46

AUTOMATIC ACQUISITION AND USE OF SOME OF THE KNOWLEDGE IN PHYSICS TEXTS

by

JOHN DINO BATALI
B. A. Biophysics,
The University of California at Berkeley
(1978)
M. S. Electrical Engineering and Computer Science,
Massachusetts Institute of Technology
(1981)

Submitted to the
DEPARTMENT OF ELECTRICAL ENGINEERING
AND COMPUTER SCIENCE
in partial fulfillment of the requirements
for the Degree of
DOCTOR OF PHILOSOPHY
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
September, 1991

Signature redacted

Signature of author _____
Department of Electrical Engineering and Computer Science
July 1, 1991

Signature redacted

Certified by _____
Gerald Jay Sussman
Matsushita Professor of Electrical Engineering
Thesis Supervisor

Signature redacted

Accepted by _____
Chair, Department Committee on Graduate Students

AUTOMATIC ACQUISITION AND USE OF SOME OF THE KNOWLEDGE IN PHYSICS TEXTS

by

JOHN DINO BATALI

Submitted to the Department of Electrical Engineering and Computer Science
on July 1, 1991 in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy.

Abstract

This report describes an investigation of the workings of expository technical texts with the goal of being able to automatically acquire and use some of the knowledge that they contain. The motivating question is: How does the presentation of material in college-level physics textbooks make it possible to solve problems based on that material?

My hypothesis is that a lot of what is needed to solve problems is encoded fairly straightforwardly in the linguistic structure of the presentation text. Presentation passages often begin by describing a scene or telling a simple story. The physics is then introduced as a commentary on the described scene or story, perhaps by deriving an equation relating quantities in the story, or by introducing a physical phenomenon that occurs in the situations described. Problems are also presented as simple stories, and solving a problem requires locating in the presentation text a description of a situation that subsumes the one in the problem so that the associated equations can be applied.

A program, called "Sagredo," was written to illustrate these ideas. Sagredo can read a passage of text describing elementary kinematics and can solve problems based on that material. As the program processes the source text, it interprets a set of definitions and rules in a unification-grammar based logic programming formalism, and constructs a representation of the occurrences described by the text. When it encounters passages that present new material, the program records definitions and rules that it can later instantiate when it reads and solves problems.

One result of my reading lots of physics books was the realization that textbooks do much more than just list the laws of physics or show how to solve problems. Especially in the earlier chapters, where the fundamental principles and methods are treated, physics textbooks often resort more to rhetorical persuasion than to the sort of logical presentation that a program can easily follow. A complete account of the workings of expository technical texts must attend to wider roles that such texts play in the creation and maintenance of their communities of practice.

Thesis Supervisor: Gerald Jay Sussman
Title: Matsushita Professor of Electrical Engineering

This project began at the Schlumberger Palo Alto Research laboratory. I learned a great deal, about gears, and linkages, and oil wells, and jackhammers, and other interesting things, from Don Brown, Pat Hayes, Glenn Kramer, Reid Smith, Marty Tenenbaum and Jeremy Wertheimer.

Schlumberger moved south, and I moved up the hill to the Xerox Palo Alto Research Center, where most of this work was done. I received advice, encouragement and moral support from Marti Hearst, Gregor Kiczales, James Mahoney, Geoffrey Nunberg, Eric Saund, and Annie Zaenan.

I am especially grateful to Per-Kristian Halvorsen and Brian Smith for their confidence in me and for their bureaucratic agility. My salary came from the Center for the Study of Language and Information at Stanford University. Office space, computers, and copying equipment were provided by Xerox.

I appreciate the efforts and patience of my committee: Patrick Winston, David McAllester, and especially, Gerry Sussman, at whose side I became a programmer.

I would like to thank the Cognitive Science Department of the University of California at San Diego for providing the ultimate impetus to getting this done.

Contents

Chapter 1

Introduction

This report describes an investigation of the workings of expository technical texts with the goal of being able to automatically acquire and use some of the knowledge that they contain. The motivating question is: How does the presentation of material in college-level physics textbooks make it possible to solve problems based on that material?

My hypothesis is that a lot of what is needed to solve problems is encoded fairly straightforwardly in the linguistic structure of the presentation text. Presentation passages often begin by describing a scene or telling a simple story. The physics is then introduced as a commentary on the described scene or story, perhaps by deriving an equation relating quantities in the story, or by introducing a physical phenomenon that occurs in the situations described. Problems are also presented as simple stories, and solving a problem requires locating in the presentation text a description of a situation that subsumes the one in the problem so that the associated equations can be applied.

A program, called "Sagredo," was written to illustrate these ideas. Sagredo can read a passage of text describing elementary kinematics and can solve problems based on that material. As the program processes the source text, it interprets a set of definitions and rules in a unification-grammar based logic programming formalism, and constructs a representation of the occurrences described by the text. When it encounters passages that present new material, the program records definitions and rules that it can later instantiate when it reads and solves problems.

One result of my reading lots of physics books was the realization that textbooks do much more than just list the laws of physics or show how to solve problems. Especially in the earlier chapters, where the fundamental principles and methods are treated, physics textbooks often resort more to rhetorical persuasion than to the sort of logical presentation that a program can easily follow. A complete account of the workings of expository technical texts must attend to wider roles that such texts play in the creation and maintenance of their communities of practice.

1.1 Textbooks as Technology

> My purpose is to set forth a very new science dealing with a very ancient subject. There is, in nature, perhaps nothing older than motion, concerning which the books written by philosophers are neither few nor small; nevertheless I have discovered by experiment some properties of it which are worth knowing and which have not hitherto been either observed or demonstrated. Some superficial observations have been made, as, for instance, that the natural motion of a heavy falling body is continuously accelerated; but to just what extent this acceleration occurs has not yet been announced; for so far as I know, no one has yet pointed out that the distances traversed, during equal instances of time, by a body falling from rest, stand to one another in the same ratio as the odd numbers beginning with unity.
>
> It has been observed that missiles and projectiles describe a curved path of some sort; however no one has pointed out the fact that this path is a parabola. But this and other facts, not few in number or less in worth knowing, I have succeeded in proving; and what I consider more important, there have been opened up to this vast and most excellent science, of which my work is merely the beginning, ways and means by which other minds more acute than mine will explore its remote corners.
>
> This discussion is divided into three parts; the first part deals with motion which is steady or uniform; the second treats of motion as we find it accelerated in nature; the third deals with the so-called violent motions and with projectiles.[1]

Thus opens the first modern presentation of kinematics. True to the advertisement, Galileo's results became the foundation of the scientific understanding of motion. Supplemented by the theories of relativity and quantum mechanics, but essentially intact, they are still a vital part of modern physics.

Even as he invented the subject, Galileo invented the way to explain it. Physics students still learn kinematics pretty much as Galileo taught it in the *Two New Sciences.* The order of presentation of topics is the same, much of the terminology is the same, as are many of the examples. The main significant differences are: Galileo used geometric constructions where modern presentations use algebra and calculus; and Galileo presented his results alongside philosophical arguments directed at contemporary Scholastic philosophers, whose physics, based on their interpretations of Aristotle, Galileo was to subvert. Modern physics textbooks are much less concerned with providing detailed philosophical support for their claims.

Isaac Newton continued the development of the physics of motion begun by Galileo, and also contributed to the refinement of its exposition. Newton's algebraic

[1] Galileo Galilei (1638) *Dialogues Concerning Two New Sciences,* [115], p. 190.

methods are more recognizable to modern students than Galileo's geometric methods. Even more familiar, perhaps, are the everyday examples Newton uses. Consider the following passage:

> **Law III**
> ***To every action there is always opposed an equal reaction: or, the mutual actions of two bodies upon each other are always equal, and directed to contrary parts.***
> Whatever draws or presses another is as much drawn or pressed by that other. If you press a stone with your finger, the finger is also pressed by the stone. If a horse draws a stone tied to a rope, the horse (if I may so say) will be equally drawn back towards the stone; for the distended rope, by the same endeavor to relax or unbend itself, will draw the horse, and will obstruct the progress of the one as much as it advances that of the other.[2]

Stones and ropes and horses have been featured in physics books every since. Furthermore, the technique of presenting an abstract principle followed by several concrete instances of that principle is still a popular means of exposition.

The technical textbook itself was probably invented by the Alexandrian mathematician Euclid, whose *Elements* of geometry [95] was a central part of technical education in Europe, the Middle East, and northern Africa for more than twenty centuries. Both Galileo and Newton use methods described by Euclid in their presentations. Word-problems seem to have been around even longer: the ancient Greek writers blame them on antecedent Egyptians and Phœnicians.[3]

It is not surprising that technical textbooks developed along with their subject matter. But I think that textbooks are more than epiphenomenonal. Textbooks are themselves objects of sophisticated complexity, the result of a long and continual process of development, modification, controversy, and, it is hoped, improvement. Furthermore, changes to the means of presentation of a technical subject, changes that eventually appear in textbooks, can often influence the subject itself. A new system of notation or nomenclature can affect how technical problems are considered. Even the mere existence of a standard presentation or set of examples or problems can provide part of the shared background that facilitates communication among the members of an otherwise diverse technical community. Technical textbooks are therefore instances of a *technology*, and it makes sense to inquire into the ways that they are used, and the ways that they work.

My research has centered on the means by which physics textbooks make it possible to solve word problems. I don't mean to suggest that solving word problems is

[2]Isaac Newton (1686) *Principia Mathematica*, [230], pp. 13–14.
[3]E.g., Proclus, in [95], pp. 144–147.

the most important activity that textbooks enable. A student studying a technical subject is training to be able to deploy technical concepts and methods in the analysis of real situations or the design of real mechanisms or experiments. Word problems are, at best, realistic approximations of such applications. At worst they are barely disguised glosses of the equations needed to solve them.

Still the ability to solve problems does form the basis for any ultimate competence in a technical domain. After all, it was the techniques of mathematical analysis of physical situations introduced by Galileo and Newton that made modern science and engineering possible. A thorough grounding in even the most shallow aspects of solving word problems is needed before any more subtle intuitions and analytical skills can develop. And practicing engineers and scientists do spend a great deal of time setting up and solving equations. My concentration on problems is intended to be a starting point, a foundation for deeper understanding, in much the same way as the ability to solve problems is itself a starting point in the acquisition of technical expertise.

1.2 My Project

My project can be summarized as an attempt to answer the following question:

> *How does the presentation of material in a technical textbook make it possible to solve problems based on that material?*

I presume that the text can be viewed as a *resource* for problem solving. By this I mean that there are properties or features of the presentation text that can be associated with corresponding aspects of the problem text, and that these associations can be used to determine the correct solutions to problems. Whether or not any reader attends to the specific properties or features I describe, whether a reader attends to all of them, or attends to them in the way that I describe, are all interesting questions, but they are not my question. My question is about the *text* itself, and what it makes available to an agent. So I am not presenting a cognitive model of the understanding of technical texts, but an account of what it is that a cognizing agent has to work with when it confronts such a text. I believe, in fact, that one ought to ask questions about cognitive processing only *after* one has a clear understanding of what the cognition is about.

My focus on problem solving allows the coherent investigation of aspects and levels of language which are often viewed as separate (or at least: viewed separately). For example in my treatment of grammar, I am not interested in determining which syntactic combinations are or are not legal or acceptable in English. Instead I am interested in which combinations are useful, and what they can be used for. And at the level of discourse, I ask: exactly how does a new physical principle get introduced? My intent is not to find general solutions to all of the linguistic issues that I face, but

to understand and illustrate the specific solutions that I find deployed in real physics book. I therefore share the same outlook as Winograd did in the design of SHRDLU [304] and believe that it is worthwhile to attack a whole, real problem, and to worry about the component problems only inasmuch as they affect the entire solution.

An objection that to arises to this approach is that it might seem entirely *ad hoc*. Given a relatively narrowly defined project (like solving physics problems) it might be possible to come up with an account that performs the project adequately but illuminates nothing else about representation or language in general. My only response to this possibility is to admit it and hope it does not render my results meaningless. But to some degree the possibility still raises an interesting empirical question: Given existing physics books, which are at least moderately complex objects, exactly what does it take, *ad hoc* or not, to be able to deal with them with any adequacy all?

I began this project by reading a lot of physics books, as well as textbooks in other domains. Whenever a problem was posed, or an argument presented that the reader was supposed to follow, I tried to locate the earlier passages in the texts that presented the information needed to solve the problem or follow the argument. I pretended that the texts were presenting logically valid arguments. So I attempted to characterize the axioms and premises and rules of inference that the texts used.

As I got better and better at this, I began to realize that a great deal of work was being done by relatively low-level linguistic structures. No deep semantic models were involved, at least for the simpler presentations and problems. For example a presentation passage may introduce an equation that is valid "when a force is exerted on a body." A later problem may describe "a tugboat pushing a barge." It is not necessary to understand what a tugboat is, or what a barge is, or the complicated social and economic and historical stories to be told about why it is that tugboats go around pushing barges—at least it is not necessary to understand all this in order to be able to apply the equation. All that is needed to be understood is that pushing is a means of exerting force, and that the force is exerted on whatever is described by the direct object of the verb "pushing."

Note that in this example the presentation text uses a prepositional phrase "on a body" where the problem text uses the direct object position "(pushing) a barge" to present the object that receives the force. The linguistic theory of "case frames," originally championed by Fillmore [99], is about the abstract level of linguistic structure where such relations are manifest. A given predicate is analyzed as taking a set of "case" or argument roles, such as "agent" "patient" "instrument" "location," etc. A verb is then described by associating it with a predicate, and listing how the argument roles are expressed. The verbs "give" and "receive," for example, are associated with the same predicate, but with the verb "give" the "initial-owner" role is expressed by the subject of the sentence and the "recipient" is expressed by the direct object, while for the verb "receive," the recipient is the subject and the initial owner is usually expressed as the object of a prepositional phrase headed by "from."

Case frames are an important level of linguistic analysis because they can serve as a sort of mediate link between syntax and semantics. For many verbs, the syntactic subject maps to the agent case role. And the agent case role, in turn, typically expresses the actual agent of the action described by the predicate associated with the main verb of the clause. General rules of syntax, and specific lexical entries for each verb determine the mapping of syntactic expression to case roles, and then different rules of interpretation map case roles to semantic argument positions. Case frames are thus an abstract level of linguistic structure postulated because of the simplicity and elegance they impart to a larger account of language.

In many cases, the relation between presentation and problem texts that enables problem solving can be expressed in terms of the relations between case frames alone, without any deeper semantic characterization of the texts. The presentation text would describe a situation in which an equation applied by setting up a set of case frames and relating the participants. A problem text would then set up a more specific set of frames, but it would be possible to see the case frames set up by the problem as a special case of the one set up by the presentation text. If so, the equation introduced by the presentation could be used to compute the values of desired quantities in the problems.

For a lot of problems that was all that seemed to be needed. Some presentations and problems, especially in the domain of motion, required just a bit more. Motion problems often involve temporally related events and situations. Rather than introduce a new level of semantic representation, I just extended the representational power of the case frame approach to allow that the case frames could be temporally related. I call these representational entities "occurrences." Like case frames, they contain specific participant slots, but they also can participate in temporal orderings. An "interval" occurrence is related to "start" and an "end" "moment" occurrences, and so forth.

This simple model of temporal relations is an example of what I call "linguistics physics." The term is mean to contrast with the idea of a "naive" or "commonsense" physics [146, 160] that supposedly accounts for or summarizes the understanding of the physical world that people possess either innately or as a result of their experience with the world. Linguistic physics is the way that physical concepts are expressed in language and inform our understanding and use of language about physical concepts. In general this model is pretty simple. Objects can be in specific spatial locations at specific times, they can move from place to place, and such motion takes time. The simple model of linguistic physics is expressed in grammar of spatial prepositions (such as "in" and "on" and "from" and "during"), in the complements certain verbs can take (thus "put" requires a location complement in sentences like "they put it on the shelf"), and in the temporal relations expressed by verb tense and aspect. Physics books rely on linguistic physics to present "real" physics partly because they have to: sentences about time and space and motion will of necessity involve the grammatical expression of the relations of linguistic physics. It is much less clear whether such texts

rely on any further naive understanding of the physical world. Indeed, as I discuss in chapter 10, textbooks often work to *banish* inappropriate physical intuitions.

These ideas were developed and tested in conjunction with a program that can read a simple passage presenting the laws of kinematics and can learn enough to be able to solve some real physics problems. The model of expository structure embodied in the program is not meant to be comprehensive—the point of the implementation is to illustrate the main theoretical ideas and demonstrate that they are precise enough to be somewhat useful. Most of the actual work that is reported here involved treating the many linguistic details that inhabit even the simplest-appearing natural language text.

The program is called "Sagredo," after the participant in Galileo's *Dialogues Concerning Two New Sciences* who learns about kinematics (and about the strengths of materials—the other "new science") from Galileo's prolocutor, Salviati, over the objections of the Scholastic philosopher, Simplicio.

The program is an interpreter for a logic programming language based on the "unification grammar" formalisms in computational linguistics [188, 272]. It begins by parsing the text into grammatical constituents. Associated with the analysis of each clause is a representation of the occurrence expressed by the main verb of the clause. Succeeding clauses of the text are used to construct more complex occurrence descriptions. Certain words and phrases introduce specific occurrence descriptions which are associated with algebraic equations relating the values of quantities in the occurrences. If the passage of text being processed is a problem, the set of equations thus introduced is finally solved to yield the value of the desired quantity.

If the passage of text being processed presents some new physics, the occurrence descriptions introduced by the text are collected into definitions of new classes of occurrence descriptions. These new class definitions can then be instantiated when processing later presentation or problem passages. The response of the program is organized around the notion of "discourse events," which are a generalization of "speech acts" [265]. Each discourse event, for example, the appearance of a referring noun phrase, or a question, or the definition of a quantity, triggers a response by the program, to find a co-referring expression, or to answer the question or to record the definition of the new type of occurrence description. The program is able to learn the basics of linear kinematics and can solve 20 simple problems based on what it has learned.

The reader may be struck by the use of the word "some" in the title of this report. Part of the reason for its inclusion is modesty based on my own respect for the complexity and sophistication of even the simplest natural language texts. But another, more specific, reason for including the word is based on the limitations of my methods of analysis. By treating the text as if it were a logically valid argument I was of necessity ignoring or minimizing the aspects of the texts that could not be treated

this way. It is impossible to present a formal argument unless the audience can be counted on to accept a common set of axioms and predicates and rules of inference. But the audience of a physics textbook is in the process of being trained, and cannot be counted on to share that set.

So the books have to do their persuasion some other way. Especially in the earlier chapters, where the more fundamental concepts and techniques are being introduced, the texts must resort to intuition, and plausibility arguments and appeals to the authority of the field. Sagredo is built, more or less, to learn physics in a straightforward way. It doesn't have to be persuaded or convinced of its plausibility.

But I think that there is an important aspect of the learning of physics that requires that one go through this process. Physics is not just a matter of definitions and laws of the sort that Sagredo can acquire. It is also a matter of intuition and concerns, and, most importantly, it is a *community* of practice, and entering that community requires being given the authority and responsibility to do so. Also I think that a lot of one's understanding of physics inheres in one's bodily reflexes and intuitions and experiences with the world. These intuitions also underlie one's ability to speak about the world. All of these aspects of the "knowledge" of physics are absent from Sagredo's domain of competence.

One of the initial motivations for this investigation was the observation that many practicing engineers and scientists rely on a small library of reference books. Many still make regular use of some of their university texts. The texts are used as repositories of equations and charts and tables, as well as examples and "how-to" advice. One practical application of this work would be the development of systems that could automate the retrieval and use of this information. For example an engineer could be considering the use of bevel gears in some application. The system could locate the equations relating the various parameters of such gears to their static and dynamic properties, set up and solve the relevant equations, perhaps asking the engineer for parameter values when necessary. If the simple reasoning that went into the design of the gear assembly were thus done interactively, later modifications to the design would also be facilitated.

The rapid pace of scientific research has resulted in an avalanche of technical papers, far in excess of what anyone has time to read, especially in such domains as biochemistry and medicine where the needs for well-informed researchers is especially urgent. It would be nice to have computers aid in the organization and assessment and retrieval of this material. But to do so requires an understanding of how technical texts structure and convey information, and some technical expertise on the part of the computers themselves. One way for computers to acquire that expertise would be for them to be able to read technical texts.

Corandic is an emurient grof with many fribs; it granks from corite, an olg which cargs like lange. Corite grinkles several other tarances, which garkers excarp by glarking the corite and starping it in tranker-clarped storbs. The tarances starp a chark which is exparged with worters, branking a slorp. This slorp is garped through several other corusces, finally frasting a pragety, blickant crankle: coranda. Coranda is a cargurt, grinkling corandic and borigen. The corandic is nacerated from the borigen by means of loracity. Thus garkers finally thrap a glick, bracht, glupous grapant, corandic, which granks in many starps.

1. What is corandic?
2. What does corandic grank from?
3. How do garkers excarp the tarances from the corite?
4. What does the slorp finally frast?
5. What is coranda?
6. How is the corandic nacerated from the borigen?
7. What do the garkers finally thrap?

Figure 1.1: From Weaver (1979) *Grammar for Teachers*, [296], p. 25.

1.3 Corandic

Consider the expository text in figure 1.1 and try to answer the associated comprehension questions. This passage is a good illustration for my project because it brings into sharp relief the aspects of language which convey the knowledge needed to answer questions or solve problems. Perhaps the most startling thing about about it is that one can evidently learn something from it—at least in the sense of "learning" implicit in the genre of comprehension tests that this passage is modeled on. Which is to say that one can answer the questions after reading the passage. It doesn't really matter much that "corandic" and "pragety" and "tranker-clarped" and so forth are all made-up words, because, after all, this kind of text is supposed to explain what they mean, or at least how they are related to one another. What is necessary to understand (and what this passage was written to illustrate) are the textual and linguistic structures and relations by which knowledge is represented in expository texts.

Understanding these linguistic structures and relations is all that is needed to answer the questions. Question 3, for example, asks, "How do garkers excarp the tarances from the corite?" A "how" question is typically answered by stating a method or means of performing the indicated action. Such methods or means are typically described with prepositional phrase or subordinate clause introduced by "by" or with "with." The second sentence of the presentation text discusses the

action in question, "glarkers escarp ...," and includes a "by" clause that can be used to answer the question: "by glarking the corite and starping it in tranker-clarped storbs." There is a further difficulty in that the question asks how the garkers excarp the torances *from* the corite, but the second sentence doesn't say that exactly. The first sentence says that corandic granks from corite and the second sentence explains that corite grinkles other tarances (presumably corandic is a tarance). It is necessary to assume that "granks" and "grinkles" are morphological variants of the same verb: if X granks from Y, then Y grinkles X.

The text in the passage and the questions illustrates a wide variety of the ways that language represents information. At the level of syntax, word endings like "-ed" and "-ing" and "-er" and "-ient" indicate inflectional and derivational inflections, and therefore signal the grammatical categories of the stems they are attached to. This, plus word order, and the presence of function words like "the" and "and," and punctuation, allow us to parse the sentences as being at least potentially meaningful. (Kupiec [196] describes a program that tags words in a text with their syntactic categories based on a hidden-Markov model. This program can accurately assign categories to a large majority of the nonsense words in this passage.)

Some of the syntactic devices provide semantic clues, as do words like "with" and "from" and "like" and "several" which are left untranslated in the passage. The syntax and semantics of sentences of the form "X is a Y" allow us to take corandic, for example, as being a mass noun, a kind of, or a subvariety of "grof," specifically, an "emurient" version of it. And whatever "fribs" are, corandic has a relatively large number of them.

Discourse structuring also plays a role in understanding the passage. The first sentence announces the main topic of the passage ("corandic"), and then begins the discussion of a related substance "corite." Then follows a sequence of clauses that describe a sequence of events, the steps in the sequence are indicated by the verb forms used ("is exparged," "branking a slorp"), and also lexically ("from," "finally," "thus"). The sequence of sentences in this part of the passage corresponds to the sequence of operations.

It is also possible to recognize in this passage a common genre that science books use when discussing substances (e.g., aluminum). Such discussions typically involve presenting the properties of the substance, specifically its useful properties; and often involve a discussion of how the substance is found or prepared, especially if the process is complex or otherwise interesting. These conventions can be used to make sense of where the discussion is going when, for example, the discussion of corandic veers immediately into a story about what happens to corite.

Having said all of this about this passage, the reader may wonder why I didn't attempt to make a program that could read passages like the corandic passage and answer their questions. The short answer is that it would be too hard. The assumptions that go into such answers can be worked out when examining such problems one by one, but in general they can go anywhere. The reason that physics is relatively

tractable is that the answers to all problems will ultimately involve setting up and solving equations. In some sense the right "logical form" for a physics problem is one that allows the appropriate equations to be introduced. It is a commentary on the difficulty of artificial intelligence that the "advanced" subject of physics should is more straightforward to treat computationally than the commonsense reasoning involved in answering comprehension questions.

1.4 Overview

The kinematics source text that Sagredo reads is printed in chapter 2. An initial demonstration of Sagredo in action on a passage from the source text is presented in chapter 3.

In chapter 4 I explain why I chose the domain of kinematics for the program. I also explain why I had to write the source text myself instead of using a passage from an existing physics textbook. Chapter 4 then discusses what it takes to be able to learn kinematics. The most important background, I claim, is an understanding of "linguistic physics"—the simple models of positions and motion that find their expression in language.

Chapter 5 describes the model of expository discourse that Sagredo embodies. The chapter begins by showing how texts use stories to present new material and to pose problems. I present a model of "occurrences" that can be used to encode the situations and events described by stories. I describe how the reasoning of the program consists mostly of annotating a structural analysis of the text by instantiating definitions and rules. The response of the program is organized around "discourse events."

The program is based on work in computational linguistics falling under the rubric of "unification grammar" in which most representational work is done by structured objects and the main computational operation involves the unification of such objects. Chapter 6 describes my version of a unification grammar interpreter, presents the formalism that it uses to express definitions and rules, and describes the heuristics and strategies it uses to analyze passages of text.

Chapter 7 describes the reasoning mechanisms used by the program. These include: a parser to compute the grammatical analysis of the sentences of the text; a "discourse event handler" that organizes the response of the program to the text; systems for locating "co-referring" expressions; an algebra system for solving and manipulating equations; and mechanisms for dealing with the temporal relations among occurrences.

A history of the program's encounter with the source text is presented in chapter 8. This chapter presents a sentence-by-sentence account of the program's operation on the presentation text, showing when and how it recognizes the presentation of new physics, and how it applies what it has learned to solve problems.

Discussions of the power and limitations of the program, extensions that could be made to it and other domains where this approach might be fruitful, are in chapter 9.

In chapter 10 I discuss the "rhetoric" of physics textbooks. Most of this chapter consists of close readings of 4 passages from the physics books that I examined, passages where something other than straightforward logical argument is in evidence. This chapter also includes a criticism of the whole approach to language exemplified by this work. Treating language on the model of rigid logical categories and rules ignores the creative, socially situated aspects of discourse on which the very possibility of meaning rests.

Related work, predecessors, and inspirations, are described in chapter 11.

****

> Sagredo: My brain already reels. My mind, like a cloud momentarily illuminated by a lightning flash, is for an instant filled with an unusual light, which now beckons to me and which now suddenly mingles and obscures strange, crude ideas.[4]

[4]Galileo, *Two New Sciences*, p. 51

Chapter 2

The Kinematics Source Text

This chapter contains the source text that Sagredo reads and the problems that the program solves. I explain why and how I wrote this text in section 4.2. The history of the program's encounter with this text occupies chapter 8. Problems that were found in physics textbooks have their sources indicated. The source texts I used are listed below. I made the rest of the problems up.

Kinematics is the branch of physics concerned with the mathematical description of motion. The quantities of position, speed, velocity, and acceleration are introduced and the relations among them explored. Problems in kinematics involve determining

---

[Bennett] Clarence E. Bennett. *Physics Problems and How to Solve Them.* College Outline Series. Barnes & Noble Books, second edition, 1972.

[Schaum] Frederick J. Bueche. *Theory and Problems of College Physics.* Schaum's Outline Series. McGraw-Hill Book Company, eighth edition, 1988.

[Giancoli] Douglas C. Giancoli. *Physics.* Prentice-Hall, Inc., 1980.

[R&H] Robert Resnick and David Halliday. *Physics, Part One.* John Wiley & Sons, third edition, 1977.

[SZY] Francis W. Sears, Mark W. Zemansky, and Hugh D. Young. *University Physics.* Addison-Wesley, fifth edition, 1976.

[Tipler] Paul A. Tipler. *Physics.* Worth Publishers, Inc., 1976.

[Weissman] Simon A. Weissman. *Modern Concepts in Physics.* Oxford Book Company, 1973.

Figure 2.1: Sources for some of the problems used in the kinematics text.

how far something will travel, how long it will take to get there, and how fast it will go. Kinematics is also concerned with describing the motion of projectiles near the surface of the earth because of the fact that such projectiles are found to accelerate downward at a constant rate regardless of their mass. The modern science of kinematics was created in virtually its present form by Galileo and first described in 1638 [115]. This chapter covers most of the topics that Galileo covered in his presentation. Treatments of kinematics in modern textbooks may also discuss rotary motion, orbits, motion with non-constant acceleration, and the kinematics of very small objects, or object moving at speeds approaching the speed of light. In both of these latter régimes, modern physics has supplanted Galileo's specific results, though his general approach remains fundamental. The related field of *dynamics* was pioneered by Newton [230], and is concerned with the causes of motion, in particular with the relation between the masses of objects, the forces on them and their resultant accelerations.

2.1 Speed

If an object moves a distance d in a time t, its *average speed* for the interval is defined to be d/t.

An object moves with a *constant speed* s during an interval when its average speed equals s for all subintervals of the interval. An object at rest has a constant speed of zero.

Problem 1.1
A cow walks 2 miles in 5 hours. What is the average speed of the cow for this walk?

Problem 1.2 [Tipler, p. 21]
How far does a car go in 5 min if its average speed is 60 mi/hr?

Problem 1.3 [Bennett 3.1.3, p. 41]
How long will it take a body moving with an average speed of 15 ft/sec to go a distance of 6000 ft?

Problem 1.4
A truck drives 200 miles in 3 hours. What must its average speed be over the next 100 miles for its average speed to be 60 mi/hr for the entire trip?

Problem 1.5
An airplane flies at a constant speed for 100 miles. In the first 10 minutes, it goes 20 miles. How long does the whole trip take?

A moving object has an *instantaneous speed* at every moment during its motion. The instantaneous speed of an object moving with a constant speed during an interval equals its constant speed at every moment during the interval. The instantaneous speed of an object at the beginning of an interval is called its *initial speed* for the interval. The object's instantaneous speed at the end of an interval is called its *final speed* for the interval.

2.2 Acceleration

A moving object whose instantaneous speed changes over an interval is said to be *accelerating* during that interval. If the instantaneous speed of an object moving along a straight line changes from s_i to s_f during a time t, its *average acceleration* for the interval is defined to be $(s_f - s_i)/t$.

An object moving along a straight line has a *constant acceleration* a during some interval of time when its average acceleration is equal to a for all subintervals of that interval. An object moving with a constant acceleration is said to accelerate at a *constant rate*, or to accelerate *uniformly*. An object moving with a constant speed has a constant acceleration of zero.

Problem 2.1
A magic carpet is moving at a constant speed of 10 m/s along a straight line. What average acceleration is needed to increase the speed of the carpet to 50 m/s in 5 s?

Problem 2.2 [Weissman, p. 44]
A car increases its speed uniformly from 10 m/sec to 30 m/sec in 10 sec. What is its average acceleration?

Problem 2.3 [Schaum 4.3, p. 33]
An object starts from rest with constant acceleration 8 m/s^2 along a straight line. Find the speed at the end of 5 s.

Problem 2.4
A car whose initial speed is 55 mi/hr begins to slow at a constant rate of 5 feet per second per second. How long does it take to stop?

A moving object has an *instantaneous acceleration* at every moment during its motion. The instantaneous acceleration of an object moving with a constant acceleration during an interval equals its constant acceleration at every moment during the interval.

2.3 Motion with Constant Acceleration

If an object is moving along a straight line with a constant acceleration a, its average speed s_{av} is given by

$$s_{av} = s_i + \frac{s_f - s_i}{2}, \tag{3.1}$$

where s_i is the initial speed of the object for the interval, and s_f is the final speed of the object for the interval.

Consider an object moving along a straight line with a constant acceleration a for a time t. From the definition of average acceleration, derive:

$$at = s_f - s_i, \tag{3.2}$$

where s_i is the initial speed of the object and s_f is its final speed. Substitute the formula for $s_f - s_i$ from equation 3.2 into equation 3.1:

$$s_{av} = s_i + \frac{at}{2}. \tag{3.3}$$

Substitute the formula for s_{av} from equation 3.3 into the definition of average speed:

$$s_i + \frac{at}{2} = \frac{d}{t}, \tag{3.4}$$

where d is the distance traveled. Now solve for d:

$$d = s_i t + \frac{1}{2}at^2. \tag{3.5}$$

Problem 3.1 [Tipler 2.31, p. 43]
A car accelerates from rest at a constant rate of 8 m/s^2. (a) How fast is it going after 10 sec? (b) How far has it gone after 10 sec?

Problem 3.2 [R&H 3.12, p. 49]
A jumbo jet needs to reach a speed of 225 mi/h on the runway for takeoff. Assuming a constant acceleration and a runway 1.1 miles long, what minimum acceleration from rest is required?

Consider an object moving along a straight line with a constant acceleration a for a time t. From the definition of average acceleration, derive:

$$t = \frac{s_f - s_i}{a}, \tag{3.6}$$

where s_i is the initial speed of the object and s_f is its final speed. From equation 3.1, derive:

$$s_{av} = \frac{s_i + s_f}{2}. \tag{3.7}$$

From the definition of average speed, derive:

$$d = s_{av}t, \tag{3.8}$$

where d is the distance traveled. Now substitute the formula for s_{av} from equation 3.7, and the formula for t from equation 3.6 into equation 3.8:

$$d = (\frac{s_i + s_f}{2})(\frac{s_f - s_i}{a}). \tag{3.9}$$

Simplify:

$$2ad = s_f^2 - s_i^2. \tag{3.10}$$

Problem 3.3 [Bennett, p. 38]
A car traveling in a straight line with initial speed of 20 miles per hour is accelerated uniformly to a speed of 40 miles per hour in 10 sec. What is the acceleration, and how far did the car travel during the interval?

Problem 3.4 [Schaum 4.10, p. 35]
A bus moving at a speed of 20 m/s begins to slow at a rate of 3 m/s each second. Find how far it goes before stopping.

2.4 Vertical Motion

An unsupported object accelerates downward at a constant rate due to gravity. The magnitude of the gravitational acceleration is approximately 9.8 m/s^2.

Suppose that an object is dropped from rest at a height h. Substitute 0 for the initial speed, h for the distance, and g for the acceleration into equation 3.5:

$$h = 0 + \frac{1}{2}gt^2, \tag{4.1}$$

where t is the time taken by the object to reach the ground. Solve for t:

$$t = \sqrt{\frac{2h}{g}}. \tag{4.2}$$

Substitute 0 for s_i, and g for a in the definition of average acceleration:

$$g = \frac{s_f - 0}{t}, \tag{4.3}$$

where s_f is the final speed of the object. Solve for s_f:

$$s_f = gt. \tag{4.4}$$

Now use the formula for t from equation 4.2:

$$s_f = g\sqrt{\frac{2h}{g}} = \sqrt{2gh}. \tag{4.5}$$

Problem 4.1 [Giancoli 1.32, p. 32]
(a) How long does it take a brick to reach the ground if dropped from a height of 80 m? (b) What will be its speed just before it reaches the ground?

Problem 4.2 [Weissman, p. 57]
A stone is thrown vertically downward with a speed of 5 m/sec. (a) How far will it go in 3 sec? (b) How fast will it be moving at the end of 3 sec?

Problem 4.3 [R&H 3.36, p. 52]
A stone is dropped into the water from a bridge 144 ft above the water. Another stone is thrown vertically down 1.0 s after the first is dropped. Both stones strike the water at the same time. What was the initial speed of the second stone?

When an object is projected vertically upward with an initial speed s_i, its instantaneous speed decreases at a constant rate. The object rises until its speed is zero. When its speed is zero, the object is at its maximum height h. Substitute h for the distance, 0 for the final speed, and $-g$ for the acceleration into equation 3.10:

$$2(-g)h = 0 - s_i^2. \tag{4.6}$$

Solve for h:

$$h = \frac{s_i^2}{2g}. \tag{4.7}$$

The time t_u taken by the object to reach the maximum height is equal to the time taken for the speed of the object to reach zero. Use equation 3.2 with 0 for the final speed, and $-g$ for a:

$$-gt_u = 0 - s_i. \tag{4.8}$$

Solve for t_u:

$$t_u = \frac{s_i}{g}. \tag{4.9}$$

Problem 4.4 [Schaum 4.13, p. 36]
A stone is thrown straight upward and it rises to a height of 20 m. With what speed was it thrown?

Problem 4.5
A plastic dinosaur tossed straight up reaches its maximum height in 3 seconds. How high did it go?

Problem 4.6 [SZY 4.22, p. 65]
With what speed must a ball be thrown vertically upward in order to rise to a height of 50 m?

Once an object projected vertically upward with an initial speed s_i reaches its maximum height, it will begin moving downward with the constant acceleration of gravity. The object will take the same time falling as it took rising. So the total time of flight t is twice the time t_u the object took to reach its maximum height:

$$t = 2t_u = 2\frac{s_i}{g}, \tag{4.10}$$

where g is the acceleration of gravity.

Problem 4.7 [Weissman 3.14.1, p. 56]
A stone is thrown straight up. It returns to the ground 6 seconds later. (a) With what speed was it thrown? (b) How high did the stone rise?

Chapter 3

A Demonstration

This chapter presents a demonstration of Sagredo in action. The program reads a passage of a physics text, follows a derivation, and uses the knowledge it acquires to solve a problem based on the presented material. This demonstration will give a picture of the capabilities of the program and preview some of the ideas on which it is based. While the presentation in this chapter will be as self-contained as I can make it, the ideas and methods discussed here will be elaborated in the next few chapters. Chapter 8 describes the adventures of the program as it deals with the rest of the source text.

A passage from section 4 of the source text is presented in figure 3.1. (It is presented in context on page 23.) I use an example from well into the text to give an idea of what the system already has to know about in order to be able to learn new information. Also we have to be this far along in order to get interesting derivations and problems.

The earlier paragraphs of section 4 were concerned with objects moving vertically under the influence of gravity. The text has just discussed the simplest case, of objects dropped from rest. An earlier paragraph introduced the constant g which stands for the acceleration of gravity.

Equation 3.10, mentioned in this passage, gave the relation between the distance traveled, the acceleration, and the initial and final speeds of an object moving with constant acceleration along a straight line:

$$2ad = s_f^2 - s_i^2. \tag{3.10}$$

The main point of this passage is to describe what happens in the first half of the flight of a vertically projected object. The rest of the flight—when the object comes down—is described later, after a few problems. The most important thing learned here is that such an object will reach a maximum height that is determined only by the initial vertical speed. The algebraic relation between the maximum height and the vertical speed can be determined because the system has already followed the derivation of equation 3.10, which applies in this situation because the gravitational

When an object affected by gravity is projected vertically upward with an initial speed s_i, its instantaneous speed decreases at a constant rate. The object rises until its speed is zero. When its speed is zero, the object is at its maximum height h. Substitute h for the distance, 0 for the final speed, and $-g$ for the acceleration into equation 3.10:

$$2(-g)h = 0 - s_i^2. \tag{4.6}$$

Solve for h:

$$h = \frac{s_i^2}{2g}. \tag{4.7}$$

Problem 4.4 [Schaum 4.13, p. 36]
A stone is thrown straight upward and it rises to a height of 20 m. With what speed was it thrown?

Figure 3.1: A passage from section 4 of the kinematics source text.

---

acceleration is constant, the initial and final speeds are known, and the distance traveled is known. Problem 4.4 is then a straightforward application of the material presented in the passage.

The program goes to work on a file of ASCII characters interspersed with LATEX [198] formatting commands. Sectioning commands print the "section" and "problem" headings and keep track of their numbers. Blank lines in the source text indicate paragraph breaks. Emphasis commands are used to put text in an italic typeface. "Math-mode" commands are used to render equations. For example the command that generates equation 4.7 is represented in the source text as:

```
\begin{equation}
h = \frac{\square{s_{i}}}{2 g}.
\label{vmmh}
\end{equation}
```

LATEX uses the braces { and } to delimit arguments. The pair of commands: `\begin{equation}` and `\end{equation}`, enclose commands that will create the equation. The command `\frac` creates a quotient, and the command `\square` puts an exponent of 2 after its argument. The sequence `s_{i}` creates the subscripted variable s_i. The `\label{vmmh}` command associates the name `vmmh` with the number

```
(SENTENCE SENTENCE-1
  (CLAUSE WHEN
    (WORD INTRODUCER "when")
    (NOUN-PHRASE SUBJECT
      (WORD DETERMINER "an")
      (WORD HEAD "object"))
    (VERB-GROUP VERB-GROUP
      (WORD AUXILIARY "is")
      (WORD HEAD "projected"))
    (PHRASE ADJUNCT
      (WORD MODIFIER "vertically")
      (WORD HEAD "upward"))
    (PREP-PHRASE WITH
      (WORD HEAD "with")
      (NOUN-PHRASE OBJECT
        (WORD DETERMINER "an")
        (WORD MODIFIER "initial")
        (WORD HEAD "speed")
        (FORMULA APPOSITIVE "$s_{i}$")))
    (PUNCTUATION ","))
  (NOUN-PHRASE SUBJECT
    (WORD DETERMINER "its")
    (WORD MODIFIER "instantaneous")
    (WORD HEAD "speed"))
  (VERB-GROUP VERB-GROUP
    (WORD HEAD "decreases"))
  (PREP-PHRASE AT
    (WORD HEAD "at")
    (NOUN-PHRASE OBJECT
      (WORD DETERMINER "a")
      (WORD MODIFIER "constant")
      (WORD HEAD "rate")))
  (PUNCTUATION PUNCTUATION "."))
```

Figure 3.2: Syntactic analysis of the first sentence of the demo text.

of this equation. In particular, a later command of the form `\ref{vmmh}` will print out the number of this equation: 4.7.

The first step in processing the text converts it into a sequence of "tokens," each of which contains a text string or a LATEX command and its arguments. Section and paragraph breaks are recorded with special tokens. The beginnings and ends of emphasized passages and sentences are also recorded. A special parser for math-mode expressions converts the LATEX representations of formulas and equations into the form that Sagredo uses to manipulate and solve algebraic expressions. Equation 4.7 is represented internally as:

```
(EQUAL (VARIABLE "h")
       (QUOTIENT (SQUARE (VARIABLE "s" "i"))
                 (PRODUCT 2 (VARIABLE "g"))))
```

The next step is to parse each sentence. The parser for Sagredo performs a top-down search that is controlled by "parse methods" for each of the major syntactic categories. The parser constructs a representation of the syntactic analysis of the sentence in a unification grammar [272] formalism.

The result of parsing the first sentence of the text in figure 3.1 is shown in figure 3.2. The syntactic analysis is presented in a parenthesized notion where the first symbol after an open parenthesis is the name of the syntactic category of the constituents and the second symbol is the name of the constituent relative to the constituent it is inside. This is followed by a list of further constituents except in the case of the categories `word` and `formula` where the last element is a string.

So, for example, the final prepositional phrase of the sentence is analyzed as:

```
(prep-phrase at
  (word head "at")
  (noun-phrase object
    (word determiner "a")
    (word modifier "constant")
    (word head "rate")))
```

This constituent is a `prep-phrase` whose name relative to the `sentence` it is inside is 'at.' (In general, prepositional phrases are named after their head prepositions.) Its 'head' is a `word` whose string is `"at"`. This `prep-phrase` contains as its 'object' a `noun-phrase` whose 'determiner' is the `word "a"`, whose 'modifier' is the `word "constant"`, and whose 'head' is the `word "rate"`.

The sentence begins with a subordinate clause introduced by the word `"when"`. The verb group ("is projected") is in the passive voice, indicating that the subject represents the thing projected. The adjunct phrase specifies the initial direction of the motion, and the 'with' prepositional phrase associates the value of the initial speed of the motion with the formula s_i.

The 'when' clause is used to indicate the time or circumstance of the event described by the main clause. The subject of the main clause is the instantaneous speed of the object. The verb group is the single word `"decreases"` and the prepositional phrase discussed above describes the rate of decrease.

The next step involves instantiating the definitions of the words in the sentence to create "occurrence descriptions" corresponding to each clause and to specify the participants in those occurrence descriptions. In general, this is done by attaching the occurrence descriptions to the 'referent' slots of certain constituents in the syntactic analysis.

As a simple example, the definition for the word `"object"` looks like this:

```
(define-word "object" (singular-noun)
  (call (<<* (parent) referent) 'object))
```

This definition indicates that the word is a singular noun, and when it is instantiated in the analysis of a sentence, the 'referent' slot of the constituent it is inside (its "parent") will be of class `object`. So in the analysis illustrated in figure 3.2, the referent of the subject noun phrase inside the clause named 'when' will be of the class `object`.

A more complicated definition is needed for the word `"projected"`:

```
(define-word ("projected" passive) (intransitive-verb)
  (in (clause)
    (call (<< verb-group referent) 'project)
    (== (<< verb-group referent theme)
        (<< subject referent))))
```

This definition defines a specific sense, named `passive`, for this verb. The verb doesn't require any complements, so it is defined as an `intransitive-verb`. The expression `(in (clause) ...)` indicates that the rest of the forms are to be interpreted inside the clause in which this verb is used. The referent of the verb group of the clause is an occurrence of class `project`. And the "theme" (i.e., the moving object) of that occurrence will be the referent of the subject.

The occurrence class `project` is a "moment" occurrence which is followed by an interval in which the projected object is moving and is not supported:

```
(define-class project (occurrence moment)
  (call after 'interval 'motion 'unsupported)
  (== theme (<< after theme)))
```

The last line of this definition indicates that the same object that is projected is the one that will be moving afterwards. A diagrammatic representation of part of the occurrence structure created by the first clause is shown in figure 3.3.

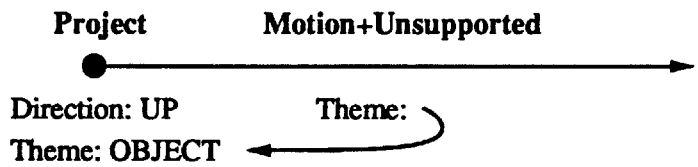


Figure 3.3: Occurrence structures created by processing "An object is projected vertically upward ..." A momentary occurrence of class `project` is followed by an interval occurrence of class `motion` and `unsupported` with the same theme.

The prepositional phrase "with an initial speed s_i" illustrates how motion-related quantities are treated in Sagredo. I noted in my studies that most such quantities are written as if they were properties or possessions of the moving object. Thus all of the following might be used to refer to an initial speed:

the object's initial speed
the initial speed of the object
the object begins moving with an initial speed of ...
the object had an initial speed of ...

Three of these constructions are typically used to describe a "possession" relation (consider: the flower's petal; the petal of the flower; the flower has a petal). The other construction ("moving with") indicates the possession relation only within a motion occurrence.

These data might suggest that the proper treatment of motion quantities might be as a property of the moving object, but there are good reasons to want to treat them as properties of the *occurrence*. For one thing, the moving object can start and stop, speed up and slow down. For another thing, a number of other quantities, such as distance and duration, do not seem to be associated with the moving body. We don't say, for example, that "an object's distance is 10 meters" to describe that the object *moved* 10 meters.

The method I use in Sagredo is to associate with each motion quantity an abstract "function application." This structure has three slots: an 'argument,' an 'occurrence' and a 'value.' For motion quantities (the only ones I deal with) the argument is the moving object, the occurrence points to the motion occurrence, and the value is the specific quantity involved, such as initial speed or average acceleration. The function application for each quantity is stored in the quantity's 'fapp' slot. Sagredo has learned that the phrase "initial speed" refers to the instantaneous speed of the moment at the start of a motion occurrence. The function application created for the phrase "with an initial speed" is shown in figure 3.4. The 'occurrence' slot points to the `project` event which was the beginning of the motion occurrence.

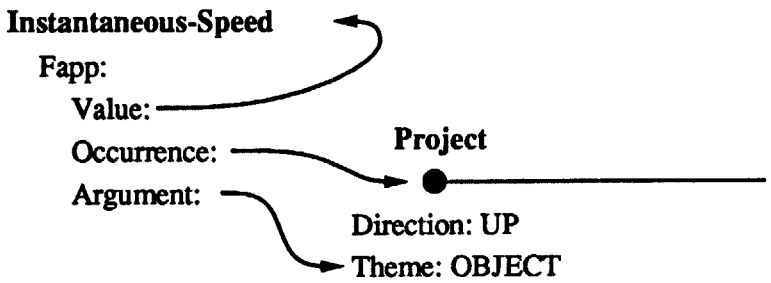


Figure 3.4: Function application created for the phrase "with an initial speed."

The subject and main verb of the main clause of the first sentence ("its instantaneous speed decreases") introduce a different kind of occurrence. Sagredo represents a `quantity-change` class of occurrence that takes as one of its participants a representation of the kind of quantity that is changing. However it has learned that when an object's instantaneous speed is changing, the object is accelerating, and furthermore it has learned that when an object's instantaneous speed changes at a constant rate, that the object has a constant acceleration. So the program creates a `motion` occurrence that has a slot named 'constant-acceleration' whose 'fapp' slot points back to the `motion` occurrence.

