A Processing Framework for Temporal Analysis and its Application to Instructional Texts

Khee Yin How

PhD
University of Edinburgh
1993

Abstract

Temporal analysis is the task of determining the temporal structure of a given text. Such a structure represents the order of the events and states mentioned in the text on a time line. The main contribution of this thesis is in presenting a new processing framework for temporal analysis.

The framework is a computational one and has been implemented in a system called TASTE for the temporal analysis of instructional texts. In particular, TASTE has been successfully tested on nine cookery recipes. Amongst the more important ideas explored in this thesis are the following:

- We integrate qualitative information (as expressed by temporal connectives like *before* and *after*) and quantitative information (as expressed in phrases like *for 20 minutes* and *20 minutes before*) in a text into the temporal analysis framework. Previous work has only considered qualitative information but ignored the quantitative kind.
- We propose a new approach to the problem of integrating the current event or state into the preceding discourse. This problem has been identified as important for solving the temporal analysis task.
- We show how information from the environment surrounding a text can affect the temporal analysis of instructional texts. In particular, we show that different temporal structures for the same text can be derived in different environments. Note that the environment information is in addition to the usual information considered in temporal analysis such as information from tense and aspect, temporal connectives and real-world knowledge. An example of information from the environment for the domain of cookery recipes is the availability of resources for carrying out an action.
- We incorporate techniques developed in the field of temporal reasoning into the temporal analysis task. In addition, we analyse the complexity of temporal reasoning algorithm needed in the temporal analysis of instructional texts.
- We propose a novel ontology for representing the composite and repetitive events that are mentioned in cookery recipes.

Finally, the thesis ends with some suggestions for extending the work reported here.

Acknowledgements

This thesis would not have been possible without the countless comments and suggestions made by my supervisors Dr Robert Dale and Dr Graeme Ritchie. My examiners Dr Karen Sparck Jones and Dr Marc Moens have also made valuable comments on the thesis. I thank all of them from the bottom of my heart.

I would also like to thank Dr Chris Mellish for the encouraging comment made after one of my presentation. It has boosted my confidence and seen me through the lonely and depressing periods of the PhD study.

I would also like to thank many of the friends and colleagues who have provided helps, advice and companionship during the PhD study. In no particular order, special thank to: Sheila Glasbey and Sheila Rock (for providing valuable comments on chapters of the thesis), Ian Frank (for reading chapters of the thesis), Hsiao Lan Fang, Wei Jen Wu, Peng Cheng Wang, Keiichi Nakata, Alistair Knott, Nam Seog Park, Robert Scott, Wamberto Vasconcelos, Maria Vargas-Vera and Rolando Carrera.

My wife, Lai Mui, has been a source of impetus for me to finish the PhD as soon as possible. I thank her here for the understanding and support and making the whole PhD pursuit worthwhile.

For financial support, I would like to thank the Government of Singapore for three years of sponsorship.

Declaration

I declare that this thesis has been composed by myself and that the research reported therein has been conducted by myself unless otherwise indicated.

Khee Yin How

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis addresses the problem of temporal analysis for natural language discourse. A discourse typically consists of multiple sentences, each of which describes either a state or an event. Temporal analysis is then the task of determining the intended temporal relations between the events and the states in the discourse. Consider the following simple discourse adapted from [Caenepeel & Moens 93]:

(1.1)
a. e_1: Jane and I had a quarrel.
b. e_2: Three months after that, she left me and
c. e_3: fell in love with someone else.
d. s_1: I was very upset.
e. e_4: Luckily, Anne came to keep me company.

In (1.1), the first three clauses[1] (a, b and c) and the last clause (e) describe events while the second last clause (d) describes a state. The temporal relations that can be extracted by temporal analysis from this discourse are: e_1 occurs before e_2, e_2 occurs before e_3, e_3 occurs before s_1 and e_4 is during s_1. These relations together define the temporal structure of (1.1) which is shown in figure 1.1.[2]

[1]We will also use the term 'clause' for simple sentence.

[2]In diagrams depicting temporal structures, a horizontal line denotes a time line while rectangular boxes denote time intervals assigned to situations. In such diagrams, time is taken to advance from

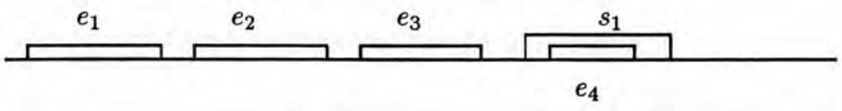


Figure 1.1: The temporal structure of example (1.1)

Temporal analysis has found applications in many research areas of Artificial Intelligence (AI). In story understanding, for example, one is interested not only in the events that occur in the story but also the order in which they occur. For instance, the order of the events and states described in (1.1) is given in its temporal structure. Knowing this structure allows one to answer questions like *What did Jane do after quarreling with me?* and *Who came to keep me company when I was upset?*.

In addition to story understanding, temporal analysis is also useful for query interpretation to temporal databases and knowledge-based systems. For instance, [Hinrichs 88] describes how temporal analysis is used in the JANUS system, a natural language system that allows a user to interface with several knowledge bases maintained by the U.S. Navy. These knowledge bases contain, among other things, information about the deployment schedules, locations and readiness conditions of ships in the Pacific Fleet. Information from temporal analysis is used in JANUS to answer questions that involve time like *When will Vincent arrive in Hawaii?* and *Who was Frederick's previous commander?* where Vincent and Frederick are the names of U.S. ships.

1.1 Problems in Temporal Analysis

Temporal analysis is not an easy task. In this section, we highlight the issues which have been identified as critical for temporal analysis. These issues are as follows:

- the integration of both qualitative and quantitative temporal information;

left to right.

- the interaction among the wide range of linguistic devices that can be used to describe temporal relations and world knowledge;
- the need for context to interpret these linguistic expressions; and
- the problem of integrating a situation described by the current clause into the previous discourse.

We will now elaborate on each of these points below. For ease of exposition, we will follow [Mourelatos 78, Comrie 76] in using the term SITUATIONS to refer to either states or events.

1.1.1 Qualitative and Quantitative Temporal Information

Both qualitative and quantitative temporal information can be described in a discourse. The former is concerned with the relative ordering between two situations or describes whether one situation contains the other, while the latter is more detailed by further specifying the extent of the time separation or containment. Consider example (1.2).

(1.2) a. 30 minutes before

b. e_1: serving,

c. e_2: add the wine.

In (1.2), both qualitative and quantitative temporal information is expressed. The *before* adverbial expresses qualitative temporal information and indicates that e_2 precedes e_1. Notice that the qualitative temporal information is not concerned with the amount by which e_2 precedes e_1: it makes no difference if e_2 precedes e_1 by ten or twenty minutes. On the other hand, the difference does matter in quantitative temporal information. In (1.2), the qualifier of the adverbial, *30 minutes*, gives quantitative information specifying that the extent which the event of adding the wine precedes the serving event by is 30 minutes.

Current processing frameworks for temporal analysis [Webber 87, Passonneau 88, Song 91b] consider only qualitative temporal information but ignore the quantitative type. This leads to a less precise analysis of the temporal structure of discourse than is possible. For instance, frameworks that ignore quantitative information would be unable to determine that in example (1.3), John came home one hour later than Mary. This determination can only be obtained via the quantitative information expressed by the *at*-adverbials.

(1.3) John reached home at 8 pm and Mary at 7 pm.

In our framework, we have successfully integrated both qualitative and quantitative information.

1.1.2 Interaction among temporal linguistic devices and world knowledge

There is a wide range of linguistic devices that can be used to describe temporal relations. These devices include tense, aspect, temporal adverbials and temporal connectives, and have been extensively studied by both linguists and computational linguists.

In English, tense is a grammatical category and provides ordering information with respect to a well-defined point on the time line, such as the time of speech. For instance, the clause *John went to see a doctor* is in the simple past tense and indicates that the event of going to see a doctor is located before the time of speech. On the other hand, aspect is a semantic category and is determined from the tense and the type of the main verb as well as the meaning of the verb's arguments, including subjects and objects. The aspect of a clause indicates which part of a situation is being described by the clause. For instance, the aspect of the clause *The water has cooled* indicates the end state of a cooling event. In contrast, on switching to a progressive tense, the aspect of the clause *The water is cooling* indicates the cooling process itself. Finally, temporal adverbials like *for three minutes* and *at three o'clock* and temporal connectives like *before* and *meanwhile* can also provide additional temporal information. For instance, in the clause *Before serving, put in the wine*, the *before* connective indicates that the put-in-the-wine event occurs before the serving event.

These linguistic devices interact with one another and with world knowledge to determine the intended temporal relation between two situations, complicating the temporal analysis task. Consider, for instance, the interaction between the *when* connective, tense and aspect, and world knowledge in (1.4) and (1.5), taken from [Singh & Singh 92].

(1.4) When Al won the race, he took steroids.

(1.5) When Al was winning the race, he took steroids.

In (1.4) and (1.5), the event of *winning the race* can be thought of [Singh & Singh 92] as consisting of three parts: the preparatory stage before the running commences, the actual running itself and the final state at the end of the race. For (1.4), we can use world knowledge to infer that Al takes the steroids before the race begins, i.e. in the preparatory stage. That is, the event of taking the steroids is before the event of winning the race. However, in (1.5), the progressive forces an interpretation in which the taking of the steroids must be during the actual running itself. No world knowledge is able to override this effect of the progressive. Thus, for (1.5), the temporal relation extracted is that the event of taking the steroids is during the event of winning the race.

Although the various linguistic devices for expressing temporal information have been extensively studied, it is only recently that researchers have started building computational models for them [Hinrichs 87, Webber 87, Moens 87, Passonneau 88, Singh & Singh 92]. Among this work, [Moens 87, Singh & Singh 92] are the most interesting from the point of view of our work since they provide an account of the interaction of the various temporal linguistic devices discussed in this section for single sentences. In this thesis, we integrate ideas from this work on single sentences into our temporal analysis framework for discourses of more than one sentence.

1.1.3 The Influence of Context

Context can be seen to influence temporal analysis, since many of the linguistic devices discussed earlier cannot be interpreted without looking at previously mentioned situations. For example, tense can relate a situation to the time of speech and also to a previously mentioned situation. Consider example (1.6):

(1.6) a. e_1: John went to the hospital.

b. e_2: He had twisted his ankle on a patch of ice.

where clause (1.6a) introduces an event e_1 prior to the time of speech while clause (1.6b) introduces another event e_2 which is interpreted as happening before the previous event e_1. This is because a past perfect tense, as in clause (1.6b), typically describes a situation that holds or occurs before another situation. Similarly, the aspect of a clause describes which part of a situation is viewed and in a discourse, a new clause may describe part of a previously mentioned situation. Consider example (1.7):

(1.7) a. e_1: Cool some soup.

b. When the soup has cooled,

c. e_2: put in the meat.

where the *when*-clause in (1.7b) is describing the end state of the cooling event mentioned in (1.7a). We can thus infer that the event of putting in the meat begins after the cooling event has finished.

Another way in which context can influence temporal analysis but which has not been noted in previous work is via the resolution of referring expressions such as pronouns and NPs. It is a well-established fact that the interpretation of referring expressions is dependent on context. However, the role which such interpretation can contribute to temporal analysis has not been investigated. Consider, for instance,

(1.8) a. Brown the beef until tender.

b. e_1: Chop one of the onions.

c. e_2: Add the chopped onion to the beef and

d. fry together.

where the NP *the chopped onion* in clause (1.8c) describes an onion that has participated in a chopping event. Since the 'chopped' property of the onion in e_2 is acquired in e_1, we have e_2 to be after e_1. Now, one might argue that it is not necessary to use such reference resolution information since it appears that the narrative convention (the narrative convention states that time moves forward for events) can be used to determine that e_2 is after e_1. However, the narrative convention may not work all the time and in some cases can be too restrictive. For instance, in example (1.9), narrative convention would dictate that the chopping of the potato must occur after the chopping of the carrot.

(1.9) a. e_1: Chop a carrot.

b. e_2: Chop a potato.

However, it is also possible that the chopping of the potato can occur at the same time as the chopping of the carrot.

Finally, the context which affects temporal analysis need not be from within the discourse, such as those cases discussed above. Information from outside the discourse can also affect temporal analysis. In this thesis, we will view such extra-linguistic information as the ENVIRONMENT surrounding a discourse. To illustrate the effect, consider example (1.9) again. In an environment where there are two agents and two chopping knives, e_1 and e_2 can occur simultaneously. On the other hand, in another environment where there is only one agent and one chopping knife, e_1 and e_2 must occur sequentially. No current processing frameworks of temporal analysis have incorporated the effects of such environmental information.

In our work, we study how information from reference resolution and the environments surrounding a text can affect temporal analysis.

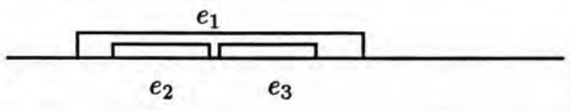


Figure 1.2: The temporal structure after processing example (1.10a) to (1.10c)

1.1.4 The Integration Problem

When carrying out temporal analysis of discourse consisting of more than one clause, a difficult problem is integrating the situation described by the current clause into the temporal structure of the previous discourse. To see why this is so, consider the following example from [Webber 87]:

(1.10) a. e_1: John went over to Mary's house.

b. e_2: On the way, he had stopped by the flower shop for some roses.

c. e_3: He picked out 5 red ones, 3 white ones and one pale pink.

d. e_4: Unfortunately they failed to cheer her up.

For (1.10), we assume that the events e_1 to e_3 have been successfully processed and that the temporal structure so far is as given in figure 1.2. The next clause to be processed is (1.10d) which gives rise to the event e_4. The integration problem is concerned with where in the temporal structure to place e_4.

A common approach [Webber 87, Song & Cohen 91b, Hwang & Schubert 91] is to determine a suitable REFERENCE from which the temporal location of e_4 can be 'measured'. One way [Eberle 92, Hwang & Schubert 91] is to look for the reference among the preceding events. Applying this approach to example (1.10), the suitable reference for e_4 is e_1. From world knowledge, we can infer that e_4 occurs after e_1. Notice that since e_4 is after e_1, the orderings of e_4 with the rest of the preceding events are also known.

However, the choice of e_1 as the reference for e_4 is far from obvious. In the first place, the clause which describes e_1 is not immediately previous to the clause which describes

e_4.

The integration problem has been looked at only recently [Webber 87, Song 91b, Hwang & Schubert 91, Eberle 92]. The solutions proposed face difficulties of one kind or another (see chapter 2). In this thesis, we provide an alternative solution to the integration problem that does not have these difficulties.

1.2 Problems to be Addressed in this Thesis

The aim of this thesis is to provide a processing framework for carrying out temporal analysis of discourse. More specifically, the framework needs to achieve the following objectives:

- To integrate qualitative and quantitative information to give a more precise temporal analysis. As noted above, previous computational work has ignored quantitative information.
- To study the effect of the environment in which a discourse is embedded on the temporal analysis task. No current temporal analysis frameworks have incorporated this kind of information.
- To study the integration problem and to present a new approach for solving it. As noted above, there has been much work in attempting to solve the integration problem but the solutions are far from satisfactory.
- To integrate previous work on the temporal analysis of single sentences [Moens 87, Singh & Singh 92] into that for discourses of more than one sentence.
- To embody the framework in a working system and thus demonstrate its practicability.

1.2.1 The Domain of Application

The richness and breadth of natural language means that any computational treatment has to narrow its focus. For example, previous work on temporal analysis has typically used only narrative texts as the genre of study. In addition, there may be the assumptions that there is only one speaker and all the utterances describe actual situations, those that are asserted to have already occurred, to be occurring, or to occur at a particular time [Passonneau 88]. Narratives with such restrictions have been called SIMPLE NARRATIVES [Song 91a]. Similarly, in our work, we have to decide on a domain of application for achieving our aims stated above.

In order to get a fresh perspective on temporal analysis, we decide to move away from the traditional genre of narrative texts. Instead, we use instructional texts as our genre of study. Now, instructional texts are very different from narrative texts. A fundamental difference between instructional and narrative texts is that the former are prescriptive while the latter are descriptive. That is, clauses in narrative texts describe situations that have already occurred or to be occurring at a particular time while instructions in instructional texts prescribe actions that are to be carried out by an agent. This means that in carrying out the temporal analysis of narrative texts, one is concerned with reconstructing the ordering of the situations that are described in the narration. However, when carrying out the temporal analysis of instructional texts, it is a case of deciding the order in which the events described in the text are to be carried out.

Given the difference between instructional and narrative texts, it is interesting to see whether a processing framework designed for instructional texts can be applied to narrative texts. We believe that it is possible, and the issue will be further discussed in chapter 7.

Since the interest in this thesis is on instructional texts, it is appropriate to note some observations made in the literature on such texts which bear on temporal analysis:

- One such observation is given in [Webber & Di Eugenio 90] which states that there need not be a direct relationship between an instruction and the action carried out by an agent. An example of a direct relationship is given by an instruction like *Jane, turn on the light*, which would evoke the turning-on-the-light behaviour of Jane. However, there are also instructions that can depart from this simple direct relation. For instance, Webber and Di Eugenio [Webber & Di Eugenio 90] give the following instruction in (1.11) which is part of the instructions for filling holes in plaster over wood lath.

(1.11) If a third coat is necessary, use prepared joint compound from a hardware store.

An agent intending to execute (1.11) will not know if a third coat is necessary until s/he sees whether the first two coats have produced a smooth level surface. This means that the instruction is conditional on information coming from the agent's environment.

Similarly, in deciding the order in which actions are to be carried out, an agent may need to make use of environmental information. Of course, there exist instructional texts where the order of execution is explicitly given:

(1.12) First, chop the carrot. Then, chop the potato. Finally, add the carrot and potato to the soup.

However, there also exist instructions where the order of the actions is not explicitly given. For instance, in example (1.9) given above and reproduced below:

(1.9) a. e_1: Chop a carrot.

b. e_2: Chop a potato.

the order of the two chopping actions is not specified. As noted earlier, this order is dependent on environmental information such as the number of agents and number of chopping knives.

Based on the discussion above, we will say that the behaviour of an agent given an instruction is SITUATED (i.e. depends on information from the agent's environment).

- Another observation is made by Dale [Dale 92] who notes that the most useful temporal ordering of the actions is often sabotaged by the author of the recipe. One reason may be due to editorial compactness so that actions which act on the same objects are often described together even if these actions are not to be carried out sequentially. Consider the following example from [Dale 92]:

  (1.13) Cook the rice; cool.
  Sprinkle apple and banana with lemon juice and add to rice.
  Steep raisins in a little boiling water for half an hour to plump;
  drain, and add to rice with sunflower seeds, mixing well.
  Fry onion in oil with curry powder for 10 minutes.

  where the instruction to *steep the raisins* comes after the instruction to *add the apple and banana to the rice* although it would clearly (in the context of the following instructions) be more sensible to start steeping the raisins at an earlier point in time.

From the observations discussed above, we can note that for the temporal analysis of instructional texts, environmental information plays an important role and that narrative convention imposes too strong a constraint.

To further focus our study, we consider only a particular type of instructional texts, namely cookery recipes. The cookery recipe domain is fairly well-defined and yet exhibits some remarkably complex phenomena. It is therefore not surprising that cookery recipe has been extensively used as a domain of investigation. For instance, in planning, Hammond [Hammond 86] describes a system that plan recipes while Tsang [Tsang 86] uses recipes as the domain for discussing the temporal aspect of planning. In natural language work, Karlin [Karlin 88] describes a natural language interface to a computer animation system that operates in the cooking domain and Dale [Dale 92] describes a system for generating referring expressions found in recipes.

An example of a recipe that we use as data is shown in figure 1.3 (taken from the Internet newsgroup, *rec.food.recipes*). This recipe is further simplified to remove any linguistic expressions that do not affect temporal analysis before processing by our

GREEN ONION AND POTATO SOUP

5–6 medium sized potatoes

$\frac{1}{2}$ a bunch of green onions, sliced

1 shiitake or chinese black mushroom

salt and pepper

bay leaves

1 chicken leg

1 cup milk

Soak the mushroom in hot water for 30–45 minutes. Peel and slice the potatoes. Saute the onions in a little bit of olive oil over medium heat. Add the potatoes and a quart of water. Put in the bay leaves and the chicken leg. Simmer for an hour. Dice the mushroom and add the mushroom pieces to the soup. When the potatoes are done, mash them with a fork or a potato masher and stir the soup well. Add the milk and season with salt and pepper. Oh, before you mash the potatoes, take the chicken leg out, cut it into little pieces and add it back in.

Figure 1.3: The GREEN ONION AND POTATO SOUP recipe

TASTE system (see section 1.3.2).

1.3 Contributions of the Thesis

In the previous section, we have explained the problems to be addressed in the thesis. Our main contributions are:

- We propose a framework for temporal analysis that embodies the aims given above. Thus, our framework is more complete and precise than previous frameworks for temporal analysis.
- Our framework is a computational one and is embodied in a working system, called TASTE (Temporal Analysis of inStructional TExts), in the domain of cook-

ery recipes.

We will now elaborate further on these two contributions.

1.3.1 Our Temporal Analysis Framework

It is common to postulate that when a person is understanding a text, he/she is building up some representation of the text. For example, Webber [Webber 87, p 147] postulates that in processing a narrative text, a listener is building up a representation of the speaker's view of the events and states being described and of their relationship to one another. According to Webber, this representation (termed an E/S STRUCTURE by Webber) reflects the listener's best effort at interpreting the speaker's ordering of those events and states in time and space.

Likewise, in our work, we postulate that in processing an instructional text, an agent is building up some form of representation which encodes the situations mentioned in the text and their relationships. However, for instructional texts, this representation does not necessarily reflect the ordering of the events in time. If the representation reflects the ordering of the events in time, then the behaviour of the agent is fixed, i.e. the agent needs only to execute the events in the order given, irrespective of the environment he/she is in. This would imply a direct relationship between instructions and an agent behaviour, contradicting the observation that the behaviour of an agent executing an instruction is situated.

Thus, we believe that in the representation built up by an agent when processing instructional texts, the relations between situations need not all be temporal. Such a representation can then interact with information coming from the environment to determine the agent behaviour. This means that the same representation can give rise to different agent behaviours in different environments. We represent this model graphically in figure 1.4.

We note that the temporal structure of a text is a representation that can be seen as specifying a particular form of agent behaviour. That is, the temporal structure

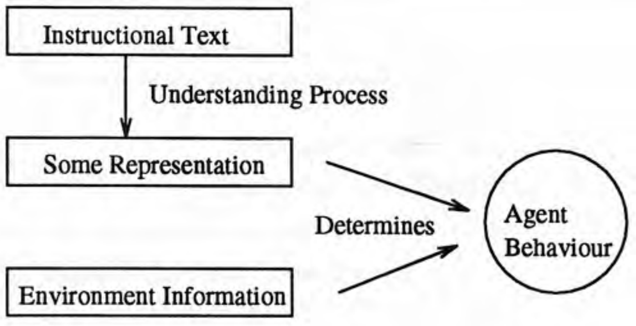


Figure 1.4: A model of the relationship between instructional text and agent behaviour

determines not only the actions that the agent needs to carry out but also when to execute them. Based on this observation, we obtain the following framework for temporal analysis (called TA2) from the model in figure 1.4.

In TA2, we postulate two levels of representation. In the first level, we have the representation that encodes the situations mentioned in the text and their relations. This representation can then be combined with information from the environment to determine the temporal structure which constitutes the second level of representation. In this thesis, we will call the first level representation, the Quasi Temporal Structure (QTS), since the relations at this level of representation need not all be temporal. The two levels of representation are depicted graphically in figure 1.5.

Informally, the QTS encodes the representation of the situations described in the text and the relations that hold between pairs of situations. We now consider some of the possible relations that can hold between two situations (see chapter 3 for the complete set):

- In some cases, the relation between two situations is given explicitly in the text. Consider example (1.14):

  (1.14) Before e_1: you mash the potato, e_2: take out the chicken leg.

  where the use of the *before*-connective indicates that e_1 (the mashing event) occurs after e_2 (the taking-out event).

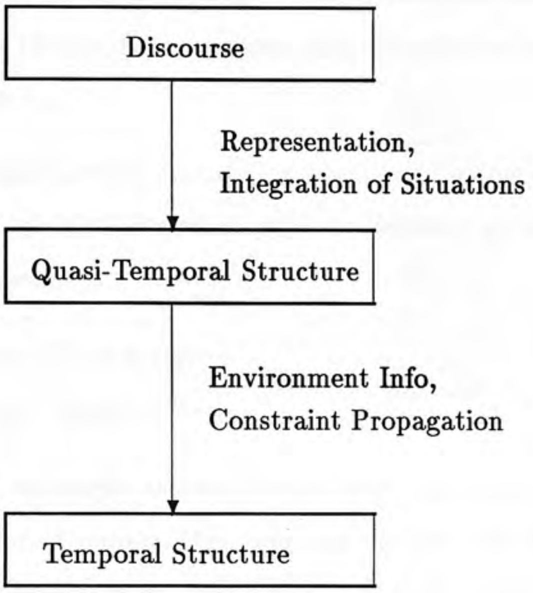


Figure 1.5: The two levels of representation in TA2

- The relation explicitly indicated can also be a non-temporal one such as in

  (1.15) Clean the wok by scrubbing thoroughly.

  where the use of the preposition *by* explicitly indicates that carrying out the scrubbing action in the event described by *scrubbing thoroughly* achieves the event described by *cleaning the wok.* In such cases, we say that the relation between the scrubbing event and the cleaning event is such that the former generates [Goldman 70] the latter. Note that knowing this relation allows one to determine that in executing the instruction given in (1.15), only one action needs to be carried out by an agent.

- For other pairs of situations, the relations involved are not explicitly given and have to be inferred. Consider, for instance,

  (1.16) a. e_1: Cook some rice.

  b. e_2: Cool.

  where the relation between e_2 and e_1 has to be inferred from the world knowledge that cooking results in the rice being hot, a condition needed to bring about the cooling event. Therefore, we can infer that the relation between e_1 and e_2 is such that e_1 enables e_2.

- Finally, it is also possible that there might not be any apparent relations holding between a pair of situations such as between e_2 and e_1 in example (1.9) reproduced below:

  (1.9) a. e_1: Chop a carrot.

  b. e_2: Chop a potato.

  In such cases, we might assume the narrative convention so that e_2 follows e_1. However, as noted earlier, this need not be the case if we take into account environmental information. Recall that if (1.9) is carried out in an environment where there are two agents and two chopping knives, then e_2 can be carried out simultaneously with e_1. Now, one might be tempted to include environmental information as another source of information in addition to other knowledge sources like world knowledge to infer the relation between events, so that in the QTS of (1.9), there exists a temporal relation between e_1 and e_2. However, this obscures the role which environmental information plays in the temporal analysis task, i.e. that under different environments, different temporal structures may be obtained from the same QTS. Thus, in the QTS of example (1.9), e_2 is represented as having no relation with e_1 since no apparent relations can be derived between e_1 and e_2 based on world knowledge and the semantic contents of the clauses describing e_1 and e_2.

The QTS is made use of in the second step of TA2. At this step, we associate each situation with a unique time interval. The ordering of the situations on a time line is then determined by considering the ordering of their respective time intervals on the time line. The ordering of the time intervals on a time line is determined from both qualitative and quantitative temporal relations between time intervals.

Qualitative temporal relations between time intervals are expressed using the relations in Allen's Interval Algebra [Allen 83] (see chapter 2). These qualitative temporal relations between time intervals are derived from the relations between the situations identified by the time intervals in the QTS. It is at this step that environmental information such as the number of agents available is made use of. Note that in different environments, situations that are indicated in the QTS to have no relation with one another can have different qualitative temporal relations between their time intervals.

Quantitative temporal relations are typically relations between the start and end time points of a time interval or a pair of time intervals. They are expressed via temporal adverbials like the *for*-adverbial as in

(1.17) e_1: Soak the carrots *for twenty minutes.*

where the *for*-adverbial indicates a quantitative temporal relation that the distance between the start and end time point of the time interval associated with e_1 is 20 minutes. Such quantitative temporal relations are extracted from the text at the same time as the QTS is built up. It is stored separately from the QTS as a set of equations involving the start and end time points of time intervals (see example (1.18) and (1.19)).

We then view qualitative and quantitative temporal relations as temporal constraints on time intervals and time points and propagate them via a constraint propagation algorithm developed in the field of temporal reasoning [Allen & Kautz 85, Kautz & Ladkin 91, Meiri 91]. Consider example (1.2) again:

(1.2) a. 30 minutes before

b. e_1, te_1: serving,

c. e_2, te_2: add the wine.

where the *before* connective indicates that e_2 occurs before e_1. In the QTS for (1.2), we represent explicitly that e_2 occurs before e_1. In the second step, we extract the qualitative temporal relations between the time interval associated with e_1 (te_1) and the time interval associated with e_2 (te_2). There are two possible such qualitative temporal relations depicted graphically in figure 1.6 where te_i^- is the start time point of time interval te_i and te_i^+ is the end time point of time interval te_i. To gloss in

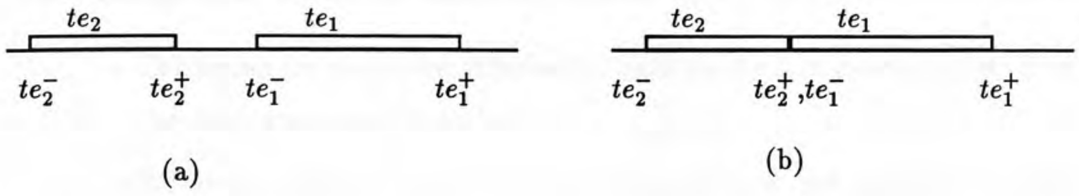


Figure 1.6: The two possible qualitative temporal relations for example (1.2)

English, figure 1.6a depicts that e_2 occurs before e_1 while figure 1.6b depicts that e_1 starts when e_2 finishes.

The quantitative temporal relation extracted from (1.2a) is given by the following equation:

(1.18) $te_1^- - te_2^- = 30$

In addition, suppose we also know the typical duration for the adding action in e_2 is less than 30 minutes:

(1.19) $te_2^+ - te_2^- < 30$

Now, if we combine the qualitative and quantitative temporal relations together via constraint propagation, only the ordering indicated in figure 1.6a remains. This is then the temporal structure of (1.2).

Based on the discussion above, we can identify the following two stages in the TA2 framework:

1. For each sentence in the text, do

   Representation: Construct the representation of the situation underlying the sentence and extract any quantitative information mentioned in the sentence.

   Integration: Integrate the situation representation into the QTS.

2. **Propagation:** Obtain the temporal structure by

   - Extracting the qualitative information between the time intervals over which the situations occur from the QTS;
   - Forming a constraint network from the qualitative and quantitative information;
   - Propagating the constraints in the constraint network and
   - Extracting the temporal structures from the resultant constraint network.

Finally, although TA2 is motivated by the analysis of naturally occurring data from cookery recipes, we believe the framework can also be applied to other types of instructional texts. In addition, as will be further discussed in chapter 7, we believe that the framework can also be applied to narrative texts.

1.3.2 An Implementation of TA2

TA2 is a computational framework and is implemented in a system called TASTE. TASTE has been tested on nine recipes, selected from cookery books and the newsgroup *rec.food.recipes.* Before passing to TASTE, the recipes need to be simplified by removing linguistic expressions that are not relevant to temporal analysis but otherwise complicate the task. The nature of these simplifications is described in Appendix A.

One of the nine recipes which TASTE processes is given in figure 1.3. This recipe is first simplified to that given in figure 1.7 before passing to TASTE. The result of the processing is given in figure 1.8 which shows the QTS (top diagram) and the temporal structure (the bottom diagram) built up by TASTE. In the QTS, the symbols P, A, C, D, N are abbreviations for the **precede**, **after**, **constituency**, **include** and **none** relations (see chapter 3) respectively.

The temporal structure shown represents the 'best' ordering of the situations according to the heuristics employed by TASTE (see chapter 5). Note from the temporal structure that TASTE determines that it is best to first prepare all the ingredients for the soup.

GREEN ONION AND POTATO SOUP

5 to 6 medium sized potatoes

$\frac{1}{2}$ a bunch of sliced green onions

1 chinese black mushroom

some salt and pepper

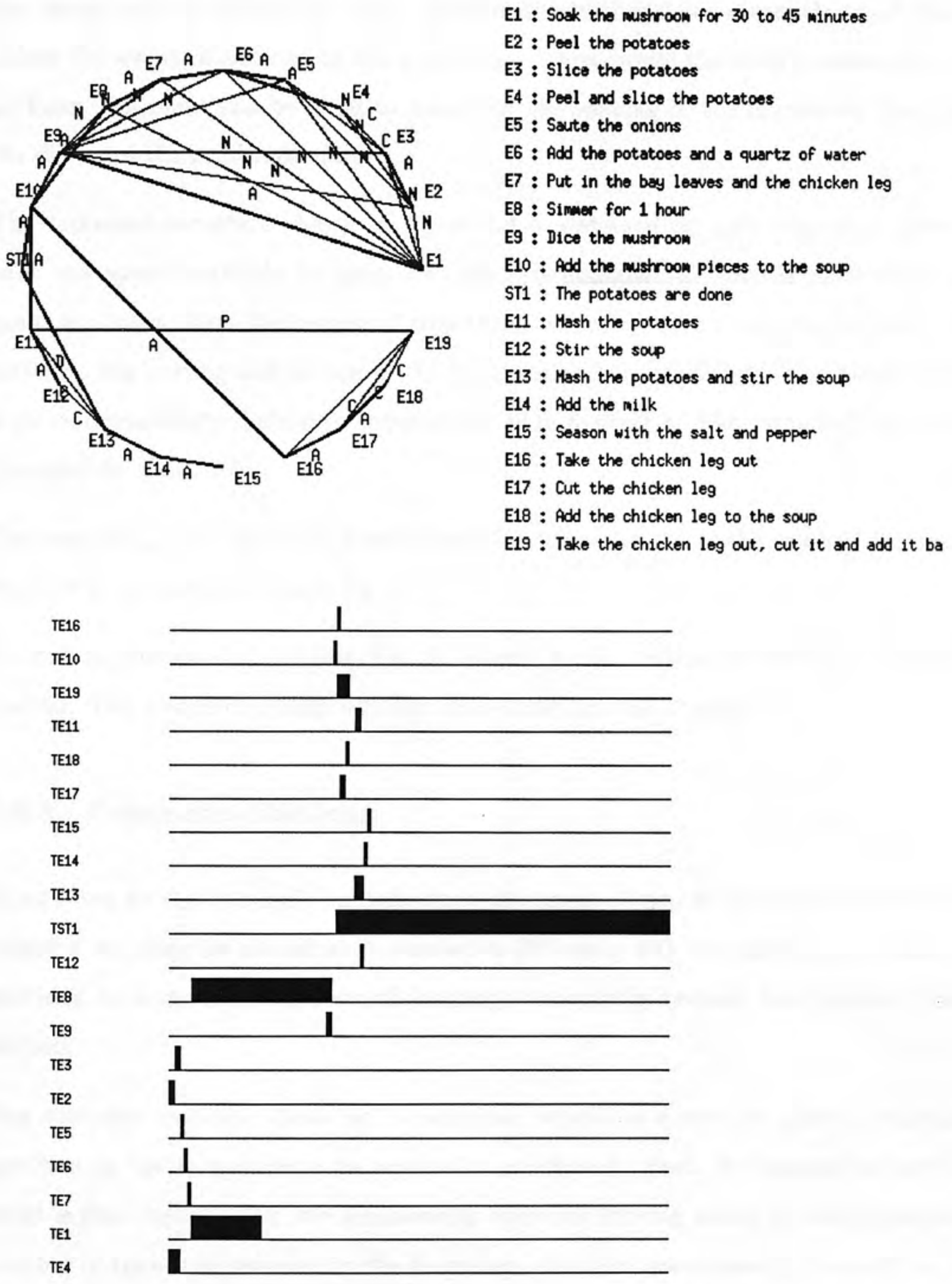
some bay leaves

1 chicken leg

1 cup of milk

Soak the mushroom for 30 to 45 minutes. Peel and slice the potatoes. Saute the onions. Add the potatoes and a quart of water. Put in the bay leaves and the chicken leg. Simmer for an hour. Dice the mushroom. Add the mushroom pieces to the soup. When the potatoes are done, mash them. Stir the soup. Add the milk. Season with salt and pepper. Before you mash the potatoes, take the chicken leg out, cut it and add it back in.

Figure 1.7: The simplified form of the GREEN ONION AND POTATO SOUP recipe



E1 : Soak the mushroom for 30 to 45 minutes
E2 : Peel the potatoes
E3 : Slice the potatoes
E4 : Peel and slice the potatoes
E5 : Saute the onions
E6 : Add the potatoes and a quartz of water
E7 : Put in the bay leaves and the chicken leg
E8 : Simmer for 1 hour
E9 : Dice the mushroom
E10 : Add the mushroom pieces to the soup
ST1 : The potatoes are done
E11 : Mash the potatoes
E12 : Stir the soup
E13 : Mash the potatoes and stir the soup
E14 : Add the milk
E15 : Season with the salt and pepper
E16 : Take the chicken leg out
E17 : Cut the chicken leg
E18 : Add the chicken leg to the soup
E19 : Take the chicken leg out, cut it and add it ba

Figure 1.8: The QTS and temporal structure of the GREEN ONION AND POTATO SOUP recipe

These preparations include peeling the potatoes (te_2), slicing the potatoes (te_3), sauteing the onions (te_5), adding the potatoes and a quart of water (te_6) and putting in the bay leaves and the chicken leg (te_7). Thus, in the temporal structure, these are placed before the event of simmering the soup (te_8). Then, while the soup is simmering for an hour, the agent can be freed to carry out the soaking of the mushroom (te_1) and the dicing of the mushroom (te_9).

The temporal structure shown in figure 1.8 is obtained by assuming that there is only one agent available to carry out the instructions. If there is more than one agent available, then the temporal structure output by TASTE will be different. For instance, the peeling and slicing of the potatoes and the sauteing of the onion can be done simultaneously instead of sequentially as in figure 1.8. This issue will be further discussed in chapter 5.

The remaining eight recipes and their simplified forms, as well as the result of processing by TASTE, are given in Appendix A.

As will be discussed in section 6.6 of chapter 6, the testing of TASTE is somewhat limited. The design of TASTE will also be detailed in that chapter.

1.3.3 Other contributions

In addition to the two main contributions discussed above, we have also developed a complex ontology for situations. As noted in [Wilensky 91], the ontology of situations provided in computational natural language processing systems has generally been limited.

Our complex ontology allows us to represent repetitive events in which the actions specified in the events are to be repeated a number of times. An example of such an event is that described by *stir occasionally* where the stirring action is to be repeated a number of times (determined by the frequency adverbial *occasionally*). In addition, we can also represent events which contain sub-events. For instance, the event described by *bring the soup to the boil stirring occasionally* can be represented as a composite event consisting of the two sub-events described by *bring the soup to the boil* and *stir* *occasionally* which are to be carried out simultaneously.

1.4 An Overview of the Thesis

The rest of the thesis is structured as follows:

Chapter 2 presents an overview of related work. It surveys the previous work that has been done in temporal analysis. In particular, it highlights the issues which previous work has focused on, the solutions proposed, and the limitations of these solutions.

Chapter 3 proposes an ontology for representing situations as needed in the REPRESENTATION step of TA2. This ontology is capable of representing complex events such as composite events with sub-events and repetitive events. In addition, we also describe the relations between situations that we identify and use in our work. Finally, the chapter discusses the issue of taking situations as holding over time intervals.

Chapter 4 looks at the issues involved in constructing the QTS as required in the INTEGRATION step of the framework. The chapter presents a solution for integrating a current situation into the QTS of the previous discourse. As noted above, the integration problem has received much attention recently. Our solution utilises information from the resolution of NPs and pronouns as well as clauses to decide on the reference events for the current situation.

Chapter 5 addresses the PROPAGATION step of the framework. The key idea is to associate each situation with a unique time interval and to extract the qualitative temporal relations between time intervals from the relations between situations in the QTS. The chapter will also introduce temporal reasoning and how this provides us with a mechanism for integrating qualitative and quantitative temporal relations. Such integration is missing from other current temporal analysis frameworks. Finally, we show how different temporal structures for a text can be obtained in different environments.

Chapter 6 discusses the TASTE system developed for the temporal analysis of instructional texts. TASTE takes natural language texts as input and the chapter discusses how TASTE implements the various steps in TA2.

Finally, chapter 7 concludes the thesis and discusses some future directions.

Chapter 2

A Survey of Temporal Analysis

This chapter surveys the work that has been done on temporal analysis. The problems which such work addresses covers both single sentences and discourse of more than one sentence.

The main issue investigated in work on the temporal analysis of single sentences (such as [Moens & Steedman 88, Passonneau 88, Singh & Singh 92]) is how the various linguistic devices like tense, aspect, temporal adverbials, and world knowledge interact to determine the temporal relation between the situations described in the sentence. Some typical sentences used for such investigation are (from [Ritchie 79, Moens & Steedman 88]):

(2.1) 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 their traffic problems.

where (2.1a) to (2.1c) are three possible main clauses. Although the three sentences in (2.1) are syntactically similar, the temporal relation between the event described by the main clause and the event described by the *when*-clause is different in each case. From world knowledge, it can be derived that the drawing up of the plans precedes the building of the bridge, the usage of the best materials occurs during the building itself and the traffic problems are solved after the bridge is completed.

In the temporal analysis of discourse of more than one sentence, previous work (such as [Hinrichs 86, Dowty 86]) first looks at simple narratives in which the order of the underlying events directly corresponds to their order of presentation. Following Webber [Webber 87], we will call such simple narratives LINEAR narratives. Some examples are:

(2.2) a. John got up. He poured himself a coffee.

b. John got up. He was in a bad mood.

For such work, the main issue studied is the problem of deciding when the current clause in the discourse is interpreted as a situation following the situation described by the previous clause (example (2.2a)) and when as overlapping it (example (2.2b)).

Subsequent work on discourse (such as [Webber 87, Song 91b]) examines texts where narrative order may not correspond to the presentation order, as in:

(2.3) a. Annie broke her leg. She fell off the bicycle.

b. Max died. John poisoned him.

where in each case the event described by the second clause is most plausibly interpreted as preceding that described by the first clause. In addition, this work looks at texts which contain an embedded discourse, such as

(2.4) a. e_1: Mary and I talked about his brother.

b. e_2: He had spent five weeks in Alaska with two friends.

c. e_3: Together, they climbed Mt. McKinley.

d. e_4: Mary was very proud of him.

where clauses (2.4b) and (2.4c) form another discourse embedded within the main discourse comprising clauses (2.4a) and (2.4d). For ease of exposition, we will call texts where narrative order may not agree with presentation order and texts which contain embedded discourses NON-LINEAR narratives.

Unlike linear narratives, non-linear narratives complicate temporal analysis since the situation described by the current clause need not necessarily be interpreted against the situation described by the *previous* clause, and this interpretation need not be one of *following* or *coincidence*; it can also be *before*. Recall from chapter 1 that we call this problem the problem of *integrating* the current situation into the preceding discourse.

In the rest of this chapter, we will discuss how previous work tackles the problems and issues discussed briefly above.

In section 2.1, we first discuss the basic concepts underlying previous work in temporal analysis.

Then, in section 2.2, we discuss work that is concerned with the temporal analysis of single sentences.

In section 2.3, we consider temporal analysis work on simple linear discourse.

In section 2.4, we consider temporal analysis work that looks at simple non-linear discourse.

Finally, in section 2.5, against the background of the earlier review material, we present a summary of the problems and issues underlying temporal analysis. This will allow us to identify more clearly the issues to be addressed in the remainder of the thesis.

2.1 Basic Concepts

In this section, we discuss the basic concepts which underlie previous work in temporal analysis. These concepts are:

- Tense
- Aspect
- Temporal Ontology

Tense	Example	Relation Between ST, RT and ET
Simple Present	John runs	ST = RT = ET
Simple Past	John ran	ET = RT < ST
Simple Future	John will run	ST < RT = ET
Present Perfect	John has run	ET < ST = RT
Past Perfect	John had run	ET < RT < ST
Future Perfect	John will have run	ST < ET < RT ST = ET < RT ET < ST < RT
Present Prospective	John is going to run	ST = RT < ET
Past Prospective	John was going to run	RT < ET < ST RT < ST = ET RT < ST < ET
Future Prospective	John will be going to run	ST < RT < ET

Table 2.1: The Reichenbachian analysis of tense

2.1.1 Tense

Tense has been studied extensively and a most influential analysis of tense is due to Reichenbach [Reichenbach 47]. In a Reichenbachian account, the underlying structure of an English tense needs to be described with three time points: the time of speech (ST), the time of the event (ET) and the reference time (RT), together with two temporal relations: precedes (<) and coincides (=). The RT is the time from which the event described in the clause is viewed. For instance, a clause in a past perfect tense like *John had slipped on ice* describes a slipping event that occurs before some past time. By using RT to denote this past time, the ET of the slipping event can be located before the RT. The tense structure of a past perfect tense is thus as follows:

(2.5) ET < RT < ST

Altogether there are nine tenses in a Reichenbachian account and the ST|RT|ET configuration that they denote is given in table 2.1.

Notice that in a Reichenbachian account, the past prospective tense and the future perfect tense are ambiguous in that there is more than one possible ordering among the ST, ET and RT.

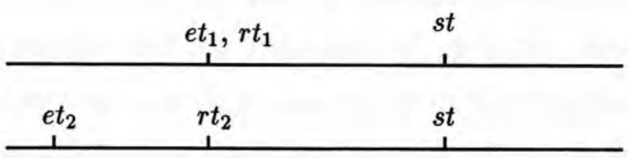


Figure 2.1: The anaphoric behaviour of tense: rt_2 corefers with et_1

The next influential development in the use of tense for temporal analysis is the view that tense is anaphoric. In this view, a tensed sentence is to be interpreted with respect to a previously established reference time which functions as its antecedent. For instance, in example (2.6), the *at*-adverbial is taken as identifying a reference time and the tensed sentence *Harry walked in* will be interpreted as coreferential with it.

(2.6) At exactly 5 o'clock, Harry walked in.

Later work by Hinrichs [Hinrichs 86] and Steedman [Steedman 82] argue that it is not tense per se that is anaphoric, but that part of tense called by Reichenbach's RT (see above). Webber [Webber 87] gave the following example to demonstrate why the RT is to be taken as anaphoric:

(2.7) a. et_1, rt_1, st: John went to the hospital.

b. et_2, rt_2, st: He had twisted his ankle on a patch of ice.

Here, it is not the ET of John's twisting his ankle that is interpreted anaphorically with respect to his going to the hospital. Rather, it is the RT of the second clause (rt_2). The referent for rt_2 is et_1 of the previous clause and since the second clause is in the past perfect tense, et_2 is positioned before rt_2 (see table 2.1). We illustrate the anaphoric nature of rt_2 in figure 2.1. Note that there is no subscript for the speech time in figure 2.1 since the discourse is assumed to be uttered at the same time.

Assuming that it is the RT of a tensed clause that is anaphoric, what is the nature of this anaphoric behaviour of tense? Work by [McCawley 71, Partee 73] claims that tense behaves like pronoun in its anaphoric behaviour. For instance, in example (2.7), just as the pronoun *he* in (2.7b) corefers with its antecedent *john* in (2.7a), the reference time rt_2 of (2.7b) corefers with the event time et_1 of (2.7a). However, as noted in [Webber 87, Moens & Steedman 88], in one respect, the past tense does not behave like a pronoun. The use of a pronoun does not change the referent to which a subsequent use of the same pronoun may refer. On the other hand, using a past tense may. Consider, for instance,

(2.8) At exactly 5 o'clock, Harry walked in. He sat down. He took off his boots.

where the referent for the successive RTs seems to advance from the time point originally set up by the adjunct. However, the referent for the pronoun *he* remains the same. To account for this discrepancy, Partee [Partee 84], Hinrichs [Hinrichs 86] and Dowty [Dowty 86] stipulate that the time point which serves as referent for successive RTs automatically advances during a narrative. This stipulation causes problems for the theory in those narratives where time appears to move backward as in example (2.3):

(2.3) a. Annie broke her leg. She fell off the bicycle.

b. Max died. John poisoned him.

To overcome the problem, Webber [Webber 87] suggests that the anaphoric behaviour of a tense is better compared to that of a definite NP. The referent for a definite NP can be a new entity which is related to a previously mentioned object. For example, in

(2.9) I picked up a banana. The skin was all brown.

where the referent of the NP *the skin* is an object which is semantically related to the object set up by the indefinite NP *a banana.* Likewise, in tense, the referent of the RT can be a new time point which is related to a previously mentioned time. Thus, for Webber, to account for the time movement in (2.8) repeated below:

(2.8) a. et_1, rt_1, st: Harry walked in.

b. et_2, rt_2, st: He sat down.

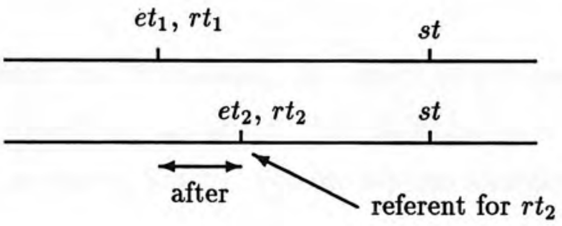


Figure 2.2: The referent for rt_2 in example (2.8)

c. et_3, rt_3, st: He took off his boots.

she would say that the RT of the second clause (rt_2) corefers with a new time point that is related to the ET of the first clause (et_1). For example (2.8), the relation is one in which the new time point is after et_1 (see figure 2.2). In other examples, the relation can be one in which the new time point is before et_1 such as in example (2.3) reproduced below:

(2.3) a. Annie broke her leg. She fell off the bicycle.

b. Max died. John poisoned him.

As to be expected, determining this relation depends on world knowledge.

To summarise, tense is commonly regarded as an anaphoric device and its anaphoric behaviour is seen as analogous to that of pronouns [McCawley 71, Partee 73] or to that of definite NPs [Webber 87]. Typically, tense is described by Reichenbach's trinity of ST, RT and ET.

2.1.2 Aspect

There are two main computational treatments of aspect represented by work like [Moens & Steedman 88, Passonneau 88] and [Song 91b, Singh & Singh 92].

Moens and Steedman and Passonneau

For Moens and Steedman and Passonneau, the aspect of a clause is taken to be the type of the situation underlying the clause. The classification of the situations used in this work is based on that of Vendler [Vendler 67] who identifies the following four kinds of situation:

State John loves Mary.

Activity John ran.

Achievement John reached the top of the mountain.

Accomplishment John built a house.

For Vendler, activities and accomplishments have successive phases while achievements and states do not. Accomplishments, activities and states extend in time while achievements are instantaneous. Finally, accomplishments are distinguished from activities in that the former have an intrinsic conclusion, whereas the latter go on in a homogeneous way. Like accomplishments, achievements also have an intrinsic conclusion. To these four Vendler's categories, Moens and Steedman [Moens & Steedman 88] add a fifth one, called POINTS, as exemplified by *John coughs.* Like achievements, points are instantaneous but unlike them, points do not have a well-defined culmination point. Moens and Steedman also use a different terminology from Vendler: activities are called PROCESSES, achievements are called CULMINATIONS, accomplishments are called CULMINATED PROCESSES. In addition, the term EVENTS is used to refer to processes, culminated processes, culminations and points.

Besides classifying the type of the situation described by a clause, Moens and Steedman also propose a tripartite structure reflecting the 'parts' of an event (Passonneau proposes a similar structure). More specifically, they suggest that an event contains a preparatory process which can lead up to (without necessarily reaching) a culmination point, and this in turn has certain consequences attached to it [Moens 87, p 47]. The tripartite structure is shown in figure 2.3.

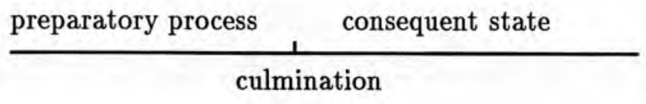


Figure 2.3: The tripartite structure of an event

Song and Singh and Singh

A different approach of handling aspect and its interaction with temporal adverbials is given in [Song 91b, Singh & Singh 92]. This work views aspect as the viewpoint of a speaker towards a situation which is distinct from situation type. More specifically, the approach postulates a structure for the event underlying a sentence and the aspect of the sentence indicates which part of this event structure is being referred to by the speaker.

The event structure proposed by [Song 91b, Singh & Singh 92] is slightly different from the tripartite structure of Moens and Steedman. Song and Singh and Singh note that one problem with the tripartite structure is that it uniformly regards situations that occur before the culmination point as being in the preparatory phase. This arises since the tripartite structure represents explicitly only the end point but not the start point of an event . Consequently, the tripartite structure makes it difficult to distinguish between events that provide preconditions to a specific event and events that elaborate details of the specific event. Temporally, the former events should be located entirely before the specific event while the latter events entirely within the specific event. For example, consider the event of making a phone call. The various steps that occur before reaching the culmination point when the phone call has been completed include looking up a phone number, picking up a phone, dialing the number and waiting for connection. Clearly, the event of looking up a phone number should occur before the event of making a phone call while the event of picking up a phone, dialing the number and waiting for connection should be during the event of making a phone call. However, using the tripartite structure, all these events are represented as before the end point of the event of making a phone call. Therefore, [Song 91b, Singh & Singh 92] suggest distinguishing

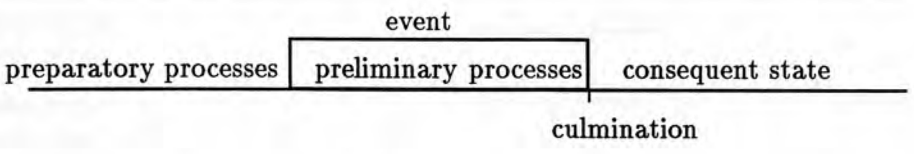


Figure 2.4: The four part structure of an event

between processes that occur during the event and leading up to the culmination point from processes that occur before the event. Following [Singh & Singh 92], we will call the former PRELIMINARY PROCESSES while the latter are called PREPARATORY PROCESSES. The structure of an event is now as in figure 2.4. In [Singh & Singh 92], three basic aspects are proposed.[1]. The PERFECTIVE aspect describes a situation as a complete whole, for example, *John ate an apple.* The PROGRESSIVE aspect presents an event internally by referring to the preliminary processes part of the event structure (see figure 2.4); for example, *John was eating an apple.* The third kind of aspect is the NEUTRAL PERFECTIVE aspect and describes an event that has ended, but not necessarily at its natural end point, for example, *John ate an apple (but not all of it).*

In our work, we adopt the event structure proposed in [Song 91b, Singh & Singh 92] and view aspect as indicating which part of this structure is referred to by the clause (see chapter 4).

2.1.3 Temporal Ontology

There has been much debate on the appropriate temporal ontology on which to base the semantics of temporal expressions in natural language. Consider, for instance, the analysis of the *before* connective in

(2.10) The compressor failed before the pump seized.

One approach [Hirschman & Story 81] takes time points as its temporal ontology and equates events with time points. The semantics of temporal expressions are then

[1]Song [Song 91b] proposes a more detailed aspect classification based on time interval.

A *before* (<) B
B *after* (>) A

A *meets* (*m*) B
B *met-by* (*mi*) A

A *overlaps* (*o*) B
B *overlapped-by* (*oi*) A

A *starts* (*s*) B
B *started-by* (*si*) A

A *during* (*d*) B
B *contains* (*c*) A

A *finish* (*f*) B
B *finished-by* (*fi*) A

A *equals* (=) B

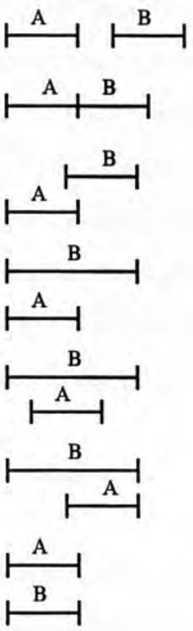


Figure 2.5: Allen's thirteen relations

taken to be indicating some relation between two time points. Now, there are only three possible ordering relations between two time points, namely either <, > or =. In example (2.10), the *before* connective is taken as indicating that the time point associated with *the compressor failed* precedes (<) the time point associated with *the pump seized.* Typically, an event is equated with the ET of Reichenbachian tense structure denoting the tense of the clause describing the event.

Another approach [Allen 83] takes time intervals as its temporal ontology and equates events with time intervals. The semantics of temporal expressions are then taken to be indicating some relation between two time intervals. Unlike the case for time points, there are more relations between time intervals. Allen [Allen 83] identifies thirteen possible such relations as given in figure 2.5. For example (2.10), assume that *the compressor failed* is assigned to time interval int_1 and *the pump seized* is assigned to time interval int_2. Then, the *before* connective is taken as indicating that int_1 is either before or meets int_2. This is commonly written as int_1 $\{<, m\}$ int_2.

A third approach [Moens & Steedman 88] argues that the semantics of temporal expressions should not be directly related to a linear time concept, that is to a model based on the number line. This is exactly what the two approaches discussed above do. Instead, Moens and Steedman argue for the use of situations as the basic temporal ontology. They give *when*-clause as an example of the mismatch between linguistic temporal connectives and a semantics based on the time line conception [Moens & Steedman 88, p 15]. For instance, Moens and Steedman point out that directly reducing situations to time points or intervals would not explain the peculiarity of the following clause:

(2.11) * When my car broke down, the sun set.

This is because, if we regard the event described by *my car broke down* as only a time point or time interval and the situation described by *the sun set* as another time point or time interval, then there will be a perfectly legitimate temporal relation existing between these two time points or time intervals. But, the unusual character of example (2.11) is because the *when-clause* predicates something more than mere temporal coincidence, i.e. some relation such as a causal link between the two events.

In our work, we adopt situations as the basic temporal ontology at the level of the QTS. On the other hand, at the level of the temporal structure, we adopt time intervals and time points as the basic temporal ontology. This allows us to overcome the limitations of the individual approaches discussed above.

2.2 The Temporal Analysis of Single Sentences

In this section, we discuss work that carries out temporal analysis of single sentences. In particular, we consider the computational work of Passonneau [Passonneau 88] in the PUNDIT system and the treatment of *when*-clauses as proposed in [Moens & Steedman 88, Singh & Singh 92].

2.2.1 Temporal Analysis in PUNDIT

The temporal analysis component in PUNDIT [Passonneau 88] is originally designed to handle CASREP messages. These messages are reports describing equipment failures on navy ships. Each of these reports consists of simple declarative clauses. Some examples of these clauses are:

(2.12) a. The pressure was low at 0800.

b. The alarm sounded at 0800.

c. The alarm was sounding at 0800.

d. The pump failed at 0800.

The main goal of the temporal analysis component in PUNDIT is to extract the temporal information given in a clause using tense and aspect.

As in [Moens & Steedman 88], PUNDIT regards the aspect of a clause as describing the type of the situation underlying the clause. A situation in PUNDIT is assumed to hold over a time interval and contained in this interval is a time point called the ET[2]. The ET may be explicitly indicated in the clause. For instance, in (2.12a – d), the *at* adjunct explicitly indicates the ET. The temporal information that is extracted from a clause by PUNDIT then consists of the following:

- the ET of the situation described by the clause; and
- the relation of ET with the time interval associated with the situation.

In order to derive the temporal information from a clause, PUNDIT identifies three main types of situations and based on the situation type to determine the relation between the ET and the time interval associated with the situation.

The three main types of situations identified in PUNDIT are:

- States (example (2.12a))

[2]This is the same as the ET in Reichenbachian tense structure.

- Processes (example (2.12b) and (2.12c))
- Transition Events (example (2.12d))

States and processes are equivalent to Vendler's states and activities. However, Vendler's achievements and accomplishments are now collapsed into PUNDIT's transition events which hold over a time interval. This is because Passonneau believes that achievements, which are considered to be instantaneous, can be argued to hold over a time interval whose duration is typically shorter than that for accomplishments.[3] We will now discuss these situations:

- A state describes a static situation and is inherently unbounded in time. For instance, the clause *The pressure was low at 0800* describes a state that holds not only at 0800, but possibly prior to and subsequent to 0800.
- A process describes a dynamic situation. As noted in [Passonneau 88], a distinction between static and dynamic situations is that the former cannot be modified by rate adverbials like *slowly* and *rapidly.* This is because states are characterised by an absence of change over time. Processes can themselves be subdivided into two kinds:
  - An unbounded process, like a state, is temporally unbounded, as for instance, *The alarm was sounding at 0800.*
  - An unspecified process like *The alarm sounded at 0800*, on the other hand, is temporally bounded. However, for an unspecified process, there is an ambiguity as to how the explicitly given time point (0800) is related to the time interval associated with the process. For our current example, 0800 can either start the time interval, end the interval or be included in the interval.
- A transition event such as that described by *The pump failed at 0800* is temporally bounded. The explicit time point of 0800 marks the end point of the process of failing. Like a process, a transition event describes a dynamic situation.

[3] In chapter 3, we make a similar argument for associating situations with time intervals.

Based on the situation type, PUNDIT can determine the relation between the ET and the time interval of the situation:

- Since a state or unbounded process is temporally unbounded, ET is assumed to be an interior point of the time interval.
- For an unspecified process, PUNDIT encodes ambiguously that the ET can either start, end or be included in the time interval.
- For a transition event, the ET is taken to be the end point of the time interval. This end point represents the transitional moment at which the event ends and a new state or a process starts.

Besides simple declarative clauses discussed above, PUNDIT also has a rudimentary treatment of temporal connectives. Consider for instance

(2.13) The compressor failed before the pump seized.

Here, the *before* connective is taken as indicating a precedence relation between the RT of the main clause and the RT of the subordinate clause. Thus, PUNDIT employs time points as the temporal ontology for describing the semantics of temporal connectives.

To summarise, PUNDIT deals only with single clauses. It also has a limited treatment of clauses connected by temporal connective. PUNDIT employs both time points and time intervals to represent the temporal information found in single clauses. This information is derived from the tense and aspect of a clause. The aspect of a clause determines the type of the situations underlying the clause. Depending on the situation type, PUNDIT postulates a relation between the ET and the time interval associated with the situation.

2.2.2 The Temporal Analysis of When-Clauses

As discussed above, PUNDIT deals mainly with simple declarative sentences and takes temporal connectives as representing the relation between the RT of the main clause

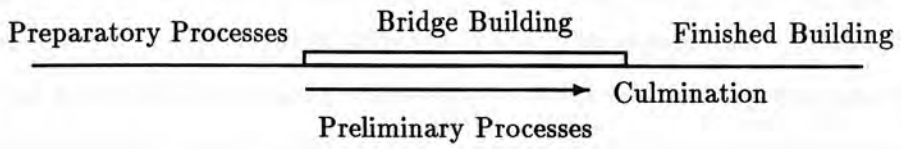


Figure 2.6: The bridge building event structure

and the RT of the subordinate clause. As pointed out in [Moens & Steedman 88], this treatment of the temporal connectives is insufficient especially in the case of the *when*-connective. Recall from section 2.1.3 that Moens and Steedman believe that the *when* connective indicates a contingency relation between the situation described in the *when* clause and that described in the main clause.

To analyse sentences with *when*-clauses, Moens and Steedman propose the following approach. According to them, a *when*-clause brings into focus a novel temporal referent which is associated with a tripartite event structure (see figure 2.3). The event described by the main clause can refer to any part of this structure conditional on support from general or discourse specific knowledge [Moens & Steedman 88, p 23]. Later work of [Singh & Singh 92] suggests essentially the same approach to analysing the *when*-clause. However, as discussed in section 2.1.2, Singh and Singh differ from Moens and Steedman in that they take aspect as the viewpoint of a speaker towards a situation and the event structure consists of four parts instead of three: the preparatory processes before the start of the event, the preliminary processes during the event, the culmination which the preliminary processes lead up to and the consequent state.

We will now consider how the above approach analyses sentence (2.14a) and sentence (2.14b) given below. In this analysis, we will adopt the notion of aspect-as-viewpoint and the four part event structure. The event structure associated with the bridge building event is given in figure 2.6.

(2.14) a. When they built the bridge, a local architect drew up the plan.

b. When they were building the bridge, a local architect drew up the plan.

In example (2.14a), the *when*-clause is describing the building event from a perfective viewpoint (this can be determined from the verb). This means that the whole event structure in figure 2.6 is available for situating the event described by the main clause. From world knowledge, the drawing up of the plan is situated in the preparatory stage of the event structure. This, in turn, gives rise to the temporal relation that the event of drawing up the plan occurs before the event of building the bridge.

In example (2.14b), the *when*-clause is describing the building event from a progressive viewpoint (this is determined from the use of the progressive tense). This means that only the preliminary processes part of the event structure are available for situating the event described by the main clause. Therefore, the event of drawing up the plan must be during the building event. Note that no world knowledge can override this effect of the progressive.

In our work, we integrate what have been done on single sentences into our temporal analysis framework for discourse of more than one sentence.

2.3 The Temporal Analysis of Simple Linear Discourse

In this section, we consider work that carries out temporal analysis of simple linear discourse of more than one sentence. In particular, we consider the work of [Hinrichs 86] and [Hirschman & Story 81].

2.3.1 Hinrichs

Hinrichs [Hinrichs 86] considers linear narratives consisting of sequences of simple past tense sentences:

(2.2) a. John got up. He poured himself a coffee.

b. John got up. He was in a bad mood.

As noted, a linear narrative is one in which the temporal order of events described does not contradict the order of the sentences. By making this assumption, Hinrichs is able to focus on the problem of deciding when the situation described by the current clause in the discourse is interpreted as following the situation described by the previous clause (example (2.2a)) and when as overlapping it (example (2.2b)).

Hinrichs bases his account on the type of the situation underlying a clause. In particular, he uses Vendler's classification of a situation into state, activity, achievement and accomplishment. Assuming a current reference time (ref_{curr}) in a discourse, a new clause describing an achievement or accomplishment will introduce a new reference time ordered after ref_{curr} and the event described by the new clause is interpreted as temporally included in this new reference time. On the other hand, a current clause describing an activity or state does not introduce a new reference time and the situation underlying the clause is taken as surrounding ref_{curr}.

Consider for instance example (2.2a):

(2.2a) John got up. He poured himself a coffee.

The first sentence in this discourse denotes an accomplishment and thus introduces a reference time r_1 which temporally includes the event underlying the sentence. The second sentence also denotes an accomplishment and therefore, it introduces a new reference time r_2 which is ordered after r_1. The event described by the second sentence is temporally included in r_2 and thus we have the interpretation that the pouring coffee event is after the getting up event.

Now, consider example (2.2b):

(2.2b) John got up. He was in a bad mood.

As in example (2.2a), the first sentence introduces a reference time r_1 which temporally includes the getting up event. The second sentence denotes a state and thus does not introduce a new reference time. The state described by the second sentence is taken as temporally surrounding r_1 and thus resulting in the interpretation that John being in a bad mood temporally surrounds the event of John getting up.

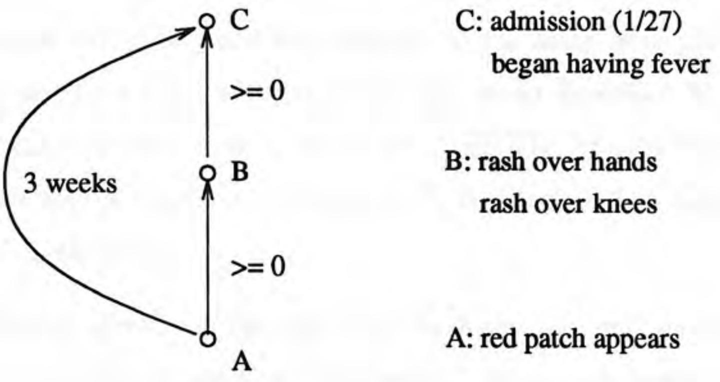


Figure 2.7: The time graph of example (2.15)

Hinrichs takes the reference time as providing a referent for Reichenbach's RT. As noted earlier, Hinrichs's account is unable to account for the backward movement of time in some discourses.

2.3.2 Hirschman and Story

Hirschman and Story [Hirschman & Story 81] describe a system for analysing the temporal information found in medical narratives. An example of such a narrative is shown in (2.15):

(2.15) a. 3 weeks prior to the current admission, a bright red patch appeared under the patient's eye.

b. The patient developed a maculopapular rash over hands and knees.

c. On 1/27 she began having a fever to 104.

where the current admission is on 1/27. The temporal information obtained from analysing (2.15) is represented in a time graph shown in figure 2.7.

The nodes in a time graph are the time points associated with the events described in the narratives while the edges denote the distance between the time points. If two events occur at the same time, they are assigned to the same time point. For example, in figure 2.7, the event described by *began having a fever* and *the current admission* occur on the same day (1/27) and are assigned to the same time point C. Node A in the figure is the time point associated with the event described by *the red patch appeared.* The edge between node A and node C indicates the distance between the events associated with A and C. In example (2.15), this distance is 3 weeks as indicated by the phrase *3 weeks prior.*

From the discussion above, we can say that the basic temporal ontology employed by Hirschman and Story is based on time points. Events are associated with time points and relations between events are reduced to relations between time points. Such relations are determined using a combination of the narrative convention and syntactically-based heuristics:

- The narrative convention provides certain default time relations between events described by consecutive clauses. More specifically, the convention provides that in narrative, time does not move backward unless an explicit time marker is provided. For example, in

  (2.16) a. 3 weeks prior to the current admission, a bright red patch appeared under the patient's eye.

  b. The patient developed a maculopapular rash over hands and knees.

  the rash-over-hands and rash-over-knees events described in (2.16b) have a default time greater than or equal to the bright-red-patch-appearing event described in (2.16a).

- The syntactically-based heuristics are used to determine the temporal relation between the time points described by various clauses in a sentence. One such heuristic deals with sentences with subordinate clause such as

  (2.17) A spinal tap was performed which revealed 806 red blood cells.

  It states that the time of a subordinate clause is equal to the time of its main clause, provided no explicit time information occurs in the subordinate clause. In (2.17), no such explicit information is given (for example, via a temporal adjunct like *after* or *before*) and thus, the events described by the main and subordinate clause are assigned to the same time point. That is, the two events are coincident.

Summary and Discussion

To summarise, Hirschman and Story use a time graph representation to represent the temporal information given in medical narratives. Their approach uses time points as the basic temporal ontology and reduces the semantics of temporal expressions to that based on the relation between time points. Narrative convention and syntactically-based heuristics are used to determine the relations between time points.

As noted in section (2.1.3), reducing the semantics of temporal expressions to that based on the relation between time points faces difficulties especially in dealing with *when*-clause. For instance, using Hirschman and Story's heuristics, one is able to obtain a coincidence relation for the apparently anomalous sentence discussed earlier:

(2.18) * When the sun set, the car broke down.

2.4 The Temporal Analysis of Non Linear Discourse

As noted in section 2.1.1, the approach for linear texts is problematic for non-linear texts where time may appear to move backwards and where the texts may contain embedded discourses. Any approach for non-linear text must account for why time moves backward and also handle the problem of integrating the situation described by the current clause into the temporal structure of the preceding discourse. Recall from chapter 1 that the integration problem involves finding suitable preceding events for interpreting the current situation and determining the nature of this interpretation.

