Technical Report 1257

Model Selection for Solving Kinematic Problems

Choon P. Goh

MIT Artificial Intelligence Laboratory

This blank page was inserted to preserve pagination.

Model Selection for Solving Kinematics Problems

by

Choon Phong Goh

S. B. Computer Science, Massachusetts Institute of Technology (1987)
S. B. Management Science, Massachusetts Institute of Technology (1989)
S. B. Cognitive Science, Massachusetts Institute of Technology (1990)

Submitted to the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

August 1990

© Massachusetts Institute of Technology 1990
All rights reserved.

This empty page was substituted for a blank page in the original document.

Model Selection
for
Solving Kinematics Problems
by
Choon Phong Goh
Submitted on 30 August 1990 to the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements
for the degree of Master of Science

ABSTRACT

There has been much interest in the area of model-based reasoning within the Artificial Intelligence community, particularly in its application to diagnosis and troubleshooting. The core issue in this thesis, simply put, is, model-based reasoning is fine, but whence the model? Where do the models come from? How do we know we have the right models? What does *the right model* mean anyway?

Our work has three major components. The first component deals with how we determine whether a piece of information is relevant to solving a problem. We have three ways of determining relevance: *derivational*, *situational* and an *order-of-magnitude* reasoning process. The second component deals with the defining and building of models for solving problems. We identify these models, determine what we need to know about them, and importantly, determine when they are appropriate. Currently, the system has a collection of four basic models and two hybrid models. This collection of models has been successfully tested on a set of fifteen simple kinematics problems. The third major component of our work deals with how the models are selected.

Thesis Supervisor: Randall Davis
Title: Professor of Computer Science and Management Science

[redacted]

ACKNOWLEDGMENTS

Working on this thesis has been a wonderful experience, intermittent frustrations notwithstanding. I have learned much from my own follies, and have benefitted even more from the wisdom of many others.

I owe my greatest gratitude to Randy Davis, my advisor, who spent many patient hours teaching me how to recognize and avoid garden-paths, and how to present ideas in a coherent and succinct manner. Each discussion session with him was intense, thought-provoking, and most refreshing. I hope I have learned at least a fraction of his teachings.

Early discussions with Mark Shirley, Dan Weld, Brian Williams and Peng Wu have influenced the broad directions of this work. Discussions with Kah-Kay Sung and Siang-Chun The inspired some of the ideas presented here. Peter Huang, Dat Nguyen, Shih-Jih Yao and Pui-Chuen Yip provided some insight into physics problem solving.

I thank my parents, my brother, and my sisters, for providing the much needed encouragement, support, and reinvigoration throughout these years.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defence under Office of Naval Research contract N00014-85-K-0124, by Digital Equipment Corporation, McDonnell Douglas Space Systems, and General Dynamics Corp.

Contents

List of Figures 6

1 Introduction 7
- 1.1 Problem Definition . . . 7
- 1.2 Domain Description . . . 9
- 1.3 System Overview . . . 9
- 1.4 Roadmap . . . 11

2 A Discussion of Relevance 12
- 2.1 Derivational-Relevance . . . 14
  - 2.1.1 An Example . . . 14
  - 2.1.2 Determining Derivational-Relevance . . . 15
- 2.2 Situational-Relevance . . . 17
  - 2.2.1 An Example . . . 18
  - 2.2.2 Determining Situational-Relevance . . . 19
- 2.3 Derivational vs Situational . . . 20
  - 2.3.1 Some Comparisons . . . 20
  - 2.3.2 Complementary Roles . . . 21
- 2.4 Order-of-Magnitude Reasoning . . . 22
- 2.5 Chapter Summary . . . 25

3 System Knowledge 26
- 3.1 The Canonical Models . . . 26
  - 3.1.1 Structure of a Canonical Model . . . 27
  - 3.1.2 Model Selection . . . 29
  - 3.1.3 Building Canonical Models . . . 32
- 3.2 General Knowledge about the World . . . 33
  - 3.2.1 Knowledge Organization . . . 34
  - 3.2.2 Event Structure . . . 35
- 3.3 Chapter Summary . . . 37

4 The Problem Solving Process **38**
4.1 Handling Distance Interval 39
4.2 Motion Handler 41
4.2.1 Identify Relevant Motion Segment 42
4.2.2 Model Selection 43
4.2.3 Model Features Update 43
4.2.4 Default Information 44
4.2.5 Equation Generation 45
4.3 Chapter Summary 49

5 System in Action **50**
5.1 Canonical Models 50
5.2 A Simple Problem 52
5.3 The Race-Car Problem 53
5.4 The Overtake Problem 57
5.5 Chapter Summary 59

6 Review of Previous Work **60**
6.1 Solving Physics Problems 60
6.2 Qualitative Reasoning 62
6.3 Chapter Summary 62

7 Future Work **64**
7.1 Short-Term Extensions 64
7.1.1 Include Other Problem Classes 64
7.1.2 Combined Derivational and Situational Relevance 65
7.1.3 Indirect Relevance 66
7.1.4 Incremental Model Generation 67
7.2 Some Long-Term Extensions 68
7.2.1 Variable Order-of-Magnitude Threshold 68
7.2.2 Create Novel Models 68
7.3 Chapter Summary 69

A Sample Kinematics Problems **71**
A.1 Problems with Just Enough Information 71
A.2 Problem with Multiple Solutions 72
A.3 Derivational and Situational Relevance 72
A.4 Order-of-Magnitude Reasoning 73
A.5 Getting Default Values 74
A.6 Reasoning with Event and Multiple Objects 74
A.7 Exploring Different Canonical Models 75

B System Outputs to Selected Problems **76**

List of Figures

1.1 A Semantic Net Representation of the OVERTAKE Problem 10

2.1 The Regular Race-Car Problem 18
2.2 The Modified Race-Car Problem 19

4.1 The Problem Solving Process 40
4.2 The Segments Between Point1 and PointN 41
4.3 The Equation Generation Process 46

Chapter 1

Introduction

There has been much interest in the area of model-based reasoning within the Artificial Intelligence community, particularly in its application to diagnosis and troubleshooting ([Patil,Szolovits&Schwartz81], [Davis84], [Pan84], [Dague87], [Goh90]). This interest is in part sustained by the advantages that a model-based system has over the traditional rule-based system: it's strongly device independent, provides more methodical coverage, better able to explain its answers, and can be less costly to use since the structural and behavioral descriptions that a model-based system needs are often used to design and build the device in the first place.

The core issue in this thesis, simply put, is, model-based reasoning is fine, but whence the model? Where do the models come from? How do we know we have the right models? What does *the right model* mean anyway?

1.1 Problem Definition

Consider what it takes to solve the following problem:

> Problem Scenario:
>
> A dark green truck of license plate 007A and length 5 m starts with a constant acceleration $6\,\mathrm{m\,s^{-2}}$. At the same instant a blue convertible car,

traveling with a constant speed of $30\,\mathrm{m\,s^{-1}}$, overtakes and passes the truck.

Query:

What is the distance between the two overtaking points?

(This problem is a modified version of [Halliday&Resnick78], P50, Q25)

Solving the *Overtake* problem above involves many non-trivial considerations. For example, does the color of the car matter? How about its license plate number and its length? Is the car moving in the same direction as the truck? What constitutes an overtaking event? Do we need any more information to solve the problem? If so how do we get it? Answer of all those questions is important to the effective modeling of the problem.