As the words of the sentence are processed, various "discourse events" occur. Discourse events involve manipulations of the occurrence structures that depend the details of the discourse, as opposed to the syntactic analysis of the constituents alone. An important class of such discourse events are involved with referring expressions. For example the subject of the 'when' clause is associated with a discourse event in which a new referring expression is introduced. The word "its," functioning as the determiner of the subject of the main clause, is associated with a discourse event that causes the program to locate a co-referring expression. The program finds the referent of the phrase "an object" from the first clause and sets the two referents equal.

Occurrences are also referents, and Sagredo responds to a class of discourse events associated with them. Most verb groups are associated with an occurrence. However in many cases the occurrences of a number of succeeding verb groups are identical or are related systematically. For example the 'when' clause of this sentence has an object "projected." The second sentence has an object "rising." As we have seen, the definition of the class `project` creates a motion occurrence which is the same occurrence as the one associated with the verb "rising." Sagredo responds to the discourse events associated with verb groups by attempting to find occurrences that can be "merged" with whichever one triggered the discourse event.

In this sentence, the phrase "decreases at a constant rate" creates a quantity

of the class `constant-acceleration`, as described above. The 'occurrence' slot of the function application inside this quantity is merged with the `motion` occurrence created by the verb "projected" in the first clause of this sentence.

Another important class of discourse events are those associated with formulas and equations. For example in the first sentence the pattern "an initial speed s_i" is known as an "appositive" construction. When Sagredo's parser encounters such a construction it generates a discourse event that associates the variable in the appositive position with the referent of the noun phrase it is inside. This association will be used in the subsequent algebraic reasoning guided by the text.

So, after the first sentence is processed, the program has constructed a description of a motion with constant acceleration, whose initial speed is associated with the variable s_u. I will describe the processing of the rest of the sentences a bit more schematically.

The second sentence of this paragraph is relatively simple:

> The object rises until its speed is zero.

As I already mentioned above, the occurrence associated with the verb "rises" is merged with that introduced by the verb "projected" in the first sentence. Also, a referent for the noun phrase "the object" is found. The main effect of this sentence is to create the end moment for the `motion` occurrence created by the first sentence, and to create an instance of `instantaneous-speed` whose 'occurrence' slot points to that moment. A constraint is created asserting that the value of the quantity at that moment equals zero.

The next sentence gives some further information about what happens at this moment:

> When its speed is zero, the object is at its maximum height h.

The phrase "maximum height" is interpreted by the program as giving the total distance of the motion occurrence. The appositive construction associates the variable h with this value. And the 'when' construction associates the 'occurrence' slot of the quantity "maximum height" with the same moment as the 'occurrence' slot of the instantaneous speed whose value is zero.

At this point the text has given the initial speed and the distance of motion with constant acceleration. Furthermore, the program has learned that any unsupported object accelerates downward with the constant acceleration of gravity, g. So it has all of the information needed to derive an equation relating the maximum height to the initial speed. And that is what happens next.

> Substitute h for the distance, 0 for the final speed, and $-g$ for the acceleration into equation 3.10:

The previous sentences were all about the object and its motion. This sentence is about an algebraic operation. Sagredo responds to this sentence by performing a number of operations on and with equations. The first step is to locate the equation 3.10. As I mentioned above, that equation gives the relation between the distance traveled, the acceleration, and the initial and final speeds of an object moving with constant acceleration along a straight line:

$$2ad = s_f^2 - s_i^2. \tag{3.10}$$

Since the motion described in the first sentence this passage satisfies the requirements for the application of this equation, a version of it has already been introduced by a rule that Sagredo learned when it was reading section 3. This instance is found. The sentence directs that three specific formulas be used in specific places in that equation, and then illustrates the result:

$$2(-g)h = 0 - s_i^2.$$

The next sentence:

Solve for h:

$$h = \frac{s_i^2}{2g}.$$

directs the system to transform the first equation into the second. This is done by the application of a number of algebraic transform rules. Here are some of the rules used in the illustrated transformation:

$$\begin{aligned} xy &\rightarrow yx \\ 0 - x &\rightarrow -x \\ x(yz) &\rightarrow (xy)z \\ xy = z &\rightarrow x = \frac{z}{y} \end{aligned}$$

Each rule is applied whenever its left hand side matches an equation or any subexpression of an equation. The expression that matches is replaced by an expression derived from the right hand side of the rule. In the case of 'solve-for' operations like this one, the program works breadth-first, backwards from the goal expression and forwards from the given expression.

The final step in the processing of the presentation part of this text is the recognition that the passage was stating a constraint that holds in all occurrences of the kind described.

The occurrence description that was created as the text was processed is collected into the "condition" part of a rule, and the constraint is converted into a form such that when the condition matches, the constraint will be introduced. Here is the rule that is created from this occurrence:

```
(define-rule project-1
    ((occ (call (<<) 'project)
          (call (<< direction) 'vertically-upward)))
  (call (<< occ after constant-acceleration)
        'constant-acceleration)
  (== (<< occ after constant-acceleration fapp occurrence)
      (<< occ after))
  (constraints
   (equal (<< occ distance)
          (quotient
           (square (<< occ instantaneous-speed))
           (product
            2
            (quantity 9.8 (meter) (second second))))))
```

To apply this rule on an occurrence named "occ," the occurrence must be of class `project` and its direction slot must be of the class `vertically-upward`. If these conditions are met, the rule creates a 'constant-acceleration' slot on the 'after' occurrence of the given occurrence. And it creates a constraint introducing the equation just derived.

The use of this rule will now be illustrated in the program's treatment of a problem. The first sentence of the program describes a simple occurrence:

> A stone is thrown straight upward and it rises to a height of 20 m.

The lexical entry for the verb `"thrown"` is identical to that for the verb `"projected"`, thus a `project` occurrence and its subsequent `motion` occurrence are created. The phrase "it rises to" directs the system to set the 'distance' slot of the motion occurrence.

Since the first sentence created a `project` occurrence, the rule `project-1` applies, and is instantiated, introducing a version of the equation described. This constraint contains enough information to solve the problem in the second sentence:

> With what speed was it thrown?

The word "thrown" here directs the co-reference mechanism to locate an occurrence of class `project`, this is the one introduced by the first sentence. The speed of this occurrence is the instantaneous speed of the initial event of the the motion occurrence.

The syntax of this sentence plus its punctuation are recognized by the program as introducing a "question" discourse event. The question directs that the value of a quantity be found. The algebra system is used to compute the values of desired quantities. It works by attempting to solve any constraint for which it knows enough

of the required values. Whenever a constraint can be thus solved, a table associates the quantity with its value.

If the desired quantity is not found after having run as many of the constraints as it can, the system applies the transform rules and keeps trying to solve equations. Eventually it will either solve for the desired quantity or it will run out of constraints that can be run. In this case the constraint as given can't be solve directly. However it can be transformed into a constraint that represents the following equation:

$$v_i = \sqrt{2gh}$$

and this can be solved to give the value:

```
(QUANTITY 19.8 (METER) (SECOND))
```

Chapter 4

Learning Kinematics

This chapter begins with a discussion of why kinematics is a good domain for the study of expository technical text. In the second section I explain why I had to write the source text, and discuss some of the considerations that went into writing it. I then consider the the question of what it takes in general to be able to learn kinematics. Part of what it takes is an understanding of the ways that physical concepts are expressed in language, what I call "linguistic physics." The last section of this chapter describes the linguistic physics of motion verbs.

4.1 Kinematics as a Domain

In considering a domain in which to investigate the workings of expository technical text, I kept two requirements in mind. The first was that the chosen domain be important and interesting, the second was that the domain be tractable. Kinematics is perhaps unique among the subdomains of physics in its simultaneous satisfaction of these two requirements.

Kinematics is a relatively small subject. Many textbooks contain a table like that shown in figure 4.1, presenting the five equations that are needed to do most kinematics problems. These equations are: the definitions of average speed and acceleration, the equation for the average speed of uniformly accelerated motion (equation 3.1), the equation for the distance moved by a uniformly accelerating body in term of the time taken (equation 3.5), and the equation for the distance moved by a uniformly accelerating body in terms of the initial and final velocities (equation 3.10). These equations, in turn, involve relations among only four kinds of quantities: position, time, speed and acceleration. As the previous chapter demonstrates, a great deal of the subject can be covered in a small number of pages of text.

At the same time, kinematics is fairly well circumscribed. There is very little other physics that one needs to know to understand kinematics. Indeed kinematics is often the first topic discussed in a physics text. The phenomenon of gravity might

$$
\begin{aligned}
s_{\mathrm{av}} &= d/t \\
a_{\mathrm{av}} &= \frac{s_f - s_i}{t} \\
s_{\mathrm{av}} &= s_i + \frac{s_f - s_i}{2} \\
d &= s_i t + \frac{1}{2} a t^2 \\
2ad &= s_f^2 - s_i^2
\end{aligned}
$$

Figure 4.1: The five kinematics equations.

appear to require appeal to concepts outside of kinematics because the complete treatment of gravity involves the dynamical account of forces on unsupported bodies. However, as Galileo discovered, all bodies near the surface of the earth fall with the same acceleration. So the consideration of the motion of falling bodies can be treated purely kinematically, with the precise characterization of the cause of the acceleration left for later.

Despite the subject's being compact and circumscribed, the presentation of kinematics still employs a large sample of the expository machinery of technical texts. The presentation of kinematics involves: the definitions of several kinds of quantities; derivations of formulas relating those quantities in specific classes of situations; and the discussion of a natural phenomenon (gravity). Each of these kinds of topics, and the methods for introducing and discussing them, are heavily used elsewhere in physics textbooks.

The domain of kinematics is fundamental for physics in two ways. The first is that the domain is a straightforward prerequisite to other physics domains. One can hardly present the dynamics of moving bodies, for example, without using the background of kinematics to describe their motion. Other domains of physics describe specific circumstances of bodies in various kinds of motion, for example when under the influence of electrostatic or magnetic forces.

The other way that kinematics is fundamental for physics is that kinematical ideas provide a very central metaphor for other physical phenomenon. This ranges from the barely metaphorical relations between the kinematical equations of straight line motion and rotary motion, to the cases such as heat and electric current "flow." In the later cases the the equations are often similar, but describe qualitatively different phenomenon. Still more metaphorical applications of kinematical notions are found in ordinary language (as described by Lakoff [197]), and are used also in physics: for example in talk of temperatures "rising," or a chemical solution "reaching" equilib-

rium, and so forth. The degree to which these metaphors rest on actual kinematic theory varies, but it seems plausible that some of the same sort of understanding needed to treat these other domains is needed to understand kinematics.

Another reason why kinematics is a good domain for a computational natural language system is that the relevant linguistic theory is becoming fairly well developed. Understanding the presentation texts and problems in a discussion of kinematics requires attention to the syntax and semantics of the linguistic means for expressing position and motion. A number of proposals have been made about how temporal expressions ought to be dealt with, and even specific proposals about the proper treatment of the verbs of motion have been made. (These are discussed in chapter 11.) Just as kinematics as a physics subdomain is foundational to other areas of physics, the linguistics of the verbs of motion and position may be foundational to other areas of linguistics.

Some of the relevant linguistic work has involved treating connected discourse. Since motion involves change in position, the sequence of sentences which describe some motion events must deal with the temporal relations of the described events. This is typically done with anaphora and with verb tense and aspect, and with relations among these properties across sentence boundaries. This general issue has received some attention in the literature, enough so that I think that the relatively simple situations that come up in kinematics presentations and problems can be handled.

This discussion of the ways that kinematics is a good domain for this investigation may have made the case too strongly. It may very well be that kinematics is atypical precisely in ways that make it simple enough to approach computationally. Whether this is true or not certainly has to await attempts to apply an approach like that described here to other domains of science. I discuss how that might be done, and what other domains might succumb, in chapter 9. I take up the question of whether physics itself might be an artificially well-behaved, and therefore misleading, domain for understanding the nature of language in chapter 10.

4.2 Why I Had to Write the Source Text

I would have liked to use passages from an existing physics textbook as the source text for Sagredo's adventures. Indeed for a while I worked with the details of the presentation in the *Schaum's Outline Series* text on college physics [43]. (Where most of kinematics is covered in two pages, by the way.) However it became clear that this was not a reasonable or interesting goal.

The main reason is perhaps obvious: physics textbooks, for all their dryness and pedantry, are written by human beings for human beings to read. An attempt is made to make the experience of reading them at least tolerable. This means that authors will attempt to avoid dull repetition, to vary linguistic constructions, to use anaphora

and ellipsis when possible. Authors might even attempt to inject some personality into the text, even some wit. Many of such techniques to make a text more interesting in these ways make it more difficult to deal with computationally. If an author uses a rare grammatical construction to make a point at one place in the text, the student can be expected to puzzle it out; perhaps the obscurity of the construction calling attention to the technical point being made. But a program is unlikely to be able to figure it out unless unless the circumstance has been foreseen, and the attempt on the part of the programmer to make the program capable of dealing with one recondite expression will likely not help with the next one.

Textbooks can sometimes be careless in their organization and argument. Topics may be presented a bit out of order, especially early in the book. I have found at least one place where a textbook's reasoning is simply invalid (see chapter 10). But such textbooks can still be useful for people because people can patch things a bit as they read to make sense of what is going on. Important passages are often read more than once, and a single textbook will not the sole source of the student's understanding. But the program that I was developing was to go through the text once, with the earlier passages preparing it to deal with the later ones, and so it was important that the presentation be as cumulative as possible.

Textbooks contain a lot of extraneous material that has nothing to do with solving problems. Many books contain biographies of famous physicists or discussions of everyday phenomenon that illustrate the physics topic being described. While these are interesting for students and may introduce them to some of the shared culture of their field, a program couldn't really do much more with them than to just recognize that the passage is irrelevant and move on. Physics textbooks often begin or motivate discussions of certain topics with appeals to the physical experiences or intuitions of the reader. Having no such experiences, and having only the "intuitions" that are build into it, a computer can't get much out of these passages.

The reasons just mentioned involve ways that the program couldn't handle aspects of existing physics texts because it doesn't have all of the abilities or interests that a human does. The following are some reasons why I didn't use some other aspects of texts. In general there is no reason in principle that a program couldn't use these aspects, it is just that I chose not to.

Many textbooks make extensive use of figures and tables and graphs. These aspects of expository texts are obviously very important and any adequate account of technical exposition is going to have to deal with them. (Some work on the uses of illustrations is described in [201].) I spent some time considering how diagrams were used in presentations and problems, but I decided that, in the cases I understood, the same work could be done with language. This is obviously not true in general, but as my primary interest is in language, I decided to forego attention to illustrations.

Calculus is crucial to a complete understanding of physics. However it turns out that kinematics can be presented with no calculus (as, in fact, I did in the source text)

provided that the equation relating the average speed of a body moving with constant acceleration to its initial and final speed (equation 3.1) is simply presented as fact, with no derivation. I chose not to deal with calculus simply because the interesting aspects of dealing with the exposition could be illustrated without it, and because kinematics could be presented without it. Clearly to extend this work to any other domains of physics (or even to a more complete understanding of kinematics) would require that the system be competent in calculus. And there is no reason to expect that this should present any especially difficult problem, given that many powerful computational systems for reasoning with calculus exist.

The source text I wrote works by first presenting a general situation in which physical quantities are defined. Special cases are described using slight specializations of those situations, and problems are presented as further specializations. This organization makes the "story subsumption" problem solving method that Sagredo uses work. But some textbooks use an approach based on examples. In such a text, an example is presented first, perhaps the motion of balls on inclined planes, or rocket ships. The equations are derived in the context of the example, often with illustrative values for the relevant quantities. The reader is then left to figure out what about the example was really central to the physics and should be applied to other examples or to problems. This sort of "example-based" learning is an interesting topic and had many books used it I would have been forced to deal with it, perhaps using an approach based on the "explanation based generalization" work described by Mitchell, et al. [220]. However the general-to-specific presentation pattern that I used is in fact the most common.

And finally, a few textbooks are actually too *easy*. The *Schaum's Outline* text, for example, doesn't derive the kinematics equations at all, but simply lists them. A few special cases are treated in examples.

I wrote the kinematics source text with these goals in mind:

1. To cover the bulk of kinematics. In particular, to provide the background needed to be able to use the 5 kinematics equations plus the equations for vertical motion.

2. To present a generous sample of the linguistic resources that expository texts exhibit. In many ways, attention to goal 1 assured that this would be satisfied. One simply can't explain kinematics adequately without exercising a lot of expository devices.

3. To be able to solve a large number of physics problems obtained from textbooks. In particular, this meant handling the linguistic constructions of example problems obtained from existing texts.

4. To include derivations of all of the major equations. "Following" a derivation is an important part of reading technical text and being able to derive equations

for particular situations is an important part of the actual use of technical knowledge.

5. To write a textbook that could conceivably be useful to a student. I don't know if I succeeded at this, and I am not sure, except by recording sales figures, how textbooks are assessed. In particular I don't know if the attention I had to pay to clarity and uniformity of style would make it easier or harder for a student to follow my text.

I first wrote the text as if I were writing my own version of a college physics textbook. As I had examined twenty or so existing texts in the early phases of this research, that was a relatively straightforward process. Although I was writing with the intention that the program would read the text, the first draft was not I think very influenced by that intention.

However when I began to run the program on it, I discovered that my text had many of the problems that I discussed above with regards to existing physics textbooks. In the course of implementing the program I modified the text in various ways. The most important modification was to make the text more grammatically uniform: if the "same thing" was being done at more than one place in the text, it would be done the same grammatical way. That is why there are so many instances of the phrase "an object moving along a straight line" and "If ..., its ...is defined to be" and so forth. In general I had to choose which linguistic issues I felt like handling and which I could avoid. The specific set of issues for which I came up with operational solutions is described in chapter 7.

As noted in the source text, many of the problems that the system can solve were taken verbatim from existing physics texts. But of course I had some choice there also: there are so many textbooks in the world, and each of them contains so many problems that it is an easy matter to find ones that illustrate whatever principles are desired and avoid the rest. Most of the problems that I chose were taken from early in the list of problems at the ends of sections and chapters. Later problems require more subtle understanding of the physical principles, or more background knowledge, or more sophisticated mathematics. Some of the kinds of problems that the program can't deal with are described in chapter 9.

4.3 What Does it Take to Learn Kinematics?

The first point to make in response to this question is that, for people, learning is not a one-shot affair. The student comes to the textbook, or the physics lecture, with some understanding of the phenomenon, which is then modified by the experience. A college physics course involves repeated treatment of the same material, in lectures, perhaps in discussion sections, in auxiliary texts, and in the main textbook for a course. Later episodes of learning some material draw upon previous understanding

in a cycle of accretion. So part of what it takes to learn kinematics is: to know kinematics.

Part of that knowledge of physics is based on a person's experience with the physical world. It is hard to imagine a person mastering physics without some prior experience of motion or force or gravity. On the other hand, it is hard to imagine a person without such experience period. To some degree this experiential understanding manifests itself in "knowledge" about the physical world that can be put into words, for example, that unsupported objects fall, and that a heavy object is harder to move than a light one. The experiential understanding is also manifest in general expectations or intuitions or bodily reflexes that accompany physical activity.

It may be that there is some sort of more or less universal "naive" theory of physics that students bring to physics classes. It may be that this theory is presumed in the treatment given the material by lectures and textbooks. (For some discussion of the nature and role of such a naive understanding of physics, see [146, 147] and the papers in [31] and [160].) From what I have been able to tell by reading a large number of physics texts, however, the content of this naive understanding of physics is extremely limited, and, as I argue in chapter 10, the attitude that textbooks take toward the intuitions and everyday experience of the student is ambivalent at best. The point of the books is to teach a new way of thinking, complete with a new set of intuitions.

What the textbooks rely on instead is what might be called "linguistic physics"—the ways that physical concepts find their expression in language. As a very simple example, understanding the sentence "the bus went from the school to the lake in three hours" requires an understanding that bodies occupy positions, that they can occupy different positions at different times, that motion involves change of position, and that movement can take time. Given that much of our concern is with the physical world, it is not surprising that language should have rich means for describing it. And I argue that most of the pre-understanding of physics that students bring to class is embodied in their knowledge of and ability to use language.

Thus in asking what it takes to learn kinematics, one must consider what what one understand about *language* in order to learn kinematics. This question directly motivates my approach because I found that most of the details involved in getting a program to read the text in chapter 2 and to solve the problems involved details of language, not of physics. It turns out that Sagredo needs very little of what might be called "naive physics" in order to learn kinematics, and all of its "knowledge" is embedded in the definitions of words and of grammatical and discourse-level rules.

Physics textbooks also rely on straightforward mathematical skills in their presentations. One must be able to follow an algebraic derivation, to be able to perform one, to be able to solve an equation for a given value, and to understand how to perform arithmetic on quantities. I have built all of this mechanism into Sagredo, but in general, the process of learning to do mathematical reasoning of the sort that is needed to learn physics is a very interesting and important topic. I should point out

that I spent some time looking the presentation styles in algebra and calculus books. The difference is that in physics the underlying domain is simple and uniform, namely bodies and positions and motions. In mathematics texts the underlying domains come from all aspects of life. Much more knowledge of the everyday and social (and even physical) world are needed to understand problems about such topics as: per-hour wages, compound interest, the heights of shadows, relative ages, and so forth.

Finally, in order to learn kinematics, indeed to learn physics at all, the student must, in some sense, be willing to *succumb* to it. Physics books are written in a characteristic genre, with a characteristic authority, they demand that the student view the world (or at least a fraction of it) as thinly disguised equations. The student must be willing to ignore the fact that the horse pulling the canal boat may be bored, or that the person carrying the heavy block may be getting tired, despite not doing any "work." The decision to accept physics is probably one of the largest hurdles that must be overcome in order to learn it.

4.4 Linguistic Physics

At the beginning of his discussion of kinematics, Galileo presents the following definition:

> By steady or uniform motion, I mean one in which the distances traversed by the moving particle during any equal intervals of time, are themselves equal.[1]

This sentence illustrates a large fraction of the linguistic foundation needed for an understanding of kinematics.

Galileo's use of the word "one" refers to an instance of the class he is defining ("steady or uniform motion"). It is perhaps a significant fact in itself that he is able to refer to a situation of motion as a singular thing. By thus "objectifying" motion situations, Galileo can use the linguistic methods for modifying referential noun phrases to characterize the particular subclass he is interested in.

Notice that Galileo uses the phrase "the moving particle." This is a definite noun phrase (introduced, as it is, by the determiner "the") and therefore should refer to some participant that has been already introduced into the discussion. But no such participant has been explicitly introduced. What has been introduced however, is a motion situation, and perhaps the most important participant in a motion situation is the moving body. (Linguists call the semantic role of the moving body, the *theme*, from the Greek word which means "that which is set down.") It hardly makes sense to speak of a motion situation without speaking of there being something that is moving. So Galileo can refer to it with a definite noun phrase. This understanding is

[1] Galileo, *Two New Sciences*, p. 191.

a clear example of what I mean by "linguistic physics." For anyone to make even the most simple sense of what Galileo is saying, one must be able to understand that the referent of "the moving particle" is the theme of the previously introduced motion situation.

Galileo also uses a definite noun phrase when he mentions "the distances traversed." That motion involves "traversing" or "covering" distance is another fundamental fact about physics that seems to be embedded in language. In particular, many of the verbs of motion take as their direct object a phrase which must refer to the distance traveled. For example: "The rock moved 10 feet."

Finally, the distances are traversed "during ... intervals of time." As I mentioned above, our linguistic understanding of physics includes the understanding that temporal modifiers may be applied to expressions denoting motion situations, and the meaning of those modifiers is to indicate the amount of time the motion took.

This understanding involves the recognition that there are characteristic kinds of situations or events or occurrences, that are related in specific ways, and involve certain participants or roles. This is one of the observations on which the linguistic theory of "case frames" is based [99]. The other basis for case frame theory is that syntactic regularities are often best expressed in terms of these occurrences and their roles. A verb is described in terms of the occurrence that it introduces into the description, and how its syntactic arguments (i.e., subject and direct object) are mapped to the participant roles.

The central verb of motion is, of course, "move." Other verbs, discussed below, are related to "move" in characteristic ways. In its straightforward sense of movement from place to place, "move" has three main arguments, illustrated by the following sentences:

The rock moved.
The truck moved the rock.
The rock moved 10 feet.
The truck moved the rock 10 feet.

In the first sentence only the theme of the motion situation is expressed, in this case as the subject of the sentence. In the second sentence, an "agent" is expressed as the subject, the theme of the sentence is the direct object of "moved." In the third sentence the theme is the subject and the distance moved is the direct object. In the fourth sentence all of the arguments are expressed: the subject is the agent, the indirect object is the theme, and the direct object is the distance traveled.

There are a number of additional participants or arguments in motion situation. These are usually expressed with prepositional phrases or adverbs:

Locations Motion situations are often described by indicating the initial or the final location or both. The standard way to do this is with the prepositions "from"

and "to:" "They moved the sofa from the living room to the den." Depending on the specific motion verb, the prepositions can be sometimes dropped, for example: "they went home," "they left home."

Duration As mentioned above, this is often expressed with the prepositions "in" or "for" with an object that describes a time period. In general, the preposition "in" is used when the motion involves a specific distance traveled, as in "she ran 100 meters in 13 seconds," where "for" is used when no such goal is expressed: "they drove for three hours."

The duration of a motion (or other kind of interval) is sometimes expressed as the indirect object of the verb "take," as in "It took 3 seconds to stop."

Rate This is specified with "with" or sometimes "at:" "running with a speed of 10 mi/hr," "growing at a rate of 1 inch per year." "With" and "at" are also used to introduce the acceleration of the motion, in general, the actual role filled by the object of the preposition depends on the type of the quantity. For example the following makes sense, though would probably not be found in a textbook: "At $t = 0$, the body moves with a speed of 15 miles per hour and with an acceleration of 20 feet per second per second."

Path This is often expressed with the preposition "along." Sometimes the preposition "on" is used, when its object is a path. For example, in the sentence "driving on a straight road" the motion is understood to be *along* the road, in addition to being supported by it. For example if the sentence were "driving across the road," the motion is still *on* the road, but is now directed perpendicular to the main direction of the road.

Trajectory/Target/Direction These usually specifies the initial path of the motion, especially when it is clear that not all of the motion will be in the same direction. Sometimes the trajectory is the same as the "to" argument: "They headed home." Sometimes it is a direction: "The ball was thrown straight up." Also the prepositions "towards" "at" "into" "away from" and others can be used.

Support/Container/Conveyance There are problems and examples where one object is supported by another, or is dropped from some object or is carried by another object. These are expressed with "on" or "in," or with a verb like "attached to" or "supports." If the support is itself a moving object, a "conveyance," the supported thing is also moving.

Manner The "manner" of the motion is often expressed with an adverb, as in "The ducks flew erratically," though "with" can be used: "The dog cycled with aplomb." The most important use of this in elementary physics is where the object moves "uniformly" or accelerates "with a constant rate."

Other verbs can be understood in terms of how the motion situation they describe is related to those described by "move:"

> Some verbs seem to be identical for the purposes of this analysis with the intransitive version of move. They can take an agent argument only when the agent and the theme are the same: "travel," "go."
>
> A large number of verbs can be considered to be similar to move, except they they imply that the motion is of a particular kind: "walk," "run," "roll," "fly," "drive." Some verbs contain their "direction" argument: "fall," "rise."
>
> An important set of verbs describe the start or end of a motion situation. For example all of the following are events that begin a projectile motion: "fire," "shoot," "propel," "throw." The verb "drop" begins a downward motion. The following verbs describe the end of a motion situation: "catch," "hit," "strike." The verbs "come," "go," "arrive," "leave," "reach," and so forth characterize an action that either starts or ends at a particular location.

Additional discussion of the linguistics of the verbs of motion is found in [7] and [204]. Sagredo's incorporation of this information is straightforward: associated with each word are a set of "unification forms" that are interpreted whenever that word is used in a sentence. These unification forms can directly specify the class of the described occurrence and can determine how the referents of the various grammatical constituents are used as participants in the described situations.

Chapter 5

A Model of Expository Discourse

This chapter develops the model of expository discourse that is the basis for the reasoning done by Sagredo. The model provides a descriptive vocabulary that is used when analyzing and responding to passages of text. It provides the means to describe the relationship between the sequence of characters that compose the text and the higher-level structures imposed on the text in the course of its interpretation. The model is also used to describe that which the text describes: the semantic domain of moving and falling objects. The model allows the expression of the relations between these two domains. And finally, the model is used to express the fact that some of the descriptions in the text are of general physical facts and laws, and that those descriptions and statements are used to solve problems about specific situations.

The first section of this chapter summarizes what I learned by studying the presentations in physics texts. I argue that most of the expository work in such texts is done with simple "stories," and that the logical relation between the presentation texts and the problem texts is a kind of subsumption. I then discuss the very simple kind of reasoning that I call "structural deduction" in which the analysis is based on annotating the input token sequence with description structures. These descriptions can capture a shallow level of semantic description which is nevertheless sufficient for solving a large variety of physics problems. The third section of this chapter describes the main representational entity used in the model, the "occurrence," and explains how occurrences are used to record what is learned about physics. The last section describes how the response of the program to the text is organized around "discourse events."

5.1 Stories and Their Uses

I begin with the observation that a great deal of the work in technical texts is done by first telling a story. The story may be very simple, barely more than a sentence describing a situation, or it may involve a complex sequence of related situations and

events.

The following story is from Galileo:

> Imagine any particle projected along a horizontal plane without friction; then we know, from what has been more fully explained in the preceding pages, that this particle will move along this same plane with a motion which is uniform and perpetual, provided the plane has no limits. But if the plane is limited and elevated, then the moving particle, which we imagine to be a heavy one, will on passing over the edge of the plane acquire, in addition to its previous uniform and perpetual motion, a downward propensity due to its own weight; so that the resulting motion which I call projection, is compounded of one which is uniform and horizontal and of another which is vertical and naturally accelerated.[1]

This passage illustrates how a simple situation is (a particle projected along a horizontal plane) is presented and then elaborated. The elaboration refers back to previously discussed material. It is a typical expository move to consider how a situation will change when various parameters are altered. The original situation where "the plane has no limits," is modified to one where "the plane is limited and elevated." This new situation, in turn, poses a narrative question: what will happen when the particle reaches the edge? In answering that question, and describing the subsequent situation, the text also defines a new term.

Here is another example. As the story is told, specific physical quantities and variables are introduced and then used as an example in the definition of new terms.

> Suppose a motorist enters a super highway in city A and heads south for city B, a distance of 180 miles. If he covers the distance in three hours, his average velocity v_{av} is obtained by dividing the displacement d, by the time t or
>
> $$v_{\text{av}} = \frac{d}{t}$$
>
> $$v_{\text{av}} = \frac{180 \text{ mi}}{3 \text{ hr}} = 60\frac{\text{mi}}{\text{hr}}$$
>
> If the motorist maintained a constant speed of 60 mi/hr for the entire distance, then his motion is described as *uniform motion.*[2]

In this example there are two stories. The first is a story about a motorist and a super highway. The second is a story about how a particular physical quantity (in this case the average velocity) may be found. The second story is told twice: first in words

[1] Galileo, *Two New Sciences*, p. 268

[2] Weissman, pp. 41–42

("is obtained by dividing ..."), and second in the form of mathematical equations illustrating what the words have described. Finally the text adds a new detail to the original story in the form of a conditional, and defines a new term that holds in the conditional situation.

Stories are also used to pose problems. The following one is a bit more interesting than most, furthermore it illustrates how each of the succeeding events in the story can be used to test the understanding of a different physical principle.

> A fifty-lb. monkey with downcast eyes has a firm hold on a light rope that passes over a frictionless pulley and is attached to a fifty-lb. bunch of bananas. The monkey happens to glance upward, sees the bananas, and starts to climb the rope to get at the bananas.
> (a) As he climbs, do the bananas move up, down, or remain at rest?
> (b) As he climbs, does the distance between him and the bananas decrease, increase, or remain constant?
> (c) The monkey releases his hold on the rope. What about the distance between him and the bananas while he is falling?
> (d) Before he reaches to ground, he grabs the rope to stay his fall. What do the bananas do?[3]

The mention of the monkey's "downcast eyes" is there I think to explain why the obviously hungry creature isn't in motion at the very beginning of the episode. This allows the description of the elaborate initial situation to be performed in the present tense. As in many stories, an otherwise random or unexpected event ("the monkey happens to glance upward") initiates the action. Each of the labeled questions concerns a subsequent event, and I note that the final situation of this story is essentially identical to the initial one, except that the monkey's eyes may betray a slightly different expression.

Though these tiny texts are very simple and not very entertaining, I think it makes sense to speak of them as "stories." There are two reasons for this, the first is semantic, the second is structural.

The "semantic" reason is that these texts are about what stories are about: characters and/or participants, situations, and a sequence of events. The "structural" reason for calling them stories is that the texts obey the general conventions for story telling. Earlier events cause or influence later ones. Roughly speaking, the sequence of sentences corresponds to the sequence of events the story is about. Noun phrases introduce or pick out certain participants in the action and verb phrases describe the participants and what happens to them.

[3]SZY 5.55, p. 88. Ans: (a) Up (b) Constant (c) Constant (d) Stop

When an object	A bus
moves	is traveling
along a straight line	at a constant speed
with a constant speed	of 30 mi/hr
v	on a straight road.
for a time t,	How far
it will go	does it go
a distance $v t$.	in 5 minutes?

Figure 5.1: Story Subsumption. The story used to present some new physics (on the left) subsumes the story used to pose a problem (on the right).

Once the story is told, the physics gets done. This may involve defining a new term, or deriving some special case introduced by the story, or posing a problem. This involves using the situations and participants described in the story in specific ways. Sometimes the situation as initially described can be used directly to define a quantity, as in the case of "average velocity" above. In other cases the initial situation is elaborated with some new situation or phenomenon or event.

Problems then test the understanding of these presentation passages in various ways. In this project I have been exploring the simplest such way. I call this relation "story subsumption" because it is based on the idea that the story told in the presentation text is more general, and therefore *subsumes* the story told in the problem text. Whenever this is the case, any physical principles that applied in the presentation text's story also apply in the problem text's story.

A very simple example of this is presented in figure 5.1. The left hand column presents some new physics and the right hand text presents a problem testing the new material. In this case the subsumption is almost a matter of lining up the rows of the columns. Everything matches exactly, except that it must be understood that a bus is a kind of object, and that if one moves on a straight road, then one is moving in a straight line.

Determining that such subsumption relations hold can often be accomplished with relatively simple reasoning. As I mentioned in chapter 1, I believe that the reason-

ing requires very little more than an appropriate representation of linguistic "case frames" because that is the linguistic level that deals actions and events and their participants. Understanding the subsumption relation also requires the ability to collect the contributions of a sequence of sentences into a single coherent story. The next few sections describe how those tasks are performed by Sagredo.

5.2 Structural Deduction

A physics student, upon opening a textbook, faces a pattern of ink on a page. Page after page, hundreds of them, each one containing a different pattern of ink. And from this is physics learned.

A pattern of ink, as such, is not of much use. From the pages of patterns, the student must somehow obtain definitions of physics terms, statements of laws, and derivations of formulas. Some patterns must be recognized as requiring a particular response: a problem to be solved, a derivation to check, a formula to memorize. This is fundamentally a process of *interpretation*, guided and authorized by the rules and conventions of orthography, language, and the specific genre of expository technical text. In general this process makes use of the complete human ability to interpret and to act based on interpretation. And I am sure that a final account of even physics textbooks would illustrate how arbitrarily profound aspects of human nature are needed understand physics texts and respond to them appropriately. I discuss some of the more obvious examples of this in chapter 10.

Sagredo, fettered by my limitations as a programmer, cannot fully participate in this interpretation process. However it turns out that a great deal of what is needed to pick up enough from a passage of text to be able to solve problems based on that passage can be accomplished with a fairly simple mechanism, organized around a notion I call *structural deduction.*

Structural deduction is based on representing a text as a computational data structure. The analysis of the text at any point consists of a data structure whose components correspond to the components of the analysis. At the lowest level the the text can be described as a sequence of character tokens.[4] Subsequent levels of descriptions involve annotations of the token sequence with further structural configurations. All reasoning about the text involves computational rules that test for the

[4]While there might be occasions to delve more deeply into the physical instantiation of a text (for example, when considering handwritten material), I will presume that the token sequence alone ultimately determines what is available in the text. In particular, a computer typesetting system, like LATEX, the one used to format the source text in chapter 2, takes as input a file consisting of a sequence of ASCII characters, and it is only that sequence of characters that determines the final appearance of the page. While this might not be completely true (who knows, for example, whether a fortuitous alignment of words might elicit some fertile intuition), it seems like a reasonable simplifying assumption.

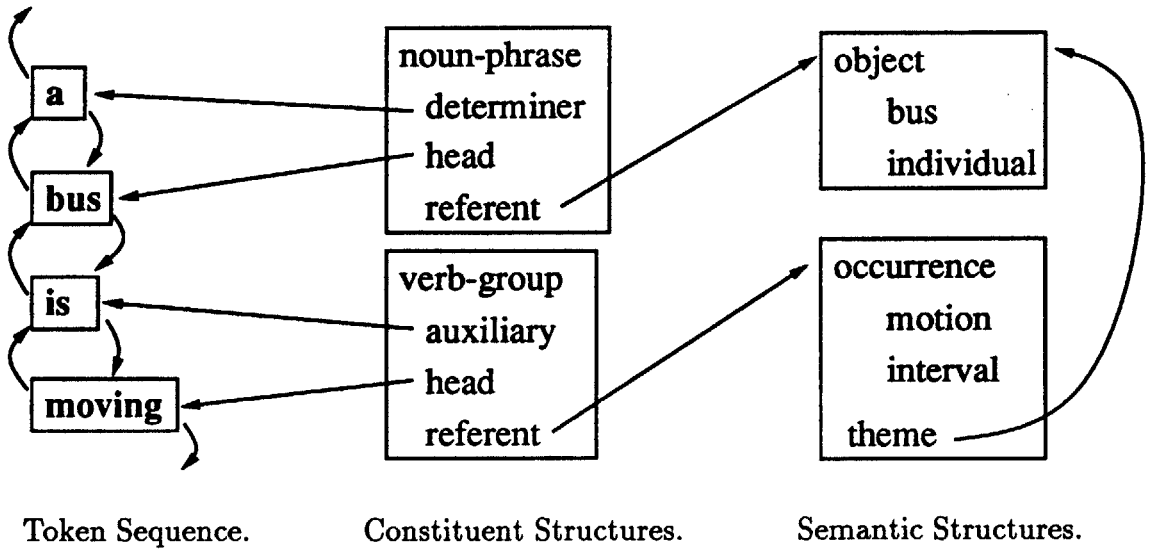


Figure 5.2: Structures deduced for a fragment of text.

presence of some structural configuration, and, if the test is satisfied, will add some additional structure to the evolving description of the text.

This process is illustrated in figure 5.2. On the left is the the original sequence of tokens, each one of them a very simple structure that records only its string, and the token before it and the token after it in the sequence. Rules that operate on this level of representation will compare the strings associated with the tokens in the sequence to those of entries in a lexicon, and check that the local sequences of tokens satisfy certain requirements. If so, "constituent structures" are created that represent the level of grammatical analysis.

The structures that represent the level of grammatical constituents are more complex than those that record the initial token sequence. They may contain several levels of structure, including, at the lowest level, pointers to the original tokens. Rules may then operate that check for configurations of representations at this level. The rules may simultaneously check for the sequencing information that is available at the level of the tokens and the types of the grammatical structures obtained from them.

The processing of the constituent structure will create instances of structures at the semantic level. For the most part, these structures occupy the 'referent' slots of grammatical constituents. But from the point of view of the program, there is nothing special about these structures—they are just further structural annotations of the input token sequence. Later rules will treat them specially, and in particular, the rules that are involved with solving problems will treat the relations among the slots of the structures in the semantic domain.

Further reasoning about the text then depends on these semantic structures. They

are grouped into other high-level structures representing the discourse and the uses to which discourse is put. The point is that all of the reasoning is about structures: there is no special distinction made between one structure being semantic and another being syntactic. The rules that direct the response of the system look at structures and do what they do based on whether the structures satisfy certain predicates. One feature of this approach is that the means are in place for describing the sorts of relations that are needed to deal with discourse properly, such as how the position of a particular sentence in a text determines how the referring expressions of that sentence are related to those of the rest of the text.

The idea of structural deduction is consistent with an approach to linguistic description based on unification. In such an approach, the description of the various levels of linguistic structuring involves solving a set of unification equations. It is a fairly simple matter to add any additional levels desired to the basic framework. Each additional level is viewed as a "projection" onto another domain, but the implementation of the idea involves simply annotating additional structure onto what is already there. The idea of projections into the semantic domain is described in Halvorsen and Kaplan, [139] and Fenstad, et al. [96] and a straightforward implementation of the idea is described in Shieber [272], section 4.3.

As will be discussed below, I have added what might be thought of as a "projection" into the discourse domain, in the guise of what I call "discourse events." The response of the program is organized around these discourse events. But the level of discourse events is just an additional level of structural annotation that begins with the lowest level token sequence.

5.3 Occurrences

As was already illustrated in figure 5.2, the semantic domain of Sagredo is populated by structured objects serving as the referents of the grammatical constituents of the sentences of the source text. These objects are instances of elements of a type hierarchy, and may contain other objects as named constituents. There is nothing mysterious about their being the referents of the grammatical constituents, by the way: those grammatical constituents that have referents have a slot named 'referent' and the process of structural deduction fills that slot with an appropriate object in response to grammatical rules. The "semantic domain" for Sagredo is simply the set of objects in the 'referent' slots of grammatical constituents.

The most important class of object in the semantic domain is that of the "occurrence." In general, an occurrence is the referent of a verb-group. An occurrences describe something that happens or takes place, or *occurs*. I chose the name "occurrence" because the names "situation" and "event" and others were already taken. The name is meant to be neutral with respect to the time duration of what happens, and also with respect to any internal details, such as whether it describes a static

state of affairs, or an ongoing process, or a complex structured action. All of these can be described by different occurrence classes.

Various kinds of occurrence classes are defined, and different classes of occurrence take different classes of objects in their constituent slots. For the kinematics domain the most important class of occurrence descriptions are those that describe motion, the occurrence class `motion` contains a number of participant slots: the moving body, or the "theme," the distance traveled, the time duration of the motion, the trajectory, and other slots (which will be described in chapter 7). Occurrence classes can be "mixed together" so long as the classes are compatible. For example the occurrence class `support` is defined, with two main participants: the 'supported' thing and the 'supporter' thing. A class 'supported-motion' can be defined as the combination of `motion` and `support` such that the 'theme' of the `motion` occurrence is the same as the 'supported' slot of the `support` occurrence.

A simple temporal model is used for occurrences. Time is viewed as consisting of a totally ordered set of *moments*. An *interval* is an ordered pair of moments, with the first moment in the interval before or equal to the second moment in the interval. The difference between the 'begin' and 'end' moments of an interval is always a real number. There are reasons to suspect that such a simple temporal model is not adequate for all of natural language. But it is adequate for (and, arguably, was invented for) Newtonian physics, and hence is adequate for Sagredo.

The temporal relations among occurrences are expressed by the contents of specific slots. A "moment occurrence" contains a 'time' slot that contains a representation of a moment. An "interval occurrence" contains two slots, 'start' and 'end', each of which contains a moment occurrence, such that the moment that is in the 'time' slot of the 'start' slot must be before the moment that is in the 'end' slot, and therefore the two moments define a temporal interval. Two interval occurrences may share a moment occurrence: a common case is where the 'end' moment of one occurrence is the same as the 'start' moment of another occurrence. The same occurrence class can be mixed with either interval or moment to yield two different classes which are nevertheless related in useful ways. In particular, the 'motion+interval' class is used to represent an extended interval during which a body is in motion. Each moment of such a interval is represented with a 'motion+moment' class.

Occurrences capture a very simple, "shallow" semantics. Essentially they capture the sort of semantics that linguistic "case frames" capture, and in many ways occurrences can be thought as case frames, except that the latter usually do not carry temporal information. Like case frames, occurrence structures can be easily computed from the grammatical structure of sentences. For the most part, the main verb of a sentence specifies the type of occurrence that will be its referent, and also indicates how the referents of the objects filling the grammatical functions in the clause that the verb heads are assigned to slots in the case frame. For example the simplest, intransitive sense of the verb "move," as used in the sentence "The truck is moving" would indicate that the referent of the subject of the clause in which the

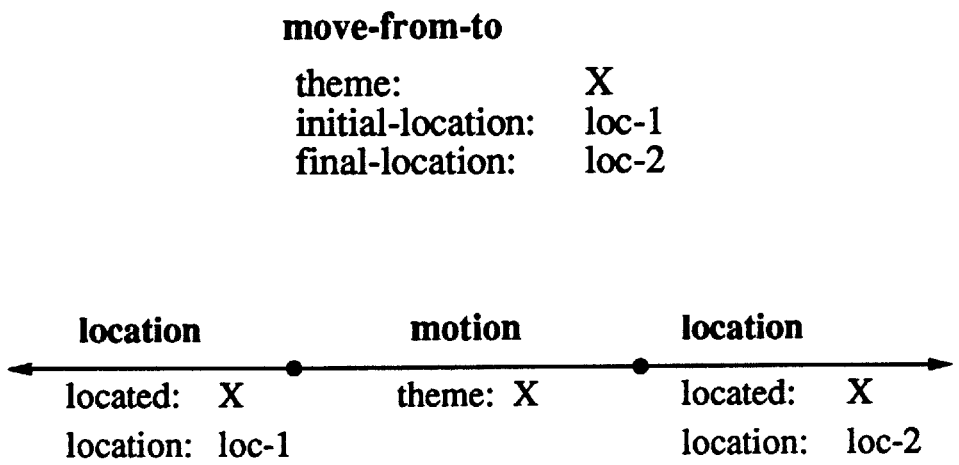


Figure 5.3: The definition of a complex occurrence as composed of a sequence of simpler occurrences.

verb appears, in this case "the truck," would occupy the 'theme' slot of the `motion` occurrence which will be the referent of the verb group "is moving."

Complex occurrence classes are defined as combinations of simpler classes of occurrences, with indications of how the participant and circumstance slots are related. For example figure 5.3 illustrates how an occurrence of a class named 'move-from-to' is defined as a sequence of occurrences of different types. The class `move-from-to` is used to represent the sense of the verb 'move' in a sentence like "The bus went from the school to the lake" where an object that initially occupies one location changes position to another location. This is represented by three interval occurrences. The first, of class `location`, has a 'located' participant, and a an object representing the location. The second occurrence is of the simple `motion` class. The use of the same variable name indicates that the same object that was the 'located' participant in the `location` occurrence is now the 'theme' of the `motion` occurrence. The dot in the figure is meant to illustrate that the final moment of the first `location` occurrence is the same as the initial moment of the `motion` occurrence. The final occurrence of the `move-from-to` occurrence definition is another `location` occurrence, again with the same 'located' participant that was the 'theme' of the `motion` occurrence, and a new 'location' argument.

Rules for reasoning about occurrences can be written that elaborate an occurrence structure with some additional structure depending on the classes of occurrences involved and on the classes and relations among the occurrences' participants. The simplest case of this involves the definition of the class, as illustrated above in the definition of the `move-from-to` class in terms of the `location` and `motion` classes.

Other sorts of elaborations involve the knowledge of "linguistic physics" discussed in the last chapter. For example the occurrence class `throw` is a moment occurrence which is elaborated such that it is the 'start' moment of an class defined as a combination of `motion` and `unsupported`. Sagredo learns from section 4 of the kinematics text that such an interval also has a downward acceleration due to gravity. All of the reasoning for such inferences involves simple elaboration of the initial occurrence structures created by the grammatical constituents.

Some of the slots of occurrences represent physical quantities, like the 'duration' slot, which represents a time period, or the 'distance' slot which represents a length, and so forth. Objects of the class `quantity` are used to represent physical quantities, they consist of a numerical value and a structured 'unit' slot that depends on the dimension of the quantity. For example in "5 miles per hour" the value is '5' and the unit is represented as `((MILE) (HOUR))`.

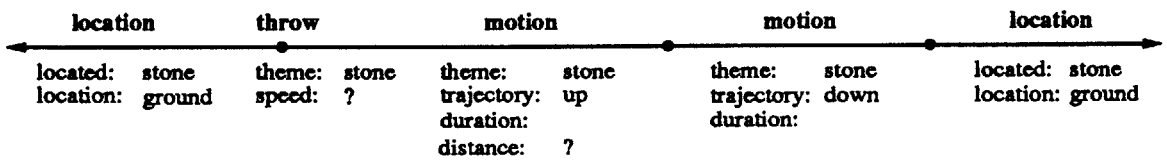
Quantity slots can be related with algebraic *constraints* that represent equations that hold among the values of quantities. For example the definition of average speed indicates that the value of the 'average-speed' slot of a `motion` interval occurrence is the quotient of the value of the 'distance' and the 'duration' slots of the occurrence. As the occurrence structure corresponding to some presentation or problem is constructed, some lexical items and rules introduce constraints among the values of quantity slots. Some sentences set the values of some of the quantity slots. An algebraic reasoning system, described in chapter 7 can manipulate sets of these constraints to solve for the value of some desired quantity, or to verify that an algebraic derivation is valid.

