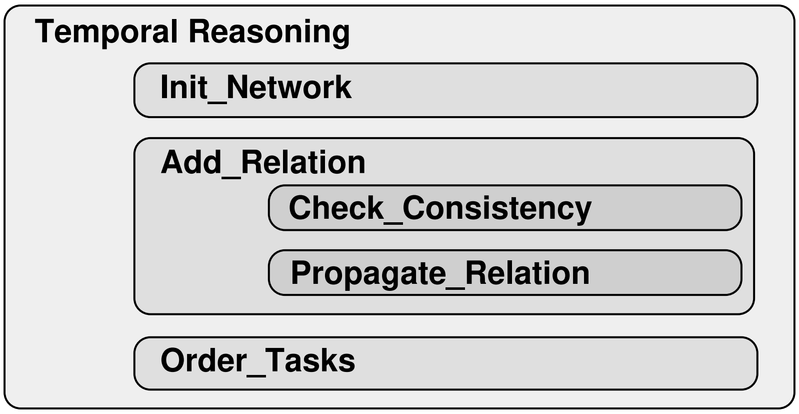


Figure 18. Temporal Reasoning Component

```
Add_Relation (in R(i,j),  inout Network, out ok)
    ok  ← true
    init_queue (ToDo)
    R(i,j)  ← N(i,j) ∩ R(i,j)
    Check_Consistency (R(i,j), N(i,j), ok)
    if ok then
        Propagate_Relation

Check_Consistency (R(i,j), N(i,j), ok)
    if R(i,j) = null_relation then
        ok ← false
    else if R(i,j) ⊂ N(i,j) then
        N(i,j) ← R(i,j)
        N(i,j) ← inverse(R(i,j))
        add_entry (ToDo, <i,j>)

Propagate_Relation
    while not empty(ToDo) do
        get_entry (ToDo, <i, j>)
        for k ← 1 to network.size do
            if k<>i and k<>j then
                R(k,j) ← N(k,j) ∩ Composition(N(k,i), N(i,j))
                Check_Consistency (R(k,j), N(k,j))
                if ok then
                    R(i,k) ← N(i,k) ∩ Composition(N(i,j), N(j,k))
                    Check_Consistency (R(i,k), N(i,k))
```

Figure 19. Another Version of Allen's Algorithm[3]

[3] This is a modification of the version found in [Song 91]

4.3.5 Response Generation

The Response Generation component (Figure 20) produces English responses based on the results of the task planning and subsequent execution. In the case of stand-alone operation, the response will just indicate the tasks that would have been performed, including the sequence of execution for those tasks. For example, in reply to

> Fill the first column of the red bench with screws
> after you hammer a nail.

the component will produce a response such as

> I will now hammer a nail into the red workbench.
> I am now decomposing the screw task.
> I will now screw a screw into column 1 of the red workbench.
> I will now screw a screw into column 1 of the red workbench.
> I will now screw a screw into column 1 of the red workbench.

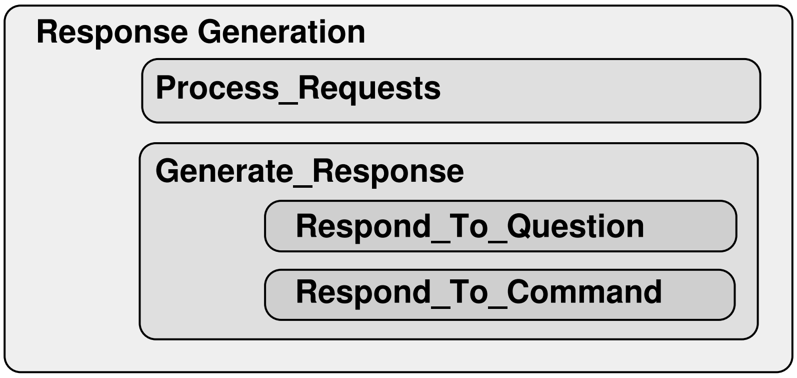


Figure 20. Response Generation Component

4.3.6 Planner Interface

The Planner Interface component (Figure 21) creates a set of task requests for the planner, usually C-barp. After sending the requests, it will interpret the responses returned by the planner. For example, the previous set of requests would have been sent to the planner as

```
From:Tempus To:C-barp #Msgs:4 Listen_To:C-barp
1.command.hammer..nail..hammer.99.99.red.w_b_hole.
2.command.screw..screw...99.1.red.w_b_hole.
3.command.screw..screw...99.1.red.w_b_hole.
4.command.screw..screw...99.1.red.w_b_hole.
```

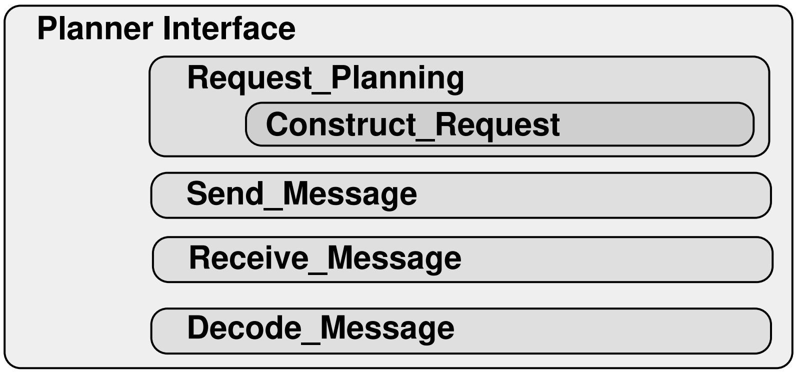


Figure 21. Planner Interface Component

5 Analysis

5.1 Implementation Data

Tempus has been implemented in C with emphasis on portability. Therefore the standalone version will run under most environments. However, the interface to MCAP uses message queues. Therefore, the Planner Interface component will be restricted to Unix environments, or ones very similar. This is no concern since the control package components of MCAP are obviously hardware specific[1].

The full version of Tempus is currently implemented on a DECstation 2100 with 12 megabytes of main memory and running Ultrix version 4.2 (Digital's version of Unix). This workstation runs at about 10.6 MIPS[2] for an optimized implementation of Koomen's random node test. In addition, the standalone version has been implemented on several types of IBM PC-compatible computers. For instance, a Gateway 2000 486/33C runs the test at approximately 10 MIPS, a Compaq Deskpro 286 runs it at just below 1 MIPS, and a Tandy 1100 FD runs it at under ½ MIPS. For this particular test, these MIPS values are good indicators of performance.

Currently, the line counts for the system components are as shown in Table 8. Note that the counts for the Discourse Analysis and Temporal Analysis components are estimates since the two overlap. In addition, for each component the first count is the total number of source lines, but the second number is the total number of lines for actual code (i.e., the source

1 Furthermore, the actual robots and their harness are not very portable. Last summer, moving them down one floor from one laboratory to another required much effort.

2 MIPS stands for million of instructions per second. Since the DECstation has a RISC architecture (reduced instruction set computing), more machine instructions need to be executed compared to a computer with a traditional architecture.

minus comments and white space). After compilation, the system consists of a 160 kilobyte executable file (110 kilobytes on the PCs).

Component	***All Lines***	***Code Alone***
System Controller	842	447
Discourse Analysis	7140	3532
Temporal Analysis	380	311
Temporal Reasoning	386	267
Response Generation	1180	625
Planner Interface	479	268
common routines	321	125
common declarations	757	494
Total:	11465	6069

Table 8.   Line Counts for C Implementation of Tempus

5.2 Experimentation

5.2.1 Analysis of Natural Language Instructions

A protocol experiment was conducted as part of this research in order to determine examples of English instructions at the level of an elementary school student, which corresponds roughly to the level of the language modeled by Tempus. A group of undergraduates were asked to write down the instructions that they might give to a child concerning tasks using the MCAP workbench (shown earlier in Figure 9). Each student gave English instructions for specified tasks and also for tasks that he or she devised.

The results indicate that simple temporal adverbials (e.g., *then*) were favored over more complicated ones (e.g., *while*). However, a more controlled experiment is necessary before any firm hypotheses can be formed since some of the students treated each task as a separate unit. Nonetheless, the results suggest natural language interfaces do not require sophisticated analysis of temporal adverbials.