Our work has three major components. The first component deals with how we determine whether a piece of information is relevant to solving a problem. Currently, we have three ways of determining relevance: *derivational*, *situational* and an *order-of-magnitude* reasoning process.

The second component deals with the defining and building of models for solving problems. We identify these models, determine what we need to know about them, and importantly, determine when they are appropriate. Currently, the system has a collection of four basic models and two hybrid models. This collection of models has been successfully tested on a set of fifteen simple kinematics problems. The third major component of our work deals with how the models are selected.

Besides the models, two other pieces of knowledge are needed for solving kinematics problems: physics knowledge and general knowledge about the world. Quite obviously, we need physics knowledge in order to solve physics problems. We need knowledge about the world in order to understand the full implication of a given problem scenario. Consider the Overtake problem for example. When we say that "the car overtakes the truck", do we mean that the leading edge of the car passes the leading edge of the truck, or do we mean that the trailing edge of the car passes the leading edge of the truck (i.e. "complete overtaking")? Clearly, the two interpre-

tations would yield different solutions since in the complete overtaking case, the car would have to move a greater distance to pass the truck.

1.2 Domain Description

We have chosen as a domain linear kinematics problems in physics. Kinematics is the branch of physics that deals with the motion of an object without regard to the forces that caused the motion. That is, we are interested in knowing how an object moves and not why it moves.

We have two basic assumptions with regard to the domain. We assume first that Newtonian Physics applies. Hence, we do not consider any relativistic or quantum effect in the problem. We also assume that objects in the problem can be treated as rigid bodies.

The linear kinematics domain was chosen in part because it is sufficiently simple and intuitive that both the models and the conditions for their use were likely to be accessible to non-experts. Working in this domain also allowed us to use as a very basic starting point the early work of [Novak76] on solving simple statics problems.

1.3 System Overview

We have implemented a system, KINEMAT, that is able to model and solve a set of fifteen simple kinematics problems, including the *overtake* problem.

The system takes as input a straightforward semantic net translation of a kinematics problem, as in Figure 1.1, and outputs a set of simultaneous equations than provides a numerical solution to the problem. We begin with a semantic net representation because we are not investigating natural language issues; we end with a set of equations because systems already exist for doing the necessary algebra (e.g. [Macsyma83] and [Mathematica88]).

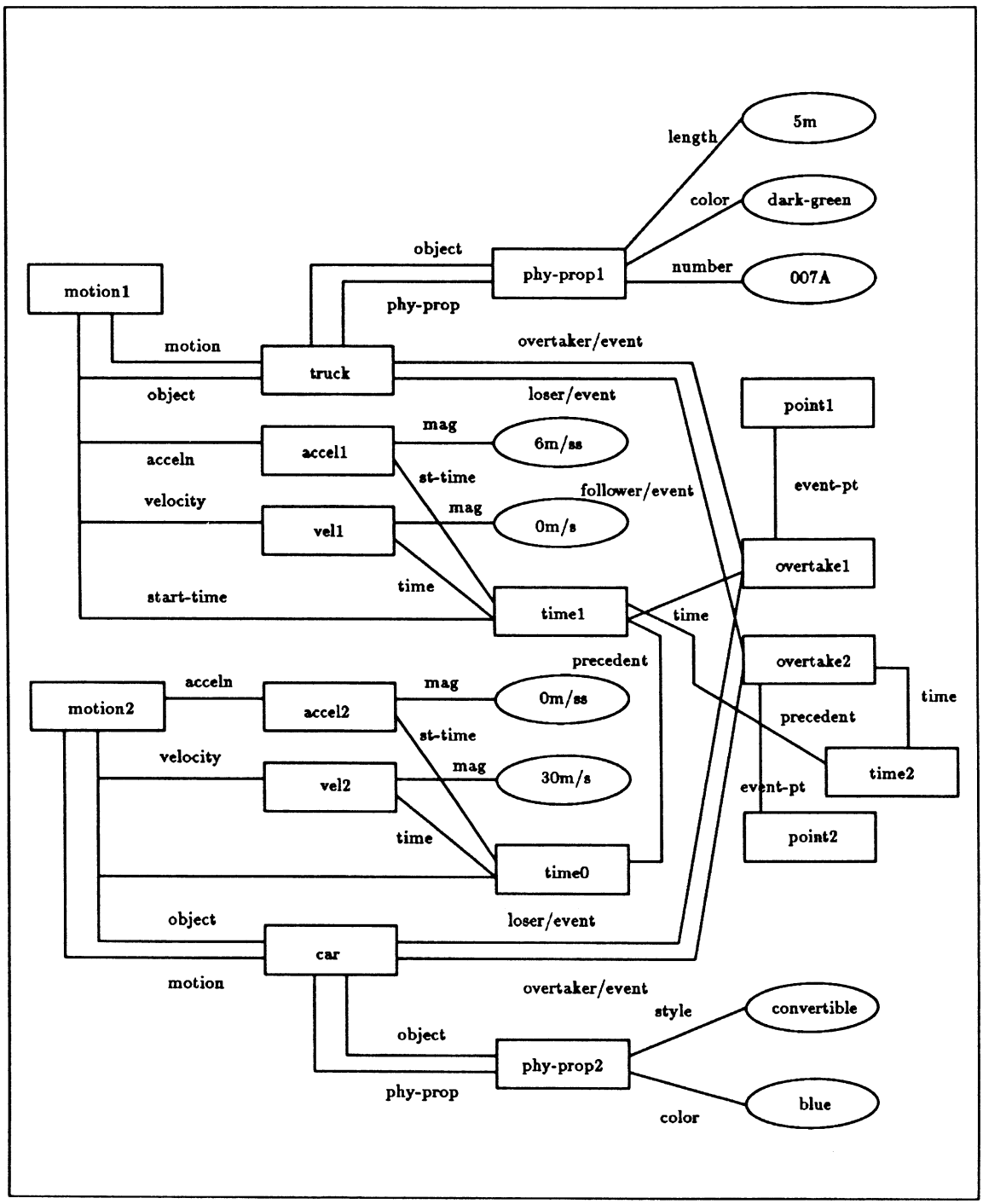

Figure 1.1: A Semantic Net Representation of the OVERTAKE Problem

Starting with a kinematics problem, the system identifies the segment of the motion that is relevant to solving the problem. It then goes through a model selection process in which the objects in the problem are appropriately modeled. Finally, the system generates a set of one or more equations sufficient for solving the problem.

1.4 Roadmap

In the next chapter we discuss how to determine if a piece of information is relevant to solving a problem. Chapter 3 describes the models in our system: what they are, how they are defined, and how they are selected. It also discusses how we deal with the general knowledge about the world. Chapter 4 discusses the problem solving process in some detail, and in chapter 5 we go through the solutions to three sample problems. Chapter 6 is devoted to reviewing related work. We conclude by discussing some possible extensions to our work, both in the short-term and in the long-term.

Chapter 2

A Discussion of Relevance

An important issue in modeling is deciding whether a piece of information is relevant to solving a problem. But what does *relevance* mean? How can one tell whether a given piece of information is relevant?

We believe that three factors affect the relevance of a piece of information: the query, the problem scenario, and the accuracy needed in the solution.

To appreciate the importance of the first factor, consider the following *race-car problem.*

> **Scenario:**
> A red car of length 4 m starts from rest with its front just behind the starting line and finishes the race when its front passes the finishing line. It moves at a constant acceleration of $10\,\mathrm{m\,s^{-2}}$ during the race. The distance between the starting and finishing lines is 1000 m.
>
> **Query:**
> What is the the velocity of the car at the point when it finishes the race?