The general form of reasoning that Sagredo can do, then, involves running rules on the grammatical structures created by the parser, and the occurrence structures introduced by the parser and elaborated by those rules. Some entries and rules introduce constraints among the values of quantities. If a sentence asks for the value of a quantity slot, the algebraic reasoning system determines the answer. If a sentence presents a step in an argument that Sagredo should be able to follow, it checks to see if the occurrence structure has been elaborated as the argument requires. In cases where the text presents some new physics information, Sagredo constructs new rules and class definitions to represent the new material.

5.4 Discourse Events

Consider the word problem at the top of figure 5.4. Below the text is shown a representation of the occurrence described by the problem. The `throw` occurrence is a moment occurrence that is preceded by a `location` occurrence and followed by a `motion` occurrence. The verb "return" designates another `motion` occurrence which itself is followed by another `location` occurrence where the 'location' is the same as the place where the throwing happened.

A stone is thrown straight up.
It returns to the ground 6 sec later.
(a) With what speed was it thrown?
(b) Compute the height the stone rises.



A stone	*introduce referrer*
is thrown	*create occurrence*
straight up	*set occurrence 'trajectory' slot*
.	*assert clause*
It	*use referrer*
returns	*create occurrence*
to	*set occurrence 'final location' slot*
the ground	*use referrer*
6 sec later	*set occurrence 'duration' slot*
.	*assert clause*
(a)	*label sentence*
With what speed	*locate occurrence 'speed' slot*
was ... thrown	*use occurrence*
it	*use referrer*
?	*wh-question*
(b)	*label sentence*
Compute	*create occurrence*
the height	*locate occurrence 'distance' slot*
the stone	*use referrer*
rises	*use occurrence*
.	*imperative*

Figure 5.4: Discourse events in processing a word problem.

The construction and manipulation of the occurrence description for a passage presentation text or a story is organized around *discourse events.* In general, any grammatical constituent may be understood as being associated with some event in the discourse that demands a response. The bottom of figure 5.4 lists events that this story involves. The noun phrase "a stone," for example, introduces a referring expression, the verb group "is thrown" creates an occurrence and sets its theme to be the referent of the subject (namely: the stone). Later discourse events refer back to the participants and occurrences that were introduced here, or introduce new ones, or set the values of various slots in the occurrences, or pose questions based on the values of those slots.

Discourse events are inspired by the notion of "illocutionary force" of speech-act theory [265, 266]. The idea there is that each sentence can be analyzed as having a particular "content" plus a particular "illocutionary force." The content of a sentence is, roughly, what it says about the world, or what the world would have to be like for the sentence to be true, or satisfied. The illocutionary force indicates, roughly speaking, the relationship of the sentence to that content, or what is done by the sentence with respect to that content. For example, the declarative sentence "the door is open *states* the content: that the door is open. The question "Is the door open?" has the same content but instead of stating the content, poses it as a question. The imperative "open the door" has, again, the same content, but now presents it as a condition for the hearer to satisfy. The point of the notion of illocutionary force, as for speech act theory in general, is that one "does things with words."

Discourse events are based on this idea but take it both higher and lower in the constituent structure of the sentence. Rather than treating only sentences as the bearer of illocutionary force, I allow any constituent of a sentence to a contribute to the discourse event sequence. The class of event associated with a constituent is a generalization of the idea of illocutionary force. So, as was illustrated, discourse event classes include assertions and questions and imperatives, but also include the introduction and subsequent use of referring expressions, and in fact, any manipulation to the objects of the semantic domain.

Objects larger than individual sentences are also taken to be the loci of discourse events. In particular, whole paragraphs are often involved in the presentation of a physical concept or a derivation or a problem. In general, the model of discourse events is parallel to the occurrence model developed for the semantic domain, and in fact, the same representational system is used for both. Sequences and patterns of lower level discourse events are grouped into higher level discourse events. A problem, for example, typically consists of a sequence of events involving the construction of an occurrence structure, followed by a question about the values of one of the slots. The idea of larger scale discourse events is similar in spirit to the "text grammars" of van Dijk and the [78], "story schemas" of Rumelhart [256] and others.

In addition to their use in the analysis of the text, discourse events are used to organize the response of the Sagredo program to the text. The organization of the

program first involves the collection of LATEX structures like formulas and equations and labels and section and subsection headings. The next step involves a fairly standard parser. As the parser analyzes the sentence, the definitions of words begin the construction of the semantic mappings for the constituents of the sentence. Each constituent is examined when the parser completes it to determine its associated discourse event. This involves the operation of a "discourse event handler" whose operation is described in chapter 7. The discourse event handler is essentially a pattern-driven inference engine. As with the rest of the structural deduction approach, it examines the structure being build up, and creates additional structure provided that certain patterns are satisfied. In this case, the patterns are sequences of discourse events of the appropriate types.

The simplest discourse events are associated with specific lexical entries and with the grammatical rules that introduce constituents, just as is the case for the objects in the semantic domain. The discourse event handler looks for groupings of these events and proposes further elaboration of the discourse event structure.

A great deal of work may happen in response to discourse level events. For example the word "it" which begins the second sentence in figure 5.4 must initiate an attempt to locate another expression which introduces the participant which "it" must refer to. And the use of the verb "returns" in the second sentence indicates that the stone must have been on or near the ground at some earlier point in time, and so that this must be where the "throw" event occurred. All of this work is under the control of the discourse event handler.

Some discourse events require some sort of external response from the program. The most important cases of this involve the questions asked in a problem text. These discourse events are easy to recognize from the grammatical form of the sentences used to present them and the use of question words like "what" and "how far" and so forth. Other kinds of discourse events that require a response are those where the text presents an argument that the program is supposed to be able to follow. These events are signaled by words like "thus" and "since." The text will direct the program's use of the algebraic reasoning system with words like "substitute" and "solve for."

Expository discourse events are those in which new physics information is presented. This may involve the definition of a new term; the statement that a certain class of phenomenon occurs in certain kinds of situations (e.g., friction is developed when one body slides on another); or the statement of a mathematical relation between physical quantities. Texts may also include explicit advice on how to solve certain kinds of problems. Recognizing expository events is relatively difficult because often the form of the sentence is a simple declarative. The system must recognize that the sentence is in a characteristic place with respect to the rest of the text, or else must simply assume that any sentence which is not understood as being part of a problem or an example is making a universal claim.

The boundaries between discourse events of these classes is not entirely sharp, and it is often the case that a specific constituent will be associated with a number

of events of different kinds. For example the sentence like "Consider a body moving along a straight line" might be used to open a passage. The illocutionary force of the whole sentence is of an imperative, and therefore the response event involves obeying the imperative. This, in turn, involves the narrative event of starting a new story, and creating the participants and occurrences described. And this sentence will be part of the larger expository event of a presentation or problem passage. I think that in general the same sorts of issues come up when describing discourse as come up when describing human activity, and for the same reasons: discourse is a kind of activity, special, perhaps, in that it exists to *describe* activity.

Chapter 6

The Interpreter

The Sagredo program is organized around an interpreter that builds data-structures, called "terms," in response to its processing of the input token sequence. Lexical entries associated with words, and various rules contribute forms which are treated by the interpreter as a set of simultaneous equations whose solution is a structure describing the text.

This chapter describes the description formalism, the kinds of forms that can be manipulated, and the use of this system to do simple reasoning. There is a brief discussion of the lowest level details of the implementation.

6.1 The Description Formalism

Terms

The representational entity used by Sagredo is the *term*. A term is a partial function that maps *names* which are LISP symbols, to *values*, which can be arbitrary LISP objects, or other terms. A term with no name-value mappings is said to be "unbound."

Terms are essentially the same as the "complex feature structures" (also called "f-structures") of LFG [180] and other unification-based grammar formalisms. My discussion of terms in this section will follow closely that of Shieber in [272], chapter 3. For those who are familiar with that book, or the issues discussed therein, the most important difference between this presentation and that one is my notation, which is compared with Shieber's in figure 6.1.

Each term is associated with a unique (but otherwise arbitrary) number, and has a name relative to some "parent" term. This number and name are used in the printed representation the term, e.g., `#<TERM T1 [460]>`. When the contents of a term are displayed, the term's printed representation is followed by its name-value mappings. If a value is itself a term, the value's contents are displayed. So term 460 in figure 6.1 maps the symbol `CAT` to the symbol `S`, maps the symbol `SUBJ` to a term which maps

$$\begin{bmatrix} cat: & S \\ head: & \boxed{1}\begin{bmatrix} subject: & \boxed{2} \end{bmatrix} \\ subj: & \begin{bmatrix} cat: & NP \\ head: & \boxed{2} \end{bmatrix} \\ pred: & \begin{bmatrix} cat: & VP \\ head: & \boxed{1} \end{bmatrix} \end{bmatrix}$$

```
#<TERM T1 [460]>
  CAT: S
  HEAD [361]:
   SUBJECT: [969]
  SUBJ:
   CAT: NP
   HEAD [969]: unbound
  PRED:
   CAT: VP
   HEAD: [361]
```

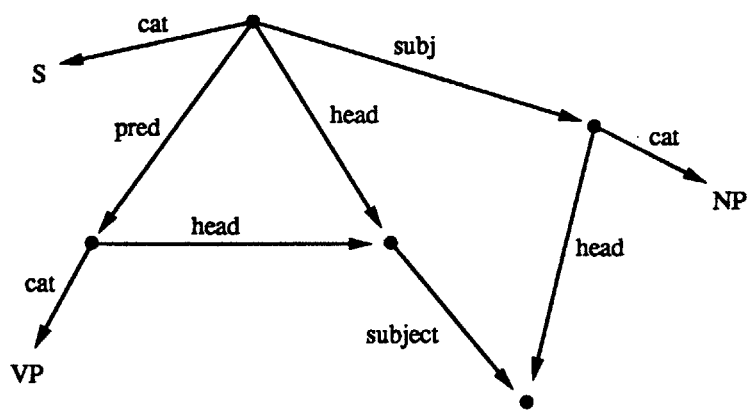


Figure 6.1: Three ways to notate a term. Top left is the one used by Shieber, taken from page 39 of [272]; Top right is my notation; Bottom is a representation as a directed graph.

the symbol CAT to the symbol NP, and maps the symbols SUBJ and PRED to other terms. An unbound term is indicated by the string unbound.

A *reentrant* mapping exists when the same term is the result of more than one name-value mapping. This situation exists in the term displayed in figure 6.1 at two places: the value of the HEAD mapping from term 460 is the same as the value of the HEAD mapping from the term that is the value of the PRED mapping from term 460; and the value of the SUBJECT mapping from the term that is the value of the HEAD mapping from term 460 is the same as the value of the HEAD mapping from the term that is the value of the SUBJ mapping from term 460. If a term is the value of a reentrant mapping, its contents are printed only once, after printing the number of that term. Other mappings to that term are indicated by printing only the number of the term.

A term T is said to map a sequence of symbols $P = \langle P_1 \ldots P_n \rangle$, called a *path*, to

a value V, when V is the result of repeated mappings, first from the initial term T and the initial symbol P_1, then from the result of that mapping and the next symbol, P_2, and so forth, through the path. If any of the intermediate values are not terms, the path mapping is not defined. For example, the term 460, above, maps the path <SUBJ CAT> to the symbol NP.

The structure of a term can be characterized as a set of pairings of paths with values. If a term has no reentrant mappings, all of the values in such a characterization will be non-terms. If a term does have reentrant mappings, each such mapping can be captured by a pairing of two paths. So the structure of term 460 can be expressed by the following set of pairings:

```
<CAT>          =    S
<HEAD SUBJECT> =  <SUBJ HEAD>
<SUBJ CAT>     =    NP
<PRED CAT>     =    VP
<PRED HEAD>    =   <HEAD>
```

The first pairing indicates that the value of the CAT mapping on term 460 is the symbol S. The second line indicates that the value of the path <HEAD SUBJECT> is the same as the value of <SUBJ HEAD>. And so forth.

Note that the following pairing also exists in term 460:

```
<PRED HEAD SUBJECT> = <HEAD SUBJECT>
```

But this pairing is redundant given the pairings above, since the paths <PRED HEAD> and <HEAD> are known to map to the same value, and hence the SUBJECT mappings from there must be the same.

Subsumption

Subsumption is the main logical relation defined between terms. The rough idea is that one term *sub*sumes another if it is *less* complex than the other, but otherwise matches. For example, the following three terms are arranged in order of subsumption:

```
#<TERM S1 [1]>
 TENSE: PRESENT

#<TERM S2 [2]>
 TENSE: PRESENT
 ASPECT: PROGRESSIVE

#<TERM S3 [3]>
 TENSE: PRESENT
 ASPECT: PROGRESSIVE
 AGREEMENT:
  NUMBER: SINGULAR
```

Each term in the above series is structurally equivalent to the one above it, with some additional name-value mappings.

More precisely: Term T_1 subsumes T_2 iff for each path P defined on T_1,

1. If T_1 maps P to a non-term value V_1, T_2 maps P to an identical value V_2. (In the implementation, V_1 and V_2 must satisfy the COMMON LISP predicate `equalp`. See [283], section 6.3.)

2. If T_1 maps P to a term T_{P_1}, then T_2 maps P to a term T_{P_2} such that T_{P_1} subsumes T_{P_2}.

3. If T_1 maps P to a term T_{P_1}, and another path R also maps P to the same term T_{P_1}, then if T_2 maps P to the term T_{P_2}, T_2 also maps R to T_{P_2}.

Note that an unbound term vacuously subsumes all other terms.

Clause 3 in this definition is needed to handle reentrant mappings. Without it, terms 4 and 5, below, would each subsume the other:

```
#<TERM T4 [4]>
 AGREEMENT:
  NUMBER: SINGULAR
 PRED:
  CAT: VP
  AGREEMENT:
   NUMBER: SINGULAR

#<TERM T5 [5]>
 AGREEMENT [56]:
  NUMBER: SINGULAR
 PRED:
  CAT: VP
  AGREEMENT: [56]
```

Now consider the following term:

```
#<TERM T6 [6]>
 AGREEMENT:
  NUMBER: SINGULAR
  PERSON: THIRD
 PRED:
  CAT: VP
  AGREEMENT:
   NUMBER: SINGULAR
   PERSON: FIRST
```

Now term 4 subsumes term 6 but term 5 does not, because the structure of term 6 violates the reentrant mapping in term 5. According to the complete definition of subsumption, term 4 subsumes term 5, but not vice versa, and neither of term 5 or term 6 subsumes the other.

The subsumption relation is a partial ordering on the domain of terms. This fact, and others about the subsumption relation, are discussed in [173].

Unification

The *unification* operation on a pair of terms results in a third term which is subsumed by both terms, provided that such a term exists, otherwise the unification is said to *fail*. Furthermore, the unification operation yields the *simplest* such term, as ordered by the subsumption relation. For example, the unification of terms 7 and 8, below, is term 9:

```
#<TERM T7 [7]>
 TENSE: PRESENT

#<TERM T8 [8]>
 ASPECT: PROGRESSIVE

#<TERM T9 [9]>
 TENSE: PRESENT
 ASPECT: PROGRESSIVE
```

Terms 10 and 11 can't unify, because they don't agree on the value of REST:

```
#<TERM T10 [10]>
 FIRST: SPIDER
 REST: NIL

#<TERM T11 [11]>
 FIRST: SPIDER
 REST:
  FIRST: SNAIL
  REST: NIL
```

Recall terms 4 and 5 above, they differed only in that term 5 contained a reentrant mapping. Consider unifying term 12 with each of them:

```
#<TERM T12 [12]>
 AGREEMENT:
  PERSON: THIRD
 PRED:
  AGREEMENT:
   PERSON: FIRST
```

With term 4, unification will will yield a term isomorphic to term 6 above:

```
#<TERM T412 [412]>
 AGREEMENT:
  NUMBER: SINGULAR
  PERSON: THIRD
 PRED:
  CAT: VP
  AGREEMENT:
   NUMBER: SINGULAR
   PERSON: FIRST
```

But unification between term 5 and term 12 will fail.

A slight, but useful, complexity in the definition of unification involves the case of unifying a term with a non-term. It would seem that, according to the definitions of subsumption and unification, this should always fail. However no inconsistency is introduced by defining the result of unifying an *unbound* term with a non-term value as being the value. So we can unify the two terms:

```
#<TERM T13 [13]>
 CASE: unbound

#<TERM T14 [14]>
 CASE: DATIVE
```

to yield term 14. This example shows why this addition to the definition is useful: it makes unbound terms "invisible"—their presence doesn't prevent the term from unifying with terms with which it would unify were the unbound term simply not present.

This unification relation (sometimes called "feature unification") is different from the "clause" or "term" unification used in logic programming. In clause unification, a clause has a fixed number of argument positions, some of which may contain variables. Unifying two clauses involves finding bindings for the variables that would make the two clauses structurally identical. Either clause, with its variables replaced by the bindings thus found, is called the "most general unifier" of the two clauses.

So, for example, in the two clauses below, `?x` and `?y` are variables:

```
P(a ?x c)
P(?y b c)
```

The two clauses would be identical were `?x` bound to `b` and `?y` bound to `a`, the most general unifier is therefore: `P(a b c)`.

Clause unification can be expressed with feature unification by treating the clause as a sequence of terms and representing variables by unbound terms:

```
#<TERM T101 [101]>
 PRED: P
 FIRST: A
 REST:
  FIRST: unbound
   REST:
    FIRST: C
    REST: NIL

#<TERM T102 [102]>
 PRED: P
 FIRST: unbound
 REST:
  FIRST: B
   REST:
    FIRST: C
    REST: NIL
```

These two terms unify to yield:

```
#<TERM T103 [103]>
 PRED: P
 FIRST: A
 REST:
  FIRST: B
   REST:
    FIRST: C
    REST: NIL
```

Feature unification can't be expressed directly with clause unification because clause unification requires that for two clauses to unify they must be of the same arity. No such restriction applies to feature unification. The fact that feature unification is strictly more powerful than clause unification means that it can be used in all cases where clause unification can be used. In particular, I will make use of it to define a simple logic programming language.

In clause-unification based logic programming, the clause is treated as a logical assertion, true or false with respect to some model. In a feature-unification based logic programming system the terms are thought of as *descriptions* of some object or objects. A useful property of feature unification for this interpretation is that the descriptions are always *partial*—more information, in the sense of more name-value mappings, can always be added to an existing description to yield a more specific, but still partial description.

The idea of "partial descriptions" is important to the semantic theories of Barwise and Perry [22] and Davidson [67]. In both approaches any specific characterization of a situation or an event is always capable of being further refined, by adding more information. The intuition behind this point of view is that any representation of an actual object or event, however true it may be, cannot capture everything about the object or event.

A clause based approach to representation could certainly be used to represent partial descriptions. In particular, one could take a set of clauses and all of their consequences (i.e., a logical theory) as the basic representational entity. New clauses can always be added to such a set provided that they do not contradict any existing members. Indeed this approach seems to be the one outlined by Davidson. However the computational overhead of such an approach, combined with the evident power of the feature unification approach, convinced me to use the latter.

6.2 Unification Equations

Another way to think about feature unification is as a way to assert that two terms are *equal*. A set of "unification equations" can then be used to describe the structure of a term.

Consider by analogy a set of simultaneous equations in algebra:

$$\begin{aligned} x &= y + 1 \\ x &= 2y \end{aligned}$$

These two equations assert that the following expressions all stand for the same number: x, $y + 1$, and $2y$. The only way that this could be true is if x stands for 2 and y stands for 1. Thus a set of simultaneous equations can be taken to be a set of assertions about the values of variables. The solution to a set of simultaneous equations is a set of assignments of values to variables that makes the assertions true.

Now recall the discussion of "paths" above. The structure of a term can be specified by listing all of the mappings of paths in the term to values. Consider again one such set of pairings presented above, with the = sign now interpreted literally as equality when it equates two paths:

<CAT>	=	S
<HEAD SUBJECT>	=	<SUBJ HEAD>
<SUBJ CAT>	=	NP
<PRED CAT>	=	VP
<PRED HEAD>	=	<HEAD>

This set of "equations" is satisfied by a term which maps the symbol CAT to the symbol S, and maps the path <HEAD SUBJECT> to the *same term* as it maps the path <SUBJ HEAD>, and so forth. We will require that the "solution" to a set of such equations is a *minimal* term (as ordered by the subsumption relation) that satisfies all of these equations, that is, one which has only the mappings specified. For example term 460, above, is a solution to this set of equations.

Solving Unification Equations

There are relatively straightforward algorithms for finding the solution to a set of unification equations. (For example one is described in [180].) My implementation is based on the following two functions:

`unify` takes two arguments. If both are terms, it attempts to add structure to one of the terms to make it the unification of the two terms. If one of the arguments is a non-term value, then if the other argument is an unbound term, that argument is unified with the value; if the other argument is a bound term, the unification fails. If both arguments to `unify` are non-terms, the unification succeeds if the values are `equalp`, otherwise it fails.

`follow-path` takes a term and a path (represented as a sequence of symbols) and tries to find or construct a term which is the value of the last mapping in the path.

The solution algorithm is run on a term, which will be called the *current term*, and a list of forms. It processes one form at a time, using the above functions to elaborate the structure of the current term. If all of the forms are processed successfully, the final structure of the current term will satisfy all of the forms.

In the simplest case, each form specifies a name-value mapping. For example, if the following forms:[1]

```
(== a 'eagle)
(== b 'crow)
(== c 'sparrow)
```

[1] I adopt the convention that any text which is or could be part of a program will be printed in lower case, while any LISP data-structure will be printed in upper case, except for LISP strings, which are printed in upper and/or lower case, depending on how they were originally typed in.

are interpreted with respect to an initially unbound term, the the result is:

```
#<TERM T20 [20]>
 A: EAGLE
 B: CROW
 C: SPARROW
```

The first form indicates that `follow-path` should be called with the current term and a path containing only the symbol `A`. The result will be an unbound term. The operator `==` indicates that this term and the symbol `EAGLE` should then be unified. This succeeds and yields a term with a single name-value mapping. The same procedure is then used for the next two forms. Note the use of the LISP quotation symbol `'` to indicate a non-term value.

A longer path is notated as a list beginning with the symbol `<<`, followed by the sequence of symbols. (This notation is based on that used in "constraint languages" based on [286], for example: [26].) So

```
(== (<< w x y) 'squirrel)
```

is solved to yield:

```
#<TERM T21 [21]>
 W:
  X:
   Y: SQUIRREL
```

If both arguments to `==` are specified by paths, `follow-path` will return two terms, and therefore the result of `unify` will be a reentrant mapping:

```
(== (<< a b) (<< w x))
(== (<< w x c) 'salamander)
```

These equations yield the term:

```
#<TERM T22 [22]>
 A:
  B [221]:
   C: SALAMANDER
 W:
  X: [221]
```

The paths `(<< a b)` and `(<< x y)` on term 22 map to the same term, term 221.

The following subsections describe the syntax and operational semantics of the forms processed by the interpreter. All *unification-forms* are lists: the first element of the

list is the *form operator*, and determines what will be done with the form. So far the only form operator we have seen is `==`. The form operator `==` is followed by two term expressions. A `==` form is *satisfied* just in case the values of those two expressions can be successfully unified.

Two additional form operators are useful:

`(in` *term-expression* . *unification-forms*`)`
The `in` operator is used to interpret interpret the sequence of *unification-forms* with respect to some other term than the current term. First the *term-expression* is evaluated, as described in the next subsection, then the *unification-forms* are interpreted with the value of *term-expression* as the current term. Among other things, `in` forms can be used to save typing. For example the following sequence of unification forms:

```
(== (<< a b c) 'w)
(== (<< a b d) 'x)
(== (<< a b e) 'y)
```

is equivalent to:

```
(in (<< a b)
  (== c 'w)
  (== d 'x)
  (== e 'y))
```

`(progn` . *unification-forms*`)`
The `progn` operator is used to group together the sequence of *unification-forms* so that they can be treated as a single form. The result of interpreting a `progn` form is the same as the result of interpreting the sequence of *unification-forms*. The `progn` form operator is used mostly inside forms headed by the `or` and `branch` disjunctive form operators, and the `if` and `when` test form operators, described below. An "empty" `progn` form, i.e., `(progn)`, is trivially satisfied.

If thinking logically is a virtue, perhaps `progn` should have been named `and`. If we think of unification forms as assertions about the structure of a term, the latter name would have been more appropriate. However, as will be seen below and in the next chapter, the sequential ordering of unification forms matters, and so I use the name `progn` to remind myself that the sequence of forms it encloses are interpreted sequentially.

Term Expressions

Term expressions are evaluated by the interpreter to construct or locate terms and values.

So far we have seen three kinds of term expressions:

- A path, notated with `<<` followed by a sequence of symbols. The value of a `<<` expression is the value that `follow-path` returns when called with the current term and the sequence of symbols. The expression `(<<)` is a valid path expression: it yields the current term.

- A single, unquoted symbol. This yields the value of `follow-path` called on the current term and the sequence consisting of the single symbol. (Note therefore that there are two equivalent ways to specify a path with a single element. For example, these two term expressions are equivalent: `cat` and `(<< cat)`.)

- A quoted LISP symbol. This yields the given symbol as a value. In general, any LISP expression can follow the `'` symbol: the value is the expression itself.

In addition, the following term expressions are defined:

- A number, written any way that the LISP reader accepts. (See [283], section 22.1.2.) The value is the LISP representation of the number.

- A LISP string, delimited with " marks. The value of this expression is the string itself.

- A list of the form `(<<*` *term-expression* `.` *symbol-sequence*`)`. First *term-expression* is interpreted. The value of the `<<*` expression is the result of calling `follow-path` with the value of *term-expression* and the symbols in *symbol-sequence.*

- A list of the form: `(parent)`. The term returned when interpreting a `(parent)` term expression is the term that "contains" the current term as the value of one of its name-value mappings. `(parent)` is often used with `<<*` to specify the structure of another subterm of the term that contains the current term. So for example:

  ```
  (== (<<* (parent) number agreement) 'singular)
  ```

  requires that the current term be inside a term in which the term expression `(<< number agreement)` is mapped to the symbol `SINGULAR`.

  A `(parent)` expression can contain an argument which must be an integer greater than or equal to one. The expressions `(parent)` and `(parent 1)` are equivalent. The expression `(parent 2)` yields the parent 2 "levels" up – that is: the `(parent)` of the `(parent)` of the current term. And so forth for any integer value. If no such "parent" exists, the term expression, and the unification form it is inside, fails.

A small number of additional term expressions are defined. They will be introduced in the next chapter.

Disjunction

A set of unification forms constructed with the form operators and the term expressions described above has at most a single solution. The following three form operators provide for sets of forms that have more than one solution.

(or . *unification-forms*)
An or form expression is satisfied if any of the *unification-forms* are satisfied. So the following expression:

```
(or (== a 'lion)
    (== a 'tiger)
    (== a 'bear))
```

Is satisfied by each of the three terms:

```
#<TERM T30 [30]>
 A: LION

#<TERM T31 [31]>
 A: TIGER

#<TERM T32 [32]>
 A: BEAR
```

Since each of the forms in an or expression is a separate disjunction, the progn operator is useful to collect multiple forms into a single disjunction:

```
(or (== person 'first)
    (== person 'second)
    (progn
      (== person 'third)
      (== number 'plural)))
```

(branch . *unification-forms*)
An branch form is equivalent to an or form if considered declaratively: it is satisfied by any term that handles any of the *unification-forms*. But the operator branch is handled differently by the interpreter. Whereas *all* solutions to or forms are considered, only the first argument to branch is considered, unless it fails to be satisfied, in which case the second form is considered, and so forth. This difference makes possible a bit of control over the operation of the interpreter, by allowing the programmer to control the order in which the disjunctions are considered. In most cases, however, that will be the only difference between using a branch form and an or form. The same solutions will be found either way,

(`optional` . *unification-forms*)
An `optional` form first attempts to satisfy the *unification-forms*. If that works, the `optional` form is itself satisfied. Otherwise the system retracts any changes that the *unification forms* made to the analysis so far, and continues as if the `optional` form were empty. An `optional` form is equivalent to:

(`branch` (progn . *unification-forms*) (`progn`))

So the equations:

```
(optional (== a 'duck)
          (== b 'goose))
(== c 'swan)
```

are satisfied by:

```
#<TERM T28 [28]>
 A: DUCK
 B: GOOSE
 C: SWAN
```

and:

```
#<TERM T29 [29]>
 C: SWAN
```

The form operator `or-optional` is also defined. The syntax is the same as for `optional`, but an `or-optional` form is equivalent to

(`or` (progn . *unification-forms*) (`progn`))

The difference in the interpretation between `optional` and `or-optional` corresponds to the difference between `branch` and `or`.

Conditionals

In all of the cases discussed so far, the result of interpreting a form with respect to a term could have two outcomes: either the term already satisfies the form, in which case no changes are made, or the term can be extended to satisfy the form. Suppose, for example, we interpret the following form on an unbound term:

```
(in (<< a b) (== c 'food))
```

to obtain:

```
<TERM T40 [40]>
 A:
  B:
   C: FOOD
```

Now consider the difference between interpreting the following two forms with respect to term 40:

```
(== (<< a b d) 'shelter)
(== (<< a b c) 'food)
```

Both of these forms will succeed. But the first form adds structure to term 40, while the second is already satisfied by term 40.

The point of this discussion is to motivate the `test` form operator. The syntax of `test` is similar to that for `progn`:

`(test` . *unification-forms*`)`

The `test` form operator is used to determine if a term satisfies a sequence of unification forms, but does not modify the term in doing so. That is: it only succeeds if the structure of the term already satisfies the equations. So, on term 40 above, this form would succeed:

```
(test (== (<< a b c) 'food))
```

while this one would not:

```
(test (== (<< a b d) 'shelter)
```

Here are the conditional operators used by Sagredo:

`(if` *condition-form consequent-form alternative-form*`)`

All of the arguments are single unification forms. The interpreter attempts to satisfy the *condition-form*, if it is successful, the *consequent-form* is evaluated. If the *condition-form* failed, the *alternative-form* is satisfied.

The *condition-form* of a `test` form is interpreted as an "ordinary" unification form, that is, it will succeed either if the forms are already satisfied, or if the current term can be modified to make them be satisfied. If the *condition-form* is a `test` form, it will be satisfied only if the term already satisfies the test. The `if-test` form operator combines this functionality. A form:

`(if-test` *condition-form consequent-form alternative-form*`)`

is equivalent to

`(if (test` *condition-form*`)` *consequent-form alternative-form*`)`

(**when** *condition-form* . *consequent-forms*)
This operator is like **if** except that it takes any number of *consequent-forms* and no *alternative-form*. If the *condition-form* is satisfied, all of the *consequent-forms* are interpreted. As with **if**, the *condition-form* may modify the current term to succeed. The operator **when-test** is also provided, so that the following two forms are equivalent:

(**when-test** *condition-form* . *consequent-forms*)
(**when** (**test** *condition-form*) . *consequent-forms*)

(**unless** *condition-form* . *alternative-forms*)
This operator is like **if** except that it takes any number of *alternative-forms* and no *consequent-form*. If the *condition-form* is *not* satisfied, all of the *alternative-forms* are interpreted. As with **if**, the *condition-form* may modify the current term to succeed. The operator **unless-test** is also provided, so that the following two forms are equivalent:

(**unless-test** *condition-form* . *alternative-forms*)
(**unless** (**test** *condition-form*) . *alternative-forms*)

The presence of unification forms headed by **test** and **if** and the operators based on them in a set of unification equations means that the order of interpretation might influence whether the set has a solution. Thus such a set of forms cannot be treated simply as a set of simultaneous equations describing a structure.

Miscellaneous Operators

The following additional unification form operators are defined:

(**succeed**)
This form is always satisfied.

(**fail**)
This form is never satisfied. An expression of the form

(**when-test** *condition-form* (**fail**))

is the closest the language here described comes to having a logical 'not' operator. The illustrated form succeeds just in case the *condition-form* does not succeed.

(**term** *term-expression*)
This is satisfied if the *term-expression* is satisfied. The **term** operator is useful in tandem with **test**. If used in an ordinary context, it will succeed either of the *term-expression* yields an already existing term, or if such a term can be constructed. If used inside a **test**, it succeeds only if the term indicated by *term-expression* already exists. Thus one can write **test** expressions that succeed if a term exists, but do not require it to have any particular value.

Classes

Sequences of forms can be collected into named "classes." Most of the knowledge in Sagredo is expressed in the definitions of classes and much of what Sagredo learns involves acquiring new class definitions.

The following LISP macro defines a class:

```
(define-class class-1 ()
  (== a 1)
  (== (<< b c d) 2))
```

The first argument to `define-class` is a name, in this case `class-1` that will be used to locate this class. The second argument is a list of "super" classes for this class, these will be discussed below, in this case the list is empty. The rest of the arguments are unification forms.

Classes are used with the form operator `call`:
`(call` *term-expression* `class-name)`
The forms associated class named *class-name* are interpreted with respect to the term which is the value of *term-expression*. So the form

```
(call (<<) 'class-1)
```

yields a term like this:

```
#<TERM T60 [60]>
 A: 1
 B:
  C:
   D: 2
```

The second argument to `define-class` is a list of class names. The forms associated with those classes (and their super classes) will be interpreted with respect to the current term before the forms associated with the given class.

Consider, for example, the following class definitions:

```
(define-class noun ()
  (== (<< class major-class) 'noun))

(define-class count-noun (noun)
  (term (<< number)))

(define-class singular-noun (count-noun)
  (== (<< number) 'singular))
```

```
(define-class plural-noun (count-noun)
  (== (<< number) 'plural))

(define-class concrete-noun (noun)
  (== (<< class concreteness) 't))

(define-class singular-concrete-noun
              (singular-noun concrete-noun))
```

The class **noun** is defined with no super classes. The class **count-noun** has as its super class the class **noun**. Note that its only form creates the **number** slot on the term, without filling in a value. The classes **singular-noun** and **plural-noun** both fill in values for the **number** slot. The class **concrete-noun** sets the value of another slot. And the class **singular-concrete-noun** inherits from both **singular-noun** and **concrete-noun**. So the form:

```
(call head 'singular-concrete-noun)
```

Would yield a term:

```
#<TERM T64 [64]>
 HEAD:
  CLASS:
   MAJOR-CLASS: NOUN
   CONCRETENESS: T
  NUMBER: SINGULAR
```

The class **count-noun** would be useful in a test, as in:

```
(when-test (call head 'count-noun)
  ...)
```

where the consequent forms of the **when-test** form would only be interpreted when the **head** satisfies the the forms associated with the class **count-noun**. This would be true if the term mapped the path **(<< class major-class)** to **NOUN**, and if it had a subterm named **NUMBER**.

This example also introduces a convention that I use when defining classes for Sagredo. Most terms have a 'class' slot that contains a number of pairings of names to symbols. The set of pairings can be used to determine the class of the term. In general, the 'major-class' slot of the 'class' term contains the name of the most important class. For example the terms used in parsing all have as their 'major-class' the symbol **grammar**.

Subclasses are then represented in various ways. In some cases, the subclasses of a specific classes form a disjoint set. If so, all of the subclasses are represented as names stored in a slot whose name is derived in the superior class. For example subclasses of the class **grammar** are represented by values in the **grammar-class** slot. Examples are **clause**, **noun-phrase**, **punctuation** and so forth.

In other cases subclasses do not form disjoint sets. These are represented by their own name-value pairings. For example the subclass representing "concrete nouns" is encoded by the pairing of the slot 'concreteness' with the value T. The details of the various type hierarchies in Sagredo will be illustrated as they come up in the next two chapters.

Rules

Logical rules can be defined for Sagredo. An example rule is:

```
(define-rule rule-1
    ((dog (call (<<) 'dog))
     (cat (call (<<) 'cat)))
  (== (<< dog enemy) cat)
  (== (<< cat enemy) dog))
```

The first element is the name of the rule, in this case `rule-1`. The next element is a list of lists. The first element if each list is the name of a slot. It is followed by a list of forms that are interpreted relative to the slot. The body of a rule consists of a list of forms interpreted relative to a term representing the rule itself.

The idea is that the rule will be applied to a pair of terms. These terms will be unified with the terms corresponding to `dog` and `cat`. If the forms following the slotnames succeed, then the forms in the body of the rule will be interpreted. Note that those forms will modify the arguments to the rule.

Consider the term

```
#<TERM T33 [33]>
  DOG:
   CLASS:
    MAJOR-CLASS: DOG
  CAT:
   CLASS:
    MAJOR-CLASS: CAT
```

The form `(apply-rule 'rule-1 dog cat)` is now interpreted within this term. The first step attempts to interpret `(call (<<) 'dog)` with respect to the term named 'dog' on term 33. This succeeds, and the form `(call (<<) 'cat)` is interpreted with respect to the term named 'cat' on term 33. This also succeeds, so the body forms

are interpreted with respect to the rule term. The effect of this is to modify both of the argument terms, yielding:

```
#<TERM T33 [33]>
  DOG [60]:
   CLASS:
    MAJOR-CLASS: DOG
   ENEMY: [70]
  CAT [70]:
   CLASS:
    MAJOR-CLASS: CAT
   ENEMY: 60
```

All of Sagredo's "knowledge" takes the form of class and rule definitions. It learns physics by constructing new class and rule definitions from the terms that it constructs while reading the source text.

6.3 Implementation

If there were no disjunction operators, terms could be implemented simply as LISP association lists and the interpreter could process forms sequentially, adding structure as it went along. However the existence of disjunction means that the interpreter has to be organized as a search. When the interpreter encounters a disjunction operator, it must decide which branch to pursue and record the other alternatives for later consideration if the initially chosen one should fail.

At first I explored a TMS approach to this problem (in particular, I implemented a version of the TME described by McAllester in [216]. However most of the work done by the interpreter involves keeping track of equalities between terms. Since unifying two terms means that all subterms of the two terms must also be unified, and since any such unification can potentially be retracted, this means that the system spends virtually all of its time dealing with equality assertions.

So I decided on a more tractable alternative that isn't as efficient about managing dependencies, but makes up for it by handling equality easily. Each disjunctive choice creates a new *context*. A context maps terms to their contents. Before anything (e.g., unification) is done to a term, it is updated with respect to the current context. If the term has already been updated for the current context, this just requires a single test. If not, the current context and all of its preceding contexts are searched for one that has recorded the contents of the current term. These contents are then copied into the current term.

This mechanism means that working within the current context is easy, as is restoring a previously visited context, as happens when unification in some context fails. It does not, however, attempt to locate the specific forms that led to the failure.

The search is managed in a task network. Each task contains its context, the next form to interpret, the next token in the sentence, and success and failure handlers. It attempts to interpret the form. If it succeeds, the success handler is called, this usually contains a continuation with the rest of the forms from some class body. if the form fails (perhaps because some unification failed), the failure handler is called. This usually contains one of the alternatives from some previous choice point. If the form is disjunctive, a new task with a new context is created for each alternative.

The program chooses which task to work on based on a simple heuristic. First of all, it works on tasks which are further along in the sentence—which have parsed more words. If two tasks have parsed the same number of words, the task which is currently working on the overall simpler analysis is chosen. This can be computed easily from the total number of mappings in the context associated with the task, and is updated whenever a context is modified.

This heuristic makes sense for two reasons: first, if there are alternative consistent parsings of the sentence, we would prefer to get the simpler one back first, if only because it is likely to be faster to find; second, if there is only one parse, we don't want the parser creating arbitrarily big analyses of the sentence. By preferring simpler analyses the program is guaranteed to find any correct analyses if they exist.

Chapter 7

Mechanisms

This chapter presents the various mechanisms used by Sagredo to process expository text. Most of these are built out of or on top of the unification grammar interpreter described in the last chapter.

7.1 Parsing

As I mentioned in chapter 3, the program begins work on a file of LATEX source text. The initial typographic processing involves pretty much what the LATEX interpreter has to do in preparing the text for formatting. This involves collecting the arguments to LATEX commands, locating the punctuation of sentences, noting paragraph and section boundaries, and so forth. The result is a sequence of "tokens" that are then used as input for the parser. Most of the tokens contain strings that correspond to words, but some contain parsed formulas and equations.

The operation of the parser will be illustrated by considering the first sentence of the source text:

> If an object moves a distance d in a time t, its *average speed* for the interval is defined to be d/t.

The rules illustrated in figure 7.1 represent a context-free version of the grammar that Sagredo uses to parse this sentence. The actual grammar that is used is more restrictive than this because of the context-sensitivity of the unification grammar mechanism.

Each grammatical category is represented by a class that inherits from the class **grammar**. The category classes just record a value for their 'grammar-class' slot, e.g.:

```
(define-class grammar ()
  (== (<< class major-class) 'grammar))
```

Clause	→	*Nominal Verb-Group Complement* Adjunct**
Sentence	→	*Clause Punctuation*
Sentence	→	*Introducer Clause* {*Punctuation*} *Clause Punctuation*
Nominal	→	{*Determiner*} {*Adjective*} *Noun* {*Formula*}
Verb-Group	→	{*Modal*} {*Auxiliary*} *Verb*
Complement	→	*Nominal* | *Clause* | *Formula*
Adjunct	→	*Prep-Phrase*
Prep-Phrase	→	*Preposition Complement*
Noun	→	distance | interval | object | speed | time
Determiner	→	a | an | its
Adjective	→	average | constant
Modal	→	to
Auxiliary	→	is
Verb	→	be | defined | moves
Preposition	→	for | in
Introducer	→	if
Punctuation	→	, | .

Figure 7.1: Example grammar rules. The symbol '*' represents that zero or more repetitions of the preceding category are allowed. A category enclosed in {braces} is optional. Alternatives are separated by '|'. The category *Formula* is recognized by a special parser when the source file is initially processed.

```
(define-class clause (grammar)
  (== (<< class grammar-class) 'clause))

(define-class nominal (grammar)
  (== (<< class grammar-class) 'nominal))
```

Certain categories are filled only by individual words. These contain a `word-class` slot:

```
(define-class preposition (grammar)
  (== (<< class word-class) 'preposition))

(define-class noun (grammar)
  (== (<< class word-class) 'noun))

(define-class verb (grammar)
  (== (<< class word-class) 'verb))
```

The class verb has the following subclasses:

```
(define-class modal-verb (verb)
  (== (<< class verb-class) 'modal))

(define-class auxiliary-verb (verb)
  (== (<< class verb-class) 'auxiliary))

(define-class head-verb (verb)
  (== (<< class verb-class) 'head))
```

Verbs are further differentiated according to the complements they can take. The complements of a clause will be collected into a sequence each of whose elements contains a 'first' and 'rest' slot. The 'first' slot contains a complement constituent, and the 'rest' slot contains either NIL or more complements.

The following three subclasses of head-verb differ only on how many complements they take. As will be described below, a head-verb will occur in the 'head' slot of the 'verb-group' of a clause. The special term expression (clause) searches up the parent pointer from a term to find the first one whose 'grammar-class' is clause or sentence. Hence an intransitive-verb has no complements, a transitive-verb has a single complement and a bitransitive-verb has two complements:

```
(define-class intransitive-verb (head-verb)
  (== (<<* (clause) complements) nil))

(define-class transitive-verb (head-verb)
  (== (<<* (clause) complements rest) nil))

(define-class bitransitive-verb (head-verb)
  (== (<<* (clause) complements rest rest) nil))
```

Words are defined with define-word forms. This is similar to a define-class form except the word's string is given as the first argument. For the purposes of this example, some of the words can be defined only as instances of their grammatical categories:

```
(define-word "distance" (noun))
(define-word "object" (noun))
(define-word "an" (determiner))
(define-word "average" (adjective))
(define-word "if" (clause-introducer))
(define-word "," (punctuation))
```

The actual definitions of individual words include information about the referents of phrases containing those words. This was illustrated in the demo in chapter 3.

Verbs are, in general, more complicated. The verb "moves" for example takes a single nominal as its complement:

```
(define-word "moves" (transitive-verb)
  (in (clause)
    (call (<< complements first) 'nominal)))
```

While the verb "defined" wants a clause as its complement:

```
(define-word "defined" (transitive-verb)
  (in (clause)
    (call (<< complements first) 'clause)))
```

The parsing procedure is invoked on a specific token (which will be called the "current token" in this discussion) and a specific term (the "current term"). That term will have already been assigned a value for its 'grammar-class' slot. The contract of the parser is to instantiate an analysis of part of the text beginning with the current token on the current term. The parser advances the current token as each word is successfully processed. The parse procedure is directed by "parse-methods" defined for each grammatical class. A parse-method is just a sequence of unification forms which are interpreted on a term representing an instance of that class. The special form operator `parse` invokes the parser on a subconstituent of the current constituent.

As a simple example, here is a parse method for the class `prep-phrase`:

```
(define-parse-method prep-phrase ()
  (call head 'preposition)
  (parse head)
  (call object 'nominal)
  (parse object))
```

Let us suppose that the parser is invoked on the term

```
#<TERM PREP-PHRASE [21]>
 CLASS:
  MAJOR-CLASS: GRAMMAR
  GRAMMAR-CLASS: PREP-PHRASE
```

and on the sequence of tokens beginning:

```
in a time $t$ ...
```

The parse method for the class `prep-phrase` is invoked. This creates a subslot of the current term named 'head' and instantiates the class `preposition` on it. The parser is now called on that term with the same current token.

When the parser reaches a word category, as in this case, it tries to find a word whose string is the same as that of the current token, and which can be instantiated on the current term, If one is found, it records the string of the word on the term. There is only one word whose string is "in" and that is indeed a preposition, so the parser succeeds and advances the current token. After the parser is finished with the 'head' slot of term 21, it looks like this:

```
#<TERM PREP-PHRASE [21]>
 CLASS:
  MAJOR-CLASS: GRAMMAR
  GRAMMAR-CLASS: PREP-PHRASE
 HEAD:
  CLASS:
   MAJOR-CLASS: GRAMMAR
   WORD-CLASS: PREPOSITION
  STRING: "in"
```

The current token is now "a". The next form in the parse method for prep-phrase now directs that a 'object' slot be created and instantiated with the class nominal.

Here is the parse method for the class nominal:

```
(define-parse-method nominal ()
  (branch
   (progn
     (call (<<) 'nominal-pronoun)
     (parse (<<)))
   (progn
     (optional
      (call determiner 'determiner)
      (parse determiner))
     (optional
      (call modifier 'adjective)
      (parse modifier))
     (call head 'noun)
     (parse head)
     (optional
      (call appositive 'formula)
      (parse appositive)))))
```

This method consists of a single top-level branch form, which, in turn contains two forms. The first form of the branch handles the case where the nominal is realized as a single pronoun. If such were the case, (parse (<<)) would immediately succeed.

That doesn't happen in this example however because the next token is the string "a". So the other form in the branch is explored. The first step is to call the term a

noun-phrase and to parse its 'determiner' slot. This succeeds because a determiner with the the string "a" is defined.

The current token is now "time". The next form in the parse method is an **optional** entry that looks for an adjective. No word whose string is "time" is recorded as an adjective, so the parser continues to the 'head' slot. That form succeeds.

The parse method for **formula** is a special-case function that looks to see of a pre-parsed formula has been stored on the current token. If so, that formula is recorded in the 'expression' slot of the current term and the parser succeeds. So the final state of term 21 when the parser is done with the **prep-phrase** parse method is:

```
#<TERM PREP-PHRASE [21]>
 CLASS:
  MAJOR-CLASS: GRAMMAR
  GRAMMAR-CLASS: PREP-PHRASE
 HEAD:
  CLASS:
   MAJOR-CLASS: GRAMMAR
   WORD-CLASS: PREPOSITION
  STRING: "in"
 OBJECT:
  CLASS:
   MAJOR-CLASS: GRAMMAR
   GRAMMAR-CLASS: NOMINAL
  DETERMINER:
   CLASS:
    MAJOR-CLASS: GRAMMAR
    WORD-CLASS: DETERMINER
   STRING: "a"
  HEAD:
   CLASS:
    MAJOR-CLASS: GRAMMAR
    WORD-CLASS: NOUN
   STRING "time"
  APPOSITIVE:
   CLASS:
    MAJOR-CLASS: GRAMMAR
    GRAMMAR-CLASS: FORMULA
   EXPRESSION: (VARIABLE "t")
```

The parse method for clauses requires that the head verb of the verb group control how many complements are allowed in the clause. This is a simplified version of the

parse method for a clause.

```
(define-parse-method clause ()
  (call subject 'nominal)
  (parse subject)
  (call verb-group 'verb-group)
  (== (<< subject agreement)
      (<< verb-group firstverb agreement))
  (parse verb-group)
  (call complements 'parse-complements)
  (call adjuncts 'parse-adjuncts))
```

The 'subject' is created and parsed. The line unifying the 'agreement' slots of the subject and the 'firstverb' of the 'verb-group' take care of number agreement in the clause. (So that, for example, "They run" would be accepted, but "They runs" would not.) The parse method for the class `verb-group` assigns to the slot 'firstverb' the first verb that appears.

After the 'verb-group' is parsed the clause will have a 'complements' slot, courtesy of the lexical entry for its head verb. For example if the head verb were `"moves"`, the clause would look like this:

```
#<TERM CLAUSE [79]>
 SUBJECT:
  ...
 VERB-GROUP:
  ...
  HEAD:
   ...
   STRING: "moves"
 COMPLEMENTS:
  FIRST:
   CLASS:
    MAJOR-CLASS: GRAMMAR
    GRAMMAR-CLASS: NOMINAL
  REST: NIL
```

The class `parse-complements` is defined like this:

```
(define-class parse-complements ()
  (branch (== (<<) nil)
          (progn
            (parse first)
            (call rest 'parse-complements))))
```

Interpreting these forms first involves trying to unify the current term with `NIL`. If that succeeds, the parse is done. Otherwise the parser is called on the 'first' slot and the class is called recursively on the 'rest' slot. For the verb `"moves"`, the 'first' slot was instantiated as a `nominal` and hence the second form in the `branch` is pursued. If that succeeds, the parser is invoked on the 'rest' slot of the 'complements.' Since `"moves"` was defined as a `transitive-verb`, that slot is `NIL`, so the first branch of the `parse-complements` form succeeds, and the parser moves on. The class `parse-adjuncts` is similar except that any number of adjuncts can appear with any verb.

