January 1988

SEAFACT: Semantic Analysis for Animation of Cooking Tasks

Robin F. Karlin
University of Pennsylvania

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-88-04.

SEAFACT: Semantic Analysis for Animation of Cooking Tasks

Abstract

SEAFACT is a natural language interface to a computer-generated animation system. SEAFACT operates in the domain of cooking tasks. The domain is limited to a mini-world consisting of a small set of verbs which were chosen because they involve rather complex arm movements which will be interesting to animate. A linguistic analysis of the language found in recipes, included here, was used to define the domain. SEAFACT allows the user to specify tasks in this domain, using a small subset of English. The system then analyzes the English input and produces a representation of the task which can drive lower level motion synthesis procedures. The output of the system contains sufficient non-geometric information needed to schedule task start and end times, describe concurrent actions, and provide reach, grasp, and motion goals.

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-88-04.

SEAFACT: SEMANTIC ANALYSIS FOR ANIMATION OF COOKING TASKS

Robin F. Karlin

MS-CIS-88-04
GRAPHICS LAB 19

Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104

January 1988

Acknowledgements: This research is partially supported by Lockheed Engineering and Management Services, NASA Grant NAG-2-4026, NSF Grants MCS-82-19196-CER, IST-86-12984 and ARO grants DAA29-84-K-0061, DAA29-84-9-0027 including participation by the U.S. Army Human Engineering Laboatory.

UNIVERSITY OF PENNSYLVANIA
THE MOORE SCHOOL OF ELECTRICAL ENGINEERING
SCHOOL OF ENGINEERING AND APPLIED SCIENCE

SEAFACT: SEMANTIC ANALYSIS FOR THE ANIMATION OF COOKING TASKS

Robin F. Karlin

Philadelphia, Pennsylvania
January 1988

A thesis presented to the Faculty of Engineering and Applied Science of the University of Pennsylvania in partial fulfillment of the requirements for the degree of Master of Science in Engineering for graduate work in Computer and Information Science.

(Adviser)

(Graduate Group Chair)

Abstract

SEAFACT is a natural language interface to a computer-generated animation system. SEAFACT operates in the domain of cooking tasks. The domain is limited to a mini-world consisting of a small set of verbs which were chosen because they involve rather complex arm movements which will be interesting to animate. A linguistic analysis of the language found in recipes, included here, was used to define the domain. SEAFACT allows the user to specify tasks in this domain, using a small subset of English. The system then analyzes the English input and produces a representation of the task which can drive lower level motion synthesis procedures. The output of the system contains sufficient non-geometric information needed to schedule task start and end times, describe concurrent actions, and provide reach, grasp, and motion goals.

Acknowledgements

I would like to thank my advisor, Dr. Norman Badler, for suggesting this fascinating project and for providing me with guidance and many valuable ideas.

I would like to thank Dr. Bonnie Webber for her guidance and ideas on Chapter 3. I would also like to thank Dr. Mark Steedman for his help with Chapter 3. Other people who also deserve thanks for commenting on that chapter are Ethel Schuster, Ellen Hays, and Dr. Rebecca Passonneau. I also want to thank Dr. Martha Palmer for getting me started on my implementation.

Special thanks go to my husband, Steven M. Albert, for constant support and many of the commas in this thesis.

This research is partially supported by Lockheed Engineering and Management Services, NASA Grant NAG-2-4026, NSF CER Grant MCS-82-19196, NSF Grant IST-86-12984, and ARO Grant DAAG29-84-K-0061 including participation by the U.S. Army Human Engineering Laboratory.

Contents

Chapter 1

Introduction

SEAFACT is a natural language interface to a computer-generated animation system. SEAFACT operates in the domain of cooking tasks. The domain is limited to a mini-world consisting of a small set of verbs which were chosen because they involve rather complex arm movements which will be interesting to animate. A linguistic analysis of the language found in recipes, included here, was used to define the domain. SEAFACT allows the user to specify tasks in this domain, using a small subset of English. The system then analyzes the English input and produces a representation of the task which can drive lower level motion synthesis procedures.

This project is part of an ongoing effort at the University of Pennsylvania's Graphics Lab to develop a system for modeling and assessing human task performance. This system will be useful as a *task analysis* tool for assessing the *actions* of one or more individuals in a given environment. The ultimate goal is to have human behavior animated and measured over variable workplace geometry, figure anthropometry, and task description.

Producing an animation of natural human behavior requires a computational understanding of motion and its "semantics". A number of components are necessary for creating the animation. The lower level components include models of the workplace and the human figures. The motion is controlled by a combination of three methods: kinematics, dynamics, and constraints.

The higher level components include agent models, knowledge bases, task definitions, and task descriptions. Agent models contain information about agent capabilities and responsibilities. The knowledge bases store information which is shared by all the other system components. Types of information include the workplace geometry data, the anthropometric data base, the agent models, the task descriptions, and the object capabilities.

Task definitions are composed of primitives which represent tasks in a form that the system can interpret as lower level concepts such as goals, constraints, affected objects,

paths, and directions. Currently, task descriptions can be provided by commands in an artificial language (NASA Flight Data File format) or in a restricted natural language [Gangel84],[Badler86]. This capability is provided by the MVP system which analyzes the tasks and produces task definitions. The SEAFACT system is part of an effort to develop interactive systems with graphics and language interfaces in a variety of domains, which allow users to specify tasks to be modelled.

A natural language interface of this type requires three components: a syntactic analysis, a semantic analysis, and output generation in a form which can be interpreted by lower level components of the graphics modelling system. The output from SEAFACT will require further interpretation by an intelligent simulation system, currently being extended [Fishwick85], [Fishwick87]. This system will be responsible for a number of tasks including replacing object references with geometrical data, resolving relative time specifications, ordering the primitive commands according to those time specifications, and translating the primitive commands into the actual syntax of the motion control software. However, there is a carefully engineered relationship between the SEAFACT output and the motion control software. This relationship is based on the fact that the primitive concepts that comprise the output have been chosen because they are known to be semantically interpretable by the lower level software.

The development of this system is interesting not only because of its practical use as part of the human task performance system but also because of theoretical issues relating to the design of semantic representations. Several questions can be asked about a semantic representation: What needs to be included in it, and how can one verify that the representation is adequate?

Answers to these question depend on the tasks for which the representation will be used. An ambitious goal is to develop a semantic representation for a domain which is general enough to be useful for a variety of tasks. This goal is proposed for a representation of human movement in [Badler86b]. Badler suggests that the representation should be useful for the tasks of performance, observation, language description, and symbolic recording. On the other hand, a more easily attainable goal is to develop a semantic representation with one particular task in mind. [Palmer85] argues that the key to producing a successful semantic representation is in narrowing the task definition from determining the "meaning" of the domain, to determining what we need to know about the domain to accomplish some task. In both Badler's and Palmer's approach, the content of the semantic representation will be determined to a great extent by the task or tasks for which it will be used.

To return to the first question, the animation task is interesting because of what it forces one to include in the semantic representation. The representation must include geometrical information about the objects in the domain, including their spatial relationships to each

other. It must also include many details about movements which are not made explicit in a natural language description. Often natural language descriptions of actions only make explicit the goal of the action and not the method for achieving it. Default values for objects, locations, and durations are also necessary because those things can legitimately be omitted from task descriptions. Thus the representation must include a "script" for each potential action which will supply these default values. Badler argues that representations developed from visual information alone are inadequate and that "representations derived solely from language omit essential information needed to reconstruct an acceptable performance" [Badler86b]. A representation which is adequate for producing an animation must be more complex than representations aimed at simpler tasks.

With regard to the second question, how one verifies the adequacy of a semantic representation, animation provides useful insight. Presumably, if the representation were designed for any specific task then its adequacy could be judged by how well it enables the performance of that task. However, the task of animation is a particularly demanding and useful test of a representation's adequacy. Animations are not subject to interpretation in the way other end product tasks are. The grossly physical qualities of form and movement, for example, differ from the language used to express answers in a question-answer system. While a recognizable "semantics" or "culture" operates in both cases, animation requires specification of **all** implicit knowledge in the representation. In a single frame, all implicit knowledge must be explicit. In a question-answer system, on the other hand, what implicit knowledge is required only becomes clear as questions are asked, and therefore the adequacy of the representation is only tested to the degree that many different questions are asked. Furthermore, "incorrect" answers which seem to point toward an inadequate representation might instead be due to a difference between the way the questioner and the system interpreted the question.

A different issue that must be answered is why an intelligent, task oriented, natural language interface is valuable for an animation system. Such a system allows the user to specify a goal without having to specify a specific method for accomplishing that goal. Thus, the user can concentrate on getting results from a system, such as a human task performance modelling system. Since language is oriented toward expressing general goals rather than specific methods for their achievement, a system which can understand even a limited set of natural language inputs will allow the user much more freedom. In addition, language and graphical animation complement each other's weaknesses. Language excells at expressing "process", while graphical animation systems strain for adequate motion control systems [Badler87]. On the other hand, graphical modelling excells at expressing spatial relationships, an area in which language tends to be weak and ambiguous.

SEAFACT makes important contributions both in the area of developing a computational understanding of adverbials and other verbal modifiers, and in the area of developing a semantic representation which can support computer-generated animation. In the former area SEAFACT demonstrates that in order to represent verbal modifiers such as adverbs and prepositional phrases, one needs a representation of verbs which decomposes their actions into at least three subparts referred to as beginning-of-motion, principal-motion, and end-of-motion. SEAFACT also reveals the areas in which common sense knowledge and lexical knowledge are necessary to represent verbal modifiers.

SEAFACT builds a semantic representation of language input which supports the performance of the tasks specified by that input. This requires a detailed decomposition of verbs into primitives which are semantically interpretable by the animation software. It also requires that the representation include default information for all the knowledge which is not made explicit in the input, but must be explicit in the animated output.

Chapter 2 describes related work in the areas of converting natural language into visual representations and vice versa.

Chapter 3 is a detailed linguistic analysis of the language found in actual recipes. It is particularly concerned with categorizing the occurrences of verbal modifiers and developing a semantic analysis of them. This analysis is then used in the implementation. The linguistic analysis also serves as the basis for choosing which grammatical constructions to include in the system.

Chapter 4 is an overview of the system flow.

Chapter 5 is a sample session with the system.

Chapter 6 describes the content and organization of the knowledge bases used in the system.

Chapter 7 describes the implementation of the verbal modifiers.

Chapter 8 explains the form of the system's output.

Chapter 9 is a general conclusion and suggestions for future research.

Chapter 2

Background

Work has been done both in developing a natural language description of visual input and in developing a visual representation of natural language input. The former category includes [Badler75] and [Okada80, Okada87]. The latter category includes [Simmons75], [Takashima87], [Zeltzer85, Zeltzer87], [Gangel84] and [Badler86], and [Kalita87]. [Waltz81] and [Waltz82] are also concerned with representing the physical information in natural language input although they are not directly concerned with producing pictorial output.

2.1 Natural Language Description of Visual Input

[Badler75] presents a computational methodology for producing English-like descriptions of events in a visual scenario. Badler's methodology involves creation of a case structured semantic network for a subset of English motion verbs from pictorial and conceptual knowledge about objects in a natural environment.

Badler's case structure representation of events differs from previous work on case structures in that it is based on the data that is obtained from visual images and knowledge of objects rather than on linguistic data. He uses the term *event* to describe meaningful changes in the pictorial representation which may or may not be associated with a particular verb. The event representation also acts as a representation for individual verbs. Badler proposes a generic verb to describe motion called *changes-location-and/or-changes-orientation.* The event representation stores information about the changes of location and/or orientation of an object. Cases associated with an event are: subject, agent, instrument, reference, direction, trajectory, velocity, axis, angular-velocity, next, start-time, end-time, and repeat-path. This list includes cases which are not included in the linguistic notion of case. Badler demonstrates that his cases can characterize most of the cases suggested by [Miller72].

Badler proposes a process by which higher level concepts can be generated from changes in the pictorial data. This process consists of a hierarchy of demons. The lowest level

demons are concerned with object recognition. Going up the hierarchy, there are demons for motion description, directional terms, repeated events, description condensations, and finally motion verbs. Directional terms are represented by adverbials. English-like output is created from the event nodes.

[Okada87] continues work begun in [Okada80]. This work attempts to present a unified theory for understanding natural language and pictures. This theory is based on the definition of an abstraction process used to understand different levels of information. The lowest level or "raw data" [Okada87 p. 173] corresponds to what is perceived by the visual organs. "Cognitive features" are then extracted from the "raw data". These two areas are considered "image-like knowledge". Conceptual knowledge makes up the remaining levels of the hierarchy. These include "conceptual features", "simple concepts" and "inter-connected synthesized concepts". An important part of this work is an exhaustive analysis of Japanese verbs and adjectives.

Okada divides concepts into things, events, and attributes which are represented by nouns, verbs, and adjectives, respectively. An event designated by a verb is considered to be a "change from one state to another" [Okada87 p. 178]. He calls the objects accompanying events, "constituents" and defines the standard verb concept as the "dynamic relationships among constituents" [Okada87 p. 179]. These constituents include: subject, object, source, goal, opponent in mutual relation, reference, instrument or means, concept which supplements attributive aspects, location, time, and reason. Any of the first eight constituents can be obligatory, meaning that "they are indispensable for the recognition of events" [Okada87 p. 180]. The other constituents are considered optional. The "kernel structure" [Okada87 p. 181] of a verb concept is defined by the combination of obligatory constituents which it requires. These "kernel" concepts are similar to the usual case frames. Okada defines another component essential to the definition of a verb concept. This is the dynamic relationship or change described by the verb. Examples of his categories of change are movement, change in direction, and color change.

Okada describes SUNLAPP, a unified system for understanding natural language and picture pattern sequences. One component of this system, called SUPP, accepts line drawings as input and produces sentences in Japanese and English, describing events and attributes in the pictures.

Okada's work is an attempt to come up with a general cognitive model of picture and language understanding and to build a comprehensive knowledge base of concepts from Japanese words.

2.2 Visual Representations of Natural Language Input

Other work has been concerned with generating pictures or computer animation from natural language descriptions. [Simmons75] generates pictures from natural language input. [Takashima87] uses stories as input, while [Zeltzer85, Zeltzer87] is concerned with task descriptions as input. [Gangel84] and [Badler86] was the first work done at the University of Pennsylvania's Graphics Lab to provide natural language task oriented input intended for a human figure animation system. [Kalita87] also describes work being done at the University of Pennsylvania's Graphics Lab. His work is concerned with developing a semantic representation of verbs of state change which will be used to drive animation software.

[Simmons75] describes a system which generates two dimensional line drawings from English descriptions. Simmons advocates using a microworld because it allows one to "define a subset of English in great depth". [Simmons75 p. 18] His microworld consisted first of a clown, a pedastal, and a pole. He then expanded this to include sentences concerning movement by adding such things as water and a boat. Obviously, two dimensional line drawings are a much simpler problem then three dimensional animation. Simmons' system relates grammatical categories of words to a procedural semantics which drives the picture drawing. "Prepositions and verbs are defined as semantic functions that explicate spatial relations among noun images." [Simmons75 p. 18] Adjectives and adverbs provide information on the size and angles of support.

[Takashima87] describes SADS, the Story Driven Animation System. To produce animation from a natural language story, it is necessary not only to extract the action descriptions from the story, but also to find all the actions which are implied but not explicitly described. The latter task requires background or common sense knowledge. A discourse analysis of the story is necessary. A situation is expressed as a collection of assertions. Reasoning is used to update the representation of the current situation to include events which are not explicit in the story. For example, given the sentences, "The hare ran," and "The hare lay on the grass," the system interpolates the action "The hare stopped".

Another module of the system does stage directing. It produces a precise scenario of the location of the actors and objects and a division of the action into scenes.

The third module of the system generates the models and the motion descriptions. The motion description language is implemented in an object-oriented paradigm. The most primitive motion is to change the joint angle, and complex motions are defined as hierarchical combinations of primitive motions. An event driven scheduling mechanism assigns concrete time values to the actions.

[Zeltzer85] proposes a 3-D character animation system with three levels of control: guiding level, animator level, and task level. The guiding level describes the behaviors directly,

the animator level describes them algorithmically, and the task level describes them implicitly in terms of events and relationships. "Task level systems give us facile control over complex motions and tasks by trading off explicit control over the details of motion." [Zeltzer85 p. 249] A task level system requires a model of the physical and geometrical properties of the world as well as common sense information such as how one leaves a room. This information can be contained in a generalization hierarchy with multiple inheritance. At the task level, the system schedules the execution of motor programs which control the figure animation.

[Zeltzer87] further explores the problem of task level animation. He points out that what seem to be simple behaviors are really very complex. They only seem simple because they are guided by unconscious processes learned in early childhood. "A fundamental problem of task level animation, therefore, is to simulate the perception-driven behaviors of agents in an environment." [Zeltzer87 p. 1] To accomplish this goal, Zeltzer outlines a theory of motor problem solving, as opposed to the conscious act of cognitive problem solving. Zeltzer bases his approach on the finding "that natural movement systems have evolved into hierarchical control structures coordinating largely autonomous subsystems." [Zeltzer87 p. 2] The subsystems can be considered functional abstractions in that each one can create a class of motions under the control of a set of hierarchically organized "motor programs".

Zeltzer's implementation consists of a task manager at the top of the motor hierarchy. Input is in the form of task descriptions. The task manager must then extract from the task description a set of movement functions called skills, where each skill represents a class of motions a figure can perform, such as walking or grasping. The skills are then implemented by motor programs which are themselves implemented by more primitive motor programs called local motor programs. Motor problem solving consists of deriving a top level goal or goal set from the task description and then trying to satisfy the goal by forward chaining without backtracking through an expectation lattice of skills. This is a lattice structure in which each skill may be potentiated or depotentiated by other skills depending on the preconditions which must be satisfied in order to execute those skills. An example of such a precondition is that one must be standing in order to walk and therefore walking potentiates the skill for standing if the figure is not already upright. Once all preconditions are satisfied, a skill may be specified as a local motor program or as a sequence of other motor skills.

Zeltzer attempts to account for the way in which a movement is modified depending on the starting position of the actor and on other attributes of the current world. This requires the skill to take into account information about the current state of the world. Domain or situation specific information is encoded in a set of execution rules which are attached to a skill whenever the execution of a motion depends on domain information. These

execution rules define further patterns of potentiation or select other skills or parameters for the current skill. He gives the example of the skill to open a door. Execution rules are necessary for opening revolving doors, sliding doors, and locked doors. It seems that this treatment of the problem of variations in movements may not be adequate. One question is whether variations in a conceptual activity, such as opening a door, can really be treated as variations in a basic motor skill. The movements necessary for opening a standard door, a sliding door, and a revolving door are very different. It is really just the goal of passing through an opening via some mechanism that remains the same in each case.

[Gangel84] describes a natural language interface which accepts commands for operating instrument panels. The output of Gangel's system is a list of actions in a standardized script format which are intended for execution by a simulation system consisting of a production rule engine and a knowledge base [Fishwick85],[Fishwick87]. The system is based on the idea of case frames and Schank's Conceptual Dependency grammar [Schank75]. Case frames are used to represent the meanings of actions described by the input. They are lists of attributes, called cases or slots, whose values are determined by the action verb and the sentence in which it is found. Each verb has a case frame associated with it since the slots vary depending on the verb. Some of the slots are syntactically obligatory while others are optional. The action verbs used as input are reduced to primitive verbs as defined by [Schank75].

Gangel's system consists of several processing stages. First, the Bottom Up Parser (BUP) [Finin84] is used to do the syntactic parse and to provide a partial semantic representation of the input which Gangel calls an incomplete case frame. Next, the verb and its roles are analyzed. This analysis accomplishes several things. First, a verb may be used in different ways, such as transitively or intransitively, and this analysis differentiates those uses. Second, not all the necessary information is provided by the input sentence. An English command often specifies what is to be done but not how to accomplish the goal. The "how to" knowledge often depends on common sense knowledge about the world and particularly about the object of the verb since this often changes how the action is carried out. Therefore, the system includes a detailed object knowledge base containing information about all the objects. Information about the controls and indicators is particularly important for knowing how to carry out the action described by a non-specific verb such as *turn on*. This analysis is carried out by a Lisp function associated with each verb. This part of the system also uses information from the previous discourse, thereby allowing simple anaphoric references to be resolved. This analysis also matches the action verb with one of the primitives. Gangel actually implemented only two primitives, PMOVE and PLOOK.

[Kalita87] describes current work being done to develop a semantic representation of verbs of state change whose task is to drive animation software. He categorizes state change

verbs according to the attributes affected by the state change. From these categories he chooses to concentrate on verbs which affect physical system integrity, such as screw, bolt and nail. Kalita builds a knowledge base with physical information about the participants in the event described by the verb. He decomposes the verb *screw* into subparts using primitives such as *grasp* and *agent-move*. He uses a diagram like Waltz' event shape diagrams for the purpose of representing timing information. Kalita allows *when* conditionals to specify the endpoints of actions and proposes the use of demons to resolve these conditionals.

Kalita uses HIRES [Fishwick85], [Fishwick87] , a rule-based simulation system, to produce the final verb simulation. He specifies preconditions and postconditions for the verbs which test the availability and physical properties of the participants in the event.

[Waltz81] is concerned with representing physical information inherent in natural language descriptions of spatial relationships and physical events. He looks at the problem of whether an English sentence is plausible, that is, whether it could correspond to a real-world event. Waltz presents the following example: *My dachshund bit our mailman* [Waltz81 p. 4]. To judge plausibility he makes inferences from the semantic representation of the input which he develops. Because his semantic representation includes geometrical data it is similar to what is needed by a graphics system. Waltz points out that English verbs take on a wide variety of meanings which depend on their objects and instruments. He proposes that different representations will be necessary for each verb-object-instrument triple.

[Waltz82] proposes a more specific representation structure called an "event shape diagram" [Waltz82 p. 85]. These are diagrams which include a time line and various scales with values. They can be used to represent concurrent processes, causation, and other temporal relations. The diagrams may include different levels of detail, with the finest level consisting of primities such as *contact, surround, and support* [Waltz82 p. 85]. He proposes that this level could be used by a vision system.

Both papers are also concerned with representing adverbs, although neither one goes into great detail on the subject. This is discussed further in Chapter 8.