Obviously, the query is an important determinant of the relevance of a piece of information. In this current problem, velocity and acceleration of the car are relevant information. However, given the same basic facts, if the query had been "What is

the color of the car?", the motion attributes of the car would no longer be relevant. Furthermore, if the question had been "What is the color of the starting line?", both the car's physical attributes and its motion attributes are now irrelevant.

To see the relevance of the problem scenario, consider asking the question "is the acceleration due to gravity relevant?" Clearly, we cannot answer this in absolute terms, but have to say "it depends on the particular facts of the case." Suppose we are interested in computing the velocity of a moving object. The acceleration due to gravity is relevant if the object is experiencing free fall, but is irrelevant if the object is moving on a flat horizontal plain.

The importance of the third factor stems from the observation that certain problems require very precise numerical solutions while some others require only rough estimations. Depending on how accurate we want the final solution to be, we may be willing to ignore information that has a sufficiently small effect on the final solution. For example, we might be willing to ignore the effect of air resistance when computing the speed of a fist-size iron ball experiencing free fall. This is because the air resistance exerts only a small retardation on the ball, and does not change the ball's velocity at any point during the fall by much. By contrast, we are unlikely to ignore air resistance if we are computing the velocity of a parachutist during his descend.

In general, determining whether one property is relevant to the computation of another is a deep and difficult problem, since under sufficiently elaborate conditions, we can arrange for it to be true. For example, the color of an object can affect the computation of its length in a rather obscure manner: the color of the object affects its capacity to absorb heat, heat in turn affects the length of the object through thermal expansion. Our pragmatic solution to this is to invoke a closed-world assumption: the system tries to relate two given properties, fails, and infers that there is no connection.

In this chapter, we discuss two kinds of relevance, *derivational-relevance* and *situational-relevance*, and describe how our system goes about detecting them. We also describe how an order-of-magnitude reasoning process is used to further simplify

a problem.

2.1 Derivational-Relevance

We say that a property (source property) is derivational-relevant to the computation of another property (target property) if there exists a derivation of one of more steps that relates the source property to the target property. A derivation is a chain of equations that, together, show the relationship between the source and the target properties.

2.1.1 An Example

Suppose a thrust-force T acts on an object and causes it to move. We want to know if the surface area (the source property) of the object is related to its acceleration (target property). A derivation chain relating the two properties is made up of the following three equations:

> The drag-force D acting against the motion of a body is proportional to the square of the instantaneous velocity v of the body, the surface area of the body A, the drag-coefficient associated with the body C, and the density of the medium through which the body moves ρ:
>
> $$D = \frac{1}{2}v^2 \rho A\, C$$
>
> The net force N acting on a moving body is given by the difference between the thrust-force T and the drag-force D:
>
> $$N = T - D$$
>
> The net force N acting on a body equals the product of the mass m and the acceleration a of that body:

$$N = ma$$

Since a derivation chain exists for the properties "surface area" and "acceleration", we say that the surface area of the object is derivational-relevant to the computation of its acceleration.

2.1.2 Determining Derivational-Relevance

To determine if a source property is derivational-relevant to the target property, the system carries out the following steps:

1. Search the physics knowledge base and retrieve all the equations containing the source property (the *source equations*).

   (a) If no source equation is found, return nil.

   (b) Otherwise, remove the source equations from the database (to prevent getting into infinite loops) and proceed with the next step.

2. If any of the source equations also contains the target property, then a derivation relating the two properties has been found and the chain of equations linking the two properties is returned.

3. Otherwise, make each of the *new* properties in each of the source equations a source property, and repeat step 1 for each of those source properties.

The above process is basically a breadth-first search, starting with the source property, and terminates when the target property is found, or when no source equation can be found.

As an example, suppose we want to know if a change in temperature (ΔT) of an object is derivational-relevant to its length. The system starts with a one-element derivation chain containing the source property:

ΔT

The system retrieves all the equations dealing with ΔT from its knowledge base. The following equations are found:

> The change in length of a body Δl due to a ΔT change in temperature is given by the product of the coefficient of linear expansion α of the body material, the change in temperature ΔT, and the length of the body prior to the temperature change l_i:
>
> $$\Delta l = \alpha l_i \Delta T$$
>
> The amount of heat ΔH required to effect a ΔT change in a body's temperature equals the product of the body's specific heat capacity c, the mass of the body m, and the change in temperature ΔT:
>
> $$\Delta H = cm\Delta T$$

The initial derivation chain is extended using each of the two equations to yield two chains:

Chain 1

1. ΔT
2. $\Delta l = \alpha l_i \Delta T$

Chain 2

1. ΔT
2. $\Delta H = cm\Delta T$

Since none of the source equations contains the target property, i.e. the length of an object after the temperature change, the system goes on to identify new source

properties. From the first equation, the new properties are Δl, α and l_i (ΔT is not included since it has been used as a source property before). From the second equation, the new properties are ΔH, c and m. The system uses each of these new properties as the source property in turn, and extends the respective derivation chain as a consequence.

Using Δl as the new source property, the system retrieves the following equation:

> The final length l_f of a body equals the sum of the initial length l_i of the body and any change in length Δl:

$$l_f = l_i + \Delta l$$

The first derivation chain is extended with this equation to yield:

1. ΔT
2. $\Delta l = \alpha l_i \Delta T$
3. $l_f = l_i + \Delta l$

Since the target property (l_f in this case) appears in the latest source equation, the system concludes that ΔT is derivational-relevant, and returns the corresponding derivation chain.

Note that derivational-relevance is independent of the problem scenario. A source property is always relevant to a target property if there exists a set of equations relating them.

2.2 Situational-Relevance

We say that a source property is situational-relevant to the target property if the relationship between the two is engendered by particular problem scenarios and may not exist in general. Hence, a slight variation in the scenario for a problem may render a previously situational-relevant property irrelevant, or vice versa.

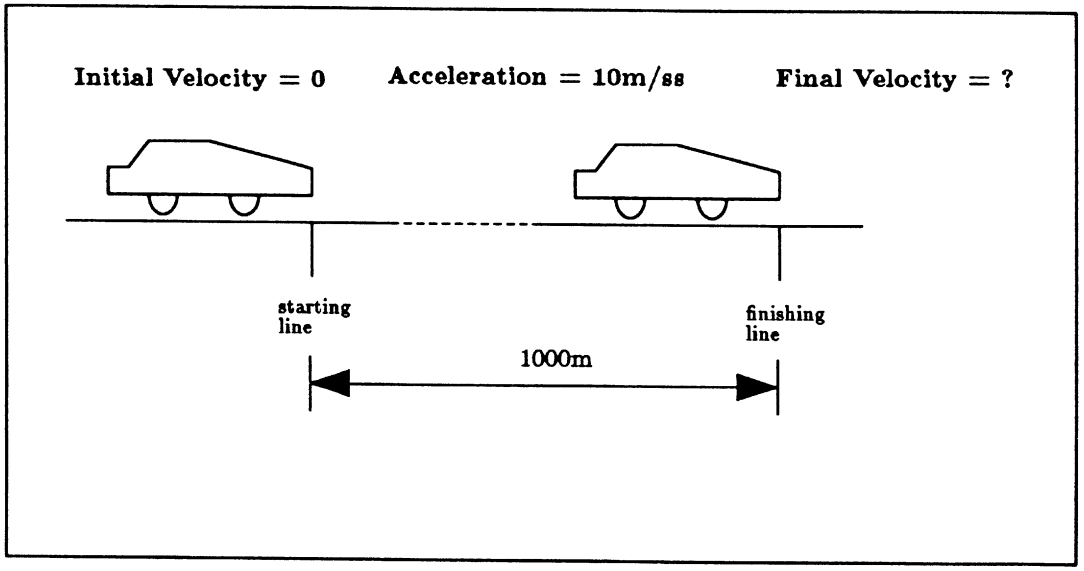

Figure 2.1: The Regular Race-Car Problem

2.2.1 An Example

As an example, consider a slight variation on the race-car problem (introduced at the beginning of this chapter) in which:

> A red car of length 4 m starts from rest with its front just behind the starting line and finishes the race when its *rear* passes the finishing line.