5.2.2 Constraint Propagation with Large Networks

The performance of the Temporal Reasoning component was tested in isolation, since it represents the bottleneck of the system. The component was run using the test developed by Johannes Koomen [1989] for illustrating the performance of the standard constraint propagation versus his automatic referencing method.

The version of the test implemented requires 1,119,447 invocations for the 101 node test[3]. The results are impressive: an optimized version of the reasoner took less than a minute for the entire test! Furthermore, the same

[3] This represents 2% more invocations than Koomen's results (see section 2.2.3). This is due to minor implementation differences, such as the replacement strategy and the random number generation. Regardless, the difference in invocations is insignificant.

test run with a 1000 node network required over a billion compositions but took less than 14½ hours to complete. The complete results are summarized in Table 9. These results show that for all practical purposes, the constraint propagation algorithm poses no problems for performance. Even for the 1000-node network, the time is not unreasonable since a human couldn't possibly perform better. Also, the storage requirement of three megabytes for this case is well within current limitations.

#Nodes	Propagations	Compositions	Relations Posted	Relations Adjusted	Elapsed Time (sec)
10	49	538	10	3	< ½
25	419	14,460	29	30	1
50	1,852	138,235	53	107	7
100	7,252	1,092,833	107	233	51
101	7,367	1,119,447	107	242	52
250	46,536	17,845,465	270	693	826
500	184,952	141,911,747	541	1,437	6,542
1000	731,725	1,119,467,871	1,106	3,027	51,210

Table 9. Results of Random Singular Relation Assignments

5.2.3 Tempus's Overall Evaluation

For the overall evaluation, four test suites of representative instructions for the MCAP domain were developed. Tempus was run against the input, producing trace information about both the task relations and the resulting task sequences. The traces for these tests appear in Appendix C. In each case the task sequence is consistent with the requests. Also, the elapsed time in each case is good, under 15 seconds per test suite. Note that recording the trace nearly doubles the elapsed time measurement. Thus Tempus usually requires less than two seconds to process a group of utterances. Therefore, Tempus is very efficient, while still producing acceptable results.

6 Discussion

6.1 Comparison of Tempus with Other Approaches

As indicated earlier, most other work in temporal analysis is concerned with the interpretation of narrative-style discourse. Since Tempus is specifically designed for task-oriented discourse, there are striking differences in the general approaches between Tempus and other systems. An exception is the work by Robin Karlin [1988] on the interpretation of recipes. Tempus and her work will be discussed first, then some of the other approaches will be considered.

The general architectures of Tempus and Karlin's system exhibit some similarities. Both use frame-based representations and process utterances in several phases: parsing with initial semantic analysis, extraction of event information, and extraction of specific temporal information. In Tempus these phases are interleaved, whereas they are sequential in her approach. Interleaved execution is preferable since inconsistent specifications can be detected without requiring full semantic processing. It also seems more cognitively plausible, but this is not of critical importance for basic natural language interfaces.

In addition, the use of domain-knowledge is crucial for the successful operation of both systems. Karlin employs a general modeling scheme to describe the objects and actions that are involved in cooking. Therefore, she models additional information over the requirements for the application. Tempus instead uses a representation at a level that is specific to the particular application domain (i.e., the workbench). Thus Karlin's system is more flexible in its analysis of discourse from related domains. But Tempus is more efficient in its analysis of discourse within the domain. The tradeoff in Tempus reflects the principle of optimizing a system for the common case.

Fei Song's [1991] work is typical of the approaches that are designed for general discourse with emphasis on narratives. His system analyzes simple stories from published sources. The approach assumes that the discourse will generally be grammatically correct, so it simply rejects ungrammatical utterances. Therefore his temporal focus maintenance heuristics determine focus movement only on the basis of tense changes. Furthermore, the analysis of adverbials enforces selectional restrictions based on the lexical and grammatical aspects. For instance, progressive constructs cannot be modified by the adverbials *in* or *for*, which are reserved for events with perfective viewpoints.

In contrast, Tempus attempts to extract whatever it can from utterances, no matter how ungrammatical. Rejecting user utterances is simply counter to the goal of providing a natural language interface that is as transparent as possible. In addition, Tempus's temporal focus maintenance relies on several types of clues, such as anaphoric event references. This is especially important for task-oriented discourse since tense variation occurs infrequently and since tense distinctions are less precise.

Michael Almeida [1987] also concentrates on the analysis of narrative-style discourse. His system analyzes fragments taken mainly from mystery novels. Therefore some of the heuristics incorporated are specific to narratives. But he does point out the differences required for modeling general discourse. In addition, the emphasis is more on the interpretation of temporal adverbials and connectives than on tense. Therefore, outside of the application area, there are similarities between his approach and Tempus. In fact, his work served as a general model for Tempus, because the analysis of temporal adverbials and connectives is most important for task-oriented discourse. However, in Tempus, the interpretation is simplified in order to provide uniform treatment of the adverbials and connectives. Furthermore, the simplification allows for a wider range of tenses, although subtle distinctions are ignored.

Many of the differences just mentioned are the result of implementation decisions. For example, the choice of the application area or discourse domain greatly influences the approach taken. Thus Tempus pays little attention to past tense constructions, which are not common in simple natural language instructions. Furthermore, many of the previous researchers are concerned with the elucidation of principles for computational linguistics. In these cases, the application domains are mainly for illustration of the concepts.

Some differences are due to the purpose of the model developed. For example, Marc Moens and Mark Steedman [1987, 1988] contributed a linguistic theory for the semantics of temporal reference. They concentrated on discriminating the subtle effects of similar natural language constructions, such as the uses of *when* and *while*. Similarly, Michael Brent [1989, 1990] developed constraints for a theory of temporal relations among events. He analyzed the compatibility of temporal connectives and tenses. Likewise, Bonnie Webber [1987, 1988] expanded upon the theory of temporal anaphoric reference in discourse. She illustrated the effects of tense on the focus of a dialog. On the other hand, Tempus is not a model developed for a linguistic theory. But it does illustrate the mechanics of language use, plain and simple.

6.2 Limitations and Advantages of the Approach

There are several limitations in the model developed for Tempus. Obviously, it makes little concession to the requirements for general discourse. Therefore it is not suitable for narratives or argumentative discourse. For instance, it will not be able to distinguish the differences in complicated references to past occurrences. For most task-oriented applications, this is of no concern, except for certain advanced applications like machine learning.

Another limitation is the level of representation used for domain knowledge. Although Tempus reasons using event and object classes, the representation is specific to the type of application. For instance, the contingency relations apply mainly to the use of tools. These will not be useful in areas using similar concepts but different objects, such as the use of utensils. However, since it is difficult to capture the semantics of specific domains, it will be even harder to achieve this with general representations.

The main benefit of Tempus is its effective temporal analysis which is accomplished in a simple and uniform manner. Its implementation as a part of MCAP shows that temporal analysis and reasoning can be very efficient and yet capture a wide variety of discourse. Furthermore this simplicity makes Tempus readily applicable to specific task-oriented domains, as long as the basic semantic analysis has been developed.

Another benefit is the emphasis on language use rather than on grammatical correctness. As mentioned previously, this is important since people usually are not strict about their usage when speaking. In particular, when giving instructions, people are often more concerned with using clear expressions instead of felicitous ones.

6.3 Goals Attained

The goals established for Tempus are reproduced in Figure 22. Tempus is a successful extension of temporal analysis and reasoning to task-oriented domains. As part of MCAP, it uses knowledge about the workbench to facilitate the processing (goals 1, 5, 6, 8 & 10). The basic constraint propagation algorithm is unmodified. But Tempus includes a heuristic technique for task sequencing (a shift in the emphasis of goal 5).

Tempus models a wide variety of temporal adverbials and connectives for task specifications (goal 2). It uses heuristics such as contingency relations to strengthen the simplified analysis of the adverbials (goal 3). In addition, events are represented by intervals including the time range and other parameters (goal 4). However, it does not include hypothetical temporal reasoning since this is not essential for the temporal analysis (goal 7). Finally, it was not applied to any other domain (goal 9) due to time constraints.