As I described in chapter 3 the complete definitions for words include forms for creating the referents of the constituents containing the words. In general, for nouns these entries simple indicate the class of the referent of the noun phrase containing the given noun. For verbs, the entries specify the class of the occurrence that is the referent of the verb group, and they also indicate how the constituents of the clause map their referents to the participant slots in the referent of the verb group.

The lexical entries for prepositions take a number of forms. In the most common set of cases, the preposition modifies the occurrence that is the referent of the verb group of the clause it is inside. For example here is the definition of one of the senses of `"in"`:

```
(define-word ("in" duration-adjunct) (preposition)
  (in (parent)
    (call (<< object referent) 'duration-quantity)
    (== (<<* (clause) verb-group referent duration)
        (<< object referent))))
```

In this notation, the name of the "sense" of the word is included after the word's string. The forms are interpreted in the `(parent)`, of the preposition, which will be a `prep-phrase`. The effect of this preposition is to require that the referent of its object be a `duration-quantity`. If it is, that referent is assigned to the 'duration' slot of the referent of the verb group of the main clause. Since the definition of the noun `"time"` is of the class `duration-quantity`, the phrase "in a time t" is parsed with this sense of the preposition.

A more complex case involves the preposition `"with"`. Recall from the demo chapter that quantities are often expressed with possessive constructions, in particular with expressions like "an object moves with a speed s". A special abstract term, called a "function application" is used to record the relation among the moving object (the 'owner'), the quantity (the 'value'), and the 'occurrence' in which the owner "has" that value.

In the case of "with," the referent of the object of the preposition is a quantity whose 'fapp' slot has to be filled in from other constituents of the clause. Here is the definition of this sense of `"with"`:

```
(define-word ("with" with-quantity) (preposition)
  (in (parent)
    (call (<< object referent) 'fapp-quantity)
    (== (<<* (clause) subject referent)
        (<< object referent fapp argument))
    (== (<<* (clause) verb-group referent)
        (<< object referent fapp occurrence))))
```

This sense of the preposition requires that its object be of class **fapp-quantity** which is a subclass of **quantity** that includes a 'function-application' slot. The next two forms fill in the values of that slot. The first form makes the 'argument' of the 'function-application' be the subject of the clause, and the second form makes the 'occurrence' slot of the 'function' application be the referent of the verb group of the clause.

7.2 The Discourse Event Handler

This first phrase of processing the text computes what might be called the "local semantics" for each constituent. This means that the only things that can influence the referent of the constituent, or the only thing that the constituent can influence, are those whose grammatical relation to the constituent can be completely predicted from the rules of grammar. The example of verbs illustrates this well. Since a **head-verb** will occupy the 'head' slot of the verb group of a clause, the definition for a verb can specify the referent of its parent verb group, and furthermore, can indicate that the referent of the subject of the clause in which it is a part be unified with a particular constituent slot in the referent of the verb group.

However not all of the occurrence structure for a given piece of text can be constructed this way because each sentence or clause can appear in a wide variety of discourse contexts, and how the occurrence structure contributed by the sentence is handled depends on the specifics of the context where the sentence or clause appears.

The simplest case of this involves "co-referring" expressions. In the first sentence of the source text, for example, the noun phrase "an object" in the 'if' clause of the sentence is to be associated with the same referent as the "owner" of the possessive noun phrase "its average speed" in the main clause. Furthermore the occurrence that is the referent of the verb group of the main clause is to be understood as the same occurrence which is associated with that average speed.

Another important case involves deciding what to do once the system has processed a whole sentence or clause. Some sentences, like the first one in the source text, are complete statements of a physical definition or law, and should be recorded as such. Other sentences in presentation passages require that a step in a formula be verified. Many of the sentences in presentation and problem passages contribute

a number of details to an occurrence description whose ultimate job is yet to be determined. Finally, some sentences are in the form of questions or imperatives that indicate a problem to be solved.

These situations are dealt with by a subsystem of Sagredo called the "discourse event handler." The discourse event handler is invoked after each constituent is parsed, and, like the parser, is organized around the definition of unification form "methods" that determine how the system responds to each discourse event. There are three ways that discourse events get created:

1. In the course of the pre-processing of the LATEX source file, certain constituents are marked as being associated with discourse events when the parser gets to them. There are two cases of this:

   (a) When the pre-processor gets to the LATEX command indicating a stretch of emphasized text, for example, all of the words in the stretch of text are associated with an 'emphasized' discourse event. These words will be recorded and, in general, used as the name for newly defined classes or rules.

   (b) When a formula or equation or quantity is parsed by one of the special-purpose parser, a discourse event is associated with the token that will create a constraint between the entity and the referent of the term containing it.

2. Many lexical items are defined to create discourse events when they are used in grammatical constituents. As will be seen in the next section, the determiner of a noun phrase determines whether the noun phrase introduces a new participant or will direct the system to locate one that was previously introduced. Also, "expository" words like "defined" and "substitute" and so forth will directly introduce the discourse events associated with performing the actions they indicate.

3. Some grammatical constituents create discourse events when they are processed. The simplest case of this involves questions, which introduce a discourse event that begins the search for the answer. In general, after each constituent is parsed, a number of more or less *ad hoc* rules are run on it to determine if it satisfies specific tests. If so, appropriate discourse events are introduced.

The remaining sections of this chapter describe the ways that most of the discourse events that Sagredo deals with are recognized and handled.

7.3 Co-Reference

A large amount of work has been done in linguistics and computational linguistics on the problem of co-reference. (See the discussion in chapter 11 for references.) This problem arises because, obviously, multiple sentences in a discourse tend to be about the same things. The problem of co-reference can be succinctly stated as the problem of determining which of the expressions in a stretch of extended discourse are about the same things. A commonly studied case is that of pronoun anaphora, for example:

> An object starts from rest. It goes three feet in the first second.

In these sentences the word "it" in the second sentence refers to the same thing as the phrase "an object" in the first sentence. But the problem happens even without pronouns:

> A riverboat is traveling against the current. The boat is moving at a speed of 10 miles per hour.

In this sentence the phrase "the boat" refers to the same thing as "a riverboat."

Verb groups also introduce participants that can be later referred to. Consider:

> A cow walks 2 miles in 5 hours. What is the average speed of the cow for this walk?

Here the noun phrase "this walk" refers to the occurrence introduced by the phrase "walks".

Verb groups and quantity expressions can also refer to previously introduced entities. For example in

> A boomerang is moving due east. Its constant speed is 10 meters per second.

the phrase "constant speed" creates a 'function-application' structure that contains an 'occurrence' of class `motion`. This motion occurrence is the same one introduced by the verb "moving" in the first sentence. Another case is where a verb group refers to an occurrence created indirectly by an earlier verb group, as in

> A ball is thrown straight up. It comes down 3 seconds later.

In this case the occurrence described by "comes down" must be understood as being related to the "thrown" occurrence. In particular, from section 4 of the kinematics source text the program has learned that when something is thrown up, it eventually comes down. So the occurrence "comes down" is understood as referring to the occurrence created as a result of the rules that get instantiated on the "thrown straight up" occurrence.

While it is easy to think of examples where locating co-referring expressions is arbitrarily difficult, no such instances come up in the kinematics source text. Physics problems are also similarly simple. Most problems are only a few sentences long and not much that is very interesting or difficult can happen in such a short space.

The simplest case involves noun phrases that refer to participants in occurrences. Sagredo maintains a list of referents that have been introduced in the current stretch of discourse. "Indefinite" noun phrases (i.e., with the determiner "a") add new referrers to this list. "Definite" noun phrases (i.e., nominal pronouns like "it" and "she", or noun phrases with the determiner "the") direct the system to locate one of the terms in the referent list and unify the referent of the current noun phrase with it. If this succeeds, the processing continues. If not, the previously introduced referent is considered. For example in the sentences:

> A bus is moving with a speed of 10 miles per hour. It begins to slow down.

The first noun phrase "a bus" is indefinite so it adds a new entry to the referents list. The second indefinite noun phrase in the first sentence "a speed ..." adds another entry. The pronoun "it" in the second sentence triggers the a discourse event that directs the system to find a co-referring expression. The word "it" is defined such that its referent must be a subclass of `object`. The co-reference mechanism looks at the referrer introduced most recently. However this one, is a subclass of `quantity`—not `object`—so the unification fails. The second referrer tried is the referent of "the bus" and this one succeeds.

This simple heuristic would not work if more than one object of the same class were available for unification. It would fail if the correct co-referent were not the *last* such object to be introduced. So in the example

> A box is being carried on a cart. It begins to slide off.

The first sentence introduces two referrers that inherit from `object`. The correct referent of the word "it" in the second sentence is the first referrer ("a box"), however the heuristic described above would unify the referent of "it" with the referent of "a cart."

Luckily this situation does not arise in the source text or any of the problems. (It *could* be handled by an extension to the mechanism for matching occurrences, which is described below, but it isn't.) The only case where more than one potentially matching referring expressions exist in a single problem happens in section 4:

> A stone is dropped into the water from a bridge 144 ft above the water. Another stone is thrown vertically down 1.0 s after the first is dropped. Both stones strike the water at the same time. What was the initial speed of the second stone?

Although the expression "the second stone" ought to direct the system to only consider the second stone, in fact the above described heuristic works for this case: since the second stone is the most recently introduced, it is the first candidate for co-reference, and so it is the one used.

Referents that are occurrence get a somewhat different treatment. There is no distinction corresponding to that between "definite" and "indefinite" noun phrases that applies to verb groups. A given verb group or a quantity expression may involve a reference to an occurrence, but one cannot determine from the sentence alone whether or not that occurrence is supposed to be the same as one that was previously introduced. The heuristic that Sagredo uses for dealing with expressions whose referents are occurrences is: "unify them all, except when you can't."

In the simplest cases this heuristic works because all of the expressions that refer to occurrences in fact refer to the same one. Consider:

> A bus is moving on a straight road. Its constant speed is 30 mi/hr. It goes 2 miles.

The verb group of the first sentence creates a `motion` occurrence. This one can't be merged with any other because there are no others to merge with. In the second sentence the quantity expression "its constant speed" creates a `motion` occurrence in the 'occurrence' slot of its function application term. According to the heuristic, this occurrence is merge with the `motion` occurrence of the first sentence. Thus this occurrence now has its 'theme,' 'path,' and 'constant-speed' slots filled. The third sentence creates another 'motion' occurrence. Once again, there is nothing stopping the system from unifying this with the referent of the verb group of the first sentence, so the effect is to add a 'distance' slot to that occurrence.

There are a number of ways that the merging of occurrences could be blocked. The simplest case is where they just don't unify. Consider the following sentence:

> A bus is moving on a straight road. A truck begins moving on the same road.

The verb groups of both these sentences refer to a `motion` occurrence. However the 'theme' slot of the first sentence is of class `bus` while the 'theme' slot of the second sentence is of type 'truck' and so the two occurrences can't unify.

Another case where occurrences can't unify is where some temporal relation is stated to hold among the occurrences that can't be satisfied by a single occurrence. Reasoning about the temporal relations between and among occurrences is handled by a special mechanism that will be discussed below. But this mechanism would prevent the two occurrences in the following sentence from merging:

> A bus moves 3 miles in 10 minutes. In the next 5 minutes it goes 4 miles.

Here the phrase "in the next 5 minutes" indicates that occurrence designated by the verb group of the second sentence begins immediately after that designated by the verb group of the first sentence.

As mentioned above, occurrences can be introduced by rules that the system learns. The verb "thrown" for example sets in motion a chain of rule applications that eventually introduce a `motion` occurrence where the path is downward. All such introduced occurrences are available for the merge heuristic.

All of these occurrences are also available for straightforward noun phrase co-reference. So after a sentence like

> A cow walks 30 yards in 10 minutes.

The phrase "the walk" will locate the referent of "walks."

The heuristics for finding co-referring expressions are based on the idea that a story, though potentially interpretable in a number of ways, is usually to be given the *simplest* interpretation compatible with the meanings of the individual expressions. Unless specifically indicated, any expressions which could refer to the same things or to the same events are taken to do so. There are notorious problems with defining a notion of "simplicity" but it would seem to make sense that an interpretation heuristic for extended discourse would begin with this idea. Other proposals have been made along these lines, in particular the notions of "coherence" and "cohesion" discussed by Hobbs [156].

7.4 Intensionalize

Most of what Sagredo learns involves building up the definition of a class or a rule from the set of terms created by sentences of the presentation text. Most of the work involved in creating these definitions is done by the function `intensionalize`. The name is based on the philosophical distinction between the "extension" and the "intension" of a concept. The former is the set of all objects in the world what satisfy the concept, or perhaps a prototypical member of that set.

The notion of an "intension" is less clear. It is variously taken to be or involve a description or some other sort of characterization of the concept of of the objects that satisfy the concept. For Sagredo, a class definition or a rule characterizes a set of terms that would be generated were that class definition interpreted. A specific term can thus serve as an element of the extension of a set of isomorphic terms, all of which have the same "intension," namely the class definition that would generate them. The function `intensionalize` is defined to take a given term and produce a set of equations that would, when interpreted, construct a term isomorphic to the given one.

Having read the first clause of the first sentence of the source text, Sagredo has constructed the following occurrence:

```
#<TERM REFERENT [950]>
 CLASS:
  MAJOR-CLASS: OCCURRENCE
  OCCURRENCE-CLASS: MOTION
  TIME-CLASS: INTERVAL
 THEME: [351]
  CLASS:
   MAJOR-CLASS: OBJECT
 DISTANCE [707]:
  CLASS:
   MAJOR-CLASS: QUANTITY
   QUANTITY-CLASS: LENGTH
   LENGTH-CLASS: DISTANCE
 DURATION: [60]
  CLASS:
   MAJOR-CLASS: QUANTITY
   QUANTITY-CLASS: TIME
   TIME-CLASS: DURATION
```

There are also two constraints in effect, associating the 'distance' and 'duration' slots of this term with variables.

The main clause of the first sentence adds a slot named 'average-speed' to this occurrence and adds a constraint expressing its relation to the 'distance' and 'duration' values. The use of the expression "is defined to be" in that sentence triggers a 'define-quantity' discourse event in which the `intensionalize` function is called to create a definition of the new class. The definition it produces is:

```
(define-class average-speed (speed fapp-quantity)
  (== (<<) (<< fapp occurrence average-speed))
  (== (<< class speed-class) 'average-speed)
  (in fapp
    (== (<< occurrence theme) argument)
    (in occurrence
      (call (<<) 'motion 'interval)
      (call distance 'distance-quantity)
      (call duration 'duration-quantity)))
  (constraints
   (equal (<<)
          (quotient (<< fapp occurrence distance)
                    (<< fapp occurrence duration)))))
```

The main operation of the `intensionalize` function is fairly simple. It is straightforward to collect a list of equations whose solution would yield a given term. One merely has to descend through the term, keeping track of sequence of slot names that leads to a specific term. If that term is unified with a value, an equation unifying the value with the path could be produced; if the term is unified with another term, an equation unifying those terms could be produced; otherwise, continue through the name-value mappings of the current term.

This algorithm will yield a set of equations all of whom use the `==` operator, either on two `<<` paths (in the case of reentrant mappings) or on a path and a value. But this simple algorithm does not take advantage of the fact that many of those equations are the result of various classes being called in the course of the construction of the term. In particular, this means that any abstraction associated with using class names is lost. In the above example, the system would generate equations like:

```
(== (<< fapp occurrence duration class major-class) 'quantity)
(== (<< fapp occurrence duration class quantity-class) 'time)
(== (<< fapp occurrence duration class time-class) 'duration)
```

When it would be much more clear what was going on if the single equation:

```
(call (<< fapp occurrence duration) 'duration-quantity)
```

were generated.

So most of the details of the `intensionalize` function involve its trying to make maximal use of the set of already-defined classes. This process is aided by the interpreter for unification forms, which records any classes instantiated on a term by the `call` operator. When `intensionalize` begins collecting forms to create a definition, it first locates the classes that were instantiated on each term. The program also attempts to keep the definitions as simple as possible by using the shorter of multiple possible path expressions to the same term, and making liberal use of `in` to group forms that use the same subpaths.

In certain discourse contexts, such as a sentence that includes a clause introduced by the word "when," the resultant discourse event causes a rule to be generated instead of a simple class. In such cases, `intensionalize` is used twice: once to produce the condition forms of the rule, and the second time to produce the action forms of the rule.

If there are any formula constraints (see below) in effect when a new quantity or rule is defined, expressions corresponding to those constraints are collect up and stored in a `constraints` form on the definition of the rule or class. This involves locating, for each variable, a path expression that leads to the term that that variable is associated with. Constraints that represent formulas are then converted to forms that create instantiated versions of those formulas.

7.5 Algebra

The algebra system of Sagredo is used to do all of the mathematical reasoning involved in following the presentations and doing the problems. Algebraic formulas are represented as various kinds of "constraints" that are used to determine the values of terms whose referents are quantities.

Quantity Expressions

Physical quantities are expressed as a LISP list:

(*value numerator-units denominator-units*)

The *value* is a number. The *numerator-units* and *denominator-units* are lists of zero to two symbols that represent basic units. A special parser reads strings from the source text and computes quantity expressions of this form. Here are some strings and the quantity expressions that get computed for them:

2 miles	`(2 (MILE) ())`
55 mi/hr	`(55 (MILE) (HOUR))`
7 feet per second	`(7 (FOOT) (SECOND))`
9.8 m/s^2	`(9.8 (METER) (SECOND SECOND))`

The quantity parser also determines for each quantity the "dimension" of the resultant quantity. This is used to determine the class of the referent of the expression containing the quantity. For example in the prepositional phrase "in 5 minutes" the object gets parsed into the expression `(5 (MINUTE) ())` which is determined to be a `duration-quantity`. This information is used by the subsequent analysis to determine the proper sense of the preposition "in" to use. Thus the proper contributions of the "in" prepositional phrases in the following two sentences can be determined:

The bus came to a stop in 5 seconds.
The bus came to a stop in 30 feet.

In the first sentence the quantity is a `duration-quantity` and so the sense of the preposition "in" that sets the 'duration' slot of the clause is used. In the second sentence the object is a `distance-quantity` and so the sense of the preposition that sets the 'distance' slot of the clause is used.

Arithmetic operations on quantities must take their units into account. The function `convert-units` takes a given quantity and a desired units specification and applies the appropriate conversion factors to compute the new quantity. For example:

```
(CONVERT-UNITS '(55 (MILE) (HOUR)) '((METER) (SECOND)))
==>
(24.6 (METER) (SECOND))
```

The function **reduce-quantity** produces a canonical representation for a quantity expression. This means that the numerator and denominator expressions are as short as possible. For example:

```
(REDUCE-QUANTITY 10 '(MILE MINUTE) '(SECOND))
==>
(600 (MILE) ())
```

Each of the arithmetic operators is associated with a **:quantity-operator** property that computes the arithmetic result. For example the **:quantity-operator** function for the '+' converts the units of its first argument to those of its second argument. It then creates a new quantity whose value is the sum of the value of the converted first argument and the value of the second argument, and whose units are the same as its second argument.

The **:quantity-operator** for quotient is defined thus:

```
(defun quantity-quotient (q1 q2)
  (reduce-quantity
   (list
    (/ (car q1) (car q2))
    (append (cadr q1) (caddr q2))
    (append (caddr q1) (cadr q2)))))
```

The value of the result is simply the quotient of the values of the two quantities. The numerator units of the result are obtained by simply appending the numerator units of the numerator and the denominator units of the denominator. The denominator units of the result are obtained by appending the denominator units of the numerator and the numerator units of the denominator. The resultant quantity is then "reduced" to give the result in a canonical form.

For example computing how long it would take a body moving at a constant speed of 40 mi/hr to go 160 furlongs requires the computation of the quotient of these two quantities:

```
(160 (FURLONG) ())
(40 (MILE) (HOUR)
```

The first step is to create the quantity

```
(4 (FURLONG HOUR) (MILE))
```

this is then reduced to

```
(1/2 (HOUR) ())
```

Sagredo's quantity manipulation mechanisms do not worry about whether the arguments to a given expression make dimensional sense. Certainly a good physics student must learn to do so. However all of the expressions that Sagredo manipulates are dimensionally correct when they are read in and the mechanisms for manipulating expressions keep them so.

Formulas

A *formula* is represented as a LISP list with the first element of the list representing the mathematical operator and the rest of the elements of the list either containing quantity expressions, pure numbers, or other formula expressions. A *variable* is a special case of a formula expression, it consists of a list whose first element is the symbol `variable` followed by a sequence of strings that identify that variable. A special parser computes formula expressions from the input text. Here are some strings and the formula expressions that get computed for them:

s	`(VARIABLE "s")`
s_{av}	`(VARIABLE "s" "av")`
d/t	`(QUOTIENT (VARIABLE "d") (VARIABLE "t"))`

Constraints

Quantities and formulas are associated with the referents of terms via "constraints." There are four kinds of constraints: *quantity* constraints associate quantity expressions with terms; *formula* constraints associate the parsed versions of formulas with terms; *variable* constraints associate variable expressions with terms; and *operator* constraints express mathematical relation among terms.

An operator constraint includes an expression like the formula in a formula constraint except that where a formula expression has variables, the operator constraint expression has terms. As a passage of presentation text is read, appositive expressions and other grammatical constituents associate various terms with variables and formulas. Once these pairings of variables and formulas with terms has been done, it is possible to "interpret" a given formula with respect to the pairings in effect to determine the operator constraint that the current pairings entail. This operator constraint will then be used in subsequent manipulations.

Operator constraints may also be introduced directly with the `constraints` unification form operator. For example an expression of the form

```
(constraints
 (equal (<< fapp value)
        (quotient (<< fapp occurrence distance)
                  (<< fapp occurrence duration))))
```

will, when interpreted with respect to a given term, introduce an operator constraint.

Manipulating Constraints

The program keeps track in any discourse situation the set of terms for which quantity constraints exist. Given a set of quantity constraints, some subset of the set of operator constraints in effect are "solvable." An operator constraint is solvable if

all of the values of its argument terms are known but one. If this is so, the quantity arithmetic functions described above are invoked to compute a value for the unknown term. This value is then recorded and may be used in subsequent computations. This process continues until none of the existing operator constraints can be solved.

The next step involves transforming the operator constraints according to the laws of algebra. Figure 7.2 lists the set of transformation rules used by Sagredo's algebra system. Any algebraic expression that matches the left hand side of one of these rules can be replaced with an expression corresponding to the right hand side. Some of the rules can only match whole equations, for example

$$\begin{aligned} x = y &\rightarrow y = x \\ x = \frac{y}{z} &\rightarrow xz = y \end{aligned}$$

Most of the rules can match any subexpression of an algebraic expression. For example, $0 + x$ can be replaced by x whenever it appears in an expression.

So if the simple solution method described above fails to find a value for a desired constraint, each unsolved operator constraint is processed with the rules in figure 7.2. Each of the patterns is compared with each subexpression in the constraint, and when one matches, a new operator constraint with that subexpression replaced with one corresponding to the right side of the rule is introduced. The system then attempts to solve this new constraint.

Formula Manipulation

Much of sections 3 and 4 of the source text involves deriving equations for particular kinematic situations. To follow such arguments, the program first interprets each equation against the current binding associations of variables and quantities with terms.

Following the derivations involves performing three kinds of operation:

1. Equations and formulas must be *located.* There are two ways that the text refers to equations: as definitions of quantities; or by the LaTeX `ref` command. Sagredo maintains a table of all known equations and how they were named. An expression like "the definition of average speed locates the constraint corresponding to the equation introduced by that quantity. An expression like "equation 3.4" locates the equation by name.

   Formulas are located within equations. A typical expression is: "the formula for $s_f - s_i$ from equation 3.2." Finding this formula requires that equation 3.2 is such that $s_f - s_i$ appears on one side of the $=$ sign. The formula to use is the expression on the other side.

$$
\begin{aligned}
x = y &\rightarrow y = x \\
xy &\rightarrow yx \\
x + y &\rightarrow y + x \\
\\
x(yz) &\rightarrow (xy)z \\
(xy)z &\rightarrow x(yz) \\
x + (y + z) &\rightarrow (x + y) + z \\
(x + y) + z &\rightarrow x + (y + z) \\
\\
x(y + z) &\rightarrow xy + xz \\
xy + xz &\rightarrow x(y + z) \\
x(y - z) &\rightarrow xy - xz \\
xy - xz &\rightarrow x(y - z) \\
\\
0 + x &\rightarrow x \\
x - 0 &\rightarrow x \\
x + y = x + z &\rightarrow y = z \\
x + x &\rightarrow 2x \\
0 - x &\rightarrow -1x \\
1x &\rightarrow x \\
x/1 &\rightarrow x \\
x/x &\rightarrow 1 \\
y\frac{1}{x} &\rightarrow \frac{y}{x}
\end{aligned}
$$

$$
\begin{aligned}
x\frac{y}{z} &\rightarrow \frac{xy}{z} \\
x = \frac{y}{z} &\rightarrow xz = y \\
\frac{xy}{xz} &\rightarrow \frac{y}{z} \\
\frac{xy}{x} &\rightarrow y \\
xy = xz &\rightarrow y = z \\
\\
xx &\rightarrow x^2 \\
x^2 &\rightarrow xx \\
x = y^2 &\rightarrow \sqrt{x} = y \\
\sqrt{x} = y &\rightarrow x = y^2 \\
x\sqrt{y} &\rightarrow \sqrt{x^2 y} \\
\sqrt{x^2} &\rightarrow x \\
\sqrt{x}^2 &\rightarrow x \\
\\
(x + y)(x - y) &\rightarrow x^2 - y^2 \\
x^2 - y^2 &\rightarrow (x + y)(x - y) \\
\\
2 \sin x \cos x &\rightarrow \sin 2x \\
\sin 2x &\rightarrow 2 \sin x \cos x \\
\sin^2 x + \cos^2 x &\rightarrow 1
\end{aligned}
$$

Figure 7.2: The transformation rules used by the algebra system.

2. A subformula may be *substituted* into an equation. This means that some subformula for a formula in the equation is to be replace with another subformula which was located somewhere else. An example is:

   Substitute the formula for $s_f - s_i$ from equation 3.2 into equation 3.1

   First the formula for $s_f - s_i$ in equation 3.2 must be found, as described above. Then equation 3.1 must be found, and $s_f - s_i$ must occur within it. Then the formula from equation 3.2 must be substituted into equation 3.1 wherever $s_f - s_i$ occurs.

3. One equation may be *converted* to another. Once the appropriate subformulas are located and substituted into the correct equations, the equations are typically converted into some simplified or otherwise canonical form. A simple bi-directional breadth-first search is performed with the "from" and "to" equations. The rules are run in a forward fashion on the "from" equation, and all equations derived from it via applications of the rules. The rules are run in a backward fashion on the "to" equation, and all equations derived from it. Once the same expression is found to have been derived from the "from" expression as from the "to" expression, the derivation is recorded.

7.6 Occurrence Temporal Relations

Sagredo contains two special mechanisms to deal with the temporal relations among occurrences. The first has to do with the creation of suboccurrences, the second has to do with quantification on suboccurrences.

Creating Suboccurrences

Expressions like "the next 5 minutes" and "at the beginning of the interval" create occurrences with a specific relation to some other occurrence. The first step in dealing with these expressions is to locate the the original occurrence. For the expression "the next 5 minutes" some already existing occurrence must be found. In the second example, the expression "the interval" is a simple definite noun phrase and the standard co-reference mechanism finds a referent for it.

The next step is to create a new occurrence whose properties are determined by the details of the expression. The first example yields an `interval` whose 'duration' is the quantity `(5 (MINUTE) ())`, while the second yields a `moment` occurrence.

Next, the relation between the new occurrence and the previous one is recorded. In the first example the new occurrence's `begin` slot contains the same moment as the previous occurrence's `end` slot. In the second example, the new moment is unified with the `begin` slot of the existing interval.

The new occurrences are related to 'super' occurrences. In the first example, both the previously existing occurrence and the new one have the same 'super' term, which extends over both of them. In the second example the new moment occurence's 'super' term is the existing term.

In cases like "the next five minutes," the suboccurrences mechanism posts two constraints that determine that the sum of the 'distance' and 'duration' slots of the suboccurrences is equal to the 'distance' and 'duration,' respectively, of the 'super' occurrence. So in a text like

A car moves 3 miles in 10 minutes. In the next 5 minutes it goes 2 miles.

The system will have introduced constraints that determine that the whole interval is 15 minutes long and the car moved a total of 5 miles.

Finally, any rules that apply to the new occurrences are run. A number of rules have patterns that look like this:

```
((SUB
  ...)
 (SUPER
  ...))
```

Such rules are run whenever a new suboccurrence is created or is determined to be related to some 'super' occurrence. The most important class of such rules will be discussed next.

'All-Subs' Quantification

One of the standard difficult issues in natural language that did not present a great deal of difficulty in the design of Sagredo is that of quantification. None of the problems describes a quantified situation. All of the noun phrases in the text of the problems describe specific concrete participants, usually individuals. And all of the presentations describe universally quantified relations among occurrence classes or among quantity values. In such cases the rules don't really have to deal with quantification as such at all. They can just be set up to run on any occurrence and when they match they elaborate the structure as indicated. All such rules correspond to first order sentences of the form:

$$\forall(x)\,[P(x) \rightarrow Q(x)]\,.$$

Such rules can just be set up to test the predicate P. Whenever an occurrence is found that satisfies P, it is elaborated to now satisfy Q.

A class of cases where this won't work involve rules which would be represented in first order logic as:

$$\forall(x)\,[P(x) \rightarrow \forall(y)R(x,y)]$$

The problem here is that the right hand side of the rule is now a rule in its own right: it asserts something true of all objects in the domain, namely that they now satisfy the relation R to whatever objects satisfied P. This kind of reasoning is difficult for the approach I am taking because the conclusions of some inferences are now to be taken as rules themselves, not just as pieces of structure.

A similar problem occurs when the implication sign goes in the other direction:

$$\forall(x)\,[P(x) \leftarrow \forall(y)R(x,y)]\,.$$

To apply this rule the system would have to test if all objects in the domain satisfy the relation R to some object. If so, the system could assert P of that object. This is not inconsistent with the approach taken in Sagredo but it would involve a lot of work checking those relations.

Unfortunately such rules come up in the kinematics source text. The following sentence is in the second paragraph of section 1:

> An object moves with a *constant speed s* during an interval when its average speed equals s for all subintervals of the interval.

(This definition, by the way, is equivalent to the one given by Galileo.[1]) It could be represented (schematically) in logic as:

$$\forall(x)\,\{\text{constant}-\text{speed}(x,s) \leftrightarrow \forall(y)\,[\text{subinterval}(x,y) \;\rightarrow\; \text{average}-\text{speed}(y,s)]\}$$

Which is a biconditional of the offensive kind under discussion. The same pattern is used to define constant acceleration.

As mentioned, this rule would seem to require the following work: For any motion occurrence about which it was known that it was a 'constant-speed' occurrence, the system would have to be able to infer that if any other interval were a subinterval of the first, it would be have the same average speed. And furthermore, if there were an interval such that all of its subintervals had the same average speed, the system would have to be able to conclude that the interval had a constant speed equal to the average speed of all of its subintervals.

Rather than implement a general solution to this problem, I deal with it in a very specific and incomplete way. A special 'all-subs' quantification is introduced which holds only when some property of an interval entails that some other property holds of all of its subintervals. A special mechanism handles the forward chaining inferences: 'All-subs' assertions are represented as rules. The suboccurrence mechanism described above instantiates these rules on any subinterval of a 'super' interval that satisfies an 'all-subs' rule.

The backward inference: reasoning from properties of all subintervals of an interval to a property of the interval, is simply not done by Sagredo.

[1]In the *Two New Sciences*, p. 191. See page 43 of this report.

Chapter 8

History of the Program in Operation

This chapter presents a history of Sagredo's confrontation with with the kinematics source text. I will explain in some detail the steps the program goes through as it analyzes the text, constructs descriptions of the situations, records what it learns about physics, and solves the problems. As will be seen, just about every presentation sentence in the source text makes use of some new linguistic or expository device. The sentences of the problems are simpler—they just describe some situation and/or ask for the value of one of the quantities. The presentation style in this chapter will be similar to that in the initial demo in chapter 3, except that now I can make freer use of the terminology I have introduced since then.

I will present the steps and internal data-structures used by the program in great detail for the earlier sections of this history. In the later sections I will just discuss whatever new issues are raised.

The sections of this chapter correspond to the sections of the source text as printed in chapter 2.

8.1 Speed

Paragraph 1

The first sentence of the first paragraph is:

> If an object moves a distance d in a time t, its *average speed* for the interval is defined to be d/t.

This single sentence illustrates a fairly large number of issues explored in this report. This sentence was already used in chapter 7 to describe Sagredo's parser. As detailed there, the 'if' clause creates the following term as the referent of the verb group:

```
#<TERM REFERENT [950]>
 CLASS:
  MAJOR-CLASS: OCCURRENCE
  OCCURRENCE-CLASS: MOTION
  TIME-CLASS: INTERVAL
 THEME: [351]
  CLASS:
   MAJOR-CLASS: OBJECT
 DISTANCE [707]:
  CLASS:
   MAJOR-CLASS: QUANTITY
   QUANTITY-CLASS: LENGTH
   LENGTH-CLASS: DISTANCE
 DURATION: [60]
  CLASS:
   MAJOR-CLASS: QUANTITY
   QUANTITY-CLASS: TIME
   TIME-CLASS: DURATION
```

The following referring terms have been introduced. The referent of the subject of the sentence is:

```
#<TERM REFERENT [351]>
 MAJOR-CLASS: OBJECT
```

The referent of "moves":

```
#<TERM REFERENT [950]>
 TIME-CLASS: INTERVAL
 OCCURRENCE-CLASS: MOTION
 MAJOR-CLASS: OCCURRENCE
```

The referent of "a distance d":

```
#<TERM DISTANCE [707]>
 LENGTH-CLASS: DISTANCE
 QUANTITY-CLASS: LENGTH
 MAJOR-CLASS: QUANTITY
```

The referent of "a time t":

```
#<TERM DURATION [60]>
 TIME-CLASS: DURATION
 QUANTITY-CLASS: TIME
 MAJOR-CLASS: QUANTITY
```

The appositive constructions in these last two noun phrases have also introduced the following constraints:

```
#<VARIABLE-CONSTRAINT ...>
 #<TERM DISTANCE [707]>
 (VARIABLE "d")

#<VARIABLE-CONSTRAINT ...>
 #<TERM DURATION [60]>
 (VARIABLE "t")
```

The main clause of the first sentence is:

> its *average speed* for the interval is defined to be d/t.

The subject of this clause is a possessive construction whose determiner is the pronoun "its." This pronoun is defined to introduce a 'use-referrer' discourse event whose referent is of the class `object`. Only one such referrer has been introduced, by the subject of the 'if' clause: term 351. This referrer is used as the 'argument' slot of a 'function-application' term.

The rest of the slots in the function-application term are filled in by the rest of the noun phrase. The lexical entries for the words 'average' and 'speed' are such that the referent of the noun phrase is of the class `average-speed`. Sagredo begins with the following definition of the classes `speed` and `average-speed`:

```
(define-class speed (quantity)
  (== (<< class quantity-class) 'speed))

(define-class average-speed (speed)
  (== (<< class speed-class) 'average-speed)))
```

These definitions are the minimum needed to allow the sentences to be parsed properly.

The definite noun phrase "the interval" creates another 'use-referrer' discourse event that locates a referrer of class 'interval'. This is the term introduced by the verb "moves" of the 'if' clause, term 950.

Once the co-referring expressions have been found, the referent of the subject of the main clause ("its *average speed*") looks like this:

```
#<TERM REFERENT [36]>
 FAPP:
  CLASS:
   MAJOR-CLASS: FUNCTION-APPLICATION
  VALUE: [36]
  ARGUMENT: [351]
  OCCURRENCE: [950]
 CLASS:
  MAJOR-CLASS: QUANTITY
  QUANTITY-CLASS: SPEED
  SPEED-CLASS: AVERAGE-SPEED
```

Furthermore, the emphasis on the words 'average' and 'speed' has been noted. This emphasized phrase will be used as the name for the class being defined.

The rest of the sentence ("is defined to be d/t") is processed in two steps. The first step involves creating a constraint equating the value of the subject with the formula d/t:

```
#<FORMULA-CONSTRAINT ...>
 #<TERM REFERENT [36]>
 (QUOTIENT (VARIABLE "d") (VARIABLE "t"))
```

This discourse event is associated with the verb "be."

The verb "defined" takes the quantity which is the referent of its subject, and the formula which is introduced by its complement, and uses the `intensionalize` mechanism to create a new definition for the class `average-speed`:

```
(define-class average-speed (speed fapp-quantity)
  (== (<<) (<< fapp occurrence average-speed))
  (== (<< class speed-class) 'average-speed)
  (in fapp
    (== (<< occurrence theme) argument)
    (in occurrence
      (call (<<) 'motion 'interval)
      (call distance 'distance-quantity)
      (call duration 'duration-quantity)))
  (constraints
   (equal (<<)
          (quotient (<< fapp occurrence distance)
                    (<< fapp occurrence duration)))))
```

The first line of this definition unifies the average speed term with the contents of the 'average-speed' slot of the 'occurrence' slot of the function-application term. This convention will be followed in the definitions of all quantities.

The second line of the definition modifies the 'class' slot of the class 'average-speed' according to the standard conventions for classes.

The next form in the definition creates some substructure in the 'fapp' slot. As will be true for all of the quantities that Sagredo learns about, the 'theme' of the 'occurrence' slot will be unified with the 'argument' slot of the function-application. And inside the 'occurrence' slot, the 'distance' slot is a `distance-quantity`, and the 'duration' slot is a `duration-quantity`.

The most important part of this definition is the constraint it introduces. The expression for this constraint was determined by `intensionalize` by locating the terms to which the variable-constraints had been assigned, and then generating path expressions for those terms from the term being defined.

Paragraph 2

The first sentence of the second paragraph is:

> An object moves with a *constant speed s* during an interval when its average speed equals *s* for all subintervals of the interval.

This sentence illustrates another way to present the definition of a new quantity class. In this sentence the main clause is first. The subordinate 'when' clause introduces a condition that will be used by Sagredo to define a rule.

The 'with' prepositional phrase creates a function-application term whose 'argument' slot is the referent of the subject, whose 'value' slot is of class `constant-speed`, and whose 'occurrence' slot is the referent of object of the preposition 'during':

```
#<TERM REFERENT [873]>
 CLASS:
  MAJOR-CLASS: QUANTITY
  QUANTITY-CLASS: SPEED
  SPEED-CLASS: CONSTANT-SPEED
 FAPP:
  CLASS:
  VALUE: [873]
  ARGUMENT: [269]
   CLASS:
    MAJOR-CLASS: OBJECT
  OCCURRENCE: [37]
   CLASS:
    TIME-CLASS: INTERVAL
```

```
    OCCURRENCE-CLASS: MOTION
    MAJOR-CLASS: OCCURRENCE
   THEME: [269]
```

As with **average-speed**, Sagredo starts work with a minimal definition of the class **constant-speed**. At this point in the sentence, the appositive expression has introduced the constraint:

```
#<VARIABLE-CONSTRAINT ...>
 #<TERM REFERENT [873]>
 (VARIABLE "s")
```

And the emphasis on "constant speed" has been noted.

The subject of the 'when' clause of this sentence instantiates the newly-learned definition for **average-speed**. As in the first paragraph, the co-referrer for the 'argument' of 'its' is easily located. The verb 'equals' introduces a constraint between the value of the average speed and the formula s.

The prepositional phrase "for all subintervals of the interval" illustrates the use of 'all-subs' quantification. First of all the final noun phrase "the interval" triggers a 'use-referrer' discourse event, to locate the referent of 'moves' introduced by the first sentence. The "all subintervals of" expression creates a subinterval term for that interval, quantified as **all-subs**. Finally, the preposition 'for' attaches this quantified interval to the 'occurrence' slot of the function-application created by "its." The results are:

```
#<TERM REFERENT [419]>
 AVERAGE-SPEED: [860]
  FAPP:
   ARGUMENT: [269]
   CLASS:
   VALUE: [860]
   OCCURRENCE [419]:
  CLASS:
 THEME: [269]
 CLASS:
  MAJOR-CLASS: OCCURRENCE
  OCCURRENCE-CLASS: MOTION
  TIME-CLASS: INTERVAL
  QUANTIFIED: ALL-SUBS
 DISTANCE:
 DURATION:
 SUPER: [37]
```

And this constraint:

```
#<VARIABLE-CONSTRAINT ...>
 #<TERM REFERENT [860]>
 (VARIABLE "s")
```

Sagredo recognizes the form of the sentence as defining a quantity. Since this new quantity involves an 'all-subs' quantification, the definition will be more complicated than was the one for the class **average-speed**.

First of all, the class **constant-speed** is defined from term 37:

```
(define-class constant-speed (speed fapp-quantity)
  (== (<<) (<< fapp occurrence constant-speed))
  (== (<< class speed-class) 'constant-speed)
  (in fapp
    (== (<< occurrence theme) argument)
    (call occurrence 'motion 'interval))))
```

This definition isn't very interesting. It mostly just assures that the new definition obeys the conventions for fapp-quantities. Most of the work is done by the following rule:

```
(define-rule constant-speed-1
    ((sub
      (call (<<) 'motion 'interval))
     (super
      (call constant-speed 'constant-speed)))
  (== (<< sub theme) (<< super theme))
  (call (<< sub average-speed) 'average-speed)
  (== sub (<< sub average-speed fapp occurrence))
  (constraints (equal (<< sub average speed)
                      (<< super constant-speed))))
```

This rule is triggered whenever a subinterval is made of a motion interval that has a 'constant-speed' slot. The effect of the rule is to unify the 'theme' slots of the two intervals, and then to create an 'average-speed' slot of the subinterval and to introduce a constraint between the value of the subinterval's 'average-speed' slot and the superior interval's 'constant-speed' slot. Note, as I mentioned in chapter 7, that this rule doesn't really capture what the sentence says, because it doesn't allow for the determination that an object has a constant speed when its subintervals all have the same average speed. This rule only works when the system is told explicitly that an object has a constant speed.

The second sentence of this paragraph presents a specific case where an object has a constant speed:

An object at rest has a constant speed of zero.

The class **rest** has been defined to be a subclass of **motion**. The verb "has" illustrates yet another way to create a function-application term. In this case the referent of the subject of 'has' occupies the 'argument' slot, the complement of 'has' occupies the 'value' slot.

Since the quantity class **constant-speed** introduces an occurrence, the 'merge-occurrence' mechanism attempts to locate an occurrence to merge with it. This is located in the referent of "at rest." The result is the following term:

```
#<TERM REFERENT [193]>
 CLASS:
  MAJOR-CLASS: OCCURRENCE
  OCCURRENCE-CLASS: MOTION
  MOTION-CLASS: REST
  TIME-CLASS: INTERVAL
 THEME: [36]
  CLASS:
   MAJOR-CLASS: OBJECT
 CONSTANT-SPEED: [746]
  CLASS:
   MAJOR-CLASS: QUANTITY
   QUANTITY-CLASS: SPEED
   SPEED-CLASS: CONSTANT-SPEED
  FAPP:
   CLASS:
   VALUE: [746]
   OCCURRENCE: [193]
   ARGUMENT: [36]
```

The definition for 'has' also introduces a constraint when there is a prepositional phrase headed by 'of' giving a quantity. Here is the constraint thus introduced:

```
#<QUANTITY-CONSTRAINT ...>
#<TERM REFERENT [36]>
(0)
```

This sentence causes the system to generate the following simple rule:

```
(define-rule rest-1 ((int (call (<<) 'rest)))
  (call (<< int constant-speed 'constant-speed))
  (== (<< int constant-speed fapp occurrence) int)
  (constraints (equal (<< int constant-speed)
                      (quantity 0))))
```

This rule applies to any instantiation of the class **rest**, allows that such a term be extended to have a 'constant-speed' slot, and constrains the value of that slot to be zero.

Problem 1.1

The first problem is a very simple application of the definition of average speed.

> A cow walks 2 miles in 5 hours. What is the average speed of the cow for this walk?

The referent of the verb group of the first sentence looks like this:

```
#<TERM REFERENT [592]>
 CLASS:
  MAJOR-CLASS: OCCURRENCE
  OCCURRENCE-CLASS: MOTION
  TIME-CLASS: INTERVAL
 THEME: [81]
  CLASS:
   MAJOR-CLASS: OBJECT
   OBJECT-CLASS: COW
 DISTANCE: [975]
  EXPRESSION: (2 (MILE) ())
  CLASS:
   MAJOR-CLASS: QUANTITY
   QUANTITY-CLASS: LENGTH
   LENGTH-CLASS: DISTANCE
 DURATION: [346]
  EXPRESSION: (5 (HOUR) ())
  CLASS:
   MAJOR-CLASS: QUANTITY
   QUANTITY-CLASS: TIME
   TIME-CLASS: DURATION
```

The following quantity constraints have been introduced:

```
#<QUANTITY-CONSTRAINT ...>
 #<TERM DISTANCE [975]>
 (2 (MILE) ())

#<QUANTITY-CONSTRAINT ...>
 #<TERM DURATION [346]>
 (5 (HOUR) ())
```

The second sentence introduces two 'use-referrer' events, one for the cow, and one for the walk. In both cases only a single possible referrer is available. The new class **average-speed** is instantiated and its slots are filled in with these referring expressions:

```
#<TERM REFERENT [80]>
 CLASS:
  MAJOR-CLASS: QUANTITY
  QUANTITY-CLASS: SPEED
  SPEED-CLASS: AVERAGE-SPEED
 FAPP:
  CLASS:
  VALUE: [80]
  OCCURRENCE: [592]
  ARGUMENT: [81]
```

The `constraints` form in the definition of average speed is interpreted to yield this constraint:

```
#<OPERATOR-CONSTRAINT ...>
(EQUAL #<TERM REFERENT [80]>
       (QUOTIENT #<TERM DISTANCE [975]>
                 #<TERM DURATION [346]>))
```

Finally, the definition of the verb "is" sets up a constraint between the referent of the subject and the referent of the complement "what,"

```
#<OPERATOR-CONSTRAINT ...>
 (EQUAL #<TERM REFERENT [80]> #<TERM REFERENT [606]>)
```

the latter, term 606, is the target for the question.

The constraint mechanism can solve this problem easily. First of all, the quotient constraint can be run because the values of the distance term 975 and the duration term 346 are known. This yields the value of term 80. The simple `equal` constraint can then be run to yield the value for term 606:

```
(2/5 (MILE) (HOUR))
```

Problem 1.2

The second problem is also a simple application of the definition of average-speed, but asks for a different quantity.

> How far does a car go in 5 min if its average speed is 60 mi/hr?

This problem is presented in a single sentence consisting of a main clause and an embedded 'if' clause.

Once all of the co-referring expressions have been found, the following occurrence is constructed:

```
#<TERM REFERENT [449]>
 CLASS:
  MAJOR-CLASS: OCCURRENCE
  OCCURRENCE-CLASS: MOTION
  TIME-CLASS: INTERVAL
 THEME [779]:
  CLASS:
   MAJOR-CLASS: OBJECT
   OBJECT-CLASS: CAR
 DISTANCE: [606]
  CLASS:
   MAJOR-CLASS: QUANTITY
   QUANTITY-CLASS: LENGTH
   LENGTH-CLASS: DISTANCE
 DURATION: [323]
  EXPRESSION: (5 (MINUTE) ())
  CLASS:
   MAJOR-CLASS: QUANTITY
   QUANTITY-CLASS: TIME
   TIME-CLASS: DURATION
 AVERAGE-SPEED: [619]
  FAPP:
   ARGUMENT: [779]
   CLASS:
   VALUE: [619]
   OCCURRENCE: [449]
  CLASS:
```

This constraint is introduced by the 'in' prepositional phrase:

```
#<QUANTITY-CONSTRAINT ...>
 #<TERM DURATION [323]>
 (5 (MINUTE) ())
```

This constraint assigns a value to the referent of the complement of the verb "is" in the 'if' clause:

```
#<QUANTITY-CONSTRAINT ...>
 #<TERM REFERENT [893]>
 (60 (MILE) (HOUR))
```

The verb "is" introduces an equality constraint between its subject and its complement:

```
#<OPERATOR-CONSTRAINT ...>
 (EQUAL #<TERM AVERAGE-SPEED [619]> #<TERM REFERENT [893]>)
```

Since the value of term 893 was given, this constraint can be run immediately to yield a value for term 619.

The definition of average-speed creates this constraint:

```
#<OPERATOR-CONSTRAINT ...>
(EQUAL #<TERM AVERAGE-SPEED [619]>
       (QUOTIENT #<TERM DISTANCE [606]>
                 #<TERM DURATION [323]>))
```

This constraint is transformed by the algebra system to yield:

```
(EQUAL #<TERM DISTANCE [606]>)
       (PRODUCT #<TERM AVERAGE-SPEED [619]>
                #<TERM DURATION [323]>))
```

This last constraint can now be solved:

```
(5 (MILE) ())
```

Problem 1.3

> How long will it take a body moving with an average speed of 15 ft/sec to go a distance of 6000 ft?