We will now describe previous work that addresses the temporal analysis of non-linear texts. The pieces of work reviewed are:

- Webber [Webber 87]

- Song and Cohen [Song & Cohen 91b]
- Hwang and Schubert [Hwang & Schubert 91]
- Eberle [Eberle 92]
- Lascarides and Asher [Lascarides & Asher 91]

2.4.1 Webber

A basic premise of Webber's approach is that in processing a narrative text, a listener is building up a representation which reflects his/her best effort at interpreting the speaker's ordering of those events and states in time and space. Webber calls this representation the event/situation structure (e/s structure). According to Webber [Webber 87, p 147], the listener's problem can then be viewed as that of establishing where in the evolving e/s structure to attach the event or state described in the next clause. This is essentially the integration problem.

Webber's approach is based on the tense-as-anaphor idea. However, as noted in section 2.1.1, instead of regarding the anaphoric behaviour of tense to be like pronoun, Webber regards tense's anaphoric behaviour to be more like that of a definite NP. Basically, Webber employs Reichenbach's tense structure and regards the RT as being anaphoric. The referent for RT can be either the ET of a previously mentioned event or a time point related in some manner to the ET of a previously mentioned event. There are three possible cases, as illustrated by the following two-clause discourses (from [Webber 87]):

(2.19) a. st, et_1, rt_1: John walked across Iowa.

b. st, et_2, rt_2: He thought about Mary.

(2.20) a. st, et_1, rt_1: John walked across Iowa.

b. st, et_2, rt_2: He crossed the state line at Council Bluffs.

(2.21) a. st, et_1, rt_1: John walked across Iowa.

b. st, et_2, rt_2: He started in Sioux City.

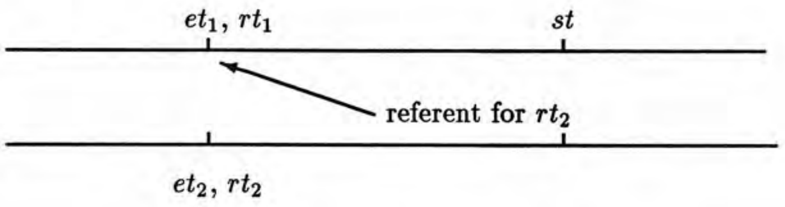


Figure 2.8: The e/s structure of example (2.19)

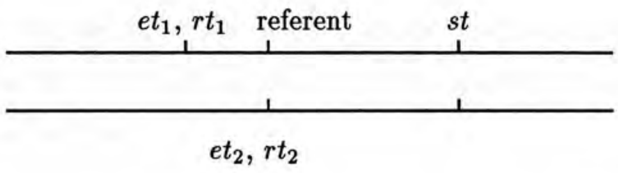


Figure 2.9: The e/s structure of example (2.20)

In (2.19), the referent for rt_2 of the second clause is et_1 itself. Since the clause is in the simple past tense, it positions et_2 (see table 2.1) as well. The e/s structure is as shown in figure 2.8. Notice that et_2 is coincident with et_1 giving rise to the interpretation that the event of John thinking about Mary is coincident with the event of John walking across Iowa.

In (2.20), the referent for rt_2 of the second clause is a time point that is after et_1. Since the clause is in the simple past tense, the referent sites et_2 also (see figure 2.9). Notice that et_2 is after et_1 resulting in the interpretation that the event of John crossing the state line follows the event of John walking across Iowa.

Finally, in (2.21), the referent for rt_2 of the second clause is a time point that is before et_1. Since the clause is in the simple past tense, the referent sites et_2 also (see figure 2.10). Note from the figure that et_2 is before et_1. This results in the temporal interpretation that the event of John starting in Sioux city is before the event of John walking across Iowa.

As acknowledged in [Webber 87, p 150], deciding which of the above three cases holds demands an appeal to world knowledge, an area not examined by Webber. The main

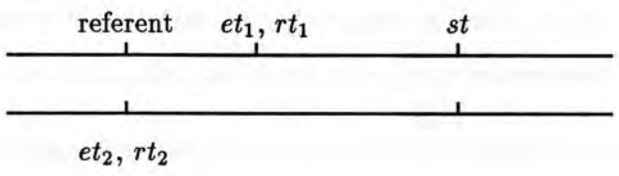


Figure 2.10: The e/s structure of example (2.21)

concern for Webber is how to determine the referent for the RT of the current clause. This referent will site the RT of the current clause and thus indirectly determines the location of the ET of the clause based on the relation between ET and RT in Reichenbach's tense structure.

To determine the referent for RT, Webber proposes a notion of temporal focus. The temporal focus (TF) is the temporal analogue of the notion of focus in the nominal domain [Sidner 79]. The referent for RT can relate to the TF in three ways: it can be either before, after or coincide with the TF. Thus, the TF serves as a time point against which the RT is to be interpreted.

Webber's Temporal Focus Mechanism

There are two basic problems to solve with the focus idea. Firstly, it is necessary to determine what a focus is; and secondly, a mechanism is needed for tracking the focus. We now examine Webber's proposal with respect to these two issues.

In Webber's proposal, the TF is typically the ET of a previous event. When there is no such previous event, Webber creates a new node of e/s structure, ordered prior to the speech time to serve as the TF. There are three potential movements of the TF: (1) maintain the current TF; (2) create a new TF; and (3) resume a previous TF.

In order to determine the possible movements of the TF, Webber proposes four heuristics which have to be applied in parallel rather than serially. When more than one heuristic is applicable, Webber resorts to an unexplained notion of plausibility to decide which heuristic should fire. The four heuristics [Webber 87, p 151] are given in

Focus Maintenance Heuristic: After interpreting clause n, the new TF is the most recent TF — i.e. the node against which rt_n was interpreted.

Embedded Discourse Heuristic 1: If et_n is different from $rt_n = \text{TF}$, treat utterance n as the beginning of an embedded narrative, reassign et_n to TF (stacking the previous value of TF, for possible resumption later) and try to interpret rt_{n+1} against this new TF.

Embedded Discourse Heuristic 2: If clause $n + 1$ is interpreted as beginning an embedded narrative, create a new node of e/s structure and assign it to be TF. Stack the previous value of TF for possible resumption later

Focus Resumption Heuristic: At the transition back from an embedded narrative, the TF prior to the embedding (stacked by an Embedded Discourse Heuristic) can be resumed.

Figure 2.11: Webber's temporal focus tracking heuristics

figure 2.11.

We will now briefly describe how the above heuristics are used to decide on the TF movement. Consider, for instance,

(2.22) a. et_1, rt_1, st: John went over to Mary's house.

b. et_2, rt_2, st: On the way, he had stopped by the flower shop for some roses.

c. et_3, rt_3, st: Unfortunately, the roses failed to cheer her up.

When clause (2.22a) is processed, there is no previous event and so a new TF (TF_1) is set up prior to the speech time. The RT of the clause (rt_1) is interpreted to coincide with TF_1 and since the clause is in the simple past, it sites the ET of the clause (et_1) also. At this point, it is necessary to decide whether to maintain TF_1 as the focus for interpreting the RT of the second clause. Only the Focus Maintenance Heuristic can

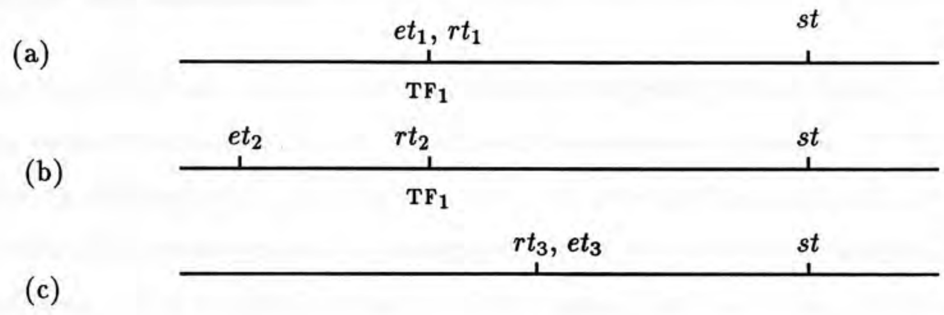


Figure 2.12: The stages of the e/s structure of example (2.22)

apply and so TF$_1$ is maintained as the current TF for interpreting the RT of the next clause. The e/s structure is as shown in figure 2.12a.

The RT of clause (2.22b) (rt_2) is interpreted as coinciding with TF$_1$. Since the clause is in the past perfect tense, the ET of the clause (et_2) is sited at a point before rt_2. At this point, it is necessary to decide the movement of the focus for interpreting the RT of the third clause. Both the Focus Maintenance Heuristic and the Embedded Discourse Heuristic 1 can apply. As noted above, an unexplained notion of plausibility is used to decide which of these heuristic should apply. In this example, it is the Focus Maintenance Heuristic which wins the plausibility stake against the Embedded Discourse Heuristic 1. Thus, TF$_1$ is maintained as the current focus. The e/s structure is shown in figure 2.12b.

The RT of the third clause (rt_3) is interpreted as immediately following TF$_1$. Since this clause is in the simple past tense, rt_3 in turn directly sites et_3. The e/s structure is now as in figure 2.12c (TF is not shown since there are no more clauses).

Notice that from the e/s structure of figure 2.12, Webber can determine that the event of john going over to Mary's house (et_1) is preceded by the event of john buying some roses (et_2) and followed by the event of the rose failing to cheer Mary up (et_3).

Summary and Discussion

Webber's work focuses mainly on the problem of integrating the situation described by the current clause into the e/s structure of the preceding discourse. In Webber's framework, this problem can be reformulated as the problem of determining the referent for the RT of the current clause. To determine this referent, a notion of temporal focus is employed. More specifically, the RT of the current clause is to be interpreted as either following, preceding or coincident with the temporal focus. Four heuristics are proposed to track the focus.

Webber's proposal has some serious problems. First, the notion of plausibility needed to decide which heuristics should apply is not very well-defined. Second, Webber does not specify how an embedded discourse is to be recognised; such information is required by Embedded Discourse Heuristic 2. Third, Webber does not elaborate further on how to determine whether the RT of a current clause is to follow, precede or coincide with the current TF. Finally, Webber essentially equates situations with time points and therefore does not consider the effect of duration.

To overcome some of these problems, Song and Cohen [Song & Cohen 91b] make some extensions to Webber's work. We will now turn to this work.

2.4.2 Song and Cohen

Song and Cohen [Song & Cohen 91b] extend Webber's work in four areas.

First, Song and Cohen incorporate aspect into Webber's framework and following [Passonneau 88] classify situations into three types: states, processes and transition events. As discussed in section 2.2.1, these situations are assumed to hold over a time interval and depending on the type of the situations, the relation between the ET of the situation and its time interval can be determined. For instance, ignoring time intervals, the temporal structure of example (2.23):

(2.23) a. et_1: John went to a hospital.

b. et_2: He had fallen on a patch of ice.

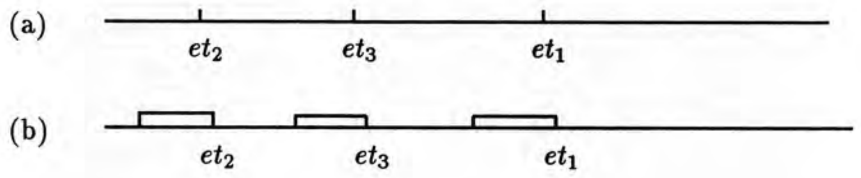


Figure 2.13: Temporal structure in terms of time intervals

c. et_3: and had twisted his ankle.

is as given in figure 2.13a. Now, all three sentences in (2.23) denote transition events and thus as noted in section 2.2.1, et_1 to et_3 are taken to be the end points of the time intervals associated with the events described by the three sentences. Incorporating time intervals, the temporal structure of example (2.23) is as given in figure 2.13b. Although Song and Cohen associate events with time intervals, they do not make use of quantitative information such as event's duration in their temporal analysis framework.

Second, they propose to use a standard SRE triple of the form [ST R1 RT R2 ET] where R1 and R2 are taken from the set of relations $\{<, >, =\}$ to denote the nine tenses in Reichenbach's account. The nine tenses and their corresponding SRE triple are shown together with Reichenbachian tense structure in table 2.2.

Notice that in Song and Cohen's scheme, each of the nine tenses is associated with a unique SRE triple.

Third, Song and Cohen argue that it is insufficient to consider only the RT of a clause as anaphoric. They argue that it is also necessary to regard the ET of a clause as anaphoric. Consider example (2.23) again:

(2.23) a. et_1, rt_1, st: John went to a hospital.

b. et_2, rt_2, st: He had fallen on a patch of ice.

c. et_3, rt_3, st: and had twisted his ankle.

For example (2.23), we should be able to decide that $et_1 > et_2$, $et_1 > et_3$ and $et_2 <$

Tense	Song and Cohen	Reichenbach
Simple Present	[ST = RT = ET]	ST = RT = ET
Simple Past	[ST > RT = ET]	ET = RT < ST
Simple Future	[ST < RT = ET]	ST < RT = ET
Present Perfect	[ST = RT > ET]	ET < ST = RT
Past Perfect	[ST > RT > ET]	ET < RT < ST
Future Perfect	[ST < RT > ET]	ST < ET < RT ST = ET < RT ET < ST < RT
Present Prospective	[ST = RT < ET]	ST = RT < ET
Past Prospective	[ST > RT < ET]	RT < ET < ST RT < ST = ET RT < ST < ET
Future Prospective	[ST < RT < ET]	ST < RT < ET

Table 2.2: Song and Cohen SRE triples

et_3 (since the falling may be a cause for the twisting). Following Webber's approach described in section 2.4.1, it is possible to determine that $et_1 > et_2$ after the processing of (2.23b). At this point, it is necessary to decide the temporal focus for interpreting rt_3. Both the Focus Maintenance Heuristic and the Discourse Maintenance Heuristic 1 are applicable. If the Focus Maintenance Heuristic is used, then rt_3 will be interpreted against the previous TF positioned at et_1 and we will be able to obtain the temporal relation that $et_1 > et_3$. However, no information is derived regarding the relation between et_2 and et_3. On the other hand, if the Embedded Discourse Heuristic 1 is applied, then rt_3 will be interpreted against a new TF positioned at et_2 and we will get the relation that $et_2 > et_3$ which is wrong, since falling on a patch of ice may be the cause of twisting one's ankle. To get the right interpretation, Song and Cohen suggest that one needs to take rt_2 as the referent for rt_3 and a point after et_2 as the referent for et_3 (see figure 2.14). In other words, both the RT and the ET of the current clause have to be anaphoric requiring previously mentioned time points as referents.

Given that Song and Cohen are taking both the RT and ET as anaphoric, Webber's one-point focus (TF) is insufficient. Instead, Song and Cohen introduce the concept

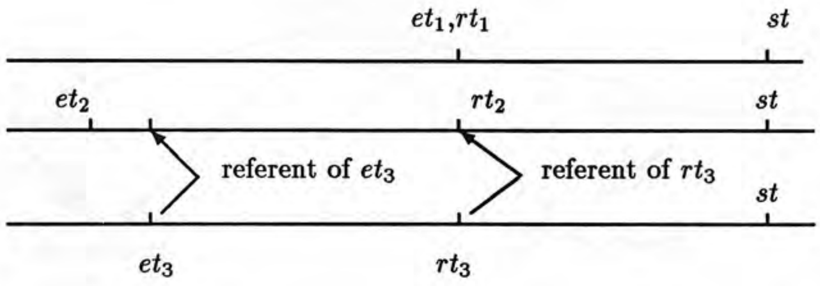


Figure 2.14: The referents of rt_3 and et_3 in example (2.23)

of a temporal focus structure (TFS) which is the SRE triple of a previous clause. The TFS contains the focus points for interpreting the RT and ET of the current clause. As in [Webber 87], there are three possible movement of the current TFS: (1) maintain the current TFS (2) create a new TFS and (3) resume a previous TFS. Unlike Webber's non-deterministic strategy of tracking the focus movement, Song and Cohen propose a deterministic strategy. In their strategy, Song and Cohen arrange the SRE triples of the nine tenses into a tense network.[4]

Song and Cohen's Tense Network

Song and Cohen's tense network is as shown in figure 2.15.[5] Note the thick and thin link in the network.

Based on the tense network, Song and Cohen propose the following scheme for tracking the movement of the TFS. At the point of deciding the focus movement, the information available includes the SRE triple of the current TFS (α) and the SRE triple of the current clause (β). Then, Song and Cohen's proposal for determining the movement of the current TFS can be summarised in the following four rules (the first two rules are tried first before the last two):

(2.24) If there is a thin link from α to β in the tense network

[4]In their work, Song and Cohen call this tense network a tense hierarchy.

[5]To ease readability, we show the actual tense instead of its corresponding SRE triple.

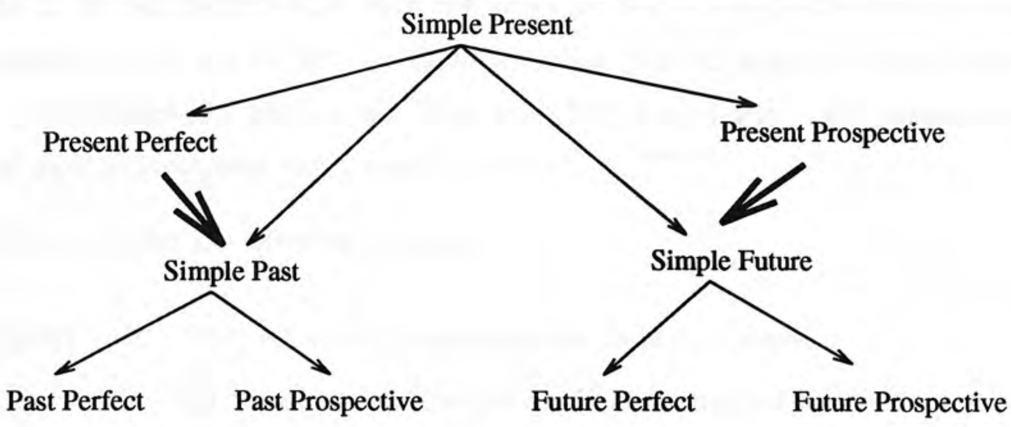


Figure 2.15: Song and Cohen's tense network for English

then create a new TFS

(2.25) If there is a thick link from α to β in the tense network
or α and β are the same tense
then maintain the current TFS

(2.26) If there is a link between α and β
then resume a previous TFS such that either
this previous TFS and β are the same or
there is a thick or thin link from this previous TFS to β

(2.27) If there is no link between α and β
then the tense sequence is incoherent
and stop processing

We are now in a position to illustrate how Song and Cohen's proposal works. First, consider example (2.28):

(2.28) a. $[st = rt_1 = et_1]$: John is staying at home.

b. $[st > rt_2 > et_2]$: He had finished his homework.

Initially, Song and Cohen set up the current TFS to be $[st = rt_1 = et_1]$ since the first sentence is in the progressive present. The second clause is now processed. This clause is in the past perfect tense with SRE triple $[st > rt_2 > et_2]$. Therefore α is simple present and β is past perfect. From the tense network, there is no link between a simple present and past perfect. Thus, rule (2.27) is applicable and the tense sequence is ruled as incoherent and processing stopped.

Next, consider the following example:

(2.29) a. $[st > rt_1 = et_1]$: John went over to Mary's house.

b. $[st > rt_2 > et_2]$: On the way, he had stopped by the flower shop for some roses.

c. $[st > rt_3 > et_3]$: He had picked 5 red ones, 3 white ones and one pale pink.

d. $[st > rt_4 = et_4]$: Unfortunately, the roses failed to cheer her up.

After processing the first sentence, the current TFS is set up to be $[st > rt_1 = et_1]$. To interpret et_2 and rt_2 of the second clause, it is necessary to decide on the movement of the current TFS. Now, α is simple past and β is past perfect. From the tense network, there is a thin link from simple past to past perfect. Rule (2.24) is therefore applicable and a new focus is created: $[st > rt_1 > et_1^{'}]$ following the tense structure of the second sentence. Note that $et_1^{'}$ is a time point before et_1. Then, rt_2 is interpreted as coinciding with rt_1 in this focus and et_2 is interpreted as coinciding with $et_1^{'}$ in the focus. As a result of this interpretation, Song and Cohen obtain the relation that $et_1 > et_2$. They also update the current TFS to $[st > rt_2 > et_2]$. The previous TFS ($[et_1 = rt_1 < st]$) is stacked.

We now interpret et_3 and rt_3 of the third clause which is in past perfect. At this point, the current TFS is $[st > rt_2 > et_2]$. Therefore, α is past perfect and β is also past perfect. Rule (2.25) therefore applies and the current focus is maintained. The ET of the current clause (et_3) is interpreted to follow et_2 ($et_3 > et_2$) in the focus while rt_3 is interpreted to coincide with rt_2. The current TFS is updated to $[st > rt_3 > et_3]$.

Finally, we interpret et_4 and rt_4 of the fourth clause which is in the simple past. At this point, the current TFS is $[st > rt_3 > et_3]$. Thus, α is past perfect while β is simple

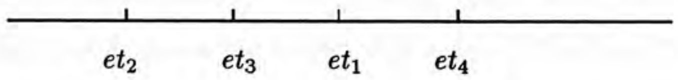


Figure 2.16: The temporal structure derived by Song and Cohen for example (2.29)

past. Rule (2.26) applies since there is a link between a past perfect and simple past but no thin or thick link from a past perfect to a simple past. The previous focus ($[et_1 = rt_1 < st]$) is resumed since this previous focus also denotes a simple past. The event et_4 is interpreted to follow et_1 ($et_4 > et_1$) and rt_4 is interpreted to coincide with rt_1.

The final temporal structure for example (2.29) is shown in figure 2.16.

Notice that et_4 is positioned correctly after et_1 as a result of resuming the previous focus. This is flagged by the explicit tense change from past perfect in (2.29c) to the simple past in (2.29d).

Summary and Discussion

Song and Cohen's approach attempts to overcome some of the difficulties of Webber's approach. In particular, they suggested that both the ET and the RT of Reichenbach's tense structure need to be treated as anaphoric. Furthermore, Song and Cohen propose a deterministic strategy using heuristics based on tense change (as captured in the tense network) to decide on the focus movement. However, these heuristics do not work all the time and in some cases would suggest the wrong result. For instance, consider example (2.30).

(2.30) a. Jack and Sue went to a hardware store to buy a new lawnmower.

b. Their old one had been stolen.

c. Sue saw the men who took it and

d. chased them down the street.

In interpreting the RT and ET of the third clause, we have to decide the movement of the current focus given by the SRE triple of the past perfect tense of the second clause. In this case, α is past perfect and β is simple past. Rule (2.26) applies and the previous focus (the SRE triple of the simple past tense of the first clause) is resumed. This means that the event of *Sue saw the men who took it* will be interpreted against the event described by *Jack and Sue went to a hardware store.* This is clearly not correct.

In addition, Hwang and Schubert [Hwang & Schubert 91] note that Song and Cohen's tense network will rule out the following coherent tense sequence in example (2.31) since there is no link between a simple present and a past perfect in the tense network.

(2.31) a. Mary is angry about the accident.

b. The other driver had been drinking.

2.4.3 Hwang and Schubert

Hwang and Schubert focus on the problem of how to represent complex events described by sentences like *Everyone looked at Mary* (see chapter 3) and the problem of how to choose the event described by a previous sentence against which the event described by the current sentence is to be interpreted. Hwang and Schubert call the previous event a REFERENCE EVENT for the current event.[6] Knowing the relation between the current event and its reference event in this interpretation is not the prime concern in [Hwang & Schubert 91]. As Hwang and Schubert note [Hwang & Schubert 91, p 35], whatever the relation may be (they use the predicate *orients* to 'subsume' the possible relations that can hold between two events), finding the right pair of events involved in such relations is of crucial importance in discourse understanding.

Like Webber and Song and Cohen, Hwang and Schubert study the role which tense contributes to a solution of finding the reference event for the current event. However, unlike [Webber 87, Song & Cohen 91b] which analyses tense in a Reichenbachian manner, uses a notion of temporal focus and a set of heuristics for tracking the movement of the focus, Hwang and Schubert make the following proposals:

[6]Hwang and Schubert use the term EPISODE instead of event.

- They regard tense as sentential operator operating on the logical form (LF) of a sentence and associate rules with sentential operators.

- They propose a contextual structure called *tense tree* which is built up by the rules associated with sentential operators as the LFs of the sentences in a narrative text get interpreted. The events extracted from the LFs are appended to the tree nodes whose positions mirror the structural positions of the corresponding tense operators in the sentence.

Hwang and Schubert's Tense Tree Mechanism

We now consider how Hwang and Schubert's approach analyses example (2.23) which is used by Song and Cohen to argue for the case that both the ET and RT of Reichenbachian tense structure have to be anaphoric. The example is reproduced below together with the LFs which the sentences give rise to and the operators (in italics) acting on them (the *decl* operator arises from the declarative nature of the sentence):

(2.23) a. John went to a hospital.
(*decl* (*past* (John went to hospital)))

b. He had fallen on a patch of ice.
(*decl* (*past* (*perfect* (He fell on a patch of ice))))

c. and had twisted his ankle.
(*decl* (*past* (*perfect* (He twisted his ankle))))

Each LF is interpreted by successively calling the rule associated with the leftmost operator which recursively calls the rule associated with the operator to its right and so on until there are no more operators. At the end of this interpretation process, the tense tree built up for example 2.23 is as in figure 2.17 where the tokens at the nodes of the tree are as follows:

- The tokens e_1, e_2 and e_3 are the events underlying respectively the clauses *John went to a hospital, He fell on a patch of ice* and *He twisted his ankle.*

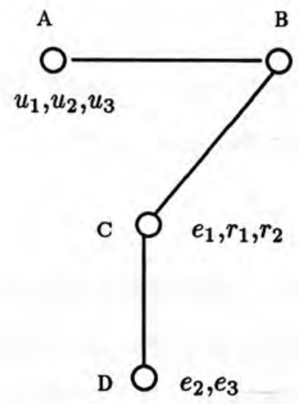


Figure 2.17: The tense tree of example (2.23)

- The tokens u_1, u_2 and u_3 are the utterance events introduced by the *decl* operator.
- The tokens r_1 and r_2 are introduced by the *perfect* operator and can be thought of as the event analogue of the RT in a Reichenbachian tense structure.

Notice from the tense tree that the positions of the tree nodes correspond to the structural positions of the sentential operators:

- Nodes A and B are created by the rule associated with the *decl* operator in (2.23a). The rule also creates event token u_1 and stores it at node A. Rules associated with *decl* operator in (2.23b) and (2.23c) creates respectively event tokens u_2 and u_3 and append them to u_1 at node A.
- Node C is created by the rule associated with the *past* operator in (2.23a) and is positioned to the left of node B. The rule also creates event token e_1 and stores it at node C. As before, rules associated with the PAST operator in (2.23b) and (2.23c) creates respectively event tokens r_1 and r_2 and append them to e_1 at node C.
- Node D is created by the rule associated with the *perfect* operator in (2.23b) and is positioned directly below node C. The rule creates event token e_2 and stores it at node D. As before, the rule associated with the *perfect* operator in (2.23c)

creates an event token e_3 and appends it to e_2 at node D.

From the tense tree of figure 2.17, Hwang and Schubert propose the following interpretation scheme to 'read off' both temporal and non-temporal relations between event tokens in the tense tree:

- An event at a node is oriented by the event to its left (if any). For example, since e_3 (the event of john twisting his ankle) is to the right of e_2 (the event of john falling on ice) at node C in figure 2.17, we have e_3 orients e_2.
- At any given moment, for a pair of event e and f that are rightmost at node n and m respectively, where m is a daughter of n, if the branch connecting the two nodes is a left branch (past), then f before e and if it is a downward branch (perfect), f before-or-until e. For instance, at a given moment in processing, e_2 and r_1 are rightmost at node C and D respectively. Since the branch connecting node C and D is a downward branch, e_2 is before-or-until r_1.

Summary and Discussion

Given that different kinds of relations can exist between episodes, Hwang and Schubert focus on the problem of finding the right pairs of episodes involved in such relations. Like Webber and Song and Cohen, Hwang and Schubert believe that tense contributes to a solution of this problem. However unlike Webber and Song and Cohen, Hwang and Schubert do not invoke Reichenbachian tense structure but instead view tense as operator. This view of tense as a sentential operator is not unique to Hwang and Schubert. It is also adopted in work such as IQ [Richards *et al* 89].

Hwang and Schubert propose a contextual structure called tense tree where the position of the nodes reflects the structural position of the corresponding tense operators in the sentence. From the tense tree representation, both temporal and non-temporal relations can be read off. In both Webber and Song and Cohen's approaches, there are no representations equivalent to the tense tree and situations are generally treated as time points or time intervals so that the only relation between them must be temporal.

Like Song and Cohen, Hwang and Schubert's approach faces the same problem when it comes to embedded discourse in which the tense change does not provide enough information. For example, like Song and Cohen's approach, Hwang and Schubert's approach will also fail on example (2.30) given above and reproduced below together with the LFs of each sentence and the operators acting on them:

(2.30) a. Jack and Sue went to a hardware store to buy a new lawnmower.
(*decl* (*past* (Jack and Sue went to a hardware store)))

b. Their old one had been stolen.
(*decl* (*past* (*perfect* (Their old one was stolen))))

c. Sue saw the men who took it and
(*decl* (*past* (Sue saw the men who took it)))

d. chased them down the street
(*decl* (*past* (Sue chased them down the street)))

In the tense tree of example (2.30), the event token extracted from the LF in (2.30a) (i.e. the going-to-hardware-store event) is stored at the same node as the event token extracted from the LF in (2.30c) (i.e the saw-the-men-who-took-it event). This is because the operators operating on the two LFs are the same *past* operator. The saw-the-men-who-took-it event token (say e_3) is stored to the right of the going-to-hardware-store event token (say e_1) and thus Hwang and Schubert would say that e_3 orients e_1 which is not right.

The common problem with the approaches of Song and Cohen and Hwang and Schubert can be attributed to the fact that they only consider the role of tense in determining the pair of events which should be related. In the following section, we discuss an approach that focuses on the role which world knowledge and information from preceding text play in determining the reference event for a current event.

2.4.4 Eberle

Eberle's work is concerned with the problem of choosing a temporal anchor among the events described in the preceding text for the event described by a new clause [Eberle 92, p 289]. For instance, Eberle asks in the following discourse:

(2.32) a. e_1: Harry handed the book about giraffes to Jane.

b. e_2: Two hours before, he had discovered it in Alfred's bookshop.

c. e_3: He had bought it,

d. e_4: had inserted a dedication and

e. e_5: had had it wrapped up.

f. e_6: Jane took the book with a smile.

how does one decide that the temporal anchor or the reference event for e_3 is not the earlier introduced e_1, but e_2 and that the reference event for e_6 is not the most recent event e_5 but e_1 ?

Note that the approach of Song and Cohen and Hwang and Schubert described above would have no problem determining the correct reference for the various events in (2.32) since the tense change provides enough information. Eberle's approach handles the same problem in a different manner. We will now consider how this approach decides the temporal anchor for e_6 in example (2.32).

At the point of processing clause (2.32f), Eberle's approach would have built up a structure representing the result of processing the preceding clauses. Eberle calls this structure the EVENT STRUCTURE whose nodes are the events described by the clauses in the discourse and the arcs between nodes represent the 'discourse' relations between events.

Events are represented in a variant of Davidsonian representation [Davidson 80]. For instance, the event described by *John came* is represented as

(2.33) $\text{come}(e_1) \wedge \text{agent}(e_1, \text{john})$

where e_1 is the constant assigned to the coming event.

There are 4 discourse relations between events considered in [Eberle 92]:

Continuation For example, in

(2.34) a. e_1: John went into the room.

b. e_2: He sat down.

e_2 is a continuation of e_1. Temporally speaking, e_2 occurs after e_1.

Flashback An example of a flashback relation is in the discourse:

(2.35) a. e_1: John bought an umbrella.

b. e_2: He had lost his old one.

where e_2 is a flashback of e_1. Note that e_2 occurs before e_1.

Elaboration In the following example, e_2 is considered to be an elaboration of e_1 since according to Eberle, e_2 describes e_1 on a more fine-grained level:

(2.36) a. Hans went to Paris.

b. At the border, he had some trouble.

Note that if e_2 elaborates e_1, then e_2 is temporally included in e_1.

Background This is typically a relation between an event and a state. For example, in

(2.37) a. e_1: At the border, he was stopped.

b. s_1: It was pretty dark.

s_1 is said to provide a background for e_1. As widely accepted, the state seems to temporally include e_1 although this may be overridden by world knowledge. Consider, for instance,

(2.38) a. Jameson entered the room, shut the door carefully and switched off the light. It was pitch-dark around him because the venetian blinds were closed.

where the state of being pitch-dark follows the event of switching off the light. Eberle calls this state a causally introduced background, but for the purpose of our discussion here, this distinction with background is not important.

The process of building up the event structure is an incremental one. Given the event structure of the preceding discourse (say ES_n), the process attempts to integrate the situation described by the current clause into ES_n. To carry out this integration, Eberle first determines an ordered set of possible preceding events which can serve as temporal anchors. Then, Eberle attempts to anchor the current situation to the first event in this set. This involves attempting to determine a discourse relation between the current situation and the event. If no such discourse relations can be determined, the next event in the set is tried and so on.

Back to example (2.32) repeated below:

(2.32) a. e_1: Harry handed the book about giraffes to Jane.

b. e_2: Two hours ago, he had discovered it in Alfred's bookshop.

c. e_3: He had bought it,

d. e_4: had inserted a dedication and

e. e_5: had had it wrapped up.

f. e_6: Jane took the book with a smile.

We assume that the first five clauses have been processed and that the event structure built up is as in figure 2.18 where *cont* and *fb* are abbreviations for the continuation and flashback relation. We now consider the integration of e_6 into the event structure of figure 2.18. There are two questions to answer first:

1. The first question is how does Eberle decide on the set of possible temporal anchors ? To do this, he uses a notion of ACCESSIBLE EVENTS which is loosely stated as follows [Eberle 92, p 290]: At the point of attempting to integrate e_5, the accessible events are e_4 and e_1 (with decreasing prominence) whereas at the point of integrating e_4, the accessible events are e_3 and e_1 (again with decreasing

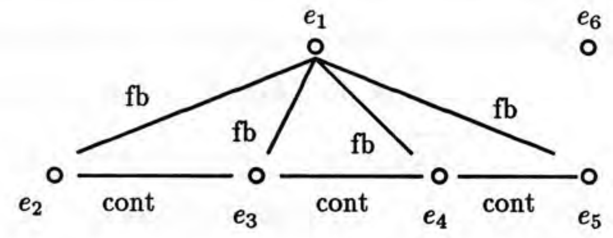


Figure 2.18: The event structure after processing example (2.32a) to (2.32e)

prominence). Based on this imprecise statement and referring to figure 2.18, we assume that Eberle's accessible events are those events at the right frontier of the event structure. Note that states are excluded from consideration by Eberle as possible temporal anchors for the current situation.

2. The second question is how does Eberle determine the discourse relation between the current situation (α) and the current prominent accessible event (β) ? To do this, he uses a sequence of 3 filters to remove those relations that cannot exist between α and β. These 3 filters are as follows:

   (a) The first filter (F_1) uses tense and aspect information. The continuation, elaboration and flashback relations are accepted if α is an event and the tense form of the clause describing α and that describing β are the same. In addition, F_1 accepts the flashback relation if the tense of the clause describing α is past perfect and that describing β is simple past. On the other hand, the background relation is accepted if α is a state and the tense form of the clause describing α and that describing β are the same.

   (b) The second filter (F_2) consists mainly of testing sort subsumptions in the context of temporal incompatibilities between calendar units and event sorts and removes those discourse relations which are logically incompatible with the information of the preceding text. For instance, an event introduced with the location time of 1987 cannot overlap with an event with location time 1988 so that the elaboration relation between the two events will be filtered out.

(c) The third filter (F_3) is an evidence filter which tries to find support for the discourse relations. The filter accepts the relation if some evidence can be found for the relation. Consider, for instance,

(2.39) a. John left Paris in the evening.

b. He went to Frankfurt.

c. At the border, he was stopped.

At the point of processing (2.39c), α is the event described by *He went to Frankfurt* and β is the event described by the current clause *At the border, he was stopped.* After F_1 and F_2, the remaining discourse relations between α and β are **continuation** and **elaboration**. Now, F_3 found supports for **elaboration** since it determines using world knowledge that *border* is on the path of a foreign travel and thus *At the border, he was stopped* could describe *He went to Frankfurt* at a finer level of detail. Therefore, the competing **continuation** relation will be disregarded if no corresponding evidence can be found for it.

Note that if at the end of applying the three filters, there is more than one possible discourse relation remaining, then the most preferred one is taken. In Eberle's approach, continuation is preferred over elaboration which is preferred to flashback.

We are now in a position to discuss the integration of e_6 into the event structure of figure 2.18. The most prominent accessible event is e_5. The tense form of the clause describing the current event e_6 is simple past while that of the clause describing e_5 is past perfect. Filter F_1 rules out all the possible discourse relations between e_6 and e_5. The next prominent accessible event e_1 is tried. This time, a continuation relation is determined between e_1 and e_6. Thus, the event e_1 can anchor e_6 and the event structure after the integration of e_6 is given in figure 2.19.

Since the discourse relations used by Eberle have direct temporal imports (see discussion above), the event structure in figure 2.19 can be taken to be another way of encoding the temporal structure on a time line given in figure 2.20.

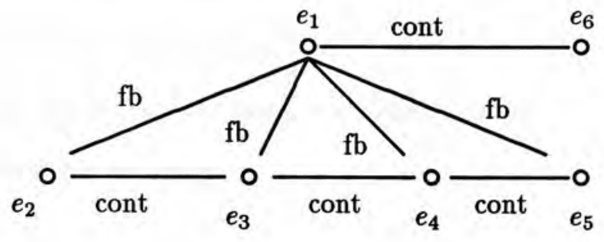


Figure 2.19: The final event structure of example (2.32)

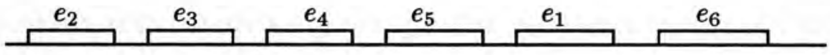


Figure 2.20: The temporal structure equivalent to the event structure in figure 2.19

Summary and Discussion

Eberle presents another approach to solving the integration problem. Unlike Song and Cohen and Hwang and Schubert who use only tense information in their approach, Eberle also uses information from the preceding text to decide on the reference events for the current situation. This information from preceding text is represented as an event structure where the nodes are the events and the arc represents the possible discourse relations used by Eberle between events.

In Eberle's approach, integrating a current situation into the event structure of the preceding text involves first identifying the set of possible accessible events. An attempt is then made to find a possible discourse relation between the current situation and the most prominent accessible event. It is at this stage that tense information is employed. If the attempt fails, the next prominent accessible event is tried and so on.

As noted by Eberle, his approach does not address the problem that it is possible to anchor new events to events that may no longer be accessible. For example, a definite description using event nominalisation can be used to make this anchoring explicit. Consider, for instance

(2.40) a. e_1: James renovated his house.

b. e_2: His friend was envious.

c. e_3: He did not see the hard work James put in

d. before the renovation.

By the time of integrating e_3, the event structure is such that e_2 is a continuation of e_1. This means that e_1 is no longer accessible as a temporal anchor for e_3. This is wrong since e_1 is explicitly indicated by the event nominalisation *the renovation* to anchor e_3.

Another problem with Eberle's approach is that the notion of an accessible event is not clearly defined. This notion seems to depend on the kind of discourse relations that hold between events in the event structure. However, discourse relations are typically talked of as holding between discourse segments. Furthermore, Canepeel and Moens [Caenepeel & Moens 93, p 11] point out that it is important to distinguish between *eventuality relations* that hold between situations in a discourse, and *discourse relations* between the discourse segments.

2.4.5 Lascarides and Asher

Finally, we consider the work of Lascarides and Asher [Lascarides & Asher 91]. The main focus of this work is to show how a formal theory incorporating defeasible rules characterising causal laws and Gricean-style pragmatic maxims can explain the two different temporal interpretations of syntactically similar texts like

(2.41) a. Max stood up. John greeted him.

b. Max fell. John pushed him.

where in the first case, the event described in the first clause precedes the event described in the second clause, while in the second case, the event described in the first clause is preceded by the event described in the second clause.

Lascarides and Asher's approach makes use of a set of discourse relations between clauses. Some of these relations are similar to that used by Eberle although Eberle regards them as relations between events. As in [Eberle 92], Lascarides and Asher use these relations to represent a text as a structure. However, for Lascarides and Asher, the nodes are the semantic contents of the clauses and the arc represents the discourse relation between the clauses. This structure is termed a discourse representation pair (DRP) by Lascarides and Asher.

Lascarides and Asher represent the semantic content of a clause in first order logic. For example, the semantic content of the clause *Max stood up* is represented as

(2.42) $\exists$ e, t [t $<$ *now* $\wedge$ hold(e,t) $\wedge$ standup(max, e)]

where *now* is the speech time and the predicate *hold(e,t)* asserts that the event e holds at the point of time t which is earlier than *now* (due to the simple past).

The DRP is built up in a manner similar to that used for building up the event structure in [Eberle 92]. That is, at the time of integrating the semantic content of the current clause into the DRP of the preceding text, a list of OPEN clauses (more correctly, the semantic content of the clauses) is first identified. A clause α is open if and only if α is the previous clause or the previous clause is subordinate to α. A clause β is subordinate to a clause α if:

- β explains α or β elaborates α, or
- there exists a clause γ such that β explains γ or β elaborates γ and γ is subordinate to α.

This definition of openness can be represented pictorially in figure 2.21. Notice from figure 2.21 that the open clauses are at the right frontier of the structure. Given this description, it appears that the notion of accessible events used in [Eberle 92] is similar to the openness notion of Lascarides and Asher.

After identifying the list of open clauses for the current clause, a non-monotonic logic MASH [Asher & Morreau 91] is used to infer the possible discourse relations holding between the current clause and each of the open clauses. The DRP is then updated with these discourse relations. The notation $\langle\alpha, \beta\rangle$ is used to mean that there is *some*

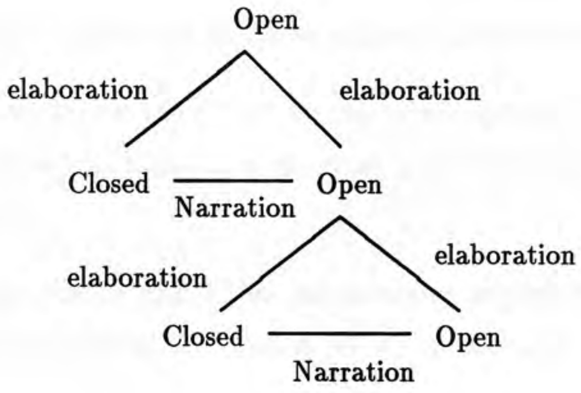


Figure 2.21: Lascarides and Asher's openness notion

discourse relation between the content α of the clause currently being processed and the content β of the clause which is currently open in the DRP representing the text so far. Lascarides and Asher say that α and β are DISCOURSE-RELATED.

We will now illustrate how Lascarides and Asher's approach infers that in example (2.41a) (reproduced below), the relation between the first and second clause is one of narration while that in example (2.41b) is one of explanation:

(2.41) a. Max stood up. John greeted him.

b. Max fell. John pushed him.

The appropriate relevant knowledge needed for example (2.41) is as follows:

(2.43) $\langle \alpha, \beta \rangle > \text{Narration}(\beta, \alpha)$

(2.44) $\langle \alpha, \beta \rangle \wedge \text{fall}(x, \text{me}(\beta)) \wedge \text{push}(x, y, \text{me}(\alpha)) > \text{cause}(\text{me}(\alpha), \text{me}(\beta))$

(2.45) $\langle \alpha, \beta \rangle \wedge \text{cause}(\text{me}(\alpha), \text{me}(\beta)) > \text{Explanation}(\beta, \alpha)$

where the function *me(α)* returns the main eventuality described in α. For example, if α is the semantic content of *Max stood up*, then *me(α)* is the constant assigned to the event of Max standing up. Glossing in English, the three laws state the following:

1. The first law states that α normally forms (indicated by the symbol '$>$') narrative with the open clause β if they are discourse related (indicated by $\langle\alpha, \beta\rangle$).

2. The second law states that α and β that are discourse related where β describes the event e_1 of x falling and α describes the event e_2 of y pushing x are normally such that e_2 caused e_1.

3. The third law states that α and β that are discourse related where the event described in α caused the event described in β are normally such that β explains α.

Consider text (2.41a):

(2.41a) Max stood up. John greeted him.

At the point of integrating the second clause, the only open clause is the first clause in the text. Therefore, α and β are respectively

(2.46) a. $\exists\ t_2, e_2\ [t_2 < \text{now} \wedge \text{hold}(e_2, t_2) \wedge \text{greet}(\text{john}, \text{max}, e_2)]$

b. $\exists\ t_1, e_1\ [t_1 < \text{now} \wedge \text{hold}(e_1, t_1) \wedge \text{standup}(\text{max}, e_1)]$

Lascarides and Asher assume that a current clause is discourse-related to each of its open clauses. Thus, in our current example, $\langle\alpha, \beta\rangle$ is asserted. Therefore, law (2.43) applies and a narration relation is obtained between α and β. From this discourse relation, Lascarides and Asher infer that e_1 occurs before e_2 using the following indefeasible rule:

(2.47) $\text{Narration}(\beta, \alpha) \rightarrow \text{me}(\beta) < \text{me}(\alpha)$

Now, consider text (2.41b):

(2.41b) Max fell. John pushed him.

As before, at the point of integrating the second clause, the only open clause is the first clause in the text. Therefore, α and β are respectively

(2.48) a. $\exists\ t_2, e_2\ [t_2 < \text{now} \wedge \text{hold}(e_2, t_2) \wedge \text{push(john,max}, e_2)]$

b. $\exists\ t_1, e_1\ [t_1 < \text{now} \wedge \text{hold}(e_1, t_1) \wedge \text{fall(max}, e_1)]$

and $\langle\alpha, \beta\rangle$ is asserted. Now, both law (2.43) and law (2.44) apply in this case. But, by the Penguin principle built into MASH that the more specific law overrides the less specific one, the inference that e_2 causes e_1 is derived (law (2.44) is more specific than law (2.43) since the premise of law (2.44) subsumes that of law (2.43)). This, in turn, causes law (2.45) to apply giving the result that α explains β. From this discourse relation, the following indefeasible law is used to infer that e_2 occurs before e_1:

(2.49) $\text{Explanation}(\beta, \alpha) \rightarrow \text{me}(\alpha) < \text{me}(\beta)$

Lascarides and Asher extend the theory to discourse of more than two clauses and in particular to explain the choice of discourse attachment points for clauses in the discourse. For example, in

(2.50) a. Guy experienced a lovely evening last night.

b. He had a fantastic meal.

c. He ate salmon.

d. He devoured lots of cheese.

e. He won a dancing competition.

the theory correctly predicts that clause (2.50e) should not be attached to the previous clause (2.50d). This problem of deciding discourse attachment points for the current clause bears striking similarity to the problem of deciding suitable preceding events for integrating the current situation.

We will now briefly explain how the discourse attachment point for clause (2.50e) is determined. At the point of integrating clause (2.50e), the DRP is as given in figure 2.22. The open clauses are (2.50d), (2.50b) and (2.50a). Since Lascarides and Asher assume that a current clause is discourse-related with each of its open clauses, the following discourse related pairs are asserted into MASH:

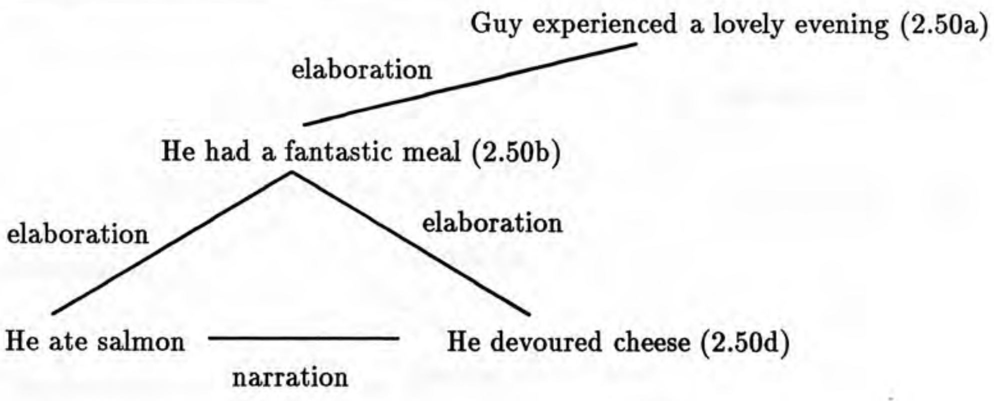

Guy experienced a lovely evening (2.50a)
elaboration
He had a fantastic meal (2.50b)
elaboration
elaboration
He ate salmon
narration
He devoured cheese (2.50d)

Figure 2.22: The DRP after processing example (2.50a) to (2.50d)

1. ⟨2.50e, 2.50d⟩
2. ⟨2.50e, 2.50b⟩
3. ⟨2.50e, 2.50a⟩

From the first discourse related pair, MASH determines from law (2.43) that (2.50e) forms a narration with (2.50d).

However, since from world knowledge that winning a dancing competition (the event described by (2.50e)) cannot be part of having a meal (the event described by (2.50b)), (2.50e) cannot elaborate (2.50b). Since (2.50d) elaborates (2.50b), another defeasible law applies to determine that (2.50e) cannot form a narration with (2.50d).

There is thus conflicting information and no discourse relation can be inferred between (2.50e) and (2.50d). From the remaining two discourse related pairs, MASH infers that (2.50e) forms a narration with (2.50b) and that (2.50e) elaborates (2.50a). The final DRP is as shown in figure 2.23.

The DRP according to Lascarides and Asher represents the discourse structure of the text. Note that the discourse relations used have a direct temporal import (see indefeasible laws (2.47) and (2.49)) so that from figure 2.23, we can extract the temporal structure of the events described in (2.50) in figure 2.24. Note that in addition to indefeasible laws (2.47) and (2.49) which give the temporal imports of the narration and

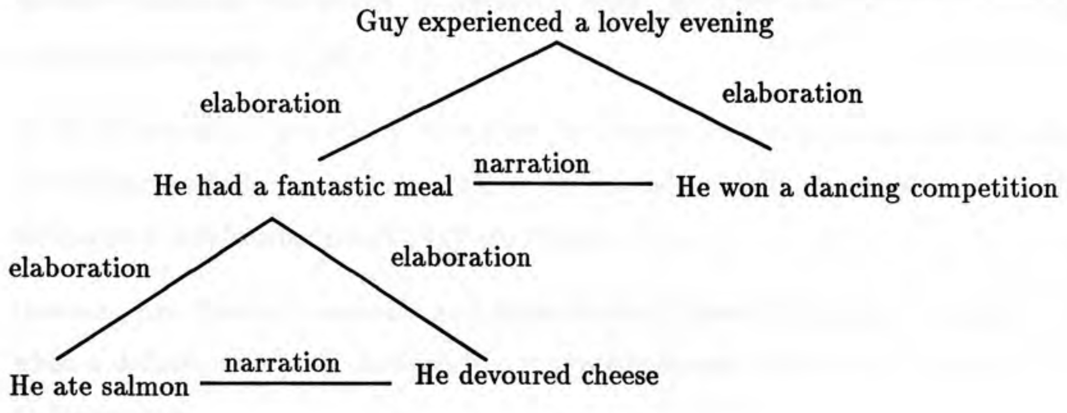


Figure 2.23: The final DRP

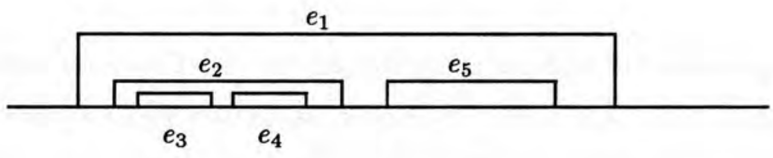


Figure 2.24: The temporal structure of example (2.50)

explanation discourse relation, there is also the law that if clause α elaborates clause β, then the event described in clause α is included in the event described in clause β.

Summary and Discussion

Lascarides and Asher describe a formal approach to understanding the temporal relations in a discourse. Their account employs world knowledge expressed as defeasible and indefeasible rules in a non-monotonic logic MASH to determine the discourse relations between clauses and from this, the temporal relations between the events described by the clauses.

Their theory is also used to determine the discourse attachment points for the current clause. The approach used is very similar to that used to determine the temporal anchor for the current event in [Eberle 92]. For instance, there appears to be a strong similarity between the notion of accessible events in [Eberle 92] with the openness notion in [Lascarides & Asher 91].

As in [Eberle 92], it is possible to extract the temporal structure from the DRP since the discourse relations used have a direct correspondence with temporal relations (see for example indefeasible laws (2.47) and (2.49)).

However, like Eberle, Lascarides and Asher do not address the problem which occurs when a definite NP is used to explicitly attach the current clause to a clause which is no longer open.

2.5 Summary and Conclusion

In this chapter, we have looked at the various approaches to temporal analysis. We began with work on single clauses and proceeded to discuss work on simple linear discourse and finally non-linear discourse. Against this background, we can now identify more specifically the issues to be tackled in this thesis:

- From our survey, it can be noted that most work only considers clauses describing simple situations. These situations are typically represented in a Davidsonian representation. The exception is the work of [Hwang & Schubert 91] which proposes a scheme for representing complex events described by clauses with quantifiers such as *Everyone looked at Mary* (see chapter 3). In our work, we propose an alternative representation for complex events which is able to represent repetitive events and composite events with sub-events.

- A problem in non-linear texts that has been actively investigated is that of integrating a current situation into the representation built up for the preceding text. This problem appears in various guises in the work on non-linear texts surveyed here:
  - In [Webber 87], the problem is phrased as determining the referent for the RT of the current clause.

  - In [Song & Cohen 91b], the problem is phrased as determining the referent for the RT and ET of the current clause.
  - In [Hwang & Schubert 91], the problem is regarded as finding the right pairs of events that are to be related.
  - In [Eberle 92], the problem is one of determining the temporal anchor for the current event from the preceding text.
  - In [Lascarides & Asher 91], the problem of determining discourse attachment point for the current clause bears a striking similarity with the integration problem.

  The solutions proposed in the surveyed work are far from satisfactory. For instance, Song and Cohen and Hwang and Schubert only consider the role of tenses and are unable to handle discourses in which tense change does not provide enough information. Although Eberle considers both information from preceding discourse and tenses, he does not consider the case where a non-accessible event is explicitly indicated to be a temporal anchor for the current event. In addition, previous work employs an intuitive notion of reference whose role is not clearly defined. In our work, we propose an alternative approach in which the role of reference is explicit and which overcomes the limitations of previous work.

- None of the work surveyed handles quantitative temporal information. Such information can definitely contribute to temporal analysis. For instance, if an event e_1 takes a duration of 20 minutes and another event e_2 takes a duration of 10 minutes, then e_1 cannot be included in e_2. In our work, we show how such quantitative temporal information can be handled and integrated with qualitative temporal information.

- Finally, in the work surveyed here, world knowledge is needed for determining the relation between situations. Lascarides and Asher provide a formal account of how such world knowledge can be used. In our work, in addition to world knowledge, we also show how information from the environment can affect temporal analysis. For the instructional domain that we are working on, such information from the environment includes resources information such as the number of

agents available to carry out the actions indicated in the events. We show how such environmental information can give rise to different temporal structures of the same text.

Chapter 3

Representational Issues

In chapter 1, we presented a processing framework of temporal analysis called TA2. As noted in section 1.2.1, the richness and breadth of natural language means that any computational treatment has to narrow its focus. Previous work has focused on narrative texts. In our work, we focus on instructional texts and in particular, on cookery recipes.

The aim of the present chapter is to develop an ontology that encompasses at least the more common of the range of situations and their relations in cookery recipes.

The chapter is organised as follows:

In section 3.1, we discuss some of the basic issues that arise from a consideration of what constitutes an event in a recipe.

In section 3.2, we survey the variety of instructions[1] that occur in recipes. This gives us a better idea of what our ontology, and therefore our representation of the world, has to deal with.

In section 3.3, we introduce an ontology that can be used to represent a considerable subset of the data previously considered. In particular, we show how the mechanism adopted can be used to deal with complex events.

[1]We will use the terms 'instruction' and 'sentence' interchangeably. Recall also that we use the term clause for simple sentences.

In section 3.4, we consider the possible kinds of relations that can exist between situations.

In section 3.5, we discuss the representation of the relations that can hold between time intervals.

Finally, in section 3.6, the representation language is summarised, and section 3.7 ends by indicating some known limitations of the approach taken here.

3.1 Some Ontological Problems in Recipes

In a domain like cookery recipes, ontology is difficult. Our primary concern in this regard is with the representation of situations that can be described in recipes. As noted in [Wilensky 91], situations on the whole have been relatively neglected in computational natural language processing systems and when they are treated at all, the ontology of situations has generally been limited. Before considering this limitation, we first discuss a well-known approach for representing events proposed by Reichenbach [Reichenbach 47] and elaborated by Davidson [Davidson 69, Davidson 80].

In this solution, typically each event is represented by a symbolic constant and information about the event is represented by asserting propositions about the constant. For example, a Davidson-like representation of the sentence *John bought the car from Mary* is as shown in example (3.1), where *john* and *mary* are constants.

(3.1) $\exists\ e, c\ \text{buying}(e) \land \text{buyer}(e, john) \land \text{seller}(e, mary) \land$
$\text{car}(c) \land \text{patient}(e, c)$

Upon skolemising (3.1), we will have a unique constant, say e_1, that corresponds to the buying event. The information about e_1 is asserted via the following predicates: *buyer*, which indicates that John is the buyer in e_1; *seller*, which indicates that Mary is doing the selling; and *patient*, which indicates the object involved in the transaction. Note that there is a finite number of such predicates needed for representing an event (in a theory like the case theory of [Fillmore 68], these predicates are also called cases).

The types of these predicates will depend on the nature of the events.

As pointed out in [Wilensky 91], there are many advantages to be gained from the Davidson-like representation discussed above:

- A unique constant is given to an event which not only serves as a locus of information about the event but also facilitates the representation of subsequent references to the event. For example, assume that the sentence *John bought the car from Mary* represented in (3.1) is followed by the sentence *This upset Jane.* Assuming that the constant assigned to the buying event is e_1, then the *this* can be resolved to this event and the representation of the second sentence is then as in example (3.2). Notice that the agent causing the upsetting is the event, e_1. The *patient* predicate indicates that the person who is being upset is Jane.

  (3.2) $\exists\ e$ upsetting(e) $\wedge$ agent(e, e_1) $\wedge$ patient(e,$jane$)

- Events are treated analogously to objects in that they are both treated as instances of types. For example, in (3.1), the buying event is represented as an instance of buying (*buying(e)*). This is similar to that for the car object which is represented as an instance of car (*car(c)*). One advantage of this can be seen in (3.2) where the buying event, e_1, is taken as the agent of the upsetting event. Another advantage is that the analogous treatment enables the same spatial meaning of a preposition like *in* to be captured irrespective of the argument. For instance, an expression like *the pen in the box* has the representation given in (3.3) where $pen1$ and $box1$ are constants.

  (3.3) in($pen1$,$box1$)

  Now, consider the sentence *The pen broke in the box.* In this case, unlike the NP, *the pen in the box*, we would like to say that it is the event of the pen breaking that occurs in the box. To do this, we can use the same *in* predicate given in (3.3) above:

  (3.4) breaking(e_1) $\wedge$ patient(e_1,$pen1$) $\wedge$ in(e_1,$box1$)

In (3.4), e_1 is the constant assigned to the breaking event and the *patient* predicate indicates that the object which broke is a pen.

- The representation allows a uniform treatment of verbal adjuncts and verbal complements. Typically, verbal adjuncts predicate something about the whole event while verbal complements constitute a component of it. An example will make this clear. Consider example (3.5).

(3.5) John gave a ring to Mary in London

In (3.5), the NPs *John* and *a ring*, and the PP *to Mary* are regarded as verbal complements since the objects they describe are typically considered components of the *giving* event. However, the PP *in London* predicates the location of the *giving* event and is a verbal adjunct. The verbal complements and adjunct in (3.5) are represented similarly as in

(3.6) $\exists\ e, r$ giving(e) $\wedge$ agent(e,$john$) $\wedge$ recipient(e,$mary$) $\wedge$ ring(r) $\wedge$ patient(e,r) $\wedge$ in(e,$london$)

where *john*, *mary* and *london* are constants. Note that the verbal adjunct *in London* is represented by the proposition *in(e,london)* which is similar to that of the three verbal complements, one example of which is *to Mary*, represented by *recipient(e,mary)*.

A uniform treatment of verbal adjuncts and verbal complements in the Davidson-like representation removes a lot of the problems that arise from maintaining a distinction between verbal complements and adjuncts in a representation scheme. For instance, it is now no longer necessary to decide on the number of arguments which verbal predicates should have or whether some arguments are optional or not. Consider, for instance, the verbal predicate corresponding to the verb *throw*. In order to cover data like the following:

(3.7) a. Mary threw the ball onto the roof.

b. Mary threw the ball.

one might say that the predicate corresponding to *throw* takes three arguments: the *agent* (mary in our example), the *patient* (ball in our example) and an optional *onto* (roof in our example). However, this is unnecessary in the Davidson-like representation discussed above.

3.1.1 The Limitations of Simple Events

The Davidson-like representation for events has many advantages as discussed above. However, the representation has its limitation when applied to more complex domains such as cookery recipes.

The Davidson-like representation given for the declarative sentences discussed above can be easily modified to represent simple instructions like

(3.8) Stir the soup.

as

(3.9) stirring(e_1) $\wedge$ object(e_1,x_1) $\wedge$ agent(e_1,hearer)

where e_1, x_1 and hearer are constants assigned to the stirring event, the soup object and the hearer of the instruction. The *object* predicate indicates that the object acted upon in e_1 is x_1 (the soup object) and the *agent* predicate denotes that the person doing the action in e_1 is the hearer of the instruction.

However, we also find more complex instructions than that shown in (3.8). For example, we can have:

(3.10) Stir the soup twice.

In such cases, we can take the view that (3.10) specifies some plural number of the event described by *stir the soup.* For ease of exposition, we will call an event like *stir the soup*, a simple event. A simple event can be thought of as a singular event without any decompositions. We can then represent (3.10) as the following conjunction of expressions[2]:

(3.11) stirring(e_1) $\wedge$ stirring(e_2)

This approach will work as long as we have a constant for each individual event in the collection of individuals described. However, this is not always the case since we can also find examples where the number of repetitions is not specified as in:

(3.12) a. Stir occasionally.

b. Stir every 5 minutes.

In these cases, the events described by these sentences cannot be represented at all if we allow ourselves only simple events since we do not know how many stirring events there are in (3.12). Our formalism must therefore be able to represent such non-simple events.

Other sentences which cause related problems include those with a progressive adjunct such as:

(3.13) Bring the soup to the boil stirring occasionally.

In the above case, it is tempting to say that (3.13) gives rise to two events, viz those described by *bring the soup to the boil* and *stirring occasionally.* However, this gives rise to an ambiguity if we follow (3.13) by the sentence *Meanwhile, chop a carrot* since it is now unclear whether the carrot chopping event is during the event described by *bring the soup to the boil* or the event described by *stir occasionally* or both. A better solution is to regard sentence (3.13) as describing an event with the two sub-events of boiling and stirring occasionally, and assigning a constant, say e_1 to it. Then, the *meanwhile* connective in *Meanwhile, chop a carrot* can be taken as indicating that the carrot chopping event occurs during e_1. Note that e_1 is not a simple event.

[2]We leave out predicates like *object* and *agent* associated with the stirring event.

Outside of recipes, observations have also been made by Hwang and Schubert [Schubert & Hwang 90] on the inadequacy of the Davidson-like representation. In particular, they point out that it is unclear how such a representation can represent sentences involving logical operators or quantifiers such as:

(3.14) Everyone looked at Mary.

although the event described by such a sentence may be referred to subsequently and participates in a causal relation. For example, sentence (3.14) may be followed by:

(3.15) This made Terry furious.

where the *this* refers to the event described by (3.14).

The problem with representing (3.14) is similar to that of representing (3.12) repeated below:

(3.12) Stir occasionally.

As discussed above, the main problem with representing (3.12) is that we do not know the number of stirring events that is being repeated. Similarly, in (3.14), we do not know the number of people looking at Mary.

3.1.2 The Individuation of Events

A related problem to that just discussed is the individuation problem, i.e., the question of deciding exactly how many distinct events are mentioned in a sentence. In our work, events are taken to be actions that are carried out by an agent. Thus, in the context of instructional texts, the individuation problem can be rephrased as that of deciding the number of actions that are to be carried out by an agent. The problem is not an easy one. Consider the following examples:

(3.16) Soak the carrot and the potato.

(3.17) Beat the egg every 5 minutes.

The question to ask in these examples is whether they describe one or more than one event, i.e. actions to be carried out by an agent.

A possible approach is to take the number of events as corresponding to the number of verbs that appeared in the sentence. Thus, under this approach, the instruction in (3.16) describes a single event in which the carrot and potato are soaked in one go. However, there is a problem since whether (3.16) corresponds to one or two events is dependent partly on the external environment in which the instruction is to be executed. In some circumstances, the agent may not want to mix the carrot and potato and soak them in one go. In this case, sentence (3.16) indicates two actions to be carried out by the agents, corresponding to:

(3.18) a. Soak the carrot.

b. Soak the potato.

A similar difficulty occurs in (3.17). Here, it would appear that the sentence is describing more than one event. However, it is not hard to imagine a circumstance in which the agent has access to an egg-beating instrument which can be set to beat every 5 minutes. Under this circumstance, sentence (3.17) really denotes a single event. In this event, the only action required of the agent is to press the switch on the egg-beating instrument.

In our work, we follow Dale [Dale 92] (who follows Hobbs [Hobbs 85]) in adopting the view that language provides the best guide to ontology. Unlike us, [Dale 92] looks at the problem of the individuation of objects, i.e. whether the NP shown in example (3.19) describes one or more objects.

(3.19) one onion, two potatoes and one carrot

Under the view that language provides the best guide to ontology, we take instructions (3.16) and (3.17) as corresponding to an event each, and permit events to have other events as constituents. Thus, we take the sentence *soak the carrot and potato* as a single event and the sentence *beat the egg every 5 minutes* as a single event. However, this does not preclude the possibility that as additional information comes to light (such as the absence or presence of the egg-beating instrument), we might wish to decompose the two events. In the case of *soak the carrot and potato*, we might want to decompose this event into two sub-events: the event of *soak the carrot* and the event of *soak the potato.* Similarly, in a circumstance where there is no egg-beating instrument and the agent has to do the beating, the event of *beat the egg every 5 minutes* can be decomposed into the individual egg-beating sub-events.

3.1.3 Summary

This section has introduced the Davidson-like representation as a common approach for representing events. We highlight the advantages of the representation and also point out its limitations when applied to non-simple events.

In developing our approach to representing the ontology in cookery recipes, we adopt Hobb's view that language provides the best guide to ontology. Before presenting this approach, we first survey the situations that can be described in cookery recipes.

3.2 A Survey of Situations in Recipes

In the previous section, we presented the idea of what we take to be an event within the cookery domain. In this section, we will go on to examine the wide variety of instructions that can be expressed in recipes. Our survey will build on the data analysis work of [Karlin 88] who carried out an empirical study of approximately 110 sentences from nine cookbooks. In particular, we supplement Karlin's work by going through recipes found in the newsgroup, *rec.food.recipes* and two cookbooks.[3]

[3]These cookbooks are *The Electric Casserole Cook Book* by the Good HouseKeeping Institute and the Harvest *Traditional British Cooking.*

3.2.1 Conjoined, Disjoined and Negated Specifications

In this section, we discuss examples of instructions involving conjoined, disjoined or negated specifications.

Conjoined Specifications

If more than one action is to be performed on an object, these are typically listed together as conjoined verbs, as in the following:

(3.20) a. *Soak, drain and rinse* the butterbeans.

b. *Peel and core* the apples.

In (3.20a), the three actions of soaking, draining and rinsing are to be carried out sequentially on the butterbeans. However, in (3.20c), the two actions of peeling and coring can be done in any order on the apples: it is not hard to imagine an agent either coring the apple first and then peeling it or peeling the apple and then coring the peeled apple.

Sometimes, when the same action is to be carried out over more than one object, the object is typically specified using a conjoined NP:

(3.21) a. Chop *the onion and garlic.*

b. Heat *the vegetable garniture and the sauce.*

where in the first case, both the onion and garlic objects are to be chopped, and in the second case, the heating action is carried out on both the sauce and the vegetable garniture.

Finally, it is common to find in recipes instructions with conjoined clauses. These clauses are themselves simple sentences or instructions. Consider for instance,

(3.22) a. Soak the kidneys beans overnight *and* drain.

b. Heat the oil in a frying pan *and* cook the onion for 5 minutes until soft.

where in both cases, two instructions are conjoined together.

Disjoined Specifications

The specification of an instruction can also involve disjunction. Such disjunction allows the agent a choice of actions to select from. Consider example (3.23) which instructs an agent on how to serve the beef stew that has been just cooked:

(3.23) Serve in its casserole, *or* arrange the stew on a platter surrounded with potatoes, noodles and decorated with parsley.

Sometimes, the choice is not in the actions to select but in the objects that are to be acted on by an action. This typically happens when different objects could play similar roles in the recipe. For instance, consider example (3.24)

(3.24) If the casserole seems dry, add *a little water, or more beer.*

where the objects of water and beer play the role of 'moistening' the casserole.

Finally, one can also find instructions like:

(3.25) Simmer *for 10 minutes or until just tender.*

where a disjoined specification is given for the ending point of the simmering action.

Negated Specifications

Finally, a third kind of specification that can be found in recipes is a negated specification; these tell the agent what *not* to do. For example, the instruction in (3.26c) tells the agent not to burn the onion while frying it.

(3.26) a. Heat the butter and half the oil in a large heavy pan.

b. Fry the onion over a medium heat until brown and crisp.

c. Do not allow it to burn.

Such negated specifications can be subsequently referred to as in:

(3.27) a. Do not use too high a heat.

b. This prevents overcooking.

where the *this* in (3.27b) refers to the 'negated event' described in (3.27a).

The representation scheme used in this thesis is adequate for representing conjoined specifications. However, we will not attempt to represent disjunctive or negated specifications. The problems associated with such specifications are discussed further at the end of this chapter.

3.2.2 Constituent Specifications

Instructions can specify actions to be taken at varying levels of detail. For example, clause (3.28b) describes the high-level action of making the icing. The subsequent instructions in (3.28c) further specify the detail on how this high-level action is to be carried out.

(3.28) a. When the cake is cooled,

b. make the icing.

c. Put the chocolate and water into a small saucepan and melt over a gentle heat. Remove from the heat and stir in the butter. When the butter has melted, beat in the icing sugar.

More specifically, the various instructions in (3.28c) describe the sub-events of the event of making the icing.