2.3 SEAFACT

SEAFACT performs a semantic analysis of natural language input which will be used to drive animation software. Its domain is the mini-world of simple cooking tasks which is potentially much larger than Simmons' micro-world. However, it makes no attempt to develop a general semantics of English, as Okada does for Japanese.

Zeltzer's work is a study of cognition and unconscious motor processes. SEAFACT only attempts to ascribe to its input the same meaning that a human being would. It does not claim that the method used to develop that meaning is a model of human cognition.

SYSTEM	DOMAIN	INPUT	OUTPUT	KNOWLEDGE REPRESENTATION
[Badler75]	traffic scenes, mechanisms	2-D pictorial of 3-D scenes	English-like	case structured semantic network
[Okada87],[Okada80]	large set of Japanese words	2-D pictorial	Japanese and English	case frames
[Simmons75]	clowns world	subset of English	2-D line drawings	procedural semantics
[Takashima87]	children's stories	Japanese	animation	object-oriented paradigm
[Zeltzer85],[Zeltzer87]	general	English task descriptions	final goal of animation	generalization hierarchy with multiple inheritance
[Gangel84]	operation of instrument panels	subset of English	reach and motion goals	case frame structure
SEAFACT	cooking tasks	subset of English	motion specifications to support animation	frame-based semantic network

Table 1: Summary of Related Work

Kalita's work is similar to SEAFACT but concentrates more on the verb decomposition and creation of a time sequenced simulation. SEAFACT is primarily concerned with accepting and representing a variety of verbal modifiers.

SEAFACT is similar to Gangel's work but attempts to expand or improve on it in the following ways. SEAFACT recognizes a larger variety of grammatical structures and of verbal modifiers. Its domain includes objects which can change location, while Gangel's work only included the motion of human figures. In order to accomplish this goal, SEAFACT's motion primitives are more varied and contain more geometrical information.

SEAFACT tries to avoid procedural encoding of domain specific information. However, its success at this endeavor is due mostly to the fact that its domain includes verbs whose meaning is less dependent on their objects than the verbs which Gangel chose.

Chapter 3

A Linguistic Analysis of Verbal Modifiers

3.1 Introduction

This chapter describes and categorizes the occurrences of verbal modifiers in the limited domain of cooking tasks and provides a semantic analysis of each of the categories. The data for this study consists of phrases and sentences from the recipes in nine different cookbooks. (Appendix A contains the complete set of data). Sentences were chosen on the basis of whether they contained verbal modifiers. The results of this study were used to choose a small subset of the cooking domain to implement. The semantic analysis of the verbal modifiers forms the basis of the implementation of a subset of those modifiers described in Chapter 7.

3.2 Syntactic Categories

Verbal modifiers can be realized by a number of syntactic categories, including adverbs, adverbial phrases, prepositional phrases, and participles.

3.3 Related Work

This section describes related work in a number of areas. [Passonneau86] describes a temporal analysis system which analyzes both tense and aspectual information. [Croft84] proposes a logical representation for adverbs and adjectives. [Huang75] is a grammatical analysis of adverbs and [Miller72] is a study of English verbs of motion.

[Passonneau86] describes the temporal analysis component of the PUNDIT text-processing system. This component derives a temporal analysis from the information in (1) the lexical

head of a predication, (2) the verbal categories of tense, grammatical aspect, and taxis, and (3) temporal connectives. The analysis consists of three parts: determining the "actual time," the "temporal structure," and the "temporal ordering" of the situation.

Passonneau shows that the information in her analysis makes possible the interpretation of temporal adverbials. Although PUNDIT only processes temporal connectives such as *when* and *before*, Passoneau briefly discusses the relationship between her temporal representation and other adverbials. She mentions three types of temporal adverbial modification: (1) adverbs which modify the manner in which situations change through time, for example, *slowly*, (2) adverbs which specify durations, for example, *for ten minutes*, and (3) adverbs which specify relations between times, for example, *now* and *before*. This paper, since it is based on a study of the relatively simplified language found in recipes, does not consider tense or grammatical aspect. Passonneau's work shows that to account for these factors requires a much more complex representation of the temporal components of language. However, she does not look at as many categories of temporal adverbials, nor does she propose a specific representation for them.

[Croft84] develops a logical representation for adverbs and adjectives (AA's). However, he does not discuss sentential adverbs referring to the time or location of an event, or verbal modifiers referring to instruments, sources, or goals. Most of the categories of adverbs in this study are not considered by Croft. However some of them fall into his category of Normal Measure adverbs which he considers one-place predicates. Verbal adverbs

> are analyzed as modifying an event variable, which can be thought of as a variable that describes an event or, more precisely, a process. [Croft84, p. 6]

An important part of his analysis is that the attributes of events such as result, direction, speed, etc., are considered higher-level types. Therefore, attribute names are not included in his logical representation because they are totally predictable from their values.

[Huang75] is a grammatical analysis of adverbs which includes a general classification of the semantics of adverbs within sentences. Huang identifies three ideas which are part of the notion of action. These are

> the state of mind of the actor, the carrying out or actual performance of the action, and the end result of action. [Huang75, p. 17]

The importance of this point is that utterances may refer to any of these parts of the concept of action. A primary difference between Huang's study and this one is that this one is concerned with a very specific and limited domain and body of utterances. Categories such as evaluative adverbs and state-of-mind adverbs are not considered in this study.

In his study of English verbs of motion [Miller72] identifies a number of semantic components that he uses to create "incomplete definitions," which express the similarities and

differences between verbs. His study analyzes the components of verbs in isolation. This study, in contrast, looks at the semantic roles of verbs and verbal modifiers used within a context. A few of Miller's components, such as instrumental and velocity, are similar to categories described in this study.

3.4 Semantic Categories

The verbal modifiers are classified according to semantic roles which are identified and defined in this study. These roles are similar to the attributes in [Croft84]. They include categories such as duration, speed, instrument, and force. The semantic roles discussed in this chapter are incorporated in the representation of sentences which is used in SEAFACT.

The following sections describe the categories of verbal modifiers found in the data for this study. Each section includes a discussion of the semantics of the category, and relevant examples from the data.

3.5 Aspect

This section discusses the notion of the temporal/aspectual types of sentences. The analysis of aspect given in [Moens87] is adopted as the basis for this analysis. Moens identifies temporal/aspectual types following Vendler but introduces new terminology. His types are divided into states and events. The events are classified as culminations, culminated processes, points, or processes. Moens claims that these types form part of the meaning of a sentence when that sentence is analyzed in its global context. Global context refers to the written as well as the real world context in which the sentence appears. Isolated verbs or sentences, on the other hand, may be placed in several of the temporal/aspectual types. This study examines sentences in isolation. However the global context of the sentences is clearly defined since all are taken from recipes in cookbooks. Although the exact context of each sentence is not included in the study, it can be inferred within the general context.

The majority of events in the cooking domain are culminated processes. A culminated process is

> a state of affairs that also extends in time but that *does* have a particular culmination associated with it at which a change of state takes place. [Moens87, p. 1]

It is intuitively obvious that cooking involves many processes. That each process must have a culmination follows from the fact that any cooking task involves a finite sequence of steps on the part of the cook. An important point about verbal modifiers in the cooking

domain, revealed in this study, is that many of them are concerned with characterizing the culmination points of processes. In many cases a verb and object alone do not specify a clear culmination point. For example, the command **beat the cream** does not contain information about the desired culmination of the process, that is, about when to stop the beating. Some sort of verbal modifier such as **for 10 minutes** or **just until it forms peaks** is necessary to specifiy the culmination of the process. These types of modifiers are analyzed in the section on **Duration of an Action.**

Another aspectual type is a culmination. A culmination is

> an event which the speaker views as accompanied by a transition to a new state of the world. This new state we will refer to as the "consequent state" of the event. [Moens87, p. 1]

A culmination is not extended in time as are processes and culminated processes.

The following are examples of culminations.

- **cover the pot**[1]
- **return the mixture to the pot**
- **discard the leaves**
- **break the egg**

Each of the actions in the examples above create consequences, which is part of the definition of a culmination. In addition, these examples are not extended in time. When considered at the level of commands in a recipe each of them is a punctual event. Notice that the global and world context influences the aspectual analysis. If one were to analyze **break the egg** in the context of developing an animation of the event then one would see this event as extending in time. The analysis in this study is done entirely at the level of understanding the language used in recipes.

3.6 The Number of Repetitions of the Action

Any expression which includes an endpoint, and therefore belongs to one of the aspectual classes of points, culminations or culminated processes, can be described as having a number of discrete repetitions. When a culminated process is described as having a number of repetitions, it is the entire process which is repeated. Process type events cannot have a number of repetitions associated with them since they do not include the notion of an end point. Consider the phrase **walk twice**. It can only be given a meaningful interpretation if

[1] All unreferenced examples are the author's.

one can view **walking** as a culminated process. For example, if the global context of **walk twice** were walking across a stage in order to model some fashions, then walking would be the culminated process of crossing from one end of the stage to another. This is a similar problem to Moens' example (6) **John ran in a few minutes** [Moens87, p. 3], taken from [Dowty79].

The sentences in this section all have end points associated with them. The number of repetitions of the event can be specified either as a cardinal number or as a frequency. It can also be specified indirectly as a result of the object of the verb being plural, having multiple parts, or being a mass term.

3.6.1 Cardinal Count Adverbials

Cardinal count adverbials [Mourelatos81, p. 205] specify an exact number of repetitions of the event.

(1) **baste** *twice* **during the cooking period** [Rombauer31, p. 350]

The following two examples, containing the verb **stir**, are particularly interesting and difficult to analyze.

(2) **stir** *once or twice* [Rombauer31, p. 349]

(3) **stir** *once* **and add collards** [Morash82, p. 131]

These examples are interesting because in the global context of cooking, sentences containing the verb stir are usually culminated processes where the culmination is described by some verbal modifier. For example, **stir until well mixed** describes a culminated process. Because **stir** is used with a cardinal count adverbial in Examples (2) and (3), and no other verbal modifiers are present, the meaning of the verb must include some inherent endpoint. When looked at from the level of understanding a recipe, it is not clear what endpoint can be associated with **stir**. When **stir** is used as a process, that process is not seen as having a substructure consisting of individual **stir** events. Rather, the process consists of a continual circular motion of a utensil inside a container. The motion does not have any inherent beginning or ending points.

There are two possibilities for the interpretation of examples (2) and (3). One can view **stir** as a culmination consisting of one revolution of the utensil around the container. However, this interpretation seems somewhat artificial. The other possibility is that examples (2) and (3) should be treated as idioms in which the meaning of the cardinal adverbials is actually less exact then usually implied by such adverbials. That is, **stir once or twice** might be taken to mean **stir for a short time**. This second interpretation is further supported by the fact that ***stir twice** or ***stir three times** does not seem to occur.

Notice that in the case of certain verbs or sentential contexts it is not possible to specify a number of repetitions for a culminated process. This is the case when the culmination involves a state change to the object which makes a repetition of the action impossible or meaningless. Consider the example

(3a) ***Freeze twice.**

Freeze is a culminated process and once the culmination has taken place the new state of the substance makes a repetition of the process redundant. The distinction between culminated processes which can and cannot be *immediately* repeated depends on world knowledge of the domain.

3.6.2 Frequency Adverbials

Frequency adverbials [Mourelatos81, p. 205] describe the number of repetitions of an action using a continuous scale with gradable terms [Croft84, p. 26] such as frequently, occasionally, and seldom.

(4) **Bring to a boil, reduce the heat, and simmer 20 minutes, stirring** *occasionally*, **until very thick.** [Poses85, p. 188]

The meaning of frequency adverbials is best captured by stating the length of the intervals between repetitions of the action. For example, the meaning of occasionally is that the number of minutes between incidents of stirring is large. An additional complication is that frequency adverbials must be interpreted relative to the total length of time during which the event may be repeated. If the total time period is longer then the intervals must be proportionately longer.

(5) **The pickles may be made up to 4 days in advance and kept covered and chilled, stirring** *occasionally.* [Gourmet86, p. 98]

In (5) the total length of time may be very long, if the pickles are made 4 days in advance, and so the intervals between stirring must also be long.

Frequency adverbials are interpreted relative to their global context in the same way as other gradable terms such as tall and short. The general domain of cooking and the specific recipe in question make up the global context in this domain. According to Croft, the meaning of gradable adverbs depends on a "reference set"

> denoting the class of individuals from which is derived the "average" value of the gradable property against which the AA in question is to be evaluated. [Croft84 p. 26]

In the case of frequency adverbials in the cooking domain the "reference set" consists of a large sample of repeated events, occurring during a set time period. An example is a set of repeated stirring events occuring during 20 minute intervals. This set is used to determine cardinal numbers for the average, low, and high values on the gradable scale. These cardinal numbers specify the length of an interval as a percentage of the total time period. The average, low, and high values are used, in turn, to specify cardinal numbers for each of the gradable terms.

Given any sentence with a frequency adverbial the following calculations may be made. The length of a single interval between incidents of the action is calculated by using the percentage value associated with the frequency adverbial to compute the length of time equal to that percentage of the total time period. IntervalTime = Percentage X TotalTime. The number of intervals present during the total time period is calculated by dividing the total time period by the sum of the length of one incident of the action and the length of a single interval.

A simplifying assumption is made here that the intervals between repetitions are equal. **Occasionally** might then mean intervals which are 25 per cent of the total time period and **frequently** might mean intervals which are 5 per cent of the total time period. This algorithm seems to coincide with the intuitive judgement that it is not normal to say **stir occasionally** during a very short time period such as 30 seconds. In such a case, the length of an individual stirring event might be longer than the total time. That is, for the domain in question there is some minimum interval between stirring events which is necessary for the term occasionally to be appropriate.

The object of the verb and the type of action may also influence the interpretation of the gradable terms. For example, given a command to **stir frequently** in the context of making something which burns very easily such as a cream sauce, frequently would be interpreted as a higher number of repetitions than it would in the context of making something less likely to burn.

3.6.3 Plural Objects

The use of plural objects or mass terms with a verb may or may not indicate that the action is to be repeated. Often this can be determined only by world knowledge.

(6) **chop the nuts**

World knowledge tells us that since nuts are small and relatively soft they can be chopped together in a group, perhaps using a cleaver.

(7) **chop the tomatoes with a knife**

World knowledge tells us that (7) usually requires a separate chopping event for each tomato. Notice that this is a case of repetition of a culminated process. This section includes events which are culminated processes and culminations. Verbal modifiers may also be used to make explicit whether an action is to be performed once on a group of objects together or separately on each object in the group.

(8) **drain dry and dip each piece** *separately* **in** [Rombauer31, p. 350]

(9) **spoon a bit on top of** *each custard* [Heatter65, p. 405]

(10) **beat in the eggs** *one at a time* [Gourmet86, p. 12]

(11) **puree the mixture** *in batches* [Gourmet86, p. 100]

(12) **add the remaining flour,** *1 tablespoon at a time*, **if necessary, until it forms a ball.** [Gourmet86, p. 80]

(13) **Stir once, and add collards,** *handful by handful*, **stirring constantly.** [Morash82, p. 131]

The events in (8) and (9) are culminations since the actions associated with **dip** and **spoon** do not extend in time and do bring about a new state of the world. The modifiers **separately** and **each** make explicit that these actions must be repeated for each object they are applied to. The events in (10) and (11) are culminated processes. In fact, the verbs **beat in** and **puree** both include inherent endpoints in their meanings. In a sentence without a verbal modifier these verbs would mean to perform the culminated process in question once on the objects indicated as in (14).

(14) **beat in 5 eggs until smooth**

However, in (10) the phrase **one at a time** makes explicit that there is to be a separate beating process for each egg. In (11), the process of pureeing is performed separately on batches of the mixture.

Depending on the sentential context, the event associated with **add** could be either a culmination or a culminated process. It cannot be a process in this domain since all substances are finite. In (12) and (13) the verbal modifiers indicate that the **adding** event consists of actions which are repeated until the culmination is reached. In these examples, **adding** is a non-repeated event which is a culminated process.

(15) **divide the pineapple** *among the buttered cups* [Heatter65, p. 407]

(16) **divide the batter** *between them* **(two buttered 9 inch round cake pans)** [Gourmet86, p. 12]

The event described by **divide** in the context of (15) and (16) is a culminated process which involves multiple objects.

3.7 The Duration of an Action

Parallel to the concept of the number of repetitions of a culmination or culminated process event is the concept of the duration of a process or culminated process event. Note that a repeated culminated process event means that the entire process is repeated while the duration of such a process refers to the amount of time it continues before the culmination of the process. Duration can be specified as a cardinal number or a gradable term, corresponding to those same categories for the number of repetitions. Duration can also be specified as co-extensive with the duration of another event.

Another important way to specify the duration of culminated processes is in terms of the change which signals the culmination. In the cooking domain this will be some type of perceptual change. This will be discussed in Section 3.8.

3.7.1 Explicit Duration in Time Units

Verbal modifiers may specify an explicit duration by giving a length of time. This can be less exact when a range of time or a minimum is specified.

(17) **stir** *for 1 minute;* **set aside.** [Morash82, p. 132]

(18) **Refrigerate and let marinate** *at least 15 minutes or up to 2 hours* [Poses85, p. 83]

(19) **bake in a preheated 350 oven** *for 30-60 minutes*, **depending on the age of the beets** [Morash82, p.24]

In (17) an explicit duration is given. In (18) and (19) a range of time is given.

3.7.2 Duration Given by Gradable Terms

The duration of an action can be specified by gradable terms on a continuous scale. The following examples show terms at the low end of the scale such as **briefly** and **slightly**. Terms from the other end of the scale, such as *for a long time*, do not seem to be common in recipes.

(19) **blend** *very briefly* [Robertson76, p. 316]

(20) **stir the yolks** *slightly* just to mix [Heatter65, p. 395]

3.7.3 Duration Co-extensive with the Duration of Another Action

In the cooking domain it is often necessary to do several actions simultaneously. For example, while heating a substance over the stove it may be necessary to stir it in order to prevent the substance from burning. In such cases it is most natural to express the duration of one of the activities in terms of the duration of the other one.

(21) **Continue to cook** *while gently folding in the cheeses* **with a spatula.** [Poses85, p. 186]

(22) **Reduce the heat to medium and fry the millet,** *stirring*, **for 5 minutes or until it is light golden.** [Sahni85, p. 283]

(23) **Add the beet greens and saute for 2-3 minutes,** *stirring*, **until they wilt and become tender.** [Morash82, p. 23]

3.8 Duration Characterized by a State Change

All processes which are carried out in the cooking domain must have culminations since cooking consists of a finite number of steps executed with limited resources. The language used to describe these processes can convey their culminations in different ways. In some cases a verb may contain inherent information about the endpoint of the action which it describes. In other cases verbal modifiers characterize the endpoint.

(24) **Chop the onion.**

Example (24) specifies a culminated process whose endpoint is defined by the state of the onion. While the desired final state of the onion could be specified more exactly by some adverb such as **finely** or **coarsely**, in the absence of such a modifier an endpoint can be established based on lexical knowledge about the state of an object which has been chopped.

In many cases, however, the meaning of the process verb does not include information on the endpoint of the process or the domain requires more specific information than what is conveyed by the verb alone. For example, the verb **beat**, in many contexts, does not supply the duration or the particular end result of the beating which would determine the duration. This is because different amounts of beating bring about different final states for many substances.

Therefore, the cooking domain includes many examples of duration of an action characterized by the specification of a state change in the object being acted on. State changes

can be perceived by any of the senses. The most common method of establishing state changes is through visual perception. However, touch, taste, and smell are also used.

There must be some test which verifies when a state change has ocurred. For visual changes the test consists of looking at the substance in question. A preparatory action is required only if the substance is not immediately visible, for example, if it is in the oven or in a closed pot. Changes which must be perceived by other senses, usually require additional actions. For example, to perform a tactile test one must touch the substance either directly or with some instrument.

This section is divided into subsections based on the type of change indicated and whether an active test (apart from looking at something) must be performed.

Phrases are classified as requiring an active test whenever the test is explicitly mentioned or world knowledge predicts that the test could not be performed passively. In those cases where the test could be active or passive depending on the circumstances, the phrase is classified in the non-active section.

Some of these phrases are actually a disjunction of an explicit duration and a state change. This section considers only the meaning of the duration specified in terms of the state change. The interpretation of disjunctions will be discussed below.

3.8.1 State Changes Perceived Visually - Without Active Test

The following examples specify duration by describing a state change which could be perceived visually without any additional action being taken. Notice that in most cases the syntactic construction describing the state change is a prepositional phrase with the preposition **until**.

(25) **steam 2 minutes or** *until mussels open* [Poses85, p. 83]

(26) **Reduce the heat to medium and fry the millet,stirring, for 5 minutes or** *until it is light golden* [Sahni85, p. 283]

(27) **Saute over high heat** *until moisture is evaporated* [Morash82, p. 131]

(28) **In a bowl sift together the flour, the baking powder, and the salt, add the butter, and blend the mixture** *until it resembles meal* [Gourmet86, p. 107]

(29) **heat in a 350 oven** *until the sauce is bubbly* **and beets are heated through.** [Morash82, p. 23]

(30) **cook for two minutes or** *until slightly colored.* [Morash82, p. 23]

(31) **saute in bacon fat** *until barely wilted and lightly colored.* [Morash82, p. 26]

(32) **Boil broth** *until reduced to 4 quarts* **to concentrate flavor.** [Morash82, p. 133]

(33) **Stir** *to coat* **with butter** [Morash82, p. 24]

(34) **"pan toast" in the butter on both sides** *until golden, watching carefully,* **as they brown very quickly.** [Poses85, p. 189]

(35) **beat the mixture** *until it is combined well (it will look curdled)* [Gourmet86, p. 107]

Notice that example (33) does not contain a prepositional phrase with **until.** However it could easily be rephrased as **Stir until coated with butter.**

Examples (34) and (35) both emphasize the visual test to be performed. (34) states that the visual test must be performed carefully in order not to miss the ideal endpoint of the process.

3.8.2 State Changes Perceived Visually - With Active Tests

The following are examples of state changes which can be perceived visually but require additional actions in order to perform the visual test.