Figure 2.1 shows the scenario for the *regular* race-car problem and Figure 2.2 shows the scenario for the *modified* race-car problem.

From the two figures, it is clear that for the modified race-car problem the length of the car affects what its final velocity turns out to be, but for the regular race-car problem the length of the car is irrelevant.

We see here that for some source and target properties (e.g. car length and velocity), whether the former matters to the latter depends on the particular facts of the problem. We call this situational-relevance. In the modified race-car problem for example, we say that the length of the car is situational-relevant to computing its velocity.

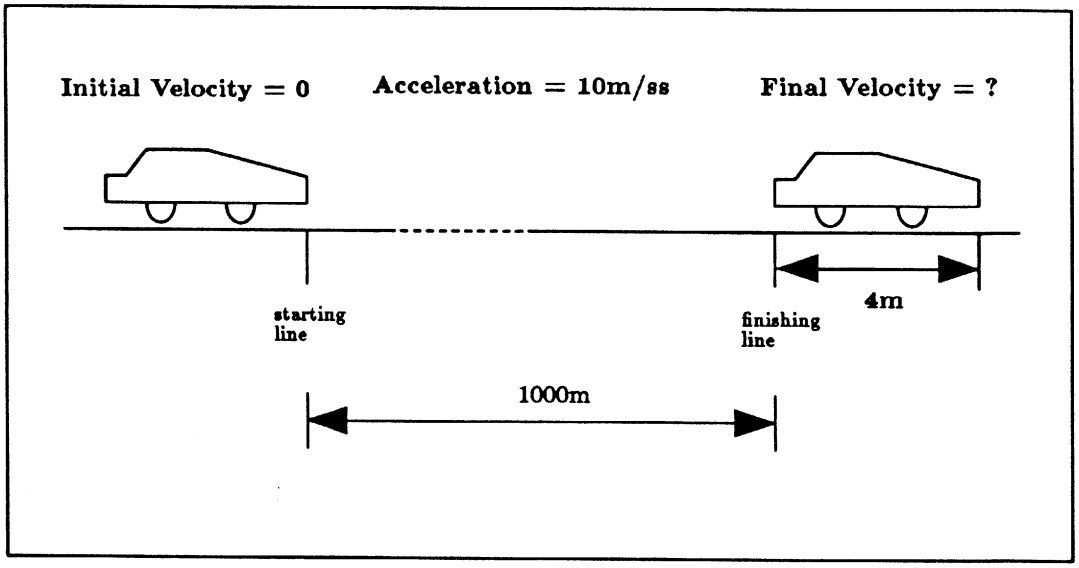

Figure 2.2: The Modified Race-Car Problem

2.2.2 Determining Situational-Relevance

It is perhaps easiest to explain how the system checks for situational-relevance through an example. Consider the regular race-car problem and the modified race-car problem again.

Assuming that the car is a rigid body, here is the basic intuition for determining situational-relevance of car length with respect to velocity: In the regular race-car problem, when the car finishes the race, every point on the car would have moved a distance exactly given by the distance between the starting and the finishing lines. For the modified race-car problem, when the car finishes the race, every point on the car would have moved a distance exactly given by the *sum* of the distance between the starting and the finishing lines *and* the length of the car. For a given acceleration, this in turn affects the value of the car's final velocity. The length of the car is therefore situational-relevant to solving the modified race-car problem.

The system carries out the reasoning by selecting a reference point on the car, in this case the default reference: the front end of the car. Next it retrieves the point on the car that corresponds to the starting point of the car's motion, in this case the

car's front end. The system then notes any displacement between that point and the car's reference point. Using the same reference point on the car, the above procedure is repeated at the ending point of the motion, with the corresponding displacement noted. The two displacements are then compared. If the displacements are the same, as in the regular race-car scenario, then the length of the car dose not matter. If the displacements are different, then the quantity corresponding to the *difference* in the displacements is relevant to solving the problem. In the modified race-car scenario, this difference corresponds to the length of the car.

The above procedure is used by our system to check for situational-relevance of an object's length, or any segment of it, whenever we are interested in computing the object's motion attribute (e.g. velocity, acceleration). We call it the *situational-length procedure.*

Clearly, whether a property is situational-relevant depends on the nature of the property and how it interacts with the other objects in the problem space. It is therefore not surprising that highly specialized, domain dependent procedures, like the situational-length procedure, are needed to detect this kind of relevance.

2.3 Derivational vs Situational

Do the derivational-relevance detection scheme (section 2.1) and the situational-relevance detection scheme (section 2.2) complement each other in their tasks? Given the two schemes, is it always possible to decide whether a relevance relationship exists between two properties? To answer those questions, we need to know the differences in emphasis and consequence of the two schemes.

2.3.1 Some Comparisons

To do derivational-relevance detection, the system uses the equation-tracing procedure described in section 2.1.2. The procedure is a general one and can be used to

check for relationship between any two properties. Any derivation chain found using this procedure is veridical and is independent of the problem scenario.

The derivational-relevance detection procedure is, however, not guaranteed to return a conclusive result. When it returns a derivation chain, we are certain that the source and target properties are related. But when it returns nil, it could be because: 1) the two properties are not related in any way, or 2) the two properties are related, but the system lacks the necessary equation(s) to relate them. In general, there is no way to tell which of the two possibilities is correct.

To do situational-relevance detection, the system uses a set of specialized procedures. Each of the procedures is useful only for detecting the relationship between two pre-specified properties, e.g. the situational-length procedure is useful only for checking whether *object length* matters when one is interested in computing some *motion attribute* of a moving object.

Unlike the derivational-relevance detection procedure, situational-relevance detection procedures are usually so specialized and narrowly focused that they can be made to guarantee conclusive outcomes. Each procedure, when activated, will end by saying either that the property is situational-relevant, or that it is situational-irrelevant.

2.3.2 Complementary Roles

In general, a property may be derivational-relevant but not situational-relevant, or vice versa, or none. When asked to determine the relevance of a source property, the system checks for both whenever possible. In fact, the system always checks for derivational-relevance, and checks for situational-relevance only when an appropriate situational-relevance detection procedure is available, i.e. when the source and target properties correspond to the two pre-specified properties of some situational-relevance detection procedure.

What happens when the system is unable to determine if a source property is derivational-relevant to computing the target property (i.e. the derivational-relevance

detection procedure returns nil), and no appropriate situational-relevance detection procedure is available? In this case, the system invokes the closed-world assumption we mentioned at the beginning of this chapter, and infers that there is no connection between the two properties. That is, the system checks for relationship between the source and target properties using the derivational-relevance detection procedure, finds none, and no suitable situational-relevance detection procedure is available, thereby concludes that the source property cannot (within its knowledge, at least) affect the target property, in other words, the source property is irrelevant.