The only really interesting thing about this problem is the use of the verb "take." One of the complements to this verb, namely: "how long," expresses the duration of the occurrence described by the other complement. Here is the occurrence term:

```
#<TERM REFERENT [123]>
 CLASS:
  MAJOR-CLASS: OCCURRENCE
  OCCURRENCE-CLASS: MOTION
  TIME-CLASS: INTERVAL
 THEME: [58]
  CLASS:
   MAJOR-CLASS: OBJECT
   OBJECT-CLASS: BODY
 DISTANCE: [565]
  EXPRESSION: (6000 (FOOT) ())
```

```
 CLASS:
DURATION: [816]
AVERAGE-SPEED: [292]
 EXPRESSION: (15 (FOOT) (SECOND))
 CLASS:
 FAPP:
```

Constraints:

```
#<QUANTITY-CONSTRAINT ...>
#<TERM AVERAGE-SPEED [292]>
 (15 (FOOT) (SECOND))

#<OPERATOR-CONSTRAINT ...>
 (EQUAL #<TERM AVERAGE-SPEED [292]>
        (QUOTIENT #<TERM DISTANCE [565]>
                  #<TERM DURATION [816]>))

#<QUANTITY-CONSTRAINT ...>
#<TERM DISTANCE [565]>
 (6000 (FOOT) ())
```

The target quantity in this case is term 816. Transformations to the quotient constraint yield:

```
(EQUAL #<TERM DURATION [816]>
       (QUOTIENT #<TERM DISTANCE [565]>
                 #<TERM AVERAGE-SPEED [292]>))
```

This is solved:

```
(400 (SECOND) ())
```

Problem 1.4

This problem is a bit more complex than the three handled so far. Each of them presented a single motion occurrence and asked for the values of one of the three quantities linked by the constraint introduced by the definition of average speed. This problem requires that more than one interval be considered. Here is the first sentence of the problem:

> A truck drives 200 miles in 3 hours.

Here is the referent of the verb group of this sentence:

```
#<TERM REFERENT [240]>
 CLASS:
  MAJOR-CLASS: OCCURRENCE
  OCCURRENCE-CLASS: MOTION
  TIME-CLASS: INTERVAL
 THEME: [385]
  CLASS:
   MAJOR-CLASS: OBJECT
   OBJECT-CLASS: TRUCK
 DISTANCE: [815]
  EXPRESSION: (200 (MILE) ())
  CLASS:
 DURATION: [530]
  EXPRESSION: (3 (HOUR) ())
  CLASS:
```

This sentence introduces these two quantity constraints

```
#<QUANTITY-CONSTRAINT ...>
 #<TERM DISTANCE [815]>
 (200 (MILE) ())

#<QUANTITY-CONSTRAINT ...>
 #<TERM DURATION [530]>
 (3 (HOUR) ())
```

The second sentence describes a more complex situation, and asks for the value of a quantity:

> What must its average speed be over the next 100 miles for its average speed to be 60 mi/hr for the entire trip?

The main clause of this sentence is an inverted "be" construction, with the complement "what" indicating that the target quantity is the value of the average speed for the interval. The possessive pronoun "its" locates term 385, the referent of "a truck" and unifies it with the 'argument' slot of the function-application inside the average-speed term.

The phrase "over the next 100 miles" will fill in the 'distance' slot of a `motion` interval, but it has to find the right such interval. The word "next" directs the suboccurrence mechanism to create a new interval and link it to the previously introduced interval, in this case term 240. This new interval is then used as the 'occurrence' slot of the function-application term inside the referent of "its average speed."

```
#<TERM AVERAGE-SPEED [752]>
 CLASS:
  MAJOR-CLASS: QUANTITY
  QUANTITY-CLASS: SPEED
  SPEED-CLASS: AVERAGE-SPEED
 FAPP:
  ARGUMENT: [385]
  CLASS:
  VALUE: [752]
  OCCURRENCE [134]
   AVERAGE-SPEED: [752]
   THEME: [385]
   CLASS:
   DISTANCE [858]:
    CLASS:
    EXPRESSION: (100 (MILE) ())
   DURATION [511]:
   BEFORE: [240]
   SUPER: [287]
    CLASS:
    DURATION: [460]
    DISTANCE: [593]
    THEME: [385]
    AVERAGE-SPEED: [731]
```

Occurrence term 134 is the one that was created in response to processing "over the next 100 miles," its 'before' slot contains occurrence term 240, which was the referent of the verb group of the first sentence. And term 287 is the 'super' of both these occurrence—it begins when term 240 begins and ends when term 134 ends.

The referent of "what" is term 742, this constraint equates its value with that of the referent of "its average speed":

```
#<OPERATOR-CONSTRAINT ...>
 (EQUAL #<TERM AVERAGE-SPEED [752]>
        #<TERM REFERENT [742]>)
```

This constraint is introduced by the quantity expression:

```
#<QUANTITY-CONSTRAINT ...>
 #<TERM DISTANCE [858]>
 (100 (MILE) ())
```

And this one by the definition of average speed:

```
#<OPERATOR-CONSTRAINT ...>
 (EQUAL #<TERM AVERAGE-SPEED [752]>
        (QUOTIENT #<TERM DISTANCE [858]>
                  #<TERM DURATION [511]>))
```

When the suboccurrence mechanism created the 'next' term 134, and the 'super' term 240, it also introduced constraints linking the 'distance' and 'duration' slots of the suboccurrences with the super occurrence:

```
#<OPERATOR-CONSTRAINT ...>
 (EQUAL #<TERM DURATION [460]>
        (SUM #<TERM DURATION [530]>
             #<TERM DURATION [511]>))

#<OPERATOR-CONSTRAINT ...>
 (EQUAL #<TERM DISTANCE [593]>
        (SUM #<TERM DISTANCE [815]>
             #<TERM DISTANCE [858]>))
```

The clause introduced by the word "for" sets the value of the average speed of "the entire trip." The adjective "entire" creates a referent that must have no 'super' occurrence, thus term 240 is found by the co-referrer mechanism. The definition of average-speed introduces this constraint for the average speed slot of term 240:

```
#<OPERATOR-CONSTRAINT ...>
 (EQUAL #<TERM AVERAGE-SPEED [731]>
        (QUOTIENT #<TERM DISTANCE [593]>
                  #<TERM DURATION [460]>))
```

These constraints are introduced by the phrase "to be 60 mi/hr":

```
#<OPERATOR-CONSTRAINT ...>
 (EQUAL #<TERM AVERAGE-SPEED [731]> #<TERM REFERENT [213]>)

#<QUANTITY-CONSTRAINT ...>
 #<TERM REFERENT [213]>
 (60 (MILE) (HOUR))
```

Term 213 is the the referent of the complement of the verb "be."

The target quantity is term 752, the average-speed of the 'next' occurrence. From the sum constraint, we can compute the value of term 953, the 'distance' slot of the 'super' occurrence:

```
(300 (MILE) ())
```

Since the system has values for the average speed of the 'super' occurrence, and its 'distance,' it can compute the value of its 'duration,' term 460:

```
(5 (HOUR) ())
```

Now the sum constraint among the 'duration' slots, the system can determine the value of term 511, the duration slot of the 'next' occurrence:

```
(2 (HOUR) NIL)
```

Now, since the 'distance' slot of this occurrence is known, the system can use the quotient constraint introduced by the definition of average speed to compute value of the average speed for the 'next' occurrence, which is the target quantity:

```
(50 (MILE) (HOUR))
```

Problem 1.5

This problem is the only one in this section that exercises the notion of constant speed.

An airplane flies at a constant speed for 100 miles.

Here is the referent of "a constant speed":

```
#<TERM CONSTANT-SPEED [979]>
 CLASS:
  MAJOR-CLASS: QUANTITY
  QUANTITY-CLASS: SPEED
  SPEED-CLASS: CONSTANT-SPEED
 FAPP:
  CLASS:
  VALUE: [979]
  ARGUMENT: [527]
   CLASS:
    MAJOR-CLASS: OBJECT
    OBJECT-CLASS: AIRPLANE
  OCCURRENCE: [14]
   CLASS:
    MAJOR-CLASS: OCCURRENCE
    OCCURRENCE-CLASS: MOTION
    TIME-CLASS: INTERVAL
   THEME: [527]
```

```
CONSTANT-SPEED: [979]
DISTANCE: [763]
 EXPRESSION: (100 (MILE) ())
 CLASS:
```

The next sentence describes a subinterval of the referent of the verb group of the first sentence:

In the first 10 minutes, it goes 20 miles.

The phrase "the first 10 minutes" triggers the creation of the suboccurrence of the occurrence from the first sentence. Since the latter is of class **constant-speed** the rule **constant-speed-1** is activated to create an 'average-speed' slot on the suboccurrence, and a constraint linking it to the value of the 'constant-speed' slot on the super interval. So here is the referent of the verb group of the second sentence:

```
#<TERM REFERENT [285]>
 CLASS:
  TIME-CLASS: INTERVAL
  MAJOR-CLASS: OCCURRENCE
  OCCURRENCE-CLASS: MOTION
 DURATION: [32]
  EXPRESSION: (10 (MINUTE) ())
  CLASS:
 AFTER:
  CLASS:
  BEFORE: [285]
  SUPER: [14]
  DURATION [912]:
  DISTANCE [805]:
  THEME [570]:
  AVERAGE-SPEED [737]:
 DISTANCE: [0]
  EXPRESSION: (20 (MILE) ())
  CLASS:
 THEME: [570]
 AVERAGE-SPEED [124]:
```

Note that the 'super' slot of this occurrence is the referent of the first sentence's verb group. The final sentence asks for the 'duration' slot of the original interval:

How long does the whole trip take?

This constraint was introduced by the first sentence:

```
#<QUANTITY-CONSTRAINT ...>
 #<TERM DISTANCE [763]>
 (100 (MILE) ())
```

These two were introduced by the quantity expressions in the second sentence:

```
#<QUANTITY-CONSTRAINT ...>
 #<TERM DURATION [32]>
 (10 (MINUTE) ())

#<QUANTITY-CONSTRAINT ...>
 #<TERM DISTANCE [0]>
 (20 (MILE) ())
```

This constraint is introduced when the `constant-speed-1` rule creates an average-speed slot on the suboccurrence:

```
#<OPERATOR-CONSTRAINT ...>
 (EQUAL #<TERM AVERAGE-SPEED [124]>
        (QUOTIENT #<TERM DISTANCE [0]>
                  #<TERM DURATION [32]>))
```

The next constraint is also introduced by the `constant-speed-1` rule. It equates the value of the average speed of the suboccurrence and the constant speed of the superior occurrence:

```
#<OPERATOR-CONSTRAINT ...>
 (EQUAL #<TERM AVERAGE-SPEED [124]>
        #<TERM CONSTANT-SPEED [979]>)
```

Finally, since any occurrence is a suboccurrence of itself, the super occurrence is itself given an 'average-speed' slot, and this constraint is introduced by the definition of `average-speed`:

```
#<OPERATOR-CONSTRAINT ...>
 (EQUAL #<TERM AVERAGE-SPEED [737]>
        (QUOTIENT #<TERM DISTANCE [805]>
                  #<TERM DURATION [912]>))
```

And this constraint links the 'average-speed' of the super occurrence with that occurrence's 'constant-speed':

```
#<OPERATOR-CONSTRAINT ...>
 (EQUAL #<TERM AVERAGE-SPEED [737]>
        #<TERM CONSTANT-SPEED [979]>)
```

This problem is solved by first computing the value of the average speed of the "first 10 minutes" suboccurrence. Since both the distance and duration of that occurrence are known, the value can be determined:

```
(2 (MILE) (MINUTE))
```

This value is then propagated to term 979, the constant speed of the 'super' occurrence. From there, the value is given to term 737, which is the average speed of the 'super' occurrence. Now that the average-speed and the distance are known, the 'duration' slot can be computed. This is the desired quantity:

```
(50 (MINUTE) ())
```

Paragraph 3

The first sentence of the final paragraph of this section defines a new term:

> A moving object has an *instantaneous speed* at every moment during its motion.

The referent of the phrase "instantaneous speed" in this sentence looks like this:

```
#<TERM INSTANTANEOUS-SPEED [948]>
 CLASS:
  MAJOR-CLASS: QUANTITY
  QUANTITY-CLASS: SPEED
  SPEED-CLASS: INSTANTANEOUS-SPEED
 FAPP:
  ARGUMENT: [195]
  OCCURRENCE: [183]
   CLASS:
    TIME-CLASS: MOMENT
    QUANTIFIED: ALL-SUBS
   INSTANTANEOUS-SPEED [948]:
   THEME: [195]
```

The definition of the class **instantaneous-speed** that results from this doesn't say much other than that its 'occurrence' slot must inherit from the class 'moment':

```
(define-class instantaneous-speed (speed fapp-quantity)
  (== (<< class speed-class) 'instantaneous-speed)
  (== (<<) (<< fapp occurrence instantaneous-speed))
  (in fapp
    (== argument (<< occurrence theme))
    (call occurrence 'moment))))
```

However since an 'all-subs' quantified occurrence was used as the referent of the 'at' prepositional phrase, the following rule is created:

```
(define-rule instantaneous-speed-1
    ((sub
      (call (<<) 'moment))
     (super
      (call (<<) 'motion 'interval)))
  (== (<< sub theme) (<< super theme))
  (call (<< sub instantaneous-speed) 'instantaneous-speed)
  (== sub (<< sub instantaneous-speed fapp occurrence)))
```

This rule simply introduces an 'instantaneous-speed' slot into any submoment of a motion interval, but says nothing about what its value will be. That is taken care of by the next sentence:

> The instantaneous speed of an object moving with a constant speed during an interval equals its constant speed at every moment during the interval.

Here is the referent of "moving with a constant speed" from this sentence:

```
#<TERM REFERENT [821]>
 CONSTANT-SPEED [205]:
  CLASS:
  FAPP:
   CLASS:
   VALUE: [205]
   OCCURRENCE: [821]
   ARGUMENT [137]:
 CLASS:
  MAJOR-CLASS: OCCURRENCE
  OCCURRENCE-CLASS: MOTION
  TIME-CLASS: INTERVAL
 THEME: [137]
```

Here is the referent of "the instantaneous speed":

```
#<TERM REFERENT [546]>
 FAPP:
  OCCURRENCE: [188]
   INSTANTANEOUS-SPEED: [546]
   THEME: [137]
   CLASS:
    TIME-CLASS: MOMENT
    QUANTIFIED: ALL-SUBS
  ARGUMENT: [137]
  VALUE: [546]
  CLASS:
 CLASS:
  MAJOR-CLASS: QUANTITY
  QUANTITY-CLASS: SPEED
  SPEED-CLASS: INSTANTANEOUS-SPEED
```

And the following constraint has been created by the verb "equals:"

```
#<OPERATOR-CONSTRAINT ...>
(EQUAL #<TERM REFERENT [546]>
       #<TERM CONSTANT-SPEED [205]>)
```

From these structures, Sagredo creates another rules for all submoments of motion intervals:

```
(define-rule constant-speed-2
    ((sub (call (<<) 'moment))
     (super
      (call constant-speed 'constant-speed)))
  (== (<< sub theme) (<< super theme))
  (call (<< sub instantaneous-speed) 'instantaneous-speed)
  (== sub
      (<< sub instantaneous-speed fapp occurrence))
  (constraints (equal (<< sub instantaneous-speed)
                      (<< super constant-speed))))
```

This rule is a bit more restrictive than the last one. It applies only when the 'super' interval contains a 'constant-speed' slot. If so, it creates and links up the 'instantaneous-speed' slot on the submoment, and adds a constraint that the 'instantaneous-speed' is equal to the constant speed of the 'super' occurrence.

The final two sentences of this section define two new kinds of quantities.

> The instantaneous speed of an object at the beginning of an interval is called its ***initial speed*** for the interval. The object's instantaneous speed at the end of an interval is called its ***final speed*** for the interval.

The 'initial-speed' of an interval is constrained to be equal to the 'instantaneous-speed' of the 'start' moment of the interval:

```
(define-class initial-speed (speed)
  (== (<< class speed-class) 'initial-speed)
  (in fapp
    (== argument (<< occurrence theme))
    (in occurrence
      (call (<<) 'motion 'interval)
      (in end
        (call instantaneous-speed 'instantaneous-speed)
        (== (<<)
            (<< instantaneous-speed fapp occurrence)))))
  (== (<<) (<< fapp occurrence initial-speed))
  (constraints
   (equal (<<)
          (<< fapp occurrence end instantaneous-speed))))
```

The 'final-speed' of the interval is defined in a like manner:

```
(define-class final-speed (speed)
  (== (<< class speed-class) 'final-speed)
  (in fapp
    (== argument (<< occurrence theme))
    (in occurrence
      (call (<<) 'motion 'interval)
      (in end
        (call instantaneous-speed 'instantaneous-speed)
        (== (<<)
            (<< instantaneous-speed fapp occurrence)))))
  (== (<<) (<< fapp occurrence final-speed))
  (constraints
   (equal (<<)
          (<< fapp occurrence end instantaneous-speed))))
```

8.2 Acceleration

Paragraph 1

This section begins by defining a new class of occurrence:

> A moving object whose instantaneous speed changes over an interval is said to be *accelerating* during that interval.

From the definition of instantaneous speed in the last section, the system computes the following referent of the phrase "whose instantaneous speed:"

```
#<TERM REFERENT [746]>
 CLASS:
  MAJOR-CLASS: QUANTITY
  QUANTITY-CLASS: SPEED
  SPEED-CLASS: INSTANTANEOUS-SPEED
 FAPP:
  CLASS:
  VALUE: [746]
  ARGUMENT: [269]
   CLASS:
    MAJOR-CLASS: OBJECT
  OCCURRENCE:
   THEME: [269]
   CLASS:
    TIME-CLASS: MOMENT
```

The verb "changes" creates a `quantity-change` occurrence. Such an occurrence is represented by having a 'changing-quantity' slot that contains a quantity. In this case, the quantity is the instantaneous speed, represented by term 746:

```
#<TERM REFERENT [218]>
 CLASS:
  MAJOR-CLASS: OCCURRENCE
  OCCURRENCE-CLASS: MOTION
  TIME-CLASS: INTERVAL
 THEME: [269]
 CHANGING-QUANTITY: [746]
```

Since the instantaneous speed is changing all during the occurrence, it is modified to be quantified 'all-subs.' From this occurrence structure, the system defines the class `accelerate`:

```
(define-class accelerate (motion interval)
  (== (<< changing-quantity fapp argument) theme)
  (in changing-quantity
    (call (<<) 'instantaneous-speed)
    (call (<< fapp occurrence) 'all-subs))))
```

The verb "accelerate" had already been defined to create an instance of the class **accelerate**. The class **accelerate** had a minimal definition: it just recorded that it was an interval occurrence.

The second sentence of this paragraph defines average acceleration in a manner similar to that which was used to define average speed:

> If the instantaneous speed of an object moving along a straight line changes from s_i to s_f during a time t, its *average acceleration* for the interval is defined to be $(s_f - s_i)/t$.

This sentence also has 'changes' as the main verb, so the occurrence will have a 'changing-quantity' slot. The prepositions 'from' and 'to' set the initial and final values of the quantity that is changing. So here is the referent of the verb group of the 'if' clause:

```
#<TERM REFERENT [950]>
 CLASS:
  MAJOR-CLASS: OCCURRENCE
  OCCURRENCE-CLASS: MOTION
  TIME-CLASS: INTERVAL
 THEME [620]:
 CHANGING-QUANTITY [413]:
  FAPP:
   ARGUMENT: [620]
   CLASS:
   VALUE: [413]
   OCCURRENCE:
    INSTANTANEOUS-SPEED: [413]
    THEME: [620]
    CLASS:
     TIME-CLASS: MOMENT
     QUANTIFIED: ALL-SUBS
  CLASS:
   MAJOR-CLASS: QUANTITY
   QUANTITY-CLASS: SPEED
   SPEED-CLASS: INSTANTANEOUS-SPEED
```

```
PATH:
START [590]:
 INSTANTANEOUS-SPEED [240]:
 CLASS:
 THEME: [620]
END [189]:
 INSTANTANEOUS-SPEED [515]:
 CLASS:
 THEME: [620]
DURATION: [607]
```

The 'if' clause also introduces these constraints:

```
#<VARIABLE-CONSTRAINT ...>
 #<TERM INSTANTANEOUS-SPEED [240]>
 (VARIABLE "s" "i")

#<VARIABLE-CONSTRAINT ...>
 #<TERM INSTANTANEOUS-SPEED [515]>
 (VARIABLE "s" "f")

#<VARIABLE-CONSTRAINT ...>
 #<TERM DURATION [607]>
 (VARIABLE "t")
```

The main clause makes use of the occurrence structure from the 'if' clause, the above constraint, and constraint introduced by the clause complement:

```
#<OPERATOR-CONSTRAINT ...>
 (EQUAL #<TERM AVERAGE-ACCELERATION [713]>
        (QUOTIENT
          (DIFFERENCE #<TERM INSTANTANEOUS-SPEED [515]>
                      #<TERM INSTANTANEOUS-SPEED [240]>)
          #<TERM DURATION [607]>))
```

And the new quantity **average-acceleration** is defined:

```
(define-class average-acceleration
    (acceleration fapp-quantity)
  (== (<<)
      (<< fapp occurrence average-acceleration))
  (== (<< class acceleration-class) 'average-acceleration)
  (in fapp
    (== (<< occurrence theme) argument)
    (in occurrence
      (call (<<) 'accelerate)
      (== (<< start theme) theme)
      (== (<< end theme) theme)
      (in start
        (call instantaneous-speed 'instantaneous-speed)
        (== (<< instantaneous-speed fapp occurrence)
            (<<)))
      (in end
        (call instantaneous-speed 'instantaneous-speed)
        (== (<< instantaneous-speed fapp occurrence)
            (<<)))
      (call duration 'duration-quantity)
      (call path 'straight 'line)))
  (constraints
   (equal (<<)
          (quotient
           (difference
            (<< fapp occurrence end instantaneous-speed)
            (<< fapp occurrence start instantaneous-speed))
           (<< fapp occurrence duration)))))
```

Paragraph 2

The first sentence of this paragraph defines constant acceleration in the same way that constant speed was defined in the previous section.

> An object moving along a straight line has a *constant acceleration* a during some interval of time when its average acceleration is equal to a for all subintervals of that interval.

As in the previous section, the system determines a definition for the new class and a rule that is invoked whenever a subinterval is created on an interval that has a 'constant-acceleration' slot.

```
(define-class constant-acceleration
    (acceleration fapp-quantity)
  (== (<<)
      (<< fapp occurrence constant-acceleration))
  (== (<< class acceleration-class)
      'constant-acceleration)
  (in fapp
    (== (<< occurrence theme) argument)
    (call occurrence 'accelerate))))

(define-rule constant-acceleration-1
    ((sub
      (call (<<) 'motion 'interval))
     (super
      (call constant-acceleration 'constant-acceleration)))
  (== (<< sub theme) (<< super theme))
  (call (<< sub average-acceleration) 'average-acceleration)
  (== (<< sub average-acceleration fapp occurrence) sub)
  (constraints (equal (<< sub average-acceleration)
                      (<< super constant-acceleration))))
```

The next sentence defines two ways to say that something has a constant acceleration: the first with a prepositional phrase, the second with an adverb:

> An object moving with a constant acceleration is said to accelerate at a *constant rate*, or to accelerate *uniformly*.

The first conjunct simply defines the class `constant-rate` as if it were a new physical quantity. `Intensionalize` discovers that this new class is exactly the same as the class `constant-acceleration`:

```
(define-class constant-rate (constant-acceleration))
```

The second conjunct defines a new subclass of `accelerate`. Since `accelerate` inherited from `motion`, the new class is simply `accelerate` plus an appropriately connected `constant-acceleration` slot:

```
(define-class uniformly-accelerate (accelerate)
  (call constant-acceleration 'constant-acceleration)
  (== (<< constant-acceleration fapp occurrence) (<<)))
```

The last sentence of the first paragraph adds a rule like the corresponding sentence about objects at rest, except that in this case, the condition requires that there be a 'constant-speed.'

An object moving with a constant speed has a constant acceleration of zero.

```
(define-rule constant-speed-3
    ((int (call constant-speed 'constant-speed)))
  (call (<< int constant-acceleration)
        'constant-acceleration)
  (== (<< int constant-acceleration fapp occurrence) (<<))
  (constraints (equal (<< int constant-acceleration)
                      (quantity 0))))
```

Problem 2.1

The first sentence of this problem creates a `constant-speed` occurrence.

> A magic carpet is moving at a constant speed of 10 m/s along a straight line.

The referent of the verb group of this sentence looks like this:

```
#<TERM REFERENT [449]>
 CLASS:
  MAJOR-CLASS: OCCURRENCE
  OCCURRENCE-CLASS: MOTION
  TIME-CLASS: INTERVAL
 THEME: [80]
  CLASS:
   MAGICNESS: T
   MAJOR-CLASS: OBJECT
   OBJECT-CLASS: CARPET
 CONSTANT-SPEED [97]:
  CLASS:
   MAJOR-CLASS: QUANTITY
   QUANTITY-CLASS: SPEED
   SPEED-CLASS: CONSTANT-SPEED
  FAPP:
   CLASS:
   VALUE: [97]
   OCCURRENCE: [449]
   ARGUMENT: [80]
 PATH: [537]
```

And this constraint has been introduced:

```
#<QUANTITY-CONSTRAINT ...>
 #<TERM CONSTANT-SPEED [97]>
 (10 (METER) (SECOND))
```

The second sentence creates a occurrence with 'changing-quantity' and 'average-acceleration' slots:

> What average acceleration is needed to increase the speed of the carpet to 50 m/s in 5 s?

The next step involves a bit of work because the occurrence described by "increase the speed" is a motion occurrence and so the system attempts to merge it with the referent of the verb group of the first sentence. This merge succeeds, and in fact the rule `constant-speed-2` can be used to determine that the 'initial-speed' of the interval is 10 m/s. However this then leads to a contradiction: the same `constant-speed-2` rule determines that the 'final-speed' of the interval must also be 10 m/s, but the value introduced by this sentence indicates that the 'final-speed' must be 50 m/s. So the system retracts all of that work, and decides that the new motion occurrence during which the speed is increasing must occur *after* the constant-speed interval.

The 'end' slot of the occurrence term 449 is thus created. Next, the rule named `constant-speed-2` creates an 'instantaneous-speed' for it:

```
#<TERM REFERENT [449]>
 ...
 END: [539]
  INSTANTANEOUS-SPEED: [868]
   CLASS:
   FAPP:
    CLASS:
    VALUE: [868]
    OCCURRENCE [539]:
    ARGUMENT: [80]
  THEME: [80]
  CLASS:
   TIME-CLASS: MOMENT
```

The rule also creates this constraint:

```
#<OPERATOR-CONSTRAINT ...>
 (EQUAL #<TERM INSTANTANEOUS-SPEED [868]>
        #<TERM CONSTANT-SPEED [97]>)
```

The referent of the verb "increase" of the second sentence looks like this:

```
#<TERM OCCURRENCE [390]>
 START: [539]
 AVERAGE-ACCELERATION: [81]
 THEME: [80]
 CLASS:
  MAJOR-CLASS: OCCURRENCE
  OCCURRENCE-CLASS: MOTION
  TIME-CLASS: INTERVAL
 CHANGING-QUANTITY [725]:
  FAPP:
  CLASS:
   MAJOR-CLASS: QUANTITY
   QUANTITY-CLASS: SPEED
   SPEED-CLASS: INSTANTANEOUS-SPEED
 END [781]:
  THEME: [80]
  INSTANTANEOUS-SPEED [123]:
  CLASS:
 DURATION [814]:
 PATH:
```

Note that its 'start' moment is the same as the 'end' moment of the previous occurrence, and its own 'end' moment was contributed by processing the preposition 'to.'

This constraint is introduced by the definition of `average acceleration`:

```
#<OPERATOR-CONSTRAINT ...>
 (EQUAL #<TERM REFERENT [81]>
        (QUOTIENT
          (DIFFERENCE #<TERM INSTANTANEOUS-SPEED [123]>
                      #<TERM INSTANTANEOUS-SPEED [868]>)
          #<TERM DURATION [814]>))
```

And these two are introduced by the quantity expressions:

```
#<QUANTITY-CONSTRAINT ...>
 #<TERM DURATION [814]>
 (5 (SECOND) ())

#<QUANTITY-CONSTRAINT ...>
 #<TERM INSTANTANEOUS-SPEED [123]>
 (50 (METER) (SECOND))
```

The quotient constraint has all of the values it needs to run and compute the answer:

```
(8 (METER) (SECOND SECOND))
```

Problem 2.2

The following problem apparently uses the class `uniformly-accelerate`:

> A car increases its speed uniformly from 10 m/sec to 30 m/sec in 10 sec. What is its average acceleration?

But in fact this problem is simpler than that. The definition of the average acceleration doesn't require that the acceleration be uniform. So although Sagredo is beguiled by the wording of this problem to create a 'constant-acceleration' slot on the motion occurrence and instantiate all of the constraints and rules that that involves, none of it is used. All that is needed is the initial and final speeds, and the duration, which are all given, and the definition constraint introduced by the class `average-acceleration`. The constraint is immediately solvable to yield:

```
(2 (METER) (SECOND SECOND))
```

Problem 2.3

> An object starts from rest with constant acceleration 8 m/s^2 along a straight line. Find the speed at the end of 5 s.

This problem exercises the rule `rest-1` that matches the `rest` occurrence introduced by the 'from' prepositional phrase. This rule creates a 'constant-speed' slot whose value is constrained to be zero. From here, the system proceeds as it did in problem 2.1, first attempting to merge the `rest` occurrence with the acceleration occurrence, failing, and then taking its 'end' slot as the 'begin' slot of the acceleration occurrence.

In this problem, though, the rule `constant-acceleration-1` is invoked to create a slot 'average-acceleration' on the occurrence. When the class `average-acceleration` is instantiated, the system has all of the constraints and values needed to compute the answer:

```
(40 (METER) (SECOND))
```

Problem 2.4

> A car whose initial speed is 55 mi/hr begins to slow at a constant rate of 5 feet per second per second. How long does it take to stop?

Sagredo can't solve this problem without help. The given acceleration value of 5 feet per second per second must be converted to a negative number before the formula can be solved. This is implied by the verb "slow" of course, but I couldn't figure out how to encode this fact in any way that was more elegant than to simply help the program directly when it encountered such a situation.

The verb "stop" is defined as a motion occurrence whose 'end' slot is of class **rest**, and so the rule **rest-1** is invoked as it was for problem 2.3 to determine that the final speed is zero. The answer is:

```
(11 (SECOND) ())
```

Paragraph 3

The last paragraph of this section is like the last one of the last section, except that initial and final accelerations are not defined. (They will never be used.)

> A moving object has an *instantaneous acceleration* at every moment during its motion.

This sentence leads to the definition of the following class and rule:

```
(define-class instantaneous-acceleration
    (acceleration fapp-quantity)
  (== (<< class acceleration-class)
      'instantaneous-acceleration)
  (in fapp
    (== argument (<< occurrence theme))
    (call occurrence 'motion 'moment)))

(define-rule instantaneous-acceleration-1
    ((sub
      (call (<<) 'moment))
     (super
      (call (<<) 'motion 'interval)))
  (== (<< sub theme) (<< super theme))
  (call (<< sub instantaneous-acceleration)
        'instantaneous-acceleration)
  (== sub (<< sub instantaneous-speed fapp occurrence)))
```

The next sentence adds a more useful rule about motion with constant acceleration:

> The instantaneous acceleration of an object moving with a constant acceleration during an interval equals its constant acceleration at every moment during the interval.

The system creates a rule corresponding to the one for **constant-speed** in the first section:

```
(define-rule constant-acceleration-2
    ((sub (call (<<) 'moment))
     (super
      (call constant-acceleration 'constant-acceleration)))
  (== (<< sub theme) (<< super theme))
  (call (<< sub instantaneous-acceleration)
        'instantaneous-acceleration)
  (== sub
      (<< sub instantaneous-acceleration fapp occurrence))
  (constraints (equal (<< sub instantaneous-acceleration)
                      (<< super constant-acceleration))))
```

8.3 Motion with Constant Acceleration

Most of the expository action of the previous two sections involved defining classes and rules that use or introduce object of those classes. This section and the next shift to deriving equations that apply in specific situations.

Paragraph 1

> If an object is moving along a straight line with a constant acceleration a, its average speed s_{av} is given by
>
> $$s_{av} = s_i + \frac{s_f - s_i}{2}, \tag{3.1}$$
>
> where s_i is the initial speed of the object for the interval, and s_f is the final speed of the object for the interval.

This sentence sets up a motion occurrence with a 'constant-acceleration' slot and an 'average-speed' slot, and creates a constraint within that occurrence between the value of the average speed and the values of the initial and final speeds. From this occurrence, the `intensionalize` function produces the following rule:

```
(define-rule constant-acceleration-2
    ((int
      (call constant-acceleration 'constant-acceleration)
      (call path 'straight 'line)))
  (call (<< int average-speed) 'average-speed)
  (== (<< int average-speed fapp occurrence) int)
  (constraints
   (equal (<< int average-speed)
          (sum (<< int start instantaneous-speed)
               (quotient
                (difference
                 (<< int end instantaneous-speed)
                 (<< int start instantaneous-speed))
                2)))))
```

This rule is presented as a simple fact. It is not derived from any other formulas nor is it the definition of any of the quantities involved. Some textbooks present this equation as an empirical result of experiments like the one that Galileo performed with balls rolling down inclined planes. Others present plausibility arguments based on the areas under graphs of distance versus position. Other texts derive it with calculus. Sagredo isn't interested in empirical results, nor with plausibility arguments, and doesn't understand calculus. So I just present the equation as a straight fact.

One other important detail is not visible in the formatted version of the text. Sagredo sees the the raw LATEX for the equation:

```
\begin{equation}
s_{av} = s_{i} + \frac{s_{f} - s_{i}}{2},
\label{cacv}
\end{equation}
```

The equation part of this is parsed, and the command `\label{cacv}` directs the system to record that name with the equation. Later steps of derivations will refer to that command by its name with the `\ref` command. The LATEX program assigns a number to each equation, in this case the equation is numbered 3.1. So when the command `\ref{cacv}` is typeset, it will appear as '3.1'.[1] Sagredo records the assignment of the name `cacv` with this equation.

Paragraph 2

The first sentence of this paragraph sets up the occurrence that will be discussed for the rest of the paragraph:

> Consider an object moving along a straight line with a constant acceleration a for a time t.

This is the referent of the verb group of the first sentence:

```
#<TERM REFERENT [506]>
 THEME: [420]
 CLASS:
  MAJOR-CLASS: OCCURRENCE
  OCCURRENCE-CLASS: MOTION
  TIME-CLASS: INTERVAL
 PATH: [546]
 CHANGING-QUANTITY:
 CONSTANT-ACCELERATION: [983]
 DURATION: [855]
```

And these two constraints associate variables with the values of quantities:

```
#<VARIABLE-CONSTRAINT ...>
 #<TERM DURATION [855]>
 (VARIABLE "t")

#<VARIABLE-CONSTRAINT ...>
 #<TERM CONSTANT-ACCELERATION [983]>
 (VARIABLE "a")
```

[1]For details on LATEX's treatment of `label` and `ref` see [198], section C.10.2.

Now the rule **constant-acceleration-1** applies, creating a 'average-acceleration,' slot on term 506 and installing a constraint.

```
#<TERM REFERENT [506]>
 ...
 AVERAGE-ACCELERATION [59]:
 END [239]:
  THEME: [420]
  INSTANTANEOUS-SPEED [585]:
  CLASS:
 START [412]:
  THEME: [420]
  INSTANTANEOUS-SPEED [562]:
  CLASS:

#<OPERATOR-CONSTRAINT ...>
 (EQUAL #<TERM AVERAGE-ACCELERATION [59]>
        #<TERM CONSTANT-ACCELERATION [983]>)
```

The quantity average acceleration comes with its definition:

```
#<OPERATOR-CONSTRAINT ...>
 (EQUAL #<TERM AVERAGE-ACCELERATION [59]>
        (QUOTIENT
          (DIFFERENCE #<TERM INSTANTANEOUS-SPEED [585]>
                      #<TERM INSTANTANEOUS-SPEED [562]>)
          #<TERM DURATION [855]>))
```

The second sentence posts a goal for the algebra system:

> From the definition of average acceleration, derive:
>
> $$at = s_f - s_i, \tag{3.2}$$
>
> where s_i is the initial speed of the object and s_f is its final speed.

First, this sentence adds two more variable constraints:

```
#<VARIABLE-CONSTRAINT ...>
 #<TERM INSTANTANEOUS-SPEED [562]>
 (VARIABLE "s" "i")

#<VARIABLE-CONSTRAINT ...>
 #<TERM INSTANTANEOUS-SPEED [585]>
 (VARIABLE "s" "f")
```

Before the derivation can be done, the system must respond to the 'locate-equation' discourse event. The equation in question is specified as a **definition** where the quantity is **average-acceleration**. The system locates a current instantiation of the constraint introduced by the definition of the class **average-acceleration**. The expression part of this constraint is:

```
(EQUAL #<TERM AVERAGE-ACCELERATION [59]>
       (QUOTIENT
         (DIFFERENCE #<TERM INSTANTANEOUS-SPEED [585]>
                     #<TERM INSTANTANEOUS-SPEED [562]>)
         #<TERM DURATION [855]>))
```

This is the initial expression that the algebra system has to transform into the goal expression.

The goal expression is determined by first parsing the input into this formula constraint expression:

```
(EQUAL (PRODUCT (VARIABLE "a")
                (VARIABLE "t"))
       (DIFFERENCE (VARIABLE "s" "f")
                   (VARIABLE "s" "i")))
```

This expression is interpreted against the current variable bindings to yield:

```
(EQUAL (PRODUCT #<TERM CONSTANT-ACCELERATION [983]>
                #<TERM DURATION [855]>)
       (DIFFERENCE #<TERM INSTANTANEOUS-SPEED [585]>
                   #<TERM INSTANTANEOUS-SPEED [562]>))
```

The transformation is performed in two steps. First the equality constraint between the value of the average acceleration and the constant acceleration is invoked to yield:

```
(EQUAL #<TERM CONSTANT-ACCELERATION [983]>
       (QUOTIENT
         (DIFFERENCE #<TERM INSTANTANEOUS-SPEED [585]>
                     #<TERM INSTANTANEOUS-SPEED [562]>)
         #<TERM DURATION [855]>))
```

This is transformed to the goal expression with the rule:

$$x = \frac{y}{z} \rightarrow xy = z$$

The system then records the label of this equation: **atv**.

The next sentence continues the derivation:

Substitute the formula for $s_f - s_i$ from equation 3.2 into equation 3.1:

$$s_{av} = s_i + \frac{at}{2}. \tag{3.3}$$

The first step here is to locate an equation. The equation is specified by the label `atv` (which is printed out as 3.2). From that equation Sagredo extracts a formula for $s_f - s_i$, which is interpreted as the expression:

```
(DIFFERENCE #<TERM INSTANTANEOUS-SPEED [585]>
            #<TERM INSTANTANEOUS-SPEED [562]>)
```

In equation 3.2, that expression is `equal` to:

```
(PRODUCT #<TERM CONSTANT-ACCELERATION [983]>
         #<TERM DURATION [855]>)
```

Now the system has to locate the equation labeled `cacv` (printed as 3.1). In the current instantiation of that equation, the expression for $s_f - s_i$ is inside a `quotient` constraint. Sagredo replaces it with the `product` expression just obtained to yield:

```
(EQUAL #<TERM AVERAGE-SPEED [347]>
       (SUM #<TERM INSTANTANEOUS-SPEED [562]>
            (QUOTIENT
             (PRODUCT #<TERM CONSTANT-ACCELERATION [983]>
                      #<TERM DURATION [855]>)
             2)))
```

This is the interpreted version of the equation labeled 3.3.

The next step is another substitution:

> Substitute the formula for s_{av} from equation 3.3 into the definition of average speed:
>
> $$s_i + \frac{at}{2} = \frac{d}{t}, \tag{3.4}$$
>
> where d is the distance traveled.

This sentence requires that the constraint introduced by the definition average speed be located. The formula that is equal to the average speed is collected from the last equation and inserted:

```
(EQUAL
 (SUM #<TERM INSTANTANEOUS-SPEED [562]>
      (QUOTIENT
       (PRODUCT #<TERM CONSTANT-ACCELERATION [983]>
                #<TERM DURATION [855]>)
       2))
 (QUOTIENT #<TERM DISTANCE [481]> #<TERM DURATION [855]>))
```

This is the interpreted version of equation 3.4. The last clause also introduced the following variable constraint:

```
#<VARIABLE-CONSTRAINT ...>
 #<TERM DISTANCE [481]>
 (VARIABLE "d")
```

The final sentence just asks that the last equation be transformed to one that isolates the variable d.

Now solve for d:

$$d = s_i t + \frac{1}{2} a t^2. \tag{3.5}$$

This is the interpreted version of equation 3.5, the goal of the algebraic transformation:

```
(EQUAL #<TERM DISTANCE [481]>
       (SUM (PRODUCT #<TERM INSTANTANEOUS-SPEED [562]>
                     #<TERM DURATION [855]>)
            (PRODUCT
              (PRODUCT (QUOTIENT 1 2)
                       #<TERM CONSTANT-ACCELERATION [983]>)
              (SQUARE #<TERM DURATION [855]>))))
```

The program has to transform equation 3.4 into equation 3.5. This derivation turns out to be the hardest one that Sagredo has to follow. Here is the solution the program finds.
Start with equation 3.4:

```
(EQUAL
  (SUM #<TERM INSTANTANEOUS-SPEED [562]>
       (QUOTIENT
         (PRODUCT #<TERM CONSTANT-ACCELERATION [983]>
                  #<TERM DURATION [855]>)
                 2))
  (QUOTIENT #<TERM DISTANCE [481]> #<TERM DURATION [855]>))
```

Apply the rule:

$$x = \frac{y}{z} \rightarrow xz = y$$

to yield:

```
(EQUAL
 (PRODUCT
  (SUM #<TERM INSTANTANEOUS-SPEED [562]>
       (QUOTIENT (PRODUCT
                  #<TERM CONSTANT-ACCELERATION [983]>
                  #<TERM DURATION [855]>)
                 2))
  #<TERM DURATION [855]>)
 #<TERM DISTANCE [481]>)
```

Now apply $x = y \rightarrow y = x$:

```
(EQUAL #<TERM DISTANCE [481]>
       (PRODUCT
         (SUM #<TERM INSTANTANEOUS-SPEED [562]>
              (QUOTIENT
                (PRODUCT #<TERM CONSTANT-ACCELERATION [983]>
                         #<TERM DURATION [855]>)
                2))
         #<TERM DURATION [855]>))
```

Apply $xy \rightarrow yx$:

```
(EQUAL #<TERM DISTANCE [481]>
       (PRODUCT
         #<TERM DURATION [855]>
         (SUM #<TERM INSTANTANEOUS-SPEED [562]>
              (QUOTIENT
                (PRODUCT #<TERM CONSTANT-ACCELERATION [983]>
                                #<TERM DURATION [855]>)
                2))))
```

Apply $x(y + z) \rightarrow xy + xz$:

```
(EQUAL #<TERM DISTANCE [481]>
       (SUM (PRODUCT #<TERM DURATION [855]>
                     #<TERM INSTANTANEOUS-SPEED [562]>)
            (PRODUCT
              #<TERM DURATION [855]>
              (QUOTIENT
                (PRODUCT #<TERM CONSTANT-ACCELERATION [983]>
                         #<TERM DURATION [855]>)
                2))))
```

Apply $xy \rightarrow yx$ to both of the resultant products:

```
(EQUAL #<TERM DISTANCE [481]>
       (SUM (PRODUCT #<TERM INSTANTANEOUS-SPEED [562]>
                     #<TERM DURATION [855]>)
            (PRODUCT
              (QUOTIENT
                (PRODUCT #<TERM CONSTANT-ACCELERATION [983]>
                         #<TERM DURATION [855]>)
                2)
              #<TERM DURATION [855]>)))
```

Apply

$$\frac{y}{x} \rightarrow y\frac{1}{x}$$

```
(EQUAL #<TERM DISTANCE [481]>
       (SUM (PRODUCT #<TERM INSTANTANEOUS-SPEED [562]>
                     #<TERM DURATION [855]>)
            (PRODUCT
              (PRODUCT
                (PRODUCT #<TERM CONSTANT-ACCELERATION [983]>
                         #<TERM DURATION [855]>)
                (QUOTIENT 1 2))
              #<TERM DURATION [855]>)))
```

Now move the coefficient 1/2 to the front with applications of the rules $xy \rightarrow yx$ and $x(yz) \rightarrow (xy)z$:

```
(EQUAL #<TERM DISTANCE [481]>
       (SUM (PRODUCT #<TERM INSTANTANEOUS-SPEED [562]>
                     #<TERM DURATION [855]>)
            (PRODUCT
              (PRODUCT (PRODUCT
                         (QUOTIENT 1 2)
                         #<TERM CONSTANT-ACCELERATION [983]>)
                       #<TERM DURATION [855]>)
              #<TERM DURATION [855]>)))
```

Now rearrange the product with $(xy)z \rightarrow x(yz)$ to get the 'duration' terms together:

```
(EQUAL #<TERM DISTANCE [481]>
       (SUM (PRODUCT #<TERM INSTANTANEOUS-SPEED [562]>
                     #<TERM DURATION [855]>)
            (PRODUCT
              (PRODUCT (QUOTIENT 1 2)
                       #<TERM CONSTANT-ACCELERATION [983]>)
              (PRODUCT #<TERM DURATION [855]>
                       #<TERM DURATION [855]>))))
```

Collect the 'duration' terms with the rule $xx \rightarrow x^2$:

```
(EQUAL #<TERM DISTANCE [481]>
       (SUM (PRODUCT #<TERM INSTANTANEOUS-SPEED [562]>
                     #<TERM DURATION [855]>)
            (PRODUCT
              (PRODUCT (QUOTIENT 1 2)
                       #<TERM CONSTANT-ACCELERATION [983]>)
              (SQUARE #<TERM DURATION [855]>))))
```

This is the desired expression. Since this derivation is the last one in a paragraph, the system records it as a rule that can be applied to any motion occurrence that has a 'constant-acceleration' slot:

```
(define-rule constant-acceleration-3
    ((int
      (call constant-acceleration 'constant-acceleration)
      (call path 'straight 'line)))
  (constraints
   (equal (<< int distance)
          (sum
           (product
            (<< int start instantaneous-speed)
            (<< int duration))
           (product
            1/2
            (product (<< int constant-acceleration)
                     (square (<< int duration)))))))
```

Problem 3.1

A car accelerates from rest at a constant rate of 8 m/s^2.

This sentence creates a motion occurrence with a 'constant-acceleration' slot whose initial speed is zero. The new rule `constant-acceleration-3` is instantiated on this occurrence.

(a) How fast is it going after 10 sec?

This question could have been answered at the end of the last section since it just requires the application of the definition of average acceleration and the constraint that the average acceleration and the constant acceleration are equal.

(b) How far has it gone after 10 sec?

This question asks for the value of the 'distance' slot, and this slot is related to the given quantities via the constraint introduced by the rule `constant-acceleration-3`.

Problem 3.2

A jumbo jet needs to reach a speed of 225 mi/h on the runway for take-off. Assuming a constant acceleration and a runway 1.1 miles long, what minimum acceleration from rest is required?

This problem is interesting because it is phrased as a requirement, and asks for the "minimum" acceleration. The program simply ignores such issues. The requirement verbs "needs" and 'required' are treated as if they just set the indicated values. The word "minimum" is similarly ignored. In essence the program sees a problem in which the jet *reaches* the speed of 225 mi/h, and does so in exactly 1.1 miles.

This "reasoning" is built into the definitions of the words involved. Once the sentences are interpreted, the program has the required quantity values to solve the problem.

Paragraph 3

Like the previous presentation paragraph, this one begins by describing a situation for consideration:

Consider an object moving along a straight line with a constant acceleration a for a time t.

Most physics texts would probably have omitted this line, and continued to use the same situation described in the previous paragraph. Indeed even the initial sentence of the previous paragraph could have been omitted because the situation was described

in the first sentence of the chapter. But Sagredo only looks for co-referring expressions within the same problem or paragraph, so I have to redescribe the situation for each one.

The remaining sentences of this section are handled essentially the same way as for the last section.

From the definition of average acceleration, derive:

$$t = \frac{s_f - s_i}{a}, \tag{3.6}$$

where s_i is the initial speed of the object and s_f is its final speed.

This step is exactly the same as the first step of the last section except that the desired transformation of the definition of average acceleration is different.

From equation 3.1, derive:

$$s_{av} = \frac{s_i + s_f}{2}. \tag{3.7}$$

This step requires a relatively simple transformation of the formula. I should point out that this is the form that equation 3.1 in which is usually presented in physics textbooks. But I think that the form I used accords better with an intuitive understanding of what is going on: the average speed is equal to the initial speed plus half of the difference between the final and initial speeds. A matter of taste I am sure.

From the definition of average speed, derive:

$$d = s_{av}t, \tag{3.8}$$

where d is the distance traveled.