3.2.3 Repetitive Specifications

There are many ways in which repetition can be specified in recipes. A common way is to specify explicitly that an action is to be repeated a certain number of times either on the same object or different objects of the same type. As noted in [Karlin 88], this can be done via cardinal count adverbials as in example (3.29a), frequency adverbials as in example (3.29b) and (3.29c) and verbal modifiers as in example (3.29d):

(3.29) a. Baste *twice* during the cooking period.

b. Stir the soup *occasionally.*

c. Stir the soup *every 15 minutes.*

d. Beat in the eggs *one at a time.*

In example (3.29a), (3.29b) and (3.29c), the same action is to be repeated over the same object and the number of such repetitions is either explicitly indicated via adverbial like *twice*) or easily computed from adverbials like *occasionally* and *every 15 minutes.* On the other hand, in (3.29c), the beating action is to be repeated over objects of the type egg and the number of such repetitions is given implicitly by the modifier *one at a time.*

Sometimes, the action to be repeated need not be a single action. It can be a whole sequence of actions as in

(3.30) a. Roll the dough out lightly on a floured board, then dab it with small pieces of butter, using a knife blade. Sprinkle with a pinch of flour. Fold the pastry towards you and pinch the edges so that it forms an envelope. Roll as before, rolling away from the joined edge and towards the fold so that the air is not forced out.

b. *Repeat* until all the butter has been used and you have an oblong about 1 cm thick.

where the action to be repeated in (3.30b) is the sequence of actions described in (3.30a).

3.2.4 Time Quantities

It is very common to find time information given in recipes. One such kind of information is the duration, which specifies the interval of time over which an action specified in an event should be carried out. There are many ways in which this can be done. For instance, verbal modifiers may be used to specify an explicit duration by an absolute value (eg (3.31a)), a range (eg (3.31b)), a minimum value (eg (3.31c)), or a fuzzy term like *about* or *around* (eg (3.31d)).

(3.31) a. Steam the fish *for 15 minutes.*

b. Steam the fish *for 10-20 minutes.*

c. Steam the fish *for at least 5 minutes.*

d. Steam the fish *for about 30 minutes.*

Another kind of time information found in recipes gives the extent which separates two events. Consider the following examples:

(3.32) a. 30 minutes before serving, add the wine.

b. 5 minutes before the end of cooking time, add the seasoning.

In (3.32a), the clause, *30 minutes before serving* expressed the information that the start time of the event of adding the wine precedes the start time of the event of *serving* by 30 minutes. However, in (3.32b), the clause, *5 minutes before the end of cooking time*, expressed the information that the start time of the event of adding the seasoning precedes the end time of the cooking event by 5 minutes.

3.2.5 Conditions

Sometimes, the starting and/or ending conditions of an action are specified in the recipes. Typically, these conditions indicate the state something is in before an action can begin or end.

We first consider starting conditions. Typically, these are expressed with *when*-clauses such as

(3.33) *When the cake is cooled*, make the icing.

where the event of making the icing can only begin when the condition indicated by *the cake is cooled* is achieved.

For terminating conditions, an *until*-clause is typically used. Examples of such clauses are as follows:

(3.34) a. Saute over high heat *until moisture is evaporated.*

b. Cook the pork *until golden brown.*

where in the first case, the sauteing event is halted when the moisture is evaporated; while in the second case, the cooking is stopped when the pork has become golden brown.

In certain cases, the terminating conditions are specified as a disjunction involving duration and state change. Consider the following instruction:

(3.35) Cook *the pork for 10 minutes or until golden brown.*

which specifies that the cooking action is to be terminated when the disjunctive condition, *for 10 minutes* or *until the pork becomes golden brown*, is reached. Notice that the disjunction need not be expressed with an explicit *or* as in (3.35).

A question that arises with disjunctive terminating conditions is whether the disjunct is giving the cook a choice. For instance, in (3.35), does the cook have a choice between cooking for 10 minutes or until the pork becomes golden brown? According to Karlin [Karlin 88], the cook does not have such a choice. Instead, the terminating condition is given by the state change, i.e. when the pork has become golden brown, while the explicit duration provides information on the usual amount of time that is needed for the state change to take place. However, for Rock [Rock 92], the explicit duration is taken as contributing to evaluating the terminating condition: if the cooking has been going for more than 10 minutes and the pork is still not golden brown, then something may be wrong.

Our mechanism is able to represent the starting and terminating conditions by regarding them as states. We are, however, not concerned with how disjunctive terminating conditions can be interpreted since they are not crucial to the temporal analysis task. As such, we arbitrarily follow Karlin in interpreting disjunctive terminating conditions.

3.2.6 Purpose Clauses

The purpose of carrying out an action is sometimes given in an instruction. For instance, in

(3.36) Heat on stove *to simmer.*

where the infinitival clause *to simmer* expresses the purpose of the heating action described by the main clause. Observe that carrying out the heating action specified in the main clause results in the simmering of the object being heated.

Purpose clauses can also indirectly constrain the way in which the action specified in the main clause is to be carried out. For instance, in (3.36), the heat supplied must be moderated to ensure that the object being heated is simmering and not boiling. As another example, consider the following from Di Eugenio [Di Eugenio 92]:

(3.37) Cut the dough in half *to create two triangles.*

who points out that there are many ways of cutting a piece of dough in half, eg either vertically, horizontally or diagonally, but the purpose clause *to create two triangles,* constrains the cutting action to cutting diagonally.

In our work, we can handle instructions with purpose clauses by treating them as corresponding to single events with two sub-events: the event described by the main clause and the event described by the purpose clause. The relation between the sub-events is typically one of either GENERATION or ENABLEMENT (see section 3.4).

However, we do not consider the issue of how purpose clauses can contribute to deriving a more specific interpretation of the action described in the main clause. For instance, in (3.37), we do not derive the more specific action of cutting the dough in half diagonally. Such an issue is peripheral to our main concern of temporal analysis where relations between events are more important. For work concerned with the role which purpose clauses play in the interpretation of instructions, see [Di Eugenio 92].

3.2.7 Some Other Observations

There are a number of other phenomena to be found in the specifications of situations. These are discussed here as they are peripheral to our work and the representation mechanism presented here does not address them.

Constraints on Actions

Instructions sometimes express the constraints on the actions to be taken. These constraints express the circumstances under which an action is to be taken and are typically specified using an *if*-clause. Consider, for example,

(3.38) a. *If a slightly softer dip is desired*, add a little extra milk.

b. *If the casserole seems dry*, add a little water.

where the circumstances under which the adding action is to be done in each case is given by the *if*-clause.

Speed

Verbal modifiers can be used to characterise the speed of an action. Some examples are

(3.39) a. Drop the laver cakes into the hot bacon fat and fry *fairly fast* for 2 minutes on each side, shaping and patting the cakes as they fry.

b. Combine the egg and melted butter. Add *slowly* to the flour.

where in the first case, the speed of the frying action is specified by the modifier *fairly fast*, and in the second case, the adding action is to be carried out slowly. Such verbal modifiers can affect temporal analysis by modifying the duration of an action. However, in our work, we ignore such modification.

3.3 Representing Situations

In this and the following sections, we present a representation scheme that is able to represent a wide range, although not all, of the examples discussed in section 3.2.

To begin the discussion, we first review the notion of Generalised Physical Objects introduced by Dale [Dale 92] for representing objects described by NPs in recipes. This is necessary as our representation for situations is analogous to Dale's representation for objects, reflecting the strong parallels that have been noted between the logic of objects and the logic of events [Mayo 61]. Then, we introduce our ontology for situations and the concept of the STRUCTURE of an event and go on to describe the representations used for representing the various information regarding events and states. Finally, we discuss the issue of assigning to each situation a unique time interval.

3.3.1 Dale's Generalised Physical Objects

Dale [Dale 92] describes a scheme for representing objects described by NPs. In particular, he is interested in representing plural NPs such as

(3.40) a. 3 pounds of carrots.

b. 350g mixed raisins, sultanas and currants.

The key idea behind Dale's scheme is that each NP describes a GENERALISED PHYSICAL OBJECT (or PHYSOBJ for short) which is defined [Dale 92, p 111] as follows:

A generalised physical object is any (not necessarily contiguous) collection of contiguous regions of space occupied by matter.

Thus, a physobj consists of one or more 'conventional' physical objects. For example, the physobj corresponding to (3.40a) consists of one or more carrot objects whose total weight is 3 pounds while the physobj described by (3.40b) consists of three objects, namely raisins, sultanas and currants whose total weight is 350g.

In addition to the notion of PHYSOBJ, Dale also proposes that a physobj has an associated STRUCTURE which reflects the PERSPECTIVE from which the physobj is conceived. In particular, a physobj can be viewed either as mass or countable. For example, when cooking, the physobj described by *four ounces of rice* will in all likelihood be viewed as a mass. However, as Dale points out, if I am a scientist examining rice grains for evidence of pesticide use, I may view that same quantity of rice as a countable set of individuals.

We can now show how Dale would represent the physobj described by (3.40a). This representation is given in (3.41). On skolemising (3.41), we will get a unique constant, say x_1, assigned to *three pounds of carrots*. The information predicated of x_1 in (3.41) can be glossed in English as follows:

The physobj is to be viewed as a set (expressed in the *structure* predicate) and the elements of this set (denoted by the variable y) have the following properties: y is to be viewed as a countable individual which is made up of the substance 'carrot-matter' (indicated by the *substance* predicate) and whose shape and size is the default 'carrot-shaped' and 'regular' respectively (represented in the *packaging* predicate).

As Dale is also interested in modelling the change in properties of a physobj, a *holds* predicate is introduced to indicate the proposition which holds in a particular state. For instance, in (3.41), the substance and quantity of a physobj are invariant properties which do not change and thus these are predicated to be true in all states (denoted by the variable s_i). However, the remaining properties can change and may be true only in the current state (denoted by the variable s_j).

$$
\begin{aligned}
(3.41)\quad & \exists\, x\ \forall\, s_i\ \mathbf{holds}(s_i, \mathbf{quantity}(x, \langle \mathbf{3}, \mathbf{pound}\rangle)) \wedge \\
& \quad \exists\, s_j\ \mathbf{current}(s_j) \wedge \mathbf{holds}(s_j, \mathbf{structure}(x, \mathbf{set})) \wedge \\
& \qquad [\forall\, y\ \mathbf{holds}(s_j, \mathbf{element}(y, x)) \supset \\
& \qquad\quad \mathbf{holds}(s_i, \mathbf{substance}(y, \mathbf{carrot\text{-}matter})) \wedge \\
& \qquad\quad \mathbf{holds}(s_j, \mathbf{structure}(y, \mathbf{individual})) \wedge \\
& \qquad\quad \mathbf{holds}(s_j, \mathbf{packaging}(y, \langle \mathbf{carrot}, \mathbf{regular}\rangle))]
\end{aligned}
$$

To complete the discussion, we show the representation of (3.40b), *350g mixed raisins, sultanas and currants.* In Dale's scheme, this object is represented as a set that itself consists of three sets:

$$
\begin{aligned}
(3.42)\quad & \exists\, x\ \forall\, s_i\ \mathbf{holds}(s_i, \mathbf{quantity}(x, \langle \mathbf{350}, \mathbf{g}\rangle)) \wedge \\
& \quad \mathbf{holds}(s_i, \mathbf{constituent}(x, [\ x_1, x_2, x_3\ ]) \wedge \\
& \quad \exists\, s_j\ \mathbf{current}(s_j) \wedge \mathbf{holds}(s_j, \mathbf{structure}(x, \mathbf{set})) \wedge \\
& \qquad [\ \forall\, z\ \mathbf{holds}(s_j, \mathbf{structure}(x_1, \mathbf{set})) \wedge \\
& \qquad\quad \mathbf{holds}(s_j, \mathbf{element}(x_1, z)) \supset \\
& \qquad\quad \mathbf{holds}(s_j, \mathbf{substance}(z, \mathbf{raisin\text{-}matter})) \wedge \\
& \qquad\quad \mathbf{holds}(s_j, \mathbf{structure}(z, \mathbf{individual})) \wedge \\
& \qquad\quad \mathbf{holds}(s_j, \mathbf{packaging}(z, \langle \mathbf{raisin}, \mathbf{regular}\rangle))\ ] \wedge \\
& \qquad [\ \forall\, z\ \mathbf{holds}(s_j, \mathbf{structure}(x_2, \mathbf{set})) \wedge \\
& \qquad\quad \mathbf{holds}(s_j, \mathbf{element}(x_2, z)) \supset \\
& \qquad\quad \mathbf{holds}(s_j, \mathbf{substance}(z, \mathbf{sultana\text{-}matter})) \wedge \\
& \qquad\quad \mathbf{holds}(s_j, \mathbf{structure}(z, \mathbf{individual})) \wedge \\
& \qquad\quad \mathbf{holds}(s_j, \mathbf{packaging}(z, \langle \mathbf{sultana}, \mathbf{regular}\rangle))\ ] \wedge \\
& \qquad [\ \forall\, z\ \mathbf{holds}(s_j, \mathbf{structure}(x_3, \mathbf{set})) \wedge \\
& \qquad\quad \mathbf{holds}(s_j, \mathbf{element}(x_3, z)) \supset \\
& \qquad\quad \mathbf{holds}(s_j, \mathbf{substance}(z, \mathbf{currant\text{-}matter})) \wedge \\
& \qquad\quad \mathbf{holds}(s_j, \mathbf{structure}(z, \mathbf{individual})) \wedge \\
& \qquad\quad \mathbf{holds}(s_j, \mathbf{packaging}(z, \langle \mathbf{currant}, \mathbf{regular}\rangle))\ ]
\end{aligned}
$$

where the predicate *constituent(x, [x_1, x_2, x_3])* indicates the constituents of the set x. As noted in [Dale 92, p 96]: the constituents of the object x are constituents not only in the current state but in all states; this reflects the fact that the constituent objects in question are physical constituents irrespective of whatever other properties are true of the objects.

Note that the substances which objects consist in are organised into a taxonomic hierarchy (see Appendix C for how such a hierarchy can be defined). For instance, the substance potato-matter is a kind of the substance vegetable-matter and we write:

(3.43) is-a(potato-matter, vegetable-matter)

3.3.2 An Ontology for Situations

In our work, we are interested in representing instructions like the following:

(3.44) Stir occasionally.

(3.45) Soak and drain the beans.

We first define a notion of GENERALISED PHYSICAL EVENT (EVENT for short), analogous to its counterpart (PHYSOBJ) in Dale's scheme:

> A generalised physical event is any (not necessarily contiguous) collection of contiguous regions of time occupied by process stuff.

As in the case of physobj, an event may consist of one or more 'conventional' physical events. For example, the event corresponding to (3.44) consists of a number of stirring events and the 'process stuff' which this event consists in is the stirring action. Similarly, the event corresponding to (3.45) consists of two events (namely, the soaking and the draining event) and that the process stuffs which the event consists in are the soaking and draining action.

In our work, events have to be physical since we are dealing with events that have physical existence. In fact, these physical events describe actions to be performed by an agent and the process stuff can be thought of as the action to be carried out by an agent.

However, unlike Dale who uniformly takes each NP as corresponding to a physobj, we do not uniformly take each sentence in the text as corresponding to a single event. Of course, there are sentences which we take as corresponding to an event such as sentences with progressive adjuncts:

(3.13) Bring the soup to the boil stirring occasionally.

and sentences with conjoined clauses:

(3.22a) Soak the kidney beans overnight and drain.

Example (3.13) is taken as describing a single event (see discussion in section 3.1.1) with the two 'conventional' physical events of *bring the soup to the boil* and *stir occasionally.* Likewise, example (3.22a) is taken as describing a single event with the two sub-events of soaking and draining. This decision is motivated by the fact that the single event described by an instruction with conjoined specifications can be subsequently referred to. For example, in

(3.46) a. Move your cursor to the third sentence and type control-K.

b. This will kill the sentence.

the *this* refers to the sequence of events described by the sentence in (3.46a) which contains two conjoined clauses.

However, a sentence with temporal adjuncts as in

(3.47) Steam the fish *until* it is tender.

is not taken as corresponding to a single event but instead gives rise to two situations (i.e. either events or states): the event of steaming the fish and the state of the fish being tender.

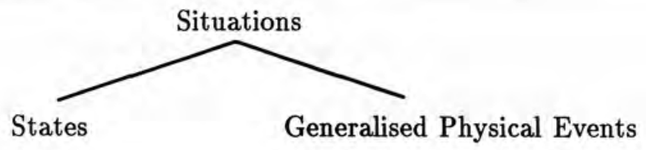


Figure 3.1: Our ontology of situations

Our ontology for situations is shown graphically in figure 3.1.

Based on the discussion above, we can now introduce the following predicates into our representation scheme:

state This is a 1-place predicate which indicates that its argument is a state. Thus, if the constant assigned to the situation described by *the water is cooled* is s_1, then the following means that s_1 is a state:

(3.48) state(s_1)

event Like the *state* predicate, this predicate has arity 1 and indicates that its argument is an event. For example, if the constant assigned to the situation described by *Bring the soup to boil* is e_1, then the following means that e_1 is an event:

(3.49) event(e_1)

In addition to classifying situations into either events or states, we also consider the STRUCTURE of an event. Like its analogue in Dale's scheme, the structure of an event indicates the perspective from which the event is to be conceived. The first question we need to answer is why we need such a notion of structure. Consider the following event described by

(3.50) Cover the pot.

The event in (3.50) appears to be a simple event which cannot be decomposed. However, one might argue that such events can be decomposed if we examine them more closely. For example, the event of covering the pot might be composed of the event of looking for the cover, the event of lifting the cover and the event of lowering the cover onto the pot. Similarly, Allen [Allen 83, p834] argues that there seems to be a strong intuition that, given an event, one can always "turn up the magnification" and look at its structure.

It seems more appropriate to talk about the view from which an event is to be conceived. That is, events are not inherently simple or complex (i.e. decomposable) but are *viewed* as such. Thus, an event can be viewed at one time as simple, and at another time as complex. For example, when cooking, I will typically view the event of covering the pot as a simple event. However, if I am an industrial researcher interested in productivity, I would regard (3.50) as a complex event which can be decomposed into its sub-events so that the timing for each of these sub-events can be taken.

In our domain of cooking, it is unlikely that there is any need to maintain more than one perspective on a given event at any one time. We thus represent the perspective as an objective property of an event. This is done by using the following predicate:

structure This is a 2-place predicate whose first argument is the event and the second argument is the perspective from which the event is to be conceived. In our work, we find it appropriate to consider three perspectives. First, we have SIMPLE events, i.e. events which are viewed as having no decompositions. Then, we have complex events which are viewed as having decompositions: within this category, we distinguish between repetitive events and non-repetitive one. Recall from section 3.2.3 that a repetitive event is one in which some actions are repeated over the same object (eg (3.29b)) or objects of the same kind (eg (3.29c)).

(3.29b) Stir the soup occasionally.

(3.29c) Beat in the eggs one at a time.

We will call complex events which are repetitive, REPETITIONS and those which are not, COMPOSITE events. Now, consider the following examples:

(3.51) a. e_1 : Steam the fish.

b. e_2 : Bring the soup to the boil stirring occasionally.

c. e_3 : Stir occasionally.

(3.51a) describes an event which is to be viewed as a simple event and thus will be represented as

(3.52) structure(e_1, simple)

However, (3.51b) describes an event which is to be viewed as a composite event with two sub-events, namely the event of boiling some soup and the event of stirring occasionally. This is represented as

(3.53) structure(e_2, composite)

Finally, (3.51c) describes an event which is to be viewed as a repetition since the stirring action is repeated; this can then be represented as

(3.54) structure(e_3, repetition)

As noted, an advantage of using the notion of the perspective of an event is that it does not preclude the possibility that as more information is obtained, a simple or repetitive event may be decomposed and be viewed as composite. We can therefore begin with a underspecified representation which can be fleshed out as processing proceeds. This is especially useful in handling the problem of the individuation of events discussed earlier. For instance, we can view the instruction *soak the carrot and potato* as a simple event. However, when more information is obtained, we could decompose this event into two individual sub-events, i.e. the event is now to be viewed as composite.

Finally, each event has a begin and an end state. This is represented using an OCCUR predicate. For example, if the end state of the event described by *steam the fish until it is tender* is s_1 and the state where the event begins is s_0, we write

(3.55) occurs(e_1, s_0, s_1)

where e_1 is the constant assigned to the steaming event. To gloss in English, (3.55) says that the state prior to the execution of the steaming action is s_0 while the state after the execution is s_1.

3.3.3 Representing States

In our work, conditions expressed in instructions (see section 3.2.5) are represented as states. We will now turn to how these can be represented. Typically, a state can be described by a formula of the form:

(3.56) $P_1 \wedge P_2 \ldots P_n$

where each P_i is a proposition that is true in that state. Following Dale [Dale 92], we will use a *holds* predicate to indicate that some propositions hold in a given state. Thus, in the sentence, *steam the fish until it is tender*, the state described by *it is tender* is represented by

(3.57) holds(s_1, tender(x_1, +))

assuming that s_1 is the state described and x_1 is the constant assigned to the physobj described by *the fish*. The representation of x_1 is as in Dale's scheme. Thus, the representation assigned to the sentence *the fish is tender* will be

(3.58) $\exists$ s, x state(s) $\wedge$
holds(s, tender(x, +)) $\wedge$
holds(s, substance(x, fish-matter)) $\wedge$
holds(s, structure(x, individual) $\wedge$
holds(s, packaging (x_1, $\langle$fish, regular$\rangle$))

3.3.4 Representing Simple Events

There are many properties which one can predicate of the constant assigned to a simple event. These properties are as follows:

- Telic
- Action

- Participants

In this section, we will describe these.

Telic

We subdivide simple events into two types: telic and atelic. Telic events are those in which the actions specified in the events have a well-defined culmination point (eg (3.59a)) while atelic events do not have such a point (eg (3.59b)). We can think of the culmination point as the time point at which the event ends and a change in state occurs.

(3.59) a. Bring the soup to the boil.

b. Steam the fish.

(3.59a) describes a telic event since the boiling action has a well-defined culmination point which is observable by an agent. This culmination point is the time point at which the boiling temperature of the soup is reached and bubbling occurs. On the other hand, the steaming action in (3.59b) does not have such a point. In fact, in recipes, we often finds that the steaming action is qualified with an *until*-clause (eg (3.60a)) or a *for*-clause (eg (3.60b)) to indicate the ending point.

(3.60) a. Steam the fish *until tender.*

b. Steam the fish *for 20 minutes.*

At this point, it is appropriate to compare Vendler's classification of situations (see chapter 2) with ours. Recall that Vendler classifies situations into 4 kinds: states, activities, achievements and accomplishments. Vendler's states correspond to our own states while Vendler's activities correspond to our atelic events. We have, however, collapsed Vendler's achievements and accomplishments into our telic events. In Vendler's scheme, achievements (eg (3.61a)) and accomplishments (eg (3.61b)) both have a well-defined culmination point and the only difference between them is that accomplishments are extended in time while achievements are punctual.

(3.61) a. Achievement: John arrived.

b. Accomplishment: John walked to the store.

In our work, we do not distinguish between achievements and accomplishments since we take all events including such apparently punctual events like *John coughs* as being extended, i.e. occurring over a time interval. We leave the discussion of this to section 3.3.6.

To indicate whether an event is telic or atelic, we use a *telic* predicate which takes as its first argument, an event, and its second argument, a value of either + or −. A value of + indicates that the event is telic while a value of − indicates that the event is atelic. For instance, the event described by *Bring the soup to the boil* is telic and would thus be represented as

(3.62) $\text{telic}(e_1, +)$

where e_1 is the constant assigned to the event. On the other hand, the event described by *Steam the fish* is atelic and thus has the following representation

(3.63) $\text{telic}(e_1, -)$

where e_1 is again the constant assigned to the steaming event.

Action

This indicates the action which an agent has to carry out in the event. For example, in the event described by *Bring the soup to the boil*, the action which an agent has to carry out is the bring to boil action. This can be represented as:

(3.64) $\text{action}(e_1, \text{bring-to-boil})$

where e_1 is the constant assigned to the bring-to-boil event.

The various actions are arranged into a taxonomic hierarchy (see Appendix C for how such a hierarchy can be defined). For example, we know that scrubbing action is a kind of (ako) cleaning action:

(3.65) ako(scrubbing, cleaning)

Participants

Each event involves an agent carrying out actions on objects. For example, in *melt the butter*, the agent is to carry out the melting action on the butter object. The agent and the objects which the agent acts on form the participants in the event.

In the cookery domain that we are dealing with, sentences are of the imperative kind and the agent is typically omitted. We therefore assume that the agent is the hearer of the instruction and this can be represented as

(3.66) agent(e_1, hearer)

where e_1 is the event corresponding to the instruction.

In addition to the agent, we also need to represent the role which objects play in the action. As pointed out in section 3.1, the choice of these roles is typically arbitrary and depends on the domain. In this thesis, we will use role names like OBJ, ADDENDUM and BASE. The meanings of these roles are typically self-explanatory. For example, the participants involved in the event of *melt the butter* can be represented as

(3.67) agent(e_1, hearer) $\wedge$ obj(e_1, x_1)

where e_1 is the melting event and x_1 is the butter object. Similarly, the participants involved in the event of *add the cheese to the pasta* can be represented as

(3.68) agent(e_1, hearer) $\wedge$ addendum(e_1, x_1) $\wedge$ base(e_1, x_2)

where e_1 is the adding event, x_1 is the cheese object and x_2 is the pasta object.

We can now show the representation of the event described by *melt the butter*:

(3.69) $\exists\ e, s_0, s_1$ event(e) $\wedge$ structure(e, simple) $\wedge$
telic(e, +) $\wedge$ action(e, melting) $\wedge$
occurs(e, s_0, s_1) $\wedge$ agent(e, hearer) $\wedge$
obj(e, x_1)

where x_1 is the constant assigned to the butter object.

3.3.5 Representing Repetitions and Composite Events

In this section, we discuss the representation of events which are to be perceived as complex, i.e. as having sub-events. We divide such complex events into the repetitive kind (repetitions) and the non-repetitive kind (composite events).

Repetitions

There are two main aspects of a repetition, namely the event to be repeated and the number of such repetitions. We divide our discussion according to how the number of repetitions is specified:

- Specification by Cardinality
- Specification by Frequency
- Implicit Specification

Specification by Cardinality First, consider a case where a cardinal count adverbial is used such as in *stir the soup twice*. Here, the number of repetitions is explicitly indicated via the adverbial, *twice*. The event that is repeated is represented using an *element* predicate while the number of such repetitions in the event is represented with a *cardinality* predicate:

(3.70) $\exists\ e, s_0, s_1$ event(e) $\wedge$ structure(e, repetition) $\wedge$ occurs(e, s_0, s_1) $\wedge$
cardinality(e, 2) $\wedge$

$$[\ \forall\ f\ \text{element}(f,\ e) \supset$$
$$\text{event}(f) \wedge \text{structure}(f,\ \text{simple}) \wedge$$
$$\text{telic}(f,\ -) \wedge \text{action}(f,\ \text{stirring}) \wedge$$
$$\text{agent}(f,\ \text{hearer}) \wedge \text{obj}(f,\ x_1)\ ]$$

where x_1 is the constant assigned to the soup object and s_0 and s_1 are the constants assigned to the state before and after the execution of the instruction respectively.

Specification by Frequency Next, consider the representation of the repetitive event described by *stir the soup occasionally.* Here, the event to be repeated is the stirring event and the frequency of carrying out this repetition is indicated by the frequency adverbial *occasionally.* As before, the event to be repeated is represented via the *element* predicate. The frequency of such repetition in the event is represented with a *frequency* predicate:

(3.71) $\exists\ e,\ s_0,\ s_1\ \text{event}(e) \wedge \text{structure}(e,\ \text{repetition}) \wedge \text{occurs}(e,\ s_0,\ s_1) \wedge$
$\text{frequency}(e,\ \text{occasionally}) \wedge$

$$[\ \forall\ f\ \text{element}(f,\ e) \supset$$
$$\text{event}(f) \wedge \text{structure}(f,\ \text{simple}) \wedge$$
$$\text{telic}(f,\ -) \wedge \text{action}(f,\ \text{stirring}) \wedge$$
$$\text{agent}(f,\ \text{hearer}) \wedge \text{obj}(f,\ x_1)\ ]$$

where x_1, s_0 and s_1 are as described in the previous section.

Implicit Specification Finally, consider a repetitive event in which the number of repetitions is not explicitly given as in *beat the eggs one at a time.* Here, the number of repetitions is implicitly given by the cardinality of the set of eggs which the NP, *the eggs* resolved to. Assuming that the constant assigned to this object is x_1, the representation of the instruction is as follows:

(3.72) $\exists\ e,\ s_0,\ s_1\ \text{event}(e) \wedge \text{structure}(e,\ \text{repetition}) \wedge \text{occurs}(e,\ s_0,\ s_1) \wedge$

$$[\ \forall\ x\ \text{egg}(x) \wedge \text{element}(x,\ x_1) \supset$$

$$\begin{aligned}&\exists f \text{ element}(f, e) \wedge \text{event}(f) \wedge \\ &\quad \text{structure}(f, \text{simple}) \wedge \text{telic}(f, -) \wedge \\ &\quad \text{action}(f, \text{beating}) \wedge \text{agent}(f, \text{hearer}) \wedge \\ &\quad \text{obj}(f, x) \,]\end{aligned}$$

Glossing in English, the above representation essentially says that for every egg that is an element of x_1, the agent is to perform the beating action on it and that each of these actions is an element of the repetition. Note that in (3.72), we used the proposition, *egg(x)* as a gloss for the actual representation using Dale's scheme:

(3.73) $$\begin{aligned}&\text{holds}(s_0, \text{substance}(x, \text{egg-matter})) \wedge \\ &\text{holds}(s_0, \text{structure}(x, \text{individual})) \wedge \\ &\text{holds}(s_0, \text{packaging}(x, \langle\text{egg,regular}\rangle))\end{aligned}$$

Composite Events

Unlike repetitions, composite events are events in which the sub-events are typically explicitly indicated. Consider, for example, the representation of the composite event described by *soak and drain the carrot*:

(3.74) $$\begin{aligned}\exists\, e, f, g, s_0, s_1 \text{ event}(e) \wedge \text{structure}(e, \text{composite}) \wedge \text{occurs}(e, s_0, s_1) \wedge \\ &\text{constituent}(e, [f, g]) \wedge \\ &[\text{ event}(f) \wedge \text{structure}(f, \text{simple}) \wedge \\ &\quad \text{action}(f, \text{soaking}) \wedge \text{telic}(f, +) \wedge \\ &\quad \text{agent}(f, \text{hearer}) \wedge \text{obj}(f, x_1) \,] \wedge \\ &[\text{ event}(g) \wedge \text{structure}(g, \text{simple}) \wedge \\ &\quad \text{action}(g, \text{draining}) \wedge \text{telic}(g, +) \wedge \\ &\quad \text{agent}(g, \text{hearer}) \wedge \text{obj}(g, x_1) \,]\end{aligned}$$

where x_1 is the constant assigned to the physobj described by *the carrot*. Essentially, the representation says that the instruction *soak and drain the carrot* is to be viewed as a composite event with two sub-events, namely the soaking event (f) and the draining event (g). These two sub-events are said to be the CONSTITUENTS of the composite event. Note that no ordering is implied between f and g in the notation [f,g].

Combining Repetitions and Composite Events

It is possible in instructions that a composite event can include repetition as a sub-event, and for a composite event to be repeated. These can all be represented in our representation scheme. Consider the following examples

(3.75) a. Bring the soup to the boil stirring occasionally.

b. Flake the salmon discarding all bones.

(3.75a) describes a composite event with a repetition as a sub-event. This can be represented as

(3.76)
$$\begin{aligned}
\exists\, e, f, g, s_0, s_1\ & \mathrm{event}(e) \wedge \mathrm{structure}(e, \mathrm{composite}) \wedge \mathrm{occurs}(e, s_0, s_1) \wedge \\
& \mathrm{constituent}(e, [f, g])) \wedge \\
& [\ \mathrm{event}(f) \wedge \mathrm{structure}(f, \mathrm{simple}) \wedge \mathrm{telic}(f, +) \wedge \\
& \quad \mathrm{action}(f, \mathrm{boiling}) \wedge \mathrm{agent}(f, \mathrm{hearer}) \wedge \\
& \quad \mathrm{obj}(f, x_1)\ ] \wedge \\
& [\ \mathrm{event}(g) \wedge \mathrm{structure}(g, \mathrm{repetition}) \wedge \\
& \quad \mathrm{frequency}(g, \mathrm{occasionally}) \wedge \\
& \quad [\ \forall h\ \mathrm{element}(h, g) \supset \\
& \qquad \mathrm{event}(h) \wedge \mathrm{structure}(h, \mathrm{simple}) \wedge \\
& \qquad \mathrm{telic}(h, -) \wedge \mathrm{action}(h, \mathrm{stirring}) \wedge \\
& \qquad \mathrm{agent}(h, \mathrm{hearer}) \wedge \mathrm{obj}(h, x_1)\ ]\ ] \wedge \\
& \quad \mathrm{simultaneous}(f, g)
\end{aligned}$$

where f is the event of *bring the soup to the boil* and g is the repetitive event of *stir occasionally.* The relation between f and g is one of simultaneity and this is represented using the *simultaneous* predicate (see section 3.4).

(3.75b) describes a repetitive event in which at each repetition, a piece of salmon flesh is flaked and any bone in this piece discarded. We can represent this as

$$
\begin{array}{ll}
(3.77) & \exists\ e, s_0, s_1\ \text{event}(e) \wedge \text{structure}(e, \text{repetition}) \wedge \text{occurs}(e, s_0, s_1) \wedge \\
& \quad [\ \forall\ x\ \text{flesh}(x) \wedge \text{part-of}(x, x_1) \supset \\
& \qquad \exists\ f, g, h\ \text{element}(f, e) \wedge \text{structure}(f, \text{composite}) \wedge \\
& \qquad \text{constituent}(f, [g, h]) \wedge \\
& \qquad [\ \text{event}(g) \wedge \text{structure}(g, \text{simple}) \wedge \text{telic}(g, +) \wedge \\
& \qquad\ \ \text{action}(g, \text{flaking}) \wedge \text{agent}(g, \text{hearer}) \wedge \\
& \qquad\ \ \text{obj}(g, x)\ ] \wedge \\
& \qquad [\ \text{event}(h) \wedge \text{structure}(h, \text{repetition}) \wedge \\
& \qquad\ \ [\ \forall y\ \text{bone}(y) \wedge \text{part-of}(y, x) \supset \\
& \qquad\quad \exists\ i\ \text{element}(i, h) \wedge \\
& \qquad\qquad \text{structure}(i, \text{simple}) \wedge \text{telic}(i, +) \wedge \\
& \qquad\qquad \text{action}(i, \text{discarding}) \wedge \text{agent}(i, \text{hearer}) \wedge \\
& \qquad\qquad \text{obj}(i, y)\ ]\ ] \wedge \\
& \qquad\ \ \text{simultaneous}(g, h)\ ]
\end{array}
$$

where x_1 is the constant assigned to the physobj described by *the salmon* and flesh(x) and bone(x) are again glosses for the actual representation in Dale's scheme:

$$
\begin{array}{lll}
(3.78) & \text{a.} & \text{holds}(s_0, \text{substance}(x, \text{flesh-matter})) \wedge \\
& & \text{holds}(s_0, \text{structure}(x, \text{mass})) \\
& \text{b.} & \text{holds}(s_0, \text{substance}(x, \text{bone-matter})) \wedge \\
& & \text{holds}(s_0, \text{structure}(x, \text{individual})) \wedge \\
& & \text{holds}(s_0, \text{packaging}(x, \langle \text{bone}, \text{regular} \rangle))
\end{array}
$$

Glossing in English, the above representation says that (3.75b) describes a repetition in which the event to be repeated is the composite event of executing g and h simultaneously: the event g is the flaking of a flesh from the salmon and h is the repetitive event of discarding all the bones from the flesh which has been flaked.

3.3.6 Situations and Time Intervals

In the cookery recipes that we have looked at, there are no instances of instructions expressing instantaneous actions: all actions require some time to be carried out. Outside of cooking, Allen [Allen 83, p834] talks about being able to 'magnify' any events so that they can be regarded as having duration. This is especially true with modern day technology like slow-motion pictures which can be used to magnify apparently punctual events like *John coughs.*

There are, however, objections to such arguments since there exist sentences like *John reached the summit of the mountain* in which there is one moment when John had not yet reached the summit, and another moment when he had, with no time intervening between the two. As Comrie [Comrie 76] points out, no matter how slowly presented the film of John's mountaineering exploits, the interval between these two moments would always be zero. However, one can also argue that the sentence is describing the culmination point of a climbing event. That is, the sentence is not introducing a new event but describing a part of a climbing event. Such a climbing event occurs over a time interval.

Hence, in our work, we regard situations as having duration and occurring over a time interval. To reflect this, we introduce the following predicate:

interval This is a 2-place predicate whose first argument is a situation and the second argument is a time interval associated with the situation. For example, if the constant assigned to the event described by *Bring the soup to the boil* is e_1, then the time interval associated with e_1 is represented as follows:

(3.79) $\text{interval}(e_1, te_1)$

where te_1 is the time interval associated with e_1.

A time interval has two well-defined time points representing the beginning (start time point) and ending (end time point) of the interval. We define two functions that take an interval and return its start and end time points:

start This is a function which takes a time interval and returns the start time point of the interval.

end This is a function which takes a time interval and returns the end time point of the interval.

The unit of time used in our work is taken to be the minutes. Thus, if we have an instruction like *bake the cake for half an hour*, we would convert the hour time unit into the minute time unit, i.e *for half an hour* is converted into *for thirty minutes.* The choice of the minutes as the default time unit is the right grain size for the cooking domain. However, it may not be for other domains. For instance, in the domain of sub-atomic physics, a more appropriate time unit will be of the order of nanoseconds.

We now have the necessary means to represent the information on time quantities discussed in section 3.2.4. The remaining functions and predicates that we need are those from the arithmetic domain, namely, $=, >, <, \leq, \geq, +, -$. The representations of the time information given in (3.31) are as follows:

(3.31) a. Steam the fish *for 15 minutes*

$\text{end}(te_1) - \text{start}(te_1) = 15$

b. Steam the fish *for 10-20 minutes*

$\text{end}(te_1) - \text{start}(te_1) \geq 10 \wedge$

$\text{end}(te_1) - \text{start}(te_1) \leq 20$

c. Steam the fish *for at least 5 minutes*

$\text{end}(te_1) - \text{start}(te_1) \geq 5$

d. Steam the fish *for about 30 minutes*

$\text{end}(te_1) - \text{start}(te_1) \geq 25 \wedge$

$\text{end}(te_1) - \text{start}(te_1) \leq 35$

where te_1 is the time interval associated with the event of steaming the fish. Note that the fuzzy term, *about*, is treated as indicating a range of time. This range is arbitrarily set to be about $\pm$ 5 in our work. For instance, *about 30 minutes* is taken to mean 30 $\pm$ 5.

Finally, we illustrate how we represent the time quantity information given in sentences like

(3.80) a. 30 minutes before serving, add the wine.

b. Just before serving, add the wine.

In (3.80), we assume that the time interval associated with the serving event is te_1 and the time interval associated with the event of adding the wine is te_2. For (3.80a), the information expressed in *30 minutes before* can then be represented as

(3.81) $\text{start}(te_1) - \text{start}(te_2) = 30$

In (3.80b), we take the modifier of the *before*-adjunct, *just*, as an indicator of quantitative information like *30 minutes*. Here, one might argue that it is possible to take the phrase *just before* as indicating that the serving event begins immediately after the add-the-wine event. However, such an approach seems to miss some important generalisations. We would, for instance, need to associate a temporal relation with every structurally similar phrase like *30 minutes before* and *20 minutes before*. A better way is to take a compositional approach and consider the contribution of the semantics of *just* and *before* separately. This is what we do in our work.

More specifically, we take the word *just* as conveying quantitative information, specifying that the difference between the start time of serving and the start time of adding the wine is equal to the duration of adding the wine, i.e.

(3.82) $\text{start}(te_1) - \text{start}(te_2) = \text{end}(te_2) - \text{start}(te_2)$

which can be further simplified to

(3.83) $\text{start}(te_1) = \text{end}(te_2)$

The full representation of the instruction in (3.32a), *30 minutes before serving, add the wine*, is as follows:

(3.84) $\exists\ e, te, f, tf, s_0, s_1, s_2, s_3$

$[\ \text{event}(e) \wedge \text{structure}(e, \text{simple}) \wedge$
$\text{occurs}(e, s_0, s_1) \wedge \text{interval}(e, te) \wedge$
$\text{action}(e, \text{serving}) \wedge \text{telic}(e, +) \wedge$
$\text{agent}(e, \text{hearer}) \wedge \text{obj}(e, x_1)\ ] \wedge$
$[\ \text{event}(f) \wedge \text{structure}(f, \text{simple}) \wedge$
$\text{occurs}(f, s_2, s_3) \wedge \text{interval}(f, tf) \wedge$
$\text{action}(f, \text{adding}) \wedge \text{telic}(f, +) \wedge$
$\text{agent}(f, \text{hearer}) \wedge \text{obj}(f, x_2)\ ] \wedge$
$\text{precede}(f, e) \wedge$
$\text{start}(te) - \text{start}(tf) = 30$

where x_1 and x_2 are the constants assigned to the object that is served and the wine object. Notice that f precedes e as indicated by the *before*-adjunct.

3.3.7 Related Work

As pointed out above, outside of the cooking domain, there has been some work on addressing the problem associated with representing sentences involving logical operators or quantifiers such as

(3.14) Everyone looked at Mary.

One proposal is made by Hwang and Schubert [Hwang & Schubert 91]. In this proposal, a sentence like (3.14) is essentially taken as a proposition that completely describes or characterises a situation e_1. According to Hwang and Schubert, a proposition is said to completely characterise a situation if everything else one can say about the situation is entailed by that proposition. An operator '**' is introduced to connect the proposition to the situation it completely describes:

(3.85) $\exists\ e_1\ (**(\forall\ x\ (\text{Person(x)} \rightarrow \text{Look-at(x, mary)}), e_1))$

Observe that the scheme of Hwang and Schubert essentially allows one to tag the propositions expressed in sentences with situations. Thus, if there are n sentences which gives rise to n propositions, one would need to decide which of the propositions are associated with the same situation identifiers. Given the propositions associated with the same situation, one would then need to decide which of these is characterising and which is entailed. These decisions may not be obvious. For example, Hwang and Schubert consider the situation associated with (3.86) to be characterised by the main clause, *John fired the gun* but not by that corresponding to the subordinate clause, *Looking down the barrel of the gun.*

(3.86) Looking down the barrel of the gun, John fired the gun.

However, as pointed out in Wilensky [Wilensky 91], the same would then apply to the sentence

(3.87) John fired his gun looking down the barrel.

But this sentence could be followed by

(3.88) This enabled him to focus clearly on the target

This sentence is problematic since we have no event characterised by *looking down the barrel* and it is this event that enables John to *focus clearly on the target.*

In addition to the above problem, Hwang and Schubert's proposal also loses some of the advantages associated with a Davidson-like representation discussed in section 3.1. For example, there is no longer an analogous treatment of objects and events and verbal adjuncts and verbal complements are treated separately.

A second proposal is made by Wilensky [Wilensky 91]. Basically, he proposes to include complex events which have sub-events into the event ontology. Then, sentence (3.14) repeated below:

(3.14) Everyone looked at Mary.

can be taken as describing such a complex event and represented as follows:

$$
\begin{aligned}
(3.89)\quad & \exists\, c1\ \text{aio}(c1, \text{complex-event}) \wedge \\
& \quad \forall\, x\ [\ \text{aio(x, person)} \supset \\
& \qquad \exists\, l\ \text{aio}(l, \text{looking-at}) \wedge \\
& \qquad\quad \text{actor}(l, x) \wedge \text{patient}(l, \text{mary}) \wedge \\
& \qquad\quad \text{sub-event}(c1, l)\ ]
\end{aligned}
$$

where the *aio* predicate is used to denote that the first argument is an instance of the second argument. Thus, $c1$ is predicated to be an instance of a complex-event and l is predicated to be an instance of the looking-at event.

Our own representation scheme is closer to Wilensky's suggestion than to Hwang and Schubert's. For instance, our representation extended in the obvious way to sentence (3.14) is:

$$
\begin{aligned}
(3.90)\quad & \exists\, e\ \text{event}(e) \wedge \text{structure}(e, \text{composite}) \wedge \\
& \quad [\ \forall\, x\ \text{person}(x) \supset \\
& \qquad \exists\, f\ \text{element}(f, e) \wedge \\
& \qquad\quad \text{event}(f) \wedge \text{structure}(f, \text{simple}) \wedge \\
& \qquad\quad \text{action}(f, \text{looking-at}) \wedge \\
& \qquad\quad \text{agent}(f, x) \wedge \text{patient}(f, \text{mary})\ ]
\end{aligned}
$$

Unlike Hwang and Schubert, our scheme and Wilensky's preserve the advantages of the Davidson-like representation. However, unlike Wilensky, we do not regard complex events as part of the event ontology. Instead, we see them more as one of the perspectives from which an event can be viewed. As discussed previously, this means that our representation does not preclude the possibility that an event can be decomposed later if additional information merits it. Moreover, we distinguish between complex events that are repetitive and those that are not.

Temporal Relations	Non-Temporal Relations
1. Precedes and After	5. Enables and Enabled-By
2. Includes and Included-In	6. Generates and Generated-By
3. Meets and Met-By	7. Constituent and Constituent-In
4. Simultaneous	8. None

Table 3.1: The set of relations between situations

3.4 The Relations Between Situations

In the previous section, we have seen how situations can be represented. In this section, we discuss the relations that can hold between pairs of situations.

The choice of our relations that we use in our work is motivated by the linguistic data that occurs in recipes. Based on this data, we arrive at the following set of relations (and their inverses) in table 3.1. Note that the **none** relation is a default relation used when we do not know which of the seven relations and their inverses (numbered 1 to 7 in table 3.1) hold between a pair of situations.

We will now discuss these various relations including their representation and the data expressing them. It must be stressed that in many cases, the relations are not directly given in the data but have to be inferred.

3.4.1 The Precedes Relation

Informally, if a situation, sit_1 precedes another situation, sit_2, then sit_1 occurs before sit_2. We will use the notation

(3.91) precedes(sit_1, sit_2)

to indicate that sit_1 occurs before sit_2.

A common way of indicating the precedence relation in recipes is via the use of the *before*-adjunct to connect the clauses describing the two situations. If we let α and β

be the two clauses, then there are two ways of connecting them with a *before*-adjunct, namely either β *before* α or *Before* α, β. Consider the following examples:

(3.92) a. *Before serving*, stir in the soured cream and sprinkle with parsley.

b. Bring the water to boil *before putting in the spaghetti.*

In these examples, the *before*-adjunct serves to indicate that the event described in the main clause precedes the event described in the *before*-clause. For instance, in (3.92a), the complex event of *stir in the soured cream and sprinkle with parsley* precedes the event of *serving*; and in (3.92b), the event of *bring the water to boil* precedes the event of *putting in the spaghetti.*

Another way of indicating the precedence relation is through the use of *then*-adjunct:

(3.93) Mix together the flour and salt in a large mixing bowl and *then rub in the butter.*

In such cases, the event described in the main clause precedes the event described in the *then*-clause.

Finally, the *after*-adjunct can also be used as in

(3.94) *After cooking the meat for 1 hour*, add half the mushrooms and half the dumplings to each pan.

where the event described in the *after*-clause precedes the event described in the main clause.

3.4.2 The Includes Relation

If the action in an event is to be carried out in the course of an action in another event, then the temporal relation between them is one of inclusion. We will use the notation

(3.95) includes(e_1, e_2)

to indicate that the event e_1 **includes** e_2, i.e. e_2 is done during e_1.

There are many ways in which a temporal inclusion relation can be expressed in recipes. One way is to use a *meanwhile*-adjunct as in

(3.96) a. Put the prepared leeks into boiling, salted water and cook for twenty minutes.

b. *Meanwhile, make the cheese sauce.*

where the event described in the *meanwhile*-clause is temporally included in the event described by (3.96a).

Another way is to use the *while*-adjunct as in

(3.97) *While the pastry is cooking*, mix all the vegetables into the sauce and season to taste.

In this case, the event described in the *while*-clause includes the event described in the main clause. Note that the *while*-clause need not be at the front, it can also be at the end of the sentence as in

(3.98) Transfer the poussins to a heated serving dish and keep hot *while making the gravy.*

3.4.3 The Meets Relation

A **meets** relation is said to hold between two situations if one situation happens just after another situation. We specify this with

(3.99) meet(sit_1, sit_2)

where sit_1 **meets** sit_2.

A common way of specifying that one situation meets another is via the use of the *until*-adjunct. This is the case when expressing the culmination point of an event. For example, in

(3.100) Boil the broccoli *until* it is tender.

the boiling event meets the state in which the broccoli is tender.

Another way is to qualify the *before* and *after* adjuncts with modifiers like *just* as shown in (3.80b):

(3.80b) Just before serving, add the wine.

Unlike the case of the *until*-adjunct, the **meets** relation in (3.80b) is the result of the semantics of *just* and *before*. As explained in section 3.3.6 and section 3.4.1, we take *just* as indicating quantitative information and *before* as indicating the temporal relation of **precedes**. These two kinds of information are then combined (in the second step of TA2) to determine that the **meets** relation holds in (3.80b) (see chapter 6). This compositional treatment of *just* and *before* is much better than simply treating *just before* as a phrase that indicates the **meets** relation since it allows important generalisations in the data to be captured.

3.4.4 The Simultaneous Relation

Two situations are said to be simultaneous if they happen during the same time. We specify this with

(3.101) simultaneous(sit_1, sit_2)

to indicate that sit_1 is **simultaneous** with sit_2.

The most common way of expressing simultaneity in recipes is to use progressive adjuncts as in

(3.102) a. Bring the soup to the boil *stirring occasionally.*

b. Line a greased bowl with bread, *cutting and fitting it* to ensure there are no gaps.

where in the first case, the event of stirring occasionally is to be done at the same time as when the soup is brought to a boil, and in the second case, the action of cutting and fitting are to be carried out at the same time as the lining.

3.4.5 The Generation Relation

The generation relation has been extensively studied in the literature [Goldman 70, Pollack 90, Grosz & Sidner 90, Balkanski 90, Di Eugenio 92]. Intuitively, generation is the relation between actions conveyed by the preposition *by* in English, as in *turn on the light by flipping the switch.* More formally, an event e_1 **generates** another event e_2 if and only if when e_1 occurs, a set of conditions C hold such that the joint occurrence of e_1 and C imply the occurrence of e_2. In the case of the event described by *turn on the light by flipping the switch*, e_1 is the event of flipping the switch and e_2 is the event of turning on the light. The conditions that must hold when e_1 is performed include that the wire, the switch and the bulb are working. As another illustration, consider *open the door by pressing the button.* Here, e_1 is the event of pressing the button and together with the conditions that the button and the electrical connection to the door is working will result in the door opening.

An important property of the generation relation is that if it holds of two events, e_1 and e_2, then performing e_1 simultaneously leads to the performance of e_2. We specify the generation relation with

(3.103) $\text{generates}(e_1, e_2)$

to indicate that e_1 **generates** e_2. Now, if e_1 **generates** e_2, then e_1 is **simultaneous** with e_2.

Having defined the generation relation, we now discuss how it is expressed in recipes. In her study on instructions from a how-to-do book on installing wall coverings, and from two craft magazines, Di Eugenio [Di Eugenio 92] finds that the use of *by*-clauses to convey the generation relation is not that common. Instead, purpose clauses (see section 3.2.6) are commonly used. This appears to be true also in cookery recipes where purpose clauses are used to indicate the generation relation as in

(3.104) a. Heat on the stove *to simmer.*

b. Now, steam the chicken. Make a few slits on the chicken *to expose the meat inside.*

where in the first case, the event of heating on the stove simultaneously results in simmering, and in the second case, the event of making a few slits results in the meat being exposed at the same time. One can reexpress the examples in (3.104) using the *by*-clause:

(3.104) a. Simmer by heating on the stove.

b. Now, steam the chicken. Expose the meat inside by making a few slits on the chicken.

3.4.6 The Enablement Relation

Following first Pollack [Pollack 90] and then Balkanski [Balkanski 90], we take the view that the enablement relation holds between two events, e_1 and e_2, if and only if an occurrence of e_1 brings about a set of conditions that are necessary (but not necessarily sufficient) for the subsequent performance of e_2. Unlike the generation relation, both e_1 and e_2 have to be performed by an agent. For instance, [Balkanski 90] gives the following example:

(3.105) Mary inserted a diskette to back up her file.

where the action of inserting a diskette does not simultaneously result in her backing up the file. Instead, the action brings about a condition necessary for the subsequent performance of backing up a file. We will use the notation

(3.106) enables(e_1, e_2)

to denote that e_1 **enables** e_2.

As noted in [Di Eugenio 92], a purpose clause can sometimes, although rarely, be used to express the enablement relation:

(3.107) Unscrew the protective plate to expose the box.

where the event of unscrewing the plate enables the event of taking the plate off which generates the event of exposing the box.

In recipes, it appears that it is equally hard to find instructions involving purpose clauses which express enablement. Instead, the enablement relation is expressed through examples like

(3.108) a. Soak and drain the carrots.

b. Cook some rice. Cool.

where in the first case, the event of soaking brings about a condition of *carrots in water* necessary for the performance of the draining action, and in the second case, the event of cooking brings about a condition of *rice is hot* which is necessary for the cooling action. Notice that in these cases, linguistic cues do not provide much help in determining the relation between events; instead, real-world knowledge is required.

We note that if an event e **enables** another event f, then e **precedes** f since e brings about the conditions necessary for the execution of f.

3.4.7 The Constituency Relation

Recall that a composite event is an event which can be further decomposed into sub-events. In this case, we say that a **constituent** relation holds between an event and each of its sub-events. If the sub-events of a composite event c are $c_1 \ldots c_n$, then the relation between the c_i and c is as follows:

(3.109) constituent(c, c_i) $1 \leq i \leq$ n

If an event e is a constituent of an event f, then f **includes** e.

3.4.8 The None Relation

As noted, this is a default relation used when we do not know yet which of the seven relations and their inverses holds between a pair of situation. Consider, for instance

(3.110) a. e_1: Chop the carrots.

b. e_2: Chop the potatoes.

In this example, no explicit temporal relation is given between e_1 and e_2 via temporal connectives like *meanwhile*. In addition, it is not possible to infer **enablement**, **generation** and **constituency** between the two events.

In such cases, we use the default **none** relation as the relation holding between e_1 and e_2:

(3.111) none(e_1, e_2)

Unlike the other seven non-default relations discussed above, the **none** relation can give rise to different temporal relations in different environments (see section 3.5).

3.5 The Relations Between Time Intervals

In the previous section, we presented the possible relations that can hold between situations. As noted, these relations can be either temporal or non-temporal. In this section, we discuss the relations that can hold between the time intervals associated with situations via the *interval* predicate (see section 3.3.6). Clearly, the relations between time intervals can only be temporal. In our work, we make use of the seven temporal relations and their inverses proposed in [Allen & Kautz 85] to represent the possible temporal relations between time intervals. These relations are given in figure 2.5 (see chapter 2).

At this point, it is appropriate to point out the relationship between the relations of two situations as discussed in the previous section with the temporal relations between their time intervals:

- We first consider the temporal relations between two situations. Typically, the temporal relations between two situations as expressed via linguistic devices like *before*-adjunct are more vague than the temporal relations that can hold between two time intervals. For example, a *before*-adjunct connecting two clauses, α and β, only indicates that the situation described by α must precede that described by β. However, it does not specify whether the time interval assigned to the situation described by α should be $<$ or *meets* the time interval assigned to the situation described by β. That is, a **precedes** relation between two situations is equivalent to the following Allen relations between their time intervals:[4]

  (3.112) sit_1 **precedes** $sit_2 \equiv [\ t_{sit_1} < t_{sit_2}\ ] \vee [\ t_{sit_1}\ m\ t_{sit_2}\ ]$
  $\equiv t_{sit_1}\ \{<, m\}\ t_{sit_2}$

  The corresponding Allen relations for the inverse of the **precedes** relation (i.e. the **after** relation) is:

  (3.113) sit_1 **after** $sit_2 \equiv t_{sit_1}\ \{>, m\}\ t_{sit_2}$

  Similarly, the **includes** relation between two situations is equivalent to the following Allen relations between two time intervals:

  (3.114) sit_1 **includes** $sit_2 \equiv [\ t_{sit_1}\ di\ t_{sit_2}\ ] \vee [\ t_{sit_1}\ si\ t_{sit_2}\ ] \vee [\ t_{sit_1}\ fi\ t_{sit_2}\ ]$
  $\equiv t_{sit_1}\ \{di, si, fi\}\ t_{sit_2}$

  The corresponding Allen relations for the inverse of the **include** relation (i.e. the **included-in** relation) is:

  (3.115) sit_1 **included-in** $sit_2 \equiv t_{sit_1}\ \{d, s, f\}\ t_{sit_2}$

  The two remaining temporal relations (and their inverses if any) between situations correspond directly to their Allen counterparts for time intervals:

  (3.116) sit_1 **meets** $sit_2 \equiv t_{sit_1}\ m\ t_{sit_2} \equiv t_{sit_1}\ \{m\}\ t_{sit_2}$
  sit_1 **met-by** $sit_2 \equiv t_{sit_1}\ \{mi\}\ t_{sit_2}$

[4]For clarity, we used the infix form, i.e sit_1 precedes sit_2 instead of precedes(sit_1, sit_2). Similarly, we use $t_{sit_1} < t_{sit_2}$ instead of $<(t_{sit_1}, t_{sit_2})$ where t_{sit_1} and t_{sit_2} are the time intervals associated with sit_1 and sit_2 respectively.

(3.117) sit_1 **simultaneous** $sit_2 \equiv t_{sit_1} = t_{sit_2}$
$\equiv t_{sit_1} \; \{=\} \; t_{sit_2}$

The above observation on the vagueness of temporal relations indicated by temporal adjuncts is also noted in [Song & Cohen 88]. Such vagueness may be made more precise in language via other means. One example of such a device is quantitative information. For instance, the quantitative information expressed by *30 minutes* and *just* in example (3.80) reproduced below:

(3.80) a. 30 minutes before serving, add the wine.
b. Just before serving, add the wine.

makes precise the intended temporal relation between the serving event and the add-the-wine event. In the first case, the time interval associated with the add-the-wine event is entirely before the time interval associated with the serving event. On the other hand, in the second case, the time interval associated with the add-the-wine event meets (m) the time interval associated with the serving event. This implies that any temporal analysis framework which does not consider quantitative information will necessarily produce a less precise temporal structure than one which does.

- Of the non-temporal relations, the **enablement** and **generation** relation (and their inverses) have a direct mapping into Allen relations:[5]

(3.118) sit_1 **enables** $sit_2 \rightarrow sit_1$ **precedes** sit_2
$\equiv t_{sit_1} \; \{<, m\} \; t_{sit_2}$
sit_1 **enabled-by** $sit_2 \rightarrow sit_1$ **after** sit_2
$\equiv t_{sit_1} \; \{>, mi\} \; t_{sit_2}$

(3.119) sit_1 **generates** $sit_2 \rightarrow sit_1$ **simultaneous** sit_2
$\equiv t_{sit_1} \; \{=\} \; t_{sit_2}$
sit_1 **generated-by** $sit_2 \rightarrow sit_1$ **simultaneous** sit_2
$\equiv t_{sit_1} \; \{=\} \; t_{sit_2}$

[5]Again, we use the infix form i.e sit_1 enables sit_2 instead of enables(sit_1, sit_2).

However, the **constituent** and the default **none** relation does not have a direct correspondence with Allen relations (see chapter 5). In fact, as will be further discussed in chapter 5, the **none** relation can give rise to different Allen relations in different environments. For instance, consider example (3.110) given above and reproduced below:

(3.110) a. e_1: Chop the carrots.

b. e_2: Chop the potatoes.

where there is a **none** relation between e_1 and e_2. In an environment where there is only one agent and one chopping knife, the time interval associated with e_1 and e_2 must be disjoint since both e_1 and e_2 require the agent to bring it to completion. On the other hand, if there are two agents and two chopping knives, then e_1 and e_2 can be carried out simultaneously.

Based on the discussion above, the correspondence of all the relations between situations except the **none** and **constituent** relation and the Allen relations between the time intervals assigned to the situations is given in table 3.2. We will discuss this correspondence further in chapter 5.

3.6 The Representation Language Summarised

This section summarises the details of the representation language presented in the preceding sections.

In our work, we are primarily interested in the representation of situations. A situation can be either a state or generalised physical event (or event for short).

A state is described by the set of conjoined propositions which HOLD in that state.

An event is defined as follows:

> any (not necessarily contiguous) collection of contiguous regions of time occupied by process stuff

sit_1 RELATION sit_2	t_{sit_1} {ALLEN RELATIONS} t_{sit_2}
sit_1 **precedes** sit_2	t_{sit_1} $\{<, m\}$ t_{sit_2}
sit_1 **after** sit_2	t_{sit_1} $\{>, mi\}$ t_{sit_2}
sit_1 **includes** sit_2	t_{sit_1} $\{si, fi, di\}$ t_{sit_2}
sit_1 **included-in** sit_2	t_{sit_1} $\{s, f, d\}$ t_{sit_2}
sit_1 **meets** sit_2	t_{sit_1} $\{m\}$ t_{sit_2}
sit_1 **met-by** sit_2	t_{sit_1} $\{mi\}$ t_{sit_2}
sit_1 **simultaneous** sit_2	t_{sit_1} $\{=\}$ t_{sit_2}
sit_1 **enables** sit_2	t_{sit_1} $\{<, m\}$ t_{sit_2}
sit_1 **enabled-by** sit_2	t_{sit_1} $\{>, mi\}$ t_{sit_2}
sit_1 **generates** sit_2	t_{sit_1} $\{=\}$ t_{sit_2}
sit_1 **generated-by** sit_2	t_{sit_1} $\{=\}$ t_{sit_2}

Table 3.2: The relations between situations and their time intervals

and can be viewed as either a SIMPLE event, a COMPOSITE event or a REPETITION. Simple events are further divided into whether they have a well-defined culmination point or not, i.e. whether they are TELIC or not. In addition to these properties, the following are necessarily true of events:

- Every event OCCURS from a begin state to an end state.
- Every event has an ACTION indicating the action which an agent has to carry out in the event.
- Every event has a finite number of participants. An example of a participant is the AGENT responsible for carrying out the action.

Both events and states are associated with a time INTERVAL over which the situations occur.

The predicates used for representing the various kinds of information discussed above are given in table 3.3. In addition, there are two main functions in the representation

Predicate	Meaning
state(S)	S is a state.
event(E)	E is an event.
structure(E, STRUCT)	The structure of E is STRUCT whose value is either simple, composite or repetition.
element(F, E)	F is an element of E.
constituent(E, [$F_1, \ldots F_n$])	The events $F_1 \ldots F_n$ are the sub-events of event E.
telic(E, TELICITY)	E is telic if TELICITY is + and atelic if TELICITY is −.
occurs(E, S_1, S_2)	E occurs from a begin state, S_1, to end state, S_2
action(E, ACTION)	The action which an agent in E has to carry out
holds(S, PROP)	Proposition PROP holds in state S
interval(SIT, INT)	Situation SIT occurs over interval INT

Table 3.3: The predicates in our representation language

language and these are shown in table 3.4.

We also represent the relations that can hold between situations. These relations can be both temporal or non-temporal and are represented via the predicates given in table 3.5.

Finally, we represent the relations between the time intervals associated with situations. Here, we employ the seven relations and their inverses defined in Allen Interval Algebra

Function	Meaning
start(INT)	This indicates the start time point of interval INT
end(INT)	This indicates the end time point of interval INT

Table 3.4: The functions in our representation language

Predicate	Meaning
precede(SIT_1, SIT_2)	Situation SIT_1 precedes situation SIT_2.
include(SIT_1, SIT_2)	Situation SIT_1 includes situation SIT_2.
simultaneous(SIT_1, SIT_2)	Situation SIT_1 is simultaneous with SIT_2.
meet(SIT_1, SIT_2)	Situation SIT_1 meets situation SIT_2.
enable(E_1, E_2)	Event E_1 enables event E_2.
generate(E_1, E_2)	Event E_1 generates event E_2.
constituent(E_1, F_1)	Event E_1 has constituent F_1
none(SIT_1, SIT_2)	There is no information about the relation between SIT_1 and SIT_2.

Table 3.5: The predicates for the relations between situations

and these are represented via the predicates given in table 3.6. The meanings of these predicates directly correspond to the meanings of their respective relations shown in figure 2.5.

3.7 Conclusions and Limitations

This chapter presented an ontology for representing a wide range of situations that can occur in recipes. Outside of recipes, Wilensky's proposal [Wilensky 91] for representing sentences with logical operators like *Everyone looked at Mary* is compatible with our idea presented here.

We have also presented a set of relations between situations that are motivated by data in cookery recipes. We believe this set is also applicable to other instructional domains.

Finally, we made use of Allen's seven relations and their inverses to represent the relations between time intervals associated with situations. As noted, these temporal relations between time intervals are more exact than the temporal relations between situations that are conveyed by adjuncts like *before* and *meanwhile*. In chapter 6, we

Predicate	Meaning
$<(\text{TSIT}_1, \text{TSIT}_2)$	$\text{TSIT}_1 < \text{TSIT}_2$
$d(\text{TSIT}_1, \text{TSIT}_2)$	$\text{TSIT}_1\ d\ \text{TSIT}_2$
$s(\text{TSIT}_1, \text{TSIT}_2)$	$\text{TSIT}_1\ s\ \text{TSIT}_2$
$f(\text{TSIT}_1, \text{TSIT}_2)$	$\text{TSIT}_1\ f\ \text{TSIT}_2$
$m(\text{TSIT}_1, \text{TSIT}_2)$	$\text{TSIT}_1\ m\ \text{TSIT}_2$
$o(\text{TSIT}_1, \text{TSIT}_2)$	$\text{TSIT}_1\ o\ \text{TSIT}_2$
$=(\text{TSIT}_1, \text{TSIT}_2)$	$\text{TSIT}_1 = \text{TSIT}_2$

Table 3.6: The predicates for Allen relations between time intervals

will show how the relations between time intervals can be deduced from the relations between situations. These relations between time intervals together define the temporal structure of a discourse.

Before concluding this chapter, we end by noting some limitations of our ontology.

3.7.1 Limitations

In this section, we consider some of the potential problems which our approach to ontology might give rise to. In particular, we consider the issue of representing instructions with negated specifications and those with disjunctive specifications.

Negated Specifications

Consider example (3.27) given above and reproduced below:

(3.27) a. Do not use too high a heat.

b. This prevents overcooking.

In (3.27), there appears to be a need for a 'negated event' representing sentence (3.27a) which the diectic *this* in (3.27b) can refer to. An obvious approach in accordance with our idea is to extend the ontology for situations to include negated-event so that a possible representation of (3.27a) will then be

(3.120) $\exists\ e, x$ negated-event(e) $\wedge$ action(e, using)
agent(e, hearer) $\wedge$ obj(e, x) $\wedge$
heat(x) $\wedge$ intensity(x, high)

However, such 'negated events' may seem ontologically suspect as noted in [Hobbs 85, Hirst 89].

Disjunctive Specifications

Instructions with disjunctive specifications pose another problem. Consider example (3.23).

(3.23) Serve in its casserole, or arrange the stew on a platter surrounded with potatoes, noodles and decorated with parsley.

where the disjunction appears to be giving a hearer a choice of which actions to take. If this is the case, then it is appropriate to take (3.23) as giving rise to two events. This is what we do for instructions with temporal adjuncts like

(3.121) Before serving, add the wine.

where the instruction gives rise to two events with the relation between them being the precede relation. However, in the case of disjunction in (3.23), what kind of relations exist between the two events? An obvious answer might be to introduce an *or* relation but again this seems to be ontologically suspect.

Chapter 4

The Quasi-Temporal Structure

In chapter 3, we presented our approach to representing the situations (events or states) mentioned in a text. In this chapter, we consider the issue of how the representations of successive instructions are built up into a Quasi-Temporal Structure (QTS), the first level of representation in our temporal analysis framework, TA2.

Recall from chapter 1 that informally the QTS encodes the representation of the situations described in the text and the relations that hold between pairs of situations.

The QTS is built up by incrementally INTEGRATING the situation described by the current sentence into the QTS of the text preceding the current sentence. At the same time as the QTS is built up, any quantitative information mentioned in the text is extracted and stored.

As noted in chapter 2, many approaches have been proposed recently on how to carry out the integration process [Webber 87, Song & Cohen 91b, Hwang & Schubert 91, Lascarides & Asher 91, Eberle 92]. However, these approaches all have their limitations. The main purpose of this chapter is to describe a new approach to carrying out the integration process that overcomes some of the limitations of previous approaches.

Section 4.1 gives a formal definition of the QTS and describes an approach for building it.

Section 4.2 focuses on the integration problem within the approach for building a QTS. We present our integration approach and compare this with related work.

Our integration approach makes use of information from reference resolution and in section 4.3, we present our reference resolution model. This model incorporates the effects of carrying out simulation of the events described in the text.

Section 4.4 considers the issue of updating the QTS with event decompositions when necessary.

Finally, section 4.5 presents a conclusion and summary.

4.1 Building the Quasi-Temporal Structure

4.1.1 A Formal Definition of the QTS

We now give a formal definition of the QTS which is described informally above. The QTS of a text is defined as a 2-tuple $\langle \text{G}, \text{R} \rangle$:

- $\text{G} = \{sit_i \mid 1 \leq i \leq \text{n} \land sit_i \text{ is either an event or state}\}$
- $\text{R} \subseteq \text{RELNS} \times \text{G} \times \text{G}$ such that $\langle \text{r}, sit_i, sit_j \rangle \in \text{R}$ only if
  $2 \leq i \leq \text{n} \land 1 \leq \text{j} < \text{i}$

where RELNS is the set of possible relations between situations which in the case of cookery recipes is given by the seven relations (**precede, include, simultaneous, meet, enablement, generation** and **constituency**) and their inverses, and the default **none** relation discussed in chapter 3.

The set G is the set of situations (either events or states) mentioned in the text. Note that in this set, we use event subscripts to indicate order of processing, for example, e_2 is processed later than e_1 and e_3 later than e_2 and e_1. The set R indicates the relations that can hold between a situation and the situations processed prior to it. As noted earlier, the set of possible relations is given by RELNS.

Given a QTS, if $\forall\ i,\ j\ \exists\ \langle r_{ij},\ sit_i,\ sit_j \rangle \in$ R, we say that the QTS is COMPLETE. Otherwise, the QTS is said to be REDUCED.

We will now define what is meant for a current situation to be INTEGRATED in a QTS. Recall that in our framework, a situation is either an event or state and that an event can be either composite (with sub-events) or non-composite (i.e. either a simple or repetitive event).

We first define what is meant for a non-composite event or state to be integrated:

<u>Definition 1</u>

A non-composite event or state is INTEGRATED in the QTS if it is related with respect to all the situations processed prior to it in the QTS.

A composite event is made up of sub-events. If each of these sub-events is integrated in the QTS, the composite event will also be. We thus have the following definition:

<u>Definition 2</u>

A composite event is INTEGRATED in the QTS if all its sub-events are integrated in the QTS.

We can now define what is meant for a non-composite event or state to be related with respect to a previous situation:

<u>Definition 3</u>

A non-composite event or state CURR is said to be RELATED WITH RESPECT TO a previous situation PREV if either

1. there is a relation between CURR and PREV or
2. there exists a chain of situations $\text{INTERM}_1 \ldots \text{INTERM}_n$ from CURR to PREV ($\text{CURR} \rightarrow \text{INTERM}_1 \ldots \rightarrow \text{INTERM}_n \rightarrow \text{PREV}$) such that the relation between CURR and PREV can be deduced from the relations between the situations in the chain.

The relation is one of the seven relations and their inverses (if any) and the default **none** relation.