It is important to note that as the system acquires new physics knowledge (thereby expanding the boundary of the closed-world), a property that has previously been declared irrelevant through the invocation of the closed-world assumption may now become relevant. Old problems containing that property may then be re-solved to yield more accurate solutions. This is contrasted with the case where information is declared irrelevant by a situational-relevance detection procedure. In that case, information is ignored based on some sound deductive reasoning and will not be rendered relevant through the acquisition of new physics knowledge.

2.4 Order-of-Magnitude Reasoning

Even when a piece of information has been determined to be relevant through either the derivational-relevance detection scheme or the situational-relevance detection scheme, it may still be ignored if we are convinced that ignoring it has a negligible effect on the solution. This can be done using an order-of-magnitude reasoning process.

Consider the regular race-car problem with the following addition:

> The temperature of the car goes up by 10 K during the race. The car is made of steel, which has a coefficient of linear expansion of $10.5 \times 10^{-6}\,\mathrm{K}^{-1}$.

For a given acceleration, a change in car length during the race changes the time

needed for its front to reach the finishing line, which in turn changes the value of its velocity at that point. While checking to see if temperature change affects car length, the system, as shown in section 2.1.2, arrives at the following derivation chain:

1. ΔT
2. $\Delta l = \alpha l_i \Delta T$
3. $l_f = l_i + \Delta l$

Since a derivation chain exists, it is clear that temperature change does bring about a change in car length, but is the resultant change in length significant enough to be taken into consideration? To answer that, the system carries out the following steps:

1. Capture the relationship between the source and the target properties in a single equation, with the target property as the independent variable (i.e. it appears on the left-hand-side (LHS) of the equation).

2. Separate the right-hand-side (RHS) terms into two groups: those with the source property (source-group) and those without (non-source-group). Note that a term is an expression containing one or more variables that are related to each other through either multiplication or division. Terms are related to each other by either addition or subtraction.

3. Retrieve or compute the order-of-magnitude of each term in the source-group. The order-of-magnitude of a term is the sum of the order-of-magnitude value of the individual variable that composed the term. The order-of-magnitude of a variable is the base 10 logarithm of its value. Obtain the maximum order-of-magnitude value (source-max-value).

4. Retrieve or compute the order-of-magnitude of each term in the non-source-group. Obtain the minimum order-of-magnitude value (non-source-min-value).

5. Compare the source-max-value and the non-source-min-value. If the former is at least some *threshold* order-of-magnitude smaller than the latter, declare that the source property is to be treated as irrelevant. This is a conservative approach: we ignore the source property only if the *smallest* of the terms in the non-source-group is sufficiently larger than the *largest* of the terms in the source-group.

In general, determining the appropriate threshold value for doing the order-of-magnitude reasoning is a difficult open problem, and depends critically on the nature of the task. Our pragmatic solution to this is to provide a default threshold value (currently set at "2"), but allow the user to set a new value if desirable.

Consider the length and temperature change example. The system starts by expressing the target property l_f in terms of ΔT through a substitution process (i.e. substituting the expression for ΔT into equation 3) to yield:

$$l_f = l_i + \alpha l_i \Delta T$$

The system then collects the RHS terms into source-group and non-source-group and computes the order-of-magnitude value for each term in the two groups:

Source-Group

1. l_i – order-of-magnitude: 1

Non-Source-Group

1. $\alpha l_i \Delta T$ – order-of-magnitude: -6 + 1 + 1 = -4

Finding the maximum value from the source-group and the minimum value form the non-source-group are trivial in this example as there is only one element in each group. Since the non-source-min-value is five (i.e., 1 - (-4) = 5) orders of magnitude larger than that of the source-max-value, the system proceeds to treat temperature change as irrelevant.

2.5 Chapter Summary

The system distinguishes two kinds of relevance: derivational-relevance and situational-relevance. Derivational-relevance is independent of the problem scenario: a source property is always relevant to a target property if there exists a set of equations relating them. Situational relevance, on the other hand, is highly dependent on the problem scenario: a property sometimes matters and sometimes doesn't, depending on the particular facts of the problem.

Different mechanisms are used for detecting derivational-relevance and situational-relevance. The derivational-relevance detection mechanism involves the use of an equation-tracing procedure. The situational-relevance detection mechanism is made up of a collection of specialized procedures, each is useful only for detecting the relationship between two pre-specified properties.

An order-of-magnitude reasoning process is employed to further simplify the problem by getting rid of information that has negligible effects on the solution.

In general, determining whether one property is relevant to the computation of another is a deep and difficult problem, since under sufficiently elaborate conditions, we can arrange for it to be true. Our pragmatic solution to this is to invoke a closed-world assumption: the system tries to relate one property to another, fails, and infers that there is no connection.

Chapter 3

System Knowledge

We believe that the following types of knowledge are essential to problem solving: physics knowledge, canonical models, and some general knowledge about the world.

It's clear why physics knowledge is needed: we are solving physics problems. More specifically, physics knowledge is used by the system in both derivational-relevance detection and in equation generation. The former is explained in section 2.1 and the latter is explained in section 4.2.5.

In this chapter, we concentrate on the second and the third types of knowledge. In particular, we explain what canonical models are, how are they selected, and how they may be conceived. We also discuss what we refer to as the general knowledge about the world. We describe how this knowledge is relevant to problem solving, how we organize it, and why we organize it the way we did.

3.1 The Canonical Models

Canonical models are used for modeling the physical objects mentioned in a problem statement. In many cases of problem solving not all the attributes of an object are needed. For example, the color of a car may not matter if we are trying to find its velocity. A canonical model is an idealization that allows us to attend to only a very

few features of an object. Such a model is simpler and therefore easier to reason with than the actual object. The bulk of our work has been directed toward building and defining canonical models for solving kinematics problems.

We currently have four basic canonical models and two hybrid models. The basic canonical models are: Linear-Motion-Object, Point-Object, One-Dimensional-Object and Physical-Object. One hybrid model is formed by combining the Linear-Motion-Object model and the Point-Object model, and is used to model an object that can be treated as a point in linear motion. The other hybrid model is formed by combining the Linear-Motion-Object model and the One-Dimensional-Object model.

3.1.1 Structure of a Canonical Model

Effective problem solving, as we have argued earlier, depends a great deal on how we model a problem. We believe there are three important considerations to defining a model: 1) when is the model *helpful* to solving a problem, i.e., when *should* we use the model, 2) when is it *legitimate* to use the model, i.e., when *can* we use the model, and 3) what is the consequence of using the model, i.e., *what attributes* of an object are relevant when it is mapped onto the model.

A canonical model is made up of three components, the *features*, the *motivating conditions* and the *permitting conditions*. The motivating conditions of the canonical model answer the first question raised above, and the permitting conditions answer the second question. The motivating conditions help to make explicit the circumstances under which a model would be helpful, while the permitting conditions specify the circumstances under which the use of the model is justified.

The answer to the third question lies with the features component of the canonical model. This component highlights a set of attributes that are relevant to thinking of an object in terms of the model, e.g. when we think of an object as a Linear-Motion-Object, the features in the Linear-Motion-Object model are all that we need to pay attention to.