1. To extend temporal analysis to task-oriented domains, specifically the MCAP workbench cell.
2. To provide a broad treatment of temporal adverbials for the specification of tasks.
3. To develop domain-specific heuristics to facilitate temporal analysis.
4. To determine a suitable representation of events for tasks in the domain.
5. To extend temporal reasoning to the workbench domain with emphasis on restricting the possible relations among tasks (by task classifications).
6. To investigate methods of expediting the search of temporal orderings.
7. To determine if it is feasible to include hypothetical temporal reasoning as part of the temporal analysis.
8. To explore the use of other mechanisms to increase the reasoning capability of the Tempus system.
9. To show how this approach can be applied to other natural language processing applications, such as the interpretation of recipes.
10. To conduct experiments to provide quantitative assessments of the performance of Tempus (e.g., scalability) in temporal analysis & reasoning.

Figure 22. Goals for Research and Development

6.4 Suggestions for Future Work

It is desirable to extend Tempus's discourse processing to handle parenthetical comments. But to support these types of utterances, mechanisms must be developed for converting the comments into actions performed in the domain. Thus, it is an ideal application for machine learning techniques. For example, the evaluation of previous actions would be helpful:

> That was ok, but you made some minor errors.
> First, you started hammering the nail too soon.
> And you applied too much torque when fastening that blue screw.
> Do it again, but please be careful.

The first comment indicates that the actions can serve as a positive instance for the associated problem solving. The next two comments indicate the actions that need improvement. Note that the first improvement can be achieved by modifying the heuristics used for ordering the tasks. Unfortunately, existing machine learning techniques are still too weak. But the remaining suggestions are more feasible.

An immediate extension to Tempus would be the analysis of written instructions. The basic model should still apply, but some of the assumptions would no longer be valid. Feedback during the processing would no longer be guaranteed. In addition, the instructions would use more sophisticated language constructs. Therefore the temporal analysis and reasoning would have to be more precise.

Finally, the application of this approach to the temporal analysis of general discourse would be an interesting experiment. By design, Tempus is not suitable for making fine distinctions in the temporal ordering of events. However it would be an excellent tool for extracting the general flow of the events. For instance, with its broad treatment of adverbials specifying dates, Tempus would be helpful in the analysis of historical accounts.

7 Conclusion

The treatment of time is required for many intelligent tasks. For example, the analysis of instructions is incomplete without including the intended temporal ordering. The Tempus system was designed to perform temporal analysis and reasoning in task-oriented domains, specifically as an interface to MCAP. In addition to accepting English descriptions of the tasks to be performed, Tempus permits the user to specify the ordering using temporal constructs such as the connectives *before* and *while* or the adverbials *at NP* and *on NP*.

Like much work in temporal analysis, Tempus distinguishes the event time, reference time, and speech time of a task. It models tasks as intervals specified by parameters for the acceptable date and time ranges, the duration and extent, and also the number of repetitions to be performed. An explicit event decomposition is not supported. But the modification of the reference time yields an acceptable approximation. Furthermore, domain-specific knowledge is applied to refine the event time relationships inferred through Tempus's simplified analysis of temporal adverbials. For instance, by using the contingency relations, *when* connective will be given an acceptable interpretation.

In Tempus, the temporal reasoning is based on the propagation of the interval relations derived from the specification. The standard constraint propagation technique is used without modification. This technique verifies the specification's consistency and determines its implicit relations. Then, using precedence relations, the resulting network is ordered. The precedence relations include domain-specific knowledge such as the contingency relations. Finally, the actual starting times for the tasks are established.

Tempus, is unlike most other work in temporal analysis for several reasons. First, the analysis of tense receives less attention than the analysis of temporal adverbials and connectives. Thus the Tempus approach provides a useful model for these important constructs. In addition, it is practical. Since Tempus has been designed for task-oriented discourse, there are more immediate uses for it over work that analyzes narratives. Also, the system emphasizes language use over proper usage. This is important for the users of natural language interfaces. Furthermore, Tempus is efficient. To achieve this efficiency, subtle distinctions were ignored in order to retain a simple uniform mechanism.

With regard to temporal reasoning, Tempus illustrates important points. First, the experimentation with large-scale constraint propagation shows that the standard algorithm can still be effective with hundreds of events. Also, since the algorithm is well-suited for parallel processing, even greater numbers of events can be handled. The task sequencing is kept separate, so it does not add complexity to the constraint propagation, allowing for better compiler optimization. Finally, it uses the contingency relations applied in temporal analysis. Thus, domain knowledge is incorporated uniformly.

The specific combination of temporal analysis and reasoning in Tempus represents an important application that illustrates several AI techniques. It shows that effective discourse processing of a rich subset of English can be achieved without resorting to time-consuming backtracking. It also shows that temporal reasoning will not adversely affect performance. In short, it adds new meaning to the proverb *tempus fugit.*

Appendix A: TODUS Dialog Examples

This appendix contains sample TODUS dialogs, with the user input shown in italics. These have been taken from transcripts of actual runs. These examples are also applicable to Tempus, except that grammatical warnings are issued only if a run-time option is set.

First insert a nail, then a screw.

I will now hammer a nail.
I will now screw a screw.

Hammer a nail into the upper left corner of the blue bench. Now insert a screw into the hole in the third column, second row. Then into column 1, row 3.

I will now hammer a nail into row 1, column 1 of the blue workbench.
I will now screw a screw into row 2, column 3 of the blue workbench.
I will now screw a screw into row 3, column 1 of the blue workbench.

Use the red hammer to insert a green nail into the blue bench. With the blue screwdriver, screw a green screw. Into the red bench. Upper right corner.

I will now hammer a green nail into the blue workbench using the red hammer.
I will now screw a green screw into row 1, column 5 of the red workbench using the blue screwdriver.

Use the green hammer to hammer a red nail. Now use it to hammer a blue nail. Now use it to insert a screw.

Your utterance:
now use it to insert a screw
cannot be executed because a screw cannot be inserted with a hammer.

I will now hammer a red nail using the green hammer.
I will now hammer a blue nail using the green hammer.

Grab the red hammer and use it to hammer a nail into the blue bench.

I will now hammer a nail into the blue workbench using the red hammer.

Hammer a green nail into the red bench. Now another one into the blue one.

I will now hammer a green nail into the red workbench.
I will now hammer another green nail into the blue workbench.

Screw a screw into the center right hole of the red bench. Now into the upper center.

Your utterance:
now into the upper center
is not grammatically correct, but I have understood as much of the utterance as possible.

I will now screw a screw into row 2, column 5 of the red workbench.
I will now screw a screw into row 1, column 3 of the red workbench.

Hammer a red. Screw a screw into the blue.

Your utterance:
hammer a red
is not grammatically correct, but I have understood as much of the utterance as possible.
Your utterance:
screw a screw into the blue
is not grammatically correct, but I have understood as much of the utterance as possible.

I will now hammer a red nail.
I will now screw a screw into the blue workbench.

Would you please hammer a?

Your utterance:
would you please hammer a
is not grammatically correct, but I have understood as much of the utterance as possible.

I will now hammer a nail.

Hammer a nail into the bottom lower row of the blue bench.

Your utterance:
hammer a nail into the bottom lower row of the blue bench
contains redundant information: 'bottom lower' row.

I will now hammer a nail into row 3 of the blue workbench.

Hammer 3 nails. Insert them into the first row of the blue bench.

I will now hammer a nail into row 1 of the blue workbench.
I will now hammer another nail into row 1 of the blue workbench.
I will now hammer another nail into row 1 of the blue workbench.

Fill the second column of the red bench with nails.

I will now hammer a nail into column 2 of the red workbench.
I will now hammer another nail into column 2 of the red workbench.
I will now hammer another nail into column 2 of the red workbench.

Hammer a nail into the red bench. The blue one. Now the red one.

I will now hammer a blue nail into the red workbench.
I will now hammer the red nail into the red workbench.

Is the green nail in the blue bench? Is there a red screw in the bench?

No, I don't see any green nails in the blue workbench.
Yes, I see a red screw in the blue workbench.

Are there any blue fasteners in the red bench? In the blue bench?

Your utterance:
in the blue bench
is not grammatically correct, but I have understood as much of the utterance as possible.