This is a simple rearrangement of the definition of average speed. The final step involves two substitutions at the same time:

Now substitute the formula for s_{av} from equation 3.7, and the formula for t from equation 3.6 into equation 3.8:

$$d = (\frac{s_i + s_f}{2})(\frac{s_f - s_i}{a}). \tag{3.9}$$

The final transformation requires 9 steps (I won't list them):

Simplify:

$$2ad = s_f^2 - s_i^2. \tag{3.10}$$

And again, since this is the last derivation in a paragraph, the system uses the derived equation to create a rule that posts a constraint corresponding to the equation:

```
(define-rule constant-acceleration-4
    ((int
      (call constant-acceleration 'constant-acceleration)
      (call path 'straight 'line)))
  (constraints
   (equal
    (product 2 (product (<< int constant-acceleration)
                        (<< int distance)))
    (difference
     (square (<< int end instantaneous-speed))
     (square (<< int start instantaneous-speed))))))
```

Problem 3.3

> A car traveling in a straight line with initial speed of 20 miles per hour is accelerated uniformly to a speed of 40 miles per hour in 10 sec.

This sentence provides all of the quantity information that will be needed to solve the problems.

> What is the acceleration,

As with problem 3.1, this part could have been answered at the end of the last section.

> and how far did the car travel during the interval?

The rule `constant-acceleration-4` introduced the constraint between the initial and final instantaneous speeds. That constraint can be solved to yield the answer.

Problem 3.4

> A bus moving at a speed of 20 m/s begins to slow at a rate of 3 m/s each second. Find how far it goes before stopping.

As with the last problem of the last section, the program requires help to solve this problem because the rate of acceleration must be understood to be negative. Once this is done, all of the required quantities are available for the constraint introduced by the rule `constant-acceleration-4`.

8.4 Vertical Motion

This section is about the phenomenon of gravity. The most important thing illustrated here is the way that learned rules can elaborate a described occurrence by adding new occurrences to it.

Paragraph 1

The first sentence simply states a new fact:

> An unsupported object accelerates downward at a constant rate due to gravity.

This sentence is interpreted to yield a rule that applies to any occurrence of the class `unsupported`:

```
(define-rule unsupported-1 ((int (call (<<) 'unsupported)))
  (call (<< int constant-acceleration
            'constant-acceleration))
  (== (<< int constant-acceleration fapp occurrence) int)
  (call (<< int direction) 'downward)
  (== (<< int constant-acceleration cause)
      (global-referent 'gravity)))
```

This rule just adds a 'constant-acceleration' slot to any instance of `unsupported`. Note the 'cause' slot in the 'constant-acceleration' of the created occurrence. This was introduced by the "due to" phrase. Like "the ground," "gravity" is a "global-referent" which means that a co-referrer is always available. The system will learn nothing further about what gravity is. It is recorded as the 'cause' of the acceleration so that expressions like the one in the next sentence can be interpreted:

> The magnitude of the gravitational acceleration is approximately 9.8 m/s^2.

This sentence adds the rule:

```
(define-rule constant-acceleration-5
    ((int (call constant-acceleration 'constant-acceleration)
          (== (<< constant-acceleration cause)
              (global-referent 'gravity))))
  (constraints
   (equal (<< int constant-acceleration)
          (quantity 9.8 (meter) (second second)))))
```

Paragraph 2

The first sentence sets up the occurrence that will be discussed for the rest of the paragraph:

> Suppose that an object is dropped from rest at a height h.

The definition of the verb "dropped" is relatively complex. It creates a moment occurrence which is the 'end' slot of a `support` occurrence and the 'begin' slot of an `unsupported` occurrence. The prepositional phrase headed by 'from' fills in some of the details of the initial occurrence: the constant speed is zero (from the rule `rest-1`), and the 'vertical-position' is constrained to the variable h.

The just-learned rule `unsupported-1` elaborates the final `unsupported` occurrence to add a 'constant-acceleration' slot, and the rule `constant-acceleration-5` creates a constraint assigning the value of the acceleration to be 9.8 m/s^2.

This is a `constant-acceleration` occurrence, so rule `constant-acceleration-3` has introduced the constraint represented by equation 3.5:

$$d = s_i t + \frac{1}{2}at^2.$$

The next sentence describes some substitutions of values into this equation. Actually, all of these were already done when the constraint specification was instantiated on this occurrence.

> Substitute 0 for the initial speed, h for the distance, and g for the acceleration into equation 3.5:

$$h = 0 + \frac{1}{2}gt^2, \tag{4.1}$$

The last clause of this sentence requires some special reasoning:

> where t is the time taken by the object to reach the ground.

First of all, "the ground" is another "global referent." It is recorded as always having a 'vertical-position' slot which is constrained to be zero. The verb "reach" designates an interval occurrence whose 'end' slot is an instance of `support`. In this case the support is the ground, hence the object's 'vertical-position' in that occurrence is also zero. A special rule now sets the 'distance' slot of the intervening motion occurrence to be the difference between these two vertical positions. This whole passage was cleverly written so that the absolute values of all of the quantities could be used correctly. A more careful text would discuss how the different directions of motion require different signs for the values of the quantities.

The next sentence completes the current derivation:

Solve for t:

$$t = \sqrt{\frac{2h}{g}}. \tag{4.2}$$

The algebra system easily follows this derivation from equation 4.1.

This equation is now used to form a rule for the relatively complex situation just described:

```
(define-rule unsupported-2
    ((int (call (<<) 'unsupported)
          (call before 'rest 'support)
          (call end 'support)
          (call (<< end location)
                (global-referent 'the-ground))))
  (constraints
   (equal (<< int distance)
          (<< int before vertical-position))
   (equal (<< int duration)
          (sqrt (quotient (product 2 (<< int distance))
                          (<< int constant-acceleration))))))
```

The next sentence begins the derivation of another equation that is valid in the current situation.

Substitute 0 for s_i, and g for a in the definition of average acceleration:

$$g = \frac{s_f - 0}{t}, \tag{4.3}$$

where s_f is the final speed of the object.

This equation is easily transformed:

Solve for s_f:

$$s_f = gt. \tag{4.4}$$

The final step involves inserting a previously derived formula, and then performing a two-step transformation of the result:

Now use the formula for t from equation 4.2:

$$s_f = g\sqrt{\frac{2h}{g}} = \sqrt{2gh}. \tag{4.5}$$

From this, Sagredo records the rule:

```
(define-rule unsupported-3
    ((int (call (<<) 'unsupported)
          (call before 'rest 'support)
          (call end 'support)
          (call (<< end location)
                (global-referent 'the-ground))))
  (constraints
   (equal (<< int distance)
          (<< int before vertical-position))
   (equal (<< int end instantaneous-speed)
          (sqrt (product
                 2
                 (product (<< int constant-acceleration)
                          (<< int distance))))))
```

Problem 4.1

(a) How long does it take a brick to reach the ground if dropped from a height of 80 m?

This sentence is a straightforward application of the material just learned. The one detail is that the problem does not specify that the brick was dropped *from rest* at the indicated height, so therefore the rule `unsupported-2` cannot immediately apply. This is an example of a fairly general phenomenon in textbooks and problems where otherwise relevant information is left out and the student is required to make the "obvious" assumptions necessary to make the problems solvable. In fact it took me a while before I could understand why Sagredo couldn't do such an easy problem. So I had to annotate the occurrence description by hand, adding the class `rest` to the initial `support` occurrence. Once this was done, the rule `unsupported-2` can introduce the constraint represented by equation 4.2 and the problem can be solved.

(b) What will be its speed just before it reaches the ground?

This sentence is phrased in an interesting way. The problem would be simpler for Sagredo if is just said "what will be its speed *when* it reaches the ground" because the 'end' event of the `motion` occurrence is the beginning of the `support` event. I think that the writer of this text was being careful in a weird way: strictly speaking, when the object reaches the ground it will decelerate rapidly to zero, and might even bounce back up. However at the moment instantaneously before it hits, it will still be moving down, and its speed will be just instantaneously less than its final speed.

Sagredo ignores all of this and treats "just before" as if it were "when." So the constraint represented by equation 4.5 applies and the problem is solved.

Problem 4.2

A stone is thrown vertically downward with a speed of 5 m/sec.

This sentence describes an occurrence which is not covered by the rules derived in this section. The program has only the rule `unsupported-1` to work with. The verb "thrown" introduces an `unsupported` occurrence, but this one gets a nonzero initial speed.

(a) How far will it go in 3 sec?

Since the rule `unsupported-1` has introduced a 'constant-acceleration' slot to the `motion` occurrence, and the rule `constant-acceleration-5` has introduced a constraint specifying the value of the acceleration, this problem can be solved by applying equation 3.5 from the last section.

Similarly, this question can also be solved as a straightforward constant acceleration problem:

(b) How fast will it be moving at the end of 3 sec?

Problem 4.3

This problem is more interesting. The first sentence describes a `drop` event.

A stone is dropped into the water from a bridge 144 ft above the water.

The rules that apply to unsupported motion starting from rest can be applied to this occurrence. This problem specifies the location of the `support` occurrences that bound the motion occurrence, the first is the bridge and the second is the water. (Okay, I know that water can't really support stones. Pretend it is ice or something.)

Another stone is thrown vertically down 1.0 s after the first is dropped.

The phrase "1.0 s after the first is dropped" creates an interval between the two `drop` occurrences, and sets the 'duration' slot to be 1 second. The occurrence described in this sentence, like the one in problem 4.2 involves a nonzero initial speed.

Both stones strike the water at the same time.

This is the one place in the whole text where Sagredo has to deal with a noun phrase that designates more than a single object. This sentence is interpreted as introducing two separate `moment` occurrences, each of type `support`, such that their times are the same.

What was the initial speed of the second stone?

The initial speed of the first stone is known to be zero. The final speed of the first stone is known from equation 4.5. This can be used to compute the time the first stone spent falling. So the second stone spent 1 second less time than this. Since the distance, the acceleration, and the time are known, the initial speed can be computed from equation 3.5.

Paragraph 3

Sagredo's handling of the first part of this paragraph and the first problem was described in the demo in chapter 3.

> When an object affected by gravity is projected vertically upward with an initial speed s_i, its instantaneous speed decreases at a constant rate. The object rises until its speed is zero. When its speed is zero, the object is at its maximum height h. Substitute h for the distance, 0 for the final speed, and $-g$ for the acceleration into equation 3.10:
>
> $$2(-g)h = 0 - s_i^2. \tag{4.6}$$
>
> Solve for h:
>
> $$h = \frac{s_i^2}{2g}. \tag{4.7}$$

> **Problem 4.4** [Schaum 4.13, p. 36]
> A stone is thrown straight upward and it rises to a height of 20 m. With what speed was it thrown?

The second part of this paragraph a simple derivation. It begins by declaring that two moments are the same:

> The time t_u taken by the object to reach the maximum height is equal to the time taken for the speed of the object to reach zero.

This sentence is just a restatement of what was said earlier. It does not add any structure to the occurrence description. The next sentence begins a derivation:

> Use equation 3.2 with 0 for the final speed, and $-g$ for a:
>
> $$-gt_u = 0 - s_i. \tag{4.8}$$

And this is then solved for the desired value:

> Solve for t_u:
>
> $$t_u = \frac{s_i}{g}. \tag{4.9}$$

From this, the following rule is defined. Its condition and the first forms of its action are identical to the rule `project-1` that was defined in the demo chapter. The difference is the constraint that it introduces, based on the derivation just completed:

```
(define-rule project-2
    ((occ (call (<<) 'project)
          (call (<< direction) 'vertically-upward)))
  (call (<< occ after constant-acceleration)
        'constant-acceleration)
  (== (<< occ after constant-acceleration fapp occurrence)
      (<< occ after))
  (constraints
   (equal (<< occ duration)
          (quotient
           (<< occ instantaneous-speed)
           (quantity 9.8 (meter) (second second))))))
```

Neither of the following two problems are worth discussing in detail. They are both straightforward applications of the material just learned.

Problem 4.5
A plastic dinosaur tossed straight up reaches its maximum height in 3 seconds. How high did it go?

Problem 4.6 [SZY 4.22, p. 65]
With what speed must a ball be thrown vertically upward in order to rise to a height of 50 m?

Paragraph 4

This paragraph continues the story of what happens when an object is projected vertically:

> Once an object projected vertically upward with an initial speed s_i reaches its maximum height, it will begin moving downward with the constant acceleration of gravity.

Actually Sagredo already has this information because the rules it learned from the last section determined that the 'end' moment of the vertically upward motion of a `project` event is an unsupported `rest` event. And so the rule `unsupported-1` applies, creating the subsequent downward motion occurrence.

> The object will take the same time falling as it took rising.

This fact should be deducible by Sagredo also, but it requires understanding that the distances are the same. But to do this would require that the program be able to reason about distances and the signs of quantities, which is is not. So it records this statement as a fact and introduces the appropriate constraint.

> So the total time of flight t is twice the time t_u the object took to reach its maximum height:
>
> $$t = 2t_u = 2\frac{s_i}{g}, \tag{4.10}$$
>
> where g is the acceleration of gravity.

The first step in this derivation just represents what was stated in the previous sentence. The next step involves inserting the value for the time from equation 4.9. This constraint is now incorporated into a rule.

Problem 4.7

> A stone is thrown straight up.

This sentence creates a `project` occurrence, which means that the rules introduced in paragraphs 3 and 4 get instantiated.

The next sentence adds the value of a quantity:

> It returns to the ground 6 seconds later.

The first question requires the application of equation 4.10:

> (a) With what speed was it thrown?

Given that the initial-speed has now been computed, its value can be used in equation 4.7 to answer the next question:

> (b) How high did the stone rise?

Chapter 9

Discussion

In the kinematics source text, Sagredo encounters 91 sentences plus 69 additional embedded or subordinate clauses, making use of a lexicon of 241 words. The program handles a range of syntactic, semantic and discourse-level phenomena, specifically and most importantly, the phenomenon of exposition itself, where a sequence of sentences make a statement that is taken to be a general law or the definition of a new term or the description of a phenomenon. The program learns enough physics to be able to follow the derivation of 20 equations, and to use them in solving 20 physics problems stated in English, a majority of them taken verbatim from college-level physics textbooks.

In this chapter I will go beyond this litany of numbers and discuss in some additional detail why the program works, the kinds of situations it can't handle, and how and into which domains it might be extended and improved.

9.1 Summary of the Material Learned

Sagredo acquires and makes use of the definitions of the following physics concepts:

average speed
constant speed
instantaneous speed
initial/final speed
average acceleration
constant acceleration
instantaneous acceleration
gravity

For each of these concepts, the program acquires a description of the occurrence in which the concept applies, and for some of them acquires rules that can be used

to elaborate a situation in which the concept applies. The definitions and the rules introduce the equations that relate the values of the quantities involved.

The program learns the 5 motion equations described in figure 4.1 on page 37. 2 of those equations are the definitions of quantities. One is simply stated as a fact. The other two are derived. In addition, the program follows the derivation of 5 important equations specifically for dealing with vertical motion.

The program learns the following rules that can elaborate an occurrence of one type into a more complex occurrence:

> The instantaneous speed at any moment during an interval of motion at a constant speed is equal to the constant speed.
>
> The instantaneous acceleration at any moment during constant acceleration is equal to constant acceleration.
>
> Unsupported objects accelerate downward with the constant rate g.
>
> A rising object eventually stops at a maximum height.

Everything that the program learns from the presentation text is exercised in at least one problem.

9.2 Why It Works

The reason that the program works is that for a lot of physics problems, story subsumption works. That is: the situation described by the presentation text is more general than the situation described by the problem text. In the implementation of Sagredo, I was able to capture the notion of "more general" with a computational model based on subsumption between a representation of the occurrence described in the presentation text and the occurrence described by a problem. In a textbook where the presentations are always strictly more general than the problems it is possible to solve a given problem by locating a presentation passage that describes a more general version of the situation described in the problem, and which presents or uses an equation that involves the quantities that the problem requires. The specific quantity values from the problem text can then be inserted into the equations introduced by the presentation text. This is what Sagredo does.

As I said at the outset, it is not a goal of this project to worry about whether this is the way that people solve physics problems. But it is clear that, whatever they actually do, people are *not supposed* to solve physics problems this way. Some textbooks explicitly vilify this mode of problem-solving:

> To obtain a benefit commensurate with the time expended, problem-solving should be considered as much more than merely substituting numbers for the symbols in a formula, or fitting together the pieces of a jigsaw puzzle. To merely thumb through the book until you find a worked-out example that resembles the problem, is a waste of time and effort.
> [SZY, p. 12.]

> ... merely poring through a set of formulas for just the one which expresses the sought-for quantity and then simply substituting numerical values and solving for the unknown quantity by arithmetic is hardly the proper goal of the serious physics student. [Bennett, p. 5.]

Both of these texts go on to explain why it is that this mode of problem solving is not approved—that the student must attempt to understand the underlying physical principles and to understand how the problem situation instantiates those principles. But I think that it is interesting that the textbook writers feel it is necessary to speak to the issue. It suggests to me, first, that this mode of problem-solving is relatively common among beginning students, and, second, that it often works.

That this "pattern-matching" style of problem-solving is common might suggest that it makes use of a style of reasoning that people often use before they come to physics. This style of reasoning pays close attention to the linguistic form of the presentation, whether it be a scientific text or a set of instructions, and attempts to see a specific case as an instantiation or specific instance of the case described by the text. If the text is the instruction sheet for a VCR, for example, it is hardly necessary or desirable (or possible?) to understand it as illustrating some fundamental principles. No, the point is to get a job done, and the text describes a sort of story, such that if you can understand your situation as an instance of the story, you can get the job done. It would seem that lots of everyday instructions and advice are like this. Part of the reason that textbooks have to include cautionary passages like those above is that this natural mode of instruction following isn't by itself adequate for becoming competent in physics. And furthermore, unlike VCR instructions and recipes and street directions, there *are*, in physics, underlying principles that can be articulated and grasped.

That the pattern-matching style of problem-solving often works illustrates that textbook writers either can't or don't want to invent problems whose solutions require a more profound understanding of the underlying physical principles. Naturally, in the text in chapter 2, I chose only simple problems, and all of the texts that I used as sources of problems included more difficult ones. But I think that there are good reasons why textbooks include this sort of "plug-in" problems. First, it is necessary for the student to have *at least* the same sort of competence with the physics presentation as with everyday instructions; second, new physics terms and concepts often involve characteristic and idiosyncratic means of linguistic expression,

or else emphasize different aspects of the expressions, and so some problems have to be included to exercise the new linguistic competence that accompanies competence in the technical subject; third, even in difficult problems, some of the aspects of the problems are still presented straightforwardly—it isn't possible for *everything* about a problem to be difficult.

Research on how people actually think about physics problems suggests that in fact novices indeed invest most of their attention in the linguistic form of the presentation. (See, for example, [54].) In fact it even goes further than this: many beginning students seem to focus on the specific objects involved. For example a student might see all "block on inclined plane" problems as of the same type, even though some involve statics, some involve constant acceleration, and others might involve energy considerations.

In any case Sagredo is certainly stuck at the most basic level of understanding of physics, the level closest to the linguistic presentation of the problem. It is possible that a human with the same limitations would never achieve better than a grade of B– or so in a college physics class. On the other hand, B– isn't all that bad. For one thing if using only "linguistic physics" as Sagredo does, gets one to a B– level, it makes sense to move up from there, to understand how an understanding of the deeper principles can be based on the linguistic physics level, rather than demanding that the deep understanding be present from the start. It seems to me that this sort of linguistic understanding of physics is always present in people, perhaps to different degrees and in different forms for different people, but it is one resource among many that underlies competence in physics.

And furthermore, recall that one of the original goals of this project was to build a system that could assist a mechanical engineer. It might be very useful to have a B– student assisting you as you are designing something. It might not matter if the assistant doesn't really have an intuitive or a deep understanding of the material, provided that it has instant access to all of the equations that you might not immediately recall, and remembers the specific situations where they apply, and can help you with the mathematics.

Sagredo seems to require very little special background knowledge. Mostly that is because I was able to encode in the grammar and discourse event rules whatever extra knowledge was needed to understand the problems. But for the most part the building in of background knowledge in fact involves *leaving out* everything but what is relevant. For example it is not necessary to know what a "magic carpet" is to do problem 3.2 except that it is a kind of carpet (so the co-referring noun phrase "the carpet" in the second sentence of the problem can be dealt with). It is not even necessary to know that a magic carpet is something that can move because the text of the problem *says* that it is moving. As in the "corandic" passage on page 14, the grammatical structures supply most of the information that is needed to set up the occurrence descriptions, and the relevant linguistic physics then allows the problems

to be solved. So most of the background knowledge that is built in to the program consists merely of the assignment the appropriate grammatical categories to the words of the lexicon, and associating the words with sets of equations that will contribute to the construction of the appropriate occurrence descriptions for sentences in which the words appear.

A more important and subtle kind of "background knowledge" is embedded in the conventions of expository text. Certain kinds of grammatical and pragmatic structures are understood in specific ways because they appear in a physics book: they would be understood quite differently, or would be unintelligible if they appeared in some other context. Except perhaps for a literary theorist or a rhetorician, the working of such expository structures are never articulated, though it seems clear that students are capable of responding to them. Of course in Sagredo, this knowledge is embedded in the rules for constructing discourse events and in determining how to respond to them.

So another important reason why the program works is that enough of the background knowledge needed to understand expository text can be expressed in the simple rules the program uses to process the text. But there is nothing really deep here—those rules were built in by me because I knew that the program would encounter one kind of (indeed one instance of) text. The interesting problem of how it is that people understand the kind of text they are confronted with and how they determine how to respond is simply not addressed by this research.

9.3 What Sagredo Can't Do

The most obvious example of a situation that Sagredo can't handle is one where story subsumption doesn't work. Consider this presentation passage:

> Suppose a motorist enters a super highway in city A and heads south for city B, a distance of 180 miles. If he covers the distance in three hours, his average velocity v_{av} is obtained by dividing the displacement d, by the time t or
>
> $$v_{\mathrm{av}} = \frac{d}{t}$$
>
> $$v_{\mathrm{av}} = \frac{180 \text{ mi}}{3 \text{ hr}} = 60\frac{\text{mi}}{\text{hr}}$$
>
> [Weissman, pp. 41–42.]

(Note that this passage is the first and only place that "average velocity" is defined in Weissman's text.) The problem here is that this presentation text is too specific. The moving object is a "motorist" (furthermore it is male), the motion is on a "super

highway," between two cities 180 miles apart, and takes 3 hours. The only ways that a problem could be more specific than this would involve fixing the identities of the motorist and the two cities. But that wouldn't change anything quantitative.

Somehow the reader is expected to make the obvious generalizations. Indeed the presentation of the equation for average velocity before the specific version of it with the values filled in is a sort of local version of the more standard general-to-specific method of exposition. But it is not obvious in which situations the equation applies. For example, does it require that the displacement be horizontal, or that it be southward?

Not many texts use this method to define such fundamental terms as "average velocity." But a number of texts present important special cases this way, with an example in which a specific situation with values for quantities is presented, and the student is expected to infer the relevant generalization. This is actually a bit more reasonable because the reasoning that the textbook applies to the example situation ought to apply intact to any other situations which are qualitatively similar. For example some texts introduce projectile motion by stating the general principle that the horizontal and vertical components can be treated separately and then illustrate this principle by first describing the case of an object projected horizontally with some initial velocity. It is clear from the subsequent derivation that the specific value of the initial velocity doesn't matter for the form of the equations used, all that matters is that the initial velocity is horizontal to the ground.

Although Sagredo can't deal with such situations now, it probably wouldn't be all that difficult for it to do so. In particular, it could easily be extended to handle cases where it has to generalize from specific quantity values to a description in which the quantities are replaced by formulas. More difficult would be cases where certain qualitative relations have to hold among the values of the quantities. A great deal of work in artificial intelligence on "explanation based generalization" [220] would probably be relevant here, and in fact would be useful if the system were to be extended to real engineering domains where a great deal of expertise involves understanding the vagaries of many different specific situation types.

There are problems in physics and especially in engineering where the problem statement does not instantiate all of the constraints needed to solve the problem. Sometimes this involves an understanding of the real-world situation, in which case that background knowledge could probably be built in. But many engineering problems presume a general background understanding of engineering design principles. In general, engineering solutions are expected to simultaneously optimize cost and weight and size and safety and reliability and other factors. Not all of these can be optimized at once, but a good engineering solution is one where the tradeoffs are reasonable, and the standard of reasonableness at work here is not a matter of solving simple algebraic equations.

There are of course other sorts of problem where the correct solution requires a deep understanding of the physics or of the problem statement. It might be that there is an axis of symmetry, perhaps, or some change of coordinates or variables that makes an apparently difficult problem easy. One example of this that I spent some time considering is the question of when to use energy considerations versus when to use force and momentum considerations alone. In many instances using one rather than the other saves a lot of work, though in general, either approach will yield a solution. This I think is where real physical intuition, whatever that is, begins to be important. Perhaps making computers capable of this would indeed require the sort of qualitative reasoning mechanisms described in [31], or the meta-level search control strategies described in [45].

There are a large number of problems that Sagredo can't handle because solving them requires doing calculus, or understanding diagrams, or any of a number of other skills that I chose not to explore. Some problems that ought to be in Sagredo's competence are not because they are just weird. Consider the following:

> A dog sees a flowerpot sail up and then back past a window 5.0 ft high. If the total time the pot is in sight is 1.0 s, find the height above the window that the pot rises. [Resnick & Halliday, 3.46, p. 52.]

This is an interesting problem in a number of ways. For one thing, there is the sheer theatrical interest in what is going on. Who is throwing flowerpots around? Some angry cat perhaps? Does the dog have any interest in the projectile activity outside the window?

This problem requires an understanding of the notion of "being in sight." The only role that the dog plays is as the "perceiver" of an event, and details of the dog's perceptual experience of the event are used to fill in the physical quantities needed to solve the problem. Clearly for any computational system to be able to solve this problem it would have to be able to deal with such linguistic constructions.

More significantly, the problem as presented cannot be solved in any way true to the physics. In order to apply the equations of motion, it is necessary to know the vertical distance that the pot traversed in the 1.0 s. The distance that one is *supposed* to use is 5.0 ft, but this cannot the actual distance the pot moved unless the dog were infinitely far from the window. But in that case it is difficult to understand how the dog could notice, much less accurately measure, the pot's time in view.

It is difficult to know what to make of such problems. They are apparently added to make the text more "interesting" to students, yet I wonder if they might confuse as often as they entertain. For a computer program to handle these problems it would either have to stupidly scan the text for instances of expressions of specific quantity types, or smart enough to recognize the absurdity of the problem, understand what is going on here, and use the incorrect values to derive the "correct" answer.

9.4 Improvements & Extensions

The most straightforward potential improvements to Sagredo have already been mentioned:

- Competence at algebra and other mathematical areas.
- The ability to do "explanation-based" generalization.
- The ability to use and communicate using diagrams and figures and tables.
- Understanding of a wider range of linguistic constructions, a larger lexicon, more kinds of occurrences and more relations among them.
- Access to more "background knowledge" about the world of everyday objects and events.

It seems to me that much of the material in a standard college physics book could be mastered by the basic approach described here. I would expect roughly a B– competence to be achievable in many other domains of physics, in addition to kinematics.

In fact some other domains of physics might even be more easy than kinematics. Some domains of physics require such specific kinds of situation, and require that the quantities characterizing them be listed explicitly, that solving problems really amounts to little more than listing the set of quantities given and solving for the rest. For example "Bernoulli's equation" describes the relation between the pressure and velocity of a fluid moving through a pipe of varying thickness and height. It is difficult to describe a situation in which Bernoulli's equation could potentially be applied without mentioning that a fluid is involved, and some container, and giving some subset of the pressure and the velocities and the thicknesses and the height. And once one has given all of those values, there aren't many *other* equations that might apply.

Some aspects of physics will always be beyond the simple "situation matching" mode of Sagredo's reasoning. Extending a program to deal with those aspects will require better models of physical situations and more sophisticated reasoning techniques. I expect that the approach I have pursued here will eventually connect with the investigations of "qualitative physics" whose goals are deep and accurate computational models of physical systems. The linguistic physics I explored here will continue to be an important part of the reasoning of such systems, even if it is only needed to convert the problem descriptions from English to whatever form the more sophisticated models require.

Many domains of engineering require relatively simple physics applied to fairly complex situations. These ought to be domains where a Sagredo-based program could be successful. As I mentioned, however, engineering domains, more than physics,

often involve unstated general requirements, and complex tradeoffs among multiple factors. Some of these could probably be implemented relatively straightforwardly, and if the tradeoffs were among numerical quantities, it might be possible to use mathematical optimization techniques. A true understanding of what really ought to be traded off against what requires a much fuller participation in the world of human values than a program is capable of achieving any time soon.

Still some engineering is a lot like physics word problems in that it involves computing the value of some quantity given a number of other quantities and a description of a situation. And, as I mentioned, in engineering it is often the case that there are many very specific equations that might be relevant, and so the shallow understanding that Sagredo achieves would be useful.

For a lot of college physics, equations and simple diagrams suffice. The main mode of problem solving is that involved in setting up and manipulating equations. But a number of other scientific and engineering domains involve different styles of reasoning. For example mechanical engineering requires an understanding of shapes and spatial relations, an understanding similar to what is deployed in organic chemistry. This reasoning is quite definitely not like the simple linguistic reasoning needed to compare presentations stated in English with problems also stated in English, and I don't imagine that a Sagredo-like program could master such domains without new reasoning abilities in these specific domains.

One domain in which a lot of the reasoning involves linguistic descriptions is that of the law. In the legal domain, the "presentation text" is either a statute or the decision of a court, and the "problem" is the description of some real situation by an advocate or a witness. The job of the jurist is to determine which of a great number of different potential cases best characterizes the one under consideration. This reasoning is in some ways similar to the "story subsumption" that Sagredo is capable of.

The legal domain takes this process several steps further since the process of determining whether a precedent subsumes a case is itself a matter of interpretation, and there may be precedents about how to decide how to decide whether a given case or situation applies. And so forth. While the need for a much deeper understanding of real world knowledge is apparent, this does seem to be another domain in which it would be fruitful to ask about the specific linguistic resources relating a given "problem" text with the "presentation" text appropriate to it.

Chapter 10

The Rhetoric of Physics Textbooks

The successful operation of Sagredo relies on the fact that a lot of physics passages operate as if they were straightforward logical arguments. Terms are defined and associated with equations. Physical laws are presented as axiomatic (with perhaps a glance at their history). Special case applications are described and the relevant equations are derived in an apparently valid combination of logical and algebraic reasoning. It all seems neat and tidy.

10.1 The Unkempt Foundations

But it isn't. And it can't be. Even a valid argument in formal logic requires first establishing the set of axioms and premises and rules of inference that will be used. And those can't be proved the same way. At some point logical argument must ground out and some other means must be found to establish agreement and understanding.

I was forcibly impressed by this fact in the course of my reading and analysis of physics text. As I traced the arguments back to their assumptions and premises I noticed the logical quality of the arguments degrading. The texts spent much more time dealing with intuitions or presenting quaint examples from everyday life, or correcting expected errors and misconceptions.

"Rhetoric" can be defined succinctly as the craft of persuasion. Especially for the more fundamental concepts, physics textbooks must do more persuading than simple informing. The student will, for example, have to learn the definitions of new terms, or will have to learn to use familiar words in unfamiliar ways. The student must develop intuitions that are accurate and appropriate for physics, and must banish whatever intuitions are inconsistent with physics. The student must learn to be persuaded by arguments that the physics community has deemed to be valid.

None of this can be done by straightforward logical argument because such argu-

ment requires that the text and its audience accept the same set of premises and rules of inference. But this is especially not the case for an undergraduate textbook where the explicit point is to *train* people to enter the community of physics. Perhaps when they are done with their undergraduate education they will have come to accept and share the intuitions of the community. But before that they have to be persuaded by other means.

The analysis of the rhetoric of textbooks begins by locating passages where the texts explicitly call attention to the use of terms or seek to correct or demonstrate ways of thinking or arguments that should be understood and followed. The next steps involve trying to figure out the means by which the persuasion is being achieved. Of necessity this often involves reading in to the choices of wording and examples perhaps more than anyone intended. But I am interested mostly in the effects that these texts have on their readers, not necessarily whether anyone intended them precisely as I have analyzed. After all, the authors of these texts went through an initiation into the field much like they are giving their readers, and so might see their rhetorical techniques as transparent.

The following three sections present some passage from physics texts where the rhetoric is relatively thick. Clearly not a lot can be concluded from a handful of such passages. But they do illustrate the general idea. And as I argue in the last section of this chapter, they also illustrate what I think is a profound critique of the approach I took in the design of Sagredo.

10.2 The Meanings of "Work"

The physical quantity named "work" is defined (in the simplest case) as the product of the magnitude of a force and the distance a body undergoing the force moves. Most textbooks include discussions like those shown in figure 10.1 alongside their presentation of the precise mathematical definition of the term. Two points about these passages can be made immediately. The first is gratuitous: it is testimony to the uniformity of college physics textbooks in this country that these two different books get around to discussing the same issue within 5 pages of each other! The second point is that the term "work" *is* confusing to a lot of students, and for the reasons adduced in the passages. Something has to be said about it. I am interested in how it is said.

The first passage precedes that text's definition of the term "work." In fact the first two sentences of this passage open the chapter titled "Work and Energy." Thus the first topic of business is the relation between the use of words in "everyday life" and that in physics. The first sentence [lines 1–2] contrasts the relatively careless application of the term to "any form" of activity, with the "very specific sense" of the term in physics [lines 2–3]. Whatever the attitude this passage is expressing with

1 In everyday life, the word *work* is applied to any form of activity that
2 requires the exertion of muscular or mental effort. In physics however,
3 the term is used in a very specific sense.
4 ...although it would be considered "hard work" to hold a heavy ob-
5 ject stationary at arm's length, no work would be done in the technical
6 sense because there is no motion. [SZY, pp. 113–114]

7 Work as we have defined it proves to be a very useful concept in
8 physics. Our special definition of the word "work" does not correspond
9 to the colloquial usage of the term. This may be confusing. A person
10 holding a heavy weight at rest in the air may say that he is doing hard
11 work—and he may work hard in the physiological sense—but from the
12 point of view of physics we say that he is not doing any work. We
13 say this because the applied force causes no displacement. ... In many
14 scientific fields words are borrowed from our everyday language and are
15 used to name a very specific concept. The words "basic" and "cell,"
16 for example, mean quite different things in chemistry and biology than
17 in everyday language. [Resnick & Halliday, p. 119]

Figure 10.1: The Meanings of "Work"

respect to everyday life, it is making the point that the use of terminology in physics is an important matter, and that the correspondence between the physics use of the term and that of everyday life will be limited at best.

The next sentence [lines 3–6] comes from several paragraphs later in the text, and is part of an extended discussion of the work done in various situations. The point here is that one's experience of exertion is not to be correlated with the physical notion of work. The passive verb form "be considered" and the use of quotation marks in the phrase "hard work" serve to distance the text from the (physically improper) use of the term.

The second passage begins on a friendlier note. The first sentence begins by offering an apology for the new definition. The second and third sentences address the confusion created by the "colloquial usage" of the term [lines 8–9]. The term "colloquial," even more than "everyday life," seems to suggest that the non-physics use of language is somehow informal in a way that physics language is not.

The next sentence [lines 9–12] addresses the same situation as in the previous passage. Once again the text distances itself from an erroneous statement, this time

by putting it in the mouth of the person holding the heavy weight. Furthermore the clause between the dashes suggests that there might be a technical sense of the word "work" in the science of physiology such that holding a heavy block at rest does indeed involve such work.

This is a very misleading thing to say because of course the "physiological sense" of the word "work" is exactly the same as the physics sense of the word. It is just that the felt experience of exerting physiological effort is not necessarily associated with the doing of work on external objects. Feynman [98] (page 14-2) addresses this issue squarely and discusses how the movement involved in the twitching of muscle fibers entails that actual physical work is done when one is holding a weight at rest. The point that *ought* to be made here is that no work is being done *on the weight.*

The next sentence [lines 12–13] is similarly imprecise. It is not important whether the force *causes* the displacement or not. For example gravity is treated as doing (negative) work on a body as it is lifted. Indeed the notion of "cause" is never given a technical sense in modern physics at all. But the tone of this sentence is interesting. The subject is the word "we," meaning, I suspect, the community of physicists, who apparently know what they are talking about. So even though it is not quite correct physics to require that the force cause a displacement for the force to be associated with a work, that heuristic is good enough for many important situations. Physics language, like "everyday" language, is often imprecise and, again as in everyday life, the imprecision often does not get in the way of effective communication.

The last two sentences [lines 13–17] describe how this process of appropriating words is standard practice in scientific disciplines. Whether this is meant as an excuse or as exposition is unclear. The claim made by the first of these sentences is that such names are used for "very specific concepts." And then the second sentence gives as examples "basic" and "cell." The former term is an interesting example because the "everyday" use of the term "basic" is entirely different from the chemistry sense. At least with the word "work" there are cases (for example when lifting boxes) when the everyday usage and the physics usage coincide. However it would seem that one never uses the word "basic" in the chemistry sense unless one knows the relevant chemistry. It could be even argued that the two senses are completely different: the "everyday" sense having to do with fundamental and perhaps simple matters, the chemistry sense having to do (originally) with compounds made of the base metals.

I am not sure what the point is about the example of the word "cell." The sense of the word where it describes a small room or compartment is entirely disjoint from the biological sense (although Leuwenhoek introduced the name for the latter from its resemblance to the former). Perhaps it is the case that there is a colloquial use of the word "cell" that overlaps the biological sense. But this would seem to be a case where the colloquial sense is just wrong. The success of the theories of work and energy did not change the fact that it is still correct and informative to say that, for example, a philosopher does work, even though no masses may be moved through any distances in the course of it. On the other hand, the cell theory of biology introduced

a new sense of the word to the language, a sense which is "owned" by biologists. If someone were to argue (for example) that a hair is a cell, this could be shown to be false biologically and thus just false. The colloquial sense of the word "cell" just *is* the biological sense, while the word "work" has a completely valid life of its own outside physics.

These two passages illustrate a number of general points about the rhetoric of physics textbooks. The most obvious is the fact of the appropriation of language by the scientific discipline. Clearly in a matter of dispute, being able to impose one's definitions of terms is a tremendous advantage. And having precise definitions, and sticking to them, is necessary if one is to make reasonable progress in any inquiry.

The student must learn to accept the importance of this process as well as its results. So the texts pay attention not only to the new definitions, but also to the process of definition itself. As I mentioned above, it is perhaps inevitable that *some* attention must be paid to the redefinition of such a familiar word. On the other hand it serves a useful rhetorical purpose to have such a familiar word defined so early in physics because this focuses attention on the process of definition. Texts also pay some attention to the precise definitions of words like "velocity" and "force" and "mass" but is this with word that the differences between the standard meaning and the physics meaning show most clearly.

Another important issue illustrated by these passages is the ambivalent relationship that textbooks maintain between physics and everyday life. On the one hand the textbooks seek to ground the plausibility of arguments and examples in the readers' experiences with motions and forces and weights. On the other hand, the intuitions based on those experiences are not to be trusted, for a large number of reasons, including the fact that the real world, with friction and air resistance, is not the ideal world of Galilean kinematics. That the texts even attempt to make use of everyday intuitions is interesting. My hunch is that the textbooks are attempting to get the reader to learn to focus on those aspects of experience which are consistent with physical theory. The rest must be discarded or treated as suspect.

So the reader must come to suspect all of those intuitions that are based on everyday experience and must learn to analyze them from the standpoint of the theory being learned. The student must develop appropriate "physics intuitions" by analyzing examples mathematically. This is of course necessary, but it is a difficult feat rhetorically. Ordinarily, a real argument grounds out when the participants share common intuitions (or not finding any, fails to ground out). But the texts are explicitly denying the validity of such intuitions in their readers. Some other means have to be used to get the reader to go along.

The next two passages will illustrate how this is done more clearly but I think that the general idea is illustrated by the word "we" in line 12. That pronoun is referring to a community that the reader/student is seeking to join. The argument thus grounds out not in any existing shared intuitions, but in the fact that the readers

1 The average velocity can always be found by dividing the displace-
2 ment by the corresponding time interval. For example, if a ball having
3 an initial velocity of zero rolls down an inclined plane, it gains speed
4 at a constant rate. If the ball rolls a total distance of 20 m in 5 sec its
5 average velocity is 50 m ÷ 5 sec or 4 m/sec. If is reasonable to expect
6 that the results would be the same if the ball had moved the full length
7 of the plane with a constant velocity equal to the average velocity of 4
8 m/sec. Since the initial velocity of the ball is zero and the average is
9 4 m/sec, the velocity at the bottom of the plane, or the final velocity,
10 must be 8 m/sec.
11 That is, the average velocity v_{av} is half the sum of the initial velocity
12 v_i and the final velocity v_f:

$$v_{av} = \frac{v_i + v_f}{2}$$

13 This is to be expected. The average of two numbers is half their
14 sum. [Weissman, p. 42]

Figure 10.2: Average Velocity

---

recognize that it *does* ground out for members of the physics community, and must work to acquire the intuitions that will make it ground out for them.

10.3 Average Velocity

The next passage is from a high-school textbook. The text has just defined average velocity as the quotient of the displacement and the time taken: $v_{av} = d/t$. Then follows the discussion shown in figure 10.2.

As I mentioned in the context of Sagredo's encounter with this equation, the relation expressed by the equation after line 12 of this passage is the one place in the presentation of kinematics where something beyond simple algebra is needed. Once this equation is introduced, the rest of the kinematics equations can be derived, as they were in Sagredo's kinematics source text. Different textbooks deal with the problem in different ways. Some present the equation as a purely empirical fact. Others motivate it with graphs of distance versus time. And a few define instantaneous velocity as the derivative of displacement with respect to time, and derive the equation by integration.

Galileo makes use of a geometric construction in his proof of this relation.[1] He draws a diagonal line whose distance from a vertical line represents the speed of the object at each moment. He then points out that the area of the resultant triangle is equal to the area of a rectangle of the same height (representing the time) whose width is half the difference between the final and initial velocities. But he doesn't seem to realize that the area of the triangle is proportional to the distance traveled. Instead he proceeds to argue that "what the momenta may lack in the first part of the accelerated motion is made up by the momenta" in the second half of the motion, and so can be made to correspond to the "momenta" of a body moving with uniform motion. It is difficult for me, working with a translated text, to determine if Galileo's proof is really valid. But at least he addresses the issue carefully.

Weissman, on the other hand, resorts to pure chicanery. Remember that this text is supposed to be for high school students. On the first page of the text he addresses a typical reader as an "intelligent young person." So let us consider this passage as such a person might read it.

The first sentence repeats the just-given definition of average velocity in words. The phrase "can always be found" is a bit peculiar because, after all, the operation described is the definition of the quantity. Perhaps the point is that the definition of average velocity doesn't involve any velocities, but only displacements and times. It is in some sense an interesting fact therefore that one can determine the average velocity without knowing any actual velocities whatsoever.

The next sentence [lines 2–4] describes Galileo's experimental apparatus and simply states as a fact that the object accelerates uniformly. The following sentence [lines 4–5] provides some values for the relevant quantities and computes the value of the average velocity given those values. So far everything seems fine.

The next sentence [lines 5–8] describes a ball moving at at constant velocity equal to the average velocity. The sentence says that such a ball would move the same total distance in the same time. But it makes this claim in a sentence introduced by the phrase "it is reasonable to expect." What is the point of that introduction? The reason that it "is reasonable to expect" that result is that that is what would happen, according to the definitions of average and constant speed. It is not a matter of reasonable expectations at all, but of physics.

The final sentence of the paragraph [lines 8–10] springs the trap. It is phrased in the form of an argument: it begins with the word "since" and the last verb is "must." But look what is going on: the concept of "average velocity" was defined as a quotient between distances and times, it is not, as the name might imply, an "average" of "velocities." But suddenly we are using "the average" and the initial velocity to compute what the final velocity "must" be. Suddenly the notion of average velocity has been changed to mean "the average of the velocities."

The game continues in the next paragraph where the first sentence now just treats

[1] It is Theorem 1, Proposition 1 of The Third Day in his *Two New Sciences* [115], p. 208.

average velocity as being the average of the initial and final velocities. The original definition above is completely ignored. Furthermore, the text goes on to say that "this is to be expected" [line 13]. Is this the same sort of expectation that was "reasonable" above? And finally it is mentioned that "the average of two numbers is half their sum" which is true, but is relevant here only because the acceleration is constant.

This may in fact be the worst example I have found of how a textbook resorts to mystification in the presentation of absolutely fundamental physics. And it would certainly seem that it should just be fixed. But I think that the tone is interesting and revealing. It *appears* to be an argument. There are many attempts to connect with the intuitions of the reader, and many uses of terms like "reasonable" and claims that results are "to be expected" and so forth. The reader is supposed to follow this, or is supposed to think that it should be followed. Even though it doesn't follow. Ultimately all the student can do is to accept the author's word that it does follow, and get on to the real business of solving problems and preparing for the SATs. The unfortunate result, which I think happens in lots of cases, is that the student comes away feeling that physics isn't as much to be understood as submitted to.

This passage illustrates quite well in its use of the words "reasonable" and "it is to be expected" how the student's intuitions are being explicitly shaped. Whether or not readers find any of these arguments convincing, they understand that they *ought* to find them convincing. They are beginning to understand how valid arguments in physics look. Eventually they will develop the intuitions needed to make them *feel* valid also. What is tragic in this passage is that no physicist would claim that the argument is in fact valid.

Not all texts are as devious as this one. But there is often a tension between the complete story and what the student needs to know or can be expected to understand at any point in the presentation of a subject. Some books are careful to present all of Galilean and Newtonian physics as an approximation. Others might make some mention of relativity and quantum mechanics before they start. Others just get on with treating classical physics and then deal with retracting its assumptions. In texts like Weissman's I think it is clear that the dishonesty should just be removed. But there might be positive value in a bit of dishonesty: students must be willing to change their understanding as the science proceeds. They shouldn't trust their books too much.

But this is too much like the excuse of the professor who, found to have made an error at the blackboard, claims "I was just testing you."

10.4 Centrifugal Force

The passage in figure 10.3 illustrates a case where an obscure issue is "explained" as much by intimidation as by anything else, The text has just presented the analysis of the motion of a pendulum bob attached to a string and free to swing in a horizontal circle. The passage begins by describing the wish of "some readers" [lines 1–2]. It will become clear that any actual readers will not want to be included in this group despite the fact that they may well have similar wishes.

First note that the subsequent discussion is not superfluous to the issue of problem-solving. The method of analyzing a dynamic system by drawing vectors for each force on a body has been developed in earlier chapters, and the derivation of the equation of motion for the circular pendulum makes sophisticated use of it. This method is a precise way to compute the total force on a body but it requires that its user draw only the actual forces on the body. So drawing an extra force will have specific deleterious effects in the use of this method. It is (apparently) a common problem, and must be attended to.

These "some readers" are an interesting group. Obviously the writers expect that some of the readers of the text will have the indicated wish. But the text is apparently addressed to another group of readers. The point I think is that the "desired" audience of the text consists of those whose physics intuitions are honed to the point where no such wishes would be entertained. But the "expected" audience of the text includes a subset who are not yet thus attuned. The explicit task of the text as a whole is to transform its audience into a part of the physics community. So the text is addressed to that community even though many of its readers are known not to be there yet.

Imagine how a reader might respond who has the wish described in the first sentence. As soon as the word "wish" is encountered, such a student will realize that something is wrong. Physics is not a matter of wishes, it is a matter of solid, valid, concrete facts. Perhaps such a reader indeed framed the reason for the wish in one of the phrases illustrated. Such a student is going to be in a very unstable position during the following passages. On the one hand, such students are explicitly excluded from the discussion: their thoughts are put in quotation marks, and the text is about to step back and "examine" their "point of view" rather like a pathologist might examine a diseased organ. On the other hand, it is such students who are supposed to learn the most from this passage. While any of the text's readers might be expected to sharpen their intuitions by following the subsequent argument, it is only the "some readers" who specifically need the correction it offers. The text is in some sense forcing these readers into the position of trying to deny their intuitions. To be valid readers of the text they must not be the very readers that the text is intended to help.

There are two reasons why such misguided souls might want to add a new force to the diagram [lines 2–3]. Both of the reasons are presented in quotes, putting the

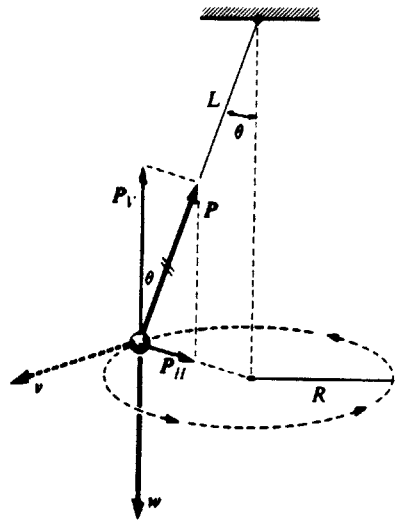


Fig. 6–13 *The conical pendulum.*

Some readers may wish to add to the forces shown in Fig. 6–13 an outward, "centrifugal" force, to "keep the body out there," or to "keep it in equilibrium." ("Centrifugal" means "fleeing a center.") Let us examine this point of view. In the first place, to look for a force to "*keep* the body out there" is an example of faulty observation, because the body doesn't stay there! A moment later it will be at a different position on its circular path. At the instant shown it is moving in the direction of the velocity vector v, and unless a resultant force acts in it, it will, according to Newton's first law, continue to move in this direction. If an outward force *were* acting on it, equal and opposite to the inward component of the force P, there would be no resultant inward force to deviate it sidewise from its present direction of motion.