The Linear-Motion-Object model is shown here (the internal representation has been converted to simplified English to make the content more obvious):

Linear-Motion-Object Model

1. Features

   - initial velocity
   - final velocity
   - acceleration
   - starting point
   - ending point
   - length
   - starting time
   - ending time
   - duration
   - reference frame

2. Motivating Conditions (oneof):

   - Interested in some motion attribute of an object.
   - Been requested to treat the object as a Linear-Motion-Object.

3. Permitting Conditions (all):

   - The object is in linear motion.
   - The object has zero or one dimension.

Canonical models may be nested, that is, one canonical model may be embedded within another. This happens when a canonical model requires that the object also

be successfully modeled by another canonical model, and it results in the creation of hybrid models.

Consider the Linear-Motion-Object model above. One of its permitting conditions requires that the object has zero or one dimension. To check that, the system first determines if the object can be treated as having zero-dimension, i.e., the system tries to model the object as a point (the Point-Object model). If that fails, the system tries to model the object as a one-dimensional object (the One-Dimensional-Object model). Currently, the system announces failure if it is unable to model the object as either a point or a one-dimensional object, since we do not have a Two-Dimensional-Object model or a Three-Dimensional-Object model.

The Point-Object model is shown here:

Point-Object Model

1. Features
   - Nil
2. Motivating Conditions (oneof):
   - Been requested to treat the object as a Point-Object.
3. Permitting Conditions (all):
   - The object is a rigid body.
   - The object's dimensions remain constant.
   - The dimensional properties of the object do not affect the value we are trying to compute.

3.1.2 Model Selection

Model selection may be seen as a two-phase process: 1) check motivating conditions, and 2) check permitting conditions.

The first phase involves looking for a model whose motivating conditions have been satisfied. The motivation for preferring a particular model may be obtained from the problem statement, e.g. if we are interested in computing some linear motion attribute of a moving object, then we are motivated to model the object as a Linear-Motion-Object. For the case of embedded models, the motivation to use a certain model may be supplied by the permitting condition of the superseding model (i.e. the model that initiates the request), e.g. the Linear-Motion-Object model requests that the object has zero or one dimension, hence we are motivated to try mapping the object to the Point-Object model or the One-Dimensional-Object model. Having found a model with its motivating conditions satisfied, we move on to the next phase.

The second phase involves checking the permitting conditions of the chosen model. Three mechanisms are available for performing this task:

1. A permitting condition may be satisfied by some global assumption, e.g. the rigid body requirement in the Linear-Motion-Object model is satisfied by the global assumption that all objects are rigid bodies.

2. A permitting condition may be checked by performing some simple reasoning with given information. For example, to check if the linear motion requirement of the Linear-Motion-Object model is met, we examine the acceleration and the velocity of the object in question. There are a few cases to consider here:

   - If the acceleration is 0, then the object is moving with a constant velocity, and is therefore in linear motion.
   - If the object starts from rest with a constant acceleration, then the direction of its motion is given by the direction of its acceleration, which is a constant. Hence, the object is in linear motion.
   - If the object is moving with a constant acceleration, and the direction of its velocity and the direction of its acceleration are either the same or

directly opposite, then the object is in linear motion since any change in the direction of motion will still be along a straight line.

Our current implementation does not deal with motions with a changing acceleration.

3. Checking a permitting condition may involve determining the relevance of certain object attribute. For example, one of the permitting conditions in the Point-Object model requires that the object's dimensions remain constant. To check that, we first retrieve all the non-dimensional properties of the object given in the problem, and determine for each of them whether it affects the object's dimensions. The derivational-relevance detection scheme (section 2.1) and the situational-relevance detection scheme (section 2.2) are used for this task. Relevant properties with a negligible effect on the object's dimensions may be ignored with the help of the order-of-magnitude reasoning process (section 2.4). If none of the non-dimensional properties is found to affect the object's dimensions (or at least not significantly), we conclude that the object's dimensions must not have been changed, i.e. they remain constant.

For the case of embedded models, a permitting condition of the superseding model may require that the object be mapped onto some subsidiary canonical model, e.g. the Linear-Motion-Object model requires that the object be treated as a Point-Object or a One-Dimensional-Object. The permitting conditions of the subsidiary model are then checked as before.

If all the permitting conditions are satisfied, the current model is selected for further computation. Otherwise, the model is rejected, and a new model is examined. The system announces failure if no appropriate model is found after the collection of canonical models has been exhausted.

3.1.3 Building Canonical Models

How do we conceive and build canonical models? Since the main purpose of a model is to facilitate problem solving, the model should preferably be applicable to a class of problems. Fortunately, the kinematics domain (and in fact, physics in general) offers many clearly defined classes of problems, such as those involving linear motion, circular motion, projectile motion, satellite motion etc.

Within each of the general classes of problems mentioned above, there are sub-classes of problems. For example, within the linear motion class of problems, there is a sub-class of problems, namely the point-linear-motion problems, that deals with rigid objects that can be treated as a point in linear motion. For each sub-class, there is a set of equations that are adequate for solving the problems within it. For example, the set of linear motion equations shown here is adequate for solving problems within the point-linear-motion sub-class.

$v = u + at$

$v^2 = u^2 + 2a(s - s0)$

$s = ut + 0.5at^2 + s0$

$s = 0.5(u + v)t + s0$

where:

u – initial velocity

v – final velocity

a – acceleration

t – time duration of motion

$s0$ – initial distance covered by motion

s – distance covered by motion over duration t.

As it turns out, objects in a sub-class of problems can be thought of in terms of some canonical model. Hence, a canonical model may be viewed as a stereotypical way to treat an object in problems belonging to a particular sub-class.

We need to organize information about the object in a sub-class of problems. Three types of information are needed: 1) the knowledge to recognize that a problem belongs to a particular sub-class, 2) the underlying constraints of that sub-class of problems, and 3) the attributes of the object that we should pay attention to when we solve problems in that sub-class (the variables in the corresponding set of equations often give some hint on this task).

The first type of information gives us the motivating conditions of the canonical model, the second type of information gives us the model's permitting conditions, and the third type, the features component of the model.

Some canonical models do not belong exclusively to one sub-class of problems, but are building blocks for other models. For example, problems in both the point-linear-motion sub-class and the point-projectile-motion sub-class implicitly assumed that the object concerned is or can be treated as a point. The Point-Object model is therefore a building block for the models in those two sub-classes of kinematics problems.

3.2 General Knowledge about the World

Physics knowledge is needed to predict an object's behavior under the different circumstances specified by a problem. For example, Newton's second law of motion tells us that if we apply a force F on an object of mass m resting on a flat frictionless surface, it will accelerate at a rate given by F/m.

The circumstance in a problem however, is seldom specified in a manner that readily matches the variables in a physics equation. For example, instead of saying that a force acts on the object, the problem might just state that the object is being pushed. In order to solve for the object's acceleration, the system must now know that pushing the object constitutes exerting a force on it, which in turn causes the object to move with an acceleration.

Hence, in order to successfully solve a physics problem, not only does the system need to possess the relevant physics knowledge, it also needs to understand the implications of the circumstances referred to in the problem. We refer to the latter as the general knowledge about the world. In this section, we discuss how this knowledge is organized and represented.

3.2.1 Knowledge Organization

The circumstances described in a problem may involve interactions between two or more objects, as in the case where one car overtakes another, or it may refer to the relationship between an external influence and an object, as in the case where pushing an object causes it to move.