(36) **cook for 3-5 minutes or** *until the scallops are uniformly white (test by cutting into one)* [Poses85, p. 82]

(37) **Add the lemon juice and water, cover, and cook about 15 minutes or until the liquid is absorbed.** [Poses85, p. 188]

(38) **Cook the mixture over moderately low heat, stirring,** *until the curd is thick enough to coat the back of a wooden spoon,* **but do not let it boil.** [Gourmet86, p. 110]

In example (36) the active test is stated explicitly. In order to see if the scallops are uniformly white it is necessary to cut into one. In example (37) the active test must be inferred based on world knowledge. A substance is to be cooked in a covered pot. In order to see if the liquid has been absorbed, it is necessary first to uncover the pot. Example (38) states the active test fairly explicitly. The mixture is to be cooked until it is thick enough to coat the back of a wooden spoon. From this information it is not hard to arrive at the test of dipping a wooden spoon into the curd and then pulling it out. However this test is not as explicit as the one in (36).

3.8.3 State Changes Perceived by Touch - Require Active Test

The following examples require active tests involving touching to determine when the specified state change has occurred. Determining what type of test is required is dependent on detailed world knowledge about the domain.

(39) **Bring to a boil, reduce the heat, and simmer 20 minutes, stirring occasionally,** *until very thick.* [Poses85, p. 188]

In (39) the test consists of stirring. The thickness of the substance can be perceived by the sense of touch through the instrument used to stir.

(40) **Pour boiling water over the lentils to cover by an inch and let soak 15-30 minutes or** *until tender.* [Poses85, p. 188]

In (40) the test consists of stabbing some lentils with a fork. The sense of touch will reveal how much force is necessary to pierce the lentils. When the amount of force necessary is small, then the lentils are tender.

(41) **heat in a 350 oven until the sauce is bubbly and** *beets are heated through* [Morash82, p. 23]

In (41) the test would be to cut open one of the beets and touch it with your finger.

(42) **Add the beet greens and saute for 2-3 minutes, stirring, until they wilt and** *become tender.* [Morash82, p. 23]

In (42) the test would be to see if the beet greens can be cut easily.

3.8.4 State Change Perceived by Taste - Without Active Test

The title of this section is an apparent anomaly since it is not possible to taste something without performing an action. In fact the title reflects the conventional meaning associated with the phrase **according to taste**. The meaning of this phrase is to prepare the substance in question according to one's personal preference. In many cases the cook may already know the duration of the action necessary to achieve the preferred taste. Thus it is not necessary to taste the substance each time the action is repeated.

(43) **Crack the eggs into the simmering water and poach them** *to taste* **(2-4 minutes).** [Poses85, p. 190]

In (43) world knowledge tells us that actual tasting is not necessary. This is due to the fact that eggs always cook in a standard amount of time.

3.8.5 Disjunctions of Explicit Durations and State Changes

Many examples from the previous sections (e.g. (25), (26),(30),(36),(37), and (40)) consist of a disjunction of an explicit duration and a state change. Example (25) is repeated here.

(25) **steam 2 minutes or** *until mussels open* [Poses85, p. 83]

The meaning of these sentences is not the same as that of logical disjunction. Example (25) does not give the cook a choice between steaming for 2 minutes or until the mussels open. The actual meaning of these disjunctions is that the state change is to be used to determine the duration of the action. The explicit duration provides information on the usual amount of time that is needed for the state change to take place. In example (25) this information would be useful so that the cook knows that after starting to steam the mussels he does not have long to wait before he must check to see if they have opened. This particular test does not involve a complicated activity. However, in other cases, knowledge of the approximate time when a test should be applied can be crucial to the success of a recipe.

(44) **Bake for an hour or until the loaf sounds hollow**[2]

The baking time depends on the loaf of bread sounding hollow. To determine when this change has occurred one must remove the bread from the oven. To do this too often would result in damage to the baking process. Therefore, the approximate time at which to check the bread is very important. The disjunction **for an hour or...** provides this information.

[Ball85] discusses problems that arise in the semantic interpretation of what she calls metalinguistic or non-truth functional disjunction. For example, the previous sentence illustrates metalinguistic disjunction. "The first clause is asserted, and the right disjunct provides an alternate, more accessible description of the referent of the left disjunct." [Ball85, p. 3] The truth of these sentences depends on the truth of the first disjunct. Ball claims that if the first disjunct is true and the second is not then the sentence is still true although "our impression will be that something has gone wrong." [Ball85, p. 3]

The disjunctions of explicit durations and state changes seem to be another type of metalinguistic disjunction. They are very similar to the examples given by Ball except that it is the right disjunct which determines the truth of the sentence and the left disjunct which provides an alternate description. Furthermore, this alternate does not have to be strictly synonymous with the right disjunct. The semantics of these disjunctions includes the notion that the left disjunct is only an approximation.

[2]Thanks to Bonnie Webber for this example and the following interpretation.

3.9 Time Relations Between Actions

A variety of syntactic constructions, including adverbs and prepositional phrases, express the temporal relationship between the action in a command and the actions described in earlier or later commands.

(45) *immediately* **pour the carmelized sugar** [Heatter65, p. 400]

(46) **stir them gently 3 to 4 times** *in the next 5 to 10 seconds* [Jaffrey81, p. 196]

(47) *At the last minute,* **stir in the scallions.** *Serve at once* **accompanied by the sauce and rice.** [Poses85, p. 188]

(48) **Pour in about 3 tablespoons of batter and** *quickly* **swirl to coat the bottom evenly.** [Poses85, p. 191]

(49) **Peel** *as soon as they can be handled* **and** *while still warm* **cut them into 1/2-inch cubes.** [Morash82, p. 25]

In (45) the adverb **immediately** means that the pour action is to take place at the first opportunity after the previous action has been completed. (46) mentions a definite time, 5 to 10 seconds after the current time, in which to stir. In (47) the phrase **at the last minute** refers to the moment before serving the food. This fact must be part of the world knowledge about the domain. In (48), **quickly** dictates that the swirling must immediately follow the previous action of pouring the batter. (49) differs from the previous examples in that the other event referred to is actually the state of the substance with regard to its temperature.

When-clauses are also found in recipes. [Moens87] describes the semantics of **when**-clauses as introducing a *novel temporal referent into focus* [Moens87, p. 4]. In this article Moens discusses **when**-clauses which are events. He proposes a structure for all events which consists of a nucleus.

> A nucleus is defined as a structure comprising a culmination, an associated preparatory process, and a consequent state. [Moens87, pp. 3-4]

An event can either be decomposed into the parts which make up a nucleus, or the entire event can be considered as the culmination of another nucleus. The temporal referent introduced by a **when**-clause consists of the entire nucleus, and the main clause event can refer to any part of that nucleus.

In most of the following examples, the **when**-clause describes a state. In the spirit of Moens' analysis, such **when**-clauses could be considered as a consequent state and could be

composed into a nucleus with the event that brought about that state. In all of the examples considered, the event of the main clause refers to the time of the consequent state and not to the preparatory process or the culmination. The action of the main clause is supposed to take place after the state described in the **when**-clause has been established. This same meaning is more commonly expressed by a sentence describing an action which is to be performed **until** the desired state is established. The next action, described in the following sentence without any temporal modifiers, is understood to occur after the completion of the first action. Example (50) could also be expressed as **While you preheat the oven to 400 degrees, chill the (pie) shells for about 15 minutes or** *until the shells are cold and firm.* **Line the shells with waxed paper or foil and fill them with raw rice or beans.**

(50) **Chill the (pie) shells for about 15 minutes while you preheat the oven to 400 degrees.** *When the shells are cold and firm, line them with waxed paper or foil and fill them with raw rice or beans.* [Thomas78, p. 342]

When-clauses are commonly used when some other action intervenes between the action which will establish the state described in the **when**-clause and the action of the main clause. In such cases the **when**-clause is necessary to bring the previously mentioned temporal referent into focus. (51) is an example of this.

(51) **In the large bowl of an electric mixer, at high speed beat the eggs, yolks, salt and sugar** *for about 15 minutes* **until mixture is as thick as soft whipped cream.** *Meanwhile... When egg mixture has been sufficiently beaten, add vanilla and almond extracts, and cognac or rum.* [Heatter65, p. 430-431]

In example (52) the **when**-clause is a culminated process type of event. The culmination is the point where the eggs reach the top of the bowl. This example is like the examples in which the **when**-clauses are states, in that the event of the main clause refers to the state which is consequent to the culmination of the event in the **when**-clause.

(52) **In the small bowl of an electric mixer beat the eggs and sugar at high speed for a few minutes.** *When eggs expand to reach top of bowl, transfer to large bowl of mixer and continue to beat for 5 minutes more* [Heatter65, p. 424]

3.10 The Quantity of the Object

This category of modifiers is somewhat problematic. Most of the examples could be placed under the categories of **Duration Given by Gradable Terms** or **Force**. However, a

separate category seems justified because the intent of each of these modifiers is to modify the quantity of the object of the verb. Although modifying the duration or force of an action can result in modifying the quantity of the object of that action, it does not necessarily do so. Also, there are other instances in which these same modifiers do not affect the quantity of the object. See example (87).

(53) **rub** *generously* **with clarified butter** [Rombauer31, p. 350]

(54) **brush the mixture** *lightly* **onto the rough sides of the pita triangles** [Gourmet86, p. 82]

(55) **Sift the confectioners sugar** *lightly* **over the torte.** [Gourmet86, p. 82]

(56) **sprinkle with lemon juice** *to taste* [Morash82, p. 24]

(57) **Add salt, pepper, and sage** *to taste* [Poses85, p. 190]

The meaning of the adverb **generously** in (53) does not affect the actual method of rubbing. Rather it modifies the amount of butter to be used in the rubbing. In (54) **lightly** could refer to the force with which the brushing is done. However, since the pita triangles are not fragile, it is clear that the real intention of the modifier is to modify the amount of the mixture which is brushed. Use of light force alone might not achieve this if, for instance, the brush was first soaked in the mixture. That is, it might also be necessary to modify the amount of the mixture placed on the brush. (55) is similar in that depending on the type of sifter in use, force may not affect the amount of sugar which is sifted. Yet, the intention of the modifier is clearly that a small amount of the sugar is desired. Notice that these are also gradable terms which must be interpreted relative to their context.

The last two examples use the phrase **to taste**. In these cases it might be appropriate to actually taste the substance. The intent is to modify the amount of the substance added, according to the personal desire of the cook.

3.11 The End Result

This category consists of verbal modifiers which characterize the desired end result of an action. It differs from the state change category in that the desired end results in that category depend on the *duration* of an action. This category includes those end results which depend on any factors other than duration.

(58) **float a snow-pea half** *prettily* **over the top** [Jaffrey81, p. 194]

(59) **Cook the sausage, breaking it up** *as finely as possible.* [Poses85, p. 190]

(60) **slice them** *thin* [Gourmet86, p. 10]

(61) **let them cool** *completely* [Gourmet86, p. 12]

(62) **cut the cucumbers lengthwise** *into 1/8-inch-thick "noodles"* [Gourmet86, p. 68]

(63) **slice** *into shreds or diagonal slices* [Morash82, p. 131]

(64) **Dice all the vegetables** *the same size* **for best appearance.** [Morash82, p. 25]

In each of these examples the verbal modifier describes the desired end result of the action but gives no information about how to achieve that end result. That information has to come from world knowledge about the domain.

3.12 The Location of the Action on the Object

In these examples the verbal modifiers describe the location on the object at which the action is to take place.

(65) **slash it** *in several places* [Rombauer31, p. 350]

(66) **cut** *from head to tail* [Rombauer31, p. 351]

(67) **press** *onto top of apples* [Robertson76, p. 307]

(68) **Combine the Parmesan and melted butter and distribute** *over the eggs* [Poses85, p. 190]

(69) **split the underside of the tail shells** *lengthwise* [Gourmet86, p. 82]

(70) **Cut stems and leaves** *1 inch above beet crowns*, **and put leaves aside.** [Morash82, p. 23]

3.13 Instrument

In these examples the verbal modifiers introduce the instrument to be used in the action.

(71) **cut in the margerine** *with a pastry cutter or two knives* [Robertson76, p. 313]

(72) **place** *by tablespoon* [Robertson76, p. 314]

(73) **Cook over medium heat while folding the mixture** *with a spatula* **to blend in the cream cheese.** [Poses85, p. 186]

(74) *using a long handled fork* [Gourmet86, p. 10]

(75) *with a slotted spatula* [Gourmet86, p. 12]

3.14 The Speed

The verbal modifiers in this section characterize the speed of the action.

(76) **stir in buttermilk** *with swift strokes* [Robertson76, p. 313]

(77) *quickly* **tilt and turn the dish** [Heatter65, p. 400]

(78) *very gradually* **pour** [Heatter65, p. 393]

(79) **beat in the remaining 2 tablespoons sugar** *gradually* [Gourmet86, p. 110]

Example (79) differs from the other examples in that the modifier **gradually** actually modifies the speed at which the sugar is added rather than the speed of the beating. World knowledge of the domain has to supply this fact.

3.15 Force

The verbal modifiers in this section characterize the force with which the action is executed.

(80) **pour** *gently* [Rombauer31, p. 351]

(81) *gently* **heat** [Robertson76, p. 314]

(82) **Continue to cook while** *gently* **folding in the cheeses with a spatula.** [Poses85, p. 186]

(83) **without being too thorough,** *gently* **fold** [Heatter65, p. 419]

(84) **turn them around** *gently* [Jaffrey81, p. 193]

(85) **toss the compote** *gently* [Gourmet86, p. 66]

(86) **cook it, swirling the skillet** *gently* [Gourmet86, p. 132]

(87) **Prick the shells (pastry)** *lightly* **with a fork.** [Gourmet86, p. 110]

(88) *gently* **squeeze to remove water** [Morash82, p. 131]

The modifiers in examples (82)-(88) characterize the physical force to be applied either directly by the cook's hand or via an instrument. In example (80) the affect of force on the action is less direct. In order to pour gently one applies less force to tilting the container. In (81) the force involved is the heat itself. To heat gently means to apply less heat.

3.16 Degree

The verbal modifiers in this section actually modify other semantic roles of the action to be performed. **Well** is another gradable term, but it is difficult to define what the scale is intended to measure. Clearly, in this domain, the opposite end of the scale is not **poorly.** It might be something like **slightly**. But **slightly** seems most closely associated with the duration of the action. However, to do something well in this domain may involve factors other than the duration of the action, such as the force or the speed. To do something well means to increase whichever factors contribute to accomplishing the task. For example, if force is needed to accompish a task, such as mixing some substances, and if more force would do the job better, then mix well indicates that the force should be increased.

(89) **Beat egg, add ground meat, and mix** *well* **with fork.** [Morash82, p. 24]

(90) **In a bowl stir together** *well* **the sugar and the cinnamon.** [Gourmet86, p. 110]

(91) **drain and dry** *thoroughly* [Rombauer31, p. 350]

(92) **stirring** *well* [Robertson76, p. 308]

3.17 Purpose

The verbal modifiers in this section supply a purpose or justification for the action. If the cook understands the purpose behind an action then he is more likely to carry out the action in a way which will ensure its success. This is especially important in those cases where the purpose is not obvious.

(93) **Cook over medium heat while folding the mixture with a spatula** *to blend in the cream cheese* [Poses85, p. 186]

(94) **"pan toast" in the butter on both sides until golden, watching carefully,** *as they brown very quickly.* [Poses85, p. 189]

(95) **Pour in about 3 tablespoons of batter and quickly swirl** *to coat the bottom evenly.* [Poses85, p. 191]

(96) **Prick and blanch for 5 minutes** *to release fat* [Morash82, p. 132]

(97) **Boil broth until reduced to 4 quarts** *to concentrate flavor* [Morash82, p. 133]

(98) **Dice all the vegetables the same size** *for best appearance.* [Morash82, p. 25]

(99) **gently squeeze** *to remove water* [Morash82, p. 131]

For example, in (98) it might not seem important to the cook to make sure that all the diced vegetables are the same size. If the cook is told directly that the motivation for this command is the good appearance of the vegetables then he can decide whether that motivation is sufficient to warrant the extra trouble of dicing them all the same size.

3.18 Conclusion

This analysis has identified categories of verbal modifiers which are found frequently in recipes. While all of these categories are found in other domains as well, some of them are particularly prevalant in this domain because the purpose of recipes is to describe procedures. Categories such as instrument, speed, force, and location of the action on the object would commonly be found in many domains. On the other hand, the temporal category which characterizes the duration of an action by a state change is particularly common in recipes for two reasons. First, the physical process of cooking always involves state changes to objects and second, the meaning of many verbs used to describe cooking processes does not include information about the state change which should trigger the culmination of the process. Therefore, verbal modifiers are necessary to make the desired state changes explicit.

This analysis has also shown a relationship between aspectual categories of events and the modifiers which may co-occur with them. For example, the categories of modifiers which express the number of repetitions of an action can only modify expressions which include an endpoint, that is, points, culminations, or culminated processes.

The analysis of the verbal modifier categories reveals many areas where common sense knowledge or physical knowledge about the world is required to represent the semantics of these categories. For example, when an action is performed on a plural object, physical knowledge about the size and consistency of the objects and about the action itself is necessary to tell us whether it must be repeated for each of the objects separately or performed on all the objects in a group.

Each of the categories identified in this analysis can be considered a semantic role in the representations of the verb or the event associated with the natural language expression. The next chapters describe how some of these semantic roles have been implemented in the SEAFACT system using the semantic analyses presented in this chapter. The choice of which roles to implement was influenced by several factors. First, there was the need to limit the amount of world knowledge needed by the system, and second there was a desire to choose roles which would add to the interest of an animation of the natural language expression.

Chapter 4

A General Overview of SEAFACT

This chapter provides a general description of the components of the SEAFACT system and the flow of control. Figure 1 is a flow chart of the SEAFACT system.

The system accepts as input a restricted set of English from the domain of cooking tasks. Chapter 5 contains examples of this input. All English input must be in the form of commands since the purpose of the input is to specify cooking tasks to be animated.

Parsing and the production of an intermediate semantic analysis of the sentence is done by BUP, A Bottom Up Parser [Finin84]. BUP is a bottom up syntactic analyzer which accepts a grammar and lexicon in the form of rules. It uses these rules to do either a syntactic parse or a transduction of the input. The grammar can be either context-free or an extended phrase structure grammar. The SEAFACT grammar is an extended phrase structure grammar. Each rule consists of a left hand side and a right hand side which is the standard form for a context free grammar. In addition, many rules specify tests which must evaluate to true in order for the rule to apply. These tests give the grammar the power of an extended phrase structure grammar.

The following is a rule from the grammar:

```
(ADVP (ADV) '(SPEED (,#>1)) -if (not (null (eval '($ isa ,#>1 speed-term)))))
```

The meaning of this rule is that an ADVP can be rewritten as an ADV, where ADV is the non-terminal symbol which rewrites to various items of the lexicon which are adverbs and ADVP is a non-terminal symbol standing for adverbial phrase. However, in order for the rule to apply, the test, which follows the *if*, must succeed. The third component of the rule,

```
'(SPEED (,#>1))
```

, specifies the intermediate semantic structure which the rule is to produce. This means that the rule is being used as a transducer. The test says that the rule should apply only if the term ADV is classified in the Linguistic Term KB as a speed-term. The knowledge bases

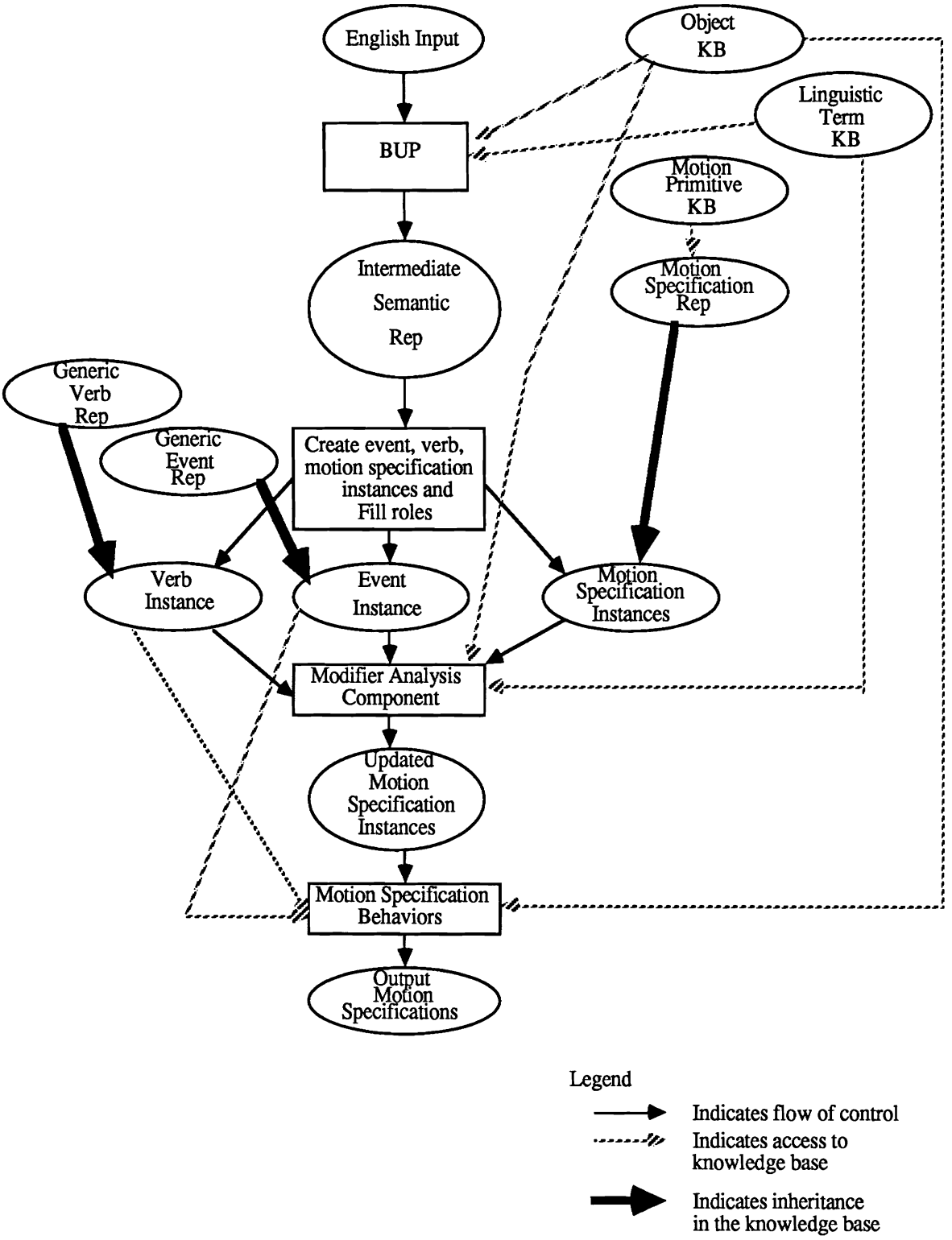

Figure 1: **System Flow Chart**

used in SEAFACT are described in Chapter 6. If the test succeeds, then the rule produces the appropriate intermediate semantic representation for an adverb pertaining to speed. The tests attached to rules in the grammar access both the Object KB and the Linguistic Term KB. Rules which access the Object KB check the semantic types of objects. In this way the rules enforce selectional restrictions on the objects. The grammar for SEAFACT consists of 109 rules.