Yes, I see a blue fastener in the red workbench.
Yes, I see blue fasteners in the blue workbench.

How many nails are there in the blue bench? How many in the red one?

Your utterance:
how many in the red one

is not grammatically correct, but I have understood as much of the utterance as possible.
I see 2 nails in the blue workbench.
I see 3 nails in the red workbench.

How many blue nails are there in the blue bench? Where are they?

I see 2 blue nails in the blue workbench.
There is:
  a blue nail in row 1, column 2 of the blue workbench.
  a blue nail in row 3, column 1 of the blue workbench.

Hammer a workbench into a nail.

Your utterance:
  hammer a workbench into a nail
cannot be executed because 'workbench' is not a valid fastener.
Your utterance:
  hammer a workbench into a nail
cannot be executed because 'nail' is not a valid destination.

Hammer a blue green nail. Hammer a nail into the red bench into the blue bench.

Your utterance:
  hammer a blue green nail
cannot be accepted because 'blue green' is not a valid nail color
What color nail are you thinking of -
  blue?
  red?
  green?
Your utterance:
  hammer a nail into the red bench into the blue bench
cannot be accepted because 'red blue' is not a valid bench color
What color bench are you thinking of -
  blue?
  red?

I will now hammer a blue nail.
I will now hammer a nail into the red workbench.

First, insert a screw into the center of the blue bench, and then hammer a nail into the center corner.

Your utterance:
  then hammer a nail into the center corner
cannot be accepted because 'center' is not a valid corner row.
Which row are you thinking of?
  upper?
  lower?
Your utterance:
  then hammer a nail into the center corner
cannot be accepted because 'center' is not a valid corner column.
Which column are you thinking of?
  left?
  right?

I will now screw a screw into row 2, column 3 of the blue workbench.
I will now hammer a nail into row 1, column 1 of the blue workbench.

Hammer a nail into either bench. Hammer a nail into the red bench, anywhere.
Screw a screw somewhere in the upper row of the blue bench.

I will now hammer a nail into one of the workbenches.
I will now hammer a nail into one of the holes in the red workbench.
I will now screw a screw into one of the holes in row 1 of the blue workbench.

Appendix B: Tempus Lexicon

This appendix lists the entire lexicon for Tempus. Each line shows a word entry that corresponds to the LexiconEntry data structure shown previously in Figure 13. For words that have the temporal tag set, the listing includes a readable version of the relation that is specified in the code field.

Word	Tags	PrimitiveCode	(Relation)
1 :	{ det num col_pos row_pos }	1	
10 :	{ det num }	10	
12 :	{ det num }	12	
15 :	{ det num }	15	
2 :	{ det num col_pos row_pos }	2	
2001 :	{ det num }	2001	
3 :	{ det num col_pos row_pos }	3	
30 :	{ det num col_pos row_pos }	30	
4 :	{ det num col_pos }	4	
5 :	{ det num col_pos }	5	
6 :	{ det num }	6	
7 :	{ det num }	7	
8 :	{ det num }	8	
9 :	{ det num }	9	
a :	{ det }	1	
after :	{ prep fill date time temp }	2	{ > }
am :	{ noun time }	99	
an :	{ det }	1	
and :	{ delim fill }	99	
another :	{ det }	1	
any :	{ adj det col_pos row_pos }	0	
anywhere :	{ prep src dest hole }	0	
are :	{ verb ques }	99	
area :	{ noun src }	1	
at :	{ prep time temp }	5380	{ d s f = }
august :	{ noun date }	99	
back :	{ adj row_pos }	1	
bang :	{ verb command }	1	
be :	{ aux }	99	
before :	{ prep fill date time temp }	1	{ < }
bench :	{ noun dest }	1	
blue :	{ adj color }	1	
bottom :	{ adj row_pos }	3	
but :	{ delim fill }	99	
can :	{ aux }	99	
center :	{ adj noun num hole col_pos row_pos }	6	
column :	{ noun column }	99	
corner :	{ noun hole }	99	
could :	{ aux }	99	
drive :	{ verb command }	3	
during :	{ prep fill date temp }	1284	{ d s f }
eight :	{ det num }	8	
either :	{ adj color }	0	
fasten :	{ verb command }	3	

fastener :	{ noun fast }	99	
fasteners :	{ noun plur fast }	99	
fifth :	{ plur num col_pos }	5	
fill :	{ verb command }	5	
filling :	{ verb command prog }	5	
first :	{ adj num col_pos row_pos fill }	1	
five :	{ det num col_pos }	5	
four :	{ det num col_pos }	4	
fourth :	{ adj num col_pos }	4	
from :	{ prep src }	99	
front :	{ adj row_pos }	1	
get :	{ verb command }	4	
grab :	{ verb command }	4	
grasp :	{ verb command }	4	
green :	{ adj color }	3	
hammer :	{ noun verb command tool }	1	
hammered :	{ noun verb command tool }	1	
hammering :	{ noun verb command tool prog }	1	
have :	{ aux }	99	
hit :	{ verb command }	1	
hold :	{ verb command }	6	
holding :	{ verb command prog }	6	
hole :	{ noun hole }	99	
holes :	{ noun plur hole }	99	
how :	{ num ques }	99	
in :	{ prep dest }	99	
insert :	{ verb command }	3	
into :	{ prep dest }	99	
is :	{ verb ques }	99	
it :	{ noun pron }	99	
last :	{ adj col_pos }	5	
left :	{ adj col_pos }	1	
left-hand :	{ adj col_pos }	1	
lefthand :	{ adj col_pos }	1	
lower :	{ adj row_pos }	3	
many :	{ }	99	
measure :	{ verb command }	7	
measuring :	{ verb command prog }	7	
middle :	{ adj noun num hole col_pos row_pos }	6	
nail :	{ noun verb command fast }	1	
nails :	{ noun plur fast }	1	
next :	{ delim new_action }	99	
nine :	{ det num }	9	
now :	{ delim fill new_action }	99	
of :	{ prep dest }	99	
on :	{ prep fill date temp }	1284	{ d s f }
one :	{ det noun pron one }	1	
pick :	{ verb command }	4	
place :	{ verb command }	3	
please :	{ fill }	99	
pm :	{ noun time }	99	
pound :	{ verb command }	1	
put :	{ verb command }	3	
red :	{ adj color }	2	
right :	{ adj col_pos }	5	
right-hand :	{ adj col_pos }	5	
righthand :	{ adj col_pos }	5	
row :	{ noun row }	99	

saturday :	{ noun date }	99	
screw :	{ noun verb command fast }	2	
screwed :	{ noun verb command fast }	2	
screwdriver :	{ noun tool }	2	
screwing :	{ verb command prog }	2	
screws :	{ noun plur fast }	2	
second :	{ adj num col_pos row_pos }	2	
seven :	{ det num }	7	
side :	{ noun column }	99	
six :	{ det num }	6	
some :	{ adj det col_pos row_pos }	0	
somewhere :	{ prep src dest hole }	0	
space :	{ noun hole }	99	
take :	{ verb command }	4	
ten :	{ det num }	10	
the :	{ det }	1	
them :	{ noun pron plur }	99	
then :	{ delim fill new_action }	99	
there :	{ }	99	
they :	{ noun pron plur }	99	
third :	{ adj num col_pos row_pos }	3	
three :	{ det num col_pos row_pos }	3	
to :	{ aux }	99	
top :	{ adj row_pos }	1	
tuesday :	{ noun date }	99	
turn :	{ verb command }	2	
two :	{ det num col_pos row_pos }	2	
up :	{ }	99	
upper :	{ adj row_pos }	1	
use :	{ prep tool }	99	
using :	{ prep tool }	99	
when :	{ delim fill temp }	4096	{ = }
where :	{ ques dest }	99	
while :	{ fill temp }	4096	{ = }
white :	{ adj color }	5	
will :	{ aux }	99	
with :	{ prep tool }	99	
workarea :	{ noun src }	1	
workbench :	{ noun dest }	1	
would :	{ aux }	99	
yellow :	{ adj color }	4	
you :	{ aux }	99	