More generally, we can view a circumstance as always involving some action by one or more agents on one or more target objects. In the overtaking example, the overtaking car is the agent, the car that's being overtaken is the target object, and the action involved is one in which the overtaking car gets ahead of the other car. In the pushing example, the mechanism that does the pushing is the agent, and the object that is being pushed is the target object, the action involved is one in which a force is exerted on the object. Hence, we can define a circumstance in terms of a combination of agents, target objects and an action.

Understanding the implications of a circumstance involves making explicit three pieces of knowledge: roles of the corresponding agents, roles of the corresponding target objects, and the consequences of the corresponding action. We introduce the notion of *events* to capture those three pieces of knowledge for different circumstances. An event is the generic way to describe a particular circumstance.

Given that we organize knowledge about the world around events, how do we conceive and build new events? At first glance, it seems that we need to build an event for each combination of agents, target objects and action. In practice however, the action component usually imposes constraints on the type and the number of

agents and target objects involved. Hence, it is reasonable to expect one new event for each new action encountered. Fortunately, [Schank&Rieger74] and [Schank75] showed that most of the common actions in the world can be described in terms of a relatively small set of primitives like move, propel, see, smell, think-about etc. The system should therefore do quite well by having one event for each action in Schank's set of primitives. We currently have three events in our system: propel, overtake and visit.

3.2.2 Event Structure

A typical event has attributes describing its participants as well as where and when the event occurs. It may also have an action component that: 1) prescribes steps to make explicit the role of each individual participant in the event, 2) prescribes steps to make explicit the interactions among the participants of the event, and 3) checks for satisfiability of certain boundary conditions.

A typical event is the overtake event, shown here (the internal representation has been converted to simplified English to make the content more obvious):

Overtake

1. Attributes:
   - Overtaker – the overtaking object.
   - Loser – the object that is being overtaken.
   - Event-Time – time when the overtaking takes place.
   - Event-Point – point where the overtaking takes place.
2. Actions:
   (a) Create a local participant node with the following attributes:
   - Object = Overtaker.

- Leading-pt = (Default: reference point of overtaker.)
- Participating-part = Leading point of overtaker.

(b) Create a local participant node with the following attributes:

- Object = Loser
- Leading-pt = (Default: reference point of Loser.)
- Participating-part = Leading point of loser.

(c) Check if the directions of the velocity of motion of both the overtaker and the loser are the same:

- If they are the same, do nothing.
- If they are not the same, reports the error.
- If only one of the directions is known, set the other one to the same direction.
- If none is given, set both to a common unknown direction D.

The attributes component of the overtake event has slots that contain information that is typically given in the problem statement: that some trailing object (overtaker) overtakes some leading object (loser) at a certain time and place. Notice that this information is likely to be different for different overtaking events.

The action component, by contrast, makes explicit what constitute an overtaking event. As stated, an overtaking event happens when the leading edge of the overtaker passes the leading edge of the loser. The action component also checks to make sure that the two participants are traveling in the same direction during the course of the event. Notice that the information provided by the action component is not explicitly given in the problem statement, and necessarily holds true for all overtaking events.

It is clear that without such general knowledge about the world, we will not be able to understand most of the problems, let alone solving them.

3.3 Chapter Summary

In this chapter, we describe two types of system knowledge in detail: 1) canonical models, and 2) general knowledge about the world.

For canonical models, the important observation is that there are general classes of problems in the world, each of which can be further constrained into sub-classes. As it turns out, objects in those sub-classes of problems can be thought of in terms of canonical models. Hence, a canonical model is really a stereotypical way to treat an object in problems belonging to a particular sub-class.

A canonical model has three components: 1) features, 2) motivating conditions, and 3) permitting conditions. The features component highlights a set of object attributes that are relevant to thinking of the object in terms of the model. The motivating conditions tell us when we should use the model. The permitting conditions tell us when we can use the model.

For knowledge about the world, the important observation is that in order to successfully solve a physics problem, not only does the system need to possess the relevant physics knowledge, it also needs to understand the implications of the circumstances referred to in the problem.

Understanding the implications of a circumstance involves making explicit three pieces of knowledge: roles of the corresponding agents, roles of the corresponding target objects, and the consequences of the corresponding action. We introduce the notion of events to capture those three pieces of knowledge for different circumstances. An event is the generic way to describe a particular circumstance.

An event typically has attributes describing its participants as well as where and when the event occurs. It may also have an action component that: 1) prescribes steps to make explicit the role of individual participant in the event, 2) prescribes steps to make explicit the interactions among the participants of the event, and 3) checks for satisfiability of certain boundary conditions.

Chapter 4

The Problem Solving Process

KINEMAT is able to solve three types of problems: 1) compute the distance between two reference points, 2) compute the time interval between two reference times and 3) compute the value of the velocity, acceleration, duration of motion, and length of motion of a moving object.

When a new problem is presented to the system, KINEMAT determines which one of the three types the problem belongs to. This is done by a straightforward examination of the query part of the problem.

If the problem concerns computing the distance between two reference points, the system first determines the path that connects the two reference points, and then identifies those segments of unknown length within the path. Next, the system calls the *motion handler* to compute each of the unknown segment-lengths. Finally, the system sums over all the segments along the path. Computing the time interval between two reference times is done in a similar manner. The system first determines the path (this time it is made up of time segments) that connects the two reference times, and then it calls the motion handler to compute each of the unknown time segments. Finally, the system sums over all the time segments within the path.

If the problem involves computing velocity, acceleration, duration of motion or length of motion, it is passed on to the motion handler directly. The motion handler

is responsible for computing the motion attribute of a moving object. To do that, the motion handler carries out a sequence of four steps. It begins by identifying the segment of the object's motion that is relevant to solving the problem. Canonical models are then chosen to model the object. Next, the motion handler tries to fill as many the feature-slots of each selected model as possible, using both given and default information. Finally, equations for solving the problem are generated, and where necessary, sub-problems are set up to seek more simultaneous equations.

Figure 4.1 shows the problem solving process.

In the next section we describe how the system computes the distance between two points. We do not describe how the system handles time interval since it is similar to the handling of distance interval. In a later section we describe the motion handler in detail.

4.1 Handling Distance Interval

Distance interval refers to the straight-line distance between two points. In general, the distance between two points may have one or more distinct segments. In Figure 4.2 for example, the distance between Point1 and PointN is made up of N-1 segments.

Our current implementation assumed that all the segments in a path lie along the straight line joining the source points (the two points whose distance apart is to be determined), that is, all intermediate points along the path are located on that line. With this assumption, the problem of computing the distance between the source points is reduced to that of computing the length of each segment of unknown length within the path.

The task of the system is first to determine the path that connects the two source points, and then to identify those segments of unknown length within the path. Next, the system calls the motion handler to compute each of the unknown segment-lengths. Finally, the system sums over all the segments along the path.