BUP produces an intermediate semantic structure from the English input. This structure specifies semantic categories for each of the verbal modifiers. It identifies the verb in the sentence but performs no semantic analysis of it. The semantic structure is described further in Chapter 7.

SEAFACT uses the information in the intermediate semantic structure to create verb and event instances for the input. These instances are concepts in the Verb KB and the Event KB. These concepts include roles whose values are filled by the information in the intermediate semantic structure. For example, the event instance has a speed role which would be filled by a speed term that was present in the English input. The system takes all the information in the intermediate semantic structure and uses it to fill in the appropriate roles in the verb and event instances. The system also creates a motion specification instance for each motion section in the motion specification of the verb. These concepts are added to the Motion Specification KB. They contain roles which specify the duration of each of the primitive actions which specify the motion of the verb.

Some of the verbal modifiers require additional processing. These are the verbal modifier categories of speed, repetitions, and duration. The Modifier Analysis Component of the system checks the event instance for the presence of any of these modifiers and carries out any additional processing. This is described fully in Chapter 7. The result of this processesing is updated values for the time specifications of the event. These are placed in the motion specification instance.

Finally, the system invokes the Motion Specification Behaviors which produce the final motion specifications. These behaviors are functions which access the verb, event, and motion specification instances in order to fill in the values in the primitive motions. This process is described fully in Chapter 8.

SEAFACT is missing several components which are necessary to produce a complete specification for the computer-generated animation. First it must have a State-of-the-World KB which contains information about the complete state of the world. This KB must be updated as commands are executed. The system also needs to have pre and post conditions attached to the verb representations. The preconditions would specify what must be true in the state of the world for the action described by the verb to be carried out. They would also invoke other actions which would bring about a state of the world in which the

current action could be carried out. The postconditions would alter the state of the world after an action is carried out. To implement the preconditions and postconditions a rule based system like HIRES is necessary. The work done by Kalita could be used to add these components to SEAFACT.

Chapter 5

Sample Session of SEAFACT System

This chapter contains a sample session with SEAFACT. Comments in **bold face** have been added to explain the examples. Otherwise the examples are just as the system produced them with one exception. Where long sequences of primitive motions were repeated, some of the repetitions have been removed. The missing repetitions are indicated by 3 vertical dots. The system can accept sequences of input tasks of arbitrary length. The clock begins at 0 and is incremented by each command. The first sequence of examples contains 5 commands. Subsequent sequences each contain one input command. This was done so that the start times would not get unwieldy.

This system was implemented in Common Lisp on a VAX-11/785. The average cpu time per query for this sample session was 10.05 seconds.

```
DCRL> (top-level)
```

This is an example of a verb with a lexical aspect of culmination. This verb does not have a beginning-of-motion section.

```
Enter your command
>>> cover the pot
BEGINNING-OF-MOTION

PRINCIPAL-MOTION
(GRASP1 PREFERRED_ARM (HANDLE PARTOF LID) (START (S 0) DURATION (S 1) END NIL))
(HOLD2 PREFERRED_ARM (HANDLE PARTOF LID) (START (S 1) DURATION NIL END (UNTIL CANCEL HOLD2)))
(MOVE3 PREFERRED_ARM LID (TOP OF (POT)) (START (S 1) DURATION (S 2) END NIL))
(ALIGN4 (VERTICAL-AXIS OF LID) (VERTICAL-AXIS OF (POT)) (START (S 1) DURATION (S 2) END NIL))
(CANCEL5 HOLD2 (START (S 3) DURATION (S 0) END NIL))
(RELEASE6 PREFERRED_ARM (HANDLE PARTOF LID) (START (S 3) DURATION (S 1) END NIL))

END-OF-MOTION
```

```
(MOVE-TO7 PREFERRED_ARM REST-POSITION (START (S 4) DURATION (S 1) END NIL))
```

This is an example of a speed modifier with a culmination.

```
>>> cover the pot quickly
BEGINNING-OF-MOTION

PRINCIPAL-MOTION
(GRASP8 PREFERRED_ARM (HANDLE PARTOF LID) (START (S 5) DURATION (S 0.5) END NIL))
(HOLD9 PREFERRED_ARM (HANDLE PARTOF LID) (START (S 5.5) DURATION NIL END (UNTIL CANCEL HOLD9)))
(MOVE10 PREFERRED_ARM LID (TOP OF (POT)) (START (S 5.5) DURATION (S 1.0) END NIL))
(ALIGN11 (VERTICAL-AXIS OF LID) (VERTICAL-AXIS OF (POT)) (START (S 5.5) DURATION (S 1.0) END
NIL))
(CANCEL12 HOLD9 (START (S 6.5) DURATION (S 0.0) END NIL))
(RELEASE13 PREFERRED_ARM (HANDLE PARTOF LID) (START (S 6.5) DURATION (S 0.5) END NIL))

END-OF-MOTION
(MOVE-TO14 PREFERRED_ARM REST-POSITION (START (S 7.0) DURATION (S 0.5) END NIL))
```

This is an example of an instrument modifier and a duration modifier

```
>>> stir the batter with a wisk for 2 minutes
BEGINNING-OF-MOTION
(GRASP15 PREFERRED_ARM (HANDLE-END PARTOF (WISK)) (START (S 7.5) DURATION (S 1) END NIL))
(HOLD16 PREFERRED_ARM (HANDLE-END PARTOF (WISK)) (START (S 8.5) DURATION NIL END (UNTIL CANCEL
HOLD16)))
(MOVE17 PREFERRED_ARM (STIR-END PARTOF (WISK)) (CENTER OF BOWL) (START (S 8.5) DURATION (S 2)
END NIL))
(ALIGN18 (LONGITUDINAL-AXIS OF (WISK)) (CENTRAL-AXIS OF BOWL) (START (S 8.5) DURATION (S 2) END
NIL))

PRINCIPAL-MOTION
(CONSTRAINT19 (STIR-END PARTOF (WISK)) (INSIDE OF BOWL) (START (S 10.5) DURATION (S 120) END
NIL))
(ROTATE20 PREFERRED_ARM (WISK) (CENTRAL-AXIS OF BOWL) (- (RADIUS OF CONTAINER) SMALL-AMOUNT)
(REPS 40) (START (S 10.5) DURATION (S 3) END NIL))

END-OF-MOTION
(MOVE21 PREFERRED_ARM (WISK) COUNTER (START (S 130.5) DURATION (S 2) END NIL))
(ALIGN22 (LONGITUDINAL-AXIS OF (WISK)) (LONGITUDINAL-AXIS OF COUNTER) (START (S 130.5) DURATION
(S 2) END NIL))
(CANCEL23 HOLD16 (START (S 132.5) DURATION (S 0) END NIL))
(RELEASE24 PREFERRED_ARM (HANDLE-END PARTOF (WISK)) (START (S 132.5) DURATION (S 1) END NIL))
```

This is an example of speed, instrument, and duration modifiers

```
>>> stir the batter quickly with a wisk for 2 minutes
BEGINNING-OF-MOTION
```

```
(GRASP25 PREFERRED_ARM (HANDLE-END PARTOF (WISK)) (START (S 133.5) DURATION (S 1) END NIL))
(HOLD26 PREFERRED_ARM (HANDLE-END PARTOF (WISK)) (START (S 134.5) DURATION NIL END (UNTIL CANCEL
HOLD26)))
(MOVE27 PREFERRED_ARM (STIR-END PARTOF (WISK)) (CENTER OF BOWL) (START (S 134.5) DURATION (S
2) END NIL))
(ALIGN28 (LONGITUDINAL-AXIS OF (WISK)) (CENTRAL-AXIS OF BOWL) (START (S 134.5) DURATION (S 2)
END NIL))

PRINCIPAL-MOTION
(CONSTRAINT29 (STIR-END PARTOF (WISK)) (INSIDE OF BOWL) (START (S 136.5) DURATION (S 120) END
NIL))
(ROTATE30 PREFERRED_ARM (WISK) (CENTRAL-AXIS OF BOWL) (- (RADIUS OF CONTAINER) SMALL-AMOUNT)
(REPS 80) (START (S 136.5) DURATION (S 1.5) END NIL))

END-OF-MOTION
(MOVE31 PREFERRED_ARM (WISK) COUNTER (START (S 256.5) DURATION (S 2) END NIL))
(ALIGN32 (LONGITUDINAL-AXIS OF (WISK)) (LONGITUDINAL-AXIS OF COUNTER) (START (S 256.5) DURATION
(S 2) END NIL))
(CANCEL33 HOLD26 (START (S 258.5) DURATION (S 0) END NIL))
(RELEASE34 PREFERRED_ARM (HANDLE-END PARTOF (WISK)) (START (S 258.5) DURATION (S 1) END NIL))
```

This example demonstrates that the system requires any verb whose lexical aspect is a process to be accompanied by a duration modifier

```
>>> stir the soup
THIS TASK CANNOT BE ANIMATED
DURATION TIME MUST BE SPECIFIED BECAUSE
THE LEXICAL-ASPECT OF
STIR5
IS A PROCESS
>>>
The Knowledge Base contains representations of the following verb instances:
(COVER1 COVER2 STIR3 STIR4 STIR5)
and the following event-instances:
(EVENT1 EVENT2 EVENT3 EVENT4 EVENT5)
GOOD-BYE
```

The system has been restarted here. This is an example of a verb whose lexical aspect is a culminated process.

```
Enter your command
>>> slice the bread
BEGINNING-OF-MOTION
(GRASP1 PREFERRED_ARM (HANDLE-END PARTOF KNIFE) (START (S 0) DURATION (S 1) END NIL))
(HOLD2 PREFERRED_ARM (HANDLE-END PARTOF KNIFE) (START (S 1) DURATION NIL END (UNTIL CANCEL HOLD2)))
(GRASP3 FREE_ARM (LEFT-SIDE OF BREAD) (START (S 1) DURATION (S 1) END NIL))
(HOLD4 FREE_ARM (LEFT-SIDE OF BREAD) (START (S 2) DURATION NIL END (UNTIL CANCEL HOLD4)))
```

```
PRINCIPAL-MOTION
(MOVE5 PREFERRED_ARM (CUTTING-END PARTOF KNIFE) (TOP OF (- (RIGHT-SIDE OF BREAD) 0.75)) (START
(S 2) DURATION (S 2) END NIL))
(ALIGN6 (LONGITUDINAL-AXIS OF KNIFE) (HORIZONTAL-AXIS OF BREAD) (START (S 2) DURATION (S 2) END
NIL))
(MOVE7 PREFERRED_ARM (CUTTING-END PARTOF KNIFE) CUTTING-BOARD (START (S 4) DURATION (S 2) END
NIL))
(ALIGN8 (LONGITUDINAL-AXIS OF KNIFE) (HORIZONTAL-AXIS OF CUTTING-BOARD) (START (S 4) DURATION
(S 2) END NIL))
(MOVE9 PREFERRED_ARM (CUTTING-END PARTOF KNIFE) (+ (TOP OF BREAD) (INCHES 3)) (START (S 6) DURATION
(S 2) END NIL))
(ALIGN10 (LONGITUDINAL-AXIS OF KNIFE) (HORIZONTAL-AXIS OF BREAD) (START (S 6) DURATION (S 2)
END NIL))
                                .
                                .
                                .
```

This sequence is repeated until the bread has been sliced. The number of repetitions is based on the length of the bread divided by the width of a slice

```
(MOVE77 PREFERRED_ARM (CUTTING-END PARTOF KNIFE) (TOP OF (- (RIGHT-SIDE OF BREAD) 0.75)) (START
(S 74) DURATION (S 2) END NIL))
(ALIGN78 (LONGITUDINAL-AXIS OF KNIFE) (HORIZONTAL-AXIS OF BREAD) (START (S 74) DURATION (S 2)
END NIL))
(MOVE79 PREFERRED_ARM (CUTTING-END PARTOF KNIFE) CUTTING-BOARD (START (S 76) DURATION (S 2) END
NIL))
(ALIGN80 (LONGITUDINAL-AXIS OF KNIFE) (HORIZONTAL-AXIS OF CUTTING-BOARD) (START (S 76) DURATION
(S 2) END NIL))
(MOVE81 PREFERRED_ARM (CUTTING-END PARTOF KNIFE) (+ (TOP OF BREAD) (INCHES 3)) (START (S 78)
DURATION (S 2) END NIL))
(ALIGN82 (LONGITUDINAL-AXIS OF KNIFE) (HORIZONTAL-AXIS OF BREAD) (START (S 78) DURATION (S 2)
END NIL))

END-OF-MOTION
(CANCEL83 HOLD4 (START (S 80) DURATION (S 0) END NIL))
(RELEASE84 FREE_ARM (LEFT-SIDE OF BREAD) (START (S 80) DURATION (S 1) END NIL))
(MOVE85 PREFERRED_ARM KNIFE COUNTER (START (S 81) DURATION (S 2) END NIL))
(ALIGN86 (LONGITUDINAL-AXIS OF KNIFE) (LONGITUDINAL-AXIS OF COUNTER) (START (S 81) DURATION (S
2) END NIL))
(CANCEL87 HOLD2 (START (S 83) DURATION (S 0) END NIL))
(RELEASE88 PREFERRED_ARM KNIFE (START (S 83) DURATION (S 1) END NIL))
```

When you exit from the system it lists the names of the verb and event instances which were created. *Restart* starts the system after removing all previous verb and event instances and resetting the clock to 0. The knowledge

base information and restart commands have been removed from subsequent examples to conserve space.

```
The Knowledge Base contains representations of the following
verb instances:
(SLICE1)
and the following event-instances:
(EVENT1)
GOOD-BYE
DCRL> (restart)
```

This is an example of a plural object which specifies that the action must be repeated.

```
Enter your command
>>> slice 2 tomatoes
BEGINNING-OF-MOTION
(GRASP1 PREFERRED_ARM (HANDLE-END PARTOF KNIFE) (START (S 0) DURATION (S 1) END NIL))
(HOLD2 PREFERRED_ARM (HANDLE-END PARTOF KNIFE) (START (S 1) DURATION NIL END (UNTIL CANCEL HOLD2)))
(GRASP3 FREE_ARM (LEFT-SIDE OF TOMATO) (START (S 1) DURATION (S 1) END NIL))
(HOLD4 FREE_ARM (LEFT-SIDE OF TOMATO) (START (S 2) DURATION NIL END (UNTIL CANCEL HOLD4)))

PRINCIPAL-MOTION
(MOVE5 PREFERRED_ARM (CUTTING-END PARTOF KNIFE) (TOP OF (- (RIGHT-SIDE OF TOMATO) 0.5)) (START
(S 2) DURATION (S 2) END NIL))
(ALIGN6 (LONGITUDINAL-AXIS OF KNIFE) (HORIZONTAL-AXIS OF TOMATO) (START (S 2) DURATION (S 2)
END NIL))
(MOVE7 PREFERRED_ARM (CUTTING-END PARTOF KNIFE) CUTTING-BOARD (START (S 4) DURATION (S 2) END
NIL))
(ALIGN8 (LONGITUDINAL-AXIS OF KNIFE) (HORIZONTAL-AXIS OF CUTTING-BOARD) (START (S 4) DURATION
(S 2) END NIL))
(MOVE9 PREFERRED_ARM (CUTTING-END PARTOF KNIFE) (+ (TOP OF TOMATO) (INCHES 3)) (START (S 6) DURATION
(S 2) END NIL))
(ALIGN10 (LONGITUDINAL-AXIS OF KNIFE) (HORIZONTAL-AXIS OF TOMATO) (START (S 6) DURATION (S 2)
END NIL))
                                .
                                .
                                .
(MOVE47 PREFERRED_ARM (CUTTING-END PARTOF KNIFE) (TOP OF (- (RIGHT-SIDE OF TOMATO) 0.5)) (START
(S 44) DURATION (S 2) END NIL))
(ALIGN48 (LONGITUDINAL-AXIS OF KNIFE) (HORIZONTAL-AXIS OF TOMATO) (START (S 44) DURATION (S 2)
END NIL))
(MOVE49 PREFERRED_ARM (CUTTING-END PARTOF KNIFE) CUTTING-BOARD (START (S 46) DURATION (S 2) END
NIL))
(ALIGN50 (LONGITUDINAL-AXIS OF KNIFE) (HORIZONTAL-AXIS OF CUTTING-BOARD) (START (S 46) DURATION
(S 2) END NIL))
```

```
(MOVE51 PREFERRED_ARM (CUTTING-END PARTOF KNIFE) (+ (TOP OF TOMATO) (INCHES 3)) (START (S 48)
DURATION (S 2) END NIL))
(ALIGN52 (LONGITUDINAL-AXIS OF KNIFE) (HORIZONTAL-AXIS OF TOMATO) (START (S 48) DURATION (S 2)
END NIL))

END-OF-MOTION
(CANCEL53 HOLD4 (START (S 50) DURATION (S 0) END NIL))
(RELEASE54 FREE_ARM (LEFT-SIDE OF TOMATO) (START (S 50) DURATION (S 1) END NIL))

BEGINNING-OF-MOTION
(GRASP55 FREE_ARM (LEFT-SIDE OF TOMATO) (START (S 51) DURATION (S 1) END NIL))
(HOLD56 FREE_ARM (LEFT-SIDE OF TOMATO) (START (S 52) DURATION NIL END (UNTIL CANCEL HOLD56)))

PRINCIPAL-MOTION
(MOVE57 PREFERRED_ARM (CUTTING-END PARTOF KNIFE) (TOP OF (- (RIGHT-SIDE OF TOMATO) 0.5)) (START
(S 52) DURATION (S 2) END NIL))
(ALIGN58 (LONGITUDINAL-AXIS OF KNIFE) (HORIZONTAL-AXIS OF TOMATO) (START (S 52) DURATION (S 2)
END NIL))
(MOVE59 PREFERRED_ARM (CUTTING-END PARTOF KNIFE) CUTTING-BOARD (START (S 54) DURATION (S 2) END
NIL))
(ALIGN60 (LONGITUDINAL-AXIS OF KNIFE) (HORIZONTAL-AXIS OF CUTTING-BOARD) (START (S 54) DURATION
(S 2) END NIL))
(MOVE61 PREFERRED_ARM (CUTTING-END PARTOF KNIFE) (+ (TOP OF TOMATO) (INCHES 3)) (START (S 56)
DURATION (S 2) END NIL))
(ALIGN62 (LONGITUDINAL-AXIS OF KNIFE) (HORIZONTAL-AXIS OF TOMATO) (START (S 56) DURATION (S 2)
END NIL))
                                        .
                                        .
                                        .
(MOVE99 PREFERRED_ARM (CUTTING-END PARTOF KNIFE) (TOP OF (- (RIGHT-SIDE OF TOMATO) 0.5)) (START
(S 94) DURATION (S 2) END NIL))
(ALIGN100 (LONGITUDINAL-AXIS OF KNIFE) (HORIZONTAL-AXIS OF TOMATO) (START (S 94) DURATION (S
2) END NIL))
(MOVE101 PREFERRED_ARM (CUTTING-END PARTOF KNIFE) CUTTING-BOARD (START (S 96) DURATION (S 2)
END NIL))
(ALIGN102 (LONGITUDINAL-AXIS OF KNIFE) (HORIZONTAL-AXIS OF CUTTING-BOARD) (START (S 96) DURATION
(S 2) END NIL))
(MOVE103 PREFERRED_ARM (CUTTING-END PARTOF KNIFE) (+ (TOP OF TOMATO) (INCHES 3)) (START (S 98)
DURATION (S 2) END NIL))
(ALIGN104 (LONGITUDINAL-AXIS OF KNIFE) (HORIZONTAL-AXIS OF TOMATO) (START (S 98) DURATION (S
2) END NIL))

END-OF-MOTION
(CANCEL105 HOLD56 (START (S 100) DURATION (S 0) END NIL))
(RELEASE106 FREE_ARM (LEFT-SIDE OF TOMATO) (START (S 100) DURATION (S 1) END NIL))
```

```
(MOVE107 PREFERRED_ARM KNIFE COUNTER (START (S 101) DURATION (S 2) END NIL))
(ALIGN108 (LONGITUDINAL-AXIS OF KNIFE) (LONGITUDINAL-AXIS OF COUNTER) (START (S 101) DURATION
(S 2) END NIL))
(CANCEL109 HOLD2 (START (S 103) DURATION (S 0) END NIL))
(RELEASE110 PREFERRED_ARM KNIFE (START (S 103) DURATION (S 1) END NIL))
```