Appendix C: Extended Traces of Tempus Examples

In this appendix, several traces are reproduced for the standalone version of Tempus. For most of the examples, the network is displayed after each relation is added. However, the first trace shows the treatment of two complicated task decompositions, requiring a 32-node task network. Therefore, the network is displayed only once at the end. Note that the elapsed times are roughly double the time required when the tracing is not in effect. In addition, the test suites are reproduced below for convenience. For each one, the timing information shows the more accurate elapsed times. Considering the complexity of some of the examples, Tempus is therefore very efficient.

```
test suite 1:
        Fill the red bench with screws while you fill the blue one with nails.
        2.081u 1.406s 0:07.36 47.2% 63+206k 1+1io 5pf+0w

test suite 2:
        when you hammer a nail in the red bench measure the bench
        when you hammer a nail in the red bench be measuring a screw
        when hammering a nail in the red bench measure the bench
        when hammering a nail in the red bench be measuring a screw
        0.777u 1.031s 0:05.03 35.7% 71+188k 3+2io 15pf+0w

test suite 3:
        When screwing three screws in the third column of the blue bench,
        hammer a nail in the first column.
        After august, hammer a nail.
        And screw the screw before 2 pm.
        Before screwing a screw in the lower lefthand corner of the blue bench,
        screw one in the second row.
        Screw in a red screw in the blue bench before you hammer a nail.
        And hammer the nail, before you fasten a blue screw in the red bench.
        But fasten the blue screw before you screw the red one.
        0.578u 0.945s 0:04.26 35.4% 67+175k 3+2io 14pf+0w

test suite 4:
        screw a red screw on 15 august 2001
        hammer a nail
        hammer a nail
        0.414u 0.539s 0:02.57 36.5% 58+152k 2+2io 5pf+0w
```

```
% time tempus tstsuite.fill

                Welcome  to  Tempus

        A System for Temporal Analysis & Reasoning

         Developed at East Stroudsburg University

              As an Extension to TODUS

      The Task Oriented Discourse Understanding System

Do you want a trace?

Current user input:
-------------------
```

Fill the red bench with screws while you fill the blue one with nails.

```
ORDERING Tasks:
TEMPUS Response:
I will now  row 3, the red workbench.
I will now  row 3, the blue workbench.
I will now  row 3, the blue workbench.
I will now hammer a nail into row 3, the blue workbench.
I will now  row 1, the red workbench.
I will now screw a screw into row 1, the red workbench.
I will now  row 3, the red workbench.
I will now screw a screw into row 3, the red workbench.
I will now screw a screw into row 1, the red workbench.
I will now screw a screw into row 1, the red workbench.
I will now screw a screw into row 1, the red workbench.
I will now screw a screw into row 1, the red workbench.
I will now  row 2, the red workbench.
I will now screw a screw into row 2, the red workbench.
I will now screw a screw into row 2, the red workbench.
I will now screw a screw into row 2, the red workbench.
I will now screw a screw into row 2, the red workbench.
I will now screw a screw into row 2, the red workbench.
I will now screw a screw into row 3, the red workbench.
I will now screw a screw into row 3, the red workbench.
I will now screw a screw into row 3, the red workbench.
I will now screw a screw into row 3, the red workbench.
I will now  row 1, the blue workbench.
I will now hammer a nail into row 1, the blue workbench.
I will now hammer a nail into row 1, the blue workbench.
I will now hammer a nail into row 1, the blue workbench.
I will now hammer a nail into row 1, the blue workbench.
I will now hammer a nail into row 1, the blue workbench.
I will now  row 2, the blue workbench.
I will now hammer a nail into row 2, the blue workbench.
I will now hammer a nail into row 2, the blue workbench.
I will now hammer a nail into row 2, the blue workbench.
I will now hammer a nail into row 2, the blue workbench.
I will now hammer a nail into row 2, the blue workbench.
I will now hammer a nail into row 3, the blue workbench.
I will now hammer a nail into row 3, the blue workbench.
I will now hammer a nail into row 3, the blue workbench.
I will now hammer a nail into row 3, the blue workbench.
```

<u>2.132u 1.730s 0:11.28 34.2% 63+205k 6+1io 5pf+0w</u>

The network that resulted:

1-2: { f fi = } 1-3: { si } 1-4: { si }
1-5: { di } 1-6: { di } 1-7: { di }
1-8: { di } 1-9: { di } 1-10: { di }
1-11: { di } 1-12: { di } 1-13: { di }
1-14: { di } 1-15: { fi } 1-16: { di }
1-17: { di } 1-18: { di } 1-19: { di }
1-20: { fi } 1-21: { > di oi mi si } 1-22: { > di oi mi si }
1-23: { > di oi mi si } 1-24: { > di oi mi si } 1-25: { > di oi mi si }
1-26: { > di oi mi si } 1-27: { > di oi mi si } 1-28: { > di oi mi si }
1-29: { > di oi mi si } 1-30: { > di oi mi si } 1-31: { > di oi mi si }
1-32: { > di oi mi si } 1-33: { f fi = } 1-34: { > di oi mi si }
1-35: { > di oi mi si } 1-36: { > di oi mi si } 1-37: { > di oi mi si }
1-38: { f fi = } 2-3: { > di oi mi si } 2-4: { > di oi mi si }
2-5: { > di oi mi si } 2-6: { > di oi mi si } 2-7: { > di oi mi si }
2-8: { > di oi mi si } 2-9: { > di oi mi si } 2-10: { > di oi mi si }
2-11: { > di oi mi si } 2-12: { > di oi mi si } 2-13: { > di oi mi si }
2-14: { > di oi mi si } 2-15: { f fi = } 2-16: { > di oi mi si }
2-17: { > di oi mi si } 2-18: { > di oi mi si } 2-19: { > di oi mi si }
2-20: { f fi = } 2-21: { si } 2-22: { si }
2-23: { di } 2-24: { di } 2-25: { di }
2-26: { di } 2-27: { di } 2-28: { di }
2-29: { di } 2-30: { di } 2-31: { di }
2-32: { di } 2-33: { fi } 2-34: { di }
2-35: { di } 2-36: { di } 2-37: { di }
2-38: { fi } 3-4: { si } 3-5: { di }
3-6: { di } 3-7: { di } 3-8: { fi }
3-9: { < di o m fi } 3-10: { < di o m fi } 3-11: { < di o m fi }
3-12: { < di o m fi } 3-13: { < di o m fi } 3-14: { < di o m fi }
3-15: { < o m } 3-16: { < di o m fi } 3-17: { < di o m fi }
3-18: { < di o m fi } 3-19: { < di o m fi } 3-20: { < o m }
3-33: { < d o m s } 3-38: { < d o m s } 4-5: { < di o m fi }
4-6: { < di o m fi } 4-7: { < di o m fi } 4-8: { < o m }
4-9: { < di o m fi } 4-10: { < di o m fi } 4-11: { < di o m fi }
4-12: { < di o m fi } 4-13: { < di o m fi } 4-14: { < di o m fi }
4-15: { < o m } 4-16: { < di o m fi } 4-17: { < di o m fi }
4-18: { < di o m fi } 4-19: { < di o m fi } 4-20: { < o m }
4-33: { < d o m s } 4-38: { < d o m s } 5-8: { < d o m s }
5-15: { < d o m s } 5-20: { < d o m s } 5-33: { < d o m s }
5-38: { < d o m s } 6-8: { < d o m s } 6-15: { < d o m s }
6-20: { < d o m s } 6-33: { < d o m s } 6-38: { < d o m s }
7-8: { < d o m s } 7-15: { < d o m s } 7-20: { < d o m s }
7-33: { < d o m s } 7-38: { < d o m s } 8-15: { < d o m s }
8-20: { < d o m s } 8-33: { < d o m s } 8-38: { < d o m s }
9-10: { si } 9-11: { di } 9-12: { di }
9-13: { di } 9-14: { fi } 9-15: { < d o m s }
9-20: { < d o m s } 9-33: { < d o m s } 9-38: { < d o m s }
10-11: { < di o m fi } 10-12: { < di o m fi } 10-13: { < di o m fi }
10-14: { < o m } 10-15: { < d o m s } 10-20: { < d o m s }
10-33: { < d o m s } 10-38: { < d o m s } 11-14: { < d o m s }
11-15: { < d o m s } 11-20: { < d o m s } 11-33: { < d o m s }
11-38: { < d o m s } 12-14: { < d o m s } 12-15: { < d o m s }
12-20: { < d o m s } 12-33: { < d o m s } 12-38: { < d o m s }
13-14: { < d o m s } 13-15: { < d o m s } 13-20: { < d o m s }
13-33: { < d o m s } 13-38: { < d o m s } 14-15: { < d o m s }
14-20: { < d o m s } 14-33: { < d o m s } 14-38: { < d o m s }
15-16: { si } 15-17: { di } 15-18: { di }
15-19: { di } 15-20: { fi } 15-21: { > di oi mi si }
15-22: { > di oi mi si } 15-23: { > di oi mi si } 15-24: { > di oi mi si }
15-25: { > di oi mi si } 15-26: { > di oi mi si } 15-27: { > di oi mi si }
15-28: { > di oi mi si } 15-29: { > di oi mi si } 15-30: { > di oi mi si }
15-31: { > di oi mi si } 15-32: { > di oi mi si } 15-33: { f fi = }
15-34: { > di oi mi si } 15-35: { > di oi mi si } 15-36: { > di oi mi si }