Those who wish to add a force to "keep the body in equilibrium" forget that the term *equilibrium* refers to a state of rest, or of motion *in a straight line* with constant speed. Here, the body is *not* moving in a straight line, but in a circle. It is *not* in equilibrium, but has an acceleration toward the center of the circle and must be acted on by a resultant or *un*balanced force to produce this acceleration. In this example there is *no outward force* on the body! [SZY, p. 101]

Figure 10.3: Centrifugal Force

phrases in the mouths of the "some readers." The parenthesized remark in line 3 is interesting, and it ought to be suspicious, or perhaps comforting, that the incorrect view is associated an apparently technical term, complete with Latin etymology.

The next sentence begins to "examine this point of view." This sentence frames the subsequent discussion, but also distances the text even further from the "point of view" under consideration. It is being made quite clear here that this point of view is not appropriate for physics. The next sentence begins "in the first place" [line 5], which provides the rhetorical effect of alerting the reader than a barrage of arguments is about to come—even if the first argument is not convincing, more are on the way. The objector might as well give up now.

I think that this sentence is very sloppily argued. The original wish of the "some readers" was to *add* an additional force. This sentence replaces that with a *search* ("to look") for the force, and suggests that performing this search "is an example of faulty observation." I guess that the point here is that the initial faulty observation is what led to the wish which is what leads to the search for the force. Claiming that someone is guilty of "faulty observation" is about as damning as one can get in as rigorously empirical subject as physics.

But this sentence doesn't even respond to the initial wish. The original reason was stated as "to keep the body out there." The rephrasing of that wish in this sentence emphasizes the word "keep." It is then pointed out that "the body doesn't stay there." But note that the word "out" has been dropped. The intuition behind the initial wish was not that the body stays anywhere—of course it is moving—but that somehow there is something to be explained in the fact that the pendulum is not vertical.

Rather than respond to this intuition, the next few sentences [lines 6–12] explain what would happen if either no force were acting on the body, or if a force equal and opposite to the inward force were acting on the body. But neither of these are necessarily what the reader had in mind in suggesting that a new force be added to the diagram. I get the strong impression here (and in the next paragraph) that there is no interest in responding to the naive intuitions at work here. The physical reasoning described in these sentences is perfectly valid and useful and in fact deals with the consequences of the intuitions, but does not address them directly. It seems almost as if the faulty intuitions are just going to be banished.

The first sentence of the second paragraph [lines 13–15] responds to the other reason that a reader might give for adding a spurious force. This response pays even less attention to whatever intuitions might drive the question. Instead the text reminds the reader that physics has appropriated the word "equilibrium" for its own purposes. No other use of the term is allowed, no matter that the "everyday" English sense of the word would seem to allow that something like equilibrium exists for an object moving smoothly in a horizontal circle.

The final sentences in this passage are amusing to me in their almost violent sup-

pression of the incorrect reasoning. Negative words and morphemes are emphasized ("*not* moving ...," "*not* in equilibrium," "*un*balanced force"). And the final sentence simply states the negation of the original intuition, this time all emphasized: "*no outward force.*"

I believe that the intuition that prompts the initial question is the Aristotelean one that every effect has a specific cause, and that, the pendulum bob being "out there" is an effect which must be explained. This intuition is combined with the vague understanding that causation in modern physics involves forces. It is similar to the question of "what holds the moon up?" As if something, presumably a force, must do so. But this model of causation isn't really appropriate for this situation. Once the initial mathematical specification of the system is given, the "explanation" such as it is, of the dynamics of the system is given by the equations of motion. There is nothing "causing" the bob to stay out there, unless we take a very wide view of causation and say that Newton's laws are causing it. Perhaps the most correct answer is to say that whoever or whatever set the system in motion "caused" is subsequent behavior, but agency is often not dealt with in physics. All of this is interesting and a great deal of discussion in the philosophy of science concerns the differing roles and types of causal versus mathematical explanations in modern physics. But I think that it is clear that the text is not interested in dealing with those issues. Once again, the student is being shown an argument that must be understand as being persuasive, whether or not it is actually felt to be persuasive. One of the things that will be learned is to be persuaded by such arguments.

I should also mention that the issue of "centrifugal force" is a controversial topic in physics textbooks, one of the details on which modern books show the most diversity. Some of them, like this one, simply deny that any such force ever exists. But other books use the term to refer to the non-inertial force that would be felt by a body attached to the revolving pendulum bob. Indeed Feynman [98] begins with the intuitions about such non-inertial forces to begin a discussion of the theory of general relativity!

10.5 Learning as Entering a Community of Practice

I don't intend to consolidate these observations into a coherent theory of textbook rhetoric. However I think it important in describing my project to point out how small a view of physics books is one that treats them as merely stating sets of facts or as presenting sequences of logical arguments.

I mentioned several times above the idea of a "community" of physics. This notion is based on an approach to the study of learning due to Lave and Wenger [203] wherein learning is viewed as entering a community of practice. Where a "cognitive" perspective on learning would focus on the specific facts and skills that an individual

acquires, this alternative focuses on the external, social aspects of the acquisition of expertise.

A "community of practice" is a group of people who work together to perform some social function, for example workers in a factory, or lawyers in a law firm, or programmers in a research laboratory. The boundaries of a community of practice may be more or less sharp and certainly open out to larger communities in various ways. So an individual law firm is part of the legal community in a particular city, which is in turn a part of the United States legal community. And a particular research group is part of a larger laboratory and its members may be part of professional associations and so forth.

Participating in a community of practice involves much more than just knowing a set of facts or having particular skills. Indeed the bare knowledge of facts is clearly inadequate. One must be able to participate in what the community does, and that involves knowing how as much as knowing what. And knowing how, involves getting practice and being given responsibility. This involves the community accepting a new member as a legitimate participant, and allowing that new member to begin to participate in the activity of the community, at first as a peripheral member, and then as a more and more central participant of the community.

The main illustration of this view of learning is the role of the apprentice. The apprentice is at first assigned relatively straightforward, drudge jobs that involve little or no experience of the field. As apprentices performs these tasks, under the guidance of the master, they will be given more and more responsibility, and the jobs assigned will be more and more central to the function being performed by the community. But also, by being in contact with the practicing members of the community apprentices will absorb other aspects of the culture than just the details of the job. They will hear "war stories" [236] of particularly difficult jobs, they will learn the jokes and myths and legends of the field, they will see behavior that is admired and behavior that is scorned.

Now obviously there is a problem in characterizing the "community of practice" that a reader and user of a physics textbook is entering. Clearly one candidate is the communities of scientists and engineers, but then not everyone who takes physics ends up being a scientist or an engineer. And in any case, that will happen several years later. A more local "community" is that of the members of the class or other students taking the same course. The problem here is that this is harder to characterize as an ongoing community since it usually lasts only for a single school term.

But on the other hand there is some aspects of the community of practice at work. There is clearly the "elder members" of the community present—the instructor and the teaching assistants. And the textbook itself often speaks with the authority of the whole community. The actual "practice" of this community is limited to those of taking a course in physics: the classroom, doing problem sets, perhaps doing laboratory work.

I believe that by taking this wider view of learning, much of what appears weird or perhaps irrational in physics books can be understood. The passages in question can often be understood not as teaching the facts of physics but as playing some role in the creation and the maintenance of the physics community.

It is important, on this interpretation, to understand that for whatever reasons, the student *wants* to join this community. The student is therefore willing to accept what the books say as much as the means they use to present the information. The student must recognize that this new community (like all communities) will use certain words in characteristic ways, and will have certain attitudes about various aspects of the work that it does. In deciding to study physics, the student has decided to accept what physics has to give.

Whether this is the way that physics education *should* be, I have no competence to judge. Certainly the more egregious passages, like those of Weissman, ought to be fixed. But physics just can't be understood without some modification of one's "everyday" intuitions, otherwise it wouldn't have taken so long to be worked out by scientists.

What I think is more important here however is that the observation that physics textbooks and physics as a whole makes use of "extra-logical" means of persuasion seems to contradict some of the claims that physicists and philosophers of science make about how it works. Physics is often held as the model of the empirical and deductive method, and, as such, is sometimes even taken to be the ideal of cognition itself. Now perhaps physics is indeed worthy of that distinction. But if so, the ideal should be based on what physics actually is, not how it is idealized as being. Physics is a social practice, a collaborative project among many people. The history of physics is the story of a long discourse about the physical world, carried on by participants whose outlooks vary widely, yet who believe that they have something to share with one another. Some work has been done on the actual sociological and rhetorical processes underlying scientific research and education, for example Gross' discussion of the rhetorical techniques in scientific papers [131], Jordan and Lynch's study of the sociology of a biology laboratory [175], and Traweek's account of the apprenticeship process of physicists [291].

10.6 Self-Critique

Beyond the specific technical limitations described in the last chapter, I believe that there is something utterly false about the approach to language I took in the design and implementation of Sagredo. I want to briefly outline the reasons for that assessment and consider whether any computational approach to language will of necessity share the same inauthenticity.

Before I do that I want to emphasize that the goal of this research was not necessarily to elucidate the fundamental nature of language. This project was motivated by

a more or less pure engineering goal: to build a program that could read some realistic kinematics text and learn enough to solve word-problems. In doing that piece of engineering I availed myself of whatever computational and conceptual apparatus seemed useful, independent of any theoretical implications the use or the success of such tools might have. The fact that, as I will discuss, these tools are motivated by what I feel is an inadequate conception of linguistic activity, does not of necessity make them bad tools. Indeed it might even make them very good tools, given that considerations of computational tractability were strong influences on their development.

Still, as I described in this chapter, I was struck, when analyzing the style of argument used at some places in textbooks, at how little a clean, logical conception of language accounted for what I was seeing. And then, as I began writing rules for Sagredo, and "defining" words by writing programs that constructed just the structure needed to solve problems, I was very aware that what I was building, however well it worked, was not very much like real linguistic activity. And it is not that having seen all of the myriad details of the system I can no longer (if I ever could) think of it as "intelligent." The problem is that those details are not like the details of language, and the differences are essential.

I think that the best way to get at what bothers me is to think about the rules that a program like Sagredo is organized around. The contribution at the heart of modern linguistics is, arguably, Chomsky's [55] suggestion that linguistic structures could be characterized in terms of the operation of abstract machines. So that the structure of a sentence, for example, in terms of its analysis into a noun phrase and a verb phrase and so forth, was, in the ideal case at least, was a purely formal matter: a machine, with no understanding of the meaning or potential uses of the sentence, could determine the structure.

Linguists had been characterizing the abstract structure of language at least since Aristotle[2], but these characterizations had always been relatively informal. Aristotle, for example, gave no indication of whether or how the structure of a complex plot is qualitatively different from the structure of a sentence. But Chomsky's proposal made possible a much more precise approach to the characterization of linguistic structure. Whether or not the plots of tragic plays could yield to such a structural analysis, it seemed that sentences would. Thus linguistics could define itself the project of characterizing the possible abstract structures of a particular language, and potentially, of all possible human languages.

Given that this new approach to the understanding of linguistic structure is based on the operation of abstract machines, it was natural that computational linguists should attempt to program computers to recognize and manipulate such structures. And ambitious projects, of which the one reported herein is an example, extended the realm of computationally manipulated structures out of pure syntax into the areas of

[2]In *The Poetics* [9], chapter 20.

semantics and discourse analysis. In all of these new areas the computational aspects of the formalisms remain central, whether for theoretical or practical reasons.

For example the assignment of semantics to sentences based on truth values or "logical form" is also motivated ultimately by a theoretical approach that values the formality of the theoretical objects. All that is important, theoretically, about the syntactic and semantic structures involves, are their formal properties. Since those properties are the only ones that a computer has access to, computational approaches accede to such requirements comfortably.

One thing that this approach does not address is where the rules and structures come from. The linguistic system is treated as a fixed entity, consisting of a set of rules and lexical items, combinations of which can be interpreted as having specific semantic properties. The mechanistic approach also does not treat the creative or novel aspects of language use. The speaker of a sentence is viewed as creating a structure according to those rules that, when interpreted against the same rules, yields the precise message intended. (Many of these issues are discussed by Harris [143], who suggests that such a mechanistic approach to language might have been inevitable coming, as it did, from an "Institute of Technology.")

This whole approach works fine for Sagredo because the sentences do have a specific "logical form" in the kinematic equations, and each sentence can be given a specific and precise role to play in the presentation of new material or its testing in problems.

But the rhetoric passages illustrate what seems to me is going on underneath, and what may be, fundamentally more important for the understanding of language. The rigid rules that Sagredo interprets, and the paths it follows while interpreting them are like the well-lit center of a complex system of corridors. So long as the program can keep to the center of the corridors and can walk straight, everything is fine. However the model fails as the walker begins to grope along the walls, or loses sight of the lights entirely.

The rhetoric passages are the attempts of the text to provide some guidance in this logical periphery. The guidance takes many forms, not always logical or systematic, or especially friendly. And the goal of the guidance is always to get the reader back on track, back into the mode of understanding and following the step by step elucidation of physical principles.

However nice the corridors that these rules define are, they are not the whole reality of language. The corridors can only be properly explained by understanding how deviations from them are handled, and what mechanisms exist to correct for those deviations. But those mechanisms can't be the same sort of formal mechanisms that recognize abstract linguistic patterns because they require attention to the meanings of expressions, and the interests and concerns of the participants. The rhetorical passages, for example, can only be understood in terms of a student who wants to be a physicist, who doesn't want to be insulted, who needs to pass the class to keep

getting financial aid.

The rules that I used to implement Sagredo are the fossilized remnants of the operation of those informal processes of guidance and negotiation and maintenance that underlie any real social activity, in particular language use. While working out such formal summaries of linguistic activity might be a useful data-gathering step, it can never capture the underlying real operation of language. Language is human activity, and as such, is subject to everything that influences the other aspects of human activity, from wants and needs, to social and historical influences. A complete theory of language ought to take all of this into account.

Chapter 11

Related Work

11.1 Word Problems

Bobrow's "Student" program [30] is the original progenitor of Sagredo. Student solves simple algebra word-problems stated in English. The program works because a large number of algebraic values and relations are expressed in easily recognizable ways. The program locates such expressions and transforms the input sentences into LISP expressions that can be manipulated and solved for the desired quantities.

For example the sentence "The number of geese is twice the number of hogs" expresses the relation $g = 2h$, where g is the number of geese and h is the number of hogs. The verb 'is' expresses the '=' relation and the word 'twice' expresses the operation of multiplying by two. In this example, the relations are all expressed directly in English. Student has a small amount of knowledge built into it that can be deployed when characteristic expressions are used. It has access to formulas for speed and the areas of geometric shapes, for example. Charniak's system [51, 52] was similar to Student except that it could solve calculus problems.

Novak [232, 233] describes a program named "Isaac" for solving physics word problems stated in English. All of the problems that Isaac can solve involve rigid bodies in static equilibrium. The program is organized around a set of "object frames" that describe the relevant features of the objects that participate in the problems, for example levers, pivots, weights, ropes and pulleys. As the sentences of a problem are parsed, Isaac constructs an internal model out of instances of these object frames. When the relations among objects are expressed by the problem statement, the corresponding relations are imposed among the object frame instances. In general these relations (such as support or attachment) introduce algebraic relations among quantities. When the problem statement is complete, the set of equations can be solved for the desired solution.

Problem-solving in Isaac is thus almost exactly the same as it is in Sagredo. The most obvious difference is just that Sagredo's "frames" describe occurrences rather

than objects, but this makes sense because Sagredo solves problems about motion, and the relevant relations are among the parameters of motion occurrences, like distance and duration, instead of the forces and masses that Isaac deals with.

The most important difference between Isaac and Sagredo is that Sagredo learns the frames that it deploys. Its ability to do so is a result of the other important difference between the two programs: Sagredo's occurrence representation is as close to the linguistic level as possible. This makes it possible to easily construct the occurrence descriptions as new concepts are learned, and it also makes it possible to easily construct them when they are used to solve problems or to follow arguments. Isaac's model frames were designed with no such constraints in mind, although there was certainly some attention paid to the problem of constructing them as the text was read.

A number of subsequent investigations have grown out of Novak's work. Bulko [44] developed a more sophisticated approach to the understanding of the kinds coreferring expressions that appear in physics word-problems. Kook [195] further elaborated the knowledge representation aspects of the approach. Along with being able to solve word-problems, Isaac also used the object frame representation to create a diagram of the problem situation. The use of diagrams in solving word problems was further investigated by Novak and Bulko [234].

Another project investigating the problem of solving physics problems is that of Bundy, et al. [45]. Again, the approach involves the manipulation of pre-programmed templates that are instantiated as the text is read. The initial focus of Bundy's group was on physics problems as a domain in which to investigate meta-level reasoning. The program was able to determine which of a number of strategies ought to be used to organize the solution for specific kinds of problems. Sagredo doesn't need to make use of such strategies because the kinds of problems that it can solve all involve cases where the problem statement instantiates all of the occurrence descriptions needed to solve the problem. So the problems can be solved by manipulating the constraints thus introduced. More difficult problems, where a search must be done to locate the proper occurrence descriptions to use, would probably require the sort of search control reasoning investigated by Bundy's group.

A more recent investigation from this group is described by Palmer [237]. Her focus is a linguistic analysis of word problems in a very restricted domain (pulley problems). Just about every other project that involves solving word problems (mine certainly included) has cheated a lot in the details of the connections between the syntax of the problem statements and the semantic representation. Palmer's work is an attempt to be much more careful about the connections.

de Kleer's "Newton" program [71, 72] reasoned about kinematics problems. In particular, Newton could predict what would happen when a ball rolled down a ramp continued up another ramp. The reasoning strategy began by partitioning the possible outcomes into a set of qualitatively different spaces (such as those where the

first ramp was lower than the second, versus where it was higher), and then applied kinematic equations to each of the spaces separately. Sagredo doesn't do this sort of qualitative analysis of physical systems. All of the problems it can solve are ones where the problem statement and the introduction of the relevant quantities are sufficient to create a set of equations that is adequate for a solution, and all of the required solutions are the direct results of algebraic manipulation. My feeling is that such qualitative reasoning about physical systems is an important problem both for understanding how humans think about physics, and to make computers able to do so. But an important step is to understand the initial relations between the linguistic description of physical situations and the equations, and that is what I have investigated in this project. Later approaches to the issues involved in qualitative reasoning about physical systems are described in the papers in Bobrow [31], and Weld and de Kleer [300].

Another important area of investigation is that of "naive" or "commonsense" physics. de Kleer mentions that this is one aspect of his project, and Hayes [146, 147] sets the formalization of naive physics as an important research goal. Some specific attempts toward this goal are described in the papers in Hobbs [160].

To the degree that there is a difference between naive physics and qualitative physics, the former is an attempt to model human's understanding of the world, whether or not that understanding is correct according to "real" physics, while the latter is an attempt to formalize and implement non-numeric reasoning strategies about the physical world, whether or not they are the ones that people use. My approach is along another dimension. The naive physics that I make use of is that encoded or expressed linguistically. It turns out that this is a very small and simple subset of physics. As I argued in chapter 10, textbooks make ambivalent use of the rest of people's intuitions. Furthermore, it is not clear what form these intuitions take. Perhaps they are indeed expressed (or are expressible) in the first order logic notation advocated by Hayes. More likely, it seems to me, they take the form of bodily expectations and reflexes and specific intuitions based on one's experience in the physical world. These are also the ultimate basis of the understanding that lies behind language use.

A number of investigators have studied the psychology of physics and mathematics problem-solving. Chi, et al. [54] looked at how "novice" and "expert" problem solvers differ in their categorization of word problems. Their findings lend support to the idea that experts understand problem statements in terms of the more fundamental physical principles involved, while novices concentrate more on the surface form of the presentation. Sagredo is much more like a novice in this sense than an expert. Later work by Chi, et al. [53] investigated how different students use self-explanations to understand examples to solve problems.

Larkin, et al. [200] characterized and implemented two different "models of competence" of physics problem-solving. These programs accepted a model of the physics

problem as a set of propositions and differed in how they explored the space of alternatives. A later theory of word problem solving is presented in Briars and Larkin [40].

Hayes et al. [145] investigate how students identify the relevant aspects of a problem text. Another investigation by this group involves an account of meaning and representation in algebra problems in Hinsley, et al. [153]. Kintsch and Greeno [192] report on an investigation of the understanding and solving of word arithmetic problems. Sowder [281] looks at how children think about story problems. And Hall, et al. [136] explore the different "episodes" in the solving of algebra problems.

While these psychological investigations are interesting, I did not make much use of them in the development of Sagredo. The reason is first that Sagredo is not intended to be a psychological model of problem solving or language use. I am looking at an aspect of the problem that is "closer to the surface" than many of these cognitive approaches in that I concentrate on how the linguistic form of the presentation and problem texts can be used to relate the problem text to the presentation text.

11.2 Syntax and Semantics

I began this project by reading Huddleston's 470-page "introduction" to the grammar of English [166]. That book, and other descriptive grammars of English, for example Jespersen [172], Quirk, et al. [246], and Matthews [213] are similar in not imposing autonomy on any particular level, and not requiring that all explanations be completely precise and airtight. Details of grammatical structuring are presented along with the more or less systematic ways that such structures are related to their meanings and use.

I found such discussions useful because they strike a balance between two poles: on the one hand an attempt to formalize syntactic regularity independently of semantics and pragmatics, perhaps leaving those to be filled in later; and on the other hand the tendency to abstract completely away from language into a realm of logical propositions or some other "knowledge representation" medium. The descriptive linguistics texts I mentioned present language itself as a knowledge representation medium, and describe how it is can can be used as such.

There is nothing especially novel about such discussions. In fact many informal grammar books work this way. For example: Gordon [125] is a guide for writers, Huddleston [167], is an "outline" (but still 205-page) version of [166], Liles [206] is a text for high-school students, Azar [12] is a text for teaching English as a second language. The best example of the interactions between syntactic structuring and use, in my opinion, is the 'corandic' passage I quoted in the first chapter. It comes from Weaver [296], a guide for teaching high-school English. I also examined the presentation in a number of first year foreign language texts: French [90], Spanish [127], Swahili [241], and Classical Greek [255].

When I began thinking about formalizing the analysis I was doing of textbook passages, I chose to use a "unification-based" grammar, mostly because Shieber's book [272] was so clear an introduction. That book is a summary of work that began with Kay [186, 188], and has led to (among other things), the "Lexical Functional Grammar" theory [180, 178, 179]. While my grammar is certainly not orthodox LFG, I was strongly influenced by that theory and the approaches to its implementation.

As I discuss in chapters 5 and 6, the fundamental structure-building operation in unification based grammars (and in Sagredo) involves solving a set of unification equations for the minimal structure that satisfies the set of equations, This structure can include semantic and discourse-level information, along with the purely syntactic characterization of the sentence. Some of these issues are discussed in Shieber's book, and also by Fenstad, et al. [97] and Karttunen [184].

Halvorsen [137] was one of the first to describe how the feature structures of LFG could be given semantic interpretation, and Halvorsen and Kaplan [139] describe how the interpretation process can be done by defining "projections" among various domains, including the semantic domain. Fenstad, et al, [96], show how the Situation Theory [22] can be used as the domain for the semantic mapping from a unification-based representation of the syntax. A proposal for using projections to do translation is presented in [181]. Colban [58] shows how the semantics of prepositional phrases can be integrated into this approach. My own contribution to this trend involves adding the discourse to the set of projection domains.

Perhaps the strongest source of my intuition that this project could work was the "case grammar" work originating with Fillmore [99, 101, 104]. Another approach to some of the problems of case grammar is described by Anderson [8], this book is reviewed in [28]. Closely related to the notion of case frames are the ideas of "thematic" [39] or "theta" [247] roles.

What I found most important about all of this work is that the level of the representation of the case-frame relations in a clause is a sort of "halfway semantics." On the one hand the kinds of case-frames and the contents and relations among the participants are in general determined by fairly precise grammatical rules, and on the other hand, those frames can be interpreted as characterizing relations among the participants and some situations or events. My own "occurrence" model is essentially just case frames with some additional temporal structure added. Since the case frames can be interpreted as asserting some relation among their contents, they can be used, as I do, to do simple logical reasoning.

Deeper approaches to semantics tend to be concerned with the relations between predicate arguments and quantifiers. Kempson [189], and Barwise and Etchemendy [19], provide introductions to some of the issues. Perhaps the most influential formal work has been that of Montague [224, 225], which has influenced a large number of workers. Bach [14], Halvorsen [138] and Halvorsen and Ladusaw [140], provide useful

summaries of this approach. Thomason and Stalnaker [288] show how adverbs can be treated in a generally Montague-style approach. This work has been inspirational to me but not immediately useful because it focuses mostly on predicate-argument relations, of which I get fairly simple instances, and quantifiers, of which only the "all-subs" case really requires any attention.

A lot of work has gone into "situation semantics" as originally developed by Barwise and Perry [17, 18, 20, 21, 22, 23, 77]. That work has been organized around the idea that a proper semantic account for natural language will involve taking "situations" as the fundamental semantic type, as opposed to the model-theoretic approach of Montague. My "occurrences" were motivated by the situations of situation theory, however situation theory has focused more on how the various parameters of situations (in particular, the participants in the discourse, and its time and place) are used to characterize the meanings of sentences. For the most part these relations are quite simple in Sagredo, and are embedded in the operation of the interpreter.

More useful to me were investigations of the semantics of words. Some of the earliest work along these lines was done by Shank [258, 259] who characterizes the meanings of words, and from them, sentences, in terms of combinations of a small number of primitive concepts. My own approach is quite similar to this, except that I am even more restricted in the set of concepts and relations that I need to use because I know in advance what the relevant aspects of the meaning would be such that the physics could be learned and the problems solved.

A number of workers in linguistics have looked at similar approaches to characterizing meaning. For example Jackendoff's "conceptual semantics" also tries to use a set of primitive relations in deriving the meanings of words and sentences [169, 170, 171], as does Langacker in his "cognitive grammar" [199]. Fillmore investigated a similar approach to semantics [102, 107, 109], and his motivation also had to do with the realization that case-frames are useful for the syntactic characterization of sentences, and are also a simply-computed semantic representation.

Cruse [63] presents an introduction to lexical semantics from a linguistic perspective. Other discussions of this topic appear in Fillmore [103] and Gawron [117]. Hobbs, et al. [159] attempts to connect some of the work in commonsense physics with a representation for lexical semantics. Levin [204] collects linguistic information about the syntax and semantics of a large number of verbs. Siskind [278] describes a program that can learn the semantic entries for verbs based on "viewing" simple animated scenes and "hearing" sentences that describe what is going on. Dahlgren [65] presents a simple semantics for English words, and Higginbotham [150] discusses some of the general issues surrounding lexical semantics.

In thinking about the implementation of Sagredo, I made specific use of treatments of the verbs of motion and the representation of time and space. Amsler [7] presents a very useful summary of the properties of motion verbs. The short discussion at the end of chapter 4 was based mostly on this work. Davidson [67] discusses some of

the general issues surrounding the proper analysis of "action verbs." His argument is that they cannot be characterized as fixed-arity predicates because it is always possible to describe more participants. This work is consistent with the situation theory approach to semantics (though not with all attempts to implement the latter). My treatment of prepositions, in particular, was strongly influenced by Davidson's discussions. Herskovits [149] describes the linguistics of spatial prepositions, and Mellish [218] describes how spatial relations are expressed in descriptions.

A great deal of useful work has been done on representations of time and its relation to linguistic tense and aspect. Two of the most important early works are Reichenbach [248], who made some influential suggestions on the treatment of tense, and Vendler [292] whose topic was aspect. While Sagredo's utilization of tense and aspect is very limited, I found the discussions in these papers and their large number of descendants very useful in developing the occurrence model and figuring out how to connect it to the syntax.

The first step in a treatment of tense requires a formal model of time and temporal intervals and moments. A number of workers have presented models of "events" that differ, for example, in the length of the event (moments versus intervals), and in whether the event has a natural endpoint (so-called "progressive" versus "perfective") events and so forth. Vendler presents a version of this model that he argues is based on ideas of Aristotle. Bach [13] discusses one such hierarchy. Dowty [85, 86] describes a similar one.

Linguistic discussions of tense include the textbook by Comrie [61], and work by Taylor [287]. Linguistic discussions of aspect include a different textbook by Comrie [60]. Freed [113] studied the linguistics of verbs like "start" and "finish" that refer to or modify the aspectual structure of the clauses that they take as complements. Parsons [238] discusses the progressive aspect in English.

An important topic is the relation between the tenses and aspects of the sentences in a discourse or a narration, and the events that the sentences are about. Nerbonne [229] is specifically concerned with narration. Bartsch [16] considers the scopes of temporal expressions in discourse, Hinrichs [151] discusses "temporal anaphora" in discourse. Cooper [62] applies situation semantics to the analysis of tense in discourse. Lo Cascio and Vet [48] analyze temporal structure in sentences and in discourse, this work is reviewed by Hinrichs in [152]. Both Dowty [87] and Smith [280] present arguments wondering whether the proper treatment of aspect in discourse might be a matter of pragmatics as much as, or more than, semantics.

A number of computational approaches have been taken to the problem of representing and reasoning about time. One of the most influential has been that of Allen [4, 6] which breaks time into intervals that are ordered by a small number of relations, such as "overlap" and "meet." These structures can then be used to organize reasoning about temporal activities. McDermott [217] presents another temporal logic for reasoning about processes and plans. Gomez [121] describes a temporal model for

reasoning about biological systems.

Moens and Steedman [222, 223] describe a temporal ontology for natural language based on the work of Vendler and Dowty, and present a system for reasoning about temporal objects. One of the interesting aspects of this work is that they allow that a given expression can designate different classes of temporal objects and show how their system can determine the appropriate class.

Harper and Charniak [142] present a computational system for reasoning about time and tense in English. Dalyrmple [66] shows how tense and aspect expressions can be interpreted. Brent [37] investigates the treatment of causal and temporal connectives. Passoneau [239] describes a system for reasoning about situations and intervals. Hirschman [154] takes a computational approach to some of the time relations in narrative. And Karlin [182] implemented a system that could handle the verbal modifiers in the domain of cooking tasks.

11.3 Language in Action

When I began the close reading of textbooks I was struck by the feeling that "something was going on" in the textbooks, that the text in itself constituted an activity. Reading is one way to participate in that activity, but there seemed to be a sense that the textual resources could be described independently of any specific occasion of reading. The idea of discourse events was developed in order to talk about the places where the text seems to be doing something.

I was most influenced by the theory of "speech acts." This theory got its start with the discussion of "illocutionary" forces by Austin [11], and was systematized by Searle [264, 265, 266]. A very interesting application of speech act style analysis performed by Bowers [34], who looks at the language used in legislative texts. Katz [185] presents an alternative approach to the treatment of illocutionary force. Ross [254] argues that all sentences have a "performative" component expressed in the grammar that encodes what speech act theorists would call the illocutionary force. As I discuss below, this seems to confuse the linguistic form of an utterance with the uses to which it is put in a particular situation. I was also motivated by Wittgenstein's later philosophy [306] and his stress on "meaning as use." I don't think that speech act theory captures the full sense of Wittgenstein's intuitions about meaning as use, however it is a first attempt.

For the most part, speech act theory is concerned with the illocutionary force of whole sentences. (Although Searle does consider referential noun phrases in [265].) Another approach to the analysis of language, called "conversation analysis" looks closely at much finer details of the use of language. Fragments of sentences, gestures, and pauses can be seen to be crucial aspects of participation in linguistic activities. Heritage, in chapter 8 of [148], and Levinson, in chapter 6 of [205], provide very useful summaries of this approach. Sacks, et al. [257] and the papers in Schenkein [260]

present examples of original research. While my analyses of textbook passages are certainly not instances of the conversation analysis methodology, I was very influenced by their commitment to the fine-structure of linguistic activity. In many passages I was motivated to ask why in particular a particular word or syntactic structure was used.

A number of workers have looked at what might be called the "situated" use of language, where the significance of expressions is highly dependent on the situation of use. Obviously any use of "indexical" expressions (like "I" or "now") will have this property. Fillmore [100] discusses the linguistics of such cases. Clark and Wilkes-Gibbs [57] show how the use of referring expressions can be understood as a collaborative process. Nunberg [235] argues that the majority of referring expressions are comprehensible only against a wide range of potential interpretations. Chapman [49] shows how the details of the current situation can be a crucial resource for the interpretation and following of instructions. Sibun [275] uses the details of the ongoing project (describing room layouts, or family relationships) to organize text generation. Agre and Batali [1] examine how pairs of subject make use of linguistic and situational resources in determining how to perform a task while following a sheet of instructions. The importance of these investigations for my work has to do that Sagredo makes very particular use of what is going on in the text as it figures out how to respond to it. Much of the information that it has to use is obtained as a resource in the text.

There is a large literature on the linguistics of discourse. I found Levinson's [205] textbook very useful. Longacre [207], and Brown and Yule [42] also provide good overviews. Fillmore [106, 110] discusses some general issues in linguistics of discourse.

Grice [128, 129] describes a number of issues surrounding the ways that conversation works. Gordon and Lakoff [124] argue that conversation can only be understood as involving a set of implicit "conversational postulates." Schiffrin [261] describes how "discourse markers" are used to organize conversation. Fox and Thompson [112] use discourse considerations to analyze the syntactic structure of relative clauses. Jones [174] studies how the notion of "theme" in sentences is realized in English.

The issue of locating co-referrers for anaphoric expressions has received a lot of attention in the computational linguistics community. The implementation of Sagredo required only the simplest mechanisms to do this. For the most part, the correct co-referring expression for a gives anaphoric expression is the last referring expression whose referent can be unified with the given expression. In some sense, the computational approaches to the resolution of anaphora begin by considering when and how this heuristic doesn't work. Sidner [276] and Hobbs [155] present heuristics that are better than the simple-minded one. A promising approach to anaphora resolution is to consider the idea of the "topic" or "focus" of a discourse. If the topic and/or focus could be understood, this could be used to sharpen the search for referring expressions. Grosz [132], Reichman [249] and Sidner [277] investigate this

idea.

Hans Kamp [177] is developing a formal theory of discourse whose goals but not whose specific details are consistent with the Montague-inspired approaches. This approach and others are described in [271, 270, 308]. Karttunen [183] and Vestre [294] present formal accounts of the syntax and semantics of questions.

The papers in Brady and Berwick [36] present a number of computational systems for dealing with discourse phenomenon. Weber [297] describes how the use of tense in discourse can be understood computationally.

There are a number of different and more or less incompatible accounts of the large-scale structure of texts. Baker's *Practical Stylist* [15] is a popular account whose goal is to help writers. Irmscher [168] is a guide for teaching writing.

van Dijk [83, 81] discusses the discourse structure of texts. He is known in particular for pursuing the notion of "text grammars" [78], discussed also in Dresser [89]. The notion of using a grammar for characterizing the structure of texts has its roots in Propp's [245] account of the "morphology" of folktales. In Propp's analysis, folktales are constructed out of a small number of participants (e.g., hero, antagonist, victim) and situations (trial, rescue, etc.). Now while this account has some similarities with the notion of a grammar, it is not the same notion exactly because the elements of Propp's morphological analysis are not purely formal objects. The role of "hero," for example, is not completely defined in terms of relations and events in the story, but requires the hearer's pre-understanding of what a hero is and what a hero can be expected to do. The relations in the story might be described formally but they are formal relations between non-formal entities.

Whether this matters depends on what it is that the analyst is trying to explain. If the goal is, for example, to characterize the formal relations that the parts of a text can exhibit, then a formal grammar-like approach is probably useful. However such an approach can do little but provide a means for summarizing the possible relations. In the case of syntax this might be enough: there seems to be phenomenon in language for which an adequate account simply involves systematically compiling what are and what aren't legal expressions. But at the level of text, it seems useful to be able to explain what *happens* in or with the text, and the text grammar approach does not, in itself, provide any help with that. Of course a grammar approach could be grafted on to an account of language as activity, as to some degree Ross does with his "performative deletion" analysis of sentences, but it seems to me that at best this merely records grammatically phenomenon that have to be explained elsewhere.

Besides the work of van Dijk, a number of other studies have investigated story-grammar-like approaches to texts, for example Rumelhart's [256] "schema" for stories. Colby [59] makes use of a methodology similar to Propps's in his account of Eskimo folktales. Charniak's [50] investigation of children's stories focused more on the complexity of the commonsense reasoning needed to understand them.

Wilensky [303] criticizes story grammars for reasons similar to mine and argues

instead that an account of the structure of stories has to be based on an account of human action, specifically the notions of intentions and plans. He presents his own theory of "story points" in [302] which is based on an account of planning and understanding [301]. Actually van Dijk [79] apparently agrees that an account of the philosophy of human action is needed for the proper treatment of narrative so it is not clear how this is compatible with text grammars being "grammars." Other looks at the relation between action and narrative is found in Beaugrande and Colby [70], and Sidner and Grosz [133].

Speech act theory, on the other hand, is explicitly based on activity. The analysis of an utterance as a speech act is not a "grammatical" analysis in that it does not treat the act as a formal object, but as a full-fledged human action, with meaning, and with physical causes and effects. Searle and Vanderveken [268] present an "illocutionary logic" that is meant to provide part of the formal connection between the speech act theory and an account of action. I haven't seen a full-fledged attempt to use speech act theory in the analysis of texts, though Bowers' [34] treats legislative language in this framework, and Fish [111] analyzes a Shakespeare play within the speech act framework.

Another approach to the analysis of texts involves considering their "rhetorical structure." Again it is not always clear whether these structures are purely formal, or whether they have to be understood against a model of activity. In any case Horowitz [161], Hovy [163, 164, 165], Mann and Thompson [211, 212], and Scott and de Souza [262, 263] discuss approaches to the rhetorical structure of text. It is interesting to me that much of this work involves natural language generation, where the elements of the rhetorical structure are seen as satisfying various discourse goals. This relates back to the idea of language as activity.

A number of models of the reading process have been proposed, some based on psychological experiments. Clark [56] and Haviland and Clark [144] investigated the inferences that are involved in comprehending texts. van Dijk [80, 82, 84] and Kintsch and van Dijk [191] consider some of the strategies that readers use when confronting texts. Fillmore [105, 108] considers how the activity of the reader must be understood when accounting for discourse structure. The papers in [176] and [209] consider the cognitive processes in the comprehension of text. Kintsch [190] is about comprehending stories. Meyer looks at how the structure of prose affects reading and memory. Miyake [221] considers the understanding of stories to be a constructive process. Mandler [210] investigates the psychological reality of story structure. Hobbes [156, 157] considers the question of what makes a text coherent.

And finally, literary critics have developed tools for analyzing the structure of text. These tend to be more informal than those of linguists both in the sense of being less precise, and also in that they require for their deployment more of a sense of what the units are about. Accounts of narrative include that of Ricoeur [251], Rimmon [252], and Toolan [290].

A general discussions of the issues and methods of literary criticism is found in Eagleton [91]. In his chapter on political criticism he discusses the sort of investigation of the rhetorical aspects of textbooks that I undertake in chapter 10. Fish's collection of essays [111] is also useful in examining the moment-by-moment response of the reader to a text. Gass [116] focuses on language as a lived, bodily, experience. Culler [64] and Norris [231] provide accounts of some modern methods of literary criticism. Hobbs [158] tries to make computational sense of the analysis of literature. Burke's [46] discussion of "form" in literature is anything but "formal" in the modern sense, yet it seems to be accurate:

> Form in literature is an arousing and fulfillment of desires. A work has form in so far as one part of it leads a reader to anticipate another part, to be gratified by the sequence. [p. 124]

11.4 Expository Text

The specific domain of expository text has received a fair share of attention. Dee-Lucas and Larkin, for example performed experiments monitoring how novices read and assess the importance of different passages of scientific text [73, 75], and have investigated attentional strategies employed by readers of scientific texts [74]. Gomez and Segami investigated the recognition and classification of concepts in scientific texts [122], and Gomez presents a model of the comprehension of scientific text [120]. This model is the basis for a system which performs automatic acquisition and use of knowledge from biology textbooks [123]. Larkin and Simon [201] describe some of the ways that diagrams are used in expository text, and Funt [114] describes a program that makes use of diagrams to do certain kinds of reasoning. Roe [253] examines how details of the linguistic structure of scientific texts make some of them more difficult to understand than others. Goldman and Durán [119] describe a conceptual model of questioning answering from oceanography texts based on learner, task, and question characteristics.

Many of these approaches cited above are based on a model of an expository text as presenting a set of propositions. Britton and Black [41] develop this idea in detail and provide a number of analyses of texts in these terms. The approach I took with Sagredo is not exactly like this because, for one thing, the texts I look at provide laws and rules, not specific facts. Now perhaps the laws of physics could be represented as a set of propositions, and the problem statements could certainly be represented as sets of propositions. But this is counter to the spirit of my approach. In some sense a set of propositions are too "far" away from the linguistic form. The reader (or a program) has to first determine the syntactic form of the sentences, then their semantics, and from them collect a set of propositions. This seems like a lot of unnecessary work, especially when, as I have shown, a great deal of important

reasoning can be done in a representation that is much closer to the syntactic form of the sentences.

A number of researchers have investigated the notion of a "sublanguage" which is used for a particular purpose or in a particular domain. The idea is that by looking at such a restricted domain, the linguistic issues might be come more tractable. This is certainly true for the case of physics problems. Grishman and Kittredge [130] and Kittredge and Lehrberger [193] provide overview of some of the issues in the description and processing of sublanguages. Bonzi [33] looks specifically at the sublanguages associated with four scientific domains.

Hamburger [141] edited a special issue of the *International Journal of Expert Systems* devoted to the use of natural language in expert systems. Most of the papers (and indeed a great deal of work in applications of natural language) involves the use of natural language as a "front end" to reasoning in other formalisms. Manaster [208] discusses the "expertise" involved in knowing a natural language, and describes how this can be used in a system to learn the rules of a second language. Vorkar and Roach [295] describe a system based on a "semantic grammar" [47] that reads research paper abstracts. This system incorporates a frame system given by the researchers to classify papers about pharmacology.

I also found useful a number of studies on how people learn and reason about mathematics and physics. Haertel [135] describes high-school students' developing understanding of physics. Austern [10], Eisenbud [94] and Weinstock [298] discuss how the laws of classical mechanics ought to be taught. Davydov [69] investigates how children learn arithmetic. And Lave [202] discusses the gulf between the formal mathematics people learn in school and the practical calculations they perform in everyday life.

One of the things that struck me when I was reading all of the textbooks that I read was how similar they are. They all present the same material, of course, but many of them present it with the same examples and illustrations (some of them dating back to those in Newton and Galileo) and problems. Gould [126] discusses this phenomenon by tracing the lineage of a particular phrase in biological writing back through several generations of textbooks.

11.5 Implementation

As I mentioned above, Shieber's discussion of unification grammars [272], was one of my initial motivations. After I read that book, I decided to implement an interpreter for unification equations, to solidify my understanding of the issues. The present implementation of Sagredo is an descendent of that interpreter. Kay's paper on parsing unification grammars [187] and the "notes from the unification underground" [273, 274] were all useful as the implementation progressed.

Although it did not involve unification, some of my own earlier work [26, 27]

involved constructing structured objects by interpreting sequences of forms. The '<<' notation that I use for paths was based directly on that work, which was in turn based on the notation of the "constraints" languages of Sussman and Steele [282, 286]. I also made use of some of the ideas in the latter work in my own algebraic constraint system. While not usually thought of in the same context, the unification-based grammars share the same general idea of structured objects with "object oriented" programming systems described in [284, 32]. My own implementation of "methods" for use during parsing and processing of discourse events is close to the latter paradigm. The use of parse methods attached to specific classes of grammatical constituents results in a parser that is perhaps closest spirit to "augmented transition networks" [307].

Kaplan and Bresnan [180] outline an algorithm for interpreting unification equations. The one I describe in chapter 6 is similar, though I include various provisions for disjunction and parsing. Johnson [173] discusses the logic of feature-structures and describes and compares a number of algorithms that deal with such structures. Knight [194] reviews a number of papers on unification. Maxwell and Kaplan [214] and Eisele [93] describe algorithms for "disjunctive unification." The interpreter for Sagredo doesn't use disjunctive unification as such: disjunction is expressed and recorded in the control of the interpreter.

Aït-Kaci and Lincoln [3] describe a natural-language processing system that has a lot in common with mine. Specifically they also make use of a simple type-inheritance mechanism built on top of a unification-based formalism. Pereira and Warren's [240] system is sort of like an inverse to Sagredo: they use a logic programming language to implement a unification-based grammar. Pollack and Pereira [243] discuss its use in integrating syntactic, semantic, and pragmatic information. Delmonte [76] uses an LFG-based system to do "semantic parsing," which involves constructing semantic representations as projections from the syntax. Another approach to unification based semantic interpretation is presented by Moore [227].

McAllester and Givan [215] argue that there are close connections between natural language syntax and logical form, provided that the right representations are used for both. Burton's "semantic grammars" [47] are described as an "engineering technique" for constructing natural language systems. Like Sagredo, they make use of the fact that a lot of useful information can be determined more or less immediately from the syntactic relations in sentences. Haas [134] uses such simply-acquired information to use and apply knowledge from texts [134]. Mooney and DeJong [226] describe a system to acquire "schemata" for natural language processing.

Of natural language textbooks, I found Allen's text [5] most useful, especially in its sections on computing logical form. Winograd's book [305] was useful especially in its simplified overview of English grammar.

In many ways the inspiration for this work was Winograd's thesis [304], which presents the famous SHRDLU program. That introduction to that book argues that although language understanding requires solutions to a number of formidable problems, none of them has to be solved completely. Instead each solution can be heuristic,

and can be reliable only in a subset of the case it will likely face. It seems to me that in many ways the focus of natural language processing has been away from that idea, concentrating instead on the best possible solutions to individual problems. I think that in real language use the best possible solution is almost never necessary, or else is pretty much the only one available. A challenge for computational linguistics is to figure out how programs can participate in the same sorts of negotiation and repair that people perform when conversation goes awry.

Chapter 12

Conclusion

The main contributions of this work are:

Learning Physics My project is part of a long tradition in artificial intelligence wherein computers are programmed to solve word problems stated in natural language. What Sagredo does that is most novel is to read the presentation passages and thus acquire the knowledge that it then uses when doing problems.

Syntax & Semantics of Motion Since Sagredo's domain is kinematics, the text it reads deals with moving bodies. So it incorporates a model of occurrences that allows for the representation of the temporal relations between occurrences, sub-occurrence relations, and the various participants in an occurrence. This model is connected to the linguistic expression of those relations via lexical entries for verbs and prepositions and subordinate clauses, and rules for their combination.

Discourse Events For Sagredo, reading is an active process. Many of the grammatical constituents it encounters require a response, from recording a referring expression or finding a co-referent, to recording a new physics principle, or computing the value of a quantity. The response of the program to the text is organized by discourse events introduced by lexical items or grammatical rules.

Structural Deduction The reasoning that Sagredo has to do is very restricted. All reasoning involves annotation of the text, by instantiating types of objects and testing structural relations. In particular, this makes computational sense of the idea of "story subsumption" that is an important expository technique. The mechanism is based on solving sets of unification equations to annotate the input text structure.

This project emerged from the collapse of my investigations into "introspective" and "reflective" computational systems, based on the work of Jon Doyle [88] and Brian Smith [279]. My own summary of that work [24] and attempt to take it further [25]

was based on the idea that the activity of an agent could be organized by considering reasoning itself to be an activity, potentially under the reasoned control of the agent.

To make something like that work, I had to figure out how an agent could articulate its ongoing activity and the considerations that went into deciding what to do (and, sometimes, in deciding how to decide, and so forth). I found myself most bogged down in the issue of representational practice: it is easy enough in a simple "planning" program to make a plan be a program that can be executed. But what status does the plan have if the execution process is something about which reasoning can be done? Among other things, the plan becomes much like any other of the external resources that an agent can make use of in its ongoing participation in the world, and specifically, it begins to seem like a *text*, subject to all of the interpretations that texts are subject to. The work of Suchman [285] and Agre & Chapman [2] spoke directly to my impasse, and an account of Nietzsche's philosophy that treats the interpreting and creating of texts as central to human activity [228] convinced me that I should back off from the general consideration of rationality as such, and focus on how representations are used.

Even the issue of "using representations" is difficult to pin down as it seems to devolve into the question of what counts as a representation. I felt that the one area about which there could be no such question was natural language: if linguistic objects, sentences, or perhaps utterances, are not representations then it would seem that nothing is. So I began looking for discussions of how language is used and what sorts of systematicity can be achieved in discussions of the relations between the forms of linguistic objects and the activities that involve those objects.

I found payoff in the first place I looked: Huddleston's 483-page "introduction" to the grammar of English [166]. The book is an instance of fairly straightforward descriptive linguistics, attempting to connect regularities of grammar with the ways that the resultant structures can be used, whether that use is as a constituent in some other structure, or as an utterance in a discourse. There ought to be nothing surprising about this: as I mentioned in the last chapter, it is standard practice in grammar books for high-school students, or writers, or for learning foreign languages.

But it seemed to be missing from much of the theoretical linguistics I looked at. The latter seemed to concern itself mostly with parsing issues, or the semantics of quantifiers. (I know now that I was badly un-informed. Still the tables of contents of most texts in computational linguistics conform to my initial expectations.) And the other side of the story, the work in artificial intelligence on "knowledge representation" seemed disconnected from the linguistic means in which real knowledge is represented.