This is an example of a duration modifier with a process verb. The plural object here does not cause the action to be repeated.

```
Enter your command
>>> cook 2 tomatoes for 5 minutes
BEGINNING-OF-MOTION
(GRASP1 PREFERRED_ARM (HANDLE PARTOF POT) (START (S 0) DURATION (S 1) END NIL))
(HOLD2 PREFERRED_ARM (HANDLE PARTOF POT) (START (S 1) DURATION NIL END (UNTIL CANCEL HOLD2)))
(MOVE3 PREFERRED_ARM POT (BURNER PARTOF STOVE) (START (S 1) DURATION (S 2) END NIL))
(ALIGN4 (CENTRAL-AXIS OF POT) (CENTRAL-AXIS OF (BURNER PARTOF STOVE)) (START (S 1) DURATION (S
2) END NIL))
(CANCEL5 HOLD2 (START (S 3) DURATION (S 0) END NIL))
(RELEASE6 PREFERRED_ARM (HANDLE PARTOF POT) (START (S 3) DURATION (S 1) END NIL))
(GRASP7 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (START (S 4) DURATION (S 1) END NIL))
(HOLD8 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (START (S 5) DURATION NIL END (UNTIL CANCEL HOLD8)))
(ROTATE-TO9 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (CENTRAL-AXIS OF (BURNER-KNOB PARTOF STOVE))
(RADIUS OF (BURNER-KNOB PARTOF STOVE)) COUNTER-CLOCKWISE (DEGREES 300) (START (S 5) DURATION
(S 2) END NIL))
(CANCEL10 HOLD8 (START (S 7) DURATION (S 0) END NIL))
(RELEASE11 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (START (S 7) DURATION (S 1) END NIL))

PRINCIPAL-MOTION
(NO-ACTION12 (TOTAL-DURATION (S 300)) (START (S 8) DURATION NIL END NIL))

END-OF-MOTION
(GRASP13 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (START (S 308) DURATION (S 1) END NIL))
(HOLD14 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (START (S 309) DURATION NIL END (UNTIL CANCEL
HOLD14)))
(ROTATE-TO15 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (CENTRAL-AXIS OF (BURNER-KNOB PARTOF STOVE))
(RADIUS OF (BURNER-KNOB PARTOF STOVE)) CLOCKWISE (DEGREES 300) (START (S 309) DURATION (S 2)
END NIL))
(CANCEL16 HOLD14 (START (S 311) DURATION (S 0) END NIL))
(RELEASE17 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (START (S 311) DURATION (S 1) END NIL))
(GRASP18 PREFERRED_ARM (HANDLE PARTOF POT) (START (S 312) DURATION (S 1) END NIL))
(HOLD19 PREFERRED_ARM (HANDLE PARTOF POT) (START (S 313) DURATION NIL END (UNTIL CANCEL HOLD19)))
(MOVE20 PREFERRED_ARM POT COUNTER (START (S 313) DURATION (S 2) END NIL))
(ALIGN21 (VERTICAL-AXIS OF POT) (VERTICAL-AXIS OF COUNTER) (START (S 313) DURATION (S 2) END
NIL))
(CANCEL22 HOLD19 (START (S 315) DURATION (S 0) END NIL))
(RELEASE23 PREFERRED_ARM (HANDLE PARTOF POT) (START (S 315) DURATION (S 1) END NIL))
```

This is an example of a container modifier and a duration modifier.

```
Enter your command
>>> cook the soup in a large pot for 10 minutes
BEGINNING-OF-MOTION
(GRASP1 PREFERRED_ARM (HANDLE PARTOF ((POT LARGE))) (START (S 0) DURATION (S 1) END NIL))
(HOLD2 PREFERRED_ARM (HANDLE PARTOF ((POT LARGE))) (START (S 1) DURATION NIL END (UNTIL CANCEL
HOLD2)))
(MOVE3 PREFERRED_ARM ((POT LARGE)) (BURNER PARTOF STOVE) (START (S 1) DURATION (S 2) END NIL))
(ALIGN4 (CENTRAL-AXIS OF ((POT LARGE))) (CENTRAL-AXIS OF (BURNER PARTOF STOVE)) (START (S 1)
DURATION (S 2) END NIL))
(CANCEL5 HOLD2 (START (S 3) DURATION (S 0) END NIL))
(RELEASE6 PREFERRED_ARM (HANDLE PARTOF ((POT LARGE))) (START (S 3) DURATION (S 1) END NIL))
(GRASP7 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (START (S 4) DURATION (S 1) END NIL))
(HOLD8 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (START (S 5) DURATION NIL END (UNTIL CANCEL HOLD8)))
(ROTATE-TO9 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (CENTRAL-AXIS OF (BURNER-KNOB PARTOF STOVE))
(RADIUS OF (BURNER-KNOB PARTOF STOVE)) COUNTER-CLOCKWISE (DEGREES 300) (START (S 5) DURATION
(S 2) END NIL))
(CANCEL10 HOLD8 (START (S 7) DURATION (S 0) END NIL))
(RELEASE11 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (START (S 7) DURATION (S 1) END NIL))

PRINCIPAL-MOTION
(NO-ACTION12 (TOTAL-DURATION (S 600)) (START (S 8) DURATION NIL END NIL))

END-OF-MOTION
(GRASP13 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (START (S 608) DURATION (S 1) END NIL))
(HOLD14 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (START (S 609) DURATION NIL END (UNTIL CANCEL
HOLD14)))
(ROTATE-TO15 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (CENTRAL-AXIS OF (BURNER-KNOB PARTOF STOVE))
(RADIUS OF (BURNER-KNOB PARTOF STOVE)) CLOCKWISE (DEGREES 300) (START (S 609) DURATION (S 2)
END NIL))
(CANCEL16 HOLD14 (START (S 611) DURATION (S 0) END NIL))
(RELEASE17 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (START (S 611) DURATION (S 1) END NIL))
(GRASP18 PREFERRED_ARM (HANDLE PARTOF ((POT LARGE))) (START (S 612) DURATION (S 1) END NIL))
(HOLD19 PREFERRED_ARM (HANDLE PARTOF ((POT LARGE))) (START (S 613) DURATION NIL END (UNTIL CANCEL
HOLD19)))
(MOVE20 PREFERRED_ARM ((POT LARGE)) COUNTER (START (S 613) DURATION (S 2) END NIL))
(ALIGN21 (VERTICAL-AXIS OF ((POT LARGE))) (VERTICAL-AXIS OF COUNTER) (START (S 613) DURATION
(S 2) END NIL))
(CANCEL22 HOLD19 (START (S 615) DURATION (S 0) END NIL))
(RELEASE23 PREFERRED_ARM (HANDLE PARTOF ((POT LARGE))) (START (S 615) DURATION (S 1) END NIL))
```

This is an example of a speed modifier with a process verb. Notice that only the principal-motion section is affected by the speed modifier.

```
Enter your command
>>> stir the batter quickly for 2 minutes
```

```
BEGINNING-OF-MOTION
(GRASP1 PREFERRED_ARM (HANDLE-END PARTOF SPOON) (START (S 0) DURATION (S 1) END NIL))
(HOLD2 PREFERRED_ARM (HANDLE-END PARTOF SPOON) (START (S 1) DURATION NIL END (UNTIL CANCEL HOLD2)))
(MOVE3 PREFERRED_ARM (STIR-END PARTOF SPOON) (CENTER OF BOWL) (START (S 1) DURATION (S 2) END
NIL))
(ALIGN4 (LONGITUDINAL-AXIS OF SPOON) (CENTRAL-AXIS OF BOWL) (START (S 1) DURATION (S 2) END NIL))

PRINCIPAL-MOTION
(CONSTRAINT5 (STIR-END PARTOF SPOON) (INSIDE OF BOWL) (START (S 3) DURATION (S 120) END NIL))
(ROTATE6 PREFERRED_ARM SPOON (CENTRAL-AXIS OF BOWL) (- (RADIUS OF CONTAINER) SMALL-AMOUNT) (REPS
80) (START (S 3) DURATION (S 1.5) END NIL))

END-OF-MOTION
(MOVE7 PREFERRED_ARM SPOON COUNTER (START (S 123) DURATION (S 2) END NIL))
(ALIGN8 (LONGITUDINAL-AXIS OF SPOON) (LONGITUDINAL-AXIS OF COUNTER) (START (S 123) DURATION (S
2) END NIL))
(CANCEL9 HOLD2 (START (S 125) DURATION (S 0) END NIL))
(RELEASE10 PREFERRED_ARM (HANDLE-END PARTOF SPOON) (START (S 125) DURATION (S 1) END NIL))
```

This is another example of a speed modifier with a process verb

```
Enter your command
>>> stir the soup slowly for 3 minutes
BEGINNING-OF-MOTION

(GRASP1 PREFERRED_ARM (HANDLE-END PARTOF SPOON) (START (S 0) DURATION (S 1) END NIL))
(HOLD2 PREFERRED_ARM (HANDLE-END PARTOF SPOON) (START (S 1) DURATION NIL END (UNTIL CANCEL HOLD2)))
(MOVE3 PREFERRED_ARM (STIR-END PARTOF SPOON) (CENTER OF BOWL) (START (S 1) DURATION (S 2) END
NIL))
(ALIGN4 (LONGITUDINAL-AXIS OF SPOON) (CENTRAL-AXIS OF BOWL) (START (S 1) DURATION (S 2) END NIL))

PRINCIPAL-MOTION
(CONSTRAINT5 (STIR-END PARTOF SPOON) (INSIDE OF BOWL) (START (S 3) DURATION (S 180) END NIL))
(ROTATE6 PREFERRED_ARM SPOON (CENTRAL-AXIS OF BOWL) (- (RADIUS OF CONTAINER) SMALL-AMOUNT) (REPS
30) (START (S 3) DURATION (S 6) END NIL))

END-OF-MOTION
(MOVE7 PREFERRED_ARM SPOON COUNTER (START (S 183) DURATION (S 2) END NIL))
(ALIGN8 (LONGITUDINAL-AXIS OF SPOON) (LONGITUDINAL-AXIS OF COUNTER) (START (S 183) DURATION (S
2) END NIL))
(CANCEL9 HOLD2 (START (S 185) DURATION (S 0) END NIL))
(RELEASE10 PREFERRED_ARM (HANDLE-END PARTOF SPOON) (START (S 185) DURATION (S 1) END NIL))
```

This is an example of a frequency modifier with a duration modifier

```
Enter your command
>>> stir the soup occasionally for 2 minutes
```

```
BEGINNING-OF-MOTION
(GRASP1 PREFERRED_ARM (HANDLE-END PARTOF SPOON) (START (S 0) DURATION (S 1) END NIL))
(HOLD2 PREFERRED_ARM (HANDLE-END PARTOF SPOON) (START (S 1) DURATION NIL END (UNTIL CANCEL HOLD2)))
(MOVE3 PREFERRED_ARM (STIR-END PARTOF SPOON) (CENTER OF BOWL) (START (S 1) DURATION (S 2) END
NIL))
(ALIGN4 (LONGITUDINAL-AXIS OF SPOON) (CENTRAL-AXIS OF BOWL) (START (S 1) DURATION (S 2) END NIL))

PRINCIPAL-MOTION
(CONSTRAINT5 (STIR-END PARTOF SPOON) (INSIDE OF BOWL) (START (S 3) DURATION (S 3) END NIL))
(ROTATE6 PREFERRED_ARM SPOON (CENTRAL-AXIS OF BOWL) (- (RADIUS OF CONTAINER) SMALL-AMOUNT) (REPS
NIL) (START (S 3) DURATION (S 3) END NIL))

(END-OF-MOTION)
(MOVE7 PREFERRED_ARM SPOON COUNTER (START (S 6) DURATION (S 2) END NIL))
(ALIGN8 (LONGITUDINAL-AXIS OF SPOON) (LONGITUDINAL-AXIS OF COUNTER) (START (S 6) DURATION (S
2) END NIL))
(CANCEL9 HOLD2 (START (S 8) DURATION (S 0) END NIL))
(RELEASE10 PREFERRED_ARM (HANDLE-END PARTOF SPOON) (START (S 8) DURATION (S 1) END NIL))

INTERVAL
NO-ACTION
(START (S 9) DURATION (S 30.0) END NIL)
```

This sequence is repeated an appropriate number of times as computed for the frequency adverbial. In this case it is repeated 3 times.

.
.
.

```
BEGINNING-OF-MOTION
(GRASP21 PREFERRED_ARM (HANDLE-END PARTOF SPOON) (START (S 78.0) DURATION (S 1) END NIL))
(HOLD22 PREFERRED_ARM (HANDLE-END PARTOF SPOON) (START (S 79.0) DURATION NIL END (UNTIL CANCEL
HOLD22)))
(MOVE23 PREFERRED_ARM (STIR-END PARTOF SPOON) (CENTER OF BOWL) (START (S 79.0) DURATION (S 2)
END NIL))
(ALIGN24 (LONGITUDINAL-AXIS OF SPOON) (CENTRAL-AXIS OF BOWL) (START (S 79.0) DURATION (S 2) END
NIL))

PRINCIPAL-MOTION
(CONSTRAINT25 (STIR-END PARTOF SPOON) (INSIDE OF BOWL) (START (S 81.0) DURATION (S 3) END NIL))
(ROTATE26 PREFERRED_ARM SPOON (CENTRAL-AXIS OF BOWL) (- (RADIUS OF CONTAINER) SMALL-AMOUNT) (REPS
NIL) (START (S 81.0) DURATION (S 3) END NIL))

(END-OF-MOTION)
(MOVE27 PREFERRED_ARM SPOON COUNTER (START (S 84.0) DURATION (S 2) END NIL))
(ALIGN28 (LONGITUDINAL-AXIS OF SPOON) (LONGITUDINAL-AXIS OF COUNTER) (START (S 8
4.0) DURATION (S 2) END NIL))
```

```
(CANCEL29 HOLD22 (START (S 86.0) DURATION (S 0) END NIL))
(RELEASE30 PREFERRED_ARM (HANDLE-END PARTOF SPOON) (START (S 86.0) DURATION (S 1
) END NIL))

INTERVAL
NO-ACTION
(START (S 87.0) DURATION (S 30.0) END NIL)
```

This is another example of a frequency modifier with a duration modifier.

```
Enter your command
>>> stir the soup frequently for 2 minutes
BEGINNING-OF-MOTION
(GRASP1 PREFERRED_ARM (HANDLE-END PARTOF SPOON) (START (S 0) DURATION (S 1) END NIL))
(HOLD2 PREFERRED_ARM (HANDLE-END PARTOF SPOON) (START (S 1) DURATION NIL END (UNTIL CANCEL HOLD2)))
(MOVE3 PREFERRED_ARM (STIR-END PARTOF SPOON) (CENTER OF BOWL) (START (S 1) DURATION (S 2) END
NIL))
(ALIGN4 (LONGITUDINAL-AXIS OF SPOON) (CENTRAL-AXIS OF BOWL) (START (S 1) DURATION (S 2) END NIL))

PRINCIPAL-MOTION
(CONSTRAINT5 (STIR-END PARTOF SPOON) (INSIDE OF BOWL) (START (S 3) DURATION (S 3) END NIL))
(ROTATE6 PREFERRED_ARM SPOON (CENTRAL-AXIS OF BOWL) (- (RADIUS OF CONTAINER) SMALL-AMOUNT) (REPS
NIL) (START (S 3) DURATION (S 3) END NIL))

(END-OF-MOTION)
(MOVE7 PREFERRED_ARM SPOON COUNTER (START (S 6) DURATION (S 2) END NIL))
(ALIGN8 (LONGITUDINAL-AXIS OF SPOON) (LONGITUDINAL-AXIS OF COUNTER) (START (S 6) DURATION (S
2) END NIL))
(CANCEL9 HOLD2 (START (S 8) DURATION (S 0) END NIL))
(RELEASE10 PREFERRED_ARM (HANDLE-END PARTOF SPOON) (START (S 8) DURATION (S 1) END NIL))

INTERVAL
NO-ACTION
(START (S 9) DURATION (S 12.0) END NIL)
```

This sequence is repeated six times

.
.
.

```
BEGINNING-OF-MOTION
(GRASP51 PREFERRED_ARM (HANDLE-END PARTOF SPOON) (START (S 105.0) DURATION (S 1) END NIL))
(HOLD52 PREFERRED_ARM (HANDLE-END PARTOF SPOON) (START (S 106.0) DURATION NIL END (UNTIL CANCEL
HOLD52)))
(MOVE53 PREFERRED_ARM (STIR-END PARTOF SPOON) (CENTER OF BOWL) (START (S 106.0) DURATION (S 2)
END NIL))
(ALIGN54 (LONGITUDINAL-AXIS OF SPOON) (CENTRAL-AXIS OF BOWL) (START (S 106.0) DURATION (S 2)
END NIL))
```

```
PRINCIPAL-MOTION
(CONSTRAINT55 (STIR-END PARTOF SPOON) (INSIDE OF BOWL) (START (S 108.0) DURATION (S 3) END NIL))
(ROTATE56 PREFERRED_ARM SPOON (CENTRAL-AXIS OF BOWL) (- (RADIUS OF CONTAINER) SMALL-AMOUNT) (REPS
NIL) (START (S 108.0) DURATION (S 3) END NIL))

(END-OF-MOTION)
(MOVE57 PREFERRED_ARM SPOON COUNTER (START (S 111.0) DURATION (S 2) END NIL))
(ALIGN58 (LONGITUDINAL-AXIS OF SPOON) (LONGITUDINAL-AXIS OF COUNTER) (START (S 111.0) DURATION
(S 2) END NIL))
(CANCEL59 HOLD52 (START (S 113.0) DURATION (S 0) END NIL))
(RELEASE60 PREFERRED_ARM (HANDLE-END PARTOF SPOON) (START (S 113.0) DURATION (S 1) END NIL))

INTERVAL
NO-ACTION
(START (S 114.0) DURATION (S 12.0) END NIL)
```

This is an example of concurrent actions. The duration of the second action is coextensive with the duration of the first action.

```
Enter your command
>>> cook the soup for 2 minutes \, stirring occasionally
BEGINNING-OF-MOTION

(GRASP1 PREFERRED_ARM (HANDLE PARTOF POT) (START (S 0) DURATION (S 1) END NIL))
(HOLD2 PREFERRED_ARM (HANDLE PARTOF POT) (START (S 1) DURATION NIL END (UNTIL CANCEL HOLD2)))
(MOVE3 PREFERRED_ARM POT (BURNER PARTOF STOVE) (START (S 1) DURATION (S 2) END NIL))
(ALIGN4 (CENTRAL-AXIS OF POT) (CENTRAL-AXIS OF (BURNER PARTOF STOVE)) (START (S 1) DURATION (S
2) END NIL))
(CANCEL5 HOLD2 (START (S 3) DURATION (S 0) END NIL))
(RELEASE6 PREFERRED_ARM (HANDLE PARTOF POT) (START (S 3) DURATION (S 1) END NIL))
(GRASP7 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (START (S 4) DURATION (S 1) END NIL))
(HOLD8 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (START (S 5) DURATION NIL END (UNTIL CANCEL HOLD8)))
(ROTATE-TO9 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (CENTRAL-AXIS OF (BURNER-KNOB PARTOF STOVE))
(RADIUS OF (BURNER-KNOB PARTOF STOVE)) COUNTER-CLOCKWISE (DEGREES 300) (START (S 5) DURATION
(S 2) END NIL))
(CANCEL10 HOLD8 (START (S 7) DURATION (S 0) END NIL))
(RELEASE11 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (START (S 7) DURATION (S 1) END NIL))

PRINCIPAL-MOTION
(NO-ACTION12 (TOTAL-DURATION (S 120)) (START (S 8) DURATION NIL END NIL))

END-OF-MOTION
(GRASP13 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (START (S 128) DURATION (S 1) END NIL))
(HOLD14 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (START (S 129) DURATION NIL END (UNTIL CANCEL
HOLD14)))
```

```
(ROTATE-TO15 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (CENTRAL-AXIS OF (BURNER-KNOB PARTOF STOVE))
(RADIUS OF (BURNER-KNOB PARTOF STOVE)) CLOCKWISE (DEGREES 300) (START (S 129) DURATION (S 2)
END NIL))
(CANCEL16 HOLD14 (START (S 131) DURATION (S 0) END NIL))
(RELEASE17 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (START (S 131) DURATION (S 1) END NIL))
(GRASP18 PREFERRED_ARM (HANDLE PARTOF POT) (START (S 132) DURATION (S 1) END NIL))
(HOLD19 PREFERRED_ARM (HANDLE PARTOF POT) (START (S 133) DURATION NIL END (UNTIL CANCEL HOLD19)))
(MOVE20 PREFERRED_ARM POT COUNTER (START (S 133) DURATION (S 2) END NIL))
(ALIGN21 (VERTICAL-AXIS OF POT) (VERTICAL-AXIS OF COUNTER) (START (S 133) DURATION (S 2) END
NIL))
(CANCEL22 HOLD19 (START (S 135) DURATION (S 0) END NIL))
(RELEASE23 PREFERRED_ARM (HANDLE PARTOF POT) (START (S 135) DURATION (S 1) END NIL))

BEGINNING-OF-MOTION
(GRASP24 PREFERRED_ARM (HANDLE-END PARTOF SPOON) (START (S 8) DURATION (S 1) END NIL))
(HOLD25 PREFERRED_ARM (HANDLE-END PARTOF SPOON) (START (S 9) DURATION NIL END (UNTIL CANCEL HOLD25)))
(MOVE26 PREFERRED_ARM (STIR-END PARTOF SPOON) (CENTER OF BOWL) (START (S 9) DURATION (S 2) END
NIL))
(ALIGN27 (LONGITUDINAL-AXIS OF SPOON) (CENTRAL-AXIS OF BOWL) (START (S 9) DURATION (S 2) END
NIL))

PRINCIPAL-MOTION
(CONSTRAINT28 (STIR-END PARTOF SPOON) (INSIDE OF BOWL) (START (S 11) DURATION (S 3) END NIL))
(ROTATE29 PREFERRED_ARM SPOON (CENTRAL-AXIS OF BOWL) (- (RADIUS OF CONTAINER) SMALL-AMOUNT) (REPS
NIL) (START (S 11) DURATION (S 3) END NIL))

END-OF-MOTION
(MOVE30 PREFERRED_ARM SPOON COUNTER (START (S 14) DURATION (S 2) END NIL))
(ALIGN31 (LONGITUDINAL-AXIS OF SPOON) (LONGITUDINAL-AXIS OF COUNTER) (START (S 14) DURATION (S
2) END NIL))
(CANCEL32 HOLD25 (START (S 16) DURATION (S 0) END NIL))
(RELEASE33 PREFERRED_ARM (HANDLE-END PARTOF SPOON) (START (S 16) DURATION (S 1) END NIL))

INTERVAL
NO-ACTION
(START (S 17) DURATION (S 30.0) END NIL)
```