15-37: { > di oi mi si } 15-38: { f fi = } 16-17: { < di o m fi }
16-18: { < di o m fi } 16-19: { < di o m fi } 16-20: { < o m }
16-33: { < d o m s } 16-38: { < d o m s } 17-20: { < d o m s }
17-33: { < d o m s } 17-38: { < d o m s } 18-20: { < d o m s }
18-33: { < d o m s } 18-38: { < d o m s } 19-20: { < d o m s }
19-33: { < d o m s } 19-38: { < d o m s } 20-21: { > di oi mi si }
20-22: { > di oi mi si } 20-23: { > di oi mi si } 20-24: { > di oi mi si }
20-25: { > di oi mi si } 20-26: { > di oi mi si } 20-27: { > di oi mi si }
20-28: { > di oi mi si } 20-29: { > di oi mi si } 20-30: { > di oi mi si }
20-31: { > di oi mi si } 20-32: { > di oi mi si } 20-33: { f fi = }
20-34: { > di oi mi si } 20-35: { > di oi mi si } 20-36: { > di oi mi si }
20-37: { > di oi mi si } 20-38: { f fi = } 21-22: { si }
21-23: { di } 21-24: { di } 21-25: { di }
21-26: { fi } 21-27: { < di o m fi } 21-28: { < di o m fi }
21-29: { < di o m fi } 21-30: { < di o m fi } 21-31: { < di o m fi }
21-32: { < di o m fi } 21-33: { < o m } 21-34: { < di o m fi }
21-35: { < di o m fi } 21-36: { < di o m fi } 21-37: { < di o m fi }
21-38: { < o m } 22-23: { < di o m fi } 22-24: { < di o m fi }
22-25: { < di o m fi } 22-26: { < o m } 22-27: { < di o m fi }
22-28: { < di o m fi } 22-29: { < di o m fi } 22-30: { < di o m fi }
22-31: { < di o m fi } 22-32: { < di o m fi } 22-33: { < o m }
22-34: { < di o m fi } 22-35: { < di o m fi } 22-36: { < di o m fi }
22-37: { < di o m fi } 22-38: { < o m } 23-26: { < d o m s }
23-33: { < d o m s } 23-38: { < d o m s } 24-26: { < d o m s }
24-33: { < d o m s } 24-38: { < d o m s } 25-26: { < d o m s }
25-33: { < d o m s } 25-38: { < d o m s } 26-33: { < d o m s }
26-38: { < d o m s } 27-28: { si } 27-29: { di }
27-30: { di } 27-31: { di } 27-32: { fi }
27-33: { < d o m s } 27-38: { < d o m s } 28-29: { < di o m fi }
28-30: { < di o m fi } 28-31: { < di o m fi } 28-32: { < o m }
28-33: { < d o m s } 28-38: { < d o m s } 29-32: { < d o m s }
29-33: { < d o m s } 29-38: { < d o m s } 30-32: { < d o m s }
30-33: { < d o m s } 30-38: { < d o m s } 31-32: { < d o m s }
31-33: { < d o m s } 31-38: { < d o m s } 32-33: { < d o m s }
32-38: { < d o m s } 33-34: { si } 33-35: { di }
33-36: { di } 33-37: { di } 33-38: { fi }
34-35: { < di o m fi } 34-36: { < di o m fi } 34-37: { < di o m fi }
34-38: { < o m } 35-38: { < d o m s } 36-38: { < d o m s }
37-38: { < d o m s }

Check called 8055 times
301 propagations
8020 compositions

Script started on Fri Aug 14 00:52:41 1992
% time tempus debug tstsuite.when
Fri Aug 14 00:52:57 1992

Current user input:
----
when you hammer a nail in the red bench
measure the bench

ADD Relation 2-1: { = }
1-2: { = }
1-2: { = }

Check called 1 times
1 propagations
2 compositions
ORDERING Tasks:
Task 1 goes in slot 1
Task 2 goes in slot 2
TEMPUS Response:
I will now hammer a nail into the red workbench.
I will now measure the red workbench.
Fri Aug 14 00:53:01 1992

Current user input:
----
when you hammer a nail in the red bench
be measuring a screw

ADD Relation 2-1: { di }
1-2: { d }
1-2: { d }

Check called 1 times
1 propagations
2 compositions
ORDERING Tasks:
Task 1 goes in slot 2
Task 2 goes in slot 1
TEMPUS Response:
I will now measure .
I will now hammer a nail into the red workbench.
Fri Aug 14 00:53:03 1992

Current user input:
----
when hammering a nail in the red bench
measure the bench

ADD Relation 2-1: { d s f }
1-2: { di si fi }
1-2: { di si fi }

Check called 1 times
1 propagations
2 compositions
ORDERING Tasks:
Task 1 goes in slot 1
Task 2 goes in slot 2
TEMPUS Response:
I will now hammer a nail into the red workbench.
I will now measure the red workbench.
Fri Aug 14 00:53:09 1992

Current user input:
----
when hammering a nail in the red bench
be measuring a screw

ADD Relation 2-1: { di o }
1-2: { d oi }
1-2: { d oi }

Check called 1 times
1 propagations
2 compositions
ORDERING Tasks:
Task 1 goes in slot 2
Task 2 goes in slot 1
TEMPUS Response:
I will now measure .
I will now hammer a nail into the red workbench.
Fri Aug 14 00:53:11 1992

Hasta la vista, baybay

0.765u 0.999s 0:08.99 19.4% 69+182k 9+3io 15pf+0w

% time tempus tstsuite.temporal debug
Fri Aug 14 02:56:07 1992

Current user input:

When screwing three screws in the third column of the blue bench,
hammer a nail in the first column.

ADD Relation 2-1: { d s f }
1-2: { di si fi }
ADD Relation 3-1: { s }
1-2: { di si fi } 1-3: { si } 2-3: { > d oi mi s si f = }
ADD Relation 4-1: { d }
1-2: { di si fi } 1-3: { si } 1-4: { di }
2-3: { > d oi mi s si f = } 3-4: { < di o m fi }
ADD Relation 5-1: { f }
1-2: { di si fi } 1-3: { si } 1-4: { di }
1-5: { fi } 2-3: { > d oi mi s si f = } 2-5: { < d o m s f fi = }
3-4: { < di o m fi } 3-5: { < o m } 4-5: { < d o m s }
1-2: { di si fi } 1-3: { si } 1-4: { di }
1-5: { fi } 2-3: { > d oi mi s si f = } 2-5: { < d o m s f fi = }
3-4: { < di o m fi } 3-5: { < o m } 4-5: { < d o m s }

Check called 31 times
9 propagations
29 compositions
ORDERING Tasks:
Task 1 goes in slot 1
Task 2 goes in slot 3
Task 3 goes in slot 2
Task 4 goes in slot 5
Task 5 goes in slot 4
TEMPUS Response:
I will now column 3 of the blue workbench.
I will now screw a screw into column 3 of the blue workbench.
I will now hammer a nail into column 1 of the blue workbench.
I will now screw a screw into column 3 of the blue workbench.
I will now screw a screw into column 3 of the blue workbench.
Fri Aug 14 02:56:13 1992

Current user input:

After august, hammer a nail.
And screw the screw before 2 pm.