So I began an empirical investigation of knowledge representation. I felt that the best way to understand how knowledge representation works was go out and look at places where knowledge is actually represented. I chose to study physics textbooks because they contain convenient tests of understanding: the word-problems at the ends of the chapters. Also, an understanding of physics, and an understanding of how physics is presented, are prerequisites for computational tools to assist engineers.

I examined a number of physics books with the specific question in mind: How does the presentation of material in a technical textbook make it possible to solve problems based on that material? I began to understand that in many cases the answer was fairly simple. A presentation passage describe a class of situations by telling a very simple story. The occurrence described by the story is then annotated with equations among values of quantities, or with a claim that certain participants are identical, or with some elaboration of the occurrence that will occur whenever certain conditions obtain. A problem statement also begins by telling a simple story. The problem text then asks for the value of a quantity. For the problem to be solvable, one or more of the situations described by presentation passages must subsume the occurrence described by the story, such that the equations introduced by those occurrences suffice to determine the value of the quantity.

I wrote a program to illustrate these observations. The program interprets a text by building up annotations to the text's structure that represent the grammatical constituents in the text, and their semantic counterparts. Rules for recognizing and responding to discourse events complete the annotation and determine, in the presentation text passages, when a new physics definition or principle has been introduced. When new material is introduced, the system creates definitions of new structural types that are are then available for annotating subsequent presentation passages, for following their arguments, and for solving problems.

The program makes use of a simple semantic representation, corresponding closely to the level of the linguistic case frame, and fairly easily computable from the syntax of sentences. Furthermore, all of the reasoning with these representations, including the computational implementation of the subsumption relation between presentation and problem occurrences, involves simple structural annotations and tests. The need for real world and background knowledge is extremely limited, and mostly takes the form of assigning words to their appropriate grammatical categories.

But not all text works this way, and not even all of a physics text works this way. Perhaps only by trying to read physics texts as sets of logically valid arguments can one be as impressed as I am by the ways that they are not. The rhetoric that the books deploy at tender places in the arguments allow a peek through the latticework that the books are constructing, to see a bit of that process of construction and maintenance. The rules that I encoded into Sagredo are but the fossilized remnants of that process, and quite simplified ones at that.

Still, in facing the many many many details that were needed to make the program work, I rendered language some of its due respect.

Bibliography

[1] Philip E. Agre and John Batali. Collaborative instruction use in furniture assembly. Presented at the American Association for Artificial Intelligence (AAAI-90) Workshop on Complex systems, Ethnomethodology and Interaction Analysis, July 1990.

[2] Philip E. Agre and David Chapman. What are plans for? MIT Artificial Intelligence Memo 1050, September 1988.

[3] Hassan Aït-Kaci and Patrick Lincoln. LIFE: A natural language for natural language. Technical Report ACA-ST-074-88, MCC, Austin, TX, February 1988.

[4] James Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2), July 1984.

[5] James Allen. *Natural Language Understanding.* The Benjamin/Cummings Publishing Company, 1987.

[6] James F. Allen and Henry A. Kautz. A model of naive temporal reasoning. In [160], pages 251–268, 1985.

[7] Robert A. Amsler. The verbs of motion: A dictionary definition case study. Bell Communications Research, July 1986.

[8] John M. Anderson. *The Grammar of Case: Towards a Localist Theory*, volume 4 of *Cambridge Studies in Linguistics.* Cambridge University Press, 1971.

[9] Aristotle. *The Poetics.* The Loeb Classical Library, 1927. Translated by W. Hamilton Fyfe.

[10] N. Austern. Presentation of Newtonian mechanics. *American Journal of Physics*, 29(9):617–622, September 1961.

[11] J. L. Austin. *How to do Things with Words.* Clarendon Press, Oxford, 1962.

[12] Betty Schrampfer Azar. *Understanding & Using English Grammar.* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1981.

[13] Emmon Bach. The algebra of events. *Linguistics and Philosophy*, 9:5–16, 1986.

[14] Emmon Bach. *Informal Lectures on Formal Semantics.* SUNY Series in Linguistics. State University of New York Press, Albany, NY, 1989.

[15] Sheridan Baker. *The Practical Stylist.* Harper & Row, sixth edition, 1985.

[16] Renate Bartsch. Tenses, aspects, and their scopes in discourse. University of Amsterdam Institute for Language, Logic and Information Publication Series for Logic, Semantics and Philosophy of Language LP-88-07, 1988.

[17] Jon Barwise. Scenes and other situations. *Journal of Philosophy*, 78(7):369–397, 1981.

[18] Jon Barwise. On the circumstantial relation between meaning and content. In [92], pages 23–40, 1988.

[19] Jon Barwise and John Etchemendy. Model-theoretic semantics. In [244], pages 207–243, 1989.

[20] Jon Barwise and John Perry. Semantic innocence and uncompromising situations. In Peter A. French, Theodore E. Uehling, and Howard K. Wettstein, editors, *Midwest Studies in Philosophy, 6.* Minneapolis, 1981.

[21] Jon Barwise and John Perry. Situations and attitudes. *The Journal of Philosophy*, pages 668–691, 1981.

[22] Jon Barwise and John Perry. *Situations and Attitudes.* The MIT Press, Cambridge, MA, 1983.

[23] Jon Barwise and John Perry. Shifting situations and shaken attitudes: An interview with Barwise and Perry. *Linguistics and Philosophy*, 8:105–161, 1985.

[24] John Batali. Computational introspection. MIT Artificial Intelligence Laboratory Memo 701, 1983.

[25] John Batali. Reasoning about self-control. In Pattie Maes and Daniele Nardi, editors, *Meta-Level Architectures and Reflection*, pages 255–270. Elsevier Science Publishers B. V. (North-Holland), 1988.

[26] John Batali and Anne Hartheimer. The Design Procedure Language manual. MIT Artificial Intelligence Laboratory Memo 598, 1980.

[27] John Batali, Neil Mayle, Howard Shrobe, Gerald Sussman, and Daniel Weise. The DPL/Daedalus design environment. In *Proceedings of VLSI-81*, pages 183–192, Edinburgh, 1981. Academic Press, Inc. Also in Henry Fuchs (editor),

Selected Reprints on VLSI Technologies and Computer Graphics, pages 240–250, IEEE Computer Society Press, 1983.

[28] Laurie Bauer and Winfred Boagey. On 'The Grammar of Case'. *Linguistics and Philosophy*, 1(1):119–152, January 1977. Review of [8].

[29] Clarence E. Bennett. *Physics Problems and How to Solve Them.* College Outline Series. Barnes & Noble Books, second edition, 1972.

[30] Daniel G. Bobrow. Natural language input for a computer problem-solving system. In Marvin Minsky, editor, *Semantic Information Processing.* The MIT Press, Cambridge, MA, 1968.

[31] Daniel G. Bobrow, editor. *Qualitative Reasoning about Physical Systems.* Computational Models of Cognition and Perception. The MIT Press, Cambridge, MA, 1985.

[32] Daniel G. Bobrow, Linda G. DeMichiel, Richard P. Gabriel, Sonya E. Keene, Gregor Kiczales, and David A. Moon. Common LISP Object System specification, 1990. Chapter 28 of [283]; Also published as ANSI XJR13 Document 88-002R, June 1988.

[33] Susan Bonzi. Syntactic patterns in scientific sublanguages: A study of four disciplines. *Journal of the American Society for Information Science*, 4(2):121–131, 1990.

[34] Frederick Bowers. *Linguistic Aspects of Legislative Expression.* University of British Columbia Press, 1989.

[35] Ronald J. Brachman and Hector J. Levesque, editors. *Readings in Knowledge Representation.* Morgan Kaufmann Publishers, Inc., Los Altos, California, 1985.

[36] Michael Brady and Robert C. Berwick, editors. *Computational Models of Discourse.* The MIT Press, Cambridge, MA, 1983.

[37] Michael Brent. Causal/temporal connectives: Syntax and lexicon. MIT Artificial Intelligence Laboratory Memo 1122, September 1989.

[38] Joan Bresnan, editor. *The Mental Representation of Grammatical Relations.* The MIT Press, Cambridge, MA, 1982.

[39] Joan Bresnan and Jonni Kanerva. The thematic hierarchy and locative inversion in UG. Technical Report SSL-89-105, Xerox PARC Systems Sciences Laboratory, Palo Alto, CA, 1989.

[40] Diane J. Briars and Jill H. Larkin. An integrated model of skill in solving elementary word problems. *Cognition and Instruction*, 1(3):245–296, 1984.

[41] Bruce K. Britton and John B. Black, editors. *Understanding Expository Text.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1985.

[42] Gillian Brown and George Yule. *Discourse Analysis.* Cambridge Textbooks in Linguistics. Cambridge University Press, 1983.

[43] Frederick J. Bueche. *Theory and Problems of College Physics.* Schaum's Outline Series. McGraw-Hill Book Company, New York, eighth edition, 1988.

[44] William Charles Bulko. *Understanding Coreference in a System for Solving Physics Word Problems.* PhD thesis, The University of Texas at Austin, Artificial Intelligence Laboratory, 1989. UTA TR AI89-102/General University Microfilms International ADG89-20668.

[45] Alan Bundy, Lawrence Byrd, George Luger, Chris Mellish, and Martha Palmer. Solving mechanics problems using meta-level inference. In [219], pages 50–64, 1979.

[46] Kenneth Burke. *Counter-Statement.* University of California Press, 1931.

[47] Richard R. Burton. Semantic grammar: An engineering technique for constructing natural language understanding systems. Technical Report 3453, Bolt Beranek and Newman, Inc., 1976.

[48] Cincenzo Lo Cascio and Co Vet, editors. *Temporal Structure in Sentence and Discourse.* Foris Publications, Dordrecht, 1989. Groningen–Amsterdam Studies in Semantics (GRASS).

[49] David Chapman. Vision, instruction, and action. Technical Report 1204, MIT Artificial Intelligence Laboratory, Cambridge, MA, April 1990.

[50] Eugene Charnaik. Toward a model of children's story comprehension. Technical Report 266, MIT Artificial Intelligence Laboratory, Cambridge, MA, 1972.

[51] Eugene Charniak. CARPS: A program which solves calculus word problems. Technical Report 51, MIT Project MAC, 1968.

[52] Eugene Charniak. Computer solution of calculus word problems. In *Proceedings of IJCAI-69*, pages 303–316, 1969.

[53] Michelene T. H. Chi, Miriam Bassok, Matthew W. Lewis, Peter Reimann, and Robert Glaser. Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science*, 13:145–182, 1989.

[54] Michelene T. H. Chi, Paul J. Feltovich, and Robert Glaser. Categorization and representation of physics problems by experts and novices. *Cognitive Science*, 5:121–152, 1981.

[55] Noam Chomsky. *Syntactic Structures.* Mouton & Co., The Hague, 1957.

[56] Herbert H. Clark. Inferences in comprehension. In D. LaBerge and S. J. Samuels, editors, *Basic Processes in Reading.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1977.

[57] Herbert H. Clark and Deanna Wilkes-Gibbs. Referring as a collaborative process. *Cognition*, 22:1–39, 1986.

[58] Erik Colban and Jens Eric Fenstad. Situations and prepositional phrases. Technical Report COSMOS No. 1, Department of Mathematics, University of Oslo, March 1988. Also in *Proceedings of the Third European Conference of the Association of Computational Linguistics,* Copenhagen.

[59] B. N. Colby. A partial grammar of Eskimo folktales. *Americal Anthropologist*, 75:645–662, 1973.

[60] Bernard Comrie. *Aspect.* Cambridge Textbooks in Linguistics. Cambridge University Press, 1976.

[61] Bernard Comrie. *Tense.* Cambridge Textbooks in Linguistics. Cambridge University Press, 1985.

[62] Robin Cooper. Tense and discourse location in situation semantics. *Linguistics and Philosophy*, 9:17–36, 1986.

[63] D. A. Cruse. *Lexical Semantics.* Cambridge Textbooks in Linguistics. Cambridge University Press, 1986.

[64] Jonathan Culler. *Structuralist Poetics: Structuralism, Linguistics, and the Study of Literature.* Cornell University Press, 1975.

[65] Kathleen Dahlgren. *Naive Semantics for Natural Language Understanding.* The Kluwer International Series in Engineering and Computer Science. Kluwer Academic Publishers, Boston; Dordrecht; London, 1988.

[66] Mary Dalrymple. The interpretation of tense and aspect in English. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, pages 68–74, June 1988.

[67] Donald Davidson. The logical form of action sentences. In Nicholas Rescher, editor, *The Logic of Decision and Action.* University of Pittsburgh Press, 1967. Also in [68], chapter 6.

[68] Donald Davidson. *Essays on Actions and Events.* Oxford University Press, 1980.

[69] V. V. Davydov. The psychological characteristics of the formation of elementary mathematical operations in children. In Thomas P. Carpenter, James M. Moser, and Thomas A. Romberg, editors, *Addition and Subtraction: A Cognitive Perspective*, pages 224–238. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1982.

[70] Robert de Beaugrand and Benjamin N. Colby. Narrative models of action and interaction. *Cognitive Science*, 3:43–66, 1979.

[71] Johann de Kleer. Qualitative and quantitative knowledge in classical mechanics. Technical Report TR-352, M.I.T Artificial Intelligence Laboratory, 1975.

[72] Johann de Kleer. Multiple representations of knowledge in a mechanics problem-solver. In *Proceedings of IJCAI-77*, pages 299–304, 1977. Also in [300], pages 40–45.

[73] Diana Dee-Lucas and Jill H. Larkin. Novice strategies for processing scientific texts. *Discourse Processes*, 9:329–354, 1986.

[74] Diana Dee-Lucas and Jill H. Larkin. Attentional strategies for studying scientific texts. *Memory & Cognition*, 16(5):469–479, 1988.

[75] Diana Dee-Lucas and Jill H. Larkin. Novice rules for assessing importance in scientific texts. *Journal of Memory and Language*, 27:288–308, 1988.

[76] Rodolfo Delmonte. Semantic parsing with LFG and conceptual representations. *Computers and the Humanities*, 24:461–488, 1990.

[77] Keith Devlin. *Logic and Information, Volume 1: Infons and Situations.* Cambridge University Press, to appear, 1991.

[78] Teun A. van Dijk. *Some Aspects of Text Grammars.* Mouton & Co., The Hague, 1972.

[79] Teun A. van Dijk. Philosophy of action and theory of narrative. *Poetics*, 5:287–338, 1976.

[80] Teun A. van Dijk. Macro-structures: Knowledge frames and discourse comprehension. In [176], 1977.

[81] Teun A. van Dijk and Walter Kintsch. Cognitive psychology and discourse. In [89], 1977.

[82] Teun A. van Dijk and Walter Kintsch. *Strategies of Discourse Comprehension.* Academic Press Inc., 1983.

[83] Tuen A. van Dijk. *Text and Context: Explorations in the Semantics and Pragmatics of Discourse.* Longmans, London, 1977.

[84] Tuen A. van Dijk. Episodic models in discourse processing. In [162], pages 161–196, 1987.

[85] David R. Dowty. Toward a semantic analysis of verb aspect and the English 'imperfectve' progressive. *Linguistics and Philosophy*, 1:45–77, 1977.

[86] David R. Dowty. *Word Meaning and Montague Grammar: The Semantics of Verbs and Times in Generative Semantics and in Montague's PTQ*, volume 7 of *Synthese Language Library.* D. Reidel, Dordrecht, 1979.

[87] David R. Dowty. The effects of aspectual class on the temporal structure of discourse: Semantics or pragmatics? *Linguistics and Philosophy*, 9:37–61, 1986.

[88] Jon Doyle. A model for deliberation, action, and introspection. Technical Report AI TR 581, MIT Artificial Intelligence Laboratory, May 1981.

[89] W. Dressler, editor. *Current Trends in Textlinguistics.* de Gruyter, Berlin, 1977.

[90] Francis M. du Mont. *French Grammar.* Barnes & Noble Books, New York, second edition, 1950.

[91] Terry Eagleton. *Literary Theory: An Introduction.* University of Minnesota Press, 1983.

[92] Umberto Eco, Marco Santambrogio, and Patrizia Violi, editors. *Meaning and Mental Representations.* Advances in Semiotics. Indiana University Press, Bloomington, 1988.

[93] Andreas Eisele and Jochen Dörre. Disjunctive unification. In *Proceedings of the Workshop on Unification Formalisms — Syntax, Semantics and Implementation, Titisee*, Cambridge, MA, 1988. The MIT Press.

[94] Leonard Eisenbud. On the classical laws of motion. *American Journal of Physics*, 26(2):144–159, March 1958.

[95] Euclid. The Elements. In *Greek Mathematical Works*, volume 1, from Thales to Euclid. The Loeb Classical Library, 1939. Translated by Ivor Thomas.

[96] Jens Eric Fenstad, Per-Kristian Halvorsen, Tore Langholm, and Johan van Bentham. *Situations, Language and Logic.* D. Reidel, Dordrecht, 1987.

[97] Jens Eric Fenstad, Tore Langholm, and Espen Vestre. Representations and interpretations. Technical Report COSMOS No. 9, Department of Mathematics, University of Oslo, February 1989. Also in *Proceedings of the Workshop on Computational Linguistics and Formal Semantics*, Lugano, August-September, 1988.

[98] Richard P. Feynman, Robert B. Leighton, and Matthew Sands. *The Feynman Lectures on Physics: Mainly Mechanics, Radiation and Heat*, volume 1. Addison-Wesley Publishing Company, Reading, MA, 1963.

[99] Charles J. Fillmore. The case for case. In E. Bach and R. T. Harms, editors, *Universals in Linguistic Theory*, pages 1–210. Holt, Rinehart and Winston, 1968.

[100] Charles J. Fillmore. Santa Cruz lextures on deixis. Indiana University Linguistics Club, 1971.

[101] Charles J. Fillmore. Some problems for case gramar. In *22nd Annual Round Table. Linguistics: Developments of the Sixties—Viewpoints for the Seventies*, number 24 in Monograph Series on Languages and Linguistics, pages 35–56. Georgetown University Press, 1971.

[102] Charles J. Fillmore. An alternative to checklist theories of meaning. In *Proceedings of the First Annual Meeting of the Berkeley Linguistics Society*, pages 123–131, 1975.

[103] Charles J. Fillmore. Topics in lexical semantics. In Roger Cole, editor, *Current Issues in Linguistic Theory*, pages 76–138. Indiana University Press, 1976.

[104] Charles J. Fillmore. The case for case reopened. In Peter Cole and Jerrold Sadock, editors, *Syntax and Semantics Volume 8: Grammatical Relations*, pages 59–81. Academic Press Inc., 1977.

[105] Charles J. Fillmore. Ideal readers and real readers. In Deborah Tannen, editor, *Analyzing Discourse: Text and Talk*, Georgetown University Round Table on Language and Linguistics, pages 248–270. Georgetown University Press, 1981.

[106] Charles J. Fillmore. Pragmatics and the description of discourse. In Peter Cole, editor, *Radical Pragmatics*. Academic Press Inc., New York, 1981.

[107] Charles J. Fillmore. Frame semantics. In The Linguistic Society of Korea, editor, *Linguistics in the Morning Calm*. Hanshin Publishing Co., Seoul, 1982.

[108] Charles J. Fillmore. Monitoring the reading process. In The Linguistic Society of Korea, editor, *Linguistics in the Morning Calm*, pages 329–348. Hanshin Publishing Co., Seoul, 1982.

[109] Charles J. Fillmore. Frames and the semantics of understanding. *Quaderni di Semantica*, 6(2), December 1985.

[110] Charles J. Fillmore. Linguistics as a tool for discourse analysis. In *Handbook of Discourse Analysis, Vol. 1: Disciplines of Discourse.* Academic Press Inc., 1985.

[111] Stanley Fish. *Is There a Text in this Class? The Authority of Interpretive Communities.* Harvard University Press, Cambridge, MA, 1980.

[112] Barbara A. Fox and Sandra A. Thompson. A discourse explanation of the grammar of relative clauses in English conversation. *Language*, 66(2):297–616, 1990.

[113] Alice F. Freed. *The Semantics of English Aspectual Complementation*, volume 8 of *Synthese Language Library.* D. Reidel, Dordrecht, 1979.

[114] Brian V. Funt. Problem-solving with diagrammatic representations. *Artificial Intelligence*, 13(3):201–230, 1980. Also in [35], pages 441–456.

[115] Galileo Galilei. *Dialogues Concerning Two New Sciences.* Macmillan & Co., London, 1638/1914. Translation by Henry Crew and Alfonso de Salvio.

[116] William H. Gass. *Habitations of the Word.* A Touchstone Book. Simon & Schuster, Inc., New York, 1985.

[117] Jean Mark Gawron. *Lexical Representations and the Semantics of Complementation.* PhD thesis, The University of California at Berkeley, 1983. Department of Linguistics.

[118] Douglas C. Giancoli. *Physics.* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1980.

[119] Susan R. Goldman and Richard P. Durán. Answering questions from oceanography texts: Learner, task, and text characteristics. *Discourse Processes*, 11:373–412, 1988.

[120] Fernando Gomez. A model of comprehension of elementary scientific texts. In *Proceedings of the Workshop on Theoretical Approaches to Natural Language Understanding*, Halifax, Nova Scotia, May 1985.

[121] Fernando Gomez. Representing biological systems as temporal event hierarchies. Unpublished draft, January 1990.

[122] Fernando Gomez and Carlos Segami. The recognition and classification of concepts in understanding scientific texts. *Journal of Experimental and Theoretical Artificial Intelligence*, 1:51–77, 1989.

[123] Fernando Gomez and Carlos Segami. Knowledge acquisition from natural language for expert systems based on classification problem-solving methods. *Knowledge Acquisition*, 2:107–128, 1990.

[124] David Gordon and George Lakoff. Conversational postulates. In *Papers from the Seventh Regional Meeting of the Chicago Linguistic Society*, Chicago, 1971.

[125] Karen Elizabeth Gordon. *The Transitive Vampire: A Handbook of Grammar for the Innocent, the Eager, and the Doomed.* Times Books, Random House, Inc., 1984.

[126] Stephen Jay Gould. The case of the creeping fox terrier clone. *Natural History*, pages 16–24, January 1988.

[127] Eric V. Greenfield. *Spanish Grammar.* Barnes & Noble Books, New York, 1942.

[128] H. Paul Grice. Utterer's meaning, sentence-meaning, and word-meaning. *Foundations of Language*, 4:1–18, 1968.

[129] H. Paul Grice. Logic and conversation. In Peter Cole and Jerry L. Morgan, editors, *Syntax and Semantics: Speech Acts*, volume 3. Academic Press Inc., 1975.

[130] Ralph Grishman and Richard Kittredge, editors. *Analyzing Language in Restricted Domains: Sublanguage Description and Processing.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1986.

[131] Alan G. Gross. *The Rhetoric of Science.* Harvard University Press, 1990.

[132] Barbara Grosz. The representation and use of focus in a system for understanding dialogs. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 67–76, Cambridge, Massachusetts, 1977.

[133] Barbara Grosz and Candace Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204, 1986.

[134] N. Haas and G. Hendrix. An approach to acquiring and applying knowledge. In *Proceedings of the First National Conference on Artificial Intelligence*, pages 235–239, Los Altos, California, 1980. Morgan Kaufmann Publishers, Inc.

[135] Hermann Haertel. Learning and understanding concepts in physics. Technical report, Institute for Research on Learning, Palo Alto, CA, 1989.

[136] Rogers Hall, Dennis Kibler, Etienne Wenger, and Chris Truxaw. Exploring the episodic structure of algebra story problem solving. Technical Report 86-24, Irvine Computational Intelligence Project, University of California at Irvine,

Department of Information and Computer Science, November 1986. Revised December, 1988.

[137] Per-Kristian Halvorsen. Semantics for lexical-functional grammars. *Linguistic Inquiry*, 14(3):567–613, 1983.

[138] Per-Kristian Halvorsen. Natural language understanding and Montague grammar. *Computational Intelligence*, 2:54–62, 1986.

[139] Per-Kristian Halvorsen and Ronald M. Kaplan. Projections and semantic description in lexical-functional grammar. In *Proceedings of the International Conference on Fifth Generation Computer Systems, FGGS-88*, pages 1116–1122, Tokyo, Japan, November 1988.

[140] Per-Kristian Halvorsen and William A. Ladusaw. Montague's 'Universal Grammar': An introduction for the linguist. *Linguistics and Philosophy*, 3:185–223, 1979.

[141] Henry Hamburger. Natural language and expert systems. *International Journal of Expert Systems*, 1(4):v–vii, 1988.

[142] Mary P. Harper and Eugene Charniak. Time and tense in English. In *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*, 1986.

[143] Roy Harris. *The Language Machine.* Cornell University Press, 1987.

[144] S. E. Haviland and Herbert H. Clark. What's new? acquiring new information as a process in comprehension. *Journal of Verbal Learning and Verbal Behavior*, 13:512–521, 1974.

[145] J. R. Hayes, D. A. Waterman, and C. S. Robinson. Identifying the relevant aspects of a problem text. *Cognitive Science*, 1(3):297–313, 1977.

[146] Patrick J. Hayes. The naive physics manifesto. In [219], 1979.

[147] Patrick J. Hayes. The second naive physics manifesto. Technical Report URCS–10, University of Rochester, October 1983. In [160], pages 1–36; And in [35], pages 467–485.

[148] John Heritage. *Garfinkel and Ethnomethodology.* Polity Press, Cambridge, England, 1984.

[149] Annette Herskovits. *Language and Spatial Cognition: an Interdisciplinary Study of the Prepositions in English.* Studies in Natural Language Processing. Cambridge University Press, 1986.

[150] James Higginbotham. Elucidations of meaning. MIT Center for Cognitive Science Lexicon Project Working Paper 19, December 1987.

[151] Erhard Hinrichs. Temporal anaphora in discourses of English. *Linguistics and Philosophy*, 9:63–82, 1986.

[152] Erhard Hinrichs. Review of Lo Cascio and Vet (1989). *Linguistics and Philosophy*, 12:243–267, 1989. Review of [48].

[153] Dan A. Hinsley, John R. Hayes, and Herbert A. Simon. From words to equations: Meaning and representation in algebra word problems. In [176], pages 89–108, 1977.

[154] Lynette Hirschman and Guy Story. Representing implicit and explicit time relations in narrative. In *Proceedings of IJCAI-81*, pages 289–295, 1981.

[155] Jerry R. Hobbs. Resolving pronoun references. *Lingua*, 44:331–338, 1978.

[156] Jerry R. Hobbs. Coherence and coreference. *Cognitive Science*, 3(1):67–90, 1982.

[157] Jerry R. Hobbs. On the coherence and structure of discourse. Technical Report CSLI-85-37, Center for the Study of Language and Information, Stanford University, October 1985.

[158] Jerry R. Hobbs. *Literature and Cognition.* C.S.L.I.. Lecture Notes, Number 21. Center for the Study of Language and Information, Stanford University, 1990.

[159] Jerry R. Hobbs, William Croft, Todd Davies, Douglas Edwards, and Kenneth Laws. Commonsense metaphysics and lexical semantics. *Computational Linguistics*, 13(3–4):241–250, July–December 1987.

[160] Jerry R. Hobbs and Robert C. Moore, editors. *Formal Theories of the Commonsense World.* Ablex Publishing Corp., 1985.

[161] Rosalind Horowitz. Rhetorical structure in discourse processing. In [162], pages 117–160, 1987.

[162] Rosalind Horowitz and S. Jay Samuels, editors. *Comprehending Oral and Written Language: Critical Contrasts for Literacy and Schooling.* Academic Press, Inc., San Diego, CA, 1987.

[163] Eduard H. Hovy. *Generating Natural Language Under Pragmatic Constraints.* PhD thesis, Yale University Department of Computer Science, March 1987. YALEU/CSD/RR #521.

[164] Eduard H. Hovy. Planning coherent multisentential text. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, pages 163–169, June 1988.

[165] Eduard H. Hovy. Parsimonious and profligate approaches to the question of discourse structure relations, February 1990.

[166] Rodney Huddleston. *An Introduction to the Grammar of English.* Cambridge Textbooks in Linguistics. Cambridge University Press, 1984.

[167] Rodney Huddleston. *English Grammar: an Outline.* Cambridge University Press, 1988.

[168] William F. Irmscher. *Teaching Expository Writing.* Holt, Rinehard and Winston, 1979.

[169] Ray Jackendoff. Toward an explanatory semantic representation. *Linguistic Inquiry*, 7(1):89–150, Winter 1976.

[170] Ray Jackendoff. *Semantics and Cognition.* The MIT Press, Cambridge, MA, 1983.

[171] Ray Jackendoff. Conceptual semantics. In [92], pages 81–98, 1988.

[172] Otto Jespersen. *The Philosophy of Grammar.* The Norton Library, 1924.

[173] Mark Johnson. *Attribute-Value Logic and the Theory of Grammar.* C.S.L.I.. Lecture Notes, Number 16. Center for the Study of Language and Information, Stanford University, 1988.

[174] Linda Kay Jones. *Theme in English Expository Discourse.* Edward Sapir Monograph Series in language, Culture, and Cognition. Jupiter Press, 1977. Also in: *Forum Linguisticum,* 1:3, April 1977.

[175] Kathleen Jordan and Michael Lynch. The sociology of a genetic engineering technique: Ritual and rationality in the performance of a 'plasmid prep'. Paper presented at the 1989 Summer Conference of the International Society for the History, Philosophy, and Social Studies of Biology, held in London, Ontario, 1989.

[176] Marcel Adam Just and Patricia A. Carpenter, editors. *Cognitive Processes in Comprehension.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1977.

[177] Hans Kamp. A theory of truth and semantic representation. In Jeroen Groenendijk, Theo M. V. Janssen, and Martin Stokhof, editors, *Truth, Interpretation and Information: Selected Papers from the Third Amsterdam Colloquium.* Foris

Publications, Dordrecht, 1984. Groningen–Amsterdam Studies in Semantics (GRASS).

[178] Ronald M. Kaplan. Three seductions of computational psycholinguistics. In P. Whitelock, M. M. Wood, H. L. Somers, R. Johnson, and P. Bennett, editors, *Linguistic Theory and Computer Applications.* Academic Press, London, 1987.

[179] Ronald M. Kaplan. The formal architecture of Lexical-Functional Grammar. In *Proceedings of ROCLING II*, 1989.

[180] Ronald M. Kaplan and Joan Bresnan. Lexical-Functional Grammar: A formal system for grammatical representation. In [38], pages 173–281, 1982.

[181] Ronald M. Kaplan, Klaus Netter, Jurgen Wedekind, and Annie Zaenen. Translation by structural correspondences. In *Proceedings of the Fourth Conference of the European Chapter of the Association for Computational Linguistics*, pages 272–281, University of Manchester, April 1989.

[182] Robin F. Karlin. Defining the semantics of verbal modifiers in the domain of cooking tasks. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, pages 61–67, June 1988.

[183] Lauri Karttunen. The syntax and semantics of questions. *Linguistics and Philosophy*, 1:3–44, 1977.

[184] Lauri Karttunen. Features and values. In *Proceedings of the 10th International Conference on Computational Linguistics*, pages 28–33, Stanford, CA, 1984.

[185] Jerrold J. Katz. *Propositional Structure and Illocutionary Force.* The Language & Thought Series. Thomas Y. Crowell, New York, 1977.

[186] Martin Kay. Functional grammar. In *Proceedings of the Fifth Annual Meeting of the Berkeley Linguistics Society*, 1979.

[187] Martin Kay. Parsing in functional unification grammar. In David R. Dowty, Lauri Karttunen, and A. Zwicky, editors, *Natural Language Parsing*, pages 251–278. Cambridge University Press, 1982.

[188] Martin Kay. Functional unification grammar: A formalism for machine translation. In *Proceedings of Coling 1984*, pages 75–78, Stanford, CA, 1984.

[189] Ruth M. Kempson. *Semantic Theory.* Cambridge Textbooks in Linguistics. Cambridge University Press, 1977.

[190] Walter Kintsch. On comprehending stories. In [176], 1977.

[191] Walter Kintsch and Teun A. van Dijk. Toward a model of text comprehension and production. *Psychological Review*, 85:363–380, 1978.

[192] Walter Kintsch and James G. Greeno. Understanding and solving word arithmetic problems. *Psychological Review*, 92(1):109–129, 1985.

[193] R. Kittredge and J. Lehrberger, editors. *Sublanguage: Studies of Language in Restricted Semantic Domains.* de Gruyter, Berlin, 1982.

[194] Kevin Knight. Unification: A multidisciplinary survey. *ACM Computing Surveys*, 21(1):93–124, March 1989.

[195] Hyung Joon Kook. *A Model-Based Representational Framework for Expert Physics Problem Solving.* PhD thesis, The University of Texas at Austin, Artificial Intelligence Laboratory, May 1989. UTA TR AI89-103.

[196] Julien Kupiec. Augmenting a hidden markov model for phrase-dependent word tagging. In *Proceedings of the DARPA Speech and Natural Language Workshop*, Cape Cod, MA, October 1989.

[197] George Lakoff and Mark Johnson. *Metaphors We Live By.* The University of Chicago Press, 1980.

[198] Leslie Lamport. LaTeX*: A Document Preparation System.* Addison-Wesley Publishing Company, Reading, MA, 1986.

[199] Ronald W. Langacker. An introduction to cognitive grammar. *Cognitive Science*, 10:1–40, 1986.

[200] Jill H. Larkin, John McDermott, Dorothea P. Simon, and Herbert A. Simon. Models of competence in solving physics problems. *Cognitive Science*, 4:317–345, 1980.

[201] Jill H. Larkin and Herbert A. Simon. Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11:65–100, 1987.

[202] Jean Lave. *Cognition in Practice: Mind, Mathematics and Culture in Everyday Life.* Cambridge University Press, 1988.

[203] Jean Lave and Etienne Wenger. Situated learning: Legitimate peripheral participation. Technical Report 89-0013, Institute for Research on Learning, Palo Alto, CA, 1989.

[204] Beth Levin. Towards a lexical organization of English verbs. Unpublished draft, 1989. Department of Linguistics, Northwestern University.

[205] Stephen C. Levinson. *Pragmatics.* Cambridge Textbooks in Linguistics. Cambridge University Press, 1983.

[206] Bruce L. Liles. *A Basic Grammar of Modern English.* Prentice-Hall, Inc., Englewood Cliffs, NJ, second edition, 1987.

[207] Robert E. Longacre. *The Grammar of Discourse.* Topics in Language and Linguistics. Plenum Press, New York, 1983.

[208] Alexis Manaster-Ramer and Robert K. Lindsay. Linguistic knowledge as expertise. *International Journal of Expert Systems*, 1(4):329–343, 1988.

[209] Heinz Mandl, Nancy L. Stein, and Tom Trabasso, editors. *Learning and Comprehension of Text.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1984.

[210] Jean Mandler. On the psychological reality of story structure. *Discourse Processes*, 10:1–29, 1987.

[211] William C. Mann and Sandra A. Thompson. Rhetorical structure theory: Description and construction of text structures. In G. Kempen, editor, *Natural Language Generation: New Results in Artificial Intelligence, Psychology and Linguistics*, pages 279–300. Kluwer Academic Publishers, Dordrecht; Boston, MA, 1986.

[212] William C. Mann and Sandra A. Thompson. Rhetorical structure theory: A theory of text organization. Technical Report ISI/RS-87-190, USC/Information Sciences Institute, June 1987. Also in [242].

[213] P. H. Matthews. *Syntax.* Cambridge Textbooks in Linguistics. Cambridge University Press, 1981.

[214] John T. Maxwell, III and Ronald M. Kaplan. An overview of disjunctive constraint satisfaction. In *Proceeds of the International Workshop on Parsing Technologies*, pages 18–27, Pittsburgh, PA, 1989.

[215] David McAllester and Robert Givan. Natural language syntax and first order inference. MIT Artificial Intelligence Laboratory Memo 1176, October 1989.

[216] David A. McAllester. An outlook on truth maintenance. MIT Artificial Intelligence Laboratory Memo 551, 1980.

[217] Drew V. McDermott. A temporal logic for reasoning about processes and plans. *Cognitive Science*, 6:101–155, 1982.

[218] C. S. Mellish. *Computer Interpretation of Natural Language Descriptions.* Series in Artificial Intelligence. Ellis Horwood, Ltd., 1985.

[219] Donald Michie, editor. *Expert Systems in the Micro-electronic Age.* Edinburgh University Press, 1979.

[220] Tom Mitchell, Richard M. Keller, and Smadar T. Kedar-Cabelli. Explanation-based generalization: A unifying view. Technical Report ML-TR-2, Rutgers University, Department of Computer Science, Laboratory for Computer Science Research, 1985.

[221] Naomi Miyake. Constructive interaction and the iterative process of understanding. *Cognitive Science*, 10:151–177, 1986.

[222] Marc Moens and Mark Steedman. Temporal ontology in natural language. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, Stanford, CA, July 1987.

[223] Marc Moens and Mark Steedman. Temporal ontology and temporal reference. *Computational Linguistics*, 14(2), 1988.

[224] Richard Montague. Universal grammar. *Theoria*, 36, part 3:373–398, 1970.

[225] Richard Montague. *Formal Philosophy.* Yale University Press, New Haven, 1974. Richard Thomason, editor.

[226] Raymond Mooney and Gerald DeJong. Learning schemata for natural language processing. In *Proceedings of IJCAI*, 1985.

[227] Robert C. Moore. Unification-based semantic interpretation. In *Proceedings of the ACL conference*, 1989.

[228] Alexander Nehamas. *Nietzsche: Life as Literature.* Harvard University Press, Cambridge, MA, 1985.

[229] John Nerbonne. Reference time and time in narration. *Linguistics and Philosophy*, 9, 1986.

[230] Isaac Newton. *Philosophiae Naturalis Principia Mathematica*, 1686. Published by the University of California Press as *Sir Isaac Newton's Mathematical Principles of Natural Philosophy and His System of the World*, translated by Florian Cajori, 1934.

[231] Christopher Norris. *Deconstruction: Theory and Practice.* New Accents. Methuen & Co., Ltd., 1982.

[232] Gordon S. Novak, Jr. Computer understanding of physics problems stated in natural language. Technical Report NL-30, Computer Science Department, The University of Texas at Austin, 1976. Also in: *American Journal of Computational Linguistics,* Microfiche 53, 1976.

[233] Gordon S. Novak, Jr. Representation of knowledge in a program for solving physics problems. In *Proceedings of IJCAI-79*, pages 286–291, 1979.

[234] Gordon S. Novak, Jr. and William C. Bulko. Understanding natural language with diagrams. In *Proceedings of AAAI-90*, pages 465–470, 1990.

[235] Geoffrey Nunberg. The pragmatics of reference. Indiana University Linguistics Club, 1978.

[236] Julian Orr. Sharing knowledge, celebrating identity: War stories and community memory among service technicians. In David Middleton and Derek Edwards, editors, *Collective Remembering.* Sage, London, 1990.

[237] Martha Palmer. *Semantic Processing for Finite Domains.* Cambridge University Press, 1990.

[238] Terrence Parsons. The progressive in English: Events, states and processes. *Linguistics and Philosophy*, 12:213–241, 1989.

[239] Rebecca Passoneau. Situations and intervals. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, Stanford, CA, July 1987.

[240] Fernando C. N.. Pereira and David H. D. Warren. Definite clause grammars for natural language analysis – a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence*, 13:231–278, 1980.

[241] D. V. Perrott. *Teach Yourself Swahili.* Teach Yourself Books. Random House Inc., New York, second edition, 1957.

[242] Livia Polanyi, editor. *The Structure of Discourse.* Ablex Publishing Corp., Norwood, N.J., 1987.

[243] Martha E. Pollack and Fernando C. N. Pereira. An integrated framework for semantic and pragmatic interpretation. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, pages 75–86, June 1988.

[244] Michael I. Posner, editor. *Foundations of Cognitive Science.* A Bradford Book. The MIT Press, Cambridge, MA, 1989.

[245] Vladimir Propp. *Morphology of the Folktale.* University of Texas Press, Austin, second edition, 1968.

[246] Randolph Quirk, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik. *A Comprehensive Grammar of the English Language.* Longman Inc., New York, 1985.

[247] Malka Rappaport and Beth Levin. What to do with theta-roles. MIT Center for Cognitive Science Lexicon Project Working Paper 11, October 1986.

[248] Hans Reichenbach. *Elements of Symbolic Logic.* Macmillan & Co., London, 1947.

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

[250] Robert Resnick and David Halliday. *Physics, Part One.* John Wiley & Sons, Inc., New York, third edition, 1977.

[251] Paul Ricoeur. Narrative time. *Critical Inquiry*, pages 169–190, Autumn 1980.

[252] Shlomith Rimmon-Kenan. *Narrative Fiction: Contemporary Poetics.* New Accents. Methuen & Co., Ltd., 1983.

[253] Peter Roe. *Scientific Text: Selections from the Linguistic Evidence Presented in a Study of Difficulty in Science Text-Books.* Discourse Analysis Monographs Number 4. English Language Research, 1977.

[254] J. R. Ross. On declarative sentences. In R. A. Jacobs and P. S. Rosenbaum, editors, *Readings in English Transformational Grammar.* Ginn & Co., Waltham, MA, 1970.

[255] Carl A. P. Ruck. *Ancient Greek: A New Approach.* The MIT Press, Cambridge, MA, second edition, 1979.

[256] D. E. Rumelhart. Notes on a schema for stories. In Daniel G. Bobrow and A. Collins, editors, *Representation and Understanding*, pages 211–236. Academic Press, 1975.

[257] Harvey Sacks, Emmanuel A. Schegloff, and Gail Jefferson. A simplest systematics for the organization of turn taking for conversation. In [260], pages 7–58, 1978.

[258] Roger C. Schank. The fourteen primitive actions and their inferences. Stanford Artificial Intelligence Laboratory Memo AIM-183, March 1973.

[259] Roger C. Schank. *Conceptual Information Processing.* Fundamental Studies in Computer Science, Volume 3. North-Holland Publishing Company, 1975.

[260] Jim Schenkein, editor. *Studies in the Organization of Conversational Interaction.* Language, Thought, and Culture: Advances in the Study of Cognition. Academic Press, 1978.

[261] Deborah Schiffrin. *Discourse Markers.* Number 5 in Studies in Interactional Sociolinguistics. Cambridge University Press, 1987.

[262] Donia R. Scott and Clarisse Sieckenius de Souza. Conciliatory planning for extended descriptive texts. In *Proceedings of the VI Simpósio Brasileiro de Intelligênce Artificial*, Rio De Janeiro, November 1989.

[263] Donia R. Scott and Clarisse Sieckenius de Souza. Getting the message across in RST-based text generation. In Robert Dale, Chris Mellish, and Michael Zock, editors, *Current Research in Natural Language Generation.* Academic Press, 1990.

[264] John R. Searle. What is a speech act? In M. Black, editor, *Philosophy on America*, pages 221–239. Allen & Unwin and Cornel University Press, 1965.

[265] John R. Searle. *Speech Acts: An Essay in the Philosophy of Language.* Cambridge University Press, 1969.

[266] John R. Searle. A taxonomy of illocutionary acts. In Keith Gunderson, editor, *Language, Mind and Knowledge*, Minnesota Studies in the Philosophy of Science. University of Minnesota Press, 1975. Also in [267] pages 1–57.

[267] John R. Searle. *Expression and Meaning: Studies in the Theory of Speech Acts.* Cambridge University Press, 1979.

[268] John R. Searle and Daniel Vanderveken. *Foundations of Illocutionary Logic.* Cambridge University Press, 1985.

[269] Francis W. Sears, Mark W. Zemansky, and Hugh D. Young. *University Physics.* Addison-Wesley Publishing Company, Reading, MA, fifth edition, 1976.

[270] Helle Frisak Sem. Discourse in situation schema theory. Technical Report COSMOS No. 10, Department of Mathematics, University of Oslo, March 1988. Also in the proceedings of *The Nordic Conference on Text Comprehension in Man and Machine*, Sigtuna, October, 1988.

[271] Helle Frisak Sem. Discourse representation theory, situation schemata, and situation semantics, a comparison. Technical Report COSMOS No. 2, Department of Mathematics, University of Oslo, March 1988. Also in the proceedings of the 10th Scandinavian Conference of Linguisitics.

[272] Stuart M. Shieber. *An Introduction to Unification-Based Approaches to Grammar.* C.S.L.I. Lecture Notes, Number 4. Center for the Study of Language and Information, Stanford University, 1986.

[273] Stuart M. Shieber, Lauri Karttunen, and Fernando C. N. Pereira. Notes from the unification underground: A compilation of papers on unification-based grammar formalisms. SRI International Technical Note 327, Menlo Park, CA, June 1984.

[274] Stuart M. Shieber, Lauri Karttunen, and Fernando C. N. Pereira. More notes from the unification underground: A second compilation of papers on unification-based grammar formalisms. SRI International Technical Note 361, Menlo Park, CA, August 1985.

[275] Penelope Sibun. *Locally Organized Text Generation.* PhD thesis, Computer Science Department, University of Massachusetts at Amherst, 1991.

[276] Candace Sidner. Towards a computational theory of definite anaphora comprehension in English discourse. Technical Report 537, MIT Artificial Intelligence Laboratory, Cambridge, MA, June 1979.

[277] Candace Sidner. Focusing in the comprehension of definite anaphora. In [36], pages 267–330, 1983.

[278] Jeffrey Mark Siskind. Acquiring core meanings of words, represented as Jackendoff-style conceptual structures, from correlated streams of linguistic and non-linguistic input. Unpublished draft, December 1989.

[279] Brian C. Smith. Reflection and semantics in a procedural language. Technical Report 272, MIT Artificial Intelligence Laboratory, Cambridge, MA, 1982.

[280] Carlotta Smith. A speaker-based approach to aspect. *Linguistics and Philosophy*, 9:97–115, 1986.

[281] Larry Sowder. Children's solutions of story problems. *Journal of Mathematical Behavior*, 7:227–238, 1988.

[282] Guy L. Steele Jr. The definition and implementation of a computer programming language based on constraints. Technical Report 595, MIT Artificial Intelligence Laboratory, Cambridge, MA, 1980.

[283] Guy L. Steele Jr. *Common LISP: The Language.* Digital Equipment Corporation, second edition, 1990.

[284] Mark Stefik and Daniel G. Bobrow. Object-oriented programming: Themes and variations. *AI Magazine*, 6(4), Winter 1986.

[285] Lucy A. Suchman. *Plans and Situated Actions: The Problem of Human-Machine Communication.* Cambridge University Press, 1987.

[286] Gerald Sussman and Guy L. Steele Jr. CONSTRAINTS—a language for expressing almost-hierarchical descriptions. *Artificial Intelligence*, 14(1):1–39, 1980. Also: MIT AI Memo No. 502, November 1978.

[287] Barry Taylor. Tense and continuity. *Linguistics and Philosophy*, 1:199–220, 1977.

[288] Richmond H. Thomason and Robert C. Stalnaker. A semantic theory of adverbs. *Linguistic Inquiry*, 4(2):195–220, Spring 1973.

[289] Paul A. Tipler. *Physics.* Worth Publishers, Inc., 1976.

[290] Michael J. Toolan. *Narrative: A Critical Linguistic Introduction.* Routledge, London and New York, 1988.

[291] Sharon Traweek. *Beamtimes and Lifetimes: The World of High Energy Physicists.* Harvard University Press, 1988.

[292] Zeno Vendler. Verbs and times. *The Philosophical Review*, 61:143–160, 1957. Also in [293], pages 97–121.

[293] Zeno Vendler. *Linguistics* in *Philosphy.* Cornell University Press, 1967.

[294] Espen J. Vestre and Jens Erik Fenstad. Representing direct questions. Technical Report COSMOS No. 4, Department of Mathematics, University of Oslo, 1988.

[295] Rajesh S. Virkar and John W. Roach. Direct assimilation of expert-level knowledge by automatically parsing research paper abstracts. *International Journal of Expert Systems*, 1(4):281–305, 1988.

[296] Constance Weaver. *Grammar for Teachers: Perspectives and Definitions.* National Council of Teachers of English, 1979.

[297] Bonnie Lynn Webber. The interpretation of tense in discourse. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, pages 147–154, July 1987.

[298] Robert Weinstock. Laws of classical motion: What's F? what's m? what's a? *Americal Journal of Physics*, 29(10):698–702, October 1961.

[299] Simon A. Weissman. *Modern Concepts in Physics.* Oxford Book Company, 1973.

[300] Daniel S. Weld and Johann de Kleer, editors. *Readings in Qualitative Reasoning About Physicsl Systems.* Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[301] Robert Wikensky. *Planning and Understanding.* Addison-Wesley Publishing Company, Reading, MA, 1983.

[302] Robert Wilensky. Points: A theory of the structure of stories in memory. In W. Lehnert and M. Rengle, editors, *Strategies for Natural-Language Processing*, pages 345–374. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1982.

[303] Robert Wilensky. Story grammars vs. story points. *The Behavior and Brain Sciences*, 6(4), September 1983.

[304] Terry Winograd. *Understanding Natural Language.* Academic Press, 1972.

[305] Terry Winograd. *Language as a Cognitive Process, Volume 1: Syntax.* Addison-Wesley Publishing Company, Reading, MA, 1983.

[306] Ludwig Wittgenstein. *Philosophical Investigations.* Basil Blackwell Ltd., Oxford, 1953. Translated by G. E. M. Anscombe.

[307] William A. Woods. Transition network grammars for natural language analysis. *Communications of the ACM*, 13(10), October 1970.

[308] Henk Zeevat. A compositional approach to discourse representation theory. *Linguistics and Philosophy*, 12:95–131, 1989.