There are a finite number of rules that we use to determine whether the relation between CURR and PREV can be deduced. We will discuss these rules in section 4.2.2.

Finally, we define what is meant for a QTS to be fully-integrated:

<u>Definition 4</u>

A QTS is FULLY-INTEGRATED if all the situations in the QTS are integrated.

Note that the first situation mentioned in the text is trivially integrated.

Before leaving the section, we will illustrate the various points make above with two examples.

Example 1

Consider the following QTS:

(4.1) $\text{G} = \{e_1, e_2, e_3\}$

$\text{R} = \{\langle \text{none}, e_2, e_1\rangle, \langle \text{after}, e_3, e_2\rangle, \langle \text{none}, e_3, e_1\rangle\}$

where the relations between e_2 and e_1 and between e_3 and e_1 are the default **none** relation while e_3 occurs after e_2.

The QTS given in (4.1) is complete. In addition, the QTS is fully-integrated since all the events in the QTS are integrated:

- e_1 is trivially integrated since it is the first event,
- e_2 is integrated since it is related with respect to the previous event e_1 by the **none** relation and
- e_3 is integrated since it is related with respect to the two previous events e_2 and e_1 respectively by the **none** and **after** relation.

Example 2

Consider the following QTS:

(4.2) $\text{G} = \{e_1, e_2, e_3\}$
$\text{R} = \{\langle\text{after}, e_2, e_1\rangle, \langle\text{after}, e_3, e_2\rangle\}$

The QTS given in (4.2) is reduced since the set R does not contain an entry for the relation holding between e_3 and e_1. As in (4.1), the QTS is fully-integrated since all the events in the QTS are integrated:

- e_1 is trivially integrated since it is the first event,
- e_2 is integrated since it is related with respect to the previous event e_1 by the **after** relation,
- e_3 is integrated since it is related with respect to the previous event e_2 by the **after** relation and there exists a chain of events ($e_3 \rightarrow e_2 \rightarrow e_1$) such that the relation between e_3 and e_1 can be deduced using the following rule (see section 4.2.3):

(4.3) If after(e_3, e_2)
And after(e_2, e_1)
Then after(e_3, e_1)

Note that the QTS given in (4.2) can be made 'complete' by adding the **after** relation between e_3 and e_1 into the set R:

(4.4) $\text{G} = \{e_1, e_2, e_3\}$
$\text{R} = \{\langle\text{after}, e_2, e_1\rangle, \langle\text{after}, e_3, e_2\rangle, \langle\text{after}, e_3, e_1\rangle\}$

4.1.2 Our Approach to building the QTS

Recall from chapter 1 that TA2 takes a text as input. Each successive sentence in the text is first processed into some intermediate representation which is close to the surface syntactic structure of the sentence (see chapter 6). This intermediate representation is then further processed and the situations extracted from the intermediate representation integrated into the QTS of the preceding discourse.

The integration process ensures that the current situation is INTEGRATED in the QTS. This process computes a QTS which is reduced and fully-integrated. Leaving details to section 4.2, our approach to integrating the current situation can be outlined as follows:

1. Let PREV-EVENTS be the set of events processed prior to the current situation.
2. Pick an event (REF) from PREV-EVENTS.
3. Determine the relation between REF and the current situation CURR.
4. Remove any event (PREV) in PREV-EVENTS whose relation with CURR can be deduced from the relation between CURR and REF, i.e. there exists a chain CURR $\rightarrow$ REF ... $\rightarrow$ PREV such that the relation between CURR and PREV can be deduced from the relation between the situations in the chain.
5. Repeat step (2) to (4) till PREV-EVENTS is empty.

For ease of exposition, we will call the events that are picked at step (2) of our approach REFERENCE EVENTS. One way of thinking about these reference events for a current situation is that they provide 'anchors' from which the relations between the current situation and the remaining previous events can be determined without going through step (3). This step invokes both linguistic information and real-world knowledge and will be further discussed in section 4.2.4.

Using the above approach, a current situation will get integrated in the QTS since for any previous event PREV, the current situation is either related explicitly to PREV (step (3)) or there exists a chain of situations from the current situation to PREV such that the relation between the current situation and PREV can be deduced (step (4)). Note that by carrying out step (4) in our approach, a current situation may not be explicitly linked to every of its previous situations. In particular, the current situation will not be linked to those previous events whose relations with the current situation can be deduced (see section 4.2.3). This is why the QTS computed by our approach is a reduced one.

Note that, if at step (4) of the approach, we do not remove the set of previous events whose relations with the current situation can be deduced, but instead, add the deduced relation between the previous events and the current situation to the QTS, the QTS obtained will be a complete one. Thus, a reduced QTS computed by our approach can be easily converted into an equivalent complete one. Why then do we compute a reduced QTS instead of a complete one ? There are three reasons for this:

1. A reduced QTS distinguishes between events whose relations with the current situation are determined using a combination of real-world and linguistic knowledge from those events whose relations with the current situation can be deduced using pattern-matching rules such as rule (4.3). The former events are called reference events. Recall from chapter 2 that a notion of reference has been widely adopted in previous work on temporal analysis although such notion is often regarded as intuitive. By clearly defining what is the role of a reference event in our work and using it to compute a reduced QTS, we are in a better position to compare our approach with previous work.

2. A suitable choice of reference events can 'maximise' the number of previous events removed in step (4) of our integration approach and thus 'minimise' the need for step (3). As noted, this step typically involves real-world reasoning which may be computationally expensive.

3. A reduced QTS contains less links than a complete QTS and this implies less computational work to be done when traversing the reduced QTS to obtain the set of possible interval algebra relations between the time intervals associated with the situations (see chapter 5).

An Algorithm

We are now in a position to present the top-level algorithm for constructing a reduced and fully-integrated QTS of a text. This is given in figure 4.1. The algorithm takes as input the set of n sentences in the text ($sent_1 \ldots sent_n$) and returns as output the text's QTS. There are five main kinds of sentences we consider:

- The sentence may be a simple sentence such as the sentence *chop a carrot.*
- The sentence may contain a progressive adjunct as in the sentence *bring the soup to the boil stirring occasionally.*
- The sentence may contain conjoined clauses as in *take out the chicken leg, cut it and add it back to the soup.* Note that sentences with conjoined verbs such as *soak, drain and rinse the beans* are treated as if they are sentences with conjoined clauses. Thus, we treat the sentence *soak, drain and rinse the beans* as equivalent to the sentence *soak the beans, drain them and rinse them.*
- The sentence may have a temporal connective connecting a subordinate clause and a main clause as in the sentence *When the water is boiling, add the meat.*
- The sentence may have a temporal connective connecting a subordinate NP with a main clause as in the sentence *10 minutes before the end of cooking time, put in the seasoning.*

In the algorithm, the BUILD-INTERMEDIATE-REP function takes a sentence and returns its intermediate representation. We will further discuss this function in chapter 6.

The intermediate representation of a sentence is further processed in the INCREMENTAL-BUILD-QTS function which extracts the representations of the situations underlying the sentence and integrate these representations into the preceding QTS. We will first give a gloss of this algorithm:

- If the intermediate representation is that of a simple sentence such as *chop a carrot*, the algorithm will extract the situation underlying it using the BUILD-SITUATION-REP function (see section 4.3.2). The underlying situation is then

```
Algorithm : BUILD-QTS
Input     : $sent_1 \dots sent_n$
Output    : $\text{QTS}_n$

  1. Let $\text{QTS}_0 = (\{\}, \{\})$
  2. For i = 1 to n, do
     Let $\text{INT-REP}_i$ = build-intermediate-rep($sent_i$)
     Let $\text{QTS}_i$ = incremental-build-qts($\text{INT-REP}_i$, $\text{QTS}_{i-1}$)
  3. Return $\text{QTS}_n$
```

Figure 4.1: The top level algorithm for building the QTS of a text

integrated into the preceding QTS via the call to the INTEGRATE function. The function takes three arguments: the situation to be integrated, the preceding QTS and any explicit integration information provided in the sentence (an example of such information is the explicit mention of a reference event in the sentence). This function will be further discussed in section 4.2 when we present our approach to the integration process.

- If the intermediate representation is that of a sentence with a progressive adjunct, such as *bring the soup to the boil stirring occasionally*, the algorithm first extracts the event underlying the main clause and integrates it into the preceding QTS. Subsequently, the algorithm extracts the situation underlying the progressive adjunct clause and integrates it into the preceding QTS. When integrating the situation underlying the adjunct clause, the event underlying the main clause is taken to serve as an explicit reference event and as noted above, this is passed as explicit integration information to the INTEGRATE function. Since we take a sentence with a progressive adjunct as describing a single event (see section 3.1.1), the algorithm will build a new event with the events described by the main and progressive adjunct clauses as constituents (i.e. a composite event). No integration is needed for this composite event since a composite event is integrated if all its sub-events are integrated. The QTS is updated accordingly.

- If the intermediate representation is that of a sentence with conjoined clauses CLAUSE$_1$ ... CLAUSE$_n$, the algorithm successively extracts the situation underlying each of the CLAUSE$_i$ and integrates them into the preceding QTS. Note that when integrating the situation underlying CLAUSE$_i$, we take the situation underlying CLAUSE$_{i-1}$ to serve as an explicit reference event. Since we take a sentence with conjoined clauses as describing a single event (see section 3.1.1), the algorithm will build a new event with the events described by the conjoined clauses as constituents. As in the case of a sentence with progressive adjunct, the QTS is updated accordingly with this new event.

- If the intermediate representation is that of a sentence consisting of a main clause connected to a subordinate clause by a temporal connective, such as *before serving, add the wine*, the algorithm first extracts the situation underlying the subordinate clause. There are two cases to consider:

  - If the subordinate clause is describing a new event, it is integrated into the preceding QTS. On the other hand, if the subordinate clause is describing a state, we then find an event in the preceding QTS which can give rise to this state (via the FIND-ASSOC-EVENT function). Here, we are making the assumption that a state is brought into existence by an event. For example, in

    (4.5) a. Chop the meat.

    b. When the meat is chopped, add the seasoning.

    the *when*-clause is describing a state which is actually the end state of the chopping event described in the first sentence. The state is then integrated into the preceding QTS and the event which gives rise to the state serves as an explicit reference event for the integration.

    The new event or state underlying the subordinate clause is then used as an explicit reference to integrate the situation described by the main clause.

  - If the subordinate clause is describing a previously mentioned event (perhaps from a different aspect), no integration of the event in that clause is needed. For example, in

(4.6) a. e_1: Cool the water.

b. When the water has cooled, e_2: put in the meat.

the *when* clause in the second sentence is describing the end state of the cooling event described in the first sentence. In such cases, the previously mentioned event (e_1 in example (4.6)) and the aspect which the subordinate clause is describing this event are passed as explicit integration information to the INTEGRATE function for integrating the situation described by the main clause (e_2 in example (4.6)).

The main clause is next processed and the situation underlying it is integrated into the preceding QTS.

- If the intermediate representation is that of a sentence consisting of a main clause connected to a subordinate NP of the form *the start of* X or *the end of* X by a temporal connective, such as *10 minutes before the end of cooking, add the seasoning*, the algorithm first processes the subordinate NP. It does this by calling the BUILD-OBJ-REP which builds up the object representation underlying the subordinate NP. The event which this object relates to is then determined (via the FIND-ASSOC-EVENT function) and this event is taken to serve as an explicit reference for integrating the situation underlying the main clause.

Recall that TA2 deals with quantitative information. In the discussion above, we have left out the description of the function in the INCREMENTAL-BUILD-QTS algorithm which processes quantitative information given in a sentence. This function ADD-QUANT stores any quantitative information mentioned in the sentence in the following form:

(4.7) $X - Y \; R \; N$

where X and Y is either start(I) or end(I) (I is a time interval and START and END are functions which return the start and end time point of I (see chapter 3)), R is a relation from the set $\{<, >, \leq, \geq, =\}$ and N is a number. For example, from the sentence *cook the rice for 20 minutes*, we extract and store the following quantitative information where te_1 is the time interval associated with the cook-the-rice event:

(4.8) $\text{end}(te_1) - \text{start}(te_1) = 20$

We postpone the discussion of this function to chapter 5.

Based on the discussion above, we can now present the INCREMENTAL-BUILD-QTS algorithm. This is given in figure 4.2 and continued in figure 4.3 and figure 4.4. To illustrate the algorithm, we will step through four examples using texts containing two sentences.

Example 1

Consider the following example:

(4.9) a. $sent_1$: Cool the water.

b. $sent_2$: When the water has cooled, put in the herbs.

At the beginning, we initialise QTS_0 to

(4.10) $\text{QTS}_0 = (\{\}, \{\})$

Now, $sent_1$ is passed to the INCREMENTAL-BUILD-QTS algorithm. It is a simple sentence and thus step (1) of the algorithm in figure 4.2 is applicable. At step (1a), the BUILD-SITUATION-REP builds up the representation of the event described by $sent_1$ (say e_1). Then, the following call is made:

(4.11) $\text{integrate}(e_1, \text{QTS}_0, \emptyset)$

Leaving details to section 4.2, the INTEGRATE function integrates e_1 into QTS_0 to get QTS_1:

(4.12) $\text{QTS}_1 = (\{e_1\}, \{\})$

```
Algorithm : INCREMENTAL-BUILD-QTS
Input     : INT-REP, $\text{QTS}_p$
Output    : QTS

1. If INT-REP is that of a simple sentence, then
    (a) Let [$curr\text{-}sit$, $quant$] = build-situation-rep(INT-REP)
    (b) add-quant($curr\text{-}sit$, $quant$)
    (c) Let QTS = integrate($curr\text{-}sit$, $\text{QTS}_p$, $\emptyset$)
    (d) Return QTS

2. If INT-REP contains the intermediate representation of the main clause (say MAIN)
   and that of a progressive adjunct clause (say PROG), then
    (a) Let [$sit_{main}$, $quant_{main}$] = build-situation-rep(MAIN)
    (b) add-quant($sit_{main}$, $quant_{main}$)
    (c) Let $\text{QTS}_{main}$ = integrate($sit_{main}$, $\text{QTS}_p$, $\emptyset$)
    (d) Let [$sit_{prog}$, $quant_{prog}$] = build-situation-rep(PROG)
    (e) add-quant($sit_{prog}$, $quant_{prog}$)
    (f) Let $\text{QTS}_{prog}$ = ($\text{G}_{prog}$, $\text{R}_{prog}$)
                  = integrate($sit_{prog}$, $\text{QTS}_{main}$, [$sit_{main}$, progressive])
    (g) Let $sit_{comp}$ = an event whose constituents are $\{sit_{main}, sit_{prog}\}$
    (h) Let QTS = ($\text{G}_{prog} \cup \{sit_{comp}\}$,
                  $\text{R}_{prog} \cup \{\langle$constituent, $sit_{comp}, sit_{main}\rangle, \langle$constituent, $sit_{comp}, sit_{prog}\rangle\}$)
    (i) Return QTS

3. If INT-REP contains the intermediate representation of conjoined clauses (say
   $\text{CLAUSE}_1 \ldots \text{CLAUSE}_n$), then
    (a) Let [$sit_1$, $quant_1$] = build-situation-rep($\text{CLAUSE}_1$)
    (b) add-quant($sit_1$, $quant_1$)
    (c) Let $\text{QTS}_1$ = integrate($sit_1$, $\text{QTS}_p$, $\emptyset$)
    (d) For i = 2 to n, do
        (i) Let [$sit_i$, $quant_i$] = build-situation-rep($\text{CLAUSE}_i$)
        (ii) add-quant($sit_i$, $quant_i$)
        (iii) Let $\text{QTS}_i$ = integrate($sit_i$, $\text{QTS}_{i-1}$, [$sit_{i-1}$, and])
    (e) Let ($\text{G}_n$, $\text{R}_n$) = $\text{QTS}_n$
    (f) Let $sit_{comp}$ = an event whose constituents are $\{sit_1, \ldots sit_n\}$
    (g) Let QTS = ($\text{G}_n \cup \{sit_{comp}\}$,
                  $\text{R}_n \cup \{\langle$constituent, $sit_{comp}, sit_i\rangle \mid 1 \leq i \leq n\}$)
    (h) Return QTS
```

Figure 4.2: The INCREMENTAL-BUILD-QTS algorithm

```
4. If INT-REP contains the intermediate representation of a subordinate clause (say
   SUBORD) connected to that of a main clause (say MAIN) by a temporal connective
   CON, then
   (a) Let $[sit_{subord}, quant_{subord}]$ = build-situation-rep(SUBORD)
   (b) If $sit_{subord}$ is a state or new event, then
       i. add-quant($sit_{subord}$, $quant_{subord}$)
       ii. If $sit_{subord}$ is a state,
           A. Let EVENT = find-assoc-event($sit_{subord}$)
           B. Let $\text{QTS}_{subord}$ = integrate($sit_{subord}$, $\text{QTS}_p$, [EVENT, perfect])
           C. Let INTEG-INFO = [$sit_{subord}$, perfect, CON]
       iii. If $sit_{subord}$ is an event,
           A. Let $\text{QTS}_{subord}$ = integrate($sit_{subord}$, $\text{QTS}_p$, $\emptyset$)
           B. Let INTEG-INFO = [$sit_{subord}$, CON]
       iv. If MAIN is that of a simple sentence, then
           A. Let $[sit_{main}, quant_{main}]$ = build-situation-rep(MAIN)
           B. add-quant($sit_{main}$, $quant_{main}$)
           C. Let QTS = integrate($sit_{main}$, $\text{QTS}_{subord}$, INTEG-INFO)
       v. If MAIN contains the intermediate representation of conjoined clauses
          (say $\text{CLAUSE}_1 \ldots \text{CLAUSE}_n$), then
           A. Let $(G_{int}, R_{int})$ = build-qts($\text{CLAUSE}_1, \ldots \text{CLAUSE}_n$)
           B. Let $sit_{main}$ = an event whose constituents are $G_{int}$
           C. Let $\text{QTS}_{main} = (G_{main}, R_{main})$
                  = integrate($sit_{main}$, $\text{QTS}_{subord}$, INTEG-INFO)
           D. Let QTS = $(G_{main} - \{sit_{main}\} \cup G_{int} \cup \{sit_{main}\}$,
                  $R_{main} \cup R_{int} \cup \{\langle \text{constituent}, sit_{main}, sit_i \rangle, \mid sit_i \in G_{int}\})$
   (c) If $sit_{subord}$ describes another ASPECT of a given event, then
       i. Let $\text{QTS}_{subord} = \text{QTS}_p$
       ii. Let INTEG-INFO = [$sit_{subord}$, ASPECT, CON]
       iii. If MAIN is that of a simple sentence, then goto (4b iv)
       iv. If MAIN contains the intermediate representation of conjoined clauses
           (say $\text{CLAUSE}_1 \ldots \text{CLAUSE}_n$), then goto (4b v)
       v. If CON is a when-connective and ASPECT is neutral perfective, then
          add-quant($sit_{main}$, [$sit_{subord}$, $quant_{subord}$])
   (d) If CON is qualified by a quantitative term QUANT, then
       add-quant($sit_{main}$, [$sit_{subord}$, QUANT, CON])
   (e) Return QTS
```

Figure 4.3: The INCREMENTAL-BUILD-QTS algorithm continued

```
5. If INT-REP contains the intermediate representation of a subordinate NP (say
   SUBORD-NP) the X of Y where X is either START or END connected to that of a
   main clause MAIN by a temporal connective CON qualified by a quantitative term
   QUANT, then
   (a) Let $obj_{subord}$ = build-obj-rep(SUBORD-NP)
   (b) Let $sit_{subord}$ = find-assoc-event($obj_{subord}$)
   (c) Let $QTS_{subord} = QTS_p$
   (d) Let INTEG-INFO = $[sit_{subord}, \emptyset]$
   (e) If MAIN is that of a simple sentence, then
           Goto (4b iv)
   (f) If MAIN contains the intermediate representation of the conjoined clauses
       $CLAUSE_1 \ldots CLAUSE_n$, then
           Goto (4b v)
   (g) add-quant($sit_{main}$, [[$sit_{subord}$,X],QUANT,CON])
   (h) Goto (4e)
```

Figure 4.4: The INCREMENTAL-BUILD-QTS algorithm continued

We now process $sent_2$. This is a sentence with a *when* connective. Thus, step (4) of the algorithm in figure 4.3 is applicable. At step (4a), the BUILD-SITUATION-REP is called to build the representation of the situation underlying the subordinate clause *the water has cooled.* It finds that the subordinate clause is describing the end state of e_1, the event described by the first sentence (see section 4.3). When this happens, we say that the subordinate clause is describing another aspect of a GIVEN event. Therefore, step (4c) is applicable. At step (ii) of (4c), we record the following information which is used when integrating the event described by the main clause:

(4.13) $[e_1$, perfect, when$]$

where *perfect* indicates that the subordinate clause is describing the end state of a previously mentioned event e_1 (see section 4.3).

Since the main clause *put in the herbs* is a simple sentence, we continue with step (iii) of (4c). Assuming that the BUILD-SITUATION-REP returns e_2 for the main clause, the following call is made:

(4.14) integrate(e_2, QTS$_1$, [e_1, perfect, when])

which will integrates e_2 into QTS$_1$. An explicit reference event for e_2 is given. This reference is e_1 and e_2 is determined to occur after e_1 (see section 4.2.4). The updated QTS (QTS$_2$) is now:

(4.15) QTS$_2$ = $(\{e_1, e_2\}, \{\langle \text{after}, e_2, e_1\rangle\})$

This is then the QTS of example (4.9).

Example 2

We next consider the following example:

(4.16) a. *sent*$_1$: Bring the soup to the boil.

b. *sent*$_2$: When the soup is boiling, add the meat.

The initial processing is similar to that of example 1 above and we assume that QTS$_1$ at the end of processing *sent*$_1$ is

(4.17) QTS$_1$ = $(\{e_1\}, \{\})$

where e_1 is the event described by *sent*$_1$. We now process *sent*$_2$ which is sent to the INCREMENTAL-BUILD-QTS algorithm. Since this sentence has a *when*-connective, step (4) is applicable. The subordinate clause *the soup is boiling* is determined to describe a state s_1 (see section 4.3). Thus, step (4b) is applicable. We continue with step (ii) of (4b) since s_1 is a state and attempt to find an event that gives rise to s_1. Such an event is found in e_1 since the state of the soup boiling is actually the end state of the event of bringing the soup to the boil. The following call is then made:

(4.18) integrate(s_1, QTS$_1$, [e_1, perfect])

which integrates s_1 into QTS$_1$ by making e_1 a reference event for s_1 and determining that s_1 meets e_1. The updated QTS is QTS$_{subord}$:

(4.19) $\text{QTS}_{subord} = (\{e_1, s_1\}, \{\langle \text{meet}, s_1, e_1\rangle\})$

In addition, we also record the following information which is to be used in integrating the event described by the main clause:

(4.20) $[s_1, \text{perfect}, \text{when}]$

Now, the main clause *add the meat* is simple and so we carry out step (iv) of (4b). Supposing that the event underlying this clause is e_2, we then make the following call:

(4.21) $\text{integrate}(e_2, \text{QTS}_{subord}, [s_1, \text{perfect}, \text{when}])$

to integrate e_2 into QTS_{subord}. Now, e_2 is explicitly indicated to take reference from s_1 and that it is included in s_1. The updated QTS is QTS_2:

(4.22) $\text{QTS}_2 = (\{e_1, s_1, e_2\}, \{\langle \text{meet}, s_1, e_1\rangle, \langle \text{included-in}, e_2, s_1\rangle\})$

which is the QTS of example (4.16).

Example 3

Here, we consider the following example:

(4.23) a. *sent*$_1$: Mash the potato.

b. *sent*$_2$: Before you mash it, take out the chicken leg, cut it and add it back.

The initial processing is similar to that of example 1 and example 2 and we again assume that QTS_1 at the end of processing *sent*$_1$ is

(4.24) $\text{QTS}_1 = (\{e_1\}, \{\})$

where e_1 is the event described by *sent*$_1$. We now process *sent*$_2$ which is sent to the INCREMENTAL-BUILD-QTS algorithm. Since this sentence has a *before*-connective, step

(4) is applicable. The subordinate clause *you mash it* is determined to describe e_1. We therefore carry out step (4c) and at step (ii) of (4c), we record the following information which will be used when integrating the event described by the main clause:

(4.25) $[e_1, \text{identity, before}]$

where *identity* indicates that the subordinate clause is describing an event which is identical with an existing event.

Since the main clause *take out the chicken leg, cut it and add it back* contains conjoined clauses, step (iv) of (4c) is applicable. A recursive call is made to the BUILD-QTS algorithm to build up the quasi-temporal structure (QTS_{int}) of the set of conjoined clauses. These clauses are all simple sentences and repeatedly applying step (1) of figure 4.2, we have the following QTS_{int} where e_2, e_3 and e_4 are the events described by *take out the chicken leg*, *cut it* and *add it back* respectively.

(4.26)
$$\begin{aligned} \text{G}_{int} &= \{e_2, e_3, e_4\} \\ \text{R}_{int} &= \{\langle \text{after}, e_3, e_2\rangle, \langle \text{after}, e_4, e_3\rangle\} \end{aligned}$$

We now create a composite event e_5 whose constituents are the elements of the set G_{int}. The following call is then made to integrate e_5 into QTS_1:

(4.27) $\text{integrate}(e_5, \text{QTS}_1, [e_1, \text{identity, before}])$

In this integration, e_5 is explicitly indicated to take reference from e_1 and that e_5 precedes e_1. The updated QTS (QTS_{main}) is:

(4.28) $\text{QTS}_{main} = (\{e_1, e_5\}, \{\langle \text{precede}, e_5, e_1\rangle\})$

Finally, at step D of (4b v), we return QTS_2:

(4.29)
$$\begin{aligned} \text{QTS}_2 = (&\{e_1, e_2, e_3, e_4, e_5\}, \\ &\{\langle \text{precedes}, e_5, e_1\rangle, \langle \text{after}, e_3, e_2\rangle, \\ &\langle \text{after}, e_4, e_3\rangle, \langle \text{constituent}, e_5, e_2\rangle, \\ &\langle \text{constituent}, e_5, e_3\rangle, \langle \text{constituent}, e_5, e_4\rangle\}) \end{aligned}$$

which is the QTS of example (4.23).

Example 4

Finally, we consider the following example:

(4.30) a. $sent_1$: Cook the rice for 30 minutes.

b. $sent_2$: 10 minutes before the end of cooking time, put in the seasoning.

The first sentence $sent_1$ is simple and as in the previous three examples the first step of the algorithm in figure 4.1 applies. However, unlike the previous examples, the presence of the *for*-adverbial in $sent_1$ is analysed as indicating a duration. As noted, this quantitative information is stored separately from the QTS and in the following form:

(4.31) $\text{end}(te_1) - \text{start}(te_1) = 30$

where te_1 is the time interval associated with the cook-the-rice event e_1. After integrating e_1, the QTS (say QTS$_1$) is as follows:

(4.32) $\text{QTS}_1 = (\{e_1\}, \{\})$

We now process $sent_2$. This sentence contains a subordinate NP of the form *the end of cooking time*. Therefore, step (5) of the algorithm in figure 4.4 is applicable. We build up the representation of the end time point entity (say x_1) described by the NP. At step (5b), we determine that this entity is semantically related to e_1 and at step (5d), we record the following integration information to be used when integrating the event described by the main clause:

(4.33) $[e_1, \emptyset]$

Since the main clause is simple, we proceed to step (5e) which redirects us to step (iv) of (4b). Assuming that the event described by the main clause is e_2, the following call is made to integrate e_2 into QTS$_1$:

(4.34) integrate(e_2, QTS$_1$, [e_1, $\emptyset$])

In this integration, e_2 is explicitly indicated to take reference from e_1. However, no relation is explicitly given between e_2 and e_1. This is because in our current example, only quantitative information between e_2 and e_1 is given. We therefore need to determine the qualitative relation between e_2 and e_1. For our current example, none of the seven relations exists between e_2 and e_1 and thus the default **none** relation is indicated as holding between e_2 and e_1.

Before returning the QTS at step (5h), we store the following quantitative information explicitly given in the sentence:

(4.35) add-quant(e_2, [[e_1,end], 10, before])

This gives rise to the following equation:

(4.36) $\text{end}(te_1) - \text{start}(te_2) = 10$

The QTS returned is as follows:

(4.37) $(\{e_1, e_2\}, \{\langle \text{none}, e_2, e_1\rangle\})$

4.2 The Integration Problem

In this section, we present our approach to the integration problem needed in the INTEGRATE function in the INCREMENTAL-BUILD-QTS algorithm. As noted, the integration problem has received much attention recently [Webber 87, Song & Cohen 91b, Hwang & Schubert 91, Lascarides & Asher 91, Eberle 92].

As noted earlier, our approach computes a reduced and fully-integrated QTS and the steps involved in this approach are as follows:

1. Let PREV-EVENTS be the set of events processed prior to the current situation.

2. Pick an event (REF) from PREV-EVENTS.

3. Determine the relation between REF and the current situation CURR.

4. Remove any event (PREV) in PREV-EVENTS whose relation with CURR can be deduced from the relation between CURR and REF, i.e. there exists a chain CURR → REF ... → PREV such that the relation between CURR and PREV can be deduced from the relation between the situations in the chain.

5. Repeat step (2) to (4) till PREV-EVENTS is empty.

Recall that we call the events that are picked at step (2) REFERENCE EVENTS. In particular, the first event chosen at step (2) will be called the INITIAL REFERENCE EVENT and any subsequent events chosen at step (2) are FURTHER REFERENCE EVENTS. As noted, a suitable choice of reference events can 'maximise' the number of previous events removed at step (4) and thus 'minimise' the need for step (3) which typically involves real-world reasoning. In our work, we propose two heuristics for choosing the reference events.

Based on the discussion above, the algorithm for integrating a current situation into the preceding QTS is given in figure 4.5. As noted above, the INTEGRATE function takes three arguments: the current situation, the preceding QTS and any explicit integration information mentioned in the sentence describing the current situation. In the algorithm, the function CHOOSE-INITIAL-REFERENCE picks the initial reference event while the function CHOOSE-FURTHER-REFERENCE picks the further reference events. The function DET-RELATION determines the relation between the current situation and a reference event and will be further discussed in section 4.2.4.

4.2.1 Choosing the Initial Reference Event

Given a current situation, the simplest case occurs when the initial reference event is given explicitly. Consider, for instance, the following sentences:

(4.38) Before serving, add the wine.

```
Algorithm : INTEGRATE
Input     : CURRENT, (G, R), INTEG-INFO
Output    : Updated (G, R)

  1. If INTEG-INFO is not $\emptyset$, do
       (a) If INTEG-INFO = [REF, $\emptyset$], then
           Let RELATION = det-relation(CURRENT, REF, $\emptyset$)
       (b) If INTEG-INFO = [REF, CONNECTIVE], then
           Let RELATION = det-relation(CURRENT, REF, CONNECTIVE)
       (c) If INTEG-INFO = [REF, ASPECT, CONNECTIVE], then
           Let RELATION = det-relation(CURRENT, [ASPECT, REF], CONNECTIVE)
       (d) If INTEG-INFO = [REF, ASPECT], then
           Let RELATION = det-relation(CURRENT, [ASPECT, REF], $\emptyset$)
  2. If INTEG-INFO is $\emptyset$, then
       (a) Let REF = choose-initial-reference(CURRENT, (G, R))
       (b) Let RELATION = det-relation(CURRENT, REF, $\emptyset$)
  3. Let G = G $\cup$ {CURRENT}
     Let R = R $\cup$ {RELATION(CURRENT, REF)}
  4. Let (G, R) = choose-further-reference(CURRENT, REF, RELATION, (G, R))
  5. Return (G, R)
```

Figure 4.5: The INTEGRATION algorithm

(4.39) Bring the soup to the boil stirring occasionally.

where in the first sentence, the event described by the main clause is explicitly indicated to take reference from the event described by the subordinate *before*-clause, and in the second sentence, the event described by the progressive adjunct clause takes reference from the event described by the main clause. Note from the algorithm of figure 4.2, 4.3 and 4.4 that it is in these cases that the integration algorithm is passed some explicit integration information. For instance, in the case of sentence (4.38), at the point of integrating the event described by the main clause (say e_2), the following call is made:

(4.40) integrate(e_2, QTS$_p$, [e_1, before])

where QTS$_p$ is the preceding QTS , e_1 is the serving event described in the *before* clause. The third argument of the INTEGRATE function in (4.40) indicates explicitly that e_2 is to take initial reference from e_1 and that e_2 occurs before e_1 (determined from the *before* connective given).

We consider the case where no such explicit integration information is available such as at step (1b) in figure 4.2:

(4.41) integrate(sit_i, QTS$_p$, $\emptyset$)

In such case, the problem is to select the 'best' reference event from among the events in the preceding QTS.

The basic idea is to compute the 'similarity' between the current event and each of the events in the preceding QTS based on the objects involved in the events. More specifically, we say that an event e_1 is more similar to an event e_2 than to another event e_3 if e_1 shares more 'similar' objects with e_2 than with e_3. How then does one decides the 'similarity' of two objects? To answer this, we first consider the following examples:

Example 1 The example is as follows:

(4.42) a. e_1: Wash and core six cooking apples.

b. e_2: Put them into a fireproof dish.

The object in e_1 is the set of apple objects (x_1) described by the NP *six cooking apples.* There are two objects mentioned in e_2, namely the objects referred to by the pronoun *them* and the dish object described by *a fireproof dish.* The anaphoric *them* is resolved to x_1 so that there is an identical object in e_1 and e_2. In such case, we say that the NP *six cooking apples* in (4.42a) and the pronoun *them* in (4.42b) corefers and that they identify the same object. Thus, the event e_1 and e_2 share a common object.

Example 2 Here, we consider the following example:

(4.43) a. e_1: Break an egg.

b. e_2: Beat the egg yolk and the white so that they hold a peak.

In e_1, the object is the egg object (x_1) described by the indefinite NP *an egg.* Now, in e_2, the NPs *the egg yolk* and *the white* are referring to objects that are part of x_1. Thus, we can say that e_2 has two objects that are semantically related to the object in e_1. This semantic relation is one of WHOLE-PART.

Example 3 In the example below,

(4.44) a. e_1: Bring the soup to the boil.

b. e_2: Reduce the heat.

the entity described by the NP *the heat* is semantically related to the event e_1. This is because the heat entity is a resource that is required in bringing the soup to boil. Therefore, we can say that e_2 has an entity which acts as a resource for e_1. Alternatively, we say that the semantic link between the heat entity and e_1 is one of RESOURCE.

Based on the discussion above, we are now in a position to define a measure for the degree of similarity between the current event (*curr*) and a previous event (*prev*):

(4.45) degree-of-similarity($curr$, $prev$) = $\sum_i \sum_j f(prev, c\text{-}obj_i, p\text{-}obj_j)$

where $c\text{-}obj_i$ and $p\text{-}obj_j$ are the objects in the event $curr$ and $prev$ respectively. The function f itself is defined as follows:

$$f(prev, c\text{-}obj_i, p\text{-}obj_j) = \begin{cases} 1 & \text{if } c\text{-}obj_i \text{ and } p\text{-}obj_j \text{ are semantically related} \\ 1 & \text{if } c\text{-}obj_i \text{ and } prev \text{ are semantically related} \\ 0 & \text{otherwise} \end{cases}$$

where the possible semantic relations between $c\text{-}obj_i$ and $p\text{-}obj_j$ include 'identity', 'whole-part' and those between $c\text{-}obj_i$ and $prev$ include 'resource' (for the whole set of semantic relations we used, see section 4.3.1).

Having computed the degree of similarity between the current event and each of the events in the preceding QTS, we can then take the event in the preceding QTS which is most similar to the current event as the best initial reference event for the current event. Consider for instance the choice of the initial reference event for e_3 in the following example:

(4.46) a. e_1: Chop the meat.

b. e_2: Chop the spring onion.

c. e_3: Put the meat into the soup.

d. e_4: Bring the soup to the boil.

e. e_5: When the soup is boiling, add the spring onion.

At the point of integrating e_3, the QTS is as follows:

(4.47) $\text{QTS}_2 = (\{e_1, e_2\}, \{\langle \text{none}, e_2, e_1 \rangle\})$

Computing the degree of similarity using the formula in (4.45), we have

- degree-of-similarity(e_3, e_1) = 1
- degree-of-similarity(e_3, e_2) = 0

Thus, e_1 is taken as the initial reference event for e_3 although the clause describing e_1 is not the most recent one to the clause describing e_3.

We are still not in the clear yet since we have not considered the case where there may be more than one event in the preceding QTS with the same degree of similarity to the current event. For instance, in

(4.48) a. e_1: Simmer the soup for an hour.

b. e_2: Chop the meat.

c. e_3: Add the meat to the soup.

the QTS at the point of integrating e_3 is:

(4.49) $\text{QTS}_2 = (\{e_1, e_2\}, \{\langle \text{none}, e_2, e_1 \rangle\})$

Computing the degree of similarity between e_3 and e_1 and e_2 results in the following:

- degree-of-similarity(e_3, e_1) = 1
- degree-of-similarity(e_3, e_2) = 1

since e_3 and e_1 share one 'similar' object (namely the object described by *the soup*) and e_3 and e_2 share one 'similar' object (namely the meat object).

In such cases, the choice of initial reference from among the set of equally similar previous events (SIMILAR) is based on the following heuristic. This heuristic chooses the event among SIMILAR that is temporally the 'furthest'. We define the notion of an event being temporally further than another event as follows (recall that if an event e_1 **enables** another event e_2, then e_1 **precedes** e_2; similarly, if e_1 has **constituent** e_2, then e_1 **includes** e_2):

Definition 6

An event e is TEMPORALLY FURTHER than another event f if either

- precede(f, e) or enables(f, e) or

- meet(f, e) or
- include(e, f) or constituent(e, f)

This heuristic is chosen based on the assumption that later processed sentences tend to describe situations that occur later in time. The heuristic is implemented by removing from SIMILAR those events which **precedes, meets, enables,** is **included-in** or is **constituent-in** another event in SIMILAR. If there is more than one event remaining in SIMILAR after the removal, we choose the event that is processed the latest (again using our assumption). The steps taken to find the initial reference are thus:

1. Let MODULO = $\{e \mid e \in$ SIMILAR $\wedge \exists f \in$ SIMILAR such that
   e precedes or meets or enables
   or included in or constituent in $f\}$
2. Let REMAINDER = SIMILAR - MODULO
3. If there is only one event remaining in REMAINDER, then take this as the initial reference, otherwise
4. Choose the latest processed event in REMAINDER as the initial reference.

We now apply the above steps to determine the initial reference for e_3 in example (4.48). As discussed above, the set SIMILAR is $\{e_1, e_2\}$. Note from the QTS of this example given in (4.49) that the relation between e_1 and e_2 is **none**. The set MODULO at step 1 is therefore $\{\}$ which means that the set REMAINDER at step 2 is simply $\{e_1, e_2\}$. Therefore, we proceed to step 4 and choose e_2 as the initial reference event for e_3 since e_2 is processed later than e_1.

Now consider a case where the set MODULO at step 1 is not empty. For instance, consider the integration of e_3 in the following example:

(4.50) a. e_1: Finely chop the meat.

b. e_2: Add some sauce.

c. e_3: Leave to stand.

The QTS at the point of integrating e_3 is

(4.51) $\text{QTS}_2 = (\{e_1, e_2\}, \{\langle\text{after}, e_2, e_1\rangle\})$

Computing the degree of similarity between e_3 and e_1 and between e_3 and e_2 results in the following:

- degree-of-similarity(e_3, e_1) = 1
- degree-of-similarity(e_3, e_2) = 1

where the common object between e_3 and each of e_1 and e_2 is the meat object. The set SIMILAR is therefore $\{e_1, e_2\}$ and the set MODULO at step 1 is $\{e_1\}$ since e_1 precedes e_2. Thus, the set REMAINDER at step 2 is $\{e_2\}$ and since there is only one event in REMAINDER, this is taken as the initial reference for e_3.

Given an initial reference event for the current situation, we determine the relation between the initial reference event and the current situation. This relation is important for deciding further reference events (if any) for the current situation.

Based on the discussion above, the algorithm for choosing the initial reference event for a current situation when no explicit integration information is given is shown in figure 4.6.

4.2.2 Inferring Further Relations

Given an initial reference event (REF) for the current situation (CURR), we can find all the previous events which lie in a chain from CURR via REF such that the relations between CURR and the previous events *can be deduced.* We divide this discussion into five classes based on the possible relations that can hold between CURR and REF:

- Precedes, Meets and Enables
- After, Met-By and Enabled-By
- Includes and Constituent

```
Algorithm CHOOSE-INITIAL-REFERENCE
Input     CURR-EVENT, (G, R)
Output    REF-EVENT

  1. Let SIMILAR = $\{e \mid e \in G \wedge$
                   degree-of-similarity(CURR-EVENT, $e$) is the highest$\}$
  2. If SIMILAR = $\{e\}$, then Return $e$
  3. If SIMILAR is not a singleton set, then
      (a) Let MODULO = $\{e \mid e \in$ SIMILAR $\wedge \exists f \in$ SIMILAR $\wedge \exists r \in$ R such that
                       $r$ = precedes($e$, $f$) or meets($e$, $f$) or enables($e$, $f$)
                           or included-in($e$, $f$) or constituent-in($e$, $f$)$\}$
      (b) Let REMAINDER = SIMILAR - MODULO
      (c) If REMAINDER = $\{e\}$, then Return $e$
      (d) If REMAINDER is not a singleton set, then
          Return the event in REMAINDER that is processed the latest
```

Figure 4.6: The algorithm for choosing the initial reference event

- Included-In and Constituent-In
- None

and for ease of exposition, we will use the following equivalence:

- temporally-equal(CURR, REF) ≡ simultaneous(CURR, REF) ∨
  generates(CURR, REF)
- temporally-before(CURR, REF) ≡ precedes(CURR, REF) ∨
  meets(CURR, REF) ∨
  enables(CURR, REF)
- temporally-after(CURR, REF) ≡ after(CURR, REF) ∨
  met-by(CURR, REF) ∨
  enabled-by(CURR, REF)
- temporally-include(CURR, REF) ≡ include(CURR, REF) ∨
  constituent(CURR, REF)

- temporally-included-in(CURR, REF) ≡ included-in(CURR, REF) $\vee$
  constituent-in(CURR, REF)

Precedes, Meets and Enables: The **enablement** relation is included in this case since this relation has the direct temporal import that if an event e_1 enables an event e_2, then e_1 either precedes or meets e_2. If CURR is temporally before REF, we have

(4.52) temporally-before(CURR, REF)

Given (4.52), the previous event (PREV) whose relation with CURR can be deduced is given by the following rules:

(4.53)	a.	If	temporally-before(CURR, REF)
		And	temporally-before(REF, PREV)
		Then	precedes(CURR, PREV)
	b.	If	temporally-before(CURR, REF)
		And	temporally-include(REF, PREV)
		Then	precedes(CURR, PREV)
	c.	If	temporally-before(CURR, REF)
		And	temporally-equal(REF, PREV)
		Then	precedes(CURR, PREV)

After, Met-By and Enabled-By: This case is the reverse of the case just considered above. If CURR is temporally after the reference event REF, we have

(4.54) temporally-after(CURR, REF)

and the previous event (PREV) whose relation with CURR can be deduced is given by the following rules:

(4.55)	a.	If	temporally-after(CURR, REF)
		And	temporally-after(REF, PREV)
		Then	after(CURR, PREV)

b. If temporally-after(CURR, REF)
And temporally-include(REF, PREV)
Then after(CURR, PREV)

c. If temporally-after(CURR, REF)
And temporally-equal(REF, PREV)
Then after(CURR, PREV)

Includes and Constituent: The **constituent** relation is included in this case since this relation has the temporal import that if an event e_1 has constituent e_2, then e_1 includes e_2. If CURR temporally includes REF, we have

(4.56) temporally-include(CURR, REF)

and the previous event (PREV) whose relation with CURR can be deduced is given by the following rules:

(4.57) a. If temporally-include(CURR, REF)
And temporally-include(REF, PREV)
Then include(CURR, PREV)

b. If temporally-include(CURR, REF)
And temporally-equal(REF, PREV)
Then include(CURR, PREV)

Included-In and Constituent-In: The **constituent-in** relation is the inverse of the **constituent** relation and is included in this case since this relation has the temporal import that if an event e_1 is a constituent in an event e_2, then e_1 is included in e_2. If CURR is temporally included in REF, we have

(4.58) temporally-included-in(CURR, REF)

and the previous event (PREV) whose relation with CURR can be deduced is given by the following rules:

(4.59) a. If temporally-included-in(CURR, REF)
And temporally-included-in(REF, PREV)
Then included-in(CURR, PREV)

b.	If	temporally-included-in(CURR, REF)
	And	temporally-equal(REF, PREV)
	Then	included-in(CURR, PREV)
c.	If	temporally-included-in(CURR, REF)
	And	temporally-after(REF, PREV)
	Then	after(CURR, PREV)
d.	If	temporally-included-in(CURR, REF)
	And	temporally-before(REF, PREV)
	Then	precedes(CURR, PREV)

None: Recall from chapter 3 that the temporal import of the **none** relation is dependent on the environment. If CURR has a **none** relation with REF, we have

(4.60) none(CURR, REF)

and the previous event (PREV) whose relation with CURR can be deduced is given by the following rule:

(4.61)	If	none(CURR, REF)
	And	temporally-equal(REF, PREV)
	Then	none(CURR, PREV)

4.2.3 Choosing Further Reference Events

At times, one reference event is insufficient to relate the current situation with respect to all the situations in the preceding QTS. Consider for instance

(4.62) a. Soak and drain the beans.

b. Meanwhile, e_4: chop a carrot.

where the QTS at the point of integrating the event described by *chop a carrot* is as given in (4.63):

(4.63) $\text{QTS}_3 = (\{e_1, e_2, e_3\},$

$\{\langle$after, e_2, $e_1\rangle$, $\langle$constituent, e_3, $e_1\rangle$, $\langle$constituent, e_3, $e_2\rangle\})$

where e_1 is the event described by the clause *soak the beans*, e_2 is the event described by the clause *drain the beans* and e_3 is the composite event described by the sentence *Soak and drain the beans.*

We now integrate e_4. The *meanwhile* connective indicates explicitly that e_4 takes initial reference from the event described by the preceding sentence, i.e. e_3. In addition, e_4 is explicitly indicated to be included in e_3. Thus, the event e_4 is related with respect to e_3. But, is it related with respect to e_2 and e_1 ? The answer is no since there exists no chain from e_4 to either e_1 or e_2 such that the relations between e_4 and e_1 and between e_4 and e_2 can be deduced. This is because temporally e_4, e_1 and e_2 are all included in e_3. We thus need to pick either e_1 or e_2 as further reference. We choose e_2 since it is the later processed event and determine the relation between e_4 and e_2. The default none relation holds between e_4 and e_2. Again, there exists no chain from e_4 to e_1 that will allow the relation between e_4 and e_1 to be deduced. We therefore need to determine the relation between e_4 and e_1. This relation is again the default none relation.

The QTS of example (4.62) is thus:

(4.64) QTS$_3$ = $(\{e_1, e_2, e_3\}$,
$\{\langle$after, e_2, $e_1\rangle$, $\langle$constituent, e_3, $e_1\rangle$,
$\langle$constituent, e_3, $e_2\rangle$, $\langle$included-in, e_4, $e_3\rangle$,
$\langle$none, e_4, $e_2\rangle$, $\langle$none, e_4, $e_1\rangle\})$

In this QTS , all situations are integrated and we note that e_4 takes reference from e_3, e_2 and e_1.

Based on the discussion above, we need to find further reference events for a current situation until the situation is related with respect to all the situations in the preceding QTS . This process begins by filtering any event PREV in the preceding QTS which lies on a chain from the current situation CURR via the initial reference REF (i.e. CURR $\rightarrow$ REF ... $\rightarrow$ PREV) such that the relation between CURR and PREV can be deduced (see section 4.2.2). We then choose the next reference event for the current situation from the remaining events. To do this, we will use the 'similarity' and 'temporally further' heuristics discussed in section 4.2.1. The process then repeats.

An algorithm implementing the above process to choose further reference event for a current situation is given in figure 4.7. The CHOOSE-FURTHER-REFERENCE function takes four arguments: the current situation (CURR), a reference event (REF), the relation between the current situation and the reference event, and the preceding QTS.

Note from step (1) of the algorithm that if the relation between CURR and REF is one of either **simultaneous** or **generation**, then no further reference events are needed. This is because if CURR is **simultaneous** with REF (recall that if an event e_1 **generates** an event e_2, then e_1 is **simultaneous** with e_2), we can take CURR to be 'coextensive' with REF. Since REF is already integrated in the QTS , CURR will be.

For other relations between CURR and REF, the CHOOSE-FURTHER-REFERENCE function calls the TEMPORALLY-INFERRABLE-EVENTS function to determine the set of events in the preceding QTS whose relations with CURR can be deduced. Based on the discussion in section 4.2.2, the TEMPORALLY-INFERRABLE-EVENTS function can be implemented as a process of forward rule chaining initiated by asserting the relation between CURR and REF (see section 6.5).

4.2.4 Determining the Relations

Given a current situation and its reference event, we need to determine the relation that holds between them. To do this, we employ both linguistic information and real-world knowledge. These information sources are encoded as rules and are arranged such that rules encoding linguistic information are tried first. When linguistic information does not give a relation, rules encoding real-world knowledge are tried. If no relations are determined by the various information sources, we say that the default **none** relation holds between the current situation and the reference event.

The task of determining the relation between the current situation and a reference event is carried out in the DET-RELATION function. We will now describe the various

```
Algorithm : CHOOSE-FURTHER-REFERENCE
Input     : CURR, REF, RELATION, (G, R)
Output    : Updated (G, R)

  1. If RELATION is simultaneity or generation, then return (G, R)
  2. Otherwise, do
     (a) Let FURTHER-REF = $\{e \mid e \in$ G $\wedge\ e$ is an event $\wedge\ e \neq$ REF$\}$
     (b) Let S = FURTHER-REF −
              temporally-inferrable-events(CURR, REF, RELATION, FURTHER-REF, R)
     (c) If S is $\emptyset$, Return (G, R)
     (d) If S = $\{e\}$, let REF = $e$
     (e) If S is not a singleton set, then
          i. Let SIMILAR = $\{e \mid e \in$ S $\wedge$
                   degree-of-similarity(CURR-EVENT, $e$) is the highest$\}$
         ii. If SIMILAR = $\{e\}$, then let REF = $e$
        iii. If SIMILAR is not a singleton set, then
             A. Let MODULO = $\{e \mid e \in$ SIMILAR $\wedge\ \exists\ f \in$ SIMILAR $\wedge\ \exists\ r \in$ R such that
                     $r$ = precedes($e$, $f$) or meets($e$, $f$) or enables($e$, $f$)
                        or included-in($e$, $f$)$\}$
             B. Let REMAINDER = SIMILAR - MODULO
             C. If REMAINDER = $\{e\}$, let REF = $e$
             D. If REMAINDER is not a singleton set, let REF be the event in RE-
                MAINDER that is processed the latest.
     (f) Let RELATION = det-relation(CURR, REF)
     (g) Let R = R $\cup$ {RELATION(CURR, REF)}
     (h) Let FURTHER-REF = S − {REF}
     (i) Goto step (2b)
```

Figure 4.7: The algorithm for choosing further reference events

information sources used by this function.

Temporal Connectives, Temporal Adverbials and Aspect

The simplest case occurs when the relation is expressed explicitly via a temporal connective such as *before* and *meanwhile* (but excluding the *when*-connective). Consider, for instance

(4.65) Before e_1: you mash the potato, e_2: take out the chicken leg.

where the call to integrate e_2 (see step (4) of figure 4.3) is:

(4.66) integrate(e_2, QTS$_1$, [e_1, before])

where QTS$_1$ is the preceding QTS . From step (1b) of the integration algorithm given in figure 4.5, the call to determine the relation between e_2 and e_1 is:

(4.67) det-relation(e_2, e_1, before)

This fires the following rule which encodes the semantics of the *before* connective:

(4.68) If the current situation is CURR
And the reference event is REF
And the temporal connective is before
Then precede(CURR, REF)

Now, consider the case which involves a *when*-connective such as (4.69a) and (4.69b), (4.69a) and (4.69c) and (4.69a) and (4.69d):

(4.69) a. e_1: Cool the water.
b. When the water has cooled, e_2: put in the vegetables.
c. When the water has cooled for 5 minutes, e_2: put in the vegetables.
d. When the water is cooling, e_2: put in the vegetables.

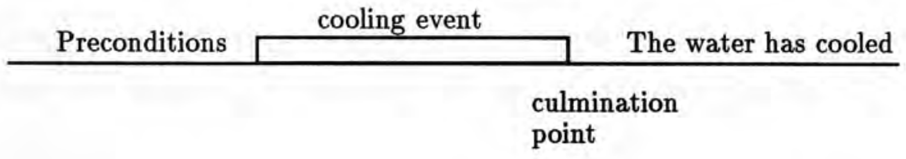


Figure 4.8: The event structure of e_1 described by *cool the water*

We assume that at the end of processing (4.69a), the QTS is QTS$_1$. When processing either (4.69b), (4.69c) or (4.69d), we know from step (4) of the algorithm in figure 4.3 that there is a call to build the situation representation underlying the *when*-clause. During this call, the *when*-clauses in (4.69b) to (4.69d) are determined to refer to different parts of the structure associated with the event e_1 (see section 4.3 for how this is done). In our work, we adopt the event structure proposed in [Song 91b, Singh & Singh 92]. Our main motivation for adopting this event structure is that it clearly indicates the start and end point of an event. In addition, we employ the following kinds of aspects:

Perfect This aspect describes the end state that occurs after the completion of the event, for example, *The water has cooled.*

Progressive This aspect presents an event internally, for example, *John was eating an apple* (see chapter 2).

Neutral Perfective This aspect is introduced in [Singh & Singh 92] and describes an event that has ended, but not necessarily at its natural end point. For example, the *when*-clause in *when the water has cooled for 5 minutes, put in the meat* describes a cooling event from a neutral perfective aspect (see chapter 2).

For example (4.69), the event structure corresponding to e_1 is shown in figure 4.8. where 'preconditions' indicates the state before the cooling begins. The parts of the structure in figure 4.8 referred to by the various *when*-clauses in example (4.69d) are as follows:

- In (4.69b), the clause *the water has cooled* is taken as referring to the end state. We say that the aspect indicated is perfect.

- In (4.69c), the clause *the water has cooled for 5 minutes* is taken as referring to those preliminary processes in the event structure associated with e_1 such that their total duration is 5 minutes. We say that the aspect indicated is neutral perfective.
- In (4.69d), the clause *the water is cooling* is taken as referring to the internal process of the cooling in e_1. We say that the aspect indicated is progressive

Therefore, the *when*-clauses in (4.69b) to (4.69d) refer to different aspects of a previously mentioned event. This means that step (4c) of the algorithm in figure 4.3 is applicable and the calls to integrate e_2 for (4.69b) to (4.69d) are respectively:

(4.69) b. integrate(e_2, QTS$_1$, [e_1, perfect, when])

c. integrate(e_2, QTS$_1$, [e_1, neutral-perfective, when])

d. integrate(e_2, QTS$_1$, [e_1, progressive, when])

These integration calls in turn invoke the DET-RELATION function to determine the relation between e_2 and e_1 (see figure 4.5):

(4.70) a. det-relation(e_2, e_1, [perfect, when])

b. det-relation(e_2, e_1, [neutral-perfective, when])

c. det-relation(e_2, e_1, [progressive, when])

From (4.70a), we determine that e_2 occurs after e_1 using the following rule:

```
(4.71)  If      the current situation is CURR
        And     the reference event is REF
        And     CURR is an event
        And     the aspect of REF is perfect
        Then    after(CURR, REF)
```

Similarly, from (4.70c), we determine that e_2 is **included in** e_1 using the following rule:

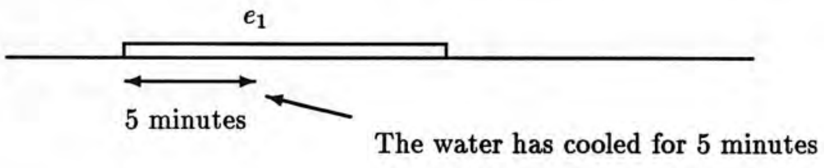


Figure 4.9: The neutral perfective viewpoint as expressed in *the water has cooled for 5 minutes*

(4.72)	If	the current situation is CURR
	And	the reference event is REF
	And	CURR is an event
	And	the aspect of REF is progressive
	Then	included-in(CURR, REF)

The case with (4.70b) needs some explanations.

As noted in section 2.1.2, when a clause is describing an event from a neutral perfective viewpoint, the clause is referring to those preliminary processes such that their total duration is as indicated in the clause. The situation for the case of *the water has cooled for 5 minutes* is depicted graphically in figure 4.9. This means that from (4.70b) it is only possible to say that the start of e_1 precedes the start of e_2 by 5 minutes. In addition, since e_1 has not ended, the start of e_2 must precede the end of e_1. As noted in chapter 3, these are quantitative relations between the time points of the time intervals associated with e_1 and e_2:

(4.73) a. $\text{start}(te_2) - \text{start}(te_1) = 5$

b. $\text{end}(te_1) - \text{start}(te_2) > 0$

where te_1 and te_2 are the time intervals associated with the event e_1 and e_2 respectively (see chapter 3).

Therefore, unlike (4.70a) and (4.70c), (4.70b) does not indicate a qualitative relation between e_1 and e_2.[1] We thus need to determine the relation between e_2 and e_1 via other information sources (see below).

Knowledge About Actions

Recall from chapter 3 that events are defined as actions to be carried out by an agent over a time interval. In our work, we employ a STRIPS representation [Fikes & Nilsson 71] for actions. In such representation, we encode the PRECONDITIONS and POSTCONDITIONS of the action, i.e., those facts about the world that must hold *before* the action can be carried out, and those facts about the world that must hold *after* the action is carried out. For example, the peeling action in the event described by *peel a potato* is represented as follows:

(4.74)	Action:	Do(Agent, PEEL(x))
	Preconditions:	$\neg$peeled(x, +)
	Postconditions:	peeled(x, +)

As noted in chapter 3, these actions are organised into a taxonomic hierarchy using an IS-A link.

This knowledge about actions is used in determining the **enablement** and **generation** relation between the current event and its reference event. Consider, for instance

(4.75) a. Cook the rice.

b. Cool.

(4.76) As you work, *clean the surface* thoroughly each time you change grits, *vacuuming off all the dust* and *wiping the wood* with a rag dampened with turpentine or paint thinner.

[1]Of course, from the quantitative relations given above and the duration of e_1 and e_2, one can deduce that e_1 either overlaps or includes e_2. Such deductions are carried out in the second step of TA2 via constraint propagation (see chapter 5).

In (4.75), a precondition for the cooling action is that the object being cooled must be hot. In the cooking action, a postcondition is that the object being cooked will get hot. Thus, the cooking action brings about a condition which is necessary for the subsequent performance of the cooling action. In other word, the cooking event **enables** the cooling event. In (4.76), the vacuuming action and the wiping action are classified as a specialisation of the cleaning action. This classification allows us to determine that performing the vacuuming and wiping action results in the cleaning action. That is, the cleaning event is **generated** by the event comprising the sequence of vacuuming and wiping event.

Information from Reference Resolution

In our work, we model the effects of an action. Such modelling is required to maintain the correct state of objects so that referring expressions can be resolved correctly. Consider, for instance,

(4.77) a. Brown the beef until tender.

b. e_2: Chop one of the onions.

c. e_3: Add the chopped onion to the beef and fry together.

where the NP *the chopped onion* is resolved correctly to the onion that was chopped in e_2. Notice that if the effect of the chopping action in e_2 is not modelled, then it would not be possible to obtain the correct referent for the NP *the chopped onion* since none of the onions would have the 'chopped' property added to it.

Now, notice that the onion object in e_3 is a result of modelling the effect of e_2. This means that e_3 must occur after e_2. Similarly, as will be discussed in section 4.3, the ellided NP in (4.78b) is resolved to the result of executing the bring-to-boil action in (4.78a):

(4.78) a. e_1: Bring the soup to the boil.

b. e_2: Stir.

This means that we can determine that e_2 occurs after e_1.

4.2.5 Summary and Related Work

To summarise, we regard the integration problem as finding the set of reference events such that the current situation is integrated in the QTS, i.e. related with respect to all the preceding situations in the QTS. Our integration process proceeds by finding an initial reference event based on the degree of similarity between the current situation and each of the possible reference events. This degree of similarity is computed based on the semantic relationships between objects in the events and between objects and events. Any previous events whose relations with the current situation cannot be deduced via the initial reference serve as possible further references. If there is more than one such preceding events remaining, we employ a notion of 'temporally further' to choose the reference event. The integration process then repeats.

Our approach to the integration problem makes explicit the notion of reference which is regarded as intuitive in previous work. In our work, the reference events for a current situation are those preceding events which can serve as an 'anchor' from which the relations between the current situation and the rest of the preceding events can be deduced. Our approach avoids the difficulties of approaches recently proposed [Webber 87, Song & Cohen 91b, Hwang & Schubert 91, Eberle 92]. Unlike [Webber 87, Song & Cohen 91b, Hwang & Schubert 91] who use only tense information, our approach is not dependent on tense change to decide on the right reference events for the current situation. Therefore, we believe that our approach can handle those discourses where tense change does not provide enough information to decide on the correct reference event (see chapter 2).

In [Eberle 92], the possible reference events for the current event are those which are regarded as 'accessible events' (see chapter 2). However, this approach does not consider the case where an inaccessible event may be explicitly mentioned as a reference event for the current event. Consider, for instance,

(4.79) a. e_1: Mash the potato.

b. e_2: Add to the soup.

c. Before you mash the potato, e_3: take out the chicken leg from the soup.

At the point of integrating e_3, the event structure built up by Eberle is such that e_2 is a continuation of e_1. Thus, using the criterion for accessible event, e_1 is no longer accessible. However, the *before* clause indicates explicitly that e_3 is to take reference from this inaccessible event e_1. In our approach, we determine that the *before* clause is describing an event which is identical with e_1 and that e_3 takes its initial reference from e_1.

4.3 Reference Resolution

In this section, we present the model of reference resolution used in our work. As noted in the discussion above, information from reference resolution is used for the following:

- Reference resolution of the NPs determines the semantic links between objects in events and between objects and events. This is needed in the computation of the degree of similarity between events.
- Reference resolution determines whether different clauses are referring to the same event (perhaps with different aspect) or a different event that may or may not be related to another event. This information helps in determining the initial reference event for a current event and also the temporal relation between events (see section 4.2.4).

Traditionally, the term REFERENCE is used to name the study of how phrases in a sentence relate to objects in the real world. For example, the NP *the chopped onion* describes a particular onion object that is chopped. This is approximated in computational systems by relating the referring expressions (i.e. linguistic expressions used for the purpose of reference) to terms in the knowledge representation which in turn represent some object in the world. In our work, a referring expression can either relate to a physobj, an event or state. For instance, the NP *five pounds of carrots* refers to a physobj which is to be viewed as a set whose elements are the individual carrots such that their total weight is 5 pounds, and the NP ***a carrot and a potato*** refers to a physobj which is to be viewed as a set whose constituents are the physobj described by the NP *a carrot* and that described by the NP *a potato*. Similarly, the sentence *bring the soup to the boil stirring occasionally* refers to an event which is to be viewed as a composite event whose constituents are the event described by the clause *bring the soup to the boil* and that described by the clause *stirring occasionally*. For ease of exposition, we will call the entity which a referring expression identifies the REFERENT of the expression.

Reference resolution can now be defined as the process of finding the entities in the knowledge representation which a referring expression identifies. A common way of tackling the reference resolution problem in a computational system is to view it as a SEARCH for the right *referent* in a model containing all the entities (represented in the knowledge representation) that have been seen so far in the discourse. Following [Sidner 79, Webber 78], we will call this model the DISCOURSE MODEL (DM). Likewise, in our work, we adopt this common approach extended to incorporate the effects of modelling the events.

In the next two sections, we will present our approach to resolving for the referents to objects and situations. There are two main questions that the approach will attempt to answer, namely *where* and *how* the referent is to be found.

4.3.1 Resolution to Objects

In this section, we consider the problem of resolving NPs to the objects in the knowledge representation they identify. This problem has received much attention in the literature [Sidner 79, Webber 78, Grosz *et al* 83]. As noted above, the basic idea is to search for the referent of an anaphoric element in the discourse model containing all the objects that are MENTIONED so far in the discourse.

In our work, we adopt the same approach that to resolve the referent of an NP, one searches among the objects in the discourse model. However, we differ in that the discourse model need not only contain objects that are mentioned in the discourse. It can also include objects that are introduced through event simulation.

We will now discuss event simulation before presenting the discourse model used in our work and the search process through the model.

Event Simulation

As noted in chapter 3, every event has a begin state (s_b) and an end state (s_e). A state is described by the set of propositions which hold in the state. Consider, for instance, the following simple recipe:

(4.80) a. x_1: an onion

b. e_1: Peel the onion.

In the begin state of e_1, the following propositions (see chapter 3) hold:

(4.81) hold(s_b, substance(x_1, onion-matter)) $\wedge$
hold(s_b, structure(x_1, individual)) $\wedge$
hold(s_b, packaging(x_1, $\langle$onion, regular$\rangle$))

We now simulate the effect of the action in e_1. As noted above, actions are represented as operator using the STRIPS notation. For instance, the peeling action in e_1 is represented using the STRIPS notation given in (4.74) above and reproduced below:

(4.74)	Action:	Do(Agent, PEEL(x))
	Preconditions:	$\neg$peeled(x, +)
	Postconditions:	peeled(x, +)

From (4.81), we know that the precondition of the peeling action is satisfied since there did not exist a proposition peeled(x, +) in the begin state of e_1 (see (4.81)). Therefore, we carry out the action and add the postcondition to the end state of e_1. In the end state of e_1, the following propositions hold:

(4.82) hold(s_e, substance(x_1, onion-matter)) $\wedge$
hold(s_e, structure(x_1, individual)) $\wedge$
hold(s_e, packaging(x_1, $\langle$onion, regular$\rangle$)) $\wedge$
hold(s_e, peeled(x_1, +))

Observe that in simulating an event, we use the default assumption that if a proposition in the begin state is not affected by an action, it persists into the end state of the event. For example, the proposition structure(x_1, individual) is not affected by the peeling action and so persists into the end state.

In addition to events that change the properties of objects, we also have events that create new objects either by combining two or more objects, or by breaking an existing object into parts. These events change the population of objects in the world. Following [Dale 92], we use the notion of WORKING SET to model the effects of these events. The working set is basically a list of the identifiably distinct objects in the domain at any point in time. It is represented as a distinguished physobj, whose constituents are just the ingredients in the recipe. For instance, in the following simple recipe:

(4.83) a. x_1: some melted butter

b. x_2: a potato

c. e_1: Add the potato to the butter.

the begin state of e_1 contains a proposition of the form:

(4.84) hold(s_b, constituent(working-set, [x_1, x_2]))

We now simulate the adding action in e_1 which is represented as the following STRIPS operator:

(4.85)

Action:	Do(Agent, ADD(x, y))
Preconditions:	$x \in$ Working Set
	$y \in$ Working Set
Postconditions:	constituents(z, [x, y])

$$x \notin \text{Working Set}$$
$$y \notin \text{Working Set}$$
$$z \in \text{Working Set}$$

Therefore, after simulation, the end state of e_1 will contain the following propositions:

(4.86) hold(s_e, constituent(working-set, [x_3])) $\wedge$
hold(s_e, constituent(x_3, [x_1, x_2]))

We will now discuss the impact of such event simulation on the discourse model.

Our Discourse Model

Traditionally, there are two ways in which new objects can be added to the discourse model:

- The first way is via the use of an indefinite NP. For example, the indefinite NP *a carrot* introduces a new carrot object into the discourse model.
- The second way is via a definite NP. For instance, in

  (4.87) a. Wash an apple.
  b. Peel off the skin.

  the definite NP *the skin* in the second sentence identifies an object that is semantically related to the object described by the NP *an apple* in the first sentence (more specifically, the skin object is part-of the apple object).

  In such cases, a new object corresponding to the NP *the skin* is created and added to the model.

In our work, we identify a third way in which new object can be added to the discourse model. This is via events that create new objects from existing objects. Consider, for instance, (4.88) a. Melt some butter.

b. Add some carrots and potatoes.

c. Saute them for 7 to 8 minutes.

d. Then, add some butterbeans, stock and milk.

e. Simmer *the soup* for 1 hour.

The referent of the definite NP *the soup* in (4.88e) is actually the product of the adding action in (4.88d). This means that any new object produced by event simulation has to be added to the discourse model. A similar argument is given in [Webber & Baldwin 92] to show why the discourse model used in reference resolution needs to incorporate entities which are introduced by event simulation. For instance, [Webber & Baldwin 92] gives the following example:

(4.89) a. Mix the flour, butter and water.

b. Knead *the dough* until smooth and shiny.

where the object identified by the definite NP *the dough* is the product of the mixing event described in the first sentence.

It is not an easy matter to determine the result of an event which produces new object. For instance, in (4.89b), the result of the mixing action in (4.89a) in some circumstances may also be a paste or a batter. Currently, in our work, we simply invoke rules encoding hard-wired knowledge to determine the result of actions that produce new objects[2]. For instance, in our work, the ADD operator given in (4.85) is actually implemented as:

(4.90) Action: Do(Agent, ADD(x, y))
Preconditions: $x \in$ Working Set
$y \in$ Working Set
Postconditions: invoke-ruleset(add-rules, x, y, z)

[2]For some initial discussions on how such hard-wired knowledge can be generalised, see [Webber & Baldwin 92].

$$x \notin \text{Working Set}$$
$$y \notin \text{Working Set}$$
$$z \in \text{Working Set}$$

where the function *invoke-ruleset* invokes the ruleset ADD-RULES to determine the result of adding x and y. The ADD-RULES ruleset is a collection of rules and the first rule that applies to x and y returns (see chapter 6). For instance, if we say *Add the vegetables to the water*, then x is the physobj described by ***the vegetables*** and y is the physobj described by *the water*. The rule that applies is:

(4.91) If either x or y is a liquid
then constituent(z, [x, y]) $\wedge$
substance(z, soup-matter)

which essentially says that if one of the objects being added is a liquid, then the result will be soup.[3]

Based on the discussion above, our discourse model contains objects that are mentioned in the ingredient list and objects that are the results of event simulation (the discourse model also contains the events and states mentioned but we postpone this discussion to section 4.3.2). In addition, we also define a current CENTER in our discourse model used for resolving inter-sentential pronouns and ellided NPs. This notion of CENTER is intuitively similar to the notion of CENTERING proposed in [Grosz *et al* 83]. In addition, we follow [Dale 92] in defining the center to be the product of simulating the current event. Consider, for instance

(4.92) a. Bring the soup to the boil.
b. Cool $\emptyset$.

where the current center at the end of processing the first sentence is the result of modelling the bring-to-boil event (the boiling soup). The ellided NP in the second sentence is then resolved to this center.

[3]This is obviously a simplification but suffices for the purpose of our work.

To summarise, the discourse model (DM) used in our work consists of the following:

- a set of objects that are either mentioned in the text or introduced through event simulation,
- the current center and
- a set of events and states mentioned in the text (see section 4.3.2).