Check called 0 times
0 propagations
0 compositions
ORDERING Tasks:
Task 1 goes in slot 2 { > } <15 Aug 1992: 0000, 31 Aug 1992: 0015>
Task 2 goes in slot 1 { < } <14 Aug 1992: 1400, 14 Aug 1992: 1415>
TEMPUS Response:
I will now screw a screw.
inactive:1.COMMAND.HAMMER..NAIL..HAMMER.99.99..W_B_HOLE.
Fri Aug 14 02:56:15 1992

Current user input:

Before screwing a screw in the lower lefthand corner of the blue bench,
screw one in the second row.

ADD Relation 2-1: { < d o m s }

Check called 1 times
1 propagations
2 compositions
ORDERING Tasks:
Task 1 goes in slot 1
Task 2 goes in slot 2
TEMPUS Response:
I will now screw a screw into row 3, column 1 of the blue workbench.
I will now screw a screw into row 2, the blue workbench.
inactive:1.COMMAND.HAMMER..NAIL..HAMMER.99.99..W_B_HOLE.
Fri Aug 14 02:56:17 1992

Current user input:

Screw in a red screw in the blue bench before you hammer a nail.
And hammer the nail, before you fasten a blue screw in the red bench.
But fasten the blue screw before you screw the red one.

ADD Relation 1-2: { < }
1-2: { < }
ADD Relation 3-2: { = }
1-2: { < } 1-3: { < } 2-3: { = }
ADD Relation 3-4: { < d o m s }
1-2: { < } 1-3: { < } 1-4: { < d o m s }
2-3: { = } 2-4: { < d o m s } 3-4: { < d o m s }
ADD Relation 5-4: { = }
1-2: { < } 1-3: { < } 1-4: { < d o m s }
1-5: { < d o m s } 2-3: { = } 2-4: { < d o m s }
2-5: { < d o m s } 3-4: { < d o m s } 3-5: { < d o m s }
4-5: { = }
ADD Relation 6-1: { = }
1-2: { < } 1-3: { < } 1-4: { < d o m s }
1-5: { < d o m s } 1-6: { = } 2-3: { = }
2-4: { < d o m s } 2-5: { < d o m s } 2-6: { > }
3-4: { < d o m s } 3-5: { < d o m s } 3-6: { > }
4-5: { = } 4-6: { > di oi mi si } 5-6: { > di oi mi si }
ADD Relation 5-6: { < }
1-2: { < } 1-3: { < } 1-4: { < d o m s }
1-5: { < d o m s } 1-6: { = } 2-3: { = }
2-4: { < d o m s } 2-5: { < d o m s } 2-6: { > }
3-4: { < d o m s } 3-5: { < d o m s } 3-6: { > }

4-5: { = } 4-6: { > di oi mi si } 5-6: { > di oi mi si }

Current user input:

Screw in a red screw in the blue bench before you hammer a nail.
And hammer the nail, before you fasten a blue screw in the red bench.
But fasten the blue screw before you screw the red one.

Your utterance:
    before you screw the red one
cannot be executed because of an inconsistent task specification
Fri Aug 14 02:56:25 1992

Later

1.074u 1.738s 0:10.18 27.5% 71+213k 9+2io 5pf+0w

% time tempus debug tstsuite.2001 x200000000
Fri Aug 14 02:56:46 1992

Current user input:
----
screw a red screw on 15 august 2001

Check called 0 times
0 propagations
0 compositions
ORDERING Tasks:
Task 1 goes in slot 1 { d s f } <15 Aug 2001: 0000, 15 Aug 2001: 0015>
TEMPUS Response:
inactive:1.COMMAND.SCREW.RED.SCREW..SCREWDRIVER.99.99..W_B_HOLE.
Tue Dec 15 21:30:07 1998

Current user input:
----
hammer a nail

Check called 0 times
0 propagations
0 compositions
ORDERING Tasks:
Task 1 goes in slot 1
TEMPUS Response:
I will now hammer a nail.
inactive:1.COMMAND.SCREW.RED.SCREW..SCREWDRIVER.99.99..W_B_HOLE.
Sun Apr 17 18:03:28 2005

Current user input:
----
hammer a nail

Check called 0 times
0 propagations
0 compositions
ORDERING Tasks:
Task 1 goes in slot 1
TEMPUS Response:
I will now screw a red screw.
I will now hammer a nail.
Sun Apr 17 18:03:28 2005

Hasta la vista, baybay

0.453u 0.511s 0:03.00 32.0% 59+153k 3+2io 5pf+0w

Glossary

Note: In the entry definitions, italics are used to indicate a cross-reference.
However, in the examples, italics are used to highlight the item being defined.

Adjunct Clause The clause containing the conjunction in a complex sentence. It is also called the dependent or subordinate clause.

When you are hammering the nail, hold the workbench.

Anaphoric Reference The reference to a previous part of a dialog. This includes the use of pronouns but also covers the use of referential phrases.

Screw in a red screw. Then hammer a nail.
But before screwing in *the red screw*, hold the ruler.

Aspect A verb characteristic that indicates whether an action is viewed as being complete (*perfective aspect*) or as being ongoing (*imperfective aspect*).

While you *are* hold*ing* the bench, fasten the screw.

C-barp (Committee-Based Architecture for planning and Re-Planning). The planning component of MCAP.

Clause A phrase in a sentence that expresses a complete thought.

-- When you are hammering the nail, *hold the workbench*.
-- *Hammer two nails into the middle row*.

Composite Task A *task* that actually specifies a series of *subtasks*.

Fill the first column with screws specifies three subtasks:
Screw a screw in row i column 1, $i \in \{1, 2, 3\}$

Consequent The part of a *nucleus event structure* that indicates the state that an event brings about.

<prepare to fasten; fasten; *have fastened*>

Contingency Relation A relation between events that indicates more than just temporal coincidence (e.g., causality or enablement).

Holding the bench *expedites* inserting a fastener.

Conversational Move A discourse construct producing a shift in the focus of a conversation (e.g., a subdialog for clarification).

You will mainly be using the red bench.
Screw in a red screw. Then hammer a nail.
You might also be using the green bench.
Fill the first row with nails and the second one with screws.
Anyway, after hammering the nail, fasten another screw.

Culmination The part of a *nucleus event structure* marking the shift to event completion (e.g., the event's climax).

<prepare to fasten; *fasten;* have fastened>

Event An abstract entity that represents either the occurrence of an action or the duration of some condition.

-- The robot fastened the screw in the upper left-hand corner.
-- There are three nails in the green bench

Event Structure The representation of an *event*: atomic event structures have no internal structure, but other event structures represent the various subparts of the event such as the climax. (See *nucleus event structure*.)

Event Time The time of the occurrence of an *event*.

Fasten the nail at *3pm* on *Monday*.

Event Type The category of a particular *event*.

Hold the ruler:	*process*
Screw the screw:	*culminated process*
The bench is full:	*state*

Imperfective Aspect Verbal *aspect* indicating that an action is ongoing. It is also called progressive aspect.

While *holding the ruler*, screw the screw.

Interval-Based Representation An approach to modeling time using a set of intervals that are partially ordered by the relations *before*, *meets*, *overlaps*, *during*, *starts*, *finishes*, and *equals*.

Matrix Clause The *clause* in a complex sentence that does not contain the conjunction. It is also called the independent or main clause.

When you are hammering the nail, *hold the workbench*.

MCAP (Multiple Cooperating Agents Project). A project at East Stroudsburg University for robot coordination.

Nonprogressive Aspect See *perfective aspect.*

Nucleus Event Structure An *event structure* that contains components for the event preparation, culmination, and consequence. It is also termed the tripartite event structure.

Hammering a nail is decomposed as
preparation: prepare to hammer the nail
culmination: the nail gets hammered
consequent: the nail having been hammered

Perfective Aspect Verbal aspect indicating that an action is complete. It is also called nonprogressive aspect.

Hammer the nail into the blue bench.

Point-Based Representation An approach to modeling time using a set of points that are ordered by the relations *before*, *after*, and *equals*.