The sequence of motion specifications for stir is repeated 3 times

.
.
.

```
BEGINNING-OF-MOTION
(GRASP44 PREFERRED_ARM (HANDLE-END PARTOF SPOON) (START (S 86.0) DURATION (S 1) END NIL))
(HOLD45 PREFERRED_ARM (HANDLE-END PARTOF SPOON) (START (S 87.0) DURATION NIL END (UNTIL CANCEL
HOLD45)))
```

```
(MOVE46 PREFERRED_ARM (STIR-END PARTOF SPOON) (CENTER OF BOWL) (START (S 87.0) DURATION (S 2)
END NIL))
(ALIGN47 (LONGITUDINAL-AXIS OF SPOON) (CENTRAL-AXIS OF BOWL) (START (S 87.0) DURATION (S 2) END
NIL))

PRINCIPAL-MOTION
(CONSTRAINT48 (STIR-END PARTOF SPOON) (INSIDE OF BOWL) (START (S 89.0) DURATION (S 3) END NIL))
(ROTATE49 PREFERRED_ARM SPOON (CENTRAL-AXIS OF BOWL) (- (RADIUS OF CONTAINER) SMALL-AMOUNT) (REPS
NIL) (START (S 89.0) DURATION (S 3) END NIL))

(END-OF-MOTION)
(MOVE50 PREFERRED_ARM SPOON COUNTER (START (S 92.0) DURATION (S 2) END NIL))
(ALIGN51 (LONGITUDINAL-AXIS OF SPOON) (LONGITUDINAL-AXIS OF COUNTER) (START (S 92.0) DURATION
(S 2) END NIL))
(CANCEL52 HOLD45 (START (S 94.0) DURATION (S 0) END NIL))
(RELEASE53 PREFERRED_ARM (HANDLE-END PARTOF SPOON) (START (S 94.0) DURATION (S 1) END NIL))

INTERVAL
NO-ACTION
(START (S 95.0) DURATION (S 30.0) END NIL)
```

This is an example of a duration specified by a state change.

```
Enter your command
>>> cook the tomatoes until they are soft
BEGINNING-OF-MOTION
(GRASP1 PREFERRED_ARM (HANDLE PARTOF POT) (START (S 0) DURATION (S 1) END NIL))
(HOLD2 PREFERRED_ARM (HANDLE PARTOF POT) (START (S 1) DURATION NIL END (UNTIL CANCEL HOLD2)))
(MOVE3 PREFERRED_ARM POT (BURNER PARTOF STOVE) (START (S 1) DURATION (S 2) END NIL))
(ALIGN4 (CENTRAL-AXIS OF POT) (CENTRAL-AXIS OF (BURNER PARTOF STOVE)) (START (S 1) DURATION (S
2) END NIL))
(CANCEL5 HOLD2 (START (S 3) DURATION (S 0) END NIL))
(RELEASE6 PREFERRED_ARM (HANDLE PARTOF POT) (START (S 3) DURATION (S 1) END NIL))
(GRASP7 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (START (S 4) DURATION (S 1) END NIL))
(HOLD8 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (START (S 5) DURATION NIL END (UNTIL CANCEL HOLD8)))
(ROTATE-TO9 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (CENTRAL-AXIS OF (BURNER-KNOB PARTOF STOVE))
(RADIUS OF (BURNER-KNOB PARTOF STOVE)) COUNTER-CLOCKWISE (DEGREES 300) (START (S 5) DURATION
(S 2) END NIL))
(CANCEL10 HOLD8 (START (S 7) DURATION (S 0) END NIL))
(RELEASE11 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (START (S 7) DURATION (S 1) END NIL))

PRINCIPAL-MOTION
(NO-ACTION12 (TOTAL-DURATION (S 900)) (START (S 8) DURATION NIL END NIL))

END-OF-MOTION
(GRASP13 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (START (S 908) DURATION (S 1) END NIL))
```

```
(HOLD14 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (START (S 909) DURATION NIL END (UNTIL CANCEL
HOLD14)))
(ROTATE-TO15 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (CENTRAL-AXIS OF (BURNER-KNOB PARTOF STOVE))
(RADIUS OF (BURNER-KNOB PARTOF STOVE)) CLOCKWISE (DEGREES 300) (START (S 909) DURATION (S 2)
END NIL))
(CANCEL16 HOLD14 (START (S 911) DURATION (S 0) END NIL))
(RELEASE17 PREFERRED_ARM (BURNER-KNOB PARTOF STOVE) (START (S 911) DURATION (S 1) END NIL))
(GRASP18 PREFERRED_ARM (HANDLE PARTOF POT) (START (S 912) DURATION (S 1) END NIL))
(HOLD19 PREFERRED_ARM (HANDLE PARTOF POT) (START (S 913) DURATION NIL END (UNTIL CANCEL HOLD19)))
(MOVE20 PREFERRED_ARM POT COUNTER (START (S 913) DURATION (S 2) END NIL))
(ALIGN21 (VERTICAL-AXIS OF POT) (VERTICAL-AXIS OF COUNTER) (START (S 913) DURATION (S 2) END
NIL))
(CANCEL22 HOLD19 (START (S 915) DURATION (S 0) END NIL))
(RELEASE23 PREFERRED_ARM (HANDLE PARTOF POT) (START (S 915) DURATION (S 1) END NIL))
```

Chapter 6

Knowledge Bases

SEAFACT includes the following knowledge bases: Object KB, Linguistic Term KB, Event Representation KB, Verb Representation KB, Motion Specification KB, State-of-the-World KB. Each of these is implemented in DC-RL [Cebula86] except for the last one which has not been implemented. DC-RL is a frame-based knowledge representation language. Some of its features are that it allows exceptions in inheritance and that roles can have a defined inheritance scheme which is different from that of concepts. It also includes object oriented programming. The following sections describe the structure and purpose of these knowledge bases.

6.1 Object KB

The Object KB contains a representation of each object which can be referred to in the natural language input. These objects are classified according to a very general conceptual structure. For example, all edible items are classified as food, cooking tools are classified as instruments, and cooking vessels are classified as containers. This classification scheme is used by the parser to enforce selectional restrictions attached to the grammatical rules. Most of the object representations do not currently include further information. However, additional information, such as geometrical data, can be added to the Object KB to support the lower level graphics systems. Representations, including geometrical data, will also have to be added for those objects which are supplied as defaults by the verb representations.

Some of the objects in the food category do contain additional information which is used in two places. There is information supplying default times to be associated with natural language descriptions of state changes. For example, given the sentence, *cook the tomatoes until they are soft*, the Modifier Analysis Component finds under the concept tomato, a default time associated with the role *soft* (representing cooked until soft).

There is also some dimension information and common sense information which is used

by the verb behaviors when variations in the action are dependent on information about the object being acted on. For example, the representation for tomato includes the role *slicing-width* which supplies a default value for the width of tomato slices.

6.2 Linguistic Term KB

The Linguistic Term KB provides a classification of adverbial modifiers which is used to select the proper grammar rule for each of the types of modifiers. Although these modifiers all belong to the same syntactic class, they require different semantic representations. For example, *frequently* is classified as a gradable term and *slowly* is classified as a speed term.

6.3 Primitive Actions KB

The Primitive Actions KB contains a representation of each of the primitive actions which make up the final output of the system. The concept *primitive-action* includes only the role *duration* which supplies values for the duration of each action. For example, *grasp* is a primitive action which has a default duration of one second. These durations are for demonstration purposes only and have no real significance. More sophisticated time values can be established. Such values may actually be calculated given various parameters such as the distance and rate of a movement. The primitive action information is accessed by the motion specifications in the Motion Specification KB.

6.4 Event Representation KB

Each verb in the input is considered the nucleus of an event. Each predicate in the input causes an event instance to be created. All event instances are classified as children of the generic concept *event*. The event representation has a *verb* role whose value is the verb instance which is the nucleus of that event. This allows an event to be matched with its corresponding verb. The purpose of the event concept is to contain information from the categories of verbal modifiers which can be found with any of the verbs. These categories all pertain to the temporal structure of the event described by the input sentence. Three of the event roles, *speed*, *duration*, and *repetitions* receive values from the verbal modifiers in the input sentence. The last role, *aspect*, has an **if-needed** function which computes the sentential aspect of the phrase, as opposed to the inherent lexical aspect of the verb. Section 7.2 discusses inherent and lexical aspect.

[Waltz81] mentions the problem of representing time expressions. He suggests that verbs be represented by case frames with structured inheritance, and that "verbs that

describe events that have duration would have ancestors that elaborated on the use of time expressions for verbs with duration" [Waltz81 p. 5]. The verb representation in SEAFACT is a kind of case frame with structured inheritance. But a separate concept, the event instance, contains information about time expressions. Since the time expression roles do not depend on individual verbs, it is more efficient to represent them with a separate concept which is associated with the verb representation. All verbs have an associated event concept, since all verbs in the domain can be modified by various types of time expressions.

6.5 Verb Representation KB

The generic concept *verb* is a class whose children are generic representations of particular verbs such as *stir*. Each verb which the system accepts in the input has a generic representation in this KB. For each verb in the input, the system creates a verb instance which is a child of the appropriate generic verb concept.

The generic concept *verb* includes those roles which are common to all of the particular verbs which are its children. The role *effector* refers to the effector of the input task. The role *lexical-aspect* refers to the inherent lexical-aspect of the verb. The role *plural-object-application* is accessed by BUP. This role is discussed in 7.3.3. The role *concurrent-motion* is used when the input specifies two actions to be carried out concurrently. In that case, this role contains the value of the verb instance created for the other verb in the sentence.

Finally, in addition to the event and verb instances created for each verb, there are instances created for each section of the output motion specifications. These concepts, related to the primitive action output, are described in Section 6.6. The roles *beginning-of-motion-prims, principal-motion-prims, and end-of-motion-prims* each contain the name of the generic representation created for each section of the output of the particular verb.

The particular verb concepts, such as *stir*, include the roles inherited from the generic *verb* concept as well as all the roles which are particular to each of the verbs. Following [Palmer85], the roles are categorized according to whether they are obligatory, essential, or optional. Roles which are categorized as obligatory contain information which must come from some element of the input sentence. This information cannot be deduced from other sources. If an obligatory role is not filled by the input sentence then information required for the animation is missing. Essential roles must be filled, but not necessarily by the input sentence. In the verb representations, all essential roles have default values which are used when no value is supplied by the input sentence. Optional roles do not have to be filled at all. The obligatory and optional categories are indicated by assigning an **if-needed** function to those roles which returns the words obligatory or optional as appropriate. When a role is accessed the system should test that the value is not equal to either of these words, thereby

indicating in the one case that an obligatory value is missing, and in the other case that the value is optional and therefore not necessary. However, this test has not been implemented.

When an input sentence has been parsed, a verb instance is created for the verb in the sentence, for example, *stir3*. This verb instance is a child of the concept *stir* and at least some of its roles are filled in when the verb instance is created. The values come from the semantic analysis of the input sentence, created by BUP. Those roles which are essential and are not filled in by the input sentence have default values associated with them. For example, the default value for the *container* role of the verb *stir* is *bowl*.

6.5.1 Output Specifications

Each particular verb concept also has some behaviors associated with it which provide the motion specification for that verb. That is, these behaviors create the output of the system which will be input to the graphical animation system. The motion specification can have up to three parts, *beginning-of-motion, principal-motion, end-of-motion.* The output is discussed fully in Chapter 8.

6.6 Motion Specification KB

The output of SEAFACT consists of a specification of the motions associated with the verbs in the input. This specification is given as a series of primitive actions, the objects associated with the actions, and the times associated with the actions. This KB contains a concept for each section of the motion specification for each verb in the input domain. For example, the concept *beginning-of-motion-prims-stir* refers to the beginning-of-motion section of the motion specification for the verb stir. This concept contains roles for the duration of each of the primitive actions used in the motion specification. The duration values are specified by an **if-needed** function which accesses the Primitive Action KB.

Verbs which have a lexical aspect of process also require two additional roles in their principal-motion motion specifications. Although processes have no inherent endpoint, they are composed of a series of one or more repeatable steps. These repeatable steps are always part of the principal-motion of the process verb. For the purposes of animation, the primitive or primitives which are associated with the repeatable steps have a duration role which supplies the duration of one repetition of the primitives. Then, the *reps* role supplies the number of repetitions of the primitives which will occupy the amount of time specified for the total duration of the process. The *total-duration* role stores the latter value.

When the verb and event representation instances are created for an input sentence, motion specification instances are also created for each section of the motion specification

for the verb. For example, the verb instance *stir1* would have associated with it the motion specification *beginning-of-motion-prims-stir1*. These are the children of the generic representation of the motion specifications discussed above. To begin with, they inherit the duration values from their parents. However, these values may be modified by the Verbal Modifier Component. The purpose of this KB is to store the default duration and repetition values and to store the updated values after they are computed by the Verbal Modifier Component. Finally, the motion specification behaviors access this KB to get the durations of the primitive actions.

6.7 State-of-the-World KB

This KB is essential for the system but has not been implemented as part of this project. Before the animation can begin, the animation software must have access to the current state of the world. It must include all objects, movable or stationary, and all human figures which can potentially be referred to in the input tasks. It must also contain any objects or figures which are given as defaults in the verb representations. It is assumed that references in the output are to unique objects which are listed in this KB. A database containing geometrical information about objects exists as part of the animation software and is accessible from DC-RL. Thus geometrical information about objects named in the State-of-the-World KB can be found in the detailed geometrical database.

Chapter 7

Implementation of the Semantic Analysis of Verbal Modifiers

7.1 Introduction

Chapter 3 gives a detailed analysis of the verbal modifiers found in recipes. The following sections describe how SEAFACT implements some of these modifiers.

The BUP grammar recognizes the various modifiers by the following process. It associates a syntactic form with some set of modifiers and then narrows down the particular modifier by selectional restrictions attached to the grammar rules. These restrictions access the Object and Linguistic Term KB's for information about the semantic types of the words in the input. For example, BUP grammar rules indicate that a prepositional phrase may be any of a number of modifier types including INSTRUMENT, SUBSTANCE2, or CONTAINER. To determine which of these choices is correct a selectional restriction checks the type of the noun which is the object of the preposition. If that noun is an instrument, for example, then the modifier type is INSTRUMENT.

For each input sentence BUP creates an intermediate semantic representation of that sentence. This representation is used by the **process-input** function to fill in the roles belonging to the verb instance and the event instance for that sentence. Some of these modifiers, for example, those having to do with repetitions and durations of the action, then require additional processing. The Modifier Analysis Component does this processing. It places the resulting values in the appropriate roles in the Motion Specification Instances. The Motion Specification Behaviors, which create the actual output specification, then access both the motion specification instances and the verb and event instances to fill in the values required for the final output specification. The particular processing done by the Modifier Analysis Component for each type of modifier is discussed in the following

sections.

7.2 Aspect: Sentential and Lexical

Section 3.5 discusses the aspectual types of sentences analyzed within a global context, following the terminology of [Moens87]. It was shown that many verbal modifiers in the cooking domain are concerned with characterizing the duration or culmination of processes. Even within the global context of cooking, in many cases the meaning of a verb and its object, without the presence of other modifiers, does not include a culmination or duration. Since cooking always consists of a sequence of finite actions, descriptions of cooking tasks include the use of many verbal modifiers which supply the culmination or duration of the action.

At the implementation level it is necessary to distinguish between the lexical aspect of the verb and the sentential aspect of the entire sentence. The lexical aspect of the verb refers to the aspectual category which can be ascribed to a verb considered outside of an utterance. However, in this case the verb is still considered within the global domain of cooking. For example, the lexical aspect of the verb *stir* is a process. However, the sentential aspect of the sentence *stir the soup for 3 minutes* is a culminated process. The lexical aspect of each verb in the implementation is given as the value of the role *lexical-aspect* belonging to that verb in the Verb Representation KB. This value is used to ensure that the sentential aspect of each input sentence containing a process verb is a culminated process. That is, there must be some verbal modifier which coerces the process into a culminated process. If this is not the case, as in the sentence *stir the soup*, then the input is rejected since it would specify an animation without an ending time.

The lexical aspect is also used in the analysis of speed modifiers, as discussed below.

The sentential aspect of each event is calculated by an **if-needed** function in the event representation. The sentential aspect turns out to be the same as the lexical-aspect except in those cases where the lexical aspect is a process which is coerced into a culminated process.

7.3 The Analysis of Modifiers Which Specify Repetitions

Section 3.6 discusses the types of verbal modifiers which specify that an action be repeated. Each of these types of modifiers has been implemented.

7.3.1 Cardinal Count Adverbials

The BUP grammar identifies adverbs which specify a cardinal number of repetitions, such as *once* and creates the following semantic representation for them: (REPETITIONS (EXPLICIT *cardinal-term*)).[1] The Modifier Analysis Component then looks up in the Object KB the cardinal number associated with *cardinal-term* and calls a function to output the motion specification for the verb the appropriate number of times.

7.3.2 Frequency Adverbials

Frequency adverbials, such as *frequently* and *occasionally*, specify that an action be repeated some number of times. But they also imply that there be an interval of time between each repetition.

The BUP grammar identifies frequency adverbials and creates the following semantic representation for them: (REPETITIONS (FREQUENCY *frequency-term*)). Frequency adverbials require more processing of this intermediate semantic representation than cardinal count adverbials do. This processing is done by a function called by the Modifier Analysis Component. This function checks that a duration has also been specified for the event which the frequency adverbial modifies. This is necessary since an event modified by a frequency adverbial without any duration has the aspectual type of a process. An example of this is the sentence *stir the soup frequently* in a global context in which no duration for this event has been mentioned. The number of repetitions and the length of the intervals between repetitions are computed according to the algorithm described in section 3.6.2. A function is then called to output the motion specification the appropriate number of times (equal to the number of repetitions), followed each time by an interval during which no action is specified. A planner could be used to further process the output by specifying another action to take place during the intervals of no action.

7.3.3 Plural Objects

The presence of a plural object can but does not always indicate that the action of the verb is to be repeated for each object. Verbs can be categorized according to whether a plural object indicates that the action is applied once to the objects as a mass or is repeated for each discrete object. *Stir*, for example, falls in the category of verbs which do not indicate a repeated action when used with plural objects. *Slice* falls in the category which does indicate that the action should be repeated for each object. A third category is composed of verbs such as *chop*. In this category, whether the action is to be repeated depends on the physical nature of the object.

[1] Words appearing in slanted type face are replaced by a value.

The role *plural-object-application* attached to the verb representation contains the information about which category the verb belongs to. Its range of values is *discrete*, *mass*, or *discrete/mass*. The BUP grammar accesses this role to decide if a plural object belongs in the REPETITIONS role of the event representation. The implementation does not include any verbs in the discrete/mass category. If it did, it would be necessary to access the Object KB to see if the particular plural object should be treated as a mass or discretely.

In the case of a verb which requires its action to be repeated BUP creates the following intermediate semantic representation: (REPETITIONS (PLURAL *plural-noun*)). It also looks up the singular case of the noun to put in the substance role since each repetition of the action is to be applied to a discrete member of the plural noun. The Modifier Analysis Component calls a function to output the motion specification the appropriate number of times, that is, once for each of the objects.

Unlike the case of the frequency adverbials, no time interval is assumed between repetitions of the action. However, because of this, it may not actually be appropriate to repeat the entire action. Some parts of the beginning and ending of the motion specification act to cancel each other in that they involve preparing for the action and then returning the world to its previous state except for whatever change the action brought about. Which parts of the action cancel each other depends on what changes between repetitions of the action. Most commonly, with plural objects, it is just the object itself which changes from one repetition to the next. A special motion specification is called in this case. However, a more intelligent solution would be to have a function which created an appropriate motion specification depending on the kind of repetition. This has not been implemented.

7.4 The Analysis of Modifiers Which Specify Duration

Sections 3.7 and 3.8 discuss those verbal modifiers which specify the duration of the action. Three of these categories of modifiers have been implemented. In the cooking domain, durations are specified for processes, thereby coercing them into culminated processes. One could also specify the duration of a culminated process, for example, *It should take 15 minutes to slice the tomatoes.* In this case the duration specifies how long the process is likely to continue before the culmination is reached. However, this form is not common in the task oriented language of recipes.

7.4.1 Explicit Duration

The BUP grammar identifies modifiers which specify an explicit duration by means of a time unit. A function is called to convert all time units to seconds. The BUP grammar

creates the following intermediate semantic representation: (DURATION (EXPLICIT (s *number*))) where s is the unit seconds.

When an explicit duration is given for a process, this duration must be translated, for the motion specification, into the number of times to repeat the motion associated with a single repetition of the process. The Modifier Analysis Component does this calculation. Therefore, the total duration for a process is divided by the duration of one repetition of that process, yielding the number of repetitions whose duration is equal to the total duration. The number of repetitions and the total duration are stored in the Motion Specification Instance for that verb and the information is accessed by the Motion Specification Behaviors when the output function is called by the Modifier Analysis Component.

7.4.2 Duration Co-extensive With That of Another Action

SEAFACT understands sentences in which two actions described by two different verbs with the same object occur concurrently. For example, *cook the soup for 5 minutes, stirring.* In these types of sentences it is the principal motion of the first verb and the entire motion of the second verb which are concurrent. The BUP grammar creates the following intermediate semantic representation: (CONCURRENT (*vp vp*)), where the *vp* are the usual intermediate semantic representations of verb phrases. The Modifier Analysis Component fills in the *duration* and *substance* roles of the second verb with those of the first verb. A special function is then called to output the motion specifications of the two verbs.

7.4.3 Duration Characterized by a State Change

Section 3.8 discusses durations characterized by a state change. For example, *cook the tomatoes until they are soft.* The BUP grammar recognizes prepositional phrases with the preposition *until* as state change modifiers. It creates the following intermediate semantic representation for them: (DURATION (STATE *descriptor*)), where *descriptor* is the adjective which describes the state change. In order to create a simulation of the action SEAFACT needs a typical or default duration to associate with the verb, object, and state change. This information is stored as a role under the object in the Object KB. The Modifier Analysis Component accesses this role to get the default duration associated with the state change. It then calls a function to output the motion specification.