Our Search Mechanism

We will now discuss how we carry out the search for the referent of a referring expression in our discourse model.

The basic approach is to carry out lexical disambiguation of the verb and the reference resolution of the noun phrases in the clause simultaneously. The main motivation for doing this is that when resolving the noun phrases we can take into account constraints from all the words in the clause.

More specifically, the approach involves first determining the sets of objects in the DM which can be the referents of the noun phrases. This determination makes use of any constraints expressed in the NP. Then, we use constraints arising from the verb to remove all those referents that are not suitable.

The above approach can be easily viewed as a problem of constraint satisfaction which is defined as:

> A constraint-satisfaction problem (CSP) consists of a set of *variables*, each of which must be instantiated in a particular *domain* of values, and a set of *predicates/constraints* which the values of the variables must simultaneously satisfy [Mackworth 87, 206].

The problem can be schematised as follows:

(4.93) $(\exists\, v_1)(\exists\, v_2)\ldots(\exists\, v_n)(v_1 \in D_1)(v_2 \in D_2)\ldots(v_n \in D_n)\ \mathrm{P}(v_1, v_2, \ldots, v_n)$

where P(v_1, v_2, ..., v_n) stands for an arbitrary conjunction of predicates over subsets of the variables $\{v_1, v_2, \ldots, v_n\}$.

We now describe how the CSP is set up for a clause:

Variables We assign a variable to each verb and NP in the clause. Consider, for instance

(4.94) a. Chop the carrots.

b. Chop the carrots and potatoes.

where in the first case the CSP contains two variables corresponding to the verb *chop* and the NP *the carrots*, and likewise in the second case, the CSP contains two variables corresponding to the verb *chop* and the NP *the carrots and potatoes.*

Domains Recall from chapter 3 that we maintain a taxonomic hierarchy of actions. Verbs are taken as linguistic expressions of these actions and the same expressions can be used for different actions. For example, the verb *have* is ambiguous between owning (*John has a piano*) and afflicted-with (*John has a fever*). Therefore, the domain for the verb variable is the possible actions which the verb can describe.

The domain for a referring expression is decided as follows:

- If the referring expression is an indefinite NP, the domain is simply the new object created from the NP.
- If the referring expression is a pronoun or ellided NP, the domain is simply the current center in the discourse model.
- If the referring expression is a definite NP, then the domain is the set of objects in the discourse model which satisfy the constraints given in the NP itself. For example, the domain of the NP *the chopped carrot* will be the set of objects in DM which are both carrots and chopped (we will discuss how this domain is obtained below).

Constraints The constraints are constraints arising from the verb in the clause. In the case where there is a pronoun, additional constraints may be derived. For instance, if the referring expression is the pronoun *them*, then the referent must be a physobj (see section 3.3.1) which is to be viewed as a set.

We now illustrate the CSP for two examples.

Example 1 The CSP corresponding to the clause *chop them* is

(4.95) $(\exists\ action_1)\ (\exists\ obj_1)$
$(action_1 \in \{\text{chopping}\})\ (obj_1 \in (\text{GET-CENTER DM}))$
$\text{structure}(obj_1, \text{set}) \wedge \text{action-constraints}(action_1, obj_1)$

where $action_1$ and obj_1 are the variables associated with the verb and the pronoun in the instruction respectively, GET-CENTER is a function which returns the current center in the discourse model and action-constraints($action_1$, obj_1) is a function which retrieves the constraints stored in $action_1$. For instance, in the course of solving the CSP, $action_1$ will get bound to the chopping action. The constraints retrieved is then simply that obj_1 is an object that can be chopped:

(4.96) choppable(obj_1)

Example 2 The CSP corresponding to the clause *chop the carrots* is:

(4.97) $(\exists\ action_1)\ (\exists\ obj_1)$
$(action_1 \in \{\text{chopping}\})\ (obj_1 \in (\text{FIND-NP-DOM NP}))$
$\text{action-constraints}(action_1, obj_1)$

where $action_1$, obj_1 and action-constraints($action_1$, obj_1) are as before. In addition, the function FIND-NP-DOM returns the objects that satisfy the constraints given in the NP.

We will now describe the approach behind the FIND-NP-DOM function.

Finding the Domain of Definite NP

If a clause contains a definite NP, the domain of the variable assigned to the NP needs to be determined. Recall that a definite NP can identify an object that is already in the discourse model or an object which is semantically linked to its referent. In our work, we use the following five kinds of link between an object described by a definite NP and a referent of the NP:

Identity: This link exists if the object described by the definite NP and a referent of the NP is identical. For instance, in

(4.98) a. Wash and core six cooking apples.

b. Put *the apples* into the oven dish.

the definite NP *the apples* and the indefinite NP *six cooking apples* corefer to the same set of apple objects. In such cases, we say that there is an identity link between the object described by *six cooking apples* and *the apples.*

Whole-part: Consider for instance

(4.99) a. Break an egg.

b. Beat *the yolk* and the white so that they hold a peak.

where the object described by the definite NP *the yolk* is part-of the egg object introduced by the indefinite NP *an egg* in the previous sentence. In such case, we say that there is a whole-part link between the egg object and the yolk object.

Class-subclass: Consider for instance

(4.100) a. Soak, drain and rinse the celery.

b. Melt the butter in a large saucepan and add *the vegetable.*

where the object described by the definite NP *the vegetable* is a superclass of the celery object mentioned earlier. In such case, we say that there is a class-subclass link between the celery object and the vegetable object.

Set-subset: Consider for instance

(4.101) a. Shell the eggs.

b. Finely chop *two of the eggs* and stir into the kedgeree with the parsley.

where the objects described by the NP *two of the eggs* is a subset of the set of egg objects described by *the eggs* in the first sentence. In such case, we say that there is a set-subset link between the object described by *two of the eggs* and the set of egg objects described by *the eggs*.

Resource: Consider for instance

(4.102) a. e_1: Bring the soup to the boil.

b. Reduce *the heat.*

where the entity described by *the heat* is a resource required in e_1. In such case, we say there is resource link between the heat entity and the bring-to-boil event.

Based on the five kinds of links given above, our approach to determining the object identified by a definite NP is as follows:

If the NP is a partitive NP of the form N of NP_1 where N is a number (such as *two of the eggs*), we take the link type between the object identified by the NP and the referent of the NP to be set-subset. The set object is determined by the reference resolution of NP_1. For each referent of NP_1, we create a new object and mark its relation with the referent as set-subset. We then return these new objects as the objects identified by the definite NP.

For the other types of NP, we do the following:

1. Hypothesize a link type from the set {identity, whole-part, class-subclass, resource}.
2. Search the DM for the referents that can be linked to the object described by the definite NP with the hypothesized link type.
3. If no referents exist, try another link type and repeat step (2).
4. If such referents exist, do the following:
   (a) If the link type is IDENTITY, return the referents as the objects identified by the definite NP.

(b) If the link type is either WHOLE-PART or RESOURCE, create a new object for each referent found and mark the relation between them appropriately. Then, we return these new objects as the objects identified by the definite NP.

(c) If the link type is CLASS-SUBCLASS, we do the following:

i. If there is only one referent found, we create a new object and mark the relation between this object and the referent as class-subclass. This object is then returned as the object identified by the definite NP. For instance, in example (4.100) given above and reproduced below:

(4.100) a. Soak, drain and rinse the celery.

b. Melt the butter in a large saucepan and add *the vegetable.*

we create a new vegetable object and mark the relation between this object and the celery object as class-subclass.

ii. If there is more than one referent, we create a new composite object whose constituents are these referents and mark the relation between this object and each of the referents as class-subclass. This object is then returned as the object identified by the definite NP. For instance, in

(4.103) a. Soak, drain and rinse the celery.

b. Chop the onions.

c. Melt the butter in a large saucepan and add *the vegetables.*

we create a new vegetable object with the celery object and the onion object as constituents.

The search through the DM at step (2) for the referents of a definite NP can be formalised again as a CSP. In this CSP, lexical disambiguation of the noun in the NP is carried out as well. To set up this CSP, we do the following:

Variables We assign a variable each to the noun in the NP and the NP itself. For instance, if the NP is *the chopped carrot*, a variable is assigned each to the noun *carrot* and to the NP *the chopped carrot.*

Domain The domain for the variable assigned to the noun is the set of possible object substances which the noun can describe. In the case where the noun is lexically ambiguous, the set will consist of more than one element. For instance, the noun *dough* in *the dough* is lexically ambiguous between an uncooked pastry and money. The domain for the variable assigned to the NP is the DM itself.

Constraints The constraints come from the NP itself and also from the type of links that can exist between the object described by the NP and its referent.

We will now give the CSP set up when the link type hypothesized is IDENTITY, WHOLE-PART, CLASS-SUBCLASS and RESOURCE. In the discussion that follows, we let *subst* and *referent* be the variables assigned to the noun and NP respectively and DM to be the discourse model at the time of resolving the NP.

Identity We consider how to search for the referents of the NP *the apples* in example (4.98):

(4.98) a. Wash and core six cooking apples.
b. Put *the apples* into the oven dish.

The CSP set up for the search is:

(4.104) $(\exists\ subst)\ (\exists\ referent)$
$(subst \in \{\text{apple-matter}\})\ (referent \in \text{DM})$
$\text{substance}(referent,\ subst) \wedge \text{structure}(referent,\ \text{set})$

The constraints given essentially say that the possible referents for *the apples* are those objects in DM which are made of apple-matter and which are viewed as a set.

Whole-Part Consider how the referents for the definite NP *the yolk* in example (4.99) are determined:

(4.99) a. Break an egg.
b. Beat *the yolk* and the white so that they hold a peak.

We set up the following CSP:

(4.105) $(\exists\ subst)\ (\exists\ referent)$
$(subst \in \{\text{yolk-matter}\})\ (referent \in \text{DM})$
part-of(*subst*, substance-of(*referent*))

where the function *substance-of* returns the substance which *referent* consists of while the predicate *part-of* is true if *subst* is part of the substance of *referent*. This predicate can be directly verified by the taxonomic hierarchy of substances.

Class-Subclass Consider the referent of the definite NP *the vegetables* in the following example:

(4.103) a. Soak, drain and rinse the butter beans.
b. Peel and chop the onion.
c. Melt the butter in a large saucepan and add *the vegetables.*

The CSP sets up is as follows:

(4.106) $(\exists\ subst)\ (\exists\ referent)$
$(subst \in \{\text{vegetable-matter}\})\ (referent \in \text{DM})$
subclass-of(substance-of(*referent*), *subst*)

where *subclass-of* is a predicate which is true if the substance of *referent* is a subclass of *subst.*

Resource Consider how the referents of the NP *the heat* are determined in

(4.107) a. e_1: Bring the soup to the boil.
b. Reduce *the heat.*

The CSP set up is as follows:

(4.108) $(\exists\ subst)\ (\exists\ referent)$
$(subst \in \{\text{heat-matter}\})\ (referent \in \text{DM})$
event(*referent*) $\wedge$ resource-required(*subst*, *referent*)

where the constraints state that the referent must be an event and that *subst* is a resource required in the event.

Solving a CSP

There are many methods of solving a CSP. The most straightforward method is the generate-and-test strategy. In this strategy, one computes the cartesian product D of all the given value domains, i.e.

(4.109) $D = D_1 \times D_2 \times \ldots D_n$

and then tests each of the n-tuples in D to see whether they satisfy the given constraints. This basic search strategy can be easily improved using techniques like backtracking. A backtracking algorithm explores the product space D by sequentially instantiating variables in the constraints list. Once a constraint is fully-instantiated, its truth value is determined: if it is false, the last variable with untried values in its domain is instantiated with another value.

Another method is to use network consistency techniques which regard the problem as one of elimination rather than search. The goal is to refine the value domains as much as possible so that values which cannot be part of a global solution to the problem are eliminated. Such techniques are applied in [Mellish 85, Haddock 88] for solving the CSP associated with noun phrase resolution and shown to be useful for incremental interpretation.

In our work, we essentially employ the generate-and-test approach for solving the derived CSPs. This is because the investigation of algorithms for CSPs is not within the scope of the current work.

The Algorithm

Based on the discussion above, the algorithm we employ for disambiguating the verb and resolving the referring expressions in a clause is given in figure 4.10.

```
Algorithm : RESOLVE-OBJ
Input     : V, $NP_1 \ldots NP_n$
Output    : the action indicated by V and the referents of the $NP_i$.

  1. Set up a CSP as follows:
     (a) Let *action* be the variable corresponding to V
     (b) Let $ref_i$ be the variables assigned to $NP_i$
     (c) Let the domain of *action* be those actions which V describes.
     (d) Determine the domain of each $ref_i$:
         i.  If $NP_i$ is a pronoun or ellided NP, then let the domain of $ref_i$ be the set
             consisting of the current center.
         ii. If $NP_i$ is a definite NP, then let the domain of $ref_i$ be the set obtained
             by calling FIND-NP-DOM($NP_i$).
     (e) Let constraints be the constraints arising from any pronouns and constraints
         from V.
  2. Solve the CSP.
  3. Return the value bound to *action* and each of the $ref_i$.
```

Figure 4.10: The object resolution algorithm

4.3.2 Resolution to Situations

In terms of the algorithm in figure 4.2, we resolve a clause to either a state or event via the function BUILD-SITUATION-REP. If the situation is a new one, it is added to the discourse model. The steps involved in this process are as follows:

1. If the clause contains a verb be+VERB and the tense is not progressive as in *the water is cooled*, then we take it as describing a state. We resolve for the object which the definite NP describes (by calling the FIND-NP-DOM function) and search for a state of the object which contains the predicate given by VERB.

2. If the clause contains a verb in present progressive form (v_{prog}) as in *the water is cooling* or *the water is boiling*, we do the following:

   (a) We take v_{prog} as a modifier of the NP in the clause. We can think of this as equivalent to forming the referring expression (the v_{prog} NP).

   (b) We attempt to find a referent in DM that has the v_{prog} property. The idea here is that if such a referent exists, then the v_{prog} property must have been added to the referent by the simulation of a previously mentioned event. In terms of the STRIPS representation, v_{prog} will be a postcondition of the operator corresponding to the action in the event. This means that the clause describes the consequent state of the previously mentioned event.

   Consider for instance

   (4.110) a. e_1: Bring the water to the boil.

   b. When *the water is boiling*, put in the meat.

   In our work, we represent the bring-to-boil operator corresponding to the action in e_1 as follows:

(4.111)	Action	: Do(Agent, BRING-TO-BOIL(x))
	Precondition	: boiling(x, −)
	Postcondition	: boiling(x, +), hot(x, +)

We use the *boiling* predicate in (4.111) since it is plausible to describe water which is brought to the boil as ***the boiling water.***

This means that after simulating e_1, the water object (x_1) will have the boiling property added. Then, we process the *when*-clause *the water is boiling* by attempting to find a referent for *the boiling water.* This succeeds with x_1 and therefore we regard the clause as describing a state (s) in which the following proposition holds:

(4.112) hold(s, boiling(x_1, +))

On the other hand, consider what happens if the example is changed to

(4.113) a. e_1: Cool the water.

b. When *the water is cooling*, put in the meat.

The cooling operator is represented as follows:

(4.114) Action	Do(Agent, COOL(x))
Precondition	: hot(x, +)
Postcondition	: cooled(x, +)

Note that unlike (4.111), we do not use a *cooling* predicate to describe the water that has cooled. This is because it is not plausible to describe the water that has cooled as *the cooling water.*

Thus, after simulating e_1, the water object (x_1) will have the cooled(x, +) property. This means that no referent will be found for the NP *the cooling water* since there exists no water object with the *cooling* property. Thus, we take the *when*-clause as not describing a state. It is then taken to be describing an event.

3. Otherwise, the clause is describing an event and we do the following:

   (a) We carry out lexical disambiguation of the verb and reference resolution of the referring expressions in the clause using the algorithm given in figure 4.10. This step essentially works out the action and participants in the instruction.

(b) We search in the DM for an event with the same action and participants. This is treated as another CSP. Assuming that the action is *action* and the participants are *participants*, then the CSP is:

(4.115) ($\exists$ *referent*) (*referent* $\in$ DM)
event(*referent*) $\wedge$ substance(*referent*, *action*) $\wedge$
same-participants(*referent*, *participants*)

where the constraints say that the referent must be an event with the same action and the same participants.

(c) If there are no referents in DM which satisfy the constraints of the CSP in (4.115), then the clause is describing a new event. We create a new event and add to the DM.

(d) Otherwise, there exists a referent event e and we determine the aspect of this event which the clause is describing. Here, we make use of the work of [Song 91b, Singh & Singh 92] as described in chapter 2. The basic idea is to use the presence or absence of perfect, *for*-adverbial and progressive to determine the aspect:

(4.116)

Linguistic Features	Aspect
Perfect + For-Adverbial	Neutral Perfective
Perfect	Perfect
Progressive	Progressive

Consider, for instance the following examples:

(4.117) e_1: Cool the water.

a. When *the water has cooled for 10 minutes*, take the meat out.

b. When *the water has cooled*, take the meat out.

c. When *the water is cooling*, take the meat out.

where the *when*-clause in (4.117a) is describing the first 10 minutes of e_1 (neutral-perfective aspect), the *when*-clause in (4.117b) is describing the end state of e_1 (perfect aspect) and the *when*-clause in (4.117c) is describing the preliminary processes leading up to the end point of e_1 (progressive aspect).

Note that in such cases, we do not create a new event for the current clause. Instead, we simply record the aspectual information derived.

The Algorithm

Based on the discussion above, the algorithm for resolving a clause into either a state or event is given in figure 4.11. The input to the algorithm is some intermediate representation of the clause.

4.4 Decomposing Events in QTS

Recall from chapter 3 that in our work, we take a sentence as being viewed as describing a simple or complex event (i.e. either as a repetition or composite event). This does not preclude the possibility that as more information comes to light, the event can be decomposed further. Such is the case especially for sentences with plural NPs (example (4.118a)) and sentences describing repetitive events (example (4.118b)).

(4.118) a. Soak *some carrots and potatoes.*

b. Stir occasionally.

In this section, we consider the problem of decomposing an event if it is deemed necessary. To do this, we discuss how the event corresponding to *stir occasionally* can be decomposed in the QTS corresponding to example (4.119):

(4.119) a. Bring the soup to the boil stirring occasionally.

b. Meanwhile, chop a carrot.

The QTS of example (4.119) is as follows:

(4.120) $\text{QTS}_3 = (\{e_1, e_2, e_3\},$
$\{\langle \text{simultaneous}, e_2, e_1\rangle, \langle \text{constituent}, e_3, e_1\rangle,$
$\langle \text{constituent}, e_3, e_2\rangle, \langle \text{included-in}, e_4, e_3\rangle, \langle \text{none}, e_4, e_2\rangle\})$

```
Algorithm : BUILD-SITUATION-REP
Input     : INT-REP
Output    : a new situation and any durational info or a given event + aspect

  1. If the verb in INT-REP contains a verb-to-be, then
     (a) Let REF = find-np-dom(the NP in INT-REP)
     (b) Let STATE = Search for a state of REF with the verb as predicate.
     (c) Add STATE to DM and return [STATE, ∅]
  2. If the verb in INT-REP is in present progressive form, then
     (a) Let AELEM = Conjoin the verb and the NP in INT-REP
     (b) Let REF = find-np-dom(AELEM)
     (c) If REF is not ∅, then
          i. Let STATE = the state which REF is in
         ii. Add STATE to DM and return [STATE, ∅]
  3. Otherwise, do
     (a) Let V be the verb in INT-REP
         Let $NP_1 \ldots NP_n$ be the NPs in INT-REP
         Let TENSE be the tense in INT-REP
         Let FOR-ADV be the for-adverbial in INT-REP
     (b) Let [ACTION, PARTICIPANTS] = resolve-obj(V, $NP_1 \ldots NP_n$)
     (c) Let EVENT = Search for an event in DM with the same
                     ACTION and PARTICIPANTS
     (d) If EVENT is ∅, then
           i. Let NEW = Create a new event from ACTION and PARTICIPANTS.
          ii. Add event to DM.
         iii. If FOR-ADV is not ∅, return [NEW, [duration, info in the for-adverbial]]
              Otherwise, return [NEW, [duration, ∅]]
     (e) Otherwise, do
           i. Determine ASPECT:
              A. If TENSE is perfect and FOR-ADV is not ∅, then assign ASPECT to be
                 neutral-perfective.
              B. If TENSE is perfect and FOR-ADV is ∅, then assign ASPECT to be
                 perfect.
              C. If TENSE is progressive, then assign ASPECT to be progressive.
          ii. Return [EVENT, ASPECT]
```

Figure 4.11: The situation resolution algorithm

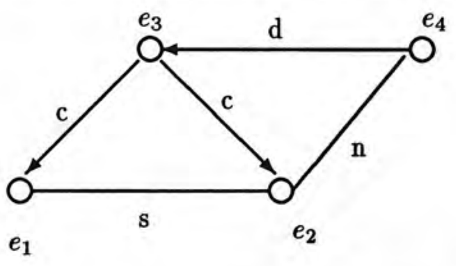


Figure 4.12: The QTS of example (4.119)

Graphically, we can represent the QTS in (4.120) in figure 4.12. where

- e_3 is the composite event of *bring the soup to the boil stirring occasionally*,
- e_1 is the event of *bring the soup to the boil*,
- e_2 is the event of *stir occasionally*,
- e_4 is the event of *chop a carrot* and
- the abbreviation of the **constituent**, **simultaneous**, **none** and **included-in** relation is c, s, n and d respectively.

Recall that the event representation corresponding to *stir occasionally* is

(4.121) $\exists e$ event(e) $\wedge$ structure(e, repetition) $\wedge$ occur(s_0, s_1, e) $\wedge$
frequency(e, occasionally) $\wedge$
[$\forall f$ element(e, f) $\supset$
event(f) $\wedge$ structure(f, simple) $\wedge$
telic(f, $-$) $\wedge$ substance(f, stirring-matter) $\wedge$
agent(f, hearer) $\wedge$ obj(f, x_1)]

where x_1 is the soup object.

The first question that has to be solved is to determine the number of individual stirring events. Following [Karlin 88], this calculation is carried out as follows:

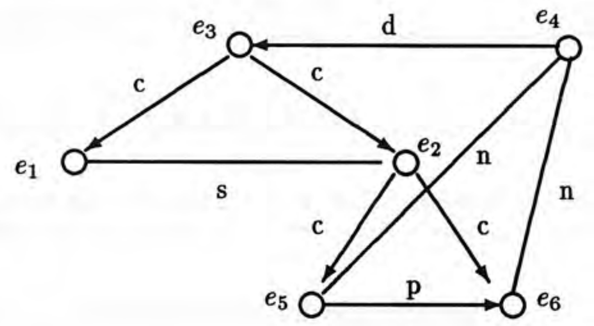


Figure 4.13: The QTS after decomposing e_2, the event described by *stirring occasionally* in figure 4.12

1. Assume that the intervals between repetitions are equal. The length of this interval (LEN) is calculated by using a percentage value stored in the lexicon with the frequency adverbial such that LEN = PERCENTAGE x TOTAL-TIME. The TOTAL-TIME is the time for the whole event of *stirring occasionally* which in our present example is equal to the duration of the *boiling some soup* event. For the adverbial *occasionally*, the percentage is set arbitrarily at twenty five percent while for adverbials like *frequently*, the percentage will be set at a lower value.

2. The number of individual stirring events (N) is then obtained from the following formula: N = TOTAL-TIME / (LEN + Duration of an individual stirring event).

The above calculation is necessarily only an approximation of the real world. Suppose that, based on the above calculation, there are two individual stirring events. The representation in (4.121) is changed to

(4.122) $\exists e$ event(e) $\wedge$ structure(e, repetition) $\wedge$ occur(s_0, s_1, e) $\wedge$
constituents(e, [e_1, e_2]) $\wedge$
[event(e_1) $\wedge$ structure(e_1, simple) $\wedge$
telic(e_1, $-$) $\wedge$ substance(e_1, stirring-matter) $\wedge$
agent(e_1, hearer) $\wedge$ obj(e_1, x_1)] $\wedge$
[event(e_2) $\wedge$ structure(e_2, simple) $\wedge$

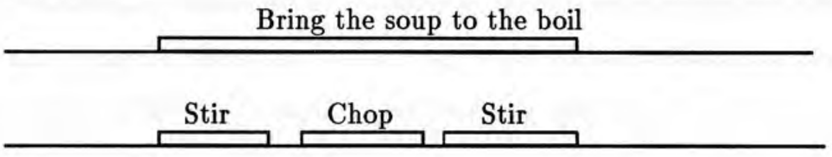


Figure 4.14: The temporal structure of example (4.119)

$$\text{telic}(e_2, -) \wedge \text{substance}(e_2, \text{stirring-matter}) \wedge$$
$$\text{agent}(e_2, \text{hearer}) \wedge \text{obj}(e_2, x_1)\ ] \wedge$$
$$\text{precedes}(e_1, e_2)$$

The QTS is updated accordingly. One updating involves altering the link to the decomposed event to ensure that the sub-events are integrated in the QTS :

1. For each event in QTS which is either **included-in** or has the **none** relation with the decomposed event, do
2. (a) Determine the relation between the event and each of the decompositions.
   (b) Remove the relation between the event and the decomposed event from the QTS .

The final QTS is as in figure (4.13) where e_5 and e_6 are the two individual stirring events in e_2.

4.5 Conclusion and Summary

In this chapter, we have presented a new approach to the problem of integrating the current situation into the structure built up for the preceding text. In this approach, we find the set of reference events needed for a current situation to be integrated in the QTS. There may be more than one reference event needed. The role which these reference events play in our TA2 framework is explained and two heuristics for choosing them are given: the similarity criterion and the temporally-further criterion.

We have also presented an approach to the reference resolution of objects and identified a new way in which new objects may be added to the discourse model, namely through event simulation. In addition to resolving references to objects, we also presented an approach to resolving references to situations. In resolving references to situations, we make use of the internal structure of event and employ aspectual information to determine the part of this event structure which the current clause is describing.

In the next chapter, we examine the issue of how the temporal structure can be extracted from the QTS . For example, from the QTS of example (4.119) (see figure 4.13) reproduced below:

(4.119) a. Bring the soup to the boil stirring occasionally.

b. Meanwhile, chop a carrot.

we are able to extract the temporal structure in figure 4.14.

Chapter 5

Temporal Structure

Recall from chapter 1 that there are two levels of representation in TA2. A text is first processed into a reduced, fully-integrated QTS which encodes the situations mentioned in the text and the relations between a situation and its reference events. We next regard each situation as occurring over a unique time interval and work out the temporal structure by determining the relations between the time intervals associated with the events and states. More formally, we can define a temporal structure as follows:

> The temporal structure of a text is defined as a total order on the start and end time points of the time intervals associated with the situations mentioned in the text.

As noted in chapter 3, there are two aspects in the relations between time intervals. First, we have the qualitative relations between time intervals which are represented using the thirteen relations in Interval Algebra (see figure 2.5). For ease of exposition, we will refer to these qualitative relations as IA relations. Second, we have quantitative relations between the start and end time points of the time intervals which indicate the extent to which time intervals are separated from one another.

Consider, for instance, how the following simple text is processed:

(5.1) Just before e_1: serving, e_2: add the wine.

The QTS corresponding to example (5.1) is:

(5.2) $(\{e_1, e_2\}, \{\langle \text{precedes}, e_2, e_1 \rangle\})$

The **precedes** relation between e_2 and e_1 is obtained from the semantics of the *before* connective, at the QTS stage, in example (5.1). Note that we do not derive the relation that e_2 **meets** e_1 although it appears that this is indicated by the phrase *just before.* This is because we regard the word *just* as an indicator of the quantitative relation between the end time point of te_2 (the time interval assigned to e_2) and the start time point of te_1 (the time interval assigned to e_1):

(5.3) $\text{start}(te_1) - \text{end}(te_2) = 0$

The quantitative relation given in (5.3) is then combined with the qualitative relations between te_2 and te_1 derived from the **precede** relation between e_2 and e_1 (see chapter 3):

(5.4) e_2 **precedes** $e_1 \equiv te_2 < te_1 \vee te_2 \; m \; te_1 \equiv te_2 \; \{<, m\} \; te_1$

where $<$ and m are two of the possible IA relations between time intervals. Note that the single **precedes** relation between e_2 and e_1 is vague between whether the time interval associated with e_2 is before ($<$) or meets (m) the time interval associated with e_1. We represent this vagueness by saying that there is a set of possible IA relations between te_2 and te_1 (i.e. $te_2 \; \{<,m\} \; te_1$).

The combination of qualitative and quantitative information is carried out through a process of constraint propagation. In the case of example (5.1), this process will eliminate $<$ from consideration and determine that $te_2 \; m \; te_1$. This is then the temporal structure for example (5.1) in which the orderings among the start and end points of te_1 and te_2 are as follows:

(5.5) $te_1 \; m \; te_2 \equiv \text{end}(te_1) = \text{start}(te_2) \wedge$

$$\begin{aligned}&\text{start}(te_1) < \text{end}(te_1) \wedge\\&\text{start}(te_2) < \text{end}(te_2)\end{aligned}$$

Recall from chapter 3 that START and END are functions which return respectively the start and end time points of a time interval.

On the other hand, if example (5.1) is changed to

(5.6) **30 minutes before e_1: serving, e_2: add the wine.**

then the qualitative relation between te_2 and te_1 is as before while the quantitative relation is changed to:

(5.7) $\text{start}(te_1) - \text{end}(te_2) = 30$

In this case, constraint propagation will eliminate m from consideration and determine that $te_2 < te_1$. This is then the temporal structure for example (5.6) where the ordering among the start and end points of te_1 and te_2 are as follows:

(5.8) $$\begin{aligned}te_1 < te_2 \equiv\ &\text{end}(te_1) < \text{start}(te_2) \wedge\\&\text{start}(te_1) < \text{end}(te_1) \wedge\\&\text{start}(te_2) < \text{end}(te_2)\end{aligned}$$

Note that the temporal structure of a text contains the time intervals of all the situations mentioned in the text. Given the temporal structure of a cookery recipe, we can easily extract a schedule specifying the actions which an agent has to carry out:

1. We remove all time intervals from the temporal structure corresponding to states.
2. We remove all time intervals from the temporal structure corresponding to composite events. This is because the agent only need to carry out the sub-events of a composite event.
3. We remove all time intervals from the temporal structure corresponding to events which are generated by other events. Recall from chapter 3 that if an event e

is generated by another event f, then the agent needs only to perform f as this simultaneously achieves e.

Notice that the above extraction steps require access to information about the relations between situations kept in the QTS.

The rest of this chapter is organised as follows:

In section 5.1, we discuss how qualitative relations between time intervals and quantitative relations between the start and end time points of time intervals can be derived. The qualitative relations are derived from the relations between situations in the QTS and may differ in different environments. The kind of environmental information we consider is the resource requirement of the events. This means that different temporal structures may be obtained from the same QTS in different environments.

In section 5.2, we show how the set of qualitative and quantitative relations extracted can be viewed as constituting a constraint satisfaction problem (CSP) (Recall also that in chapter 4, we discuss how reference resolution and lexical disambiguation can be solved by viewing them as a CSP). The CSP can be solved using algorithms developed in the field of temporal reasoning. We analyse the requirements posed by our CSP on temporal reasoning and based on this analysis choose an appropriate approach for solving the CSP.

In section 5.3, we work through a simple example to demonstrate how different temporal structures can be obtained from the same QTS in different environments.

Finally, section 5.4 ends with a conclusion and summary.

5.1 Relations Between Time Intervals

The first step in extracting the temporal structure is to derive the qualitative and quantitative relations that can hold between the time intervals associated with the situations in the QTS. The relations can be qualitative, indicating the relative ordering of the time intervals with respect to one another. Other relations are quantitative,

sit_1 RELATION sit_2	t_{sit_1} {ALLEN RELATIONS} t_{sit_2}
sit_1 **precedes** sit_2	t_{sit_1} $\{<, m\}$ t_{sit_2}
sit_1 **after** sit_2	t_{sit_1} $\{>, mi\}$ t_{sit_2}
sit_1 **includes** sit_2	t_{sit_1} $\{si, fi, di\}$ t_{sit_2}
sit_1 **included-in** sit_2	t_{sit_1} $\{s, f, d\}$ t_{sit_2}
sit_1 **meets** sit_2	t_{sit_1} $\{m\}$ t_{sit_2}
sit_1 **met-by** sit_2	t_{sit_1} $\{mi\}$ t_{sit_2}
sit_1 **simultaneous** sit_2	t_{sit_1} $\{=\}$ t_{sit_2}
sit_1 **enables** sit_2	t_{sit_1} $\{<, m\}$ t_{sit_2}
sit_1 **enabled-by** sit_2	t_{sit_1} $\{>, mi\}$ t_{sit_2}
sit_1 **generates** sit_2	t_{sit_1} $\{=\}$ t_{sit_2}
sit_1 **generated-by** sit_2	t_{sit_1} $\{=\}$ t_{sit_2}

Table 5.1: The relations between situations and their time intervals

indicating the extent to which time intervals are separated from one another. Quantitative relations are typically expressed between the start and end time points of time intervals.

5.1.1 Qualitative Relations Between Time Intervals

The qualitative relations between time intervals are expressed using the thirteen relations in Interval Algebra (IA) [Allen 83]. These relations between time intervals are derived from the relations between situations (identified by the time intervals) in the QTS. Recall from chapter 3 that we employ seven relations between situations, namely **precede, include, meet, simultaneous, constituent, enable, generate**, and the default **none** relation. In addition, we have discussed the correspondence between these eight relations and the IA relations. This correspondence for all but the **constituent** and **none** relation is given in table 3.2 and reproduced in table 5.1.

An approach to obtaining the set of possible IA relations between the time intervals assigned to the situations in the QTS is then to iteratively carry out the following steps until there are no more relations to be picked in the QTS:

1. Pick a relation from the QTS.
2. Derive the set of IA relations that corresponds to the picked relation.
3. Repeat step (1).

At step (2) of the above approach, the derivation is simply a lookup from table 5.1 for the **precedes, includes, meets, simultaneous, enables** and **generates** relation. However, this is not so in the case of the **constituent** and **none** relation. We will now discuss these two cases.

Reasoning About the NONE relation

Given a **none** relation between two events, the temporal relation between the time intervals associated with the events may differ under different assumptions of resource availability. In our work, we consider resources to be 'things' which are required to bring an event to completion. For instance, in bringing a chopping event underlying a clause like *chop a carrot* to completion, one would need the availability of instruments like a chopping knife or a chopping machine. In addition, the agent may or may not be needed to bring an event to completion. For example, in an environment where an agent only has access to a chopping knife, then the agent needs to be involved in the chopping action. On the other hand, if the agent has access to a chopping machine, then the agent needs only to perform some initial steps of starting the machine after which the chopping action can be left on its own to completion.

Depending on the environment at the time of understanding the instructions in (5.9):

(5.9) a. e_1: Chop the carrot.

b. e_2: Chop the potato.

where e_1 has a **none** relation with e_2, there can be the following kinds of scenarios:

- If the environment has only one agent or two agents with one chopping knife, there is no way in which e_1 and e_2 can be carried out simultaneously.
- If the environment has two agents and two chopping knives, then e_1 and e_2 can be performed simultaneously.
- If the environment has only one agent but one chopping knife and a chopping machine, then the agent could use the chopping machine to carry out e_1 and while e_1 is being carried out to use the chopping knife for carrying out e_2 or vice versa. If we ignore the initial steps needed to start the chopping machine, we can say that e_1 and e_2 are carried out simultaneously.

Based on the discussion above, we invoke information about resource contention to decide on the set of possible IA relations corresponding to a **none** relation. This can be done using rules. The two rules used are:

(5.10) IF none(e_1, e_2)
AND $\neg$ resource-contention(e_1, e_2)
THEN te_1 $\{<, >, m, mi, o, oi, =, s, si, f, fi, d, di\}$ te_2

(5.11) IF none(e_1, e_2)
AND resource-contention(e_1, e_2)
THEN te_1 $\{<, >, m, mi\}$ te_2

where the first rule says that if there is no resource contention between an event e_1 and another event e_2, then there is no ordering between the time intervals of the two events, and the second rule says that if the two events have a resource contention, then their time intervals must not overlap.

How do we determine whether there exists a resource contention between two events? To do this, we employ rules such as those below:

(5.12) IF there is only one agent
AND e_1 requires the agent to bring it to completion

AND e_2 requires the agent to bring it to completion

THEN resource-contention(e_1, e_2)

(5.13) IF there is only one agent

AND e_1 require the agent to bring it to completion

AND e_2 does not require the agent to bring it to completion

THEN $\neg$ resource-contention(e_1, e_2)

Notice that the clauses on the left-hand side of rules (5.12) and (5.13) need access to environment information such as the number of agents available and the need for an agent to be involved in bringing an event to completion.

Consider example (5.9) again:

(5.9) a. e_1: Chop the carrot.

b. e_2: Chop the potato.

We consider an environment where there is only one agent, one chopping knife and no chopping machine. We can represent this environment as a set of the following predicates:

(5.14) number-of-agent(1)
number-of-chopping-knife(1)
number-of-chopping-machine(0)

In this environment, the agent is needed to bring both e_1 and e_2 to completion since no chopping machine is available. Rule (5.12) is therefore applicable in this environment which means that there is a resource contention between e_2 and e_1. Thus, rule (5.11) fires and the set of possible IA relations between te_1 and te_2 is $\{<, >, m, mi\}$.

Consider another environment where there is only one agent, one chopping knife and one chopping machine:

(5.15) number-of-agent(1)

number-of-chopping-knife(1)
number-of-chopping-machine(1)

In this environment, the agent may use the chopping machine to chop the carrot in e_1 and thus she/he is not required to bring e_1 to completion. While the chopping machine is doing its work, the agent may use the chopping knife to chop the potato in e_2 and in this case, the agent is needed to bring e_2 to completion. Rule (5.13) is applicable which means that there is no resource contention between e_1 and e_2. This in turn fires rule (5.10) and the set of possible IA relations between te_1 and te_2 is {<, >, *m*, *mi*, *o*, *oi*, =, *s*, *si*, *f*, *fi*, *d*, *di*}.

From the above discussion, it is clear that in different environments, a **none** relation between two events can give rise to different sets of IA relations between the time intervals over which the events occur. As will be shown in section 5.3, this can give rise to different temporal structures for the same text in different environments.

Reasoning About CONSTITUENCY

Recall from chapter 3 that the **constituent** relation is a relation between a composite event and its sub-event. That is, given a composite event c and its sub-event $c_1 \ldots c_n$, we use

(5.16) constituent(c, c_i) $1 \leq i \leq n$

to denote that c_i is related to c by a **constituent** relation. For instance, the instruction in

(5.17) Bring the soup to the boil stirring occasionally.

describes a composite event e_3 whose constituents are the bring-to-boil event (e_1) and the stirring event (e_2). The relations between e_3 and each of e_1 and e_2 are then:

(5.18) a. constituent(e_3, e_1)

b. constituent(e_3, e_2)

Now, if an event c_1 is a constituent of c, then c_1 is temporally included in c. In the case of example (5.17), we have the following set of IA relations between te_3 and te_1 and between te_3 and te_2:

(5.19) a. te_3 $\{si, fi, di, =\}$ te_1

b. te_3 $\{si, fi, di, =\}$ te_2

However, the above correspondence fails to take into account the temporal relations among the various constituents or sub-events of the composite event. For instance, in example (5.17), the two sub-events of e_3 (i.e. e_2 and e_1) are simultaneous with one another. This actually constrains e_3 to be simultaneous with both e_2 and e_1. That is, we obtain the more 'constrained' IA relations between te_3 and each of te_2 and te_1 than those given in (5.19) above:

(5.20) te_3 $\{=\}$ te_1

te_3 $\{=\}$ te_2

As another example, consider the following instruction:

(5.21) Soak, drain and rinse the beans.

which describes a composite event e_4 whose sub-events are soaking (e_1), draining (e_2) and rinsing (e_3). The relations among the e_i are inferred to be **enablement** (see section 4.2.4) so that e_1 must occur before e_2 and e_2 before e_3. Since e_4 includes all and only these sub-events, it must be the case that e_1 starts e_4, e_2 is during e_4 and e_3 finishes e_4. That is, we have the following relations between the time interval associated with e_4 (te_4) and that associated with its sub-events e_1 (te_1), e_2 (te_2) and e_3 (te_3):

(5.22) te_4 $\{si\}$ te_1

te_4 $\{di\}$ te_2

te_4 $\{fi\}$ te_3

Given the discussion above, an approach for determining the possible IA relations between the time interval of a composite event and those of its sub-events must take into account the possible IA relations among the time intervals associated with the sub-events. One such approach can be found in [Song & Cohen 91a]. In our work, we make use of this approach which we will discuss now.

Song and Cohen's Approach

We can divide the discussion into two cases: one in which a composite event has only two sub-events and the other where the composite event has more than two sub-events.

Consider first the case where we have a composite event e (time interval te) with two sub-events e_1 (time interval te_1) and e_2 (time interval te_2). There are two possible sub-cases:

1. In the first sub-case, the IA relation between te_1 and te_2 is ONE of the thirteen IA relations. There are five cases to consider:

   (a) In the first case, the relation between te_1 and te_2 is such that te_1 is either *before*, *meets* or *overlaps* te_2. This means that the sub-events are arranged in some sequence within the composite event. Thus, it must be the case that the first event in the sequence *starts* the composite event and the last event in the sequence *finishes* it. Any events in between the first and the last event must be necessarily *during* the composite event:

   (5.23) $te\ \{si\}\ te_1$
   $te\ \{fi\}\ te_2$

   (b) The second case is the converse of the first case. That is, the relation between te_1 and te_2 is such that te_1 is either *after*, *met by* or *overlapped by* te_2. The same reasoning as in the first case applies except that we now have:

   (5.24) $te\ \{fi\}\ te_1$
   $te\ \{si\}\ te_2$

(c) In the third case, te_1 is either *during*, *starts* or *finishes* te_2. Intuitively, this means that te_1 is contained within te_2. Since te_1 and te_2 are both contained within te, it must be the case that te is simultaneous with te_2 and that te contains te_1:

(5.25) $te\ \{si,di,fi\}\ te_1$
$te\ \{=\}\ te_2$

(d) The fourth case is the converse of the third case so that te_2 is either *during*, *starts* or *finishes* te_1. The same reasoning as in the third case applies except that we now have:

(5.26) $te\ \{=\}\ te_1$
$te\ \{si,di,fi\}\ te_2$

(e) The final case is when te_2 is *equal* with te_1. Since both te_1 and te_2 are within te, it must be the case that te is *equal* with te_1 and with te_2:

(5.27) $te\ \{=\}\ te_1$
$te\ \{=\}\ te_2$

The five cases discussed above are shown graphically in figure 5.1.

2. In the second sub-case, there is a set of more than one possible IA relation between te_1 and te_2. In this case, the possible IA relations between te and te_1 and between te and te_2 are the union of the appropriate results read off from figure 5.1. Consider, for instance, what are the possible IA relations between te and te_1 and between te and te_2 if the set of possible IA relations between te_1 and te_2 is as follows:

(5.28) $te_1\ \{<,>\}\ te_2$

These relations can be computed as follows:

(a) First, recall that $te_1\ \{<, >\}\ te_2 \equiv (te_1\ \{<\}\ te_2) \vee (te_1\ \{>\}\ te_2)$.

(b) From $te_1\ \{<\}\ te_2$, we know that $te\ \{si\}\ te_1$ and $te\ \{fi\}\ te_2$. Similarly, from $te_1\ \{>\}\ te_2$, we know that $te\ \{fi\}\ te_1$ and $te\ \{si\}\ te_2$. Combining the two results by taking the disjunction of the corresponding relations, we obtain the following:

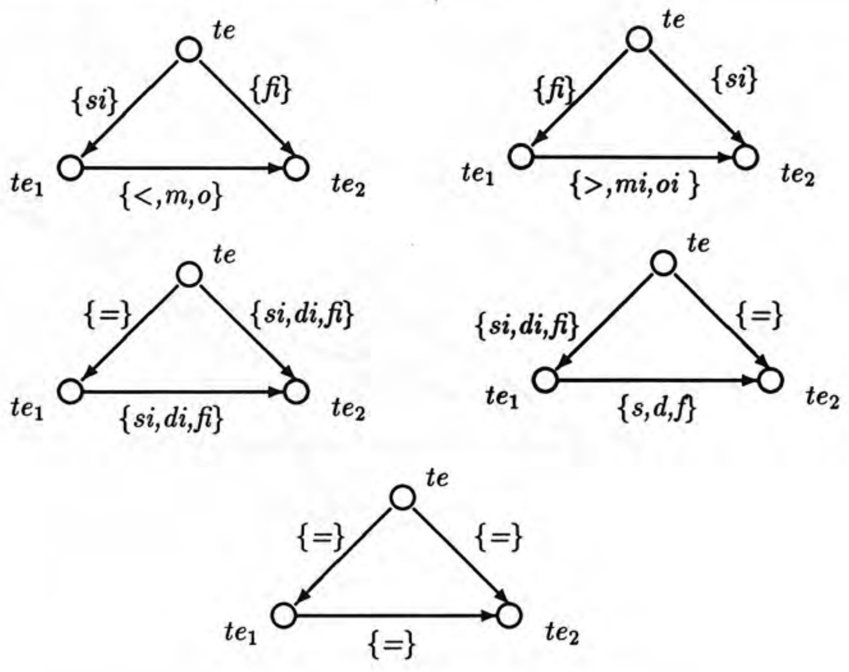


Figure 5.1: The 5 cases

i. te $\{si, fi\}$ te_1

ii. te $\{si, fi\}$ te_2

Next, we consider how the approach described above for a composite event with two sub-events (for ease of exposition, we call this approach SUB2) can be extended to a composite event with more than two sub-events. The strategy is to use SUB2 to determine the set of possible IA relations between the composite event and two sub-events at a time. One such way of carrying out this strategy can be illustrated using a composite event e (te) with three sub-events e_1 (te_1), e_2 (te_2) and e_3 (te_3) (see figure 5.2a):

1. Introduce an intermediate composite event e_{12} (te_{12}) that has as its constituents e_1 and e_2 (see figure 5.2b).

2. Apply SUB2 to obtain the set of possible IA relations between te_{12} and te_1 and between te_{12} and te_2.

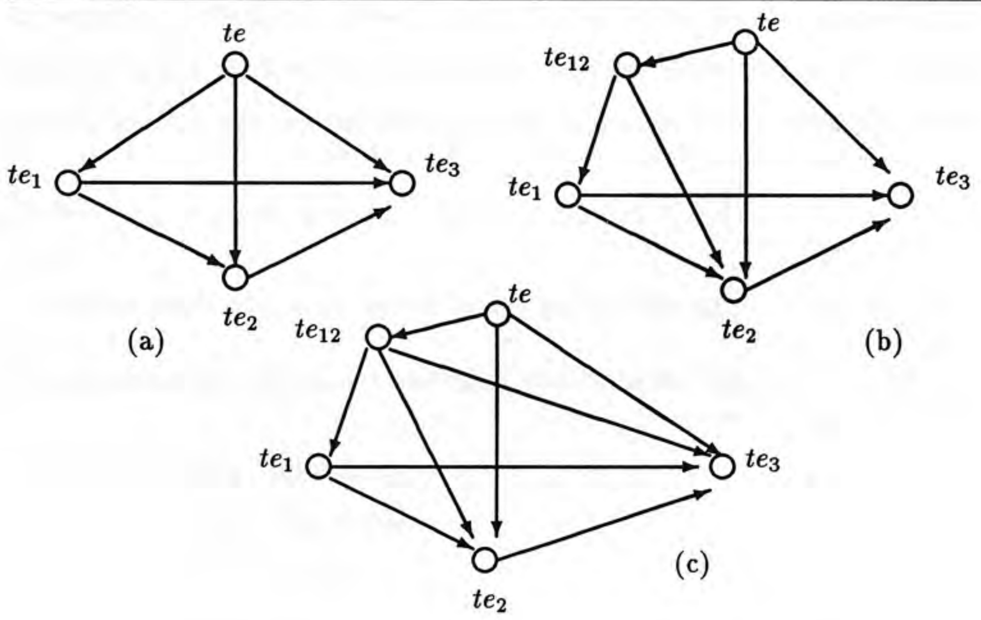


Figure 5.2: An event with three sub-events

3. Compute the set of possible IA relations between te_{12} and te_3 from the set of possible IA relations between the time intervals on the two chains: ($te_{12} \rightarrow te_1 \rightarrow te_3$) and ($te_{12} \rightarrow te_2 \rightarrow te_3$).

4. Now, from figure 5.2c, e_{12} and e_3 form two sub-events of e. Apply SUB2 to obtain the set of possible IA relations between te and te_{12} and between te and te_3.

5. Compute the set of possible IA relations between te and te_1 from the set of possible IA relations between the time intervals on the chain: ($te \rightarrow te_{12} \rightarrow te_1$). Similarly, compute the set of possible IA relations between te and te_2 from the set of possible IA relations between the time intervals on the chain: ($te \rightarrow te_{12} \rightarrow te_2$).

6. Remove te_{12} and all its links with te_1, te_2 and te_3. We are left with the set of IA relations between te and each of te_1, te_2 and te_3.

We now consider how step (3) and (5) are carried out. Given a chain ($t_a \rightarrow t_b \rightarrow t_c$), the set of possible IA relations between t_a and t_c can be computed from that between t_a and t_b and that between t_b and t_c. For ease of exposition, we let r_{ab} be the set of possible IA relations between t_a and t_b and r_{bc} be the set of possible IA relations between t_b and t_c. The set of possible IA relations between t_a and t_c (say r_{ac}) can be computed using the composition operation $\otimes$ introduced by Allen [Allen 83]:

$$(5.29) \qquad r_{ac} = r_{ab} \otimes r_{bc} = \{i_1 \times i_2 \mid i_1 \in r_{ab} \wedge i_2 \in r_{bc}\}$$

where the result of $i_1 \times i_2$ can be looked up in table 5.2.

For example, if r_{ab} is $\{<, m\}$ and r_{bc} is $\{m\}$, then we have:

$$(5.30) \qquad \begin{aligned} r_{ac} &= \{<, m\} \otimes \{m\} \\ &= \{< \times\ m\} \cup \{m \times m\} \\ &= \{<\} \cup \{\mathrm{m}\} \\ &= \{<, m\} \end{aligned}$$

If there are more than one chain from t_a to t_c, then we take the intersection of the set of possible IA relations computed from each chain.

The above approach for a composite event with three sub-events can be repeated if there are more sub-events.

Our Algorithm for Traversing the QTS

Based on the discussion above, we can now give the algorithm for traversing the arcs of the QTS and extracting the appropriate IA relations between the time intervals associated with the situations in the QTS. This algorithm is shown in figure 5.3. The input to this algorithm is the QTS of a text. Recall from chapter 4 that a QTS is given as a two tuples: the set G of situations that are mentioned in the text and the set R of relations between pairs of situations. The algorithm first traverses those arcs in the QTS whose label is not one of constituency. This results in the determination of the possible IA relations between the time intervals of all the related situations in the QTS except that between an event and its sub-events. This is necessary as the algorithm for determining the set of possible IA relations between a composite event and its sub-events need to know the set of possible IA relations between the sub-events.

A R1 B/ B R2 C	<	>	m	mi	o	oi	s	si	d	di	f	fi
<	<	no info	<	<,m o,s d	<	<,m o,s d	<	<	<,m o,s d	<	<,m o,s d	<
>	no info	>	>,mi oi,d f	>	>,mi oi,d f	>	>,mi oi,d f	>	>,mi oi,d f	>	>	>
m	<	>,mi oi,si di	<	f fi =	<	o s d	m	m	o s d	<	o s d	<
mi	<,m o,di fi	>	s si =	>	oi d f	>	oi d f	>	oi d f	>	mi	mi
o	<	>,mi oi,si di	<	oi si di	b m o	o,oi di,d =	o	o di fi	o s d	<,m o,di fi	o s d	< m o
oi	<,m o,di fi	>	o di fi	>	o,oi di,d, =	> mi oi	oi d f	> mi oi	oi d f	>,mi oi,di si	oi	oi si di
s	<	>	<	mi	< m,o	oi d,f	s	s si,=	d	<,m o,di fi	d	< m o
si	<,m o,di fi	>	o di fi	mi	o di fi	oi	s si =	si	oi d f	di	oi	di
d	<	>	<	>	<,m o,s d	>,mi oi,d f	d	>,mi oi,d f	d	no info	d	< m, o, s, d
di	<,m o,di fi	>,mi oi,si di	o di fi	oi si di	o di fi	oi di si	o di fi	di	o,oi di d,=	di	oi si di	di
f	<	>	m	>	o s,d	> mi,oi	d	> mi,oi	d	>,mi oi,si di	f	f fi =
fi	<	>,mi oi,si di	m	oi si di	o	oi si di	o	di	o s d	di	f fi =	fi

Table 5.2: Composition of two IA Relations

```
Algorithm: Extract-Qual
Input:     (G, R)
Output:    The set of IA relations between time intervals
           corresponding to the relations between situations in R

  1. Let CONSTITUENCY = $\{r \mid r \in R \wedge r$ is of type constituency$\}$
  2. Let REMAINDER = R − CONSTITUENCY
  3. For RELATION $\in$ REMAINDER, do
     (a) If type of RELATION $\in$ {precedes, includes, meet, simultaneous, enables,
         generates}, return the corresponding set of IA relations.
     (b) If type of RELATION is none, perform resource contention reasoning.
  4. Perform constituency reasoning on elements in CONSTITUENCY.
  5. Return.
```

Figure 5.3: The algorithm for transversing the QTS

We next apply the algorithm for determining the relation between the time interval of an event and that of its sub-events described above.

As an illustration, consider the traversing of the QTS of the following example:

(5.31) a. Bring the soup to the boil stirring occasionally.

b. Meanwhile, chop a carrot.

The QTS of this example is given in figure 5.4. In the QTS, we have

- e_1 is the composite event described by *Bring the soup to the boil stirring occasionally*,
- e_2 is the event of *bring the soup to the boil*,
- e_3 is the event of *stir occasionally*,
- e_4 and e_5 are the two individual stirring events after decomposing e_3 and
- e_6 is the event described by *chop a carrot*.

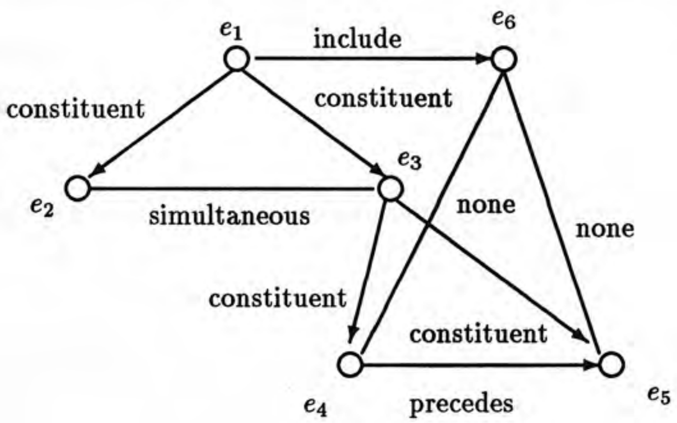


Figure 5.4: The QTS of example (5.31)

The IA relations derived for the above QTS are as below. In this derivation, we assume that the environment has only one agent, one chopping knife and no chopping machine:

1. From the three temporal relations between e_1 and e_6, between e_2 and e_3 and between e_4 and e_5, we have the following set of IA relations (see table 5.1):
   (a) te_6 $\{s,f,d,=\}$ te_1
   (b) te_2 $\{=\}$ te_3
   (c) te_4 $\{<,m\}$ te_5
2. There are two arcs marked with the **none** relation and since the chopping and stirring action both require the agent to bring it to completion and there is only one agent, we have the following set of IA relations:
   (a) te_6 $\{<,>,m,mi\}$ te_4
   (b) te_6 $\{<,>,m,mi\}$ te_5
3. Finally, we work out the set of IA relations between the time interval of the composite event and that of its sub-events. There are two composite events in figure 5.4. The IA relations derived are as follows:
   (a) te_1 $\{=\}$ te_2

(b) te_1 {=} te_3

(c) te_4 {*s*} te_3

(d) te_5 {*f*} te_3

The set of qualitative relations that can be extracted from the QTS in figure 5.4 is then as follows:

(5.32) te_6 {*s*, *f*, *d*, =} te_1
te_2 {=} te_3
te_4 {<,*m*,*o*} te_5
te_6 {<, >, *m*, *mi*} te_4
te_6 {<, >, *m*, *mi*} te_5
te_1 {=} te_2
te_1 {=} te_3
te_4 {*s*} te_3
te_5 {*f*} te_3

5.1.2 Quantitative Relations Between Time Intervals

Recall from chapter 3 that quantitative information is information about the start and end time points of time intervals associated with situations. The quantitative information is extracted from a sentence by the ADD-QUANT function during the process of building the situation representation underlying the sentence (see chapter 4). After extraction, the information is stored separately from the QTS as a set of equations of the form:

(5.33) X − Y R N

where X and Y is either start(I) or end(I) (I is a time interval), R is a relation from the set $\{<, >, \leq, \geq, =\}$ and N is a number. This set of equations will be added to the set of qualitative relations extracted from the QTS (see section 5.1.1) when carrying out constraint propagation.

To illustrate the working of the ADD-QUANT function, we consider the following examples (in these examples, we take te_i to be the time interval assigned to the event e_i):

Cook the rice for 30 minutes: In this case, the *for*-adverbial is analysed in the BUILD-SITUATION-REP function (see chapter 4) as indicating a duration. The call to add the quantitative information is:

(5.34) add-quant(e_1, [duration,30])

where e_1 is the event described by *cook the rice.* The quantitative information added is then:

(5.35) $\text{end}(te_1) - \text{start}(te_1) = 30$

10 minutes before serving, add the wine: In this case, the call to add the quantitative information is:

(5.36) add-quant(e_2, [e_1,10,before])

where e_1 is the event described by *serving* and e_2 is the event described by *add the wine.* The quantitative information added is as follows:

(5.37) $\text{start}(te_1) - \text{start}(te_2) = 10$

10 minutes before the end of cooking time, add the wine: In this example, the call to add the quantitative information is:

(5.38) add-quant(e_2, [[e_1,end],10,before])

where e_1 is the event which the time point entity described by *the end of cooking time* is semantically related and e_2 is the event described by *add the wine.* The quantitative information added is then:

(5.39) $\text{end}(te_1) - \text{start}(te_2) = 10$

When the water has cooled for 10 minutes, add the seasoning: In this case, the *when*-clause is analysed as describing the first 10 minutes of a cooling event.

Assuming that e_1 is the cooling event, we say that the *when*-clause is describing the neutral perfective aspect of e_1 (see section 4.3.2). The call to add the quantitative information is then:

(5.40) add-quant(e_2, [e_1,10,[neutral-perfective,when]])

where e_2 is the event described by ***add the seasoning***. The quantitative information added is then:

(5.41) $\text{start}(te_2) - \text{start}(te_1) = 10$

The extraction of quantitative information is complicated in those cases where durational information is not explicitly given. Consider for instance,

(5.42) a. Chop a carrot.

b. Brown the beef until tender.

where the durations of the chopping and browning event are not given. In such cases, it is not an easy issue to derive the duration of the event. Of course, there are events like the bring-to-boil event which may be given a default duration indicating the duration that the actions in the events normally last. However, there are also events where the duration of an action is dependent on the agent. For instance, a more experienced agent would take less time to chop than a less experienced one. To simplify the issue, we currently assume that each action has a default duration.

To end the section, consider example (5.31) again:

(5.31) a. Bring the soup to the boil stirring occasionally.

b. Meanwhile, chop a carrot.

Recall from the QTS of example (5.31) that e_2 is the bring-to-boil event, e_6 is the chopping event while e_4 and e_5 are the two individual stirring events. We assume that the chopping action in e_6 takes 5 minutes while the stirring action in e_4 and e_6 each takes 3 minutes. The quantitative information extracted from the events mentioned in (5.31) is thus:

(5.43)
$$\begin{aligned} \text{end}(te_2) - \text{start}(te_2) &= 30 \\ \text{end}(te_6) - \text{start}(te_6) &= 5 \\ \text{end}(te_4) - \text{start}(te_4) &= 3 \\ \text{end}(te_5) - \text{start}(te_5) &= 3 \end{aligned}$$

5.1.3 Summary

In this section, we have discussed how qualitative and quantitative relations between time intervals associated with the situations mentioned in a text are derived. The qualitative relations are derived from the relations between situations in the QTS and we show that the set of qualitative relations derived may vary in different environments.

Given the set of qualitative and quantitative relations between time intervals, the next problem is how to derive the temporal structure from the set. We now turn to this problem.

5.2 Temporal Reasoning and Temporal Structure

The problem of solving the set of qualitative and quantitative relations between time intervals can be viewed as a temporal reasoning problem. One method of solving is to view the problem as one of constraint satisfaction [Allen 83]. Recall from chapter 4 that a constraint satisfaction problem (CSP) can be defined as follows:

(5.44)
$$(\exists\, v_1)(\exists\, v_2)\ldots(\exists\, v_n)(v_1 \in D_1)(v_2 \in D_2)\ldots(v_n \in D_n)\ \mathrm{P}(v_1, v_2, \ldots, v_n)$$

where v_1, $v_2 \ldots v_n$ are variables and $\mathrm{P}(v_1, v_2, \ldots, v_n)$ stands for an arbitrary conjunction of predicates over subsets of the variables $\{v_1, v_2, \ldots, v_n\}$.

In section 5.2.1, we will show how a given set of qualitative and quantitative relations can be formalised into a CSP. We will do this in an incremental manner by first formalising the set of qualitative relations and then adding the quantitative relations into the formalisation.

In section 5.2.2, we will introduce temporal reasoning and analyse the requirements which our CSP imposes on temporal reasoning algorithms. Based on this analysis, we present the approach which we take to solving the CSP. The solution of the CSP may consist of more than one SOLUTION TUPLE each of which represents an assignment of one value to each of the variables in the CSP such that all the constraints are satisfied. Each solution tuple constitutes a possible temporal structure for the text.

5.2.1 Formalisation as a CSP

In this section, we will formalise a given set of qualitative and quantitative relations between time intervals as a CSP.

Formalising Qualitative Relations as a CSP

We consider how the qualitative relations expressed between time intervals can be regarded as a CSP. If we let S be the set of time intervals associated with the situations in a text, i.e.

(5.45) Let S $= \{te_i \mid te_i$ is the time interval associated with the i_{th} situation in the text and $1 \leq i \leq$ n$\}$

then the CSP corresponding to the set of qualitative relations is defined as follows:

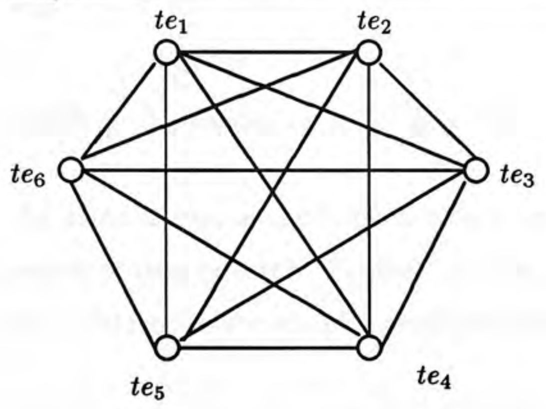
Variables The variables are the labels on the ${}^nC_2$ arcs that link the te_i in S, i.e the set of variables is:

(5.46) Let VAR $= \{r_{ij} \mid r_{ij}$ is a label on an arc linking te_i and te_j, $1 \leq i < n \wedge i + 1 \leq j \leq n\}$

Domains The domains of these variables are the set of possible IA relations that can hold between the te_i and te_j.

Constraints The constraints on the variables are binary:

(5.47) $r_{ij} \otimes r_{jk} = r_{ik}$



$r_{61} \in \{s,f,d,=\}$
$r_{23} \in \{=\}$
$r_{45} \in \{<,m,o\}$
$r_{64} \in \{<,>,m,mi\}$
$r_{65} \in \{<,>,m,mi\}$
$r_{12} \in \{=\}$
$r_{13} \in \{=\}$
$r_{43} \in \{s\}$
$r_{53} \in \{f\}$
All other arcs labels = the full set of 13 IA relations.

Figure 5.5: The network representation of the CSP for example (5.32)

where $\otimes$ is the composition operation defined in (5.29).

For example, the CSP corresponding to the set of qualitative relations given in (5.32) above and reproduced below:

(5.32) te_6 {*s, f, d,* =} te_1
te_2 {=} te_3
te_4 {<,*m*,*o*} te_5
te_6 {<, >, *m*, *mi*} te_4
te_6 {<, >, *m*, *mi*} te_5
te_1 {=} te_2
te_1 {=} te_3
te_4 {*s*} te_3
te_5 {*f*} te_3

can be visualised as in figure 5.5.

For ease of exposition, we will call the network representation of the CSP corresponding to a set of qualitative relations (such as the network in figure 5.5), a Constraint Interval Network (CIN). We now consider the problem of adding quantitative relations into the

CIN.

Adding Quantitative Relations into the CSP

As noted above, quantitative relations are relations between the start and end time points of time intervals. For instance, the quantitative relations extracted from example (5.31) given above and reproduced below:

(5.31) a. Bring the soup to the boil stirring occasionally.

b. Meanwhile, chop a carrot.

are as follows:

(5.43) $\text{end}(te_2) - \text{start}(te_2) = 30$

$\text{end}(te_6) - \text{start}(te_6) = 5$

$\text{end}(te_4) - \text{start}(te_4) = 3$

$\text{end}(te_5) - \text{start}(te_5) = 3$

where e_2 is the bring-to-boil event, e_6 is the chopping event and e_4 and e_5 are the two individual stirring events.

Following [Meiri 91], one can regard the quantitative relations between two time points, p_i and p_j, as constraining the possible values for the distance $p_j - p_i$ to a given set of intervals, i_1 to i_k, each of which may be open or closed on either side:

(5.48) $(p_j - p_i \in i_1) \vee \ldots (p_j - p_i \in i_k)$

For instance, the quantitative relations in (5.43) can be recoded as:

(5.49) $\text{end}(te_2) - \text{start}(te_2) \in [30,30]$

$\text{end}(te_6) - \text{start}(te_6) \in [5,5]$

$\text{end}(te_4) - \text{start}(te_4) \in [3,3]$

$\text{end}(te_5) - \text{start}(te_5) \in [3,3]$

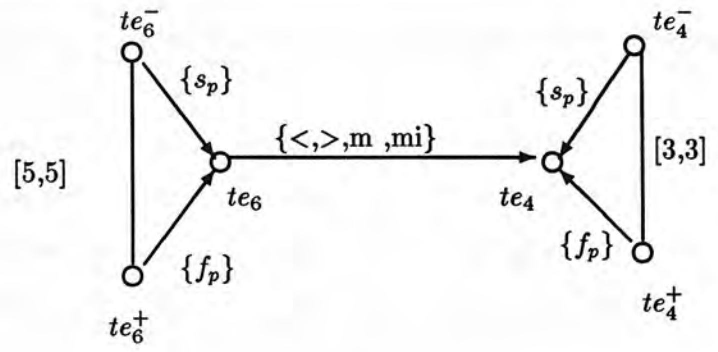


Figure 5.6: The network representation of the CSP for example (5.51)

where $[t_1, t_2]$ represents an interval that is closed on both sides or equivalently that the interval starts at t_1 and ends at t_2 on a time line. As another example, consider the quantitative relation expressed in the sentence *John left for work between 7.05 and 7.10 am.* If we assume the existence of a speech time point, say p_0 at 7.15 am, then the quantitative relation in the above sentence can be represented as:

(5.50) $p_0 - p_1 \in (5,10)$

where p_1 is the start time point of the john-go-to-work event and (5,10) represents a time interval which is open at both ends.

We are now in a position to add quantitative relations into the CIN corresponding to a set of qualitative relations. The resultant network has both time points and time intervals. For example, the CSP corresponding to the following set of qualitative and quantitative relations:

(5.51) te_6 $\{<,>,m,mi\}$ te_4
$\text{end}(te_6) - \text{start}(te_6) = 5$
$\text{end}(te_4) - \text{start}(te_4) = 3$

is given in figure 5.6. For ease of exposition, we will call the resultant network when quantitative relations are added a CIN^{+quan}. From figure 5.6, one can identify 3 kinds of variables in a CIN^{+quan}:

Relation	Symbol	Inverse	Relations on Endpoints
p before I	b_p	a_p	$p < I^-$
p starts I	s_p	si_p	$p = I^-$
p during I	d_p	di_p	$I^- < p < I^+$
p finishes I	f_p	fi_p	$p = I^+$
p after I	a_p	b_p	$p > I^+$

Table 5.3: The relations between time points and time intervals

1. One kind of variable arises from the labels on the arcs linking two time intervals such as that between te_6 and te_4. The domain of these variables comes from the thirteen possible IA relations.

2. Another kind of variable arises from the labels on the arcs linking two time points such as that between te_6^- and te_6^+. The domain of these variables are the possible intervals from $(-\infty, \infty)$.

3. The third kind of variable arises from the labels on the arcs linking a time point and a time interval. The domain of these variables are the possible relations that can hold between a time point and a time interval. These relations are given in table 5.3.

In the next section, we discuss algorithms from work in temporal reasoning that can be applied to solve a CIN^{+quan}.

5.2.2 Analysing the Temporal Reasoning Requirements of CIN^{+quan}

In this section, we are interested in algorithms that can solve the CSP corresponding to the set of qualitative and quantitative relations between time intervals discussed above. A solution tuple to the CSP constitutes a temporal structure for the text from which the set of qualitative and quantitative relations are derived.

Some Preliminaries

Typically, a constraint propagation algorithm employs network consistency methods whose goal is to refine the value domains as much as possible so that values which cannot be part of a global solution to the problem are eliminated. There are two properties of a constraint propagation algorithm that are of interest to us:

1. The algorithm must be able to detect whether the CSP is consistent, i.e. whether there exists at least one solution-tuple.

2. If the CSP is consistent, we would like to know whether all possible combinations of values from the resultant domains of each variable are solution tuples. If this is so, we say that the resultant CSP is a MINIMAL NETWORK. Given a minimal network, it is easy to obtain the possible solution tuples. To illustrate, assume that there are three variables in a CSP (say var_1, var_2 and var_3) and that in the minimal network of the CSP, the domains of the respective variables are {1, 2}, {3} and {4}. The possible solution tuples are then the possible combination of values from the three domains:

Solution Number	var_1	var_2	var_3
1	1	3	4
2	2	3	4

For ease of exposition, we will call an algorithm which detects inconsistency and returns minimal network an EXACT algorithm.

Thus, given a CSP and an exact constraint propagation algorithm to solve it, the solution tuples and thus the temporal structures can be read off from the minimal network. However, if there is no exact constraint propagation algorithm to solve a CSP, then the solution tuples cannot be read off directly. Instead, backtracking has to be employed after constraint propagation to find the solution tuples. For instance, assume that the network containing var_1, var_2 and var_3 discussed in the paragraph above is not minimal after constraint propagation. In this case, there is no guarantee that every value in the domain of each variable takes part in a solution tuple. To obtain the solution tuples, backtracking is carried out. To illustrate, assume that assigning a value of 3 to var_2 is incompatible with assigning a value of 1 to var_1. When this happens, backtracking occurs to try to reassign a value to var_2. Since there are no more possibilities for var_2, backtracking again occurs to reassign a value of 2 to var_1. Assuming that this is consistent with a value of 3 for var_2, the algorithm continues to assign a value of 4 to var_3. If this is consistent, the solution tuple will be simply:

Solution Number	var_1	var_2	var_3
1	2	3	4

Temporal Reasoning Requirements of CIN^{+quan}

In this section, we analyse the temporal reasoning requirements of CIN^{+quan} in order to determine whether there exists exact constraint propagation algorithms for solving it.