Preparation The beginning part of a *nucleus event structure* corresponding to preliminary actions required prior to the main action.

<*prepare to fasten*; fasten; have fastened>

Progressive Aspect See *imperfective aspect.*

Reference Interval A point or *temporal interval* defining the perspective of an *event*.

event interval: <fasten the screw in the upper left corner>
reference interval: <fastening the blue screw>

Reference Time The time indicating the viewpoint of an *event.*

Speech Time The time of the utterance of an expression.

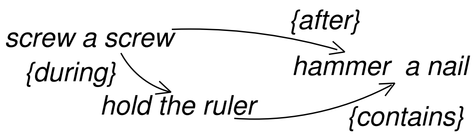
Subtask A *task* that is part of a *composite task*. It is usually an atomic task, but it can be a composite task.

"Fill the first column with screws" specifies three subtasks:
Screw a screw in row i column 1, $i \in \{1, 2, 3\}$

Task A unit of work (i.e., a series of actions required to accomplish an immediate objective).

Measure the bench with the ruler.

Task Network A graph structure representing a series of *tasks* to be performed. Each arc in the graph defines some relationship between two tasks such as one task preceding another.



Temporal Analysis The extraction of the temporal relations explicit or implicit in a series of natural language expressions.

Temporal Connective A subordinating conjunction that specifies the temporal relationship between two *clauses*.

While you fill the red bench with screws,
fill the blue one with nails.

Temporal Interval An interval that indicates the temporal extent of an *event*.

Hammer in the nail: *<start hammering . . . end hammering>*

Temporal Reasoning The determination of the complete set of temporal relationships among *events*, including the detection of inconsistencies.

TODUS (Task-Oriented Discourse Understanding System). The existing natural language interface component of *MCAP*. TODUS was the basis for the discourse analysis in Tempus.

References

Allen, James F., "Maintaining Knowledge about Temporal Intervals", *Communications of the ACM*, November 1983, pp 832-843.

Allen, James F., "Towards a General Theory of Action and Time", *Artificial Intelligence*, 23, 1984, pp 123-154.

Allen, James F., "Temporal Reasoning and Planning", in *Reasoning about Plans*, Allen, James F.; Kautz, Henry; Pelavin, Richard; and Tenenberg, Josh, editors, San Mateo, CA: Morgan Kaufmann, 1991, pp 1-68.

Almeida, Michael J., *Reasoning about the Temporal Structure of Narratives*, Ph.D. Dissertation, SUNY Buffalo, 1987.

Amori, Richard D., "A Multiple Cooperating Intelligent Agents Project Progress Report", *Proceedings of the ACM Annual Computer Science Conference*, New York: ACM Press, 1988, pp 454-461.

Amori, Richard D., "A Natural Language Interface for Task-Oriented Activities", *Proceedings of the Third International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, New York: ACM Press, 1990, pp 553-562.

Brent, Michael R., "Causal/Temporal Connectives: Syntax and Lexicon", *Proceedings of the 11th Annual Conference of the Cognitive Science Society*, 1989, pp 90-98.

Brent, Michael R., "A Simplified Theory Of Tense Representations And Constraints On Their Composition", *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics*, 1990, pp 119-126.

Dahl, Deborah A.; Palmer, Martha S.; and Passonneau, Rebecca J., "Nominalizations in Pundit", *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, 1987, pp 131-139.

Harper, Mary P.; and Charniak, Eugene, "Time and Tense in English", *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*, 1986, pp 3-9.

Karlin, Robin F., "Defining the Semantics of Verbal Modifiers in the Domain of Cooking Tasks", *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, 1988, pp 61-67.

Koomen, Johannes A., *Reasoning About Recurrence*, Ph.D. Dissertation, University of Rochester, 1989.

Moens, Marc; and Steedman, Mark, "Temporal Ontology in Natural Language", *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, 1987, pp 1-7.

Moens, Marc; and Steedman, Mark, "Temporal Ontology and Temporal Reference", *Computational Linguistics*, June 1988, pp 15-28.

Parsons, Terence, *Events In The Semantics Of English*: A Study In Subatomic Semantics, Cambridge, MA: MIT Press, 1990.

Passonneau, Rebecca J., "Situations and Intervals", *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, 1987, pp 16-24.

Passonneau, Rebecca J., "A Computational Model of the Semantics of Tense and Aspect", *Computational Linguistics*, June 1988, pp 44-60.

Percey, Deborah; and Taylor, Charles, *TODUS: A Knowledge Based Approach to a Task-Oriented Discourse Understanding System,* Joint Master's Thesis Project, East Stroudsburg University, 1986.

Peters, John, editor, "Component Communication Document: MCAP Project", Computer Science Department, East Stroudsburg University, 1989.

Quirk, Randolph; Greenbaum, Sidney; Leech, Geoffrey; and Jan Svartvik, *A Comprehensive Grammar of the English Language*, London: Longman, 1985.

Reichenbach, Hans, *Elements of Symbolic Logic*, New York: Macmillan, 1947.

Reichman, Rachel, *Getting Computers to Talk Like You and Me*: Discourse Context, Focus, and Semantics (An ATN Model), Cambridge, MA: MIT Press, 1985.

Resuta, Dave, *Committee-Based Architecture for Planning and Re-Planning*, Master's Thesis, East Stroudsburg University, 1991.

Song, Fei, *A Processing Model for Temporal Analysis and its Application to Plan Recognition*, Ph.D. Dissertation, University of Waterloo, 1991.

Song, Fei; and Cohen, Robin, "Temporal Reasoning During Plan Recognition", *Proceedings AAAI-91*, pp 247-252.

Song, Fei; and Cohen, Robin, "Tense Interpretation in the Context of Narrative", *Proceedings AAAI-91*, pp 131-136.

Vilain, Marc, and Kautz, Henry, "Constraint Propagation Algorithms for Temporal Reasoning", *Proceedings AAAI-86:* Fifth National Conference on Artificial Intelligence, 1986, pp 377-382.

Vilain, Marc; Kautz, Henry; and van Beek, Peter, "Constraint Propagation Algorithms for Temporal Reasoning: A Revised Report", in *Readings in Qualitative Reasoning about Physical Systems*, Daniel S. Weld, and Johan de Kleer, editors, San Mateo, CA: Morgan Kaufmann, 1990, pp 373-381.

Webber, Bonnie, "The Interpretation of Tense in Discourse", *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, 1987, pp 147-154.

Webber, Bonnie, "Tense as Discourse Anaphora", *Computational Linguistics*, June 1988, pp 61-73.

Index

Abstract

Full consideration of time is required for many intelligent activities. For example, the analysis of instructions is incomplete without inferring the temporal order of the tasks. ***Tempus*** performs temporal analysis and reasoning in task-oriented domains. First, it interprets English task descriptions using a variety of temporal adverbials. Then, it produces an acceptable task ordering.

Tempus models a task as an event with a time interval augmented by various parameters such as the acceptable time range. Since the actual event time is distinguished from the time of its reference, the effects of tense and aspect can easily be modeled. Temporal adverbials are modeled uniformly in their effect upon the reference time. Furthermore, domain-specific knowledge is applied to refine event time relationships resulting from the simplified analysis. Temporal reasoning is based on constraint propagation. This verifies consistency and determines implicit relations. Then, using precedence relations, the task network is ordered. Finally, the actual starting times for the tasks are established.

Thesis written by

Thomas P. O'Hara

Bachelor of Science in Computer Science
Penn State University, 1989

Master of Science in Computer Science
East Stroudsburg University, 1992

Approved by

______________________________________ Thesis Chairman
Richard D. Amori

______________________________________ Member, Advisory Committee
Felix Friedman

______________________________________ Member, Advisory Committee
Richard G. Prince

Summary

Tempus efficiently performs temporal analysis and reasoning in task-oriented domains. It models a task as an event with a time interval augmented by various parameters. Since the actual event time is distinguished from the time of its reference, the effects of tense and aspect are modeled. Temporal adverbials are treated uniformly. Furthermore, domain-specific knowledge is applied to refine the simplified analysis. Temporal reasoning is based on constraint propagation. This verifies consistency and determines implicit relations. Then, using precedence relations, the task network is ordered.