7.5 Speed

The BUP grammar recognizes adverbial modifiers such as *slowly* as modifying the speed of the action. It creates the following intermediate semantic representation for them: (SPEED

(*speed-term*)). The Linguistic Term KB contains representations of the speed terms with a role called the *speed-factor*. This role is the amount by which the duration of an action should be multiplied to arrive at the new duration specified by the speed term. The Modifier Analysis Component accesses this information and uses it to update the duration of the actions in the Motion Specification Instance.

The lexical aspect of the verb is used to decide what sections of the motion specification are affected by the speed modifier. If the verb is a process then it is just the principal motion section which is affected. For example, *stir the soup quickly for 5 minutes* means to make the repeated rotations of the instrument quickly, probably in order to prevent the soup from burning. It does not imply that the entire motion associated with stirring, which includes picking up the instrument and putting it in the soup and later removing it from the soup, must be done quickly. The latter interpretation would mean that the speed term was meant to modify the time which the entire action takes to complete. However, processes in this domain must be specified with a duration and so the duration of the entire action is already fixed.

In contrast, if the lexical aspect of the verb is a culmination or culminated process then the duration of the entire action is meant to be modified by the speed term. An example of this is *cover the pot quickly*. The Modifier Analysis Component checks the lexical aspect of the verb and updates the appropriate sections of the Motion Specification Instances.

7.6 Other Modifiers

The BUP grammar recognizes some other modifiers of the verb and creates intermediate semantic representations for them. These semantic representations are used to fill in the roles in the verb representation. However, no further analysis is needed for these other modifiers. They are instrument, substance, and container.

Chapter 8

System Output

8.1 Introduction

The output of the system contains sufficient non-geometric information needed to drive the motion synthesis procedures. Motions are described by a small set of primitives which are defined below.

However, it will still be necessary to have an intelligent program do some interpretation of the output from SEAFACT and create from it the actual input language specifications required by the motion synthesis procedures. Besides a translation to the syntax of these procedures, this interpretation will involve the following tasks. Actual geometrical information must be filled in for all the objects and human figures. The motion specification statements must be re-ordered according to their start, duration, or end time specifications.

8.2 Motion Specifications Divided Into Three Sections

The motion specifications are divided into three sections: beginning-of-motion, principal-motion, and end-of-motion. However, not every verb has all three motion sections associated with it. The principal-motion section is always present. This division is useful because it allows capture of regularities in the application of various modifiers, which appear to be applied to certain sections of the motion rather than to the entire motion. For example, section 7.5 discusses how speed is applied to the principal-motion section alone or the entire motion depending on the lexical aspect of the verb.

This division also captures a natural semantic division of the parts of an action. The beginning-of-motion corresponds to those parts of the action which are in some way preparatory to whatever is considered the essential part of the action. Note however, that preparatory actions associated with pre-conditions to an action are different. These are totally separate actions which must be carried out before the original action can be carried out.

To give a concrete example, to get an instrument, such as a spoon, out of a drawer is a precondition to stirring something. However, grasping the spoon is a part of the action of stirring since it must be done in every instance of executing the action. Thus, grasping the spoon belongs in the beginning-of-motion section.

The principal-motion section contains those parts of the action which are essential to the definition of that action. For a process, this section will always contain the repeatable part of the process.

The end-of-motion section contains those parts of the action which finish it off. For stirring this would be moving the spoon to a resting point and releasing it.

[Waltz81] also proposes division of a verb's representation into subparts which he calls "subscripts" [Waltz81 p.5]. These are supposed to provide information about the time sequence of the subparts of an action. The subscripts would also include the preconditions and postconditions of a verb. Waltz also discusses how these subparts provide a "framework in which adverbial modification has a natural representation" [Waltz81 p. 5]. His subscripts are a much more refined divison of the verbal representation than the one used for SEAFACT. He suggests that there should be subscripts for each semantic category of adverbial modifier. These categories include force or transfer of momentum, motion, and "effects". The latter category is for adverbs such as painfully or happily.

[Waltz82] makes a more concrete proposal along similar lines. There he suggests that adverbs will be represented by the scales in the event shape diagrams. For example, time adverbials will be represented by the time scale and quantity adverbials by the scale for quantity of the verbal objects. This is very similar to the approach taken in SEAFACT. The scales are represented by default amounts for the category in question, for example duration of the primitive actions.

8.3 The Primitives

The primitives include motions, constraints, and a control primitive which can cancel other primitives. These should all be understandable by the intermediate interpreter which will put them in a form understandable by the animation software. Each of the primitives and their arguments are described in this section.

The times argument is always a triple of start time, duration, and end time. Each primitive specification must fill in two of these arguments. When the duration is given as (until cancel *primitive*) this means that the primitive remains in effect until a cancel command is encountered. Two primitives are concurrent if they have the same start times.

- (grasp effector object times) - The effector must be something capable of a grasping

action, such as a hand. The motion synthesis procedures will decide exactly how the object is grasped by using information from the geometrical database.

- (hold effector object times) - The effector will continue to grasp the object for the duration of this primitive. It must always be preceded by a grasp primitive.
- (move effector object location time) - The effector will move the object to the specified location.
- (align (*axis of object1*) (*axis of object2*) time) - This specifies how two objects are to be aligned. It is always concurrent with a move command.
- (move-to (*animate object*) location time) - An animate object moves itself to the specified location.
- (rotate effector object (*axis of rotation*) (*radius of rotation*) repetitions time) - The effector rotates the object around the axis of rotation at a distance equal to the radius of rotation. Repetitions specifies the number of rotations.
- (rotate-to effector object (*axis of object*) (*radius of object*) direction distance time) - The effector rotates an object around its own axis in the direction specified, for the number of degrees specified by distance.
- (release effector object time) - The effector releases the object. This primitive presupposes that a hold command had been in effect.
- (no-action total-duration time) - This primitive specifies that no action is to take place for the time given by the total-duration argument. This extra time argument is necessary because the duration argument which is part of the time triple refers to the inherent duration of the action. In this case, there is no inherent duration. Rather, the duration is always specified by the input command.
- (cancel primitive time) - This primitive cancels another primitive whose duration was specified as (until cancel primitive).
- (constraint object location) - This primitive is a constraint rather than an action. It says that the object must get to and stay in the location specified for the duration of this command.

The object argument in the primitive specifications can take several forms. It can merely be the name of an object such as knife. Or it can specify a part of an object such as (handle-end partof knife). In this case the *partof* signifies that in the Object KB, the

concept handle-end is a child of the concept knife. The third possibility is for the object argument to specify some physical location of an object, such as (top of pot). In this case the *of* specifies that *top* is one of the roles of pot in the Object KB.

8.4 Creation of the Output

Each verb in the Verb KB has up to three Motion Specification Behaviors associated with it. That is, there is a behavior for each section of the motion specification for that verb. Behaviors are a form of object-oriented programming supported by DC-RL. A behavior is just a function which is associated with an object in the knowledge base and is inherited in the same way as roles of that object are inherited. The Motion Specification Behaviors access the various KB's as needed to fill in the arguments to the primitive actions.

There is also a Motion Specification Instance for each of the motion sections for each verb. This instance contains a role for the duration of each of the primitives in that motion section. The Motion Specification Instances are originally created at the same time as the verb and event instances. The duration values in the Motion Specification Instances are then updated by the Modifier Analysis Component to reflect any changes in the duration specified by modifiers in the input command. The motion specification instance may also contain some other roles which are accessed by the motion specification behaviors. The roles *total-duration* and *reps* are present for processes. These roles are necessary because the total duration of a process is equal to the duration of one repetition of a process multiplied by the number of repetitions. Each of these values is used by the behavior associated with a process verb.

Each motion specification instance also has a role *duration-roles* which is just a list of the duration roles which need to be included when adding the total duration of the motion specification section.

The behavior for a motion specification section accesses the Motion Specification Instance for that section in order to fill in the duration information for the primitive actions.

The output is created by invoking the Motion Specification Behaviors for each of the motion sections for each verb in the input.

Chapter 9

Conclusion

SEAFACT is a successful implementation of a natural language interface to a computer-generated animation system, operating in the domain of cooking tasks. The system implements a number of the verbal modifiers which are analyzed in Chapter 3. Although the system only accepts a small set of verbs and grammatical constructions, both the grammar and the lexicon of verbs could be extended fairly easily.

Because the semantic representation used by SEAFACT was developed to drive animation software, a great deal of information had to be included which was not explicit in the natural language input. This included physical data about the objects in the domain and default or "script" information which is necessary when objects, essential to the action, are not mentioned explicitly in the input.

The output of the system consists of motion specifications for each predicate in the input. These specifications are composed of primitive actions which are interpretable by the animation software. Adding new verbs to the implementation will require decomposition of those verbs into primitive actions. Although the primitive actions used in this implementation are sufficient for many verbs, some additional primitives may need to be added. The motion specifications also include a time component for each primitive action. This time component consists of two of the three elements: start time, duration, and end time.

9.1 Future Research

There are a number of ways in which this implementation could be expanded and improved. First, as mentioned in Section 6.7, a state of the world knowledge base must be added. This knowledge base will supply necessary information to the animation software as well as supporting pre- and post- conditions for the actions associated with the verbs. Pre- and post-conditions must also be added to the verb representations (see Chapter 4).

The linguistic ability of the system could be improved in several ways. More grammatical

constructions could be accepted by adding to the BUP grammar. The examples from recipes in Chapter 3 could be used to choose useful constructions. More of the verbal modifiers described in Chapter 3 could be implemented. Interesting ones to start with would be durations given by gradable terms (Section 3.7.2), durations characterized by state changes with active tests (Section 3.8.2), and time relations between actions (Section 3.9). A more sophisticated and complete representation of world knowledge could be built to support durations characterized by state changes. Any duration characterized by a state change requires that the system have some default information about the time associated with bringing about that state change. Finally, a small discourse analysis component could be added so that the system would keep track of what has been mentioned in previous sentences. This would allow input of a series of commands in the form of an actual recipe. Also, more verbs could be added to the lexicon, as mentioned above.

The system could be extended to other domains. Although the system incorporates alot of domain specific world knowledge, the format of the semantic representations should be transferable to other domains. The modifier analysis component should be directly applicable to other domains although different modifiers may be more prevalent depending on the domain.

Most importantly, the final connection must be made between the output from SEAFACT and the motion synthesis procedures. A sequence of input tasks should be developed for the purpose of demonstrating the actual animation. These tasks should involve movements which are interesting to animate and should present a coherent cooking procedure, for example, preparing an omelette. This will require adding new verbs to the implementation. However, the framework provided by the knowledge bases, the semantic representation of verbs and events, and the primitive motions should facilitate this expansion.

Bibliography

[Badler75] Norman I. Badler, *Temporal Scene Analysis: Conceptual Descriptions of Object Movements*, PhD, University of Toronto, Canada, 1975, University of Pennsylvania, CIS Dept, Technical Report 76-4.

[Badler86a] Norman I. Badler and Jeffrey S. Gangel, Natural Language Input for Human Task Description, in *Proc. Instrument Society of America Robexs '86: The Second International Workshop on Robotics and Expert Systems*, June 1986, pp. 137-148.

[Badler86b] Norman I. Badler, A Representation For Natural Human Movement, Technical Report - Graphics Lab 13, MS-CIS-86-23, University of Pennsylvania, March 1986.

[Badler87] Norman I. Badler, Articulated Figure Animation, *Computer Graphics and Applications*, The Computer Society of the IEEE, June 1987, pp. 10-11.

[Ball85] Catherine N. Ball, On the Interpretation of Descriptive and Metalinguistic Disjunction, unpublished paper, University of Pennsylvania, August 1985.

[Cebula86] David P. Cebula, The Semantic Data Model and Large Data Requirements, University of Pennsylvania, CIS Dept. ,Technical Report 87-79, Sept 1986.

[Croft84] William Croft, *The Representation of Adverbs, Adjectives and Events in Logical Form*, Technical Note 344, Artificial Intelligence Center, Computer Science and Technology Division, SRI International, Menlo Park, Ca, December 1984.

[Dowty79] D. Dowty, *Word Meaning and Montague Grammar*, Dordrecht:Reidel, 1979.

[Fishwick85] Paul A. Fishwick, *Hierarchical Reasoning: Simulating Complex Processes Over Multiple Levels of Abstraction*, PhD Thesis, Technical Report, University of Pennsylvania, 1985.

[Fishwick87] Paul A. Fishwick, The Role of Process Abstraction in Simulation, submitted to *IEEE Systems, Man and Cybernetics*, April 1987.

[Gangel84] Jeffrey Scott Gangel, A Motion Verb Interface To a Task Animation System, Master's Thesis, University of Pennsylvania, 1984.

[Gourmet86] *Gourmet Magazine*, Volume XLVI, Number 6, June 1986.

[Heatter65] Maida Heatter, *Maida Heatter's Book of Great Desserts*, Warner Books, N.Y., 1965.

[Huang75] Shuan-Fan Huang, *A Study of Adverbs*, Mouton, The Hague, 1975.

[Jaffrey81] Madhur Jaffrey, *Madhur Jaffrey's World-of-the-East Vegetarian Cooking*, Alfred A. Knopf, N.Y., 1981.

[Kalita87] Jugal Kalita, Simulating a Verb of State Change Using HIRES, term paper, University of Pennsylvania, 1987.

[Miller72] George A. Miller, English Verbs of Motion: A Case Study in Semantics and Lexical Memory, in *Coding Processes in Human Memory*, Arthur W. Melton and Edwin Martin (eds.), V. H. Winston & Sons, Washington, DC, 1972, pp. 335-372.

[Moens87] Marc Moens, Mark Steedman, Temporal Ontology in Natural Language, in *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, ACL, 1987, pp. 1-7.

[Morash82] Marion Morash, *Victory Garden Cookbook*, Alfred A. Knopf, N.Y., 1982.

[Mourelatos81] Alexander P. D. Mourelatos, Events, Processes, and States, in *Syntax and Semantics, Tense and Aspect*, Vol. 14, Philip Tedeschi and Annie Zaenen (eds.), Academic Press, New York, 1981, pp. 191-212.

[Okada87] Naoyuki Okada, Towards a Unified Understanding of Natural Language and Picture Patterns, in *Language and Artificial Intelligence, Proceedings of an International Symposiumm on Language and Artificial Intelligence*, Makoto Nagao (ed.), North Holland, Amsterdam, 1987, pp. 171-193.

[Palmer85] Martha S. Palmer, *Driving Semantics for a Limited Domain*, PHD Dissertation, University of Edinburgh, 1985.

[Passonneau86] Rebecca J. Passonneau, A Computational Model of the Semantics of Tense and Aspect, *Computational Linguistics* (forthcoming), Tech. Memo 43, Dec. 17, 1986, Unisys, Paoli Research Center, Paoli, Pa, Dec. 1986.

[Poses85] Steven Poses, Anne Clark, Becky Roller, *The Frog Commissary Cookbook*, Doubleday & Company, Garden City, N.Y., 1985.

[Robertson76] Laurel Robertson, Carol Flinders, Bronwen Godfrey, *Laurel's Kitchen: A Handbook for Vegetarian Cookery & Nutrition*, Bantam Books, Nilgiri Press, California, 1976.

[Rombauer31] Irma S. Rombauer, Marion Rombauer Becker, *Joy of Cooking*, Signet, New American Library, N.Y., 1931.

[Sahni85] Julie Sahni, *Classic Indian Vegetarian and Grain Cooking*, William Morrow and Co., Inc., N.Y., 1985.

[Simmons75] Robert F. Simmons, The Clowns Microworld, in *Theoretical Issues in Natural Language Processing*, R. Schank and B. Nash-Webber (eds), 1975.

[Takashima87] Yosuke Takashima, Hideo Shimazu, Masahiro Tomono, Story Driven Animation, CHI + GI '87 Proceedings, 1987, pp. 149-153.

[Tedeschi81] Philip J. Tedeschi, Annie Zaenen, *Syntax and Semantics, Tense and Aspect*, Volume 14, Academic Press, New York, 1981.

[Thomas78] Anna Thomas, *The Vegetarian Epicure Book Two*, Alfred A. Knopf, New York, 1978.

[Waltz81] David L. Waltz, Toward a Detailed Model of Processing For Language Describing the Physical World, in *IJCAI-1981*, pp. 1-6.

[Waltz82] David L. Waltz, Event Shape Diagrams, in *AAAI-1982*, pp. 84-87.

[Zeltzer85] David Zeltzer, Towards an Integrated View of 3-D Computer Animation, in *The Visual Computer*, Springer International, Volume 1 Number 4, Dec. 1985, pp. 249-259.

[Zeltzer87] David Zeltzer, Motor Problem Solving for Three Dimensional Computer Animation, in *Proc. L'Imaginaire Numerique*, Ecole d'Architecture de Saint-Etienne, Saint-Etienne, France, May 1987.

Appendix A

Data for Linguistic Analysis of Verbal Modifiers

A.1 Examples

This appendix contains all of the examples, taken from cookbooks, on which this study is based. Most of these examples are used in the paper.

A.1.1 Cookbooks

C Classic Indian Vegetarian and Grain Cooking

F The Frog Commissary Cookbook

G 6-86 Gourmet Magazine (June 1986)

H Maida Heatter's Book of Great Desserts

J Joy of Cooking

L Laurel's Kitchen

M Madhur Jaffrey's World of the East Vegetarian Cooking

T The Vegetarian Epicure Book Two

V Victoy Garden's Cookbook

A.1.2 Adverbs which describe the number of repetitions of the action (cardinal count adverbials [Mourelatos])

- stir *once or twice* (J, p. 349)

- baste *twice* during the cooking period (J, p. 350)
- stir them gently *3 to 4 times* in the next 5 to 10 minutes (M, p. 196)
- Stir *once* and add collards (V, p. 131)

A.1.3 Adverbs which describe the frequency of an action (frequency adverbials [Maurelatos])

- broil or grill for 10 minutes turning *frequently* and basting with the marinade (J, p. 349)
- Bring to a boil, reduce the heat, and simmer 20 minutes, stirring *occasionally*, until very thick. (F, p. 188)
- The pickles may be made up to 4 days in advance and kept covered and chilled, stirring *occasionally.* (G 6-86, p. 98)

A.1.4 Adverbs which describe the relation of the action to the object

These adverbs modify for example, whether the action is performed on one object, on many objects in a group, or on many objects separately.

- drain dry and dip each piece *separately* in (J, p. 350)
- spoon a bit on top of *each custard* (H, p. 405)
- divide the pineapple *among the buttered cups* (H, p. 407)
- beat in the eggs *one at a time* (G 6-86, p. 12)
- divide the batter *between them* (two buttered 9 inch round cake pans) (G 6-86, p. 12)
- puree the mixture *in batches* (G 6-86, p. 100)
- adding the remaining flour, *1 tablespoon at a time*, if necessary, until it forms a ball. (G 6-86, p. 80)
- Stir once, and add collards, *handful by handful*, stirring c constantly. (V, p. 131)
- *Layer beets and onions* in a small buttered casserole. (V, p. 24)
- Divide cabbage *in two* (V, p. 130)

A.1.5 adverbs which describe the location of the action on the object

- slash it *in several places* (J, p. 350)
- cut *from head to tail* (J, p. 351)
- shrimp split *lengthwise* (J, p. 351)
- press *onto top of apples* (L, p. 307)
- Combine the Parmesan and melted butter and distribute *over the eggs* (F, p. 190)
- split the underside of the tail shells *lengthwise* (G 6-86, p. 82)
- Halve the eggs *lengthwise* (G 6-86, p. 110)
- Cut stems and leaves *1 inch above beet crowns*, and put leaves aside. (V, p. 23)
- Cut the leaves and stalks *diagonally* into 1/2 inch slices, or, if very small, leave whole. (V, p. 23)

A.1.6 Prepositional phrases which introduce the instrument to be used in an action

- cut in the margerine *with a pastry cutter or two knives* (L, p. 313)
- place *by tablespoon* (L, p. 314)
- Cook over medium heat while folding the mixture *with a spatula* to blend in the cream cheese. (F, p. 186)
- *over moderately low heat* (G 6-86, p. 10)
- *using a long handled fork* (G 6-86, p. 10)
- *with a slotted spatula* (G 6-86, p. 12)

A.1.7 Adverbs which describe the speed of the action

- stir in buttermilk *with swift strokes* (L, p. 313)
- *quickly* tilt and turn the dish (H, p. 400)
- stir into it *gradually* until well blended (J, p. 351)
- *very gradually* pour (H, p. 393)
- beat in the remaining 2 tablespoons sugar *gradually* (G 6-86, p. 110)

A.1.8 Adverbs which describe the duration of an action

This category contrasts with the category of adverbs which describe the number of repetitions of an action. The latter is applicable to points while this category is applicable to processes or culminated processes.

- blend *very briefly* (L, p. 316)
- stir the yolks *slightly* just to mix (H, p. 395)
- stirring *constantly* (H, p. 393)
- stir gently but *continuously* (M, p. 197)
- Over medium-high heat lightly brown the vegetables, stirring them *constantly* (about 6 to 8 minutes). (C, p. 281)
- Stir once, and add collards, handful by handful, stirring *constantly.* (V, p. 131)

A.1.9 Adverbs which describe an explicit duration for an action

- Refrigerate and let marinate *at least 15 minutes or up to 2 hours* (F, p. 83)
- stir *for 1 minute*; set aside. (V, p. 132)
- Prick and blanch sausage *for 5 minutes* to release fat; (V, p. 132)
- bake in a preheated 350 oven *for 30-60 minutes*, depending on the age of the beets. (V, p. 24)

A.1.10 Phrases which describe the duration of an action as co-extensive with that of another action

- Cook over medium heat *while folding the mixture with a spatula* to blend in the cream cheese. (F, p. 186)
- Continue to cook *while gently folding in the cheeses* with a spatula. (F, p. 186)
- Reduce the heat to medium and fry the millet, *stirring*, for 5 minutes or until it is light golden. (C, p. 283)
- Add the beet greens and saute for 2-3 minutes, *stirring*, until they wilt and become tender. (V, p. 23)

A.1.11 Phrases which characterize the duration of an action in terms of a perceptual change

Some of these phrases are actually a disjunction of an explicit duration and a perceptual change. These will not be put in a seperate section from those sentences without the explicit duration since the form of the latter is regular and not very interesting.