Work in temporal reasoning [Vilain *et al* 89, Meiri 91] has shown that there exists a subset of CIN^{+quan} which can be transformed into an equivalent CSP involving time points instead of time intervals. In this equivalent CSP, variables are labels on arcs linking time points instead of time intervals and the domain of these variables are one of the following: $\{>\}$, $\{<\}$, $\{=\}$, $<, =\}$, $\{>, =\}$, $\{<, >, =\}$. For ease of exposition, we will call this equivalent CSP CPN^{+quan}.

Meiri [Meiri 91] shows that a slightly modified version of the constraint propagation algorithm of Allen [Allen 83] is exact for CPN^{+quan}. Unfortunately, not all CIN^{+quan} can be transformed into an equivalent CPN^{+quan}. For instance, a CIN^{+quan} where the domains of the variables include set of possible IA relations that denotes disjointedness such as $\{<, >, m, mi\}$ or $\{<, >\}$ cannot be transformed into an equivalent CPN^{+quan}. In such CIN^{+quan} where there is no equivalent CPN^{+quan}, Meiri [Meiri 91] shows that the CIN^{+quan} cannot be solved 'exactly'.

Now, recall from section 5.1.1 that it is possible to derive $\{<, >, m, mi\}$ as the set of possible IA relations between two time intervals in our TA2 framework. This means that there are some CIN^{+quan} from our texts that cannot be solved 'exactly'. As noted, the usual approach is to first apply the constraint propagation algorithm followed by a backtracking search.

We are now in a position to describe the approach we use to solve the CIN^{+quan} extracted from a text.

Solving CIN^{+quan}

Given that there are some CIN^{+quan} from a text which cannot be solved exactly, our approach to solving the CSP is as follows:

1. First, we employ a preprocessing strategy to refine the domains of each variables in the CSP.
2. Next, we employ a backtracking search to search for the solution tuples among the reduced domains of the variables in the CSP.

Preprocessing Strategy In our work, we adopt the preprocessing strategy proposed by Kautz and Ladkin [Kautz & Ladkin 91]. In this strategy, Kautz and Ladkin suggests splitting CIN^{+quan} into two networks: a network of time intervals whose labels take values from the set of IA relations (an IA network) and a network of time points (a metric network) whose labels take value from ($-\infty$, ∞). The network of time intervals can be processed using the constraint propagation of Allen [Allen 83] (for ease of exposition, we will call this algorithm ALL3) while the network of time points can be solved in polynomial time using the all-pairs shortest path algorithm (APS) of Aho [Aho *et al* 76, 198]. The approach can be described as follows:

1. Separately propagate the constraints in the IA network using ALL3 and in the metric network using APS.
2. Derive a set of new IA relations from the metric network and add these to the IA network. Propagate with ALL3. For instance, in the following metric network:

(5.52) $\text{end}(I) - \text{start}(I) \in (3, \infty)$
$\text{end}(J) - \text{start}(J) \in (-\infty, 2)$

where I and J are time intervals, we can deduce that I cannot be during, start, finish or equal to J. This is because the duration of I is at least 3 and the duration of J is at most two. Hence, the set of possible IA relations between I and J is {<, >, *o*, *oi*, *m*, *mi*, *c*, *fi*, *si*}.

3. Derive new quantitative relations from the IA network and add these to the metric network. Propagate with APS. For instance, in the following IA network where the set of possible IA relations between time interval I and J is:

   (5.53) I {<} J

   the quantitative relations between I and J can be obtained from the following equivalence:

   (5.54) $I \{<\} J \equiv \text{end}(I) < \text{start}(J) \wedge$
   $\text{start}(I) < \text{end}(I) \wedge$
   $\text{start}(J) < \text{end}(J)$

   That is, the quantitative relations derived are:

   (5.55) $\text{end}(I) - \text{start}(J) \in (-\infty, 0)$
   $\text{start}(I) - \text{end}(I) \in (-\infty, 0)$
   $\text{start}(J) - \text{end}(J) \in (-\infty, 0)$

4. Repeat step (2) to (3) until there is no change in the IA network and metric network.

Backtracking Search The result of preprocessing is typically a more constrained CIN^{+quan}. We can then initiate backtracking search for possible solution tuples. To do this, we sequentially assign a single value to each of the possible labels in the IA network. At each step, the chosen IA relation is propagated and any new derived metric relations are propagated as described above. This continues until no more new metric or IA relations are derived. If a conflict is detected, backtracking is initiated.

In our work, we order the choice of IA relations during backtracking using the following ordering:

1. $\{=\}$
2. $\{s,si,f,fi\}$
3. $\{o,oi\}$
4. $\{d,di\}$
5. $\{m,mi\}$
6. $\{<,>\}$

The above ordering is a heuristic for ensuring that the first solution tuple found is one where the intervals are packed as tightly as possible. Currently, we only return this solution tuple although it would be possible to return the remaining solution tuples. As noted, these solution tuples represent the possible temporal structures of a text.

5.2.3 Summary

In this section, we have shown how a set of qualitative and quantitative relations extracted from a text can be viewed as a constraint satisfaction problem. The possible solution tuples to the CSP constitute the possible temporal structures for the text.

In general, the set of qualitative and quantitative relations extracted from an instructional text maps into a class of CSP called CIN^{+quan}. In some cases, a CIN^{+quan} can be transformed into an equivalent CPN^{+quan} for which an exact algorithm exists to solve it. However, in the general case, no such exact algorithm exists and the usual approach is to have a preprocessing stage of constraint propagation to try to reduce the size of the variable domains. This is then followed by a backtracking search.

In the next section, we will work through a simple example to illustrate how solution tuples and thus the temporal structures for an instructional text can be obtained. This

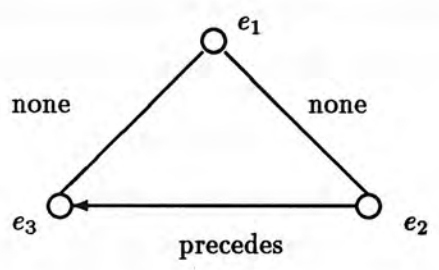


Figure 5.7: The QTS of example (5.56)

example also illustrates how different temporal structures can be obtained from the same QTS in different circumstances.

5.3 A Worked Example

We will consider the simple example in (5.56) to illustrate the extraction of temporal structures from its QTS.

(5.56) a. e_1: Chop a carrot.

b. e_2: Chop a potato.

c. e_3: Boil the potato.

The QTS of example (5.56) is shown in figure 5.7. We will divide our discussion into two scenarios to show how different temporal structures can be obtained from the same QTS.

5.3.1 Case 1

In this case, we shall assume that the agent does not need to look after the potatoes while they are boiling. The set of qualitative relations derived are as follows:

1. The **precedes** label on the arc from e_2 to e_3 means that the set of possible IA relations between te_2 and te_3 is $\{<, m\}$.

2. We reason about the **none** arc between e_1 and e_2. Since both chopping action in e_1 and e_2 require the agent to bring it to completion and there is only one agent, rule (5.11) given above and reproduced below is applicable:

   (5.11) IF none(e_1, e_2)
   AND resource-contention(e_1, e_2)
   THEN te_1 {<,>,m,mi} te_2

   The possible IA relations between te_1 and te_2 is then $\{<,>,m,mi\}$.

3. Next, we reason about the **none** arc between e_1 and e_3. Since the agent does not need to look after the potatoes while they are boiling, there is no resource contention between e_1 and e_3. Rule (5.10) given above and reproduced below:

   (5.10) IF none(e_1, e_2)
   AND $\neg$ resource-contention(e_1, e_2)
   THEN te_1 {<,>,m,mi,o,oi,=,s,si,f,fi,d,di} te_2

   is now applicable and the possible IA relations between te_1 and te_3 is $\{<, >, m, mi, o, oi, =, s, si, f, fi, d, di\}$.

Together with the quantitative information regarding the duration of e_1, e_2 and e_3, the set of qualitative and quantitative information that can be extracted for this case are as follows:

(5.57)
$$te_1 \ \{<,>,m,mi,o,oi,=,s,si,f,fi,d,di\} \ te_3$$
$$te_1 \ \{<,>,m,mi\} \ te_2$$
$$te_2 \ \{<,m\} \ te_3$$
$$te_1^+ - te_1^- = 5$$
$$te_2^+ - te_2^- = 5$$
$$te_3^+ - te_3^- = 20$$

The set of qualitative and quantitative relations in (5.57) can be transformed into the IA network of (5.58a) and the metric network of (5.58b) where a, b and c are nodes representing te_1, te_2 and te_3 respectively, a^-, b^- and c^- are nodes representing the start time points of te_1, te_2 and te_3 respectively, and a^+, b^+ and c^+ are nodes representing the end time points of te_1, te_2 and te_3 respectively.

(5.58) a. $r_{ac} \in \{<,>,m,mi,o,oi,=,s,si,f,fi,d,di\}$

$r_{ab} \in \{<,>,m,mi\}$

$r_{bc} \in \{<,m\}$

b. $r_{a^- a^+} \in [5,5]$

$r_{b^- b^+} \in [5,5]$

$r_{c^- c^+} \in [20,20]$

(5.58) is preprocessed with Kautz and Ladkin's algorithm and the IA network is now:

(5.59) $r_{ac} \in \{<,>,m,mi,o,oi,s,f,d\}$

$r_{ab} \in \{<,>,m,mi\}$

$r_{bc} \in \{<,m\}$

Note that the reduced domain in r_{ac} is due to the effect of the quantitative relation encoded in the metric network. More specifically, since the duration of e_1 is less than the duration of e_3, it is not possible for e_1 to be equal to (=), started by (*si*), finished by (*fi*) or contained in (*di*) e_3.

We now carry out the backtracking search. The trace is as follows:

1. We choose first to assign a value to r_{ac}. The possible values are from the set {<, >, *d*, *s*, *f*, *m*, *mi*, *o*, *oi*}. Following the heuristic, we choose the value *s*.

2. Upon propagating the above choice, the domain of r_{ab} is reduced to {>, *mi*} and that of r_{bc} to {<}. Again, following the heuristic, we assign the value *mi* to r_{ab}.

3. No inconsistency is detected and thus a solution-tuple is found in which r_{ac} is assigned *s*, r_{ab} is assigned *mi* and r_{bc} is assigned *m*.

The solution tuple translates into the temporal structure for example (5.56) in figure 5.8.

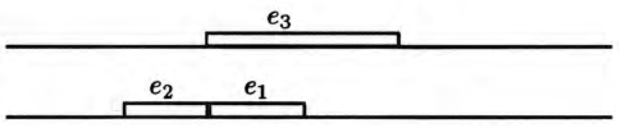


Figure 5.8: A temporal structure for example (5.56)

5.3.2 Case 2

In this case, we shall assume that the agent needs to look after the potatoes while they are boiling. The set of qualitative and quantitative relations are then as in (5.57) except for

(5.60) te_1 $\{<, >, m, mi\}$ te_3

since now there is a resource contention between the chop-a-carrot event and the boiling event. This set is given in (5.61).

(5.61) a. $r_{ac} \in \{<,>,m,mi\}$

$r_{ab} \in \{<,>,m,mi\}$

$r_{bc} \in \{<,m\}$

b. $r_{a^-a^+} \in [5,5]$

$r_{b^-b^+} \in [5,5]$

$r_{c^-c^+} \in [20,20]$

After preprocessing (5.61), there is no change in the IA network. We now carry out the backtracking search. The trace is as follows:

1. We choose first to assign a value to r_{ac}. The possible values are $\{<, >, m, mi\}$ and according to the heuristic, we choose the value m.
2. Upon propagation, the domain of r_{ab} is reduced to $\{>, mi\}$ and that of r_{bc} to $\{<\}$. According to the heuristic, we assign mi to r_{ab}.

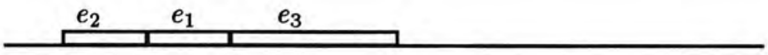


Figure 5.9: Another temporal structure for example (5.56)

3. Upon propagation, no inconsistency is detected. Thus, we have a solution-tuple, in which r_{ac} is assigned m, r_{ab} is assigned mi and r_{bc} is assigned $<$.

The solution tuple obtained above translates into the temporal structure shown in figure 5.9.

5.4 Conclusion

In this chapter, we have established the connection between temporal analysis and temporal reasoning. We accomplished this by showing that the set of qualitative and quantitative relations from a text can be transformed into a constraint satisfaction problem and solved using algorithms developed in the field of temporal reasoning.

By successfully integrating temporal analysis with temporal reasoning, we show how qualitative and quantitative information can be used together in temporal analysis. Previous work on temporal analysis has only focused on qualitative information.

We have also shown how different temporal structures for the same text can be derived in different environments.

Chapter 6

The TASTE System

This chapter presents TASTE, a system implemented using the TA2 framework for the Temporal Analysis of inSTructional tExt. TASTE takes as its input the sequence of sentences in a text and outputs the text's temporal structure.

TASTE is implemented in Common Lisp and has been tested on nine cookery recipes. These recipes are taken from two cookbooks (namely *The Electric Casserole Cook Book* by the Good HouseKeeping Institute and the Harvest *Traditional British Cooking*) and from the Internet newsgroup REC.FOOD.RECIPES. One of these recipes and the result of processing it by TASTE is given in chapter 1. The remaining eight recipes and the result of their processing by TASTE are given in Appendix A. The output of TASTE is to be interpreted as follows:

- The top diagram indicates the quasi-temporal structure which is a network whose nodes are the indices of the events (such as E1) and states (such as ST1) mentioned in the text and the arcs between the nodes are the relations between the events and states. The sentences describing the events and states are given to the left of the diagram. Finally, as the graphic routine we used do not permit directed arrow, we adopt the convention that the direction of the arrow on the arc between nodes is from the node associated with the larger index to the node associated with the smaller index. For instance, if the arc between E1 and E2 is labelled `a`, this is taken to mean that E2 is **after** E1.

- The bottom diagram indicates the temporal structure where the horizontal lines are time lines with time increasing from left to right and the black boxes represent time intervals occupied by events and states. Note that if the index of an event is E1, then the time intervals assigned to E1 is given by TE1.

Although TASTE is tested only on cookery recipes, we believe its architecture is general enough to handle other instructional texts as well. We will now give an overview of the architecture of TASTE before outlining the structure of the rest of this chapter.

6.1 An Overview Of System Architecture

Figure 6.1 gives an overview of the architecture of TASTE.[1]

TASTE accepts a recipe text as input. The system's operation can be thought of as occurring in two phases.

In the first phase, TASTE builds up the QTS of the text and records any quantitative information found in it. This phase implements the BUILD-QTS algorithm given in figure 4.1 (see chapter 4). As can be seen from figure 6.1, the BUILD-QTS algorithm takes as input the set of sentences found in the text. Each sentence goes through the following steps:

1. TASTE invokes the BUILD-INTERMEDIATE-REP module to build an intermediate representation of the sentence. This intermediate representation represents the semantic content of the sentence and is called a Recoverable Semantic Structure (RS). We will further discuss the working of the BUILD-INTERMEDIATE-REP algorithm in section 6.4.

2. The RS structure of a sentence is passed to the INCREMENTAL-BUILD-QTS algorithm (see figure 4.2, figure 4.3 and figure 4.4 in chapter 4). The algorithm

[1]In the figure, slender rectangular boxes and the dotted line box denote processes while the wider boxes denote data. A directed arrow from box A to box B means that data flows from A to B.

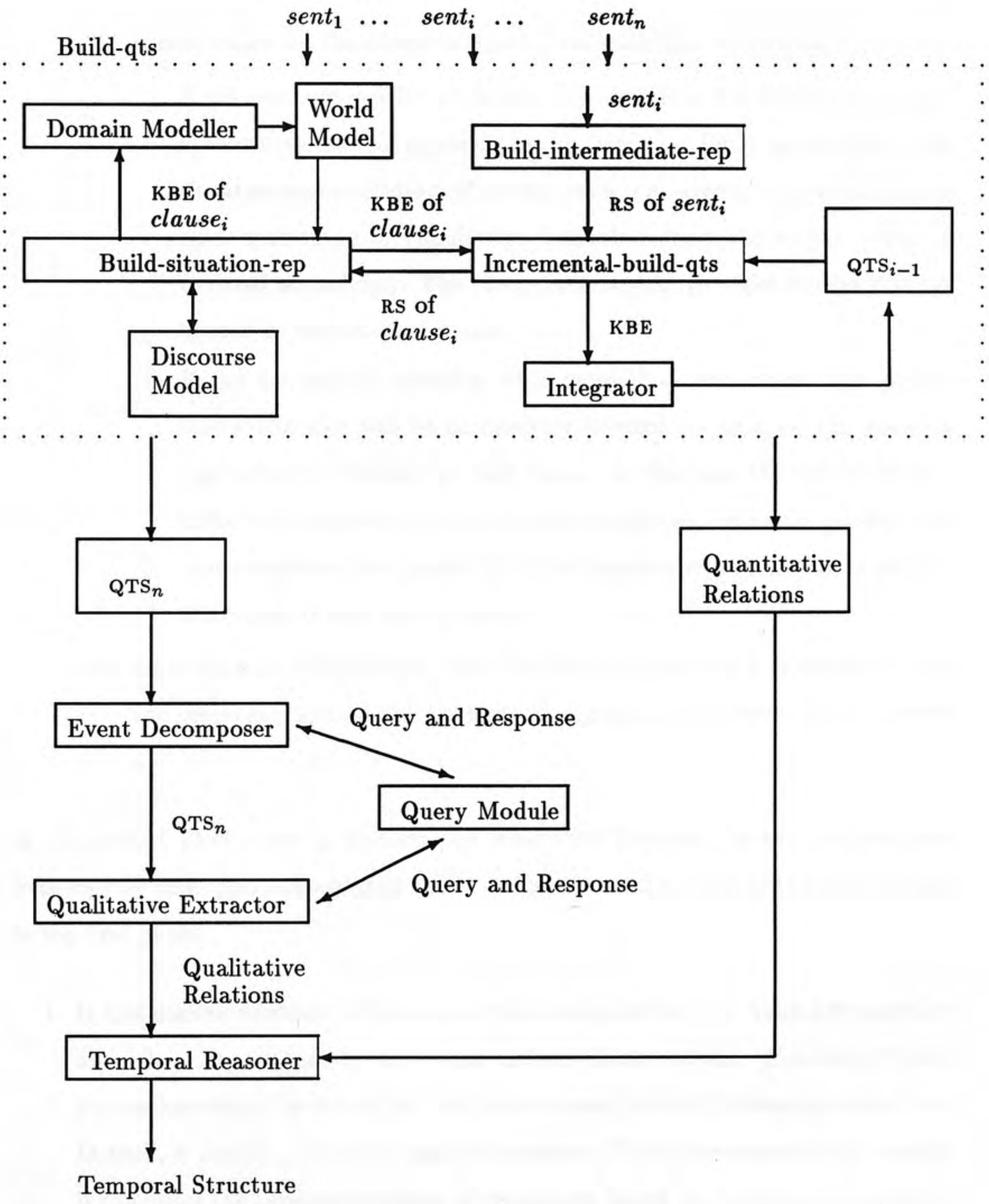


Figure 6.1: The architecture of TASTE

extracts any quantitative information found in the sentence and incrementally builds up the QTS via the following steps:

(a) It first builds up the representation of the situations underlying a sentence:

   i. If the sentence is a simple clause, the algorithm will invoke the BUILD-SITUATION-REP (see figure 4.11 in chapter 4) to build up the underlying situation representation. If the situation is an event, it is sent to the DOMAIN MODELLER for simulation. After simulation, the WORLD MODEL is updated accordingly. The DISCOURSE MODEL provides the information needed in reference resolution.

   ii. If the sentence is complex with more than one clause, the BUILD-SITUATION-REP will be successively invoked to build up the situation representation underlying each clause. In this case, the individual situation representations may be further 'composed' into a composite event representation (see chapter 3). The situation representation is called a Knowledge Based Entity (KBE).

(b) Once the KBE underlying a clause has been constructed, it is integrated into the preceding QTS by the INTEGRATE algorithm (see figure 4.5 in chapter 4).

In the second phase, TASTE extracts the qualitative Interval Algebra (IA) relations between the time intervals assigned to the situations at the nodes of the QTS built up in the first phase.

1. It first decides whether to decompose those events in the QTS which are repetitive and where the actions in the events act on plural objects. Currently, TASTE has no knowledge to determine whether an event should be decomposed or not. Instead, it simply queries for such information. If the decomposition is needed, it computes the necessary number of repetitions based on durational information (see section 4.4 in chapter 4).

2. The 'decomposed' QTS is now traversed by the QUALITATIVE EXTRACTOR module to extract the qualitative IA relations. The algorithm implemented is that given

in figure 5.3 in chapter 5. The algorithm invokes rule reasoning about resources when traversing those arcs in the QTS with the **none** label. As noted in chapter 4, the rules need access to environmental information. In TASTE, this information is obtained through querying.

The set of qualitative IA relations extracted from the QTS is now combined with the set of quantitative relations extracted from the text. As explained in chapter 5, we can regard this combined set as constituting a constraint satisfaction problem (CSP). The CSP is solved by a TEMPORAL REASONER which carries out the preprocessing algorithm of Kautz and Ladkin [Kautz & Ladkin 91] and subsequent backtracking searches for the 'best' ordering of events (see chapter 5).

The rest of the chapter is structured as follows.

Section 6.2 discusses the implementation of the various kinds of representation found in TASTE. In this implementation, feature structures serve as a core representation mechanism for the following:

1. to implement the KBE structure;
2. to implement the various intermediate levels of representation (namely the AS and RS level of representation) needed in mapping a sentence into its KBE and
3. to implement structures encoding the relations that hold between pairs of situations (as will be explained in the section, this in turn allows the QTS to be represented simply as a list of feature structures).

Section 6.3 discusses the rule mechanism implemented in TASTE. This rule mechanism is used to implement the knowledge required in the following tasks:

1. to map a sentence into its KBE structure;
2. to determine the set of previous events whose relations with the current situation can be deduced;
3. to determine the relations between situations and

4. to determine the qualitative IA relations between the time intervals of the situations described in the text.

Section 6.4 will discuss how the rule mechanism is used to implement the mapping from a sentence to the KBE structure of the situation underlying the sentence. These mappings are done in the BUILD-INTERMEDIATE-REP, INCREMENTAL-BUILD-QTS and BUILD-SITUATION-REP modules. For ease of exposition, we will call these modules the Natural Language Front End (NLFE) of TASTE.

Section 6.5 will discuss how the rule mechanism is used to implement the reasoning needs of the modules after NLFE. In particular, we show how the rule mechanism is used to determine the set of previous events whose relations with the current situation can be deduced (see section 4.1.2), the determination of the relations between situations and the determination of the qualitative IA relations between the time intervals of situations.

Section 6.6 takes a step back and presents an evaluation of the TASTE system in terms of data coverage, the methodology employed for evaluation and the nature of the results obtained by the system.

Finally, section 6.7 ends with a conclusion and summary.

6.2 The Representation Structures

There are different kinds of representation needed in TASTE:

1. First, we need to represent the various levels of representation required in the NLFE. There are three such levels since a sentence in TASTE is processed in three stages to obtain the representation of the situation underlying the sentence. These three levels of representation are similar to that used in Dale's EPICURE system [Dale 92], a Natural Language Generation System. At the first level, we have the Abstract Syntactic Structure (AS) which is mapped into the Recoverable Semantic Structure (RS). The RS structure is finally mapped into a

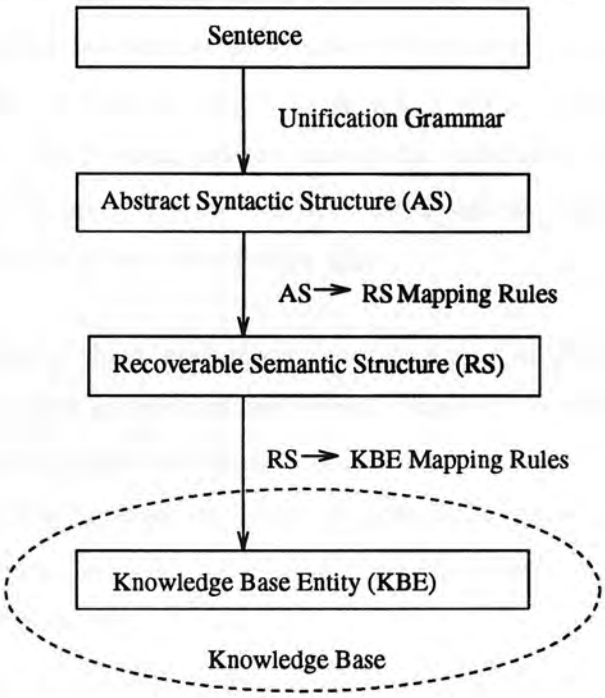

Sentence
Unification Grammar
Abstract Syntactic Structure (AS)
AS→ RS Mapping Rules
Recoverable Semantic Structure (RS)
RS→ KBE Mapping Rules
Knowledge Base Entity (KBE)
Knowledge Base

Figure 6.2: The levels of representation in the NLFE

Knowledge Base Entity (KBE). The levels of representation involved in extracting the representation of the situation underlying a sentence are given in figure 6.2.

2. We need to represent the QTS built up by TASTE. In terms of KBEs, the QTS can be thought of as encoding the relations between the various KBEs.

To implement the representations indicated above, we use feature structures as the core representation mechanism. This has the following advantages:

1. There exist unification-based chart parsers which can work with a grammar and lexicon written using the PATR notation to produce the initial AS structure compositionally. PATR is a well-defined formalism where atomic categories such as NP and V are replaced by sets of feature specifications. Such a formalism enables the grammar and lexical entries to be written in a declarative and perspicuous fashion.

2. Feature structures can be easily manipulated (i.e. accessed and extended). In fact, one can define a notion of a PATH to pick out a certain portion of a feature structure. The common notation for a path is ⟨F a b c⟩ where F is an identifier for the feature specification and the second and subsequent symbols are feature names. This easy manipulation facilitates the design of a rule reasoning system over the feature structures (see section 6.2).

We will now describe the three levels of representation used in the NLFE as well as the representation of the QTS in terms of the representation of the relations between the KBEs. In discussing the feature structure implementation of these representations, we will be using a BNF-like notation to define the grammars which describe the feature structures. The various conventions used in this notation have the following interpretations (taken from [Dale 92]):

In each rule of the form

A ::- B_1 B_2 .... B_n

the left-hand side is the name of an ATTRIBUTE which appears in a feature structure, and the right hand side specifies the possible VALUEs of that attribute. The symbols used to specify the right-hand sides of these rules should be understood as follows:

- Values in italics, as in

  state ::- s_i

  are atomic values. Thus, this rule states that the attribute STATE has as its value some symbol s_i. A subscripted item such as s_i is intended to indicate that any symbol of this form can be used, where i is replaced by an integer.

- Values which are in roman type, as in

  occurs ::- begin end

  are themselves attributes. Thus, this rule states that the attribute, OCCURS, has as its value a structure that itself consists of two attributes, BEGIN and END.

- Where a number of different values are possible, the symbol '|' is used to indicate disjunction. Thus, the rule

  index ::- x_i | e_i

  states that the attribute INDEX may have as its value either a symbol of the form x_i or a symbol of the form e_i.

- Where there are a potentially infinite set of values, these are represented either by a gloss in brackets, as in

  prop ::- ⟨an attribute-value pair⟩

  or by one or more example values followed by ellipsis, as in

  out ::- result ...

- The symbol '*' attached to an item signifies that the expression of that item in the grammar may appear zero or more times.

- An item within square brackets indicates that the item is optional.

- An item in small capitals, as in

  spec ::- structure substance packaging PROP*

  is shorthand for the expansion of that item expressed elsewhere in the grammar, i.e., it is neither an attribute or an atomic value.

6.2.1 Abstract Syntactic Structure

An Abstract Syntactic Structure (AS) is close to the syntax of the linguistic form of a sentence. For instance, the AS structure of a clause such as *chop the potatoes* is:

$$(6.1)\qquad \begin{bmatrix} \textsf{sem:} & \begin{bmatrix} \textsf{mood: imperative} & \\ \textsf{tprop:} & \begin{bmatrix} \textsf{tense: present} & \\ \textsf{prop:} & \begin{bmatrix} \textsf{agent: hearer} & \\ \textsf{action: chop} & \\ \textsf{args:} & \begin{bmatrix} \textsf{obj: } \ldots \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

where the various features have the following meanings:

- The *mood* feature indicates whether the clause is imperative or declarative. In our domain, the clauses are imperative.
- The *tprop* feature indicates the *tense* of the clause and the *prop*ositional content of the clause. The *prop* feature mirrors the subject-predicate form of a clause and contains the following features:
  - The *agent* feature indicates the agent which is to carry out the imperative clause. This is by default the hearer of the clause.
  - The *action* feature indicates the action to be carried out and is given by the verb in the clause.
  - The *args* feature contains the AS structure of the objects involved in the action. For our current example, the object involved is that described by the NP *the carrots.*

The ellided object in (6.1) is the AS structure of the definite NP *the potatoes* [Dale 92]:

$$(6.2)\qquad \begin{bmatrix} \textsf{status:} & \begin{bmatrix} \textsf{given: } + \end{bmatrix} \\ \textsf{spec:} & \begin{bmatrix} \textsf{agr:} & \begin{bmatrix} \textsf{countable: } + \\ \textsf{number: pl} \end{bmatrix} \\ \textsf{desc:} & \begin{bmatrix} \textsf{head: potato} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

where the various features have the following meanings:

- The *status* feature indicates whether the NP is describing a *given* object i.e. related to a previously mentioned object. For instance, the value of the *given* feature for a definite NP is + while that for an indefinite NP is −.

- The *spec* features indicates whether the NP is countable or not (the *countable* feature) and singular or plural (the *number* feature). In addition, it contains the *head* noun in the NP and any *mod*ifier of this noun (for the NP *the potatoes*, there is no *mod* feature).

This closeness facilitates the use of compositional interpretation during parsing to build up the AS structure (see section 6.4).

In addition to simple clauses such as *Chop the potatoes* discussed above, we also handle sentences which contain more than one clause and sentences containing adverbial clauses. We will now discuss how the AS structures of these sentences can be formed from the basic AS structure of a simple clause described above.

Example 1 We consider sentences that have more than one clause connected by a temporal adjunct. For example, the AS structure corresponding to *Before you mash the potatoes, take out the chicken leg* is as follows:

$$(6.3) \quad \left[\begin{array}{ll} \text{connector:} & \left[\text{type: before}\right] \\ \text{subord: } \ldots & \\ \text{main: } \ldots & \end{array}\right]$$

where the ellided SUBORD and MAIN features are the AS structures of the subordinate clause and the main clause respectively. The SUBORD feature is as follows:

$$(6.4) \quad \left[\begin{array}{ll} \text{mood: imperative} & \\ \text{tprop:} & \left[\begin{array}{ll} \text{tense: present} & \\ \text{prop:} & \left[\begin{array}{ll} \text{agent: hearer} & \\ \text{action: mashing} & \\ \text{args:} & \left[\text{obj: } \ldots\right] \end{array}\right] \end{array}\right] \end{array}\right]$$

where the ellided object is the AS structure of the NP *the potatoes* (see (6.2)). The MAIN feature is similar and is as follows:

(6.5)
$$\left[\begin{array}{ll} \text{mood: imperative} & \\ \text{tprop:} & \left[\begin{array}{ll} \text{tense: present} & \\ \text{prop:} & \left[\begin{array}{ll} \text{agent: hearer} & \\ \text{action: taking-out} & \\ \text{args:} & \left[\text{obj: } \ldots\right] \end{array}\right] \end{array}\right] \end{array}\right]$$

where the ellided object is the AS structure of the the NP *the chicken leg*:

(6.6)
$$\left[\begin{array}{ll} \text{status:} & \left[\text{given: } +\right] \\ \text{spec:} & \left[\begin{array}{ll} \text{agr:} & \left[\begin{array}{l} \text{countable: } + \\ \text{number: sg} \end{array}\right] \\ \text{desc:} & \left[\begin{array}{ll} \text{head: chicken} & \\ \text{mod:} & \left[\text{head: leg}\right] \end{array}\right] \end{array}\right] \end{array}\right]$$

If the sentence is instead *30 minutes before you mash the potatoes, take out the chicken leg*, then the AS structure is as in (6.3) except that the CONNECTOR feature is replaced by:

(6.7)
$$\left[\begin{array}{ll} \text{connector:} & \left[\begin{array}{ll} \text{time:} & \left[\begin{array}{l} \text{unit: minute} \\ \text{quant: 30} \end{array}\right] \\ \text{type: before} & \end{array}\right] \end{array}\right]$$

Finally, consider what happens if the sentence is *Then, take out the chicken leg.* We analyse this sentence as though it is *After ϕ, take out the chicken leg* where ϕ is to be retrieved from context. The AS structure is thus:

(6.8)
$$\left[\begin{array}{ll} \text{connector:} & \left[\text{type: after}\right] \\ \text{subord: } [] & \\ \text{main: } \ldots & \end{array}\right]$$

where the ellided MAIN value is as in (6.5).

Example 2 We now consider sentences which contain adverbial clauses. For instance, the AS structure of the sentence *stir ϕ occasionally* is:

(6.9)
$$\begin{bmatrix} \text{sem:} & \begin{bmatrix} \text{mood:} & \text{imperative} \\ \text{tprop:} & \begin{bmatrix} \text{tense:} & \text{present} \\ \text{prop:} & \begin{bmatrix} \text{agent:} & \text{hearer} \\ \text{action:} & \text{stir} \\ \text{args:} & \begin{bmatrix} \text{obj:} & \ldots \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \text{frequency:} & \text{occasionally} \end{bmatrix} \end{bmatrix}$$

where the *frequency* feature indicates the frequency adverbial used in the clause, namely *occasionally.* Note that the ellided object in (6.9) is the AS structure of ϕ in *stir ϕ occasionally* [Dale 92]:

(6.10)
$$\begin{bmatrix} \text{status:} & \begin{bmatrix} \text{given:} & + \end{bmatrix} \\ \text{spec:} & [] \end{bmatrix}$$

Finally, consider the AS structure of a sentence with a *for*-adverbial. For instance, the AS structure of the sentence *steam the fish for 10 to 20 minutes* is:

(6.11)
$$\begin{bmatrix} \text{sem:} & \begin{bmatrix} \text{mood:} & \text{imperative} \\ \text{tprop:} & \begin{bmatrix} \text{tense:} & \text{present} \\ \text{prop:} & \begin{bmatrix} \text{agent:} & \text{hearer} \\ \text{action:} & \text{stem} \\ \text{args:} & \begin{bmatrix} \text{obj:} & \ldots \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \text{duration:} & \begin{bmatrix} \text{min:} & \begin{bmatrix} \text{unit:} & \text{minute} \\ \text{quant:} & 10 \end{bmatrix} \\ \text{max:} & \begin{bmatrix} \text{unit:} & \text{minute} \\ \text{quant:} & 20 \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

where the *duration* feature indicates the duration of carrying out the steaming action given in the *for*-adverbial. The ellided object in (6.11) is the object described by the definite NP *the fish.*

The Grammar For AS

Based on the discussion above, we can now give the grammar which defines the set of possible AS structures in TASTE in (6.12). The part of the grammar for the AS structures of objects described by NPs is taken from [Dale 92].

(6.12)

AS	::- sem | [connector] subord main | connector AND-CLAUSES
connector	::- [time] type
type	::- *before*, *after*, *progressive-adjunct*, *and* ...
sem	::- SEM
main	::- SEM | AND-CLAUSES
subord	::- SEM
AND-CLAUSE	::- ⟨a list of SEM structure⟩
SEM	::- [AS-NP] | [AS-CLAUSE]
AS-CLAUSE	::- mood tprop [ADV-CLAUSE]
AS-NP	::- status spec
ADV-CLAUSE	::- frequency | cardinal-count | duration
mood	::- *imperative*
tprop	::- tense prop
tense	::- *present* | *perfect* | *progressive*
prop	::- agent action args
agent	::- *hearer*
action	::- ⟨a lexical symbol⟩
args	::- [obj][indobj]...
obj	::- status spec
indobj	::- status spec
frequency	::- ⟨a frequency term⟩
cardinal-count	::- ⟨a number⟩
duration	::- min max
min	::- time
max	::- time
time	::- unit quant
unit	::- *minute* | *hour*
quant	::- ⟨a number⟩

status	::- given
given	::- + | −
spec	::- agr desc | ⟨a list of spec structure⟩
agr	::- countable [number]
countable	::- + | −
number	::- *sg* | *pl* | 1 . . . n
desc	::- head [mod] | spec_1 spec_2 | spec set
head	::- ⟨a lexical symbol⟩
mod	::- head [mod]
set	::- status spec

6.2.2 Recoverable Semantic Structure

The recoverable semantic structure (RS) is intermediate between the AS structure and the final KBE structure. It can be thought of as representing the semantic content of a sentence. In going from the AS level to the RS level, reasoning (such as access to real-world knowledge) might be needed (more on this below). Note that no reasoning is required in building up the AS structure.

The mapping from AS structure to RS structure is many-to-one, i.e. many AS structures can correspond to one RS structure.

To illustrate the RS structure, we will contrast it with the AS structure to bring out their differences. This will also indicate that the mapping from AS to RS is not purely structural but may involve reasoning:

1. The propositional content of the RS structure is similar to a predicate-argument structure. For instance, the *propositional* content in the RS structure of *chop the potatoes* is:

   (6.13)
$$\begin{bmatrix} \text{prop:} & \begin{bmatrix} \text{action: chop} & \\ \text{participants:} & \begin{bmatrix} \text{agent: hearer} \\ \text{obj: } \ldots \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

   where the ellided object is the RS structure of the object described by the NP *the potatoes* [Dale 92]:

(6.14)
$$\begin{bmatrix} \text{status:} & \begin{bmatrix}\text{given: } +\end{bmatrix} \\ \text{spec:} & \begin{bmatrix} \text{agr:} & \begin{bmatrix}\text{countable: } + \\ \text{number: pl}\end{bmatrix} \\ \text{type:} & \begin{bmatrix}\text{category: potato}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

Compare (6.13) with the *pro*positional content in the AS structure of *chop the potatoes* given in (6.1):

(6.15)
$$\begin{bmatrix} \text{prop:} & \begin{bmatrix} \text{agent: hearer} \\ \text{action: chop} \\ \text{args: } \begin{bmatrix}\text{obj: } \ldots\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

which mirrors the standard subject-predicate form of the clause. Thus, in going from AS structure to RS structure, there is a rearrangement of information: for instance, the AGENT and ARG attribute in AS are now moved to the PARTICIPANTS attribute in RS.

2. Given NPs like *some sea salt* and *a celery stick*, the RS structure indicates correctly the modifier and the modified. For example, in *some sea salt*, *salt* is modified by *sea* while in *a celery stick*, *celery* is modified by *stick*. To derive such distinction, reasoning is needed during the mapping from the AS structure to the RS structure since in the AS structure the relationship between the modifier and the modified is simply based on the syntactic position of the word. For instance, in the AS structure of the NPs *some sea salt* and *a celery stick*, the noun to the left is taken as modifying the noun to the right. Consider the AS structure of *some sea salt*:

$$\left[\text{sem:}\left[\begin{array}{ll}\text{status:} & \left[\text{given: } -\right]\\ \text{spec:} & \left[\begin{array}{ll}\text{agr:} & \left[\begin{array}{l}\text{countable: } +\\ \text{number: pl}\end{array}\right]\\ \text{desc:} & \left[\begin{array}{ll}\text{mod:} & \left[\text{head: sea}\right]\\ \text{head:} & \text{salt}\end{array}\right]\end{array}\right]\end{array}\right]\right] \tag{6.16}$$

which is similar to the AS structure of *a celery stick*:

$$\left[\text{sem:}\left[\begin{array}{ll}\text{status:} & \left[\text{given: } -\right]\\ \text{spec:} & \left[\begin{array}{ll}\text{agr:} & \left[\begin{array}{l}\text{countable: } +\\ \text{number: pl}\end{array}\right]\\ \text{desc:} & \left[\begin{array}{ll}\text{mod:} & \left[\text{head: celery}\right]\\ \text{head:} & \text{stick}\end{array}\right]\end{array}\right]\end{array}\right]\right] \tag{6.17}$$

However, their RS structures are different. To derive the RS structure of *some sea salt* from its AS structure, the mapping needs access to real-world knowledge that sea is a source for obtaining salt. The RS structure is as follows:

$$\left[\text{sem:}\left[\begin{array}{ll}\text{status:} & \left[\text{given: } -\right]\\ \text{spec:} & \left[\begin{array}{ll}\text{agr:} & \left[\text{countable: } -\right]\\ \text{type:} & \left[\begin{array}{ll}\text{subst:} & \text{salt-matter}\\ \text{props:} & \left[\text{come-from: sea}\right]\end{array}\right]\end{array}\right]\end{array}\right]\right] \tag{6.18}$$

where the lexical symbols *salt* and *sea* have been 'disambiguated' and *sea* is taken as a property in the RS structure. Compare this RS structure with that for the NP *a celery stick*:

$$(6.19)\quad \begin{bmatrix} \text{sem:} & \begin{bmatrix} \text{status:} & \begin{bmatrix} \text{given: } - \end{bmatrix} \\ \text{spec:} & \begin{bmatrix} \text{agr:} & \begin{bmatrix} \text{number: sg} \\ \text{countable: } + \end{bmatrix} \\ \text{type:} & \begin{bmatrix} \text{subst: celery-matter} \\ \text{props: } \begin{bmatrix} \text{shape: stick} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

where the lexical symbols *celery* and *stick* have been disambiguated and *stick* is taken as a property indicating the shape of the object.

3. Given a partitive NP of the form NP_1 *of* NP_2, the RS structure indicates the relation between NP_1 and NP_2 in the AS structure. As noted in [Dale 92], a partitive NP such as *the skin of the avocado* describes an entity which is derived from (usually, is *part of*) some other entity. Consider, for instance, the AS structure of *the skin of the avocado*:

$$(6.20)\quad \begin{bmatrix} \text{sem:} & \begin{bmatrix} \text{status:} & \begin{bmatrix} \text{given: } + \end{bmatrix} \\ \text{spec:} & \begin{bmatrix} \text{agr:} & \begin{bmatrix} \text{countable: } + \\ \text{number: sg} \end{bmatrix} \\ \text{desc:} & \begin{bmatrix} \text{set: } \ldots \\ \text{spec: } \ldots \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

where the ellided SET value is given in (6.21) and the ellided SPEC value is that given in (6.22).

$$(6.21)\quad \begin{bmatrix} \text{status:} & \begin{bmatrix} \text{given: } + \end{bmatrix} \\ \text{spec:} & \begin{bmatrix} \text{agr:} & \begin{bmatrix} \text{countable: } + \\ \text{number: sg} \end{bmatrix} \\ \text{desc:} & \begin{bmatrix} \text{head: avocado} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$(6.22)\quad \begin{bmatrix} \text{status:} & \begin{bmatrix}\text{given: } +\end{bmatrix} \\ \text{spec:} & \begin{bmatrix} \text{agr:} & \begin{bmatrix}\text{countable: } + \\ \text{number: sg}\end{bmatrix} \\ \text{desc:} & \begin{bmatrix}\text{head: skin}\end{bmatrix} \end{bmatrix} \end{bmatrix}$$

In the mapping from the AS structure to the RS structure, we determine whether the set-part relation holds between *the skin* and *the avocado* (see section 6.3). For the NP *the skin of the avocado*, the set-part relation holds and therefore the RS structure is as follows:

$$(6.23)\quad \begin{bmatrix} \text{sem:} & \begin{bmatrix} \text{status:} & \begin{bmatrix}\text{given: } +\end{bmatrix} \\ \text{spec:} & \begin{bmatrix} \text{agr:} & \begin{bmatrix}\text{countable: } + \\ \text{number: sg}\end{bmatrix} \\ \text{part:} & \ldots \\ \text{set:} & \ldots \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

where the ellided *part* and *set* value is given in (6.24) and (6.25) respectively.

$$(6.24)\quad \begin{bmatrix} \text{status:} & \begin{bmatrix}\text{given: } +\end{bmatrix} \\ \text{spec:} & \begin{bmatrix} \text{agr:} & \begin{bmatrix}\text{countable: } + \\ \text{number: sg}\end{bmatrix} \\ \text{type:} & \begin{bmatrix}\text{subst: skin-matter}\end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$(6.25)\quad \begin{bmatrix} \text{status:} & \begin{bmatrix}\text{given: } +\end{bmatrix} \\ \text{spec:} & \begin{bmatrix} \text{agr:} & \begin{bmatrix}\text{countable: } + \\ \text{number: sg}\end{bmatrix} \\ \text{type:} & \begin{bmatrix}\text{subst: avocado-matter}\end{bmatrix} \end{bmatrix} \end{bmatrix}$$

4. We analyse a pseudo-partitive NP of the form NP_1 *of* N_1 as describing a quantity of some substance. Consider for instance the pseudo-partitive NP *a stick of celery.*

The AS structure of this NP is as given in (6.26).

$$(6.26)\quad \left[\text{sem}: \left[\begin{array}{ll} \text{status}: & \left[\text{given}: -\right] \\ \text{spec}: & \left[\begin{array}{ll} \text{agr}: & \left[\begin{array}{l}\text{number: sg}\\ \text{countable}: +\end{array}\right] \\ \text{desc}: & \left[\begin{array}{l} \text{spec}_1: \ldots \\ \text{spec}_2: \ldots\end{array}\right]\end{array}\right]\end{array}\right]\right]$$

where the ellided *spec*$_1$ and *spec*$_2$ is given in (6.27) and (6.28) respectively:

$$(6.27)\quad \left[\begin{array}{ll} \text{agr}: & \left[\begin{array}{l}\text{number: sg}\\ \text{countable}: +\end{array}\right] \\ \text{desc}: & \left[\text{head: stick}\right]\end{array}\right]$$

$$(6.28)\quad \left[\begin{array}{ll} \text{agr}: & \left[\begin{array}{l}\text{number: sg}\\ \text{countable}: +\end{array}\right] \\ \text{desc}: & \left[\text{head: celery}\right]\end{array}\right]$$

In the mapping from the AS structure to the RS structure, we determine that the NP *a stick of celery* is describing a quantity (specifically one) of celery which is packaged in the shape of a stick. The corresponding RS structure is as in (6.19).

The Grammar for RS

We can now give the grammar which defines the set of possible RS structures. This is found in (6.29). The part of the grammar which defines the RS structures of objects described by NPs is taken from [Dale 92].

(6.29)	RS	::- sem | [connector] subord main | connector AND-CLAUSES
	connector	::- [time] type
	type	::- *before*, *after*, *progressive-adjunct*, *and* ...
	sem	::- SEM
	subord	::- SEM

main	::- SEM | AND-CLAUSES
AND-CLAUSES	::- ⟨a list of SEM structure⟩
SEM	::- [RS-NP] | [RS-CLAUSE]
RS-NP	::- status spec
RS-CLAUSE	::- illocforce tprop [ADV]
ADV	::- cardinality | duration
cardinality	::- ⟨a frequency term⟩ | ⟨a number⟩
duration	::- min max
min	::- time
max	::- time
illocforce	::- *requesting*, *informing*, ...
tprop	::- tense prop
tense	::- *present* | *perfect* | *progressive*
prop	::- action participants
action	::- ⟨a lexical symbol⟩
participants	::- agent [obj] [indobj]...
agent	::- hearer
obj	::- status spec
indobj	::- status spec
time	::- unit quant
unit	::- *minute*, *hour* ...
quant	::- ⟨a number⟩
status	::- given
given	::- + | −
spec	::- agr type | agr quantity category | agr part set
agr	::- [number] countable
number	::- *sg* | *pl* | 1 ... n
countable	::- + | −
type	::- category [props] | subst [props]
category	::- ⟨a lexical symbol⟩
subst	::- ⟨a substance⟩
props	::- $\langle a_1, v_1 \rangle, \ldots \langle a_n, v_n \rangle$
quantity	::- unit quant
part	::- status spec
set	::- status spec
quant	::- ⟨a number⟩
unit	::- *pounds*, *gram*, ...

6.2.3 Knowledge Base Structure

The knowledge base structure (KBE) is the final representation of the situation underlying a sentence. It can be thought of as the feature structure equivalent of the FOL representation presented in chapter 3.

The mapping from a RS structure to a KBE structure is many-to-one. The main processings carried out to the RS structure to obtain the KBE structure are as follows:

- We resolve NPs to physobj and sentences to situations. This is carried out simultaneously with the lexical disambiguation of any verb and nouns within the same framework of constraint satisfaction.
- We determine the participants in the KBE.

If a sentence or NP describes a new entity, a new index is assigned to the entity and appropriate information predicated of this index. We will now show the KBE structure of some illustrative sentences.

Example 1

We consider the KBE structure of a simple example *add the salt to the soup.* The FOL representation of this sentence is:

(6.30) $\text{event}(e_1) \wedge \text{structure}(e_1, \text{simple}) \wedge \text{occurs}(s_0, s_1, e_1) \wedge \text{telic}(e_1, +) \wedge$
$\text{action}(e_1, \text{adding}) \wedge \text{agent}(e_1, \text{hearer}) \wedge$
$\text{addendum}(e_1, x_1) \wedge \text{base}(e_1, x_2)$

and the corresponding KBE structure is:

$$(6.31)\quad \begin{bmatrix} \text{index: } e_1 \\ \text{type: event} \\ \text{occurs: } \begin{bmatrix} \text{begin: } s_0 \\ \text{end: } s_1 \end{bmatrix} \\ \text{spec: } \begin{bmatrix} \text{structure: simple} \\ \text{action: adding} \\ \text{telic: } + \\ \text{participants: } \begin{bmatrix} \text{agent: hearer} \\ \text{in: } \begin{bmatrix} \text{addendum: } x_1 \\ \text{base: } x_2 \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

where x_1 and x_2 are the indices of the KBE structure corresponding to the object described by *the salt* and *the soup* respectively. Note that the index of an entity is essentially extensional whereas the SPECification is essentially intensional. No two entities can have the same index although they may have the same specification. We will now show the KBE structure of *the salt* (that for *the soup* is similar):

$$(6.32)\quad \begin{bmatrix} \text{index: } x_1 \\ \text{state: } s_0 \\ \text{spec: } \begin{bmatrix} \text{structure: mass} \\ \text{substance: salt-matter} \end{bmatrix} \end{bmatrix}$$

Example (6.31) is that of an event. To be complete, we show the KBE structure of a typical state, expressed by the sentence *the meat is tender.* Recall from chapter 4 that the FOL representation of *the meat is tender* is :

$$(6.33)\quad \text{state}(s_1) \wedge \text{hold}(s_1, \text{tender}(x_1, +))$$

where x_1 is the object described by *the meat.* The corresponding feature structure is:

$$(6.34)\quad \begin{bmatrix} \text{index: } s_1 \\ \text{type: state} \\ \text{spec: } \begin{bmatrix} \text{arg: } x_1 \\ \text{tender: } + \end{bmatrix} \end{bmatrix}$$

Example 2

We consider the KBE structure of *soak and drain some beans* whose FOL representation is

(6.35) $\text{event}(e_1) \wedge \text{structure}(e_1, \text{composite}) \wedge \text{occurs}(s_0, s_1, e_1) \wedge$
$\text{constituent}(e_1, [\ e_2, e_3\ ]) \wedge$
$[\ \text{event}(e_2) \wedge \text{structure}(e_2, \text{simple}) \wedge \text{action}(e_2, \text{soaking}) \wedge$
$\text{telic}(e_2, +) \wedge \text{agent}(e_2, \text{hearer}) \wedge \text{obj}(e_2, x_1)\ ] \wedge$
$[\ \text{event}(e_3) \wedge \text{structure}(e_3, \text{simple}) \wedge \text{action}(e_3, \text{draining}) \wedge$
$\text{telic}(e_3, +) \wedge \text{agent}(e_3, \text{hearer}) \wedge \text{obj}(e_3, x_1)\ ]$

The corresponding feature structure representation is:

(6.36)
$$\begin{bmatrix} \text{index: } e_1 \\ \text{type: event} \\ \text{occurs: } \begin{bmatrix} \text{begin: } s_0 \\ \text{end: } s_1 \end{bmatrix} \\ \text{spec: } \begin{bmatrix} \text{constituents: } [e_2, e_3] \end{bmatrix} \end{bmatrix}$$

where e_2 and e_3 are the indices of the KBE structures corresponding to the soaking event and the draining event respectively.

Example 3

We now consider repetitive events. Consider first the event described by *stir occasionally* whose FOL representation is:

(6.37) $[\ \text{event}(e_1) \wedge \text{structure}(e_1, \text{repetition}) \wedge \text{no-of-repetition}(e_1, \text{occasionally}) \wedge$
$[\ \forall e\ \text{element}(e_1, e) \supset$
$\text{event}(e) \wedge \text{structure}(e, \text{simple}) \wedge \text{telic}(e, -) \wedge$
$\text{action}(e, \text{stirring}) \wedge \text{agent}(e, \text{hearer}) \wedge \text{obj}(e, x_1)\ ]\ ]$

The equivalent KBE structure is:

(6.38)

$$\left[\begin{array}{ll}
\text{index: } e_1 & \\
\text{type: event} & \\
\text{occurs:} & \left[\begin{array}{l}\text{begin: } s_0 \\ \text{end: } s_1\end{array}\right] \\
\text{spec:} & \left[\begin{array}{ll}
\text{structure: repetition} & \\
\text{cardinality: occasionally} & \\
\text{elements:} & \left[\begin{array}{ll}
\text{structure: simple} & \\
\text{action: stirring} & \\
\text{telic: } - & \\
\text{participants:} & \left[\begin{array}{ll}\text{agent: hearer} & \\ \text{in:} & \left[\text{obj: } x_1\right]\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right]$$

where x_1 is the index of the KBE structure of the object which is being stirred. Recall from chapter 4 that x_1 is the index of the current center in the discourse model since the NP in *stir occasionally* is ellided.

Now, consider the case where the number of repetitions is specified implicitly as in *beat the eggs one at a time*. The FOL representation is:

(6.39)

$$\begin{array}{l}
\text{event}(e_1) \wedge \text{structure}(e_1, \text{repetition}) \wedge \text{occur}(s_0, s_1, e_1) \wedge \\
[\ \forall x\ \text{egg}(x) \wedge \text{element-of}(x, x_1) \supset \\
\quad \exists e\ \text{element}(e_1, e) \wedge \text{event}(e) \wedge \\
\quad\quad \text{structure}(e, \text{simple}) \wedge \text{telic}(e, +) \wedge \\
\quad\quad \text{action}(e, \text{beating}) \wedge \text{agent}(e, \text{hearer}) \wedge \\
\quad\quad \text{obj}(e, x)\ ]
\end{array}$$

where x_1 is the object described by the NP *the eggs*. The corresponding KBE structure is:

(6.40)
$$\left[\begin{array}{ll}
\text{index:} & e_1 \\
\text{type:} & \text{event} \\
\text{occurs:} & \left[\begin{array}{l}\text{begin: } s_0 \\ \text{end: } s_1\end{array}\right] \\
\text{spec:} & \left[\begin{array}{ll}
\text{structure:} & \text{repetition} \\
\text{elements:} & \left[\begin{array}{ll}
\text{structure:} & \text{simple} \\
\text{action:} & \text{beating} \\
\text{telic:} & + \\
\text{participants:} & \left[\begin{array}{l}\text{agent: hearer} \\ \text{in: } \ldots\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right]$$

where the ellided value is the SPEC of the object involved in the beating action and is given below:

(6.41)
$$\left[\begin{array}{ll}
\text{structure:} & \text{individual} \\
\text{substance:} & \text{egg-matter} \\
\text{packaging:} & \left[\begin{array}{l}\text{shape: egg} \\ \text{size: regular}\end{array}\right] \\
\text{element-of:} & x_1
\end{array}\right]$$

The index x_1 in (6.41) is that of the KBE structure of the object described by the NP *the eggs.*

Observe that in (6.40), only the intension of the object that is being beaten is given in (6.41). If the beating event is to be decomposed (see chapter 4), then we will work out the actual specific events that are to be repeated in e_1. In each of these individual beating events, the object to be beaten is an individual egg whose KBE structure can be created from the specification given in (6.41).

The Grammar For KBE

The grammar defining the set of possible KBE structures is shown in (6.42).

(6.42) KBE ::- SITUATION | PHYSOBJ

SITUATION	::- index type [occurs] spec
PHYSOBJ	::- index state spec
index	::- e_i | x_i
state	::- s_i
type	::- *event* | *state*
occurs	::- begin end
begin	::- s_i
end	::- s_i
spec	::- SIM-SPEC | COM-SPEC | REP-SPEC | PROP-SPEC |
	INDIV-SPEC [OTHER-SPEC] |
	SET-MASS-SPEC [OTHER-SPEC]
SIM-SPEC	::- structure telic action participants
COM-SPEC	::- structure constituents
REP-SPEC	::- structure [cardinality] elements
PROP-SPEC	::- arg PROP*
PROPS	::- $\langle a_1, v_1 \rangle \ldots \langle a_n, v_n \rangle$
structure	::- *simple* | *composite* | *repetition*
telic	::- + | −
cardinality	::- ⟨frequency adverbial⟩ | ⟨a number⟩
constituents	::- ⟨a list of indices⟩
elements	::- SIM-SPEC | COM-SPEC
substance	::- ⟨a substance in the SUBSTANCE kb⟩
participants	::- agent [object] ...
agent	::- hearer
object	::- ⟨an index⟩ | INDIV-SPEC [OTHER-SPEC] |
	SET-MASS-SPEC [OTHER-SPEC]
INDIV-SPEC	::- structure substance packaging PROP*
SET-MASS-SPEC	::- structure substance PROPS*
	structure cardinality elements |
	structure quantity elements |
	structure quantity constituents
OTHER-SPEC	::- part-of | element-of | subset-of
PROPS	::- $\langle a_1, v_1 \rangle \ldots \langle a_n, v_n \rangle$
structure	::- *individual* | *mass* | *set*
action	::- ⟨an action in the action kb⟩
quantity	::- unit number | min max
unit	::- *pound*, *gram* ...
number	::- 1 ... n
min	::- unit number
max	::- unit number

packaging	::- shape size
shape	::- ⟨an atomic semantic category⟩
size	::- ⟨an atomic semantic category⟩
cardinality	::- 1 . . . n
constituents	::- ⟨a list of indices⟩
part-of	::- ⟨an index⟩
element-of	::- ⟨an index⟩
subset-of	::- ⟨an index⟩
elements	::- INDIV-SPEC [OTHER-SPEC] | SET-MASS-SPEC [OTHER-SPEC]

6.2.4 The QTS

The QTS can be thought of as encoding the relations between the KBEs. We represent the relations between the KBEs as feature structures defined by the simple grammar below:

(6.43)

RELATION	::- index spec
index	::- r_i
spec	::- relation sit1 sit2
sit1	::- ⟨an index of a situation⟩
sit2	::- ⟨an index of a situation⟩
relation	::- ⟨a possible relation between situations⟩

For example, the feature structure corresponding to

(6.44) enable(e_1, e_2)

is as follows:

(6.45)
$$\begin{bmatrix} \text{index: } r_1 & \\ \text{spec:} & \begin{bmatrix} \text{relation: enable} \\ \text{sit1: } e_1 \\ \text{sit2: } e_2 \end{bmatrix} \end{bmatrix}$$

where e_1 and e_2 are indices.

Notice that the QTS can now be defined by the set of r_i where r_i is the index of the feature structure associated with each possible relation between situations. This means that in traversing the QTS (see chapter 5), we can simply iterate over the r_i. The fact that the r_i is also a feature structure means that the same reasoning system for mapping a sentence into its KBE structure can be applied to reason over the r_i.

We will now describe the reasoning system employed in TASTE.

6.3 The Rule System

This section describes the reasoning system implemented in TASTE for carrying out the following tasks:

1. to map a sentence into its KBE structure;
2. to determine the set of previous events whose relations with the current situation can be deduced;
3. to determine the relations between situations; and
4. to determine the qualitative IA relations between the time intervals of the situations described in the text.

The reasoning system uses rules as its main mechanism for representing procedural knowledge (see Appendix B for the BNF of our rule language). Consider, for instance, the rule which is invoked when mapping the *spec* feature of the AS structure of a partitive NP into the *spec* feature of the corresponding RS structure. To ground the discussion, consider the *spec* feature of the partitive NP *the skin of the avocado* extracted from (6.20):

$$(6.46) \quad \begin{bmatrix} \text{agr:} & \begin{bmatrix} \text{countable: } + \\ \text{number: sg} \end{bmatrix} \\ \text{desc:} & \begin{bmatrix} \text{set: } \ldots \\ \text{spec: } \ldots \end{bmatrix} \end{bmatrix}$$

where the ellided *set* and *spec* feature is given respectively in (6.21) and (6.22) above and reproduced below:

$$(6.21) \quad \begin{bmatrix} \text{status:} & \begin{bmatrix} \text{given: } + \end{bmatrix} \\ \text{spec:} & \begin{bmatrix} \text{agr:} & \begin{bmatrix} \text{countable: } + \\ \text{number: sg} \end{bmatrix} \\ \text{desc:} & \begin{bmatrix} \text{head: avocado} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$(6.22) \quad \begin{bmatrix} \text{status:} & \begin{bmatrix} \text{given: } + \end{bmatrix} \\ \text{spec:} & \begin{bmatrix} \text{agr:} & \begin{bmatrix} \text{countable: } + \\ \text{number: sg} \end{bmatrix} \\ \text{desc:} & \begin{bmatrix} \text{head: skin} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

This is mapped by the rule into the following *spec* feature of the RS structure (extracted from (6.23)):

$$(6.47)\quad \begin{bmatrix} \text{set:} & \begin{bmatrix} \text{status:} & \begin{bmatrix}\text{given: } +\end{bmatrix} \\ \text{spec:} & \begin{bmatrix} \text{agr:} & \begin{bmatrix}\text{countable: } + \\ \text{number: sg}\end{bmatrix} \\ \text{type:} & \begin{bmatrix}\text{subst: avocado-matter}\end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \text{part:} & \begin{bmatrix} \text{agr:} & \begin{bmatrix}\text{countable: } + \\ \text{number: sg}\end{bmatrix} \\ \text{type:} & \begin{bmatrix}\text{subst: skin-matter}\end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Now, if we let X be the identifier for the feature structure given in (6.46), then the rule which maps (6.46) into (6.47) is as follows:

```
(6.48)  ?head1 = ⟨X desc set spec desc head⟩
        ?head2 = ⟨X desc spec desc head⟩
        (constraint-find ?subst1
                         ?subst2
                         :where
                         (get-concepts ?head1)
                         (get-concepts ?head2)
                         :constraints
                         (frame-query (part-of ?subst2 ?subst1)))
        →
        ⟨Y set status given⟩ = ⟨X desc set status given⟩
        ⟨Y set spec agr⟩ = ⟨X desc set spec agr⟩
        ⟨Y set spec type subst⟩ = ?subst1
        ⟨Y part agr⟩ = ⟨X desc spec spec agr⟩
        ⟨Y part type subst⟩ = ?subst2
```

where symbols preceded by '?' such as ?head1 are variables and the symbol Y is the identifier of the feature structure output by the rule. The meanings of the various clauses in the LHS and RHS of the rule are as follows:

- The first and second clause assign ?head1 and ?head2 to that part of the feature structure X picked out by the path indicated. If no such paths exist, the clauses fail. For our current example, ?head1 is bound to the lexical symbol *avocado* and ?head2 to the lexical symbol *skin*.

- The third clause invokes a constraint-satisfaction process (see chapter 4) via the CONSTRAINT-FIND clause (see Appendix B) to solve the CSP involving the variables ?subst1 and ?subst2. The domain of these variables are given respectively after the *:where* keyword. The possible values of ?head1 and ?head2 are given by the GET-CONCEPTS function which returns the set of possible concepts in the substance knowledge base (see Appendix C for how such a knowledge base is defined) that can be described using ?head1 and ?head2. The constraint is derived from the possible relations between ?subst1 and ?subst2 suggested by the syntactic structure of the partitive NP *the skin of the avocado.* As noted, a partitive NP describes an entity which is derived from (usually, is *part of*) some other entity. Therefore, in our work, we only consider the set-part relation and thus the part-of constraint in (6.48). In order to verify this constraint, a check is made with the hierarchical substance knowledge base via the FRAME-QUERY function. The clause fails if constraint satisfaction fails. For our current example, the clause succeeds since constraint satisfaction successfully assigns ?subst1 to avocado-matter and ?subst2 to skin-matter. This is because in the substance knowledge base, skin-matter is indicated to be a part of avocado-matter.

- The various actions on the right hand side of the rule build up the feature structure to be output by the rule. Each action in rule (6.48) assigns the feature structure picked out by the path on the right hand side of the '=' symbol to be the value of the path on the left hand side. The resultant structure is that shown in (6.47).

6.3.1 Organising Rules into Rulesets

TASTE organises rules into rulesets (see Appendix B for the BNF of our ruleset language). Rulesets collect together rules that either reason over the same part of a feature structure or which perform the same task. For example, rules which suggest

Name	: KBE-SENT-TOP
Input	: ⟨RS⟩
Output	: ⟨KBE⟩
Instructions	: ()
Strategy	: First-Tried-Succeed
Rules	

1. ⟨RS connector⟩ →
   ⟨KBE⟩ = (invoke-ruleset KBE-COMP-CL ⟨RS⟩)
2. (default) →
   ⟨KBE⟩ = (invoke-ruleset KBE-CL ⟨RS⟩)

Figure 6.3: The KBE-SENT-TOP ruleset

referents for pronouns and ellided NPs are grouped together. This gives modularity to the system and any ruleset can be easily replaced with another one without affecting the rest of the system. For example, the current ruleset for pronoun resolution is a simple one based on the model proposed in [Dale 92]. This ruleset can be replaced with one based on a more complicated model, if required. An example of a ruleset is given in figure 6.3.

The various components of a ruleset are as follows:

- The NAME component specifies the name of the ruleset.
- The INPUT component specifies the feature structure that the ruleset should look at.
- The OUTPUT component specifies the feature structure to be output by the ruleset.
- The INSTRUCTIONS component specifies a list of functions to be called when the ruleset is successful in producing an output feature structure. These functions can either keep track of the output feature structure or pass control to other modules. For instance, the ruleset responsible for mapping the RS structure of an indefinite NP into its KBE structure is told to add the index of the KBE to the

discourse model (since an indefinite NP describes a new object).

- The STRATEGY component specifies a rule firing strategy. Currently, we have the FIRST-TRIED-SUCCEED strategy where the first rule in the ruleset that fires causes the ruleset to report success and returns the output given in the rule. Subsequent rules are then not tried.

- The RULES component contains the rules collected together in the ruleset.

The KBE-SENT-TOP ruleset given in figure 6.3 is in fact the top level ruleset invoked by the INCREMENTAL-BUILD-QTS module. The ruleset thus basically implements the algorithm which is given in figure 4.2 and continued in figure 4.3 and figure 4.4. The second rule in the ruleset handles the case in the algorithm of figure 4.2 where the RS structure is that of a simple sentence. This rule invokes the KBE-CL ruleset to build up the KBE structure of the simple sentence (which we also call a clause). That is, the KBE-CL ruleset implements the BUILD-SITUATION-REP algorithm (see section 6.4.3). On the other hand, the first rule in the ruleset handles the case where the RS structure is that of a sentence containing more than one clause. It invokes the KBE-COMP-CL ruleset which in turn invokes other appropriate rulesets depending on the type of the connector linking the clauses.

Based on the discussion above, we note that a ruleset can invoke other rulesets to carry out its task (via the INVOKE-RULESET function). This means that we can define a hierarchy of rulesets where higher level rulesets invoke the lower level ones. Typically, the higher level ruleset invokes lower level rulesets to look at subparts of the feature structure they are entrusted with.

The ruleset hierarchy enables us to define a simple success-failure reporting strategy. If a lower level ruleset is successful, it reports its success to the higher level ruleset that calls it. On the other hand, if it not successful, the lower level ruleset reports the failure directly to the calling module. This success-failure reporting strategy is depicted graphically in figure 6.4.

What is the advantage of the scheme presented above ? Since lower level rulesets typically reason over subparts of the feature structure the higher level ruleset is entrusted

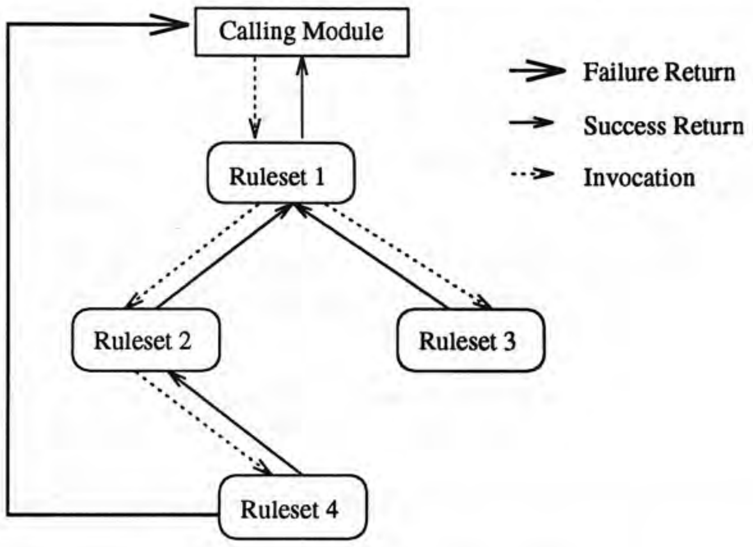


Figure 6.4: The success-failure reporting strategy

with, our success-failure reporting strategy reflects the notion that in going from one feature structure to another feature structure, the production of a meaning representation relies on the meaning representation of the subphrases [Gazdar & Mellish 89]. In addition, an advantage of this scheme is that failure can be immediately localised to a particular ruleset. For example, given the RS structure of *chop the juice*, the KBE-SENT-TOP ruleset invokes the KBE-CL ruleset (rule 2 in figure 6.3). The KBE-CL ruleset (shown in figure 6.5) will itself invoke the KBE-CL-INFO ruleset which is responsible for disambiguating the verb and resolving NPs in the clause. For this current case, type restriction failure would cause the KBE-CL-INFO ruleset to report failure directly to the INCREMENTAL-BUILD-QTS module.

6.3.2 Summary

In the section above, we have presented the rule reasoning system employed in TASTE. In this system, rules are organised into rulesets which are themselves organised into an invocation hierarchy. Using such organisation, TASTE is highly modular so that replacing one ruleset with another is less likely to affect the overall system.

Name	: KBE-CL
Input	: ⟨X⟩
Output	: ⟨Y⟩
Strategy	: First-Tried-Succeed
Rules:	

1. (invoke-ruleset KBE-CL-INFO ⟨X tprop⟩) = ?cl-info →
   ⟨Y⟩ = (invoke-ruleset DET-CL-GIVEN-NEW ?cl-info)

Figure 6.5: The KBE-CL ruleset

The BNFs of the rules and rulesets language are given in Appendix B while that for defining the object substances and actions are given in Appendix C.

6.4 Natural Language Processing

In this section, we discuss the various stages involved in mapping a sentence into its final KBE. These stages are:

1. Sentence → AS
2. AS → RS
3. RS → KBE

The first two stages occur in the BUILD-INTERMEDIATE-REP module while the third stage occurs in the INCREMENTAL-BUILD-QTS module. The BUILD-INTERMEDIATE-REP module consists of two modules:

- the parsing module; and
- the AS→RS module.

6.4.1 From Sentence to AS Structure

This is carried out in the PARSING module. A unification-based chart parser [Gent 87] is used together with a grammar and lexicon written in PATR notation to produce the initial AS structure.

In the TASTE system, we maintain a clear distinction between the information stored in the lexicon and the information stored in the more conceptual knowledge base. For us, information like selectional restrictions is better thought of as type restriction and maintained in the conceptual knowledge base. The lexicon then contains only syntactic information. For instance, an example of a lexical entry in TASTE is as follows:

(6.49) Put
$\langle$Put syn cat$\rangle$ = V
$\langle$Put syn tense$\rangle$ = present
$\langle$Put syn type$\rangle$ = finite
$\langle$Put arg1 cat$\rangle$ = NP
$\langle$Put arg2 cat$\rangle$ = PP
$\langle$Put arg2 pform$\rangle$ = in
$\langle$Put lex$\rangle$ = putting-in

Maintaining a clear distinction between lexicon and conceptual knowledge base means that there is no duplication of information in TASTE: in NLP systems that store semantic information for selectional restrictions in the lexicon, there is duplication of such information in the conceptual knowledge base. Moreover, as pointed out in [Gazdar & Mellish 89, page 312], the use of features to encode semantic information in the grammar and lexicon is a relatively unprincipled technique.

To complete the above discussion, we will briefly describe how the AS structure can be built up by unification for the NP *the skin of the avocado.* The grammar and lexicon needed for this example is given in figure 6.6.

It is simplest to think of the AS structure of *the skin of the avocado* as being built bottom-up. The steps involved are as follows:

1. First, the phrase *the skin* is analysed by Rule 2 and Rule 3 and the corresponding

Rule 1

$\text{NP}_1 \rightarrow \text{NP}_2$ PP[+of]
$\langle \text{NP}_1$ sem status$\rangle = \langle \text{NP}_2$ sem status$\rangle$
$\langle \text{NP}_1$ sem spec agr$\rangle = \langle \text{NP}_2$ syn agr$\rangle$
$\langle \text{NP}_1$ sem spec desc spec$\rangle = \langle \text{NP}_2$ sem spec$\rangle$
$\langle \text{NP}_1$ sem spec desc set$\rangle = \langle$PP sem$\rangle$

Rule 2

NP $\rightarrow$ DET N_2
$\langle$NP sem status$\rangle = \langle$DET sem status$\rangle$
$\langle$NP sem spec$\rangle = \langle \text{N}_2$ sem spec$\rangle$

Rule 3

$\text{N}_2 \rightarrow$ N
$\langle \text{N}_2$ sem spec desc head$\rangle = \langle$N lex$\rangle$
$\langle \text{N}_2$ sem spec agr$\rangle = \langle$N syn agr$\rangle$

Rule 4

PP $\rightarrow$ prep NP
$\langle$PP sem$\rangle = \langle$NP sem$\rangle$

skin

⟨syn cat⟩ = noun
⟨syn agr countable⟩ = +
⟨syn agr number⟩ = sg
⟨lex⟩ = skin

avocado

⟨syn cat⟩ = noun
⟨syn agr countable⟩ = +
⟨syn agr number⟩ = sg
⟨lex⟩ = avocado

the

⟨syn cat⟩ = DET
⟨sem status given⟩ = +
⟨lex⟩ = the

of

⟨syn cat⟩ = prep
⟨lex⟩ = of

Figure 6.6: A grammar and lexicon fragment used in TASTE

AS structure for this phrase is constructed from the path equations as given in Rule 2:

$$\left[\begin{array}{ll} \text{sem:} & \left[\begin{array}{ll} \text{status:} & \left[\text{given: } +\right] \\ \text{spec:} & \left[\begin{array}{ll} \text{agr:} & \left[\begin{array}{l}\text{countable: } + \\ \text{number: sg}\end{array}\right] \\ \text{desc:} & \left[\text{head: skin}\right]\end{array}\right]\end{array}\right]\end{array}\right] \tag{6.22}$$

2. Next, the PP *of the avocado* is analysed by Rule 4 and the feature structure corresponding to this phrase is built up from the NP *the avocado* which is itself analysed by Rule 2 and Rule 3. This results in the following feature structure for the phrase as specified in the path equations of Rule 4:

$$\left[\begin{array}{ll} \text{sem:} & \left[\begin{array}{ll} \text{status:} & \left[\text{given: } +\right] \\ \text{spec:} & \left[\begin{array}{ll} \text{agr:} & \left[\begin{array}{l}\text{countable: } + \\ \text{number: sg}\end{array}\right] \\ \text{desc:} & \left[\text{head: avocado}\right]\end{array}\right]\end{array}\right]\end{array}\right] \tag{6.21}$$

3. Now, Rule 1 combines the above two feature structures into the final AS structure for the whole NP:

$$\left[\begin{array}{ll} \text{sem:} & \left[\begin{array}{ll} \text{status:} & \left[\text{given: } +\right] \\ \text{spec:} & \left[\begin{array}{ll} \text{agr:} & \left[\begin{array}{l}\text{countable: } + \\ \text{number: sg}\end{array}\right] \\ \text{desc:} & \left[\begin{array}{l}\text{set: } \ldots \\ \text{spec: } \ldots\end{array}\right]\end{array}\right]\end{array}\right]\end{array}\right] \tag{6.20}$$

where the ellided SET value is given by the SEM value in (6.21) and the ellided SPEC value is that given in (6.22).

6.4.2 From AS to RS

The mapping from AS to RS is relatively straightforward and involves the tasks mentioned in section 6.2.2:

- rearrangement of information;
- determination of the relation between modifier and modified in a noun-noun NP such as *some sea salt*;
- determination of the relation between NP_1 and NP_2 in a partitive NP of the form NP_1 *of* NP_2 such as *the skin of the avocado*; and
- analysing a pseudo-partitive NP like *a stick of celery* as a quantity of some substance.

The ruleset hierarchy used in mapping from AS to RS is shown in figure 6.7. In the figure, a directed arrow indicates an invocation call.

The topmost ruleset in the hierarchy is RS-SENT-TOP and this is invoked by the AS→RS module. Depending on whether the sentence consists of a simple clause or multiple clauses with a connector, the RS-SENT-TOP ruleset invokes either the RScl or the RS-comp-cl ruleset respectively. The RScl ruleset is itself invoked by the RS-comp-cl ruleset for each clause in the sentence. Notice that the RS-comp-cl ruleset can also invoke the RSnp ruleset. This happens in sentences like the following:

(6.50) 30 minutes before *the end of cooking time*, take out the chicken leg.

where the RSnp ruleset is invoked by the RS-comp-cl ruleset to build up the feature structure of the the NP *the end of cooking time.*

The RS-CL ruleset builds up the RS structure of a simple clause. It does this by calling the RSnp ruleset to build up the RS structure from the AS structure of the various NPs in the clause. If there exists an adverbial clause in the sentence like *for*-adverbial and *until*-adverbial, the RSadverbial ruleset is invoked by RScl to build up the appropriate feature structure.

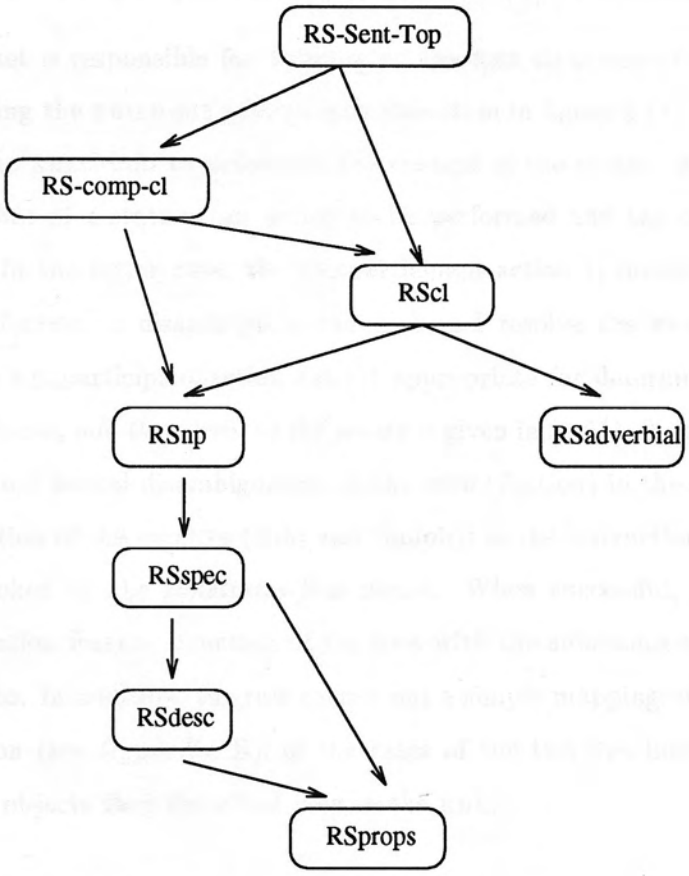


Figure 6.7: The ruleset hierarchy for AS to RS

6.4.3 From RS to KBE

The ruleset hierarchy used in mapping the RS structure into the KBE structure is given in figure 6.8. As in figure 6.7, a directed arrow in the figure indicates an invocation call. As noted previously, the KBE-SENT-TOP ruleset is the topmost ruleset invoked by the INCREMENTAL-BUILD-QTS module. The presence of temporal adjuncts causes the appropriate ruleset (for example, KBEduring, KBEbefore) to be invoked.

The KBEcl ruleset is responsible for building up the KBE structure of a simple clause (i.e. implementing the BUILD-SITUATION-REP algorithm in figure 4.11). It does this by first invoking the KBEcl-info to determine the content of the clause. A clause content can either be that of a state or an action to be performed and the objects involved in the action. In the latter case, the KBEparticipant-action is invoked to carry out constraint satisfaction to disambiguate the verb and resolve the NPs. An example of a rule in the KBEparticipant-action ruleset appropriate for determining the clause content of the clause, *add the carrot to the potato* is given in (6.51). To gloss in English, the rule carries out lexical disambiguation of the verb (?action) in the instruction and reference resolution of the two NPs (?obj and ?indobj) in the instruction via constraint satisfaction invoked by the *constraint-find* clause. When successful, the rule builds up the SPECification feature structure of the KBE with the substance which ?action is disambiguated to. In addition, the rule carries out a simple mapping via the *case-attr-mapping* function (see Appendix B), of the cases of the two NPs into the respective roles which the objects they described play in the KBE.

```
(6.51)  ?action = ⟨X action⟩
        ?obj    = ⟨X participants in obj⟩
        ?indobj = ⟨X participants in indobj⟩
        (constraint-find ?event-subst ?obj-ent ?indobj-ent
                         :where
                         (get-concepts ?action)
                         (invoke-ruleset KBEnp ?obj)
                         (invoke-ruleset KBnp ?indobj)
                         :constraints
                         (action-constraints ?event-subst ?obj-ent ?indobj-ent))
        →
```

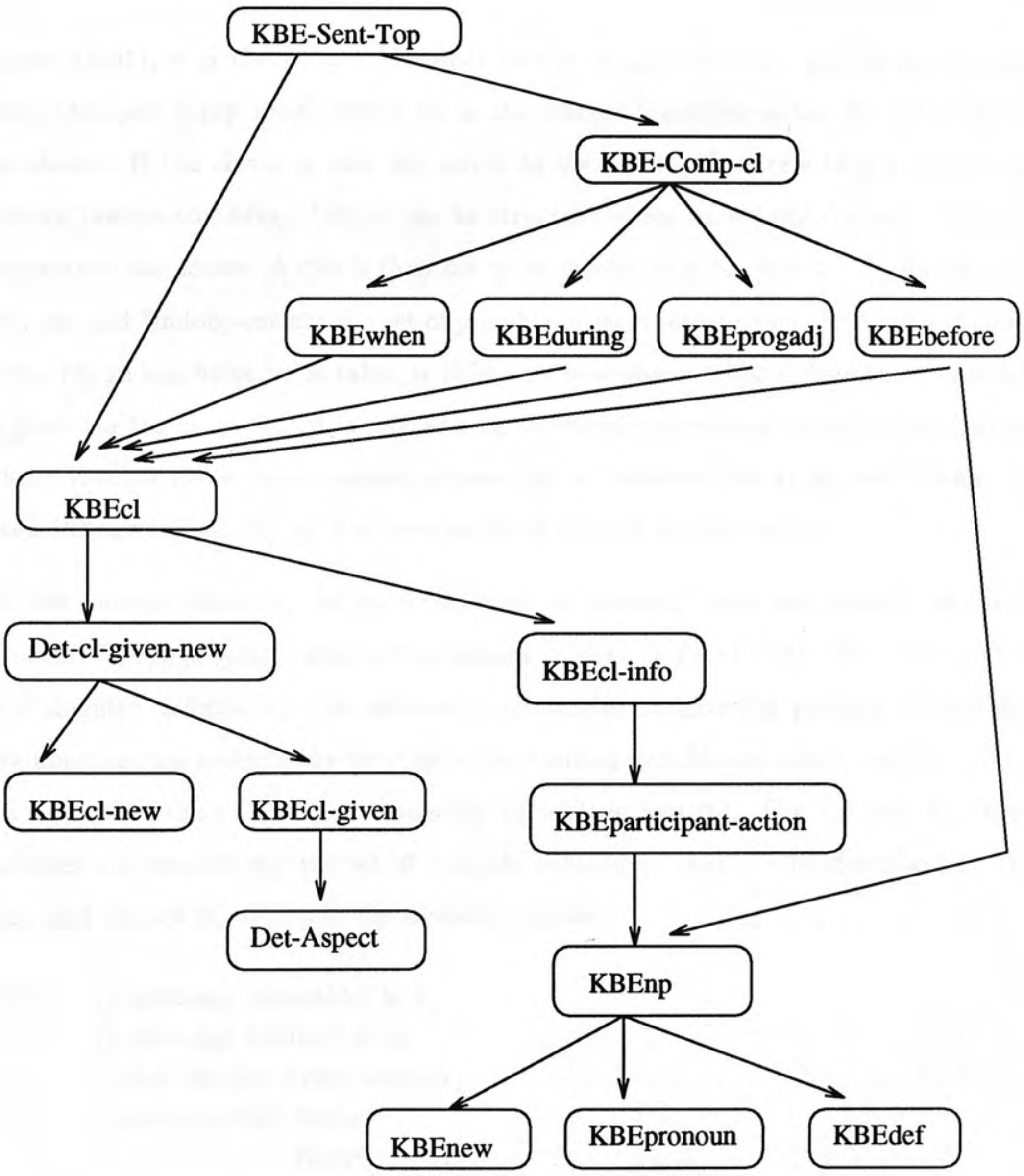


Figure 6.8: The ruleset hierarchy for RS to KBE

```
⟨KBE spec subst⟩ = ?event-subst
⟨KBE spec participants⟩ = (case-attr-mapping ?event-subst
                                              (obj ?obj-ent)
                                              (indobj ?indobj-ent))
```

In rule (6.51), X is the feature identifier of the feature structure picked out by the path, ⟨RS spec tprop prop⟩ where RS is the feature identifier of the RS structure of the clause. If the clause is *add the carrot to the potato*, then rule (6.51) applies by binding ?action to *adding*, ?obj to the RS structure of *the carrot* and ?indobj to the RS structure of *the potato*. A CSP is then set up as described in chapter 4. The domain for ?obj-ent and ?indobj-ent are the set of possible objects returned by the KBEnp ruleset. In TASTE, an indefinite NP is taken to refer to a new object while a definite NP is taken as given (in the sense that it is related to a previously mentioned object). The KBEnp ruleset invokes three other rulesets depending on whether the NP is new (KBEnew), given (KBEgiven), or the NP is a pronoun or is missing (KBEpronoun).

For our current example, we have two definite NPs and thus, the KBEdef ruleset is invoked. The appropriate rule in this ruleset is given in rule (6.52). This rule applies to a singular definite NP. It initiates a constraint satisfaction process to lexically disambiguate the noun in the NP (the corresponding variable is ?subst) and determine the referent of the NP (the corresponding variable is ?entity). The domains for these variables are respectively the set of possible substances that can be described by the noun and the set of objects in the discourse model.

```
(6.52)  ⟨X spec agr countable⟩ is +
        ⟨X spec agr number⟩ is sg
        ?cat = ⟨X spec type category⟩
        (constraint-find ?subst
                         ?entity
                         :where
                         (get-concepts ?cat)
                         Discourse Model
                         :constraints
                         ⟨?entity spec structure⟩ is individual
                         ⟨?entity spec substance⟩ is ?subst)
        →
```

Returns ?entity

In the case of *the carrot* and assuming that there is a carrot entity in the discourse model (x_1), rule (6.52) applies by binding ?cat to carrot. The function GET-CONCEPTS returns carrot-matter. Constraint satisfaction succeeds with ?entity bound to $\{x_1\}$.

When the KBE-cl-info returns successfully, the DET-CL-GIVEN-NEW ruleset is invoked to determine if the clause content is that of a new event or linked to an existing event (see chapter 4). If the clause content is new, a new event is created, otherwise, the DET-ASPECT ruleset is invoked to determine which aspect of the given event the current clause content is mentioning. Recall from chapter 4 that we use the presence and absence of the progressive, perfect and *for*-adverbial to determine whether the aspect is one of *perfect*, *progressive* or *neutral perfective.*

A successfully built new event will be sent to a DOMAIN MODELLER module where the effects of the event will be simulated (see chapter 4).

6.5 Beyond Natural Language Processing

We now consider how the rule system can be used to implement the reasoning needs of modules that manipulate on the KBEs returned by the NLFE. These are as follows:

1. to determine the set of previous events whose relations with the current situation can be deduced;
2. to determine the relations between situations; and
3. to determine the qualitative IA relations between the time intervals of the situations described in the text.

The first two needs given above occur in the INTEGRATE module whose role is to integrate the KBE of the current situation into the preceding QTS. This module implements the algorithm given in figure 4.5.

The first thing the module does is to determine the initial reference event for the current KBE. It then invokes the DET-RELATION ruleset to determine the relation between the current KBE and the KBE of the initial reference event. This ruleset takes the two KBEs and returns the feature structure representing the relation between them. An example of a rule in this ruleset is as follows:

(6.53) (enable-p ⟨KBE1⟩ ⟨KBE2⟩) →
⟨R index⟩ = (generate-relation-index)
⟨R spec relation⟩ = enable
⟨R spec sit1⟩ = ⟨KBE1 index⟩
⟨R spec sit2⟩ = ⟨KBE2 index⟩

where the function GENERATE-RELATION-INDEX creates a new index for the relation. The above rule determines whether an enablement relation exists between KBE1 (the KBE structure of the initial reference event) and KBE2 (the KBE structure of the current sentence). The predicate *enable-p* is verified as follows. Recall that the action which an agent performs in an event is represented as a STRIPS operator with preconditions and postconditions. An event e is taken to enable an event f if e brings about a condition that is needed for the performance of the action in f. The algorithm used is as follows:

1. Using the information stored in the postcondition of the operator corresponding to the action in KBE1, determine the properties which the entity involved in KBE1 will have after modelling KBE1.

2. Similarly, using the information stored in the precondition of the operator corresponding to the action in KBE2, determine the properties which the entity involved in KBE2 needs to have before modelling KBE2.

3. Check (1) and (2) above to see if there is any match. If there is, KBE1 enables KBE2. Otherwise, KBE1 does not enable KBE2.

For instance, if KBE1 is the KBE structure of the event described by *boil the rice* (e_1) and KBE2 is the KBE structure of the event described by *cool the rice* (e_2), then e_1 enables e_2 since a postcondition of e_1 is as follows (x_1 is the object described by *the rice*):

(6.54) $\text{hot}(x_1, +)$

which is the precondition for the operator corresponding to the cooling action in e_2. The feature structure returned by rule (6.53) is thus as below:

$$(6.55) \quad \begin{bmatrix} \textsf{index: } r_1 & \\ \textsf{spec:} & \begin{bmatrix} \textsf{relation: enable} \\ \textsf{sit1: } e_1 \\ \textsf{sit2: } e_2 \end{bmatrix} \end{bmatrix}$$

The next step is to determine whether KBE2 needs to be further integrated. The algorithm used is the CHOOSE-FURTHER-REFERENCE algorithm given in figure 4.7. Recall that this algorithm works by removing from the set of remaining previous events (FURTHER) those events whose relations with KBE2 can be deduced. If FURTHER is empty, integration stops; otherwise, an event is chosen from FURTHER to integrate KBE2.

The function for removing those events whose relations with KBE2 can be deduced is the TEMPORALLY-INFERRABLE-EVENTS function. As noted in chapter 4, the working of this function can be implemented as a process of forward rule chaining. We will now illustrate how this process can be implemented.

To illustrate, consider a QTS of three events, namely e_1, e_2 and e_3. The QTS is defined by the set of relation feature structures $\{r_1, r_2\}$ where r_1 and r_2 are as follows:

$$(6.56) \quad \text{a.} \quad \begin{bmatrix} \textsf{index: } r_1 & \\ \textsf{spec:} & \begin{bmatrix} \textsf{relation: include} \\ \textsf{sit1: } e_1 \\ \textsf{sit2: } e_2 \end{bmatrix} \end{bmatrix}$$

$$\text{b.} \quad \begin{bmatrix} \textsf{index: } r_2 & \\ \textsf{spec:} & \begin{bmatrix} \textsf{relation: after} \\ \textsf{sit1: } e_3 \\ \textsf{sit2: } e_1 \end{bmatrix} \end{bmatrix}$$

Now, suppose that the current event is e_4 and the initial reference event is e_3. We further suppose that e_4 is after e_3 and that the relation between e_4 and e_3 is represented by the following feature structure:

$$(6.57)\quad \begin{bmatrix} \textsf{index: } r_3 & \\ \textsf{spec: } & \begin{bmatrix} \textsf{relation: after} \\ \textsf{sit1: } e_4 \\ \textsf{sit2: } e_3 \end{bmatrix} \end{bmatrix}$$

We now invoke the TEMPORALLY-INFERRABLE-EVENTS function to determine the previous events whose relations with e_4 can be deduced. The set FURTHER for this example is as follows:

(6.58) $\text{FURTHER} = \{e_1, e_2\}$

The TEMPORALLY-INFERRABLE-EVENTS function will invoke the DET-TEMPORALLY-INFERRABLE-EVENTS ruleset with r_3 to carry out its task. The relevant rule in this ruleset for our current example is given in (6.59). This rule collapses the three rules given in (4.55).

(6.59)
```
⟨R spec relation⟩ is either after, met-by, enabled-by
?current      = ⟨R spec sit1⟩
?intermediate = ⟨R spec sit2⟩
(constraint-find ?relation
                 :where
                 (get-qts-rel)
                 :constraints
                 ⟨?relation spec relation⟩ is either after or met-by or
                 or enabled-by or include or simultaneous or
                 generates or constituents
                 ⟨?relation spec sit1⟩ is ?intermediate)
→
(store-event ⟨?relation spec sit2⟩)
(invoke-ruleset det-temporally-inferrable-events
                (make-relation-fs after ?current ⟨?relation spec sit2⟩))
```

Note that in rule (6.59), the function GET-QTS-REL gets all the relations in the QTS built up so far and the function STORE-EVENT takes an event as argument and store it into a variable RESULT. Finally, the function MAKE-RELATION-FS takes a relation and two situations and construct a feature structure representing the relation between the two situations (we do not add this feature structure to the QTS since we are building a reduced QTS (see chapter 4)).

We now work through the steps involved in choosing further reference events for e_4:

1. Rule (6.59) is applicable to r_3 since ?current is bound to e_4 and ?intermediate to e_3. The CONSTRAINT-FIND clause is true since constraint satisfaction returns with ?relation bound to r_2 reproduced below:

   $$(6.56b) \quad \left[\begin{array}{ll} \text{index:} & r_2 \\ \text{spec:} & \left[\begin{array}{l} \text{relation: after} \\ \text{sit1: } e_3 \\ \text{sit2: } e_1 \end{array}\right] \end{array}\right]$$

   The rule therefore stores e_1 into RESULT:

   $$(6.60) \quad \text{RESULT} = \{e_1\}$$

   and invokes the DET-TEMPORALLY-INFERRABLE-EVENTS ruleset with:

   $$(6.61) \quad \left[\begin{array}{ll} \text{spec:} & \left[\begin{array}{l} \text{relation: after} \\ \text{sit1: } e_4 \\ \text{sit2: } e_1 \end{array}\right] \end{array}\right]$$

2. Rule (6.59) is again applicable to (6.61) with ?current bound to e_4 and ?intermediate bound to e_1. The CONSTRAINT-FIND clause is true since constraint satisfaction returns with ?relation bound to r_1 reproduced below:

(6.56a) $$\begin{bmatrix} \textsf{index: } r_1 & \\ \textsf{spec:} & \begin{bmatrix} \textsf{relation: include} \\ \textsf{sit1: } e_1 \\ \textsf{sit2: } e_2 \end{bmatrix} \end{bmatrix}$$

The rule therefore stores e_2 into RESULT:

(6.62) RESULT $= \{e_2, e_1\}$

and invokes the DET-TEMPORALLY-INFERRABLE-EVENTS ruleset with:

(6.63) $$\begin{bmatrix} \textsf{spec:} & \begin{bmatrix} \textsf{relation: after} \\ \textsf{sit1: } e_4 \\ \textsf{sit2: } e_2 \end{bmatrix} \end{bmatrix}$$

3. No more rules fire and the processing stops.

Thus, the TEMPORALLY-INFERRABLE-EVENTS algorithm returns $\{e_1, e_2\}$. These events are to be excluded from consideration as possible reference events for e_4. Since the set of possible further reference events is also $\{e_1, e_2\}$ (given in FURTHER), no further reference events are needed for e_4.

Finally, we consider how the reasoning mechanism can be used to determine the qualitative IA relations between the time intervals of the situations described in the text.

Recall that the QTS is implemented as a set of relation feature structures $\{r_1 \ldots r_n\}$. The traversing of the QTS required in the EVENT DECOMPOSER module and the QUALITATIVE EXTRACTOR module can then be implemented as an iteration over $\{r_1, \ldots r_n\}$. The reasoning over the r_i required in the QUALITATIVE EXTRACTOR module is carried out using the DET-QUAL-RELATION ruleset. This ruleset takes r_i as an input and returns the possible qualitative relations between the two situations in r_i. An example of a rule is that used for reasoning over the **none** relation:

(6.64) $\langle$R spec relation$\rangle$ is none
?sit1 = $\langle$R spec sit1$\rangle$

```
?sit2 = ⟨R spec sit2⟩
(goal (resource-contention ?sit1 ?sit2))
→
?sit1 (<, >, m, mi) ?sit2
```

In the rule above, the *goal* predicate is verified using the RESOURCE ruleset. This ruleset determines whether there is a resource contention between ?sit1 and ?sit2. If there is a resource contention between ?sit1 and ?sit2, then the above rule fires and the time intervals associated with ?sit1 and ?sit2 are marked as being non-overlapping.

6.6 Evaluation of TASTE

In this section, we evaluate the TASTE system in the following areas:

- Data
- Methodology
- Results

6.6.1 Data

In our work, the focus is on a processing framework for temporal analysis. We therefore chose cookery recipe texts for the variety of temporal phenomena they exhibited.

Altogether, we formed a corpus of nine cookery recipes chosen from the Internet newsgroup *rec.food.recipes* and the cookbooks *The Electric Casserole Cook Book* by the Good HouseKeeping Institute and the Harvest *Traditional British Cooking.*

These nine recipes contain a total of eighty seven sentences resulting in an average of about ten sentences per recipe. The sentences range over the six kinds of sentence type handled by TASTE (see page 142). These sentence types cover a reasonable set of temporal phenomena. For example, the sentences contain temporal connectives like *when* and *meanwhile*, temporal adverbials like *occasionally* and quantitative temporal expressions like *ten minutes before.*

Out of the nine recipes, three recipes (recipes in A.7, A.8 and A.9) were used as a 'training' set for building up the system's grammar and lexicon as well as the conceptual knowledge base and also for testing the various components of the system. The remaining six recipes were then tested on the system. The typical reason for the system's failure to handle the test data was insufficient grammar and lexicon coverage. This is easily corrected by increasing the coverage.

The corpus collected is not big (as compared to the large corpus used in the Third Message Understanding Evaluation and Conference (MUC-3) [Sundheim 91]) either in quantity or the length of each text. However, we believe that the corpus contains enough temporal phenomena to test our temporal analysis framework.

6.6.2 Methodology

As noted, TASTE takes a cookery recipe text as input and outputs the temporal structure of the text. Recall from chapter 5 that the temporal structure is a total order on the start and end time points of the time intervals associated with the events and states mentioned in the text. This raises the question of how we can go about evaluating the results achieved.

Our methodology for evaluating the system can be said to be CORPUS-BASED, i.e. a corpus of inputs is given to the system and measurements are made based on the system outputs [Neal & Walter 91]. In addition, we focus on *what* the system does, i.e. measuring performance based on well-defined input/output pairs without considering how the system processes the input and generates the output.

We did not employ any formal criteria of measuring the outputs of TASTE (such as the criteria of recall and precision used in MUC-3 for information extraction). Instead, we simply use an informal criteria, namely whether the ordering of the events and states given by the system is a plausible one based on the author's (and his fellow colleagues) understanding of the text. Here, a temporal structure is said to be plausible if it satisfies the constraints on the order of the events and states given in the text and the environment.

Another possible way of measuring the outputs of TASTE would be to extract a schedule specifying the order of the actions which an agent has to carry out. This schedule can then be simulated (for example using the simulator developed at the University of Pennsylvania for instruction understanding [Webber *et al* 91]) and the correctness of the output determined from looking at the simulation. Recall from chapter 5 (page 205) that a possible way of obtaining this schedule is to remove any states, composite events and events that can be generated by other events from the temporal structure. However, we did not investigate this measurement method.

As can be seen from the description above, our method of assessing the output of TASTE is rather limited and there is much room for further work here.

6.6.3 Results

The testing of TASTE shows that the system does produce plausible temporal structure (according to our discussion in section 6.6.2) for each of the nine recipes in the corpus. In addition, by varying the environmental information (such as number of agents), we obtain different plausible temporal structures from the same QTS of a text.

The temporal structure output by TASTE shows the time intervals assigned to the events and states mentioned in the input text. In addition, it also shows the start and end time points of the time intervals. We believe that such an output is more precise than the temporal structure produced in work like [Webber 87, Song & Cohen 91b].

However, since our measurement criteria is informal, we do not have a formal mean of verifying the test results. In addition, as we focus on what the system does and not on how the system processes the input and generates the output, we are not able to ascertain how much the data used exercise the capabilities of the system. For instance, we do not have any statistics regarding the rules that were fired or not fired by the system when it carried out the temporal analysis.

6.7 Conclusion and Summary

This chapter has presented TASTE, a system for carrying out the temporal analysis of instructional texts.

The input to TASTE is a natural language text. In term of linguistic coverage, TASTE currently handles a reasonable set of linguistic phenomena needed in temporal analysis. For example, it can handle clauses containing temporal adverbials like the *for*-adverbial and the *until*-adverbial. It also handles temporal adjuncts like *before*, *meanwhile* and *when*. In particular, linguistic expressions involving both qualitative and quantitative information can be handled.

TASTE handles both qualitative and quantitative information. Typically, previous systems carrying out temporal analysis only consider qualitative information. In addition, TASTE is able to produce different temporal structures under different assumptions of resources.

The TASTE system is highly modular as reasoning rules are grouped together into rulesets which are in turn organised into an invocation hierarchy. The rulesets look at different parts of a feature structure and arranging them into an invocation hierarchy allows us to define a simple success-failure reporting strategy reflecting the notion that in going from feature structure to another feature structure, the production of a meaning representation relies on the meaning representation of the subphrases [Gazdar & Mellish 89].

TASTE has been implemented and tested on nine cookery recipes. However, as discussed in section 6.6, this testing is still rather limited.

Chapter 7

Conclusions and Future Directions

7.1 Conclusions

This thesis has addressed the problem of temporal analysis for natural language discourse. We propose a processing framework for carrying out temporal analysis called TA2. In this framework, the input is a natural language text and not some form of internal representations of sentences in the text. Temporal analysis proceeds in two steps.

In the first step, we take situations (i.e events or states) as the basic temporal ontology and build up a reduced and fully-integrated quasi-temporal structure encoding the representation of the situations described in the text and the relations between a situation and its reference events. For cookery recipes, we identify seven kinds of relations between situations: **precede**, **simultaneous**, **include**, **meet**, **enablement**, **generation** and **constituency**. When none of these relations are explicitly mentioned or inferred from the text, we mark the relation between two situations as **none**.

In the second step, we associate each situation with a unique time interval and take time intervals and time points as the basic temporal ontology. We then work out the qualitative IA relations between time intervals from the relations that hold between the situations associated with the time intervals. Of the eight possible relations between situations mentioned above, six of these have direct temporal imports in the sense that there is a direct correspondence between the relations and a set of IA relations. However, this is not the case for the **constituency** and **none** relations. In fact, the **none** relation can give rise to different temporal imports in different environments.

Finally, the set of qualitative IA relations is combined with the set of quantitative relations extracted from the text. This combined set can be viewed as a constraint satisfaction problem and solved using techniques developed in the field of temporal reasoning.

The framework is a computational one and has been successfully implemented in a system called TASTE. In working out TA2, our contributions are as follows:

- TA2 integrates both qualitative and quantitative temporal information to work out the temporal structure of a text. Previous models of temporal analysis consider only qualitative temporal information. Thus, TA2 is more complete and precise than previous models.
- TA2 shows that information from outside a text can affect temporal analysis. For instance, in the context of cookery recipes, different assumptions about resources can give rise to different temporal structures for the same text. Previous models of temporal analysis do not consider such information and thus do not show how the same text can correspond to different temporal structures in different environments.
- TA2 uses a novel approach to the integration problem that makes explicit the notion of a reference event for the current situation. More specifically, the set of reference events for the current situation are those preceding events whose relations with the current situation have to be determined from both linguistic and real-world knowledge. Two heuristics are proposed for choosing the reference events. One heuristic is based on the 'similarity' between objects where the similarity measure is computed from information obtained during reference reso-

lution. When this heuristic produces more than one possible candidate event, we use another heuristic based on the notion of an event being temporally further than another event to pick the reference event.

- In applying TA2 to cookery recipes, we introduce the notion of a generalised physical event as a way of representing simple events, composite events with sub-events and repetitive events. This knowledge representation for events has a strong parallel with that for objects used in EPICURE [Dale 92]. As noted in [Wilensky 91], the representation of situations, on the whole, has been relatively neglected and when they are provided at all, the ontology of situations has generally been limited to simple events.

- TA2 is a computational framework and a system called TASTE has been successfully built using the framework. TASTE carries out temporal analysis of cookery recipes and has been tested on nine naturally occurring recipes. As pointed out in section 6.6, the testing is still somewhat limited since we do not have a formal measurement criteria for assessing the system's outputs and the corpus does not fully exercise the capabilities of the system.

- TA2 shows how constraint-satisfaction based temporal reasoning algorithms can be incorporated into temporal analysis. We analyse the temporal reasoning requirement of instructional texts and show that no exact algorithms exist to satisfy this requirement. The usual strategy then is to have a preprocessing stage of constraint propagation followed by backtracking search.

7.2 Future Directions

There are a number of possibilities for future work arising from the work reported in this thesis.

7.2.1 Application to Simple Narratives

As noted in chapter 1, previous work on temporal analysis has used simple narratives as their domain of study. On the other hand, the design of TA2 is motivated by the study of instructional texts and in particular cookery recipes. An immediate future work is then to show how TA2 can be extended to the temporal analysis of simple narratives. We believe this is possible since our algorithm for building up the QTS and extracting the temporal structure is domain independent (the domain dependent part is that the set of relations that holds between situations might vary in different domains).

In this section, we outline how TA2 can be applied to analyse the following simple narrative (adapted from [Webber 87]):

(7.1)
a. e_1: John went over to Mary's house.
b. e_2: They talked about her brother.
c. e_3: He had spent five weeks in Alaska with a friend.
d. e_4: Together, they climbed Mt. McKinley.
e. e_5: Mary asked John whether he would want to go to Alaska some time.

In applying TA2 to example 7.1, the first task is to compute the QTS. Glossing over details, the steps involved in this task are as follows:

1. At the end of processing the first sentence, the QTS is as follows:

   (7.2) $\text{QTS}_0 = (\{e_1\}, \{\})$

2. The second sentence describes e_2 and the initial reference event for e_2 is e_1. We now determine the relation between e_2 and e_1. From world knowledge that one can only talk after they meet, we have the talking event to be after the event of John's going over to Mary's house. The updated QTS is now as follows:

   (7.3) $\text{QTS}_1 = (\{e_1,e_2\}, \{\langle \text{after}, e_2, e_1\rangle\})$

3. We now process the third sentence which describes e_3. Following our integration algorithm, we first decide on an initial reference event for e_3. Recall from chapter 4 that this is carried out by determining the degree of similarity between e_3 and each of the preceding events and taking the preceding event which is most 'similar' to the current event. The degree of similarity is computed as follows:

   (4.45) $\text{degree-of-similarity}(curr, prev) = \sum_i \sum_j f(prev, c\text{-}obj_i, p\text{-}obj_j)$

   where $c\text{-}obj_i$ and $p\text{-}obj_j$ are the objects in the event $curr$ and $prev$ respectively. The function f itself is defined as follows:

   $$f(prev, c\text{-}obj_i, p\text{-}obj_j) = \begin{cases} 1 & \text{if } c\text{-}obj_i \text{ and } p\text{-}obj_j \text{ are semantically related} \\ 1 & \text{if } c\text{-}obj_i \text{ and } prev \text{ are semantically related} \\ 0 & \text{otherwise} \end{cases}$$

   Using the above formula, we arrive at the following degree of similarity between e_3 and e_2 and between e_3 and e_1:

   - degree-of-similarity(e_3, e_2) = 1
   - degree-of-similarity(e_3, e_1) = 0

   since e_3 and e_2 share one 'similar' entities: Mary's brother, while e_3 and e_1 share no 'similar' entities. We therefore choose e_2 as the initial reference event for e_3 and determine their relation. This makes use of the following rule encoding linguistic knowledge [Lascarides & Asher 91] to determine that e_3 **precedes** e_2:

   (7.4) If the current event e is described in the past perfect tense
   and the reference event f is described in the simple past tense
   then e precedes f

   Since e_3 occurs before e_2 and e_2 occurs after e_1, it is not possible to deduce the relation between e_3 and e_1. We therefore take e_1 as further reference for e_3 and determine their relation. Using rule (7.4), we again have e_3 **precedes** e_1. The updated QTS is now as follows:

   (7.5) $\text{QTS}_3 = (\{e_1,e_2,e_3\}, \{\langle\text{after}, e_2, e_1\rangle, \langle\text{precedes}, e_3, e_2\rangle, \langle\text{precedes}, e_3, e_1\rangle\})$

4. We next process the fourth sentence which gives rise to e_4. The degree of similarity between e_4 and each of the preceding events are as follows:

   - degree-of-similarity(e_4, e_3) = 3
   - degree-of-similarity(e_4, e_2) = 1
   - degree-of-similarity(e_4, e_1) = 0

   since e_4 and e_3 share three 'similar' entities: Mary's brother, the friends of Mary's brother and Mt. McKinley which is semantically related to Alaska (Mt. McKinley is situated within Alaska), e_4 and e_2 share one 'similar' entity: Mary's brother and e_4 and e_1 have no common entity. We therefore choose e_3 as the initial reference for e_4 and determine their relation. From world knowledge that the climbing must occur during their 5-week stay in Alaska, we can determine that e_4 is included in e_3. Since e_3 precedes both e_2 and e_1, the relations between e_4 and each of e_2 and e_1 can be deduced. No further reference events are needed for e_4 and the updated QTS is as follows:

   $$\text{(7.6)} \qquad \text{QTS}_4 = (\{e_1,e_2,e_3,e_4\}, \{\langle \text{after}, e_2, e_1\rangle, \langle \text{precedes}, e_3, e_2\rangle, \langle \text{precedes}, e_3, e_1\rangle, \langle \text{included-in}, e_4, e_3\rangle\})$$

5. Finally, we process the last sentence which describes e_5. As before, we need to determine the initial reference event for e_4. From information collected during reference resolution that was carried out when processing e_1, e_2, e_3 , e_4 and e_5, we know the following:

   - The event e_5 shares two objects with e_1, namely the person Mary and John.
   - The event e_5 shares two objects with e_2, namely the person Mary and John.
   - The event e_5 shares one object with e_3, namely the place Alaska.
   - The event e_5 shares one object with e_4 since Mt. McKinley mentioned in e_4 is semantically related to Alaska mentioned in e_5.

   In addition to the above, we also note that the person Mary and John in e_5 are described respectively using the proper noun *Mary* and *John*. Similarly, the person Mary and John in e_1 are mentioned respectively using the proper noun *Mary* and *John*. However, the person Mary and John in e_2 are mentioned using the pronoun *they* which is a different form of referring expression from *Mary* and *John*.

Here, we make the hypothesis that the same form of the referring expressions used for the person Mary and John in e_5 and e_1 indicates another kind of links between objects. Such a link is called a lexical link by Halliday and Hasan [Halliday & Hasan 76]. One way of making use of this link is to expand the function f needed in formula (4.45) to the following:

$$f(prev, c\text{-}obj_i, p\text{-}obj_j) = \begin{cases} 1 & \text{if } c\text{-}obj_i \text{ and } p\text{-}obj_j \text{ are semantically related} \\ 1 & \text{if } c\text{-}obj_i \text{ and } prev \text{ are semantically related} \\ 1 & \text{if } c\text{-}obj_i \text{ and } p\text{-}obj_j \text{ are described by the same} \\ & \text{proper name} \\ 0 & \text{otherwise} \end{cases}$$

The degree of similarity between e_5 and each of the preceding events e_1, e_2, e_3 and e_4 are then as follows:

- degree-of-similarity(e_5, e_1) = 4
- degree-of-similarity(e_5, e_2) = 2
- degree-of-similarity(e_5, e_3) = 1
- degree-of-similarity(e_5, e_4) = 1

We thus choose e_1 as the initial reference event for e_5 and determine the relation between e_5 and e_1. From world knowledge, we can determine that the asking in e_5 must come after the going over event in e_1. Since e_5 is after e_1 and e_2 is after e_1, our CHOOSE-FURTHER-REFERENCE algorithm given in figure 4.7 will take e_2 as a further reference for e_5. We therefore compute the relation between e_5 and e_2 and from world knowledge determine that e_5 comes after e_2. From this relation between e_5 and e_2, our CHOOSE-FURTHER-REFERENCE algorithm can determine that the relation between e_5 and e_3 and that between e_5 and e_4 can be deduced. It is therefore not necessary to take e_3 and e_4 as reference events for e_5. The QTS is then as follows:

(7.7) $\text{QTS}_5 = (\{e_1,e_2,e_3,e_4,e_5\}, \{\langle\text{after}, e_2, e_1\rangle, \langle\text{precedes}, e_3, e_2\rangle,$
$\langle\text{precedes}, e_3, e_1\rangle, \langle\text{included-in}, e_4, e_3\rangle$
$\langle\text{after}, e_5, e_1\rangle, \langle\text{after}, e_5, e_2\rangle\})$

Notice that all the relations in the QTS are those with direct correspondence with a set of IA relations:

- after(e_2, e_1) $\equiv te_2 \; \{>, mi\} \; te_1$
- precedes(e_3, e_2) $\equiv te_3 \; \{<, m\} \; te_2$
- precedes(e_3, e_1) $\equiv te_3 \; \{<, m\} \; te_1$
- included-in(e_4, e_3) $\equiv te_4 \; \{s, f, d\} \; te_3$
- after(e_5, e_1) $\equiv te_5 \; \{>, mi\} \; te_1$
- after(e_5, e_2) $\equiv te_5 \; \{>, mi\} \; te_2$

This set of IA relations can then be combined with any quantitative information via constraint propagation and the possible temporal structures extracted.

The QTS and one of the temporal structures for example (7.1) are given in figure 7.1.[1]

Our approach as described above differs from the temporal analysis approach proposed in [Webber 87, Song 91a, Hwang & Schubert 91]. That approach uses tense change information to determine the reference events of a current situation. This produces the wrong result for example 7.1 that e_4 takes reference from e_2 and that e_5 takes reference from e_4 (see chapter 2). On the other hand, in our approach, we choose reference events based on the link that can exist between objects in events. Using this approach, we determine correctly that e_4 takes reference from e_3 and that e_5 takes reference from e_1.

[1]The symbols 'a', 'p' and 'd' are abbreviations respectively for the after, precedes and included-in relation.

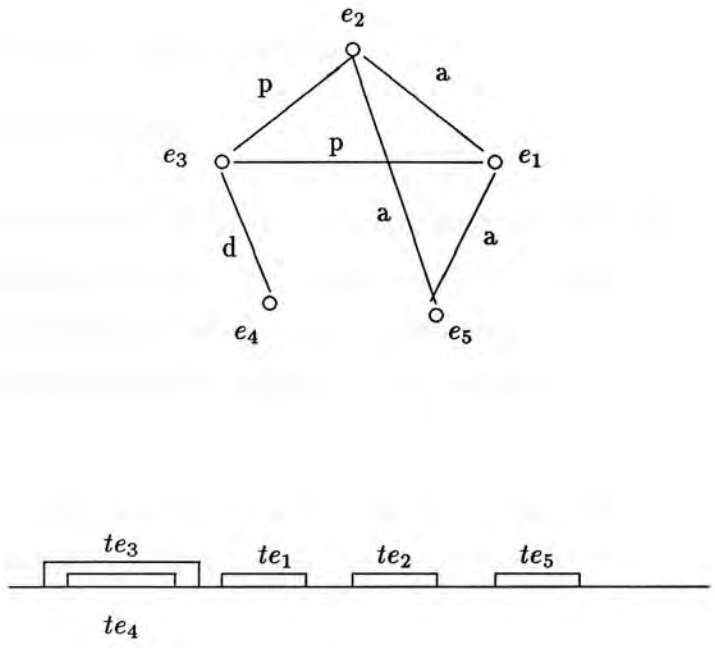


Figure 7.1: The QTS and temporal structure of example (7.1)

7.2.2 Temporal Structure and Discourse Structure

Another interesting area for future work is to investigate the interaction between the determination of discourse structure and that of temporal structure. To ground our discussion, we assume the notion of discourse structure given by Grosz and Sidner [Grosz & Sidner 86]. In Grosz and Sidner's theory, a discourse can be segmented into many distinct discourse segments (DS) which in turn may be segmented into a number of discourse segments and so on. A sequence of sentences constitutes a DS if they together serve to realise a particular DISCOURSE PURPOSE.

Using the notion of discourse structure of Grosz and Sidner, an important task in determining the discourse structure is that of identifying the appropriate discourse segments. As has been noted in the literature [Grosz & Sidner 86, Allen 87], discourse segmentation makes use of the following kinds of information:

- Information from cue phrases

- Information from tense and aspect
- Information from reference resolution
- Real-World Knowledge

However, as shown in this thesis, the last three kinds of information given above are used in temporal analysis also. This seems to imply at least some kinds of synergy between temporal analysis and discourse segmentation. The nature of this synergy and how the information flows between the two processes is an area left for further investigation.

In conclusion, we hope that the work reported in this thesis can provide a foundation which further work in temporal analysis can be based on.

Bibliography

[Aho *et al* 76] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley Publishing Co., Reading, MA, 1976.

[Allen & Kautz 85] J. Allen and H. A. Kautz. A Model of Naive Temporal Reasoning. In J.R. Hobbs and R.C. Moore, editors, *Formal Theories of the Commonsense World*, pages 251–266. Ablex Publishing Corporation, Norwood, NJ, 1985.

[Allen 83] J. Allen. Maintaining Knowledge About Temporal Intervals. *Communications of the ACM*, 11(26):832–843, 1983.

[Allen 87] J. Allen. *Natural Language Understanding.* Benjamin/Cummings Publishing Co, Inc, 1987.

[Asher & Morreau 91] N. Asher and M. Morreau. Common Sense Entailment: A Modal Theory of Non-Monotonic Reasoning. Technical Report DYANA deliverable 2.5b, Centre for Cognitive Science, 1991.

[Balkanski 90] C. Balkanski. Modelling Act-type Relations in Collaborative Activity. Technical Report Technical Report TR-23-90, Center for Research in Computing Technology, Harvard University, 1990.

[Caenepeel & Moens 93] M. Caenepeel and M. Moens. Temporal Structure And Discourse Structure. In C Vetters and C Vet, editors, *Tense and Aspect in Discourse.* Mouton de Gruyter, 1993.

[Comrie 76] B. Comrie. *Aspect.* Cambridge University Press, 1976.

[Dale 92] R. Dale. *Generating Referring Expressions: Constructing Descriptions in a Domain of Objects and Processes.* MIT Press, 1992.

[Davidson 69] D. Davidson. The Individuation of Events. In N.Rescher et al, editor, *Essays in Honor of Carl G. Hempel*, pages 216–234. Reidel, Dordrecht,Holland, 1969.

[Davidson 80] D. Davidson. *Essays on Actions and Events.* Clarendon Press, Oxford, 1980.

[Di Eugenio 92] B. Di Eugenio. Understanding Natural Language Instructions: The Case of Purpose Clauses. In *Proceedings of 30th Annual Meeting of the Association for Computational Linguistics*, pages 120–127. University of Delaware, Newark, Delaware, 1992.

[Dowty 86] D. Dowty. The Effects of Aspectual Class On The Temporal Structure of Discourse: Semantics or Pragmatics. *Linguistics and Philosophy*, 9(1):37–62, 1986.

[Eberle 92] K. Eberle. On Representing The Temporal Structure of A Natural Language Text. In *Proceedings of COLING-92*, pages 288–294, 1992.

[Fikes & Nilsson 71] R. E. Fikes and N. J. Nilsson. STRIPS: A New approach to The Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2:189–208, 1971.

[Fillmore 68] C. Fillmore. The Case for Case. In E. Bach and R. T. Harms, editors, *Universals in Linguistics Theory.* Holt, Rinehart and Winston, New York, 1968.

[Gazdar & Mellish 89] G. Gazdar and C. Mellish. *Natural Language Processing in Lisp.* Addison-Wesley Publishing Company, 1989.

[Gent 87] I Gent. A Unification Grammar Development Environment. Unpublished M.Sc. thesis, Deparment of Artificial Intelligence, University of Edinburgh, 1987.

[Goldman 70] A. I. Goldman. *A Theory of Human Action.* Princeton University Press, Princeton, NJ, 1970.

[Grosz & Sidner 86] B. Grosz and C. Sidner. Attention, Intentions And the Structure of Discourse. *Computational Linguistics*, 12:175–204, 1986.

[Grosz & Sidner 90] B. Grosz and C. Sidner. Plans for Discourse. In J.M.P. Cohen and M. Pollack, editors, *Intentions in Communication.* MIT Press, 1990.

[Grosz *et al* 83] B. J. Grosz, A. K. Joshi, and S. Weinstein. Providing a Unified Account of Definite Noun Phrases in Discourse. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, pages 44–49. Massachusetts Institute of Technology, Cambridge, Mass, 1983.

[Haddock 88] N. J. Haddock. *Incremental Semantics and Interactive Syntactic Processing.* Unpublished PhD thesis, Department of Artificial Intelligence, University of Edinburgh, 1988.

[Halliday & Hasan 76] M. Halliday and R. Hasan. *Cohesion in English.* Longman, London, 1976.

[Hammond 86] K. J. Hammond. CHEF: A Model of Case-Based Planning. In *Proceedings of the 5th National Conference on Artificial Intelligence*, pages 267–271. Philadelphia, 1986.

[Hinrichs 86] E. Hinrichs. Temporal Anaphora in Discourses of English. *Linguistics and Philosophy*, 9(1):63–82, 1986.

[Hinrichs 87] E. W. Hinrichs. A Compositional Semantics of Temporal Expressions in English. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, pages 8–15. Standford University, Standford, Ca, 1987.

[Hinrichs 88] E. Hinrichs. Tense, Quantifiers and Contexts. *Computational Linguistics*, 14(2):3–14, June 1988.

[Hirschman & Story 81] L. Hirschman and G. Story. Representing Implicit and Explicit Time Relations in Narrative. In *Proceedings of the 7th International Joint Conference of Artificial Intelligence*, pages 289–295. University of British Columbia, Columbia, Vancouver, 1981.

[Hirst 89] G. Hirst. Ontological Assumptions in Knowledge Representation. In J. R. Brachman, H. J. Levesque, and Reiter R, editors, *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, pages 157–169. Toronto, Canada, 1989.

[Hobbs 85] J. R. Hobbs. Ontological Promiscuity. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, pages 61–69. University of Chicago,Chicago,Illinois, 1985.

[Hwang & Schubert 91] C. H. Hwang and L. K. Schubert. Tense Trees as the Fine Structure of Discourse. In *1991 Fall Symposium Series on Discourse Structure in Natural Language Understanding and Generation Working Notes*, pages 33–38, 1991.

[Karlin 88] R. F. Karlin. Defining the Semantics of Verbal Modifiers in the Domain of Cooking Tasks. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, pages 61–67. State University of New York at Buffalo, Buffalo, New York, 1988.

[Kautz & Ladkin 91] H. A. Kautz and P. B. Ladkin. Integrating Metric and Qualitative Temporal Reasoning. In *Proceedings of the 9th National Conference of Artificial Intelligence*, pages 241–246. MIT Press, 1991.

[Lascarides & Asher 91] A. Lascarides and N. Asher. Discourse Relations and Common Sense Entailment. Technical Report HCRC/RP-16, Human Communication Research Centre, University of Edinburgh, January 1991.

[Mackworth 87] A. K. Mackworth. Constraint Satisfaction. *The Encyclopedia of Artificial Intelligence*, 1:205–211, 1987.

[Mayo 61] B. Mayo. Objects, Events and Complementarity. *Philosophical Review*, LXX:340–361, 1961.

[McCawley 71] J. McCawley. Tense and Time References in English. In C. Fillmore and D.T. Langendoen, editors, *Studies in Linguistic Semantics*, pages 97–114. Hoet, Rinehart and Winston, Inc, New York, 1971.

[Meiri 91] I. Meiri. Combining Qualitative and Quantitative Constraints in Temporal Reasoning. In *Proceedings of the 9th National Conference of Artificial Intelligence.* MIT Press, 1991.

[Mellish 85] C. S. Mellish. *Computer Interpretation of Natural Language Descriptions.* Ellis Horwood Limited, Chichester, 1985.

[Moens & Steedman 88] M. Moens and M. Steedman. Temporal Ontology and Temporal Reference. *Computational Linguistics*, 14(2):15–28, 1988.

[Moens 87] Moens. *Tense, Aspect and Temporal Reference.* Unpublished PhD thesis, Centre for Cognitive Science, University of Edinburgh, 1987.

[Mourelatos 78] A.P.D Mourelatos. Events, Processes and States. *Linguistics and Philosophy*, 2:415–434, 1978.

[Neal & Walter 91] J. G. Neal and S. M. Walter. Natural Language Processing Systems Evaluation Workshop. Technical Report RL-TR-91-362, Rome Laboratory, December 1991.

[Partee 73] B. Partee. Some Structural Analogies Between Tenses and Pronouns in English. *Journal of Philosophy*, 70:601–609, 1973.

[Partee 84] B. Partee. Nominal and Temporal Anaphora. *Linguistics and Philosophy*, 7(3):243–286, 1984.

[Passonneau 88] R. Passonneau. A Computational Model of the Semantics of Tense and Aspect. *Computational Linguistics*, 14(2):44–60, 1988.

[Pollack 90] M. Pollack. Plans as Complex Mental Attitudes. In J.M.P. Cohen and M. Pollack, editors, *Intentions in Communication*. MIT Press, 1990.

[Reichenbach 47] H. Reichenbach. *Elements of Symbolic Logic*. London: Macmillan, 1947.

[Richards *et al* 89] B. Richards, I. Bethke, J.v.d. Does, and J Oberlander. *Temporal Representation and Inference*. Academic Press Limited, London, 1989.

[Ritchie 79] G. Ritchie. Temporal Clauses in English. *Theoretical Linguistics*, 6:87–115, 1979.

[Rock 92] S. Rock. Understanding Repetition in Instructions: The Semantics of Extent. In *Proceedings of the 30th Annual Meeting of the ACL, Student Session*, pages 279–281. University of Delaware, Newark, Delaware, 1992.

[Schubert & Hwang 90] L. K. Schubert and C. H. Hwang. An Episodic Knowledge Representation for Narrative Texts. Technical Report Technical Report 345, Department of Computer Science, University of Rochester, May 1990.

[Sidner 79] C. L. Sidner. Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse. Technical Report 537, MIT Artificial Intelligence Laboratory, June 1979.

[Singh & Singh 92] M. Singh and M.P. Singh. Computing the Temporal Structure of Events in Natural Language. In *Proceedings of the 10th European Conference of Artificial Intelligence*, pages 528–532, 1992.

[Song & Cohen 88] F. Song and R. Cohen. The Interpretation of Temporal Relations in Narrative. In *Proceedings of the 7th National Conference on Artificial Intelligence*, pages 745–750. St Paul, Minnesota, 1988.

[Song & Cohen 91a] F. Song and R. Cohen. Temporal Reasoning during Plan Recognition. In *Proceedings of the 9th National Conference of Artificial Intelligence*, pages 247–252. MIT Press, 1991.

[Song & Cohen 91b] F. Song and R. Cohen. Tense Interpretation in the Context of Narrative. In *Proceedings of the 9th National Conference of Artificial Intelligence*, pages 131–136. MIT Press, 1991.

[Song 91a] F. Song. *A Processing Model for Temporal Analysis and its Application to Plan Recognition.* Unpublished PhD thesis, Department of Computer Science, University of Waterloo, 1991.

[Song 91b] F. Song. Temporal Analysis and Discourse Processing. In *1991 Fall Symposium Series on Discourse Structure in Natural Language Understanding and Generation Working Notes*, pages 106–111, 1991.

[Steedman 82] M. Steedman. Reference to Past Time. In R. Jarvella and W. Klein, editors, *Speech, Place, and Action*, pages 125–127. Wiley, New York, NY, 1982.

[Sundheim 91] B. M. Sundheim. Third Message Understanding Evaluation and Conference (MUC-3): Methodology and Test Results. In J. G. Neal and S. M. Walter, editors, *Natural Language Processing Systems Evaluation Workshop*, pages 1–12. Rome Laboratory Technical Report, RL-TR-91-362, December 1991.

[Tsang 86] E.P.K Tsang. Plan generation in a temporal frame. In *Proceedings of the Seventh European Conference on Artifiical Intelligence*, pages 479–493. Brighton, UK, 1986.

[Vendler 67] Z. Vendler. *Linguistics in Philosophy.* Cornell University Press, Ithaca, New York, 1967.

[Vilain *et al* 89] M. Vilain, H. Kautz, and P Van Beek. Constraint Propagation Algorithms for Temporal Reasoning: A Revised Report. In J. deKleer and D. Weld, editors, *Readings in Qualitative Reasoning about Physical Systems*, pages 373–381. Morgan Kaufmann Publishers, 1989.

[Webber & Baldwin 92] B. Webber and B. Baldwin. Accommodating Context Change. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 96–103. University of Delaware, Newark, Delaware, 1992.

[Webber & Di Eugenio 90] B. L. Webber and B. Di Eugenio. Free Adjuncts in Natural Language Instructions. In *Proceedings of 14th International Conference of Computational Linguistics*, pages 395–400, 1990.

[Webber 78] B. Webber. A Formal Approach to Discourse Anaphora. Technical Report Technical Report 3761, Bolt Beranek and Newman, Cambridge MA, 1978.

[Webber 87] B. L. Webber. The Interpretation of Tense in Discourse. In *Proceedings of 25th Annual Meeting of the Association for*

Computational Linguistics, pages 147–154. Stanford University, Stanford, Ca, 1987.

[Webber *et al* 91] B. Webber, N. Badler, B. Di Eugenio, L. Levison, and M. White. Instructing animated agents. In *Proceedings of US-Japan Workshop on Integrated Systems in Multi-Media Environments*, 1991.

[Wilensky 91] R. Wilensky. The Ontology and Representation of Situations. In J. Allen, R. Fikes, and E. Sanderwall, editors, *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, pages 558–569. Morgan Kaufmann Publishers, 1991.

Appendix A

Recipes Processed By TASTE

In this appendix, we present the remaining 8 recipes processed by TASTE together with the results of the processing. The recipes are first preprocessed before passing to TASTE. The various simplifications made to the recipes are given in section A.1.

Recall from chapter 6 that the output of TASTE is to be interpreted as follows:

- The top diagram indicates the quasi-temporal structure which is a network whose nodes are the indices of the events (such as E1) and states (such as ST1) mentioned in the text and the arcs between the nodes are the relations between the events and states. The sentences describing the events and states are given to the left of the diagram. Finally, as the graphic routine we used do not permit directed arrow, we adopt the convention that the direction of the arrow on the arc between nodes is from the node associated with the larger index to the node associated with the smaller index. For instance, if the arc between E1 and E2 is labelled **a**, this is taken to mean that E2 is **after** E1.
- The bottom diagram indicates the temporal structure where the horizontal lines are time lines with time increasing from left to right and the black boxes represent time intervals occupied by events and states. Note that if the index of an event is E1, then the time intervals assigned to E1 is given by TE1.

A.1 Simplifying the Recipes

We catalogue here the simplifications that are made to the recipes before passing to TASTE. The simplifications made to the instructions part of the recipes are:

1. We remove any constraints that are specified on actions. For example, the instruction *reheat the soup, but don't let it boil* is simplified to *reheat the soup.*
2. We remove any mention of the manner and speed in which an action is to be carried out. For instance, the instruction *stir well* is simplified to *stir.*

3. We remove disjunctions where they appear since we are unable to represent them. For instance, the instruction, *Cook on high for 2-3 hours or low for 4-6 hours* is simplified to *Cook for 2 to 3 hours* (note that *on high* is also removed since it is indicating the manner in which the cooking is to be carried out).

4. We omit any cue words like *oh.* Thus, the sentence, *Oh, before you mash the potatoes, take the chicken leg out, cut it into little pieces and add it back in* simplifies to *Before you mash the potatoes, take the chicken leg out, cut it and add it back in.*

In addition, we make the following simplifications to the ingredient list part of a recipe:

- We round any fractional quantity of an ingredient to 1. For instance, $\frac{3}{4}$ teaspoon of salt becomes *1 teaspoon of salt.*

The original recipes and their simplified forms are given below. The QTS and the temporal structure resulting from processing the recipes are also shown.

A.2 Broccoli Cheese Soup

Source: From *rec.food.recipe*

$\frac{1}{2}$ pound fresh broccoli
1 pint half-and-half (that's half milk, half cream for the non-US people, or substitute to your liking)
2 cups water
1 pound pasteurized cheese spread (like Velveeta)
$\frac{3}{4}$ teaspoon salt
$\frac{1}{2}$ teaspoon pepper
$\frac{1}{2}$ cup cornstarch

Steam broccoli until tender (can cook in microwave on high power, 6 minutes per pound). Place half-and-half and water in the top of a double boiler. Add cheese (cut in pieces), salt and pepper. Heat until the cheese is melted. Add broccoli. Mix cornstarch and water in a small bowl. Stir into the cheese mixture in the double boiler, and heat over simmering water until the soup is thick.

Interesting Features

- The order in which the events are presented may not tally with the order in which the events are to be carried out. For example, the event described by *Mix cornstarch and water* is better done before the event described by *Add broccoli.*

- There is a state expression *the cheese is melted.*

Other Kinds of Simplifications

In addition to the 'general' simplifications mentioned in section A.1, we also made the following simplification:

- We replace the NP *the cheese mixture* with the NP *the soup.* This is because currently TASTE uses hardwired knowledge to decide on the result of actions that produce new objects from existing objects. For instance, in this current recipe, after modelling the action of the mixing action mentioned in *mix the cornstarch and some water*, TASTE determines that the result is a soup entity.

Simplified Target

$\frac{1}{2}$ pound of fresh broccoli
$\frac{1}{2}$ pint of milk
$\frac{1}{2}$ pint of cream
2 cups of water
1 pound of pasteurized cheese spread
1 teaspoon of salt
$\frac{1}{2}$ teaspoon of pepper
$\frac{1}{2}$ cup of cornstarch

Steam the broccoli until it is tender.
Place the milk, cream and water in the top of a double boiler.
Add the cheese, salt and pepper.
Heat until the cheese is melted.
Add the broccoli.
Mix the cornstarch and some water.
Stir into the soup.
Heat until the soup is thick.

The result of processing the simplified recipe is given in figure A.1. The QTS is given at the top of the figure. The system extracts the qualitative Allen relations between the time intervals and any quantitative relations. The system transcript during this stage is given below (In such transcripts, durational information is given as (⟨min⟩, ⟨max⟩) and the default time unit is minutes. Notations like TR(E8 E7) mean the Allen relations between the time interval assigned to event E8 and E7.)

```
Starts dribbling to ../recipe/r1-query (1993/6/18, 14:0:54).

> (start-interval-reasoner *relation-space*)

Asserting : TR(E8 E7) = (A MI)
Asserting : TR(E8 ST3) = M
Asserting : TR(E7 E5) = (A MI)
Asserting : TR(E7 E6) = (A MI)
```

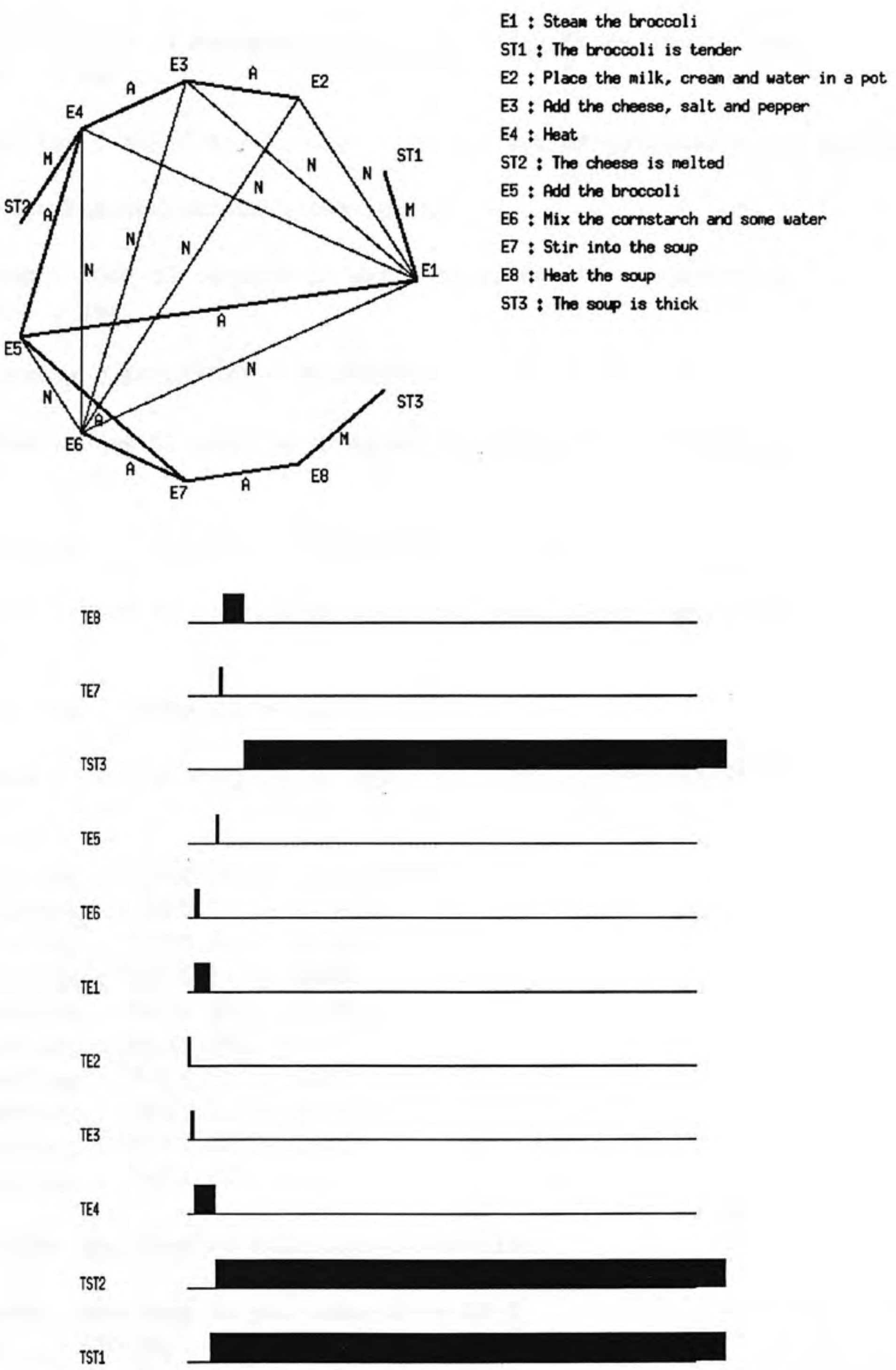


Figure A.1: The QTS and temporal structure of the BROCCOLI CHEESE SOUP recipe

```
System : Does E6 require an agent to bring it to completion ?
User   : yes

System : Does E1 require an agent to bring it to completion ?
User   : no

Asserting : TR(E6 E1) = (ANY)   /* The set of thirteen Allen relations */

/* Resource Contention Reasoning */

System : Does E2 require an agent to bring it to completion ?
User   : yes

Asserting : TR(E6 E2) = (DISJOINT)

System : Does E3 require an agent to bring it to completion ?
User   : yes

Asserting : TR(E6 E3) = (DISJOINT)

System : Does E4 require an agent to bring it to completion ?
User   : no

Asserting : TR(E6 E4) = (ANY)

System : Does E5 require an agent to bring it to completion ?
User   : yes

Asserting : TR(E6 E5) = (DISJOINT)
Asserting : TR(E5 E1) = (A MI)
Asserting : TR(E5 E4) = (A MI)
Asserting : TR(E4 E1) = (ANY)
Asserting : TR(E4 E3) = (A MI)
Asserting : TR(E4 ST2) = M
Asserting : TR(E3 E1) = (ANY)
Asserting : TR(E3 E2) = (A MI)
Asserting : TR(E2 E1) = (ANY)
Asserting : TR(E1 ST1) = M

Finished Qualitative Relations Extraction.

System : How long do you take to do E8 ?
User   : (20 20)
System : How long do you take to do E7 ?
User   : (3 3)
System : How long do you take to do E6 ?
```

```
User   : (3 3)
System : How long do you take to do E5 ?
User   : (3 3)
System : How long does E4 takes ?
User   : (20 20)
System : How long do you take to do E3 ?
User   : (3 3)
System : How long do you take to do E2 ?
User   : (3 3)
System : How long does E1 takes ?
User   : (20 20)

Starting the temporal reasoner

Asserting : TR(E8 E7) = (MI)
Asserting : TR(E4 E1) = (=)
Asserting : TR(E4 E6) = (SI)
Asserting : TR(E4 E3) = (MI)
Asserting : TR(E7 E5) = (MI)
Asserting : TR(E5 E1) = (MI)
Asserting : TR(E3 E2) = (MI)
```

Notice from the temporal structure that TASTE determines that the broccoli should be steamed during the heating of the mixture comprising milk, cream, water, cheese and salt and pepper (the steaming is during the heating since we specify to TASTE that the duration of the steaming is less than the duration of the heating). This is in spite of the fact that the instruction to *steam the broccoli* is placed at the front.