There are various kinds of perceptual changes which are used in this context. Most are visual. However, tactile, taste, and smell are also used. There must be some test which verifies when the perceptual change has ocurred. For visual changes the test consists of looking at the substance in question. A preperatory action is required only if the substance is not immediately visible, for example, if it is in the oven or in a closed pot. Other perceptual changes always require active tests. For example, to perform a tactile test one must touch the substance either directly or via some instrument. Accordingly, this section is divided into subsections based on the type of perceptual change indicated and whether an active test must be performed.

Phrases are classified as requiring an active test whenever the test is explicitly mentioned or world knowledge predicts that the test could not be performed passively. In those cases where the test could be active or passive depending on the circumstances, the phrases is classified in the non-active section.

Visual changes-without active test

- steam 2 minutes or *until mussels open* (F, p. 83)
- "pan toast" in the butter on both sides *until golden, watching carefully,* as they brown very quickly. (F, p. 189) The test is explicitly described although it is not active.
- Reduce the heat to medium and fry the millet,stirring, for 5 minutes or *until it is light golden* (C, p. 283)
- Saute over high heat *until moisture is evaporated* (V, p. 131)
- beat the mixture *until it is combined well (it will look curdled)* (G 6-86, p. 107)
- In a bowl sift together the flour, the baking powder, and the salt, add the butter, and blend the mixture *until it resembles meal* (G 6-86, p. 107)
- heat in a 350 oven *until the sauce is bubbly* and beats are heated through. (V, p. 23)
- cook for two minutes or *until slightly colored.* (V, p. 23)
- Add the beet greens and saute for 2-3 minutes, stirring, *until they wilt* and become tender. (V, p. 23)

- saute in bacon fat *until barely wilted and lightly colored.* (V, p. 26)
- Boil broth *until reduced to 4 quarts* to concentrate flavor. (V, p. 133)
- Stir *to coat* with butter (V, p. 24)

Visual changes-with active test

- cook for 3-5 minutes or *until the scallops are uniformly white (test by cutting into one)* (F, p. 82)
- Add the lemon juice and water, cover, and cook about 15 minutes or until the liquid is absorbed. (F, p. 188) The test requires uncovering the pot.
- Cook the mixture over moderately low heat, stirring, *until the curd is thick enough to coat the back of a wooden spoon*, but do not let it boil. (G 6-86, p. 110)

Tactile changes-with active test

- Bring to a boil, reduce the heat, and simmer 20 minutes, stirring occasionally, *until very thick.* (F, p. 188) The test consists of stirring.
- Pour boiling water over the lentils to cover by an inch and let soak 15-30 minutes or *until tender.* (F, p. 188) The test consists of stabbing some lentils with a fork.
- heat in a 350 oven until the sauce is bubbly and *beats are heated through* (V, p. 23) The test would be to touch the beats with your finger or taste one.
- Add the beet greens and saute for 2-3 minutes, stirring, until they wilt and *become tender.* (V, p. 23) Test by attempting to cut them with a knife or by tasting.
- Simmer, partially covered, for 2-2 1/2 hours or *until the fowl is completely tender* and the flesh is falling off the bones. (V, p. 133) Test by stabbing with a fork.
- Cook beets *until tender*, drain, rinse in cold water, and slip off skins. (V, p. 25)

Taste- without active test

- Crack the eggs into the simmering water and poach them *to taste* (2-4 minutes). (F, p. 190)

A.1.12 Adverbs which describe the time relation between this action and a previous one.

- *immediately* pour the carmelized sugar (H, p. 400)
- stir them gently 3 to 4 times *in the next 5 to 10 seconds* (M, p. 196)
- *At the last minute,* stir in the scallions. *Serve at once* accompanied by the sauce and rice. (F, p. 188)
- Pour in about 3 tablespoons of batter and *quickly* swirl to coat the bottom evenly. (F, p. 191)
- Peel *as soon as they can be handled* and *while still warm* cut them into 1/2-inch cubes. (V, p. 25)
- *Just before serving* mix beets with potato mixture and season to taste. (V, p. 25)
- In the small bowl of an electric mixer beat the eggs and sugar at high speed for a few minutes. *When eggs expand to reach top of bowl, transfer to large bowl of mixer and continue to beat for 5 minutes more.* (H, p. 424)
- In the large bowl of an electric mixer, at high speed beat the eggs, yolks, salt and sugar for about 15 minutes until the mixture is as thick as soft whipped cream. Meanwhile... *When egg mixture has been sufficiently beaten, add vanilla and almond extracts, and cognac or rum.* (H, pp. 430-431)
- *When the custard has thickened, return it to the mixing bowl and set inside a large one 1/2 full of cold water.* (T, p. 334)
- Divide the hot caramel between 6 or 8 custard cups, ... The caramel will set almost immediately. *When the caramel has set, pour the custard mixture into the cups.* (T, p. 345)
- Chill the (pie) shells for about 15 minutes while you preheat the oven to 400 degrees. *When the shells are cold and firm, line them with waxed paper or foil and fill them with raw rice or beans.* (T, p. 342)
- *When the tart is done (baking), brush the apples lightly with this glaze, or drizzle it over them.* (T, p. 341)
- Heat the oil again over high heat. *When it is very hot, add the garlic and cook, stirring and turning, for 2 minutes or until it begins to c color.* (C, p. 234)

- *When the (mustard) seeds stop splattering and turn gray, pour the contents of the pan over the soup.* (C, p. 224)

A.1.13 Adverbs which modify the quantity of the object of the verb

- rub *generously* with clarified butter (J, p. 350)
- Add salt, pepper, and sage *to taste* (F, p. 190)
- brush the mixture *lightly* onto the rough sides of the pita triangles (G 6-86, p. 82)
- Sift the confectioners sugar *lightly* over the torte. (G 6-86, p. 82)
- sprinkle with lemon juice *to taste* (V, p. 24)

A.1.14 Adverbs of Degree

- Beat egg, add ground meat, and mix *well* with fork. (V, p. 24)
- In a bowl stir together *well* the sugar and the cinnamon. (G 6-86, p. 110)
- drain and dry *thoroughly* (J, p. 350)
- stirring *well* (L, p. 308)

A.1.15 Force

- pour *gently* (J, p. 351)
- Continue to cook while *gently* folding in the cheeses with a spatula. (F, p. 186)
- *gently* heat (L, p. 314)
- *without being too thorough, gently* fold (H, p. 419)
- turn them around *gently* (M, p. 193)
- toss the compote *gently* (G 6-86, p. 66)
- cook it, swirling the skillet *gently* (G 6-86, p. 132)
- Prick the shells (pastry) *lightly* with a fork. (G 6-86, p. 110)
- *gently* squeeze to remove water (V, p. 131)

A.1.16 Adverbs which characterize the end result of the action

- float a snow-pea half *prettily* over the top (M, p. 194)
- Cook the sausage, breaking it up *as finely as possible.* (F, p. 190)
- slice them *thin* (G 6-86, p. 10) (F, p. 190)
- let them cool *completely* (G 6-86, p. 12)
- cut the cucumbers lengthwise *into 1/8-inch-thick "noodles"* (G 6-86, p. 68)
- slice *into shreds or diagonal slices* (V, p. 131)
- Dice all the vegetables *the same size* for best appearance. (V, p. 25)
- flatten them *slightly* (L, p. 314)

A.1.17 Phrases which describe a purpose or justification for the action

- Cook over medium heat while folding the mixture with a spatula *to blend in the cream cheese* (F, p. 186)
- "pan toast" in the butter on both sides until golden, watching carefully, *as they brown very quickly.* (F, p. 189)
- Pour in about 3 tablespoons of batter and quickly swirl *to coat the bottom evenly.* (F, p. 191)
- Prick and blanch for 5 minutes *to release fat* (V, p. 132)
- Boil broth until reduced to 4 quarts *to concentrate flavor* (V, p. 133)
- Dice all the vegetables the same size *for best appearance.* (V, p. 25)
- gently squeeze *to remove water* (V, p. 131)

Appendix B

Knowledge Base Examples

This appendix contains short examples from each of the knowledge bases described in Chapter 6. These knowledge bases are implemented in DC-RL. The following is a short description of this representation language. For more information see Chapter 6 and [Cebula86].

The objects in DC-RL are **concepts** and **roles** which are both typed. The type of a **concept** consists of the **concept's** *access properties*, *semantic properties*, and *definedness properties*. The *access property* of all objects in this system is *real*. This property is not important for this system. The *semantic property* can be either *instance*, *class*, or *collection* although this system does not contain any *collections*. This specifies what type of **concept** the object is. If the object is an *instance* then it specifies only one object in the world and can have no children. If it is a *class* then it specifies a logically related group of items. The *definedness property* has the values *primitive* or *defined*. The former means that there are some properties of this object which are not represented while the latter means that the object is entirely defined by its representation. Another part of the definition of a **concept** is the *concept-range* which tells the type of the children of that **concept**.

For example, **concept** *food* has definedness property *defined*, access property *real*, semantic property *class*, and concept-range *instance*.

The type of a **role** consists of the *access-property*, which is like that of a **concept**, the *semantic-type*, which is either *role-instance* or *role-type*, and the *definedness property*, which is either *essential*, *relevant*, or *accidental*. A *role-instance* has a value and no children while a *role-type* defines a **role** and its range of values and should be the ancestor to a *role-instance*. The *definedness property* tells whether a **role** must have the specified value, in which case it is *essential*, whether the **role** usually has that value, in which case it is *relevant*, or whether the value is *accidental* and the **role** may have another value at another time.

Roles can have a number of different slots. The slots used in this system are *value, default-value,* and *if-needed.* If a **role** value is requested, the search order is *value, if-needed,* and then *default-value.* The *if-needed* slot defines an action to be performed to determine a value for the **role.**

Behaviors are defined by *defbehavior* and associate some functionality with a **concept.** They are also inherited.

B.1 Object KB

```
{concept food is a class of instance
   typed as defined and real from object-type
   having (
   [role soft
   typed as real and essential role-type
   from (soft-type of cook)]

   [role length
   typed as real and essential role-type
   from universe-roles]

   [role slicing-width
   typed as real and essential role-type
   from universe-roles])}

{concept soup is an instance from food}

{concept tomatoes is an instance from food
  having (
  [role soft
   with [value = 900]]

  [role length
   with [default-value = 4]]

  [role slicing-width
   with [value = .5]]

  [role singular
   from (singular of word)
     with [value = tomato]])}
```

B.2 Linguistic Term KB

```
{concept linguistic-term is a class of class
  typed as primitive and real from universe}

{concept gradable-term is a class of instance
  typed as defined and real from linguistic-term}

{concept briefly is an instance from gradable-term}

{concept continuously is an instance from gradable-term}
```

B.3 Primitive Actions KB

```
{concept primitive-action is a class of instance
   typed as defined and real from universe
     having (
     [role duration
      typed as real and relevant role-type
      from universe-roles])}

{concept grasp is an instance from primitive-action
   having (
   [role duration
     with [value = (s 1)]])}
```

B.4 Event Representation KB

```
{concept EVENT is a class of instance
      typed as defined and real from universe
      having (
      [role verb
            typed as real and relevant role-type
            from universe-roles]

      [role aspect
            typed as real and relevant role-type from universe-roles
            with [if-needed =
                 (lambda (role conc)
                    (let* ((duration (atomize
                            (eval '($ get duration of ,conc))))
```

```
                    (repetitions (atomize
                     (eval '($ get repetitions of ,conc))))
                    (verb-instance (atomize
                     (eval '($ get verb of ,conc))))
                    (lexical-aspect (atomize
                     (eval '($ get lexical-aspect of
                                        ,verb-instance)))))
               (cond ((not (null duration))
                      (eval '($ put aspect of ,conc culminated-process)))
                     ((not (null repetitions))
                      (cond ((eq lexical-aspect 'process)
                             (eval
                             '($ put aspect of ,conc culminated-process)))
                            (t (eval
                             '($ put aspect of ,conc ,lexical-aspect)))))
                     (t
                      (eval '($ put aspect of ,conc ,lexical-aspect))))))]]


    [role speed
          typed as real and relevant role-type from universe-roles
          with [default-value = nil]]

    [role duration
          typed as real and relevant role-type from universe-roles
          with [default-value = nil]]

    [role repetitions
          typed as real and relevant role-type from universe-roles
          with [default-value = nil]])}
```

B.5 Verb Representation KB

```
{concept VERB is a class of class
      typed as defined and real from universe
      having (
      [role lexical-aspect
            typed as real and relevant role-type from universe-roles]

      [role plural-object-application
```

```
            typed as real and relevant role-type from universe-roles]

      [role effector
            typed as real and relevant role-type from universe-roles]

      [role concurrent-action
            typed as real and accidental role-type from universe-roles]

      [role beginning-of-motion-prims
            typed as real and essential role-type from universe-roles]

      [role principal-motion-prims
            typed as real and essential role-type from universe-roles]

      [role end-of-motion-prims
            typed as real and essential role-type from universe-roles])}

{concept STIR is a class of instance
      typed as defined and real from VERB
      having (
      [role lexical-aspect
            with [value = process]]

      [role plural-object-application
            with [value = mass]]

      [role effector
            with [default-value = preferred_arm]]

      [role instrument
            typed as real and relevant role-type from universe-roles
            with [default-value = spoon]]

      [role container
            typed as real and accidental role-type from universe-roles
            with [default-value = bowl]]

      [role substance1
            typed as real and accidental role-type from universe-roles
            with [if-needed = (lambda (role conc) 'obligatory)]]

      [role substance2
```

```
        typed as real and accidental role-type from universe-roles
        with [if-needed = (lambda (role conc) 'optional)]]

    [role support
        typed as real and accidental role-type from universe-roles
        with [default-value = counter]]

    [role beginning-of-motion-prims
        with [value = beginning-of-motion-prims-stir]]

    [role principal-motion-prims
        with [value = principal-motion-prims-stir]]

    [role end-of-motion-prims
        with [value = end-of-motion-prims-stir]])}
```

B.5.1 Output Specification Behaviors

```
(defbehavior beginning-of-motion of stir (&optional start-arg)
   (let ((effector (eval `($ get effector of ,self)))
         (instrument (eval `($ get instrument of ,self)))
         (container  (eval `($ get container of ,self)))
         (grasp (create-prim-action 'grasp))
         (hold (create-prim-action 'hold))
         (move (create-prim-action 'move))
         (align (create-prim-action 'align)))
    (set (intern (concatenate 'string (prin1-to-string self)
                                      (prin1-to-string 'hold)))
         hold)
        `( (,grasp ,effector
            (handle-end partof ,instrument)
            ,(compute-time 'b self 'grasp :start start-arg))

           (,hold ,effector
            (handle-end partof ,instrument)
            ,(compute-time 'b self 'hold
                           :noupdate t :end `(until cancel ,hold)))

           (,move ,effector
             (stir-end partof ,instrument)
             (center of ,container)
             ,(compute-time 'b self 'move))
```

```
(,align
  (longitudinal-axis of ,instrument)
  (central-axis of ,container)
  ,(compute-time 'b self 'align :concurrent t
                      :duration (atomize (eval '($ get
                                  move-duration of
                    ,(intern (concatenate 'string
                     (prin1-to-string 'beginning-of-motion-prims-)
                     (prin1-to-string self)))))))))))
```

B.6 Motion Specifications

```
{concept beginning-of-motion-prims-stir is a class of instance
   typed as defined and real from universe
   having (
     [role duration-roles
      typed as real and relevant role-instance from universe-roles
      with [value = (grasp-duration hold-duration move-duration
                     align-duration)]]

     [role grasp-duration
      typed as real and relevant role-type from universe-roles
      with [if-needed = (lambda (role conc)
                          (eval '($ get duration of grasp)))]]

     [role hold-duration
      typed as real and relevant role-type from universe-roles
      with [if-needed = (lambda (role conc)
                          (eval '($ get duration of hold)))]]

     [role move-duration
      typed as real and relevant role-type from universe-roles
      with [if-needed = (lambda (role conc)
                          (eval '($ get duration of move)))]]

     [role align-duration
      typed as real and relevant role-type from universe-roles
      with [if-needed = (lambda (role conc)
                          (eval '($ get duration of align)))]])}
```

Appendix C

The BUP Grammar

These are the BUP grammar rules. The BUP parser is described in Chapter 4.

```
(addrules
  '(
   (S (VP))
   (S (VP COMMA VP) `(CONCURRENT ,#>1 ,#>3) -if (feat 3 'ing))
   (S1 (NP VP) #>2 )
   (VP1 (V) `(VERB ,#>1) -if (or (feat 1 'intransitive)(feat 1 'transitive)))
   (VP1 (V NP) `((VERB ,#>1 (SUBSTANCE1
                 ,(eval `($ get singular of ,(car #>2)))))
                 (REPETITIONS (PLURAL ,(cadr #>2))))
       -if (and (feat 1 'transitive)(feat 2 'count-term)
          (eq (atomize (eval `($ get plural-object-application of ,#>1)))
              'discrete)))
   (VP1 (V NP) `(VERB ,#>1 (SUBSTANCE1 ,#>2))
       -if (and (feat 1 'transitive)(or (not (feat 2 'count-term))
                (and (feat 2 'count-term)
            (eq (atomize (eval `($ get plural-object-application of ,#>1)))
                'mass)))
               (not (null (eval `($ isa ,(car #>2) food))))))
   (VP1 (V NP) `(VERB ,#>1 (CONTAINER ,#>2))
        -if (and (feat 1 'transitive)
        (not (null (eval `($ isa ,(car #>2) container))))))
   (VP (VP1 ADJP) `(,#>2) -if (feat 1 'be))
   (VP (VP1 ADVP))
   (VP (VP1))
   (VP (VP1 ADVP PP) `(,#>1 ,#>2 ,@#>3) -if (listp (first #>3)))
   (VP (VP1 ADVP PP) -if (atom (first #>3)))
   (VP (VP1 PP ADVP) `(,#>1 ,@#>2 ,#>3) -if (listp (first #>2)))
   (VP (VP1 PP ADVP) -if (atom (first #>2)))
```

```
(VP (VP1 PP) `(,#>1 ,@#>2) -if (listp (first #>2)))
(VP (VP1 PP) `(,#>1 ,#>2) -if (atom (first #>2)))
(NP (NP1))
(NP1 (N) #>1)
(NP1 (PRONOUN))
(NP1 (DET N) `(,#>2) )
(NP1 (DET ADJP N) `(,#>3 ,#>2))
(NP1 (ADJP N) `(,#>2 ,#>1)
      -f '(count-term)
      -if  (and (feat 1 'number-mod)(feat 2 'plural)))
(NP1 (ADJP N) `(,#>2 ,#>1) -if (or (not (feat 1 'number-mod))
              (not (feat 2 'plural))))
(ADJP (ADJ))
(ADJP (ADJP ADJ))
(PP (PP1 PP) `(,#>1 ,#>2))
(PP (PP1))
(PP1 (PREP NP) `(INSTRUMENT ,#>2) -if (and (eq #>1 'with)
                      (not (null (eval `($ isa ,(car #>2) instrument))))))
(PP1 (PREP NP) `(SUBSTANCE2 ,#>2) -if (and (eq #>1 'with)
                      (not (null (eval `($ isa ,(car #>2) food))))))
(PP1 (PREP NP) `(CONTAINER ,#>2) -if (and (eq #>1 'in)
                      (not (null (eval `($ isa ,(car #>2) container))))))
(PP1 (PREP NP) `(DURATION (EXPLICIT ,(convert #>2))) -if (and (eq #>1 'for)
                      (not (null (eval `($ isa ,(car #>2) time))))))
(PP1 (PREP S1) `(DURATION (STATE ,#>2)) -if (eq #>1 'until))
(ADVP (ADV) `(DURATION (GRADABLE ,#>1)) -if (not (null (eval
                                          `($ isa ,#>1 gradable-term)))))
(ADVP (ADV) `(REPETITIONS (FREQUENCY ,#>1)) -if (not (null (eval
                                          `($ isa ,#>1 frequency-term)))))
(ADVP (ADV) `(REPETITIONS (EXPLICIT ,#>1)) -if (not (null (eval
                                           `($ isa ,#>1 cardinal-term)))))
(ADVP (ADV) `(SPEED (,#>1)) -if (not (null (eval
                                          `($ isa ,#>1 speed-term)))))

(DET the)
(DET a)
(PREP during)
(PREP until)
(PREP to)
(PREP for)
(PREP with)
(PREP by)
(PREP in)
```

```
(PRONOUN them -f '(plural))
(PRONOUN it -f '(singular))
(PRONOUN they -f '(plural))
(V stir -f '(transitive))
(V stirring 'stir -f '(transitive ing))
(V cover -f '(transitive))
(V baste -f '(transitive))
(V basting -f '(transitive ing))
(V add   -f '(transitive))
(V broil -f '(transitive))
(V grill -f '(transitive))
(V turn -f '(transitive))
(V turning -f '(transitive ing))
(V cut -f '(transitive))
(V place -f '(transitive))
(V cook -f '(transitive))
(V slice -f '(transitive))
(V are -f '(intransitive be))
(V is -f '(intransitive be))
(ADJ cooking)
(ADJ sifted)
(ADJ slotted)
(ADJ large)
(ADJ brown)
(ADJ soft)
(ADJ 1 -f '(number-mod))
(ADJ 2 -f '(number-mod))
(ADJ 3 -f '(number-mod))
(ADJ 4 -f '(number-mod))
(ADJ 5 -f '(number-mod))
(ADJ 10 -f '(number-mod))
(ADV once)
(ADV twice)
(ADV frequently)
(ADV occasionally)
(ADV briefly)
(ADV continuously)
(ADV quickly)
(ADV slowly)
(N soup)
(N period)
(N ingredients)
```

```
(N mixture)
(N batter)
(N collards -f '(plural))
(N tomatoes -f '(plural))
(N tomato)
(N bread)
(N marinade)
(N spatula)
(N tablespoon)
(N cleaver)
(N wisk)
(N bowl)
(N pot)
(N dish)
(N pan)
(N minute)
(N minutes)
(N hour)
(N hours)
(N day)
(N days)
(COMMA \,)
 ))
```