A.3 Spiced Minced Lamb with Pasta Shells

Source: From *The Electric Casserole Cook Book*

1 large onion, skinned and sliced
1 clove of garlic, skinned and crushed
1 lb minced mutton
1 tablespoon ground coriander
1 tablespoon ground ginger
$\frac{1}{2}$ tablespoon ground cummin
1 tablespoon plain flour
2 5 ounce cartons natural yogurt
$6\frac{1}{2}$ ounces tomatoes
1 small green pepper, seeded and chopped
salt and freshly ground pepper
3 ounces pasta shells

Fry the onion and garlic for 5 minutes. Add the minced lamb and cook for 5 minutes.

Stir in the spices and flour and cook for 1 minute. Stir in the yogurt, tomatoes, and seasoning. Pour into the electric casserole and place the lid in position. Cook on high for 2-3 hours or low for 4-6 hours. 45 minutes before the end of cooking time, turn the electric casserole to high, stir in the pasta shell and continue cooking.

Interesting Features

- This recipe contains a lot of quantitative information such as *for 5 minutes* and *45 minutes before the end of cooking time.*

Other Kinds of Simplifications

In addition to the 'general' simplifications, we also made the following simplifications:

- We remove the instruction *place the lid in position* and *turn the electric casserole to high.* Such removal does not affect the temporal analysis task.
- We remove the instruction *continue cooking* to simplify our processing.

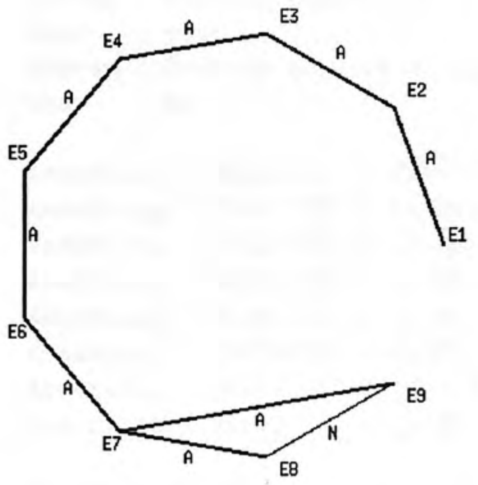
Simplified Target

1 large onion
1 clove of garlic
1 lb of minced mutton
1 tablespoon of ground coriander
1 tablespoon of ground giner
$\frac{1}{2}$ tablespoons of ground cummin
1 tablespoon of plain flour
2 cartons of natural yogurt
$6\frac{1}{2}$ ounces of tomatoes
1 small green pepper
some salt and freshly ground pepper
3 ounces of pasta shells

Fry the onion and garlic for 5 minutes.
Add the minced lamb.
Cook for 5 minutes.
Stir in the spices and flour. Cook for 1 minute.
Stir in the yogurt, tomatoes and seasonings.
Pour into the electric casserole.
Cook for 2 to 3 hours.
45 minutes before the end of cooking time, stir in the pasta shell.

The result of processing the simplified recipe is given in figure A.2. The system's transcript when extracting the qualitative Allen relations and the quantitative relations is as follows:

```
Starts dribbling to r5-ques (1993/6/11, 11:33:28).
```



E1 : Fry the onion and garlic for 5 minutes
E2 : Add the mutton
E3 : Cook for 5 minutes
E4 : Stir in the spices and flour
E5 : Cook for 1 minute
E6 : Stir in the yogurt, the tomatoes and the seasonings
E7 : Pour into the electric casserole
E8 : Cook for 1 hour
E9 : Stir in the pasta shells

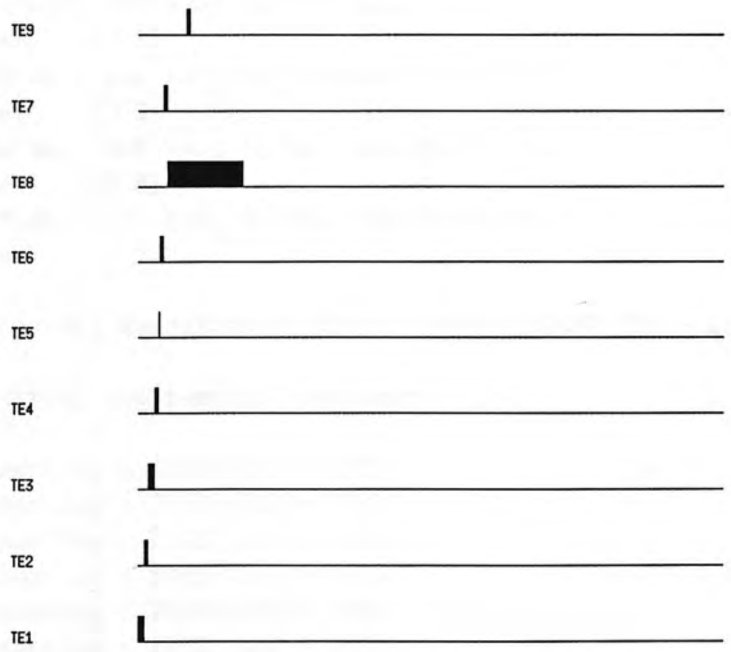


Figure A.2: The QTS and temporal structure of the SPICED MINCED LAMB WITH PASTA SHELLS recipe

```
> (start-interval-reasoner *relation-space*)

Asserting : TR(E9 E7) = (A MI)

System : Does E9 require an agent to bring it to completion ?
User   : yes
System : Does E8 require an agent to bring it to completion ?
User   : no

Asserting : TR(E9 E8) = (ANY)
Asserting : TR(E8 E7) = (A MI)
Asserting : TR(E7 E6) = (A MI)
Asserting : TR(E6 E5) = (A MI)
Asserting : TR(E5 E4) = (A MI)
Asserting : TR(E4 E3) = (A MI)
Asserting : TR(E3 E2) = (A MI)
Asserting : TR(E2 E1) = (A MI)

Finished Qualitative Relations Extraction

System : How long do you take to do E9 ?
User   :(3 3)
System : How long do you take to do E7 ?
User   :(3 3)
System : How long do you take to do E6 ?
User   :(3 3)
System : How long do you take to do E4 ?
User   :(3 3)
System : How long do you take to do E2 ?
User   :(3 3)

Asserting Quantitative Info : (45 <= RIGHT TE8 - LEFT TE9 <= 45)

Starting the temporal reasoner

Asserting : TR(E8 E7) = (MI)
Asserting : TR(E3 E2) = (MI)
Asserting : TR(E7 E6) = (MI)
Asserting : TR(E6 E5) = (MI)
Asserting : TR(E4 E3) = (MI)
Asserting : TR(E2 E1) = (MI)
Asserting : TR(E5 E4) = (MI)
```

A.4 Lemon Dressing

Source: From *rec.food.recipe*

1 lemon
1 clove garlic
$\frac{1}{2}$ teaspoon French or German mustard
3 tablespoons oil
salt and freshly ground black pepper

Remove some strips of lemon skin with a potato peeler or sharp knife. Cut into fine strips. Crush the garlic and put into a bowl, stir in the mustard, then add 1 tablespoon lemon juice and the oil. Add the salt and pepper, mix well, and add the lemon skin.

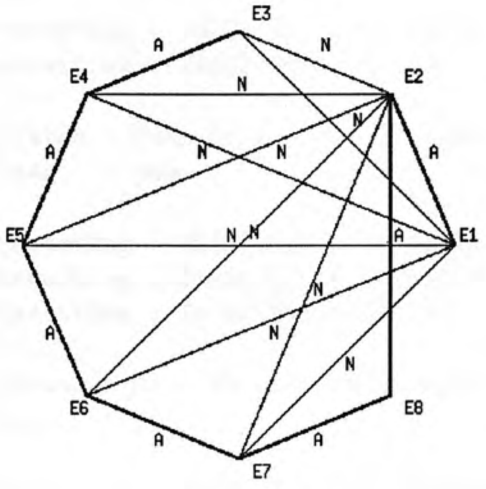
Simplified Target

1 lemon
1 clove of garlic
$\frac{1}{2}$ teaspoon of French mustard
3 tablespoons of oil
some salt and freshly ground black pepper

Remove some strips of lemon skin.
Cut.
Crush the garlic and put into a bowl.
Stir in the mustard.
Then, add one tablespoon of lemon juice and the oil.
Add the salt and pepper.
Add the lemon skin.

The result of processing the simplified recipe is given in figure A.3. The system's transcript when extracting the qualitative Allen relations and the quantitative relations is as follows:

```
Starts dribbling to ../recipe/r6-ques (1993/6/25, 9:26:23).
NIL

> (start-interval-reasoner *relation-space*)

Asserting : TR(E8 E2) = (A MI)
Asserting : TR(E8 E7) = (A MI)

System : Does E7 require an agent to bring it to completion ?
User   : yes

System : Does E1 require an agent to bring it to completion ?
User   : yes
```



E1 : Remove some strips of lemon skin
E2 : Cut
E3 : Crush the garlic
E4 : Place in a bowl
E5 : Stir in the mustard
E6 : Add one tablespoon of lemon juice and the oil
E7 : Add the salt and pepper
E8 : Add the lemon skin

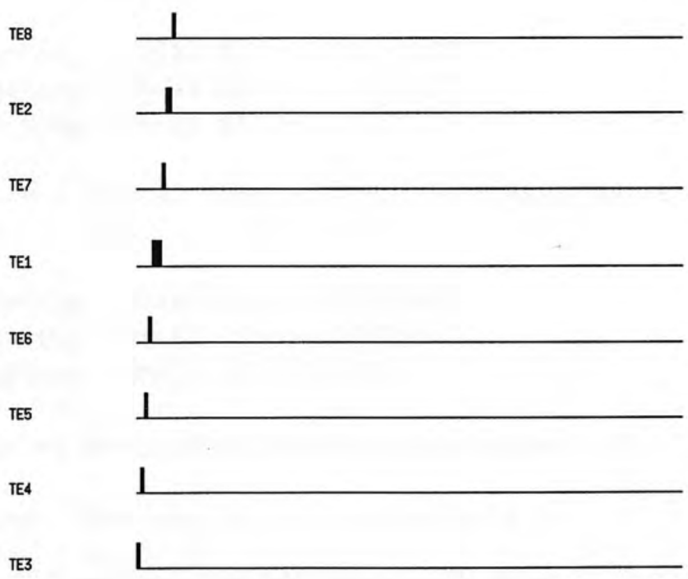


Figure A.3: The QTS and temporal structure of the LEMON DRESSING recipe

```
Asserting : TR(E7 E1) = (DISJOINT)

System : Does E2 require an agent to bring it to completion ?
User   : yes

Asserting : TR(E7 E2) = (DISJOINT)
Asserting : TR(E7 E6) = (A MI)

System : Does E6 require an agent to bring it to completion ?
User   : yes

Asserting : TR(E6 E1) = (DISJOINT)
Asserting : TR(E6 E2) = (DISJOINT)
Asserting : TR(E6 E5) = (A MI)

System : Does E5 require an agent to bring it to completion ?
User   : yes

Asserting : TR(E5 E1) = (DISJOINT)
Asserting : TR(E5 E2) = (DISJOINT)
Asserting : TR(E5 E4) = (A MI)

System : Does E4 require an agent to bring it to completion ?
User   : yes

Asserting : TR(E4 E1) = (DISJOINT)
Asserting : TR(E4 E2) = (DISJOINT)
Asserting : TR(E4 E3) = (A MI)

System : Does E3 require an agent to bring it to completion ?
User   : yes

Asserting : TR(E3 E1) = (DISJOINT)
Asserting : TR(E3 E2) = (DISJOINT)
Asserting : TR(E2 E1) = (A MI)

Finished Qualitative Relations Extraction

System : How long do you take to do E8 ?
User   : (3 3)
System : How long do you take to do E7 ?
User   : (3 3)
System : How long do you take to do E6 ?
User   : (3 3)
System : How long do you take to do E5 ?
```

```
User   : (3 3)
System : How long do you take to do E4 ?
User   : (3 3)
System : How long do you take to do E3 ?
User   : (3 3)
System : How long do you take to do E2 ?
User   : (5 5)
System : How long do you take to do E1 ?
User   : (5 5)

Starting the temporal reasoner

Asserting : TR(E2 E1) = (MI)
Asserting : TR(E2 E7) = (M)
Asserting : TR(E1 E6) = (MI)
Asserting : TR(E8 E7) = (MI)
Asserting : TR(E6 E5) = (MI)
Asserting : TR(E5 E4) = (MI)
Asserting : TR(E4 E3) = (MI)
```

A.5 Corn Soup

Source: From *rec.food.recipe*

2 tablespoons butter or margarine
1 teaspoon chili powder
1 cup each seeded and diced green and red peppers
$1\frac{1}{2}$ cups fresh or frozen thawed corn kernels
6 cups chicken broth
1 cup whipping cream
$\frac{1}{4}$ teaspoon salt

In a 3 quart pan, melt butter over medium heat. Add chili powder and peppers; cook, stirring, for 3 minutes. Add corn and broth. Bring to a boil. Simmer, uncovered, for about 5 minutes. Whip cream with salt until stiff. Pour soup into a tureen, add whipped cream, and stir lightly. Makes 8 to 10 servings.

Interesting Features

- This recipe contains interesting linguistic constructions like *Cook stirring for 3 minutes.* There is also a state expression like *it is stiff.*

Simplified Target

2 tablespoons of butter

1 teaspoon of chilli powder
1 cup of green peppers and 1 cup of red peppers
$1\frac{1}{2}$ cups of fresh corn kernels
6 cups of chicken broth
1 cup of whipping cream
$\frac{1}{4}$ teaspoon of salt

Melt the butter.
Add the chilli powder and the peppers.
Cook stirring for 3 minutes.
Add the corn and broth.
Bring to a boil.
Simmer for 5 minutes.
Whip cream with salt until it is stiff.
Pour the soup into a tureen.
Add whipped cream.
Stir.

The result of processing the simplified recipe is given in figure A.4. The system's transcript when extracting the qualitative Allen relations and the quantitative relations is as follows:

```
Starts dribbling to r7-ques (1993/6/25, 9:55:52).

> (start-interval-reasoner *relation-space*)

Asserting : TR(E12 E11) = (A MI)
Asserting : TR(E11 E9) = (A MI)
Asserting : TR(E11 E10) = (A MI)

System : Does E10 require an agent to bring it to completion ?
User   : yes

System : Does E9 require an agent to bring it to completion ?
User   : yes

Asserting : TR(E10 E9) = (DISJOINT)
Asserting : TR(E10 E8) = (A MI)

System : Does E1 require an agent to bring it to completion ?
User   : no

Asserting : TR(E9 E1) = (ANY)

System : Does E2 require an agent to bring it to completion ?
User   : yes
```

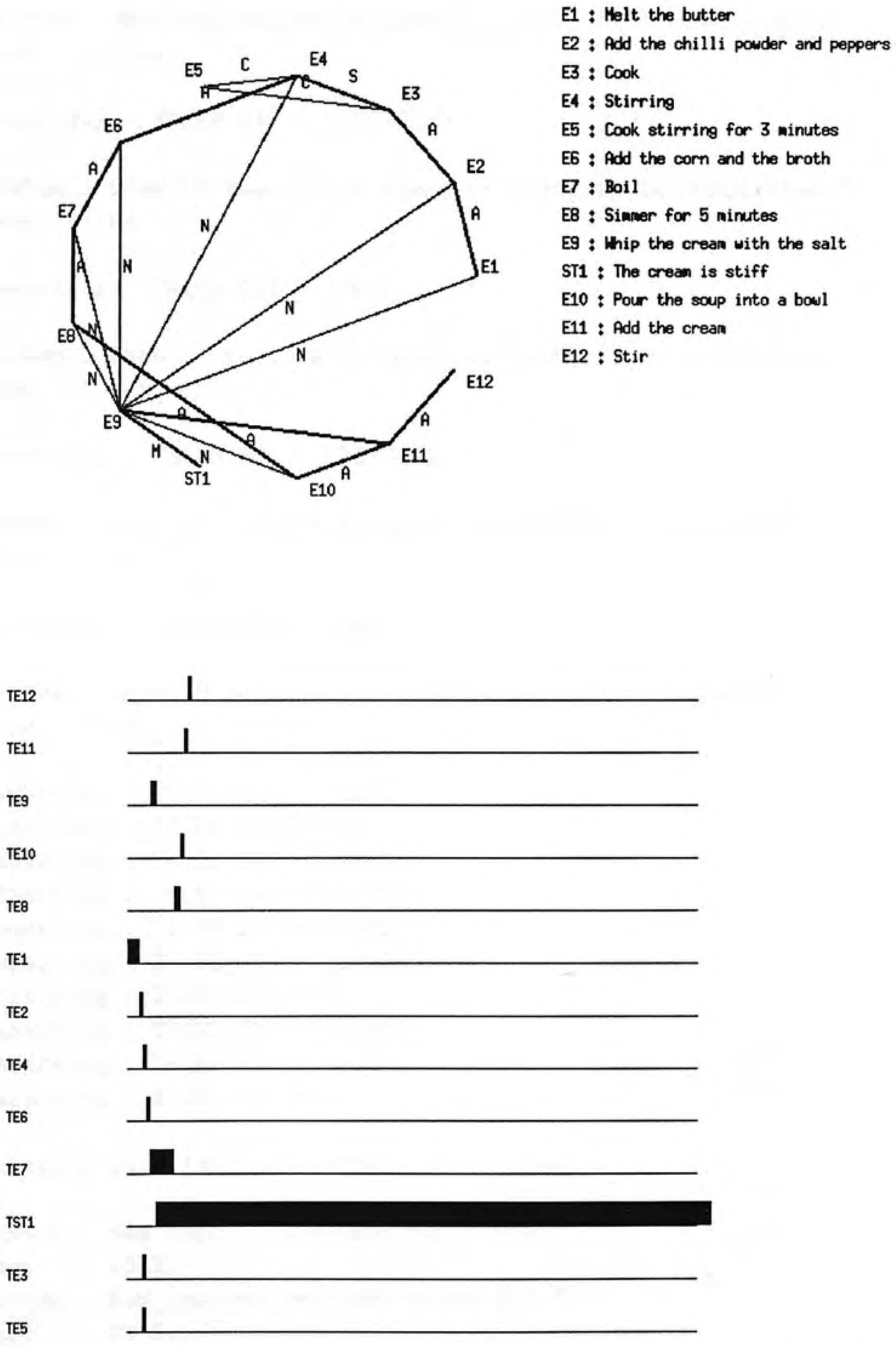

E1 : Melt the butter
E2 : Add the chilli powder and peppers
E3 : Cook
E4 : Stirring
E5 : Cook stirring for 3 minutes
E6 : Add the corn and the broth
E7 : Boil
E8 : Simmer for 5 minutes
E9 : Whip the cream with the salt
ST1 : The cream is stiff
E10 : Pour the soup into a bowl
E11 : Add the cream
E12 : Stir
TE12
TE11
TE9
TE10
TE8
TE1
TE2
TE4
TE6
TE7
TST1
TE3
TE5

Figure A.4: The QTS and temporal structure of the CORN SOUP recipe

```
Asserting : TR(E9 E2) = (DISJOINT)

System : Does E4 require an agent to bring it to completion ?
User   : yes

Asserting : TR(E9 E4) = (DISJOINT)

System : Does E3 require an agent to bring it to completion ?
User   : no

Asserting : TR(E9 E3) = (ANY)

System : Does E6 require an agent to bring it to completion ?
User   : yes

Asserting : TR(E9 E6) = (DISJOINT)

System : Does E7 require an agent to bring it to completion ?
User   : no

Asserting : TR(E9 E7) = (ANY)

System : Does E8 require an agent to bring it to completion ?
User   : no

Asserting : TR(E9 E8) = (ANY)
Asserting : TR(E9 ST1) = M
Asserting : TR(E8 E7) = (A MI)
Asserting : TR(E7 E6) = (A MI)
Asserting : TR(E6 E4) = (A MI)
Asserting : TR(E3 E2) = (A MI)
Asserting : TR(E4 E3) = =
Asserting : TR(E2 E1) = (A MI)
Asserting : TR(E5 E3) = =
Asserting : TR(E5 E4) = =

Finished Qualitative Relations Extraction

System : How long do you take to do E12 ?
User   : (3 3)
System : How long do you take to do E11 ?
User   : (3 3)
System : How long do you take to do E10 ?
User   : (3 3)
System : How long do you take to do E9 ?
```

```
User   : (5 5)
System : How long does E7 takes ?
User   : (20 20)
System : How long do you take to do E6 ?
User   : (3 3)
System : How long do you take to do E2 ?
User   : (3 3)
System : How long does E1 takes ?
User   : (15 15)

Starting the temporal reasoner

Asserting : TR(E7 E9) = (SI)
Asserting : TR(E7 E6) = (MI)
Asserting : TR(E8 E7) = (MI)
Asserting : TR(E12 E11) = (MI)
Asserting : TR(E11 E10) = (MI)
Asserting : TR(E10 E8) = (MI)
Asserting : TR(E6 E4) = (MI)
Asserting : TR(E3 E2) = (MI)
Asserting : TR(E2 E1) = (MI)
```

A.6 Chicken and Oyster Sauce Piquant

Source: From *rec.food.recipe*

2 tablespoons flour
6 tablespoons oil
3 large onions, finely chopped
$\frac{1}{2}$ cup chopped ham
8 to 10 chicken breasts, skinned
1 6-ounce can tomato paster
$\frac{1}{2}$ can water
3 cloves garlic
$\frac{1}{2}$ cup celery, finely chopped
$\frac{1}{4}$ cup parsley, finely chopped
$\frac{1}{2}$ bunch green onions, thinly sliced
1 3-ounce bottle pitted green olives, finely chopped
1 small jar sour pickles, finely chopped
$\frac{1}{2}$ rind of a lemon, grated
2 6-ounce cans slice mushrooms with liquid
salt and pepper to taste
1 pint oysters
5 cups white wine

Brown flour in oil very slowly, until dark brown. This takes about 30 minutes. Add chopped onions and cook slowly until tender. Then add chopped ham and cook a few minutes. Add raw chicken breasts and stir thoroughly to coat each piece. Add tomato paste and water. Simmer for 30 to 45 minutes.

Add olives, sour pickle, lemon rind and cook for a few minutes, turning chicken to coat. Add mushrooms with the liquid. Season with salt and peppers. Simmer (over low heat) for 4 to 5 hours. Stir occasionally to keep chicken from sticking.

About 30 minutes before serving, add $\frac{1}{2}$ cup wine and bring to a simmer. Add oysters with juice and cook until oysters curl at the edges. Just before serving, add another $4\frac{1}{2}$ cup of wine. Serve chicken on a bed of rice with sauce poured over.

Interesting Features

- This recipe contains many expressions on quantitative information in this recipe like *30 minutes before* and *for 4 to 5 hours.*

Other kinds of simplifications

- We replace the NP *a few minutes* with *three minutes.*

Simplified Target

2 tablespoons of flour
6 tablespoons of oil
3 large onions
$\frac{1}{2}$ cup of chopped ham
8 to 10 chicken breasts
1 can of tomato paste
$\frac{1}{2}$ can of water
3 cloves of garlic
$\frac{1}{2}$ cup of celery
$\frac{1}{4}$ cup of parsley
$\frac{1}{2}$ bunch of green onions
1 bottle of pitted green olives
1 jar of sour pickles
$\frac{1}{2}$ rind of a lemon
2 cans of slice mushrooms
some salt and pepper
1 pint of oysters
5 cups of white wine

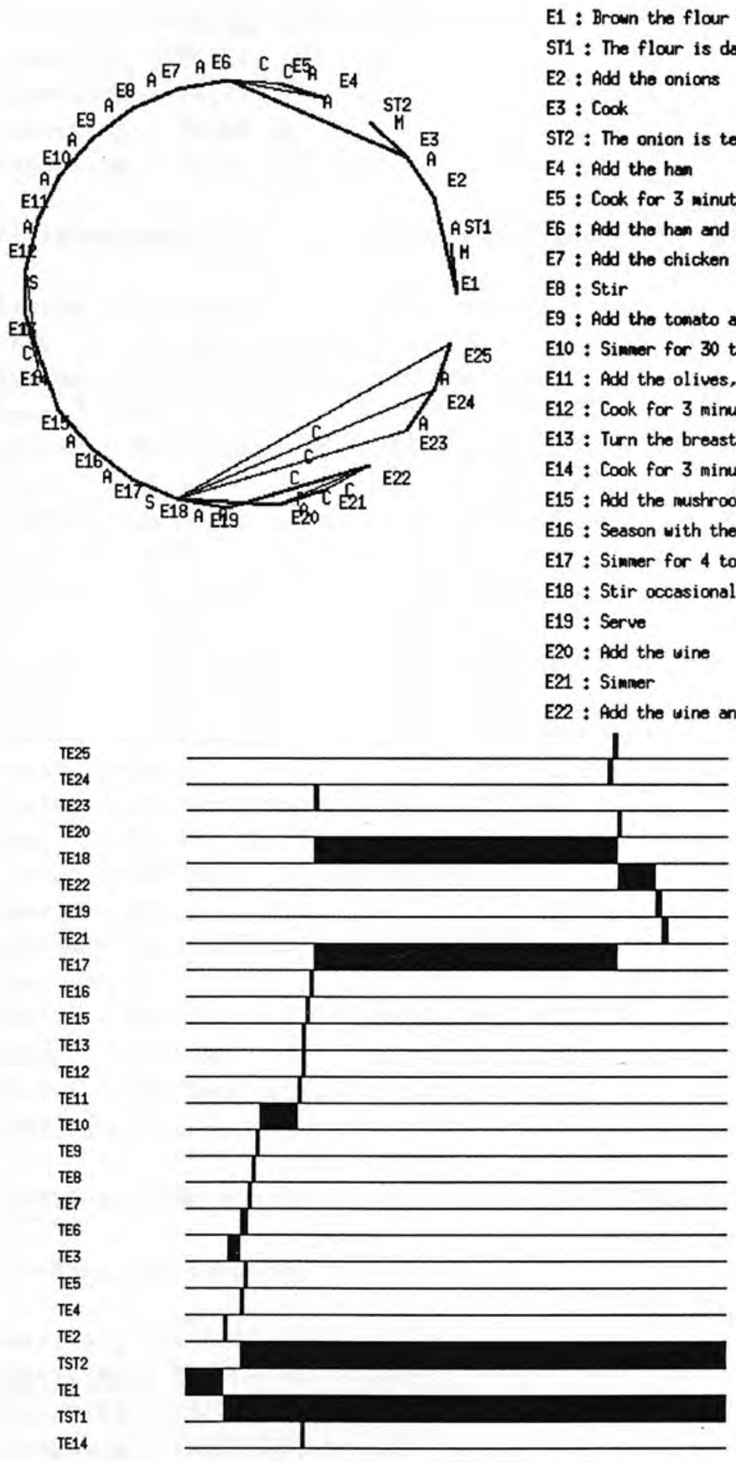
Brown flour for 30 minutes until it is dark brown.
Add the onions.
Cook until it is tender.
Then add the ham.
Cook for 3 minutes.

Add the chicken breasts.
Stir.
Add the tomato and the water.
Simmer for 30 to 45 minutes.
Add the olives, the pickle and the lemon rind.
Cook for 3 minutes turning the breasts.
Add the mushrooms.
Season with salt and peppers.
Simmer for 4 to 5 hours.
Stir occasionally.
30 minutes before serving, add the wine and simmer.

The result of processing the simplified recipe is given in figure A.5.

The system's transcript when extracting the qualitative Allen relations and the quantitative relations is as follows:

```
Starts dribbling to r9-query (1993/6/25, 10:35:7).

> (start-interval-reasoner *relation-space*)

Asserting : TR(E25 E24) = (A MI)
Asserting : TR(E24 E23) = (A MI)
Asserting : TR(E20 E18) = (A MI)
Asserting : TR(E22 E19) = (B O D M)
Asserting : TR(E21 E20) = (A MI)
Asserting : TR(E19 E18) = (A MI)
Asserting : TR(E18 E17) = =
Asserting : TR(E17 E16) = (A MI)
Asserting : TR(E16 E15) = (A MI)
Asserting : TR(E15 E13) = (A MI)
Asserting : TR(E12 E11) = (A MI)
Asserting : TR(E13 E12) = =
Asserting : TR(E11 E10) = (A MI)
Asserting : TR(E10 E9) = (A MI)
Asserting : TR(E9 E8) = (A MI)
Asserting : TR(E8 E7) = (A MI)
Asserting : TR(E7 E6) = (A MI)
Asserting : TR(E6 E3) = (A MI)
Asserting : TR(E5 E4) = (A MI)
Asserting : TR(E3 E2) = (A MI)
Asserting : TR(E3 ST2) = M
Asserting : TR(E2 E1) = (A MI)
Asserting : TR(E1 ST1) = M
Asserting : TR(E18 E23) = SI
Asserting : TR(E18 E24) = DI
```



E1 : Brown the flour for 30 minutes
ST1 : The flour is dark brown
E2 : Add the onions
E3 : Cook
ST2 : The onion is tender
E4 : Add the ham
E5 : Cook for 3 minutes
E6 : Add the ham and cook for 3 minutes
E7 : Add the chicken breasts
E8 : Stir
E9 : Add the tomato and the water
E10 : Simmer for 30 to 45 minutes
E11 : Add the olives, the pickles and the lemon rind
E12 : Cook for 3 minutes
E13 : Turn the breasts
E14 : Cook for 3 minutes turning the breasts
E15 : Add the mushrooms
E16 : Season with the salt and pepper
E17 : Simmer for 4 to 5 hours
E18 : Stir occasionally
E19 : Serve
E20 : Add the wine
E21 : Simmer
E22 : Add the wine and simmer

Figure A.5: The QTS and temporal structure of the CHICKEN AND OYSTER SAUCE PIQUANT recipe

```
Asserting : TR(E18 E25) = FI
Asserting : TR(E22 E20) = SI
Asserting : TR(E22 E21) = FI
Asserting : TR(E14 E12) = =
Asserting : TR(E14 E13) = =
Asserting : TR(E6 E4) = SI
Asserting : TR(E6 E5) = FI

Finished Qualitative Relations Extraction

System : How long do you take to do E21 ?
User   : (20 20)
System : How long do you take to do E20 ?
User   : (3 3)
System : How long do you take to do E19 ?
User   : (3 3)
System : How long do you take to do E16 ?
User   : (3 3)
System : How long do you take to do E15 ?
User   : (3 3)
System : How long do you take to do E11 ?
User   : (3 3)
System : How long do you take to do E9 ?
User   : (3 3)
System : How long do you take to do E8 ?
User   : (3 3)
System : How long do you take to do E7 ?
User   : (3 3)
System : How long do you take to do E4 ?
User   : (3 3)
System : How long do you take to do E3 ?
User   : (20 20)
System : How long do you take to do E2 ?
User: (3 3)

Asserting Quantitative Info : (30 <= LEFT TE19 - LEFT TE22 <= 30)

Starting the temporal reasoner

Asserting : TR(E17 E16) = (MI)
Asserting : TR(E10 E9) = (MI)
Asserting : TR(E22 E19) = (O)
Asserting : TR(E3 E2) = (MI)
Asserting : TR(E6 E3) = (MI)
Asserting : TR(E25 E24) = (MI)
Asserting : TR(E20 E18) = (MI)
Asserting : TR(E16 E15) = (MI)
Asserting : TR(E15 E13) = (MI)
Asserting : TR(E12 E11) = (MI)
Asserting : TR(E11 E10) = (MI)
Asserting : TR(E9 E8) = (MI)
Asserting : TR(E8 E7) = (MI)
Asserting : TR(E7 E6) = (MI)
Asserting : TR(E5 E4) = (MI)
Asserting : TR(E2 E1) = (MI)
```

For this recipe, the EVENT-DECOMPOSER decomposes E18 (the event described by *stir occasionally*). The system's transcript at this point is as follows:

```
> (decomposition-reasoner *event-stack*)

System : Can you carry out E18 in one go ?
User   : no

System : How long does it takes to do one repetition of E18 ?
User   : (5 5)

Modelling the action of STIRRING-MATTER  \* the first stirring *\

Finish Modelling

Modelling the action of STIRRING-MATTER \* the second stirring *\

Finish Modelling

Modelling the action of STIRRING-MATTER \* the third stirring *\

Finish Modelling
```

A.7 Cream of Butter Bean Soup

Source: Rose Elliot's *The Bean Book*

4 oz butter beans
1 large onion
1 medium-sized potato
2 carrots

2 sticks celery
1 oz butter
$1\frac{1}{2}$ pints water or unsalted stock
$\frac{1}{2}$ pint milk
a bouquet garni: a couple of sprigs of parsley, a sprig of thyme and a bayleaf, tied together
4-6 tablespoons of cream
sea salt
freshly ground black pepper
grated nutmeg

Soak the butter beans, then drain and rinse them. Peel and chop the onion and potato; scrape and chop the carrots; slice the celery. Melt the butter in a large saucepan and add the vegetables; saute them for 7-8 minutes, but don't them brown, then add the butter beans, water or stock, the milk and the bouquet garni. Simmer gently, with a lid half on the saucepan, for about 1.25 hours or until the butter beans are tender. Remove the herbs, then liquidise the soup, stir in the cream and add the sea salt, freshly ground black pepper and nutmeg to taste. Reheat the soup, but don't let it boil. Serve each bowl sprinkled with croutons.

Simplified Target

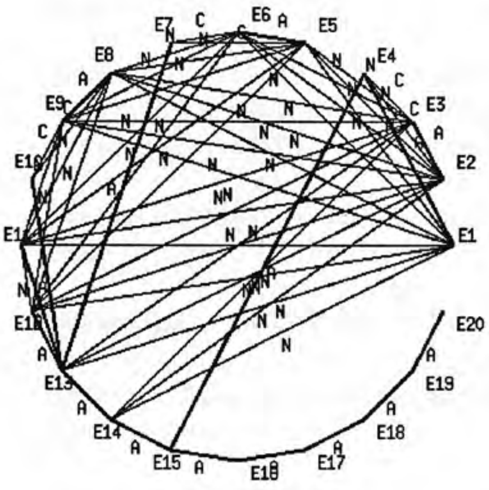
4 oz of butter beans
1 large onion
1 medium potato
2 carrots
2 sticks of celery
1 ounce of butter
1.5 pints of unsalted stock
0.5 pints of milk
four tablespoons of cream
some sea salt
some freshly ground black pepper
some grated nutmeg

Soak the butter beans.
Then, drain and rinse them.
Peel and chop the onion and potato.
Scrape and chop the carrots.
Slice the celery.
Melt the butter.
Add the vegetables.
Saute them for 7-8 minutes.
Then add the butter beans, the stock and the milk.
Simmer for about 1.25 hours.
Liquidise the soup.
Stir in the cream.
Add the sea salt, freshly ground black pepper and nutmeg.

Reheat the soup.

The result of processing the simplified recipe is given in figure A.6. The system's transcript when extracting the qualitative Allen relations and the quantitative relations is as follows:

```
Starts dribbling to r4-query (1993/6/25, 11:9:9).

> (start-interval-reasoner *relation-space*)

Asserting : TR(E20 E19) = (A MI)
Asserting : TR(E19 E18) = (A MI)
Asserting : TR(E18 E17) = (A MI)
Asserting : TR(E17 E16) = (A MI)
Asserting : TR(E16 E15) = (A MI)
Asserting : TR(E15 E4) = (A MI)
Asserting : TR(E15 E14) = (A MI)

System : Does E14 require an agent to bring it to completion ?
User   : no

System : Does E1 require an agent to bring it to completion ?
User   : no

Asserting : TR(E14 E1) = (ANY)

System : Does E3 require an agent to bring it to completion ?
User   : yes

Asserting : TR(E14 E3) = (ANY)

System : Does E2 require an agent to bring it to completion ?
User   : yes

Asserting : TR(E14 E2) = (ANY)
Asserting : TR(E14 E13) = (A MI)

System : Does E13 require an agent to bring it to completion ?
User   : yes

Asserting : TR(E13 E1) = (ANY)
Asserting : TR(E13 E3) = (DISJOINT)
Asserting : TR(E13 E2) = (DISJOINT)
Asserting : TR(E13 E7) = (A MI)
Asserting : TR(E13 E10) = (A MI)
```



E1 : Soak the butter beans
E2 : Drain the beans
E3 : Rinse the beans
E4 : Drain and Rinse the beans
E5 : Peel the onion and potato
E6 : Chop the onion and potato
E7 : Peel and chop the onion and potato
E8 : Scrape the carrots
E9 : Chop the carrots
E10 : Scrape and chop the carrots
E11 : Slice the celery
E12 : Melt the butter
E13 : Add the vegetables
E14 : Saute them for 7 to 8 minutes
E15 : Add the butter beans, the stock and the milk
E16 : Simmer the soup for about 1.25 hours
E17 : Liquidise the soup
E18 : Stir in the cream
E19 : Add the sea salt, freshly ground black pepper and nutmeg
E20 : Reheat the soup

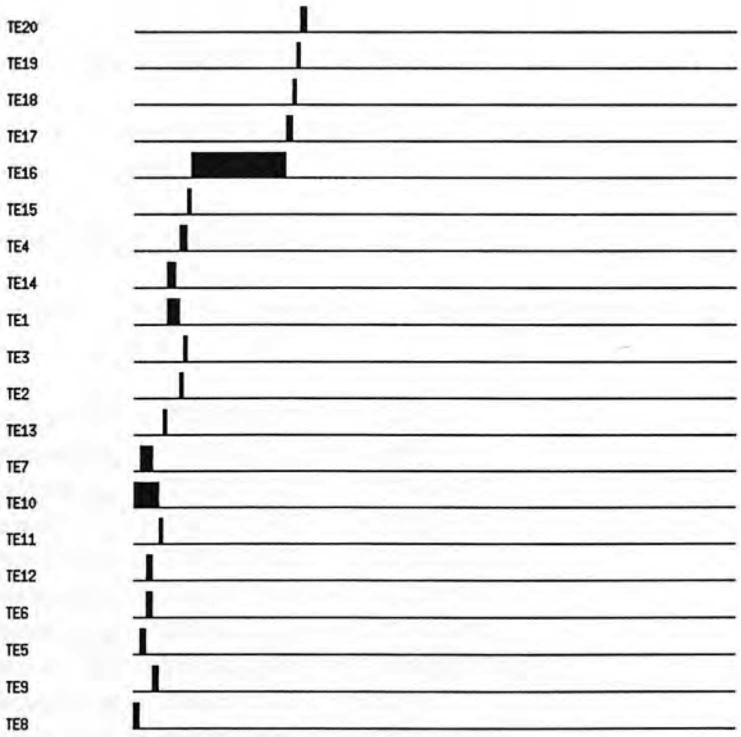


Figure A.6: The QTS and temporal structure of the CREAM OF BUTTERBEAN SOUP recipe

```
Asserting : TR(E13 E11) = (A MI)
Asserting : TR(E13 E12) = (A MI)

System : Does E12 require an agent to bring it to completion ?
User   : no

Asserting : TR(E12 E1) = (ANY)
Asserting : TR(E12 E3) = (ANY)
Asserting : TR(E12 E2) = (ANY)

System : Does E6 require an agent to bring it to completion ?
User   : yes

Asserting : TR(E12 E6) = (ANY)

System : Does E5 require an agent to bring it to completion ?
User   : yes

Asserting : TR(E12 E5) = (ANY)

System : Does E9 require an agent to bring it to completion ?
User   : yes

Asserting : TR(E12 E9) = (ANY)

System : Does E8 require an agent to bring it to completion ?
User   : yes

Asserting : TR(E12 E8) = (ANY)

System : Does E11 require an agent to bring it to completion ?
User   : yes

Asserting : TR(E12 E11) = (ANY)
Asserting : TR(E11 E1) = (ANY)
Asserting : TR(E11 E3) = (DISJOINT)
Asserting : TR(E11 E2) = (DISJOINT)
Asserting : TR(E11 E6) = (DISJOINT)
Asserting : TR(E11 E5) = (DISJOINT)
Asserting : TR(E11 E9) = (DISJOINT)
Asserting : TR(E11 E8) = (DISJOINT)
Asserting : TR(E9 E1) = (ANY)
Asserting : TR(E9 E3) = (DISJOINT)
Asserting : TR(E9 E2) = (DISJOINT)
Asserting : TR(E9 E6) = (DISJOINT)
Asserting : TR(E9 E5) = (DISJOINT)
Asserting : TR(E8 E1) = (ANY)
Asserting : TR(E8 E3) = (DISJOINT)
Asserting : TR(E8 E2) = (DISJOINT)
Asserting : TR(E8 E6) = (DISJOINT)
Asserting : TR(E8 E5) = (DISJOINT)
Asserting : TR(E9 E8) = (A MI)
Asserting : TR(E6 E1) = (ANY)
Asserting : TR(E6 E3) = (DISJOINT)
Asserting : TR(E6 E2) = (DISJOINT)
Asserting : TR(E5 E1) = (ANY)
Asserting : TR(E5 E3) = (DISJOINT)
Asserting : TR(E5 E2) = (DISJOINT)
Asserting : TR(E6 E5) = (A MI)
Asserting : TR(E4 E1) = (A MI)
Asserting : TR(E3 E2) = (A MI)
Asserting : TR(E10 E8) = SI
Asserting : TR(E10 E9) = FI
Asserting : TR(E7 E5) = SI
Asserting : TR(E7 E6) = FI
Asserting : TR(E4 E2) = SI
Asserting : TR(E4 E3) = FI

Finished Qualitative Relations Extraction

System : How long do you take to do E20 ?
User   : (10 10)
System : How long do you take to do E19 ?
User   : (3 3)
System : How long do you take to do E18 ?
User   : (3 3)
System : How long do you take to do E17 ?
User   : (5 5)
System : How long do you take to do E15 ?
User   : (3 3)
System : How long do you take to do E13 ?
User   : (3 3)
System : How long does E12 takes ?
User   : (5 5)
System : How long do you take to do E11 ?
User   : (3 3)
System : How long do you take to do E9 ?
User   : (5 5)
System : How long do you take to do E8 ?
User   : (5 5)
System : How long do you take to do E6 ?
User   : (5 5)
```

```
System : How long do you take to do E5 ?
User   : (5 5)
System : How long do you take to do E3 ?
User   : (3 3)
System : How long do you take to do E2 ?
User   : (3 3)
System : How long does E1 takes ?
User   : (15 15)

Asserting : TR(E16 E15) = (MI)
Asserting : TR(E1 E14) = (SI)
Asserting : TR(E1 E13) = (MI)
Asserting : TR(E20 E19) = (MI)
Asserting : TR(E4 E1) = (MI)
Asserting : TR(E17 E16) = (MI)
Asserting : TR(E12 E6) = (=)
Asserting : TR(E12 E5) = (MI)
Asserting : TR(E12 E9) = (M)
Asserting : TR(E12 E11) = (P)
Asserting : TR(E9 E11) = (M)
Asserting : TR(E8 E5) = (M)
Asserting : TR(E19 E18) = (MI)
Asserting : TR(E18 E17) = (MI)
Asserting : TR(E15 E4) = (MI)
Asserting : TR(E13 E11) = (MI)
Asserting : TR(E3 E2) = (MI)
```

A.8 Instant Soup

Source: From the back of an Instant Soup Packet

Empty the contents from the packet.
Add one pint of cold water.
Boil stirring constantly.
Reduce the heat.
Simmer for twenty minutes stirring occasionally.

Simplified Target

There are no simplifications made to this recipe.

The result of processing the simplified recipe is given in figure A.7. The system's transcript when extracting the qualitative Allen relations and the quantitative relations is as follows:

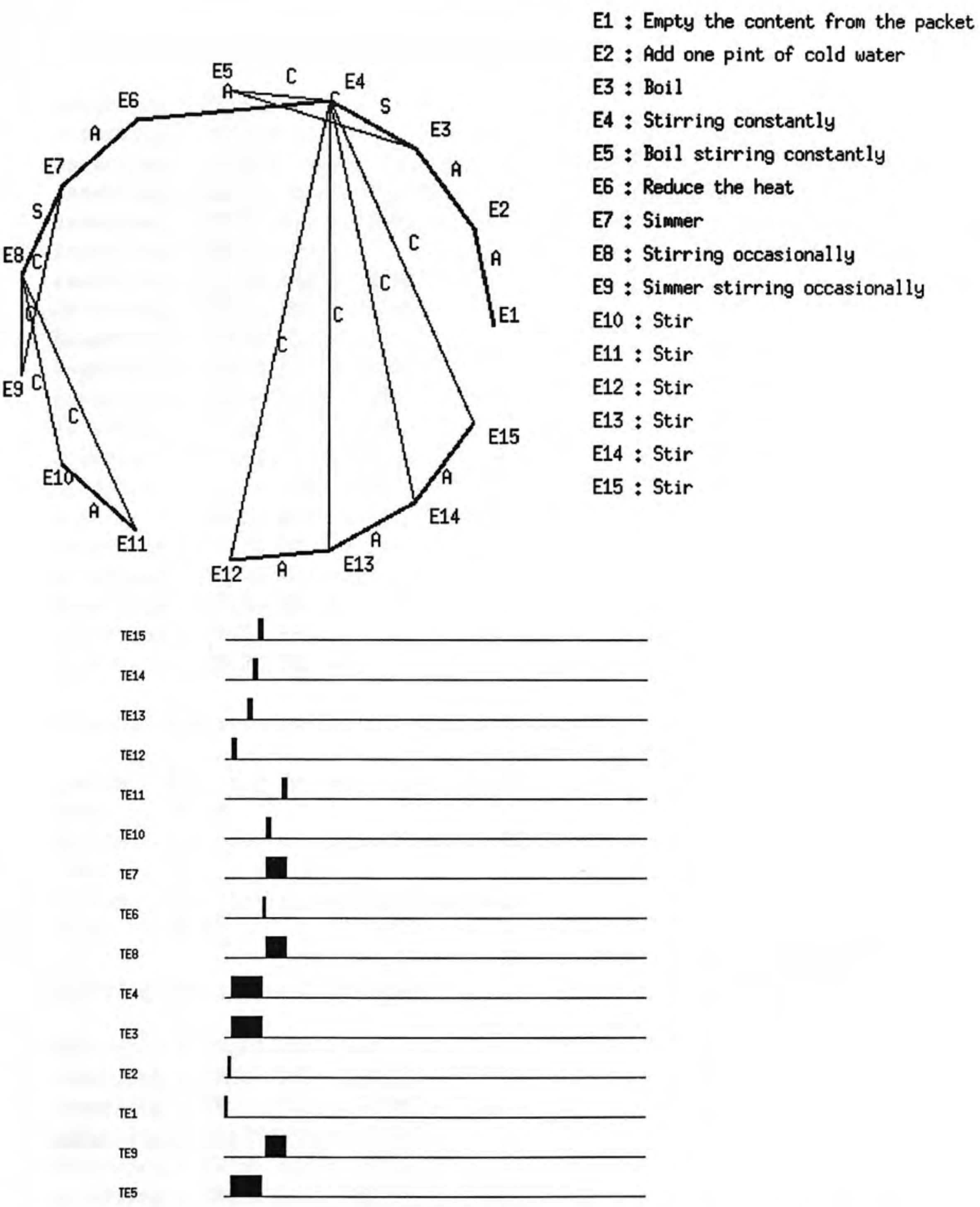


Figure A.7: The QTS and temporal structure of the INSTANT SOUP recipe

```
Starts dribbling to ../recipe/r2-query (1993/6/25, 11:25:13).

> (start-interval-reasoner *relation-space*)

Asserting : TR(E15 E14) = (A MI)
Asserting : TR(E14 E13) = (A MI)
Asserting : TR(E13 E12) = (A MI)
Asserting : TR(E11 E10) = (A MI)
Asserting : TR(E7 E6) = (A MI)
Asserting : TR(E8 E7) = =
Asserting : TR(E6 E4) = (A MI)
Asserting : TR(E3 E2) = (A MI)
Asserting : TR(E4 E3) = =
Asserting : TR(E2 E1) = (A MI)
Asserting : TR(E4 E12) = SI
Asserting : TR(E4 E13) = DI
Asserting : TR(E4 E14) = DI
Asserting : TR(E4 E15) = FI
Asserting : TR(E8 E10) = SI
Asserting : TR(E8 E11) = FI
Asserting : TR(E9 E7) = =
Asserting : TR(E9 E8) = =
Asserting : TR(E5 E3) = =
Asserting : TR(E5 E4) = =

Finished Qualitative Relations Extraction

System : How long do you take to do E6 ?
User   : (3 3)
System : How long do you take to do E2 ?
User   : (3 3)
System : How long do you take to do E1 ?
User   : (5 5)

Starting the temporal reasoner

Asserting : TR(E3 E2) = (MI)
Asserting : TR(E7 E6) = (MI)
Asserting : TR(E15 E14) = (MI)
Asserting : TR(E14 E13) = (MI)
Asserting : TR(E6 E4) = (MI)
Asserting : TR(E2 E1) = (MI)
```

For this recipe, the EVENT DECOMPOSER decomposes E8 (the event described by *stirring occasionally*) and E4 (the event described by *stirring constantly*). Note that E4 has more decompositions than E8. The system's transcript is as follows:

```
> (decomposition-reasoner *event-stack*)

System : Can you carry out E8 in one go ?
User   : no

System : How long does it takes to do one repetition of E8 ?
User   : (5 5)

Modelling the action of STIRRING-MATTER   \* First Stirring *\

Finish Modelling

Modelling the action of STIRRING-MATTER    \* Second Stirring *\

Finish Modelling

System : Can you carry out E4 in one go ?
User   : no

System : How long do you take to do E3 ?
User   : (30 30)

System : How long does it takes to do one repetition of E4 ?
User   : (5 5)

Modelling the action of STIRRING-MATTER   \* First Stirring *\

Finish Modelling

Modelling the action of STIRRING-MATTER   \* Second Stirring *\

Finish Modelling

Modelling the action of STIRRING-MATTER   \* Third Stirring *\

Finish Modelling

Modelling the action of STIRRING-MATTER   \* Fourth Stirring *\

Finish Modelling
```

A.9 A Partial Recipe

Source: From [Dale 92]

Cook the rice, cool. Sprinkle apple and banana with lemon juice and add to rice. Steep raisins in a little boiling water for half an hour to plump; drain, and add to rice with sunflower seeds.

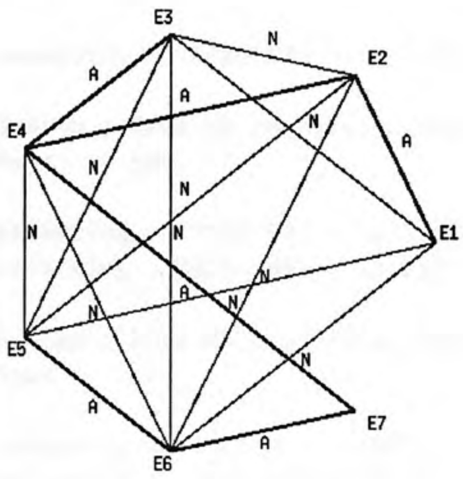
Simplified Target

some rice
an apple
a banana
some lemon juice
some raisins

Cook the rice.
Cool.
Sprinkle the apple and banana with the lemon juice.
Add to the rice.
Steep the raisins in some water for thirty minutes.
Drain
Add to the rice.

The result of processing the simplified recipe is given in figure A.8. The system's transcript when extracting the qualitative Allen relations and the quantitative relations is as follows:

```
Starts dribbling to ../recipe/r3-query (1993/6/25, 11:45:18).

> (start-interval-reasoner *relation-space*)

Asserting : TR(E7 E4) = (A MI)
Asserting : TR(E7 E6) = (A MI)

System : Does E6 require an agent to bring it to completion ?
User   : yes

System : Does E1 require an agent to bring it to completion ?
User   : no

Asserting : TR(E6 E1) = (ANY)

System : Does E2 require an agent to bring it to completion ?
User   : no
```



E1 : Cook the rice
E2 : Cool
E3 : Sprinkle the apple and banana with some lemon juice
E4 : Add to the rice
E5 : Steep the raisins in some water for thirty minutes
E6 : Drain
E7 : Add to the rice

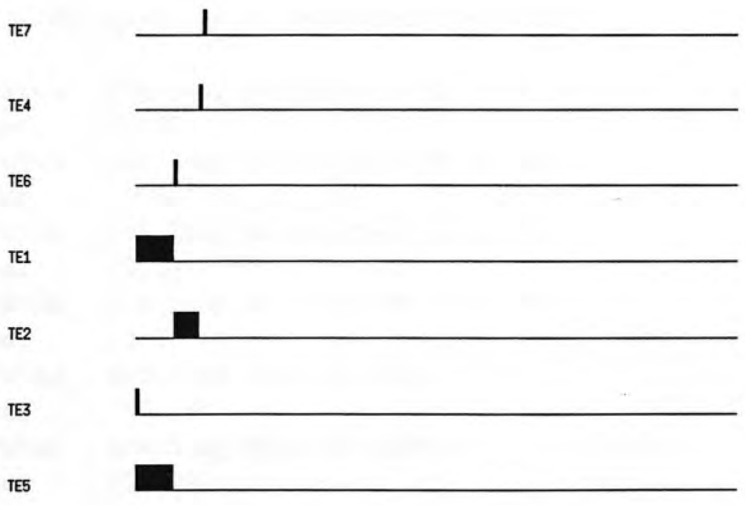


Figure A.8: The QTS and temporal structure of the PARTIAL recipe

```
Asserting : TR(E6 E2) = (ANY)

System : Does E3 require an agent to bring it to completion ?
User   : yes

Asserting : TR(E6 E3) = (DISJOINT)

System : Does E4 require an agent to bring it to completion ?
User   : yes

Asserting : TR(E6 E4) = (DISJOINT)
Asserting : TR(E6 E5) = (A MI)

System : Does E5 require an agent to bring it to completion ?
User   : no

Asserting : TR(E5 E1) = (ANY)
Asserting : TR(E5 E2) = (ANY)
Asserting : TR(E5 E3) = (ANY)
Asserting : TR(E5 E4) = (ANY)
Asserting : TR(E4 E2) = (A MI)
Asserting : TR(E4 E3) = (A MI)
Asserting : TR(E3 E1) = (ANY)
Asserting : TR(E3 E2) = (ANY)
Asserting : TR(E2 E1) = (A MI)

Finished Qualitative Relations Extraction

System : How long do you take to do E7 ?
User   : (3 3)
System : How long do you take to do E6 ?
User   : (5 5)
System : How long do you take to do E4 ?
User   : (3 3)
System : How long do you take to do E3 ?
User   : (3 3)
System : How long does E2 takes ?
User   : (10 10)
System : How long does E1 takes ?
User   : (20 20)

Starting the temporal reasoner

Asserting : TR(E5 E1) = (SI)
Asserting : TR(E5 E2) = (FI)
Asserting : TR(E5 E3) = (SI)
Asserting : TR(E5 E4) = (M)
Asserting : TR(E6 E4) = (MI)
Asserting : TR(E7 E6) = (MI)
```

Appendix B

The Ruleset and Rule Language

In this appendix, we present the language used to specify a ruleset and the rules in the ruleset. The language will be expressed in a BNF grammar using the following notations:

- An item in small cap such as RULESET is a non-terminal.
- An item in italic such as *label*$_1$ is an atomic value.
- Where there are a potentially infinite set of values, these are represented by a gloss in brackets.
- An item enclosed in braces ({}) indicates that the item is optional.
- An item suffixed with a '+' means that the expansion of that item in the grammar may appear one or more time.
- The symbol | is used to indicate disjunction.

B.1 The Ruleset Language

The BNF of the ruleset language is as follows:

RULESET	::- HEADER RULE+
HEADER	::- (in-ruleset ⟨name⟩) (input-node-label INPUT$_1$ ... INPUT$_n$) (output-node-label OUTPUT) {(firing-strategy STRATEGY)} {(maintain-kbe-history OUTPUT)} {(domain-modeller OUTPUT)} {(integrate-situation OUTPUT)}
INPUT	::- ⟨a feature identifier⟩ | ⟨a variable⟩
OUTPUT	::- ⟨a feature identifier⟩
STRATEGY	::- *first-tried-succeed*

The possible expansion of RULE is given in section B.2. We now explain the meaning of the constructs appearing in HEADER:

(in-ruleset ⟨name⟩) This specifies the name of the ruleset.

(input-node-label INPUT$_1$ … INPUT$_n$**)** This specifies the input 'parameters' which the rules in the ruleset reason about.

(output-node-label OUTPUT**)** This specifies the identifier of the feature structure output by the ruleset.

(maintain-kbe-history OUTPUT**)** This maintains the discourse model by adding any new object or event represented in OUTPUT to the model.

(firing-strategy STRATEGY**)** This specifies the strategy used for rule firing. The default strategy is *first-tried-succeed* which means that the ruleset returns the result of the first rule in the ruleset which fires. If no rules in the ruleset fire, then the ruleset returns failure.

(domain-modeller OUTPUT**)** This invokes the domain modeller to simulate the event represented in OUTPUT.

(integrate-situation OUTPUT**)** This invokes the integrator module to integrate the current situation represented in OUTPUT into the preceding QTS.

B.2 The Rule Language

RULE	::- PREMISE+ :- ACTION+
PREMISE	::- PRED | PRED-EQ | PRED-VAR-CL | CSP-CL
PRED	::- F-PATH | -F-PATH | KB-QUERY | RULESET-CL | ⟨a Lisp Predicate⟩
F-PATH	::- ⟨F a b c⟩
PRED-EQ	::- PRED is ⟨a value⟩ | PRED isnot ⟨a value⟩
KB-QUERY	::- ⟨a query to the frame knowledge base of substances⟩
PRED-VAR-CL	::- ⟨a variable⟩ = PRED
CSP-CL	::- (constraint-find VAR$_1$ … VAR$_n$:domain DOM$_1$ … DOM$_n$:constraints CONSTRAINTS)
VAR$_i$	::- ⟨the i_{th} variable⟩
DOM	::- ⟨the domain of the i_{th} variable⟩
CONSTRAINTS	::- PRED | PRED-EQ | (action-constraints VAR$_1$ … VAR$_n$)
RULESET-CL	::- (invoke-ruleset RULESET-NAME INPUT$_1$ … INPUT$_n$) | ITERATIVE-CL
INPUT$_i$	::- ⟨a feature identifier⟩ | ⟨a variable⟩
ITERATIVE-CL	::- (iterate-ruleset RULESET-NAME (INPUT$_1$ … INPUT$_n$)+)
RULESET-NAME	::- ⟨the name of a ruleset⟩
ACTION	::- F-PATH = ACTION-VALUE | ⟨a Lisp Function⟩
ACTION-VALUE	::- F-PATH | ACTION-FN

ACTION-FN	::- (invoke-ruleset RULESET-NAME INPUT$_1$ ... INPUT$_n$) | (case-attr-mapping ⟨an event substance⟩ (CASE CASE-VALUE)+) | ITERATIVE-CL
CASE	::- ⟨the possible cases of the NP⟩

We will now explain the meanings of the constructs found in a rule premise and action.

Premise Clause

We divide the possible clauses in a rule premise into four different classes. As TASTE is implemented in Lisp, we say that a premise clause is true if the clause returns a non-null value. Otherwise, the premise clause is false.

The four classes are described below:

PRED

In this class are the following predicates:

⟨F a b c⟩: This predicate is true if there exists such a path in the feature structure whose feature identifier is F.

-⟨F a b c⟩: This predicate is true if there does not exist such a path in the feature structure whose feature identifier is F.

KB-QUERY: This includes those predicates which query the frame knowledge base of substances.

(invoke-ruleset RULESET-NAME INPUT$_1$ ... INPUT$-n$**):** This predicate is true if the ruleset invoked reports success.

(iterate-ruleset RULESET-NAME (INPUT$_1$ ... INPUT$_n$)+**):** This iteratively invokes the ruleset indicated by RULESET-NAME over the list of input parameters given by (INPUT$_1$ ... INPUT$_n$). The predicate is true if each iteration of the ruleset reports success.

⟨a Lisp Predicate⟩: The predicate is true if the Lisp function returns a non-null value.

PRED-EQ

In this class are the following predicates:

PRED **is ⟨a value⟩:** This predicate is true if the value returned by PRED is equal to the value indicated.

PRED **isnot ⟨a value⟩:** This predicate is true if the value returned by PRED is not equal to the value indicated.

PRED-VAR-CL

In this class, we have the following predicate:

PRED = **⟨a variable⟩:** This predicate is true if PRED is true. In addition, there is a side effect of binding the indicated variable to the value returned by PRED.

CSP-CL

This sets up a constraint satisfaction problem and the construct is:

- (constraint-find $\text{VAR}_1 \ldots \text{VAR}_n$
  :domain $\text{DOM}_1 \ldots \text{DOM}_n$
  :constraints CONSTRAINTS)

The predicate is true if the CSP can be solved. Depending on the type of the variables in VAR_i , the domain of VAR_i can be:

1. If VAR_i is that associated with a verb or a noun, then the domain is the possible lexical senses of the verb or noun. For an ambiguous verb or noun, the domain will be a set of cardinality greater than 1.
2. If VAR_i is that associated with an NP, then the domain is the list of objects in the discourse model.
3. If VAR_i is that associated with a CLAUSE, then the domain is the list of situations in the discourse model.

The constraints in the CSP are either PRED, PRED-EQ or the constraint given by:

- (action-constraints $\text{VAR}_1 \ldots \text{VAR}_n$)

This constraint is verified by verifying the constraints stored in the action which VAR_1 gets bound to.

Action Clause

The ACTION part of the rule is executed when the PREMISE succeeds. The possible actions that can be carried out are:

F-PATH = ACTION-VALUE: This assigns that part of the output feature structure indicated by F-PATH to the value given by ACTION-VALUE. The value assigned can be from one of the following sources:

- F-PATH: The value is simply retrieved from that part of the input feature structure indicated by F-PATH.
- **(invoke-ruleset** RULESET-NAME $\text{INPUT}_1 \ldots \text{INPUT}_n$**):** The value is the value returned by the ruleset.
- **(case-attribute-mapping ⟨an event substance⟩** (CASE CASE-VALUE)+**):** This maps the case role of the objects to the frame role of the event substance.

⟨a Lisp Function⟩: This is any Lisp function.

B.3 Examples

We give some examples of the rulesets and rules employed in TASTE in this section:

```
;;; ****************************************************
;;; KBE-Sent-Top Ruleset
;;;
;;; Comment:
;;;
;;; a) The top-level ruleset is responsible for building up
;;;    the KBE structure of the sentence
;;; **************************************************

(in-ruleset kbe-sent-top)

(input-node-label RS)

(output-node-label KBE)

(invoke-component (integrate-situation output-node-label))

(rule kbe-sent-top-rule1
       < RS subordination >
       :-
       < KBE > = (invoke-ruleset KBE-COMP-CL < RS >))

(rule kbe-sent-top-rule2
      (default)
      :-
      < KBE > = (invoke-ruleset KBE-CL < RS >))

;;; ***********************************
;;; KBE-COMP-CL RULESET
;;; **********************************

(in-ruleset kbe-comp-cl)

(input-node-label RS)

(output-node-label KBE)

(maintain-KBE-history KBE)

(rule kbe-comp-cl-r1
       < RS subordination type kind > is before
      :-
      < KBE > = (invoke-ruleset KBE-before < RS >))

(rule kbe-comp-cl-r2
      < RS subordination type > is progressive-adjunct
      :-
      < KBE > = (invoke-ruleset KBE-progressive-adjunct < RS >))

(rule kbe-comp-cl-r3
      < RS subordination type > is before
      :-
      < KBE > = (invoke-ruleset KBE-before < RS >))

(rule kbe-comp-cl-r4
     < RS subordination type > is then
     :-
     < KBE > = (invoke-ruleset KBE-then < RS >))

(rule kbe-comp-cl-r5
      < RS subordination type > is during
      :-
      < KBE > = (invoke-ruleset KBE-during < RS >))

(rule kbe-comp-cl-r6
      < RS subordination type > is until
      :-
      < KBE > = (invoke-ruleset KBE-until < RS >))

(rule kbe-comp-cl-r7
      < RS subordination type > is when
      :-
      < KBE > = (invoke-ruleset KBE-when < RS >))
```

Appendix C

The Frame Knowledge Base Language

In this appendix, we present the language used to represent the knowledge about the substances which physobjs and events consist. This language is specified in a BNF grammar using the same notation as that used in Appendix B.

C.1 The Object Substance Language

The BNF of the representation language for object substance is:

```
OBJ-SUBST     ::- (define-class SUBST-NAME
                                :superclass SUPERCLASS
                                :lexical LEXICAL
                                :part-of PART-OF
                                :attributes ATTRIBUTES)
SUBST-NAME    ::- ⟨a substance⟩
LEXICAL       ::- ⟨the possible lexical forms of a substance⟩
SUPERCLASS    ::- ⟨a list of substances⟩
PART-OF       ::- ⟨a list of substances⟩
ATTRIBUTES    ::- (ATTR ATTR-VALUE)+
ATTR          ::- ⟨an attribute of a substance⟩
ATTR-VALUE    ::- ⟨the possible values of the attribute⟩
```

To summarise, an object substance is defined as a frame with various slots or attributes. These substances are arranged into a is-a hierarchy using the SUPERCLASS slot and into a part-of hierarchy by a PART-OF slot. The various possible lexicalisations of the object substance are given via the LEXICAL slots. Note that two different object substances may have the same lexicalisation giving rise to lexical ambiguity. Other attributes of the object substance are stored in the ATTRIBUTES slot.

Examples

We give some examples of object substances defined in TASTE:

```
(define-class solid-matter
              :lexical (solid))

(define-class vegetable-matter
      :superclass solid-matter
      :lexical (vegetable))

(define-class carrot-matter
      :superclass vegetable-matter
      :lexical (carrot))

(define-class salt-matter
      :superclass seasoning-matter
      :lexical (salt)
      :attributes
      (source (one.of sea))
              (receptacle (one.of tablespoon-matter)))

(define-class skin-matter
      :name (skin)
      :superclass solid-matter
      :part-of (avocado-matter lemon-matter)
              :attributes
              (shape (one.of strip)))
```

C.2 The Action Language

The BNF of the representation language for action is:

```
ACTION         ::- (define-action ACTION-NAME
                                  :superclass SUPERCLASS
                                  :participants PARTICIPANTS
                                  :case-mapping CASE-MAPPING
                                  :constraints CONSTRAINTS
                                  :telic TELIC
                                  :preconditions PRECOND
                                  :delete DELETE-LIST
                                  :add ADD-LIST)
ACTION-NAME    ::- ⟨an action⟩
SUPERCLASS     ::- ⟨a list of substances⟩
PARTICIPANTS   ::- (ROLE₁ = VAR₁) ... (ROLEₙ = VARₙ)
ROLE           ::- agent | IN-ROLE | OUT-ROLE
IN-ROLE        ::- in : ROLE-NAME
```

OUT-ROLE	::- out : ROLE-NAME
ROLE-NAME	::- ⟨the name of a role⟩
VAR	::- ⟨a variable⟩
CASE-MAPPING	::- (verb (CASE → IN-ROLE))
CASE	::- ⟨a verb case⟩
CONSTRAINTS	::- PRED | PRED-EQ | (invoke-ruleset RULESET-NAME $\text{VAR}_1 \ldots \text{VAR}_n$)
PRED	::- As in section 2.2 of Appendix B
PRED-EQ	::- As in section 2.2 of Appendix B
TELIC	::- + | − | (invoke-ruleset RULESET-NAME $\text{VAR}_1 \ldots \text{VAR}_n$)
PRECONDITIONS	::- CLAUSE+ | (not-spec-value F-PATH F-VALUE)
ADD-LIST	::- CLAUSE+
DELETE-LIST	::- CLAUSE+
CLAUSE	::- (working-set-member VAR_i) | (spec-value F-PATH F-VALUE)
F-PATH	::- As in section 2.2 of Appendix B
F-VALUE	::- ⟨a path value⟩ | (invoke-ruleset RULESET-NAME $\text{VAR}_1 \ldots \text{VAR}_n$)

To summarise, an action is defined as a frame with slots. It is organised into a IS-A hierarchy via the SUPERCLASS slot.

Three particular slots are borrowed from the STRIPS representation of operators:

Preconditions: This specifies the state of the world which has to be true before the action can be executed.

Add-list: This specifies the propositions that are to be added to the world after the action is executed. Note that the content of the propositions to be added may be derived by invoking the appropriate ruleset. This is the case for actions like adding and mixing where the result of the action is determined by invoking rule reasoning.

Delete-list: This specifies the propositions that are to be removed from the world after the action indicated by the event substance is executed.

The constructs appearing in these three slots have the following meanings:

(working-set-member VAR_i**)** In the PRECONDITION slot, it tests whether the object which VAR_i is bound to is in the working set. In the ADD-LIST slot, it add the object bound to VAR_i to the working set. In the DELETE-LIST slot, it deletes the object bound to VAR_i from the working set.

(spec-value F-PATH F-VALUE**)** In the PRECONDITION slot, it tests that the value accessed by F-PATH is equal to F-VALUE. In the ADD-LIST slot, it adds F-PATH and F-VALUE to the feature structure. In the DELETE-LIST slot, it removes F-PATH and F-VALUE from the feature structure.

(not-spec-value F-PATH F-VALUE**)** This only appears in the PRECONDITION slot. It checks that the value accessed by F-PATH is not equal to F-VALUE.

The other slots in the frame representation of an action are as follows:

Participants This slot specifies the entities that are involved in the event. The specification includes the entity as well as the role they plays in the action. For example, in the sentence ***add the seasoning to the soup***, the object described by *the seasoning* plays the *addendum* role while that described by *the soup* plays the *base* role. The *agent* role is by default taken to be the hearer.

Constraints This holds the set of constraints which the objects involved in the event must satisfy. Recall that this is the set which is verified by the call to *action-constraints* (see Appendix B).

Telic This indicates the telicity of the event. The telicity can either be explicitly indicated or derived from reasoning by invoking appropriate ruleset.

Case-Mapping This contains the information for mapping from a verb case to a frame role.

Examples

We give some examples of actions defined in TASTE:

```
(define-action produce-no-change
              :superclass action)

(define-action produce-new-entity
              :superclass action)

(define-action placing-in
              :superclass produce-no-change
              :participants
              (in  : addendum = P1)
              (in  : base = P2)
              (out : result = P1)
              :case-mapping
              (placing-in (obj -> in : addendum)
                          (indobj -> in : base))
              :telic +)

(define-action adding
               :superclass produce-new-entity
               :participants
               (in : base = P1)
               (in : addendum = P2)
               (out : result = P3)
               :case-mapping
               (adding (obj -> in : addendum)
                       (indobj -> in : base))
```

```
            :telic +
            :delete
            (working-set-member P1)
            (working-set-member P2)
            :add
            (working-set-member P3)
    (spec-value < P3 >
                (invoke-ruleset determine-mixture P1 P2)))
```