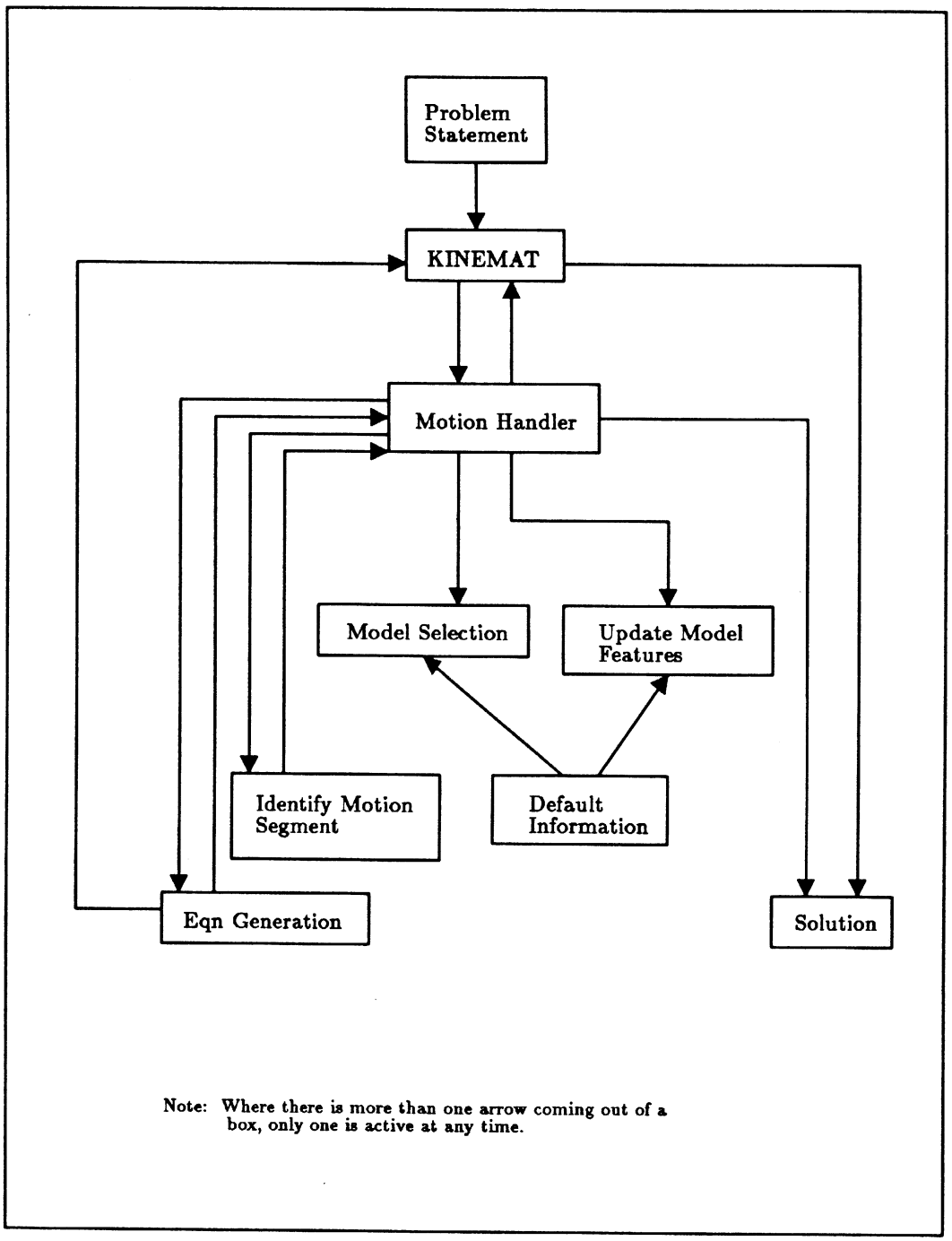

Figure 4.1: The Problem Solving Process

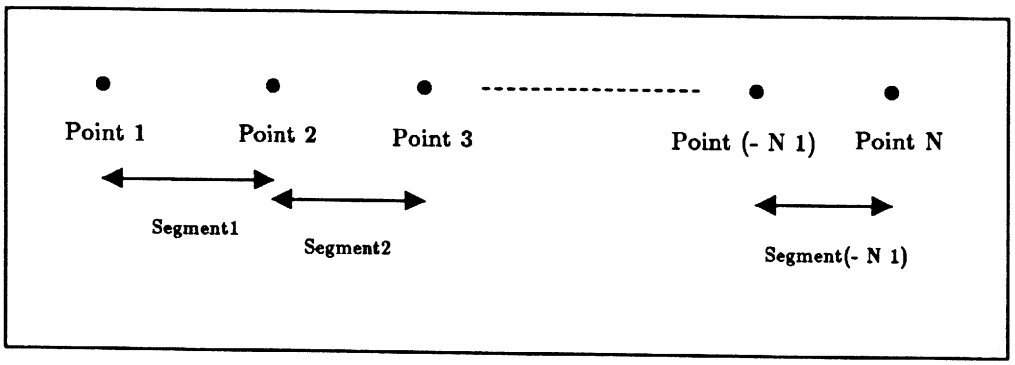

Figure 4.2: The Segments Between Point1 and PointN

To look for a path connecting the source points, the system makes a bi-directional search. The resulting path between the source points, if any, is called the *interval-path.*

Once an interval-path is found, the system retrieves from it those segments with unknown length. Note that since each segment of the path is either part of the trajectory of some moving object, or contains explicit information about its length, an unknown segment is necessarily associated with the motion of some object. The system calls the motion handler to compute the length of each unknown segment. The sum of all the segment-lengths in the path is then returned as the solution. A similar process exists for computing the interval between two time values.

4.2 Motion Handler

The motion handler is responsible for computing the motion attribute of a moving object. KINEMAT invokes the motion handler when the problem asked for the velocity, acceleration, length of motion, or the duration of motion of a moving object, or when it wants to compute the length of a motion segment within the interval-path, as explained in section 4.1.

To compute a motion attribute, the motion handler carries out a sequence of four steps. It begins by identifying a segment of the object's motion that might help in

solving the problem. This is followed by selecting appropriate canonical models for the object, and then updating the slots in the features component of the selected models. Finally, the motion handler generates the set of equations sufficient to solve the problem.

The model selection process has been explained in depth in section 3.1.2. In this section, we explain in some detail each of the other three steps: identify relevant motion segment, model features update, and equation generation. We also explain how default information is obtained.

4.2.1 Identify Relevant Motion Segment

The first thing the motion handler does is to identify the segment of the object's motion that is relevant to solving the problem. Consider the following simple problem:

> Problem Scenario:
>
> A car accelerates at a constant rate from $10\,\mathrm{m\,s^{-1}}$ to $70\,\mathrm{m\,s^{-1}}$ in $100\,\mathrm{s}$.
>
> Query:
>
> What is the car's acceleration?

Clearly, the relevant motion segment for the problem above is the segment of the car's motion between the point when its velocity is $10\,\mathrm{m\,s^{-1}}$ and the point when it reaches $70\,\mathrm{m\,s^{-1}}$.

The relevant motion segment for solving a problem might be provided explicitly, as in the case when the motion handler is asked to compute the unknown segment-lengths of an interval-path. The interesting question is how to determine the relevant segment if it is not given explicitly?

A motion segment is potentially relevant if it is delimited by any two points in space through which the object moves. In practice however, only those points for which we have some information are likely to be helpful. To choose the relevant motion segment when it is not given explicitly, the motion handler examines the motion path

of the object and retrieves those points (the reference points) that correspond to motion attributes for which we have some information. For example, we may be given the velocity of the object at those points, or the time at which the object reaches those points.

Next, the motion handler arbitrarily selects two points from the collection of reference points and uses them to identify the motion segment to be used in the next three steps, namely model selection, model features update, and equation generation. If any of the three steps fails, a new combination of two reference points will be tried by the motion handler. This process repeats until the equations for solving the problem are successfully generated, or until all combinations of two reference points are exhausted, whereupon the motion handler announces failure.

4.2.2 Model Selection

After the relevant segment has been identified, the motion handler invokes the model selection process. As described in section 3.1.2, this process involves checking the motivating and permitting conditions of each canonical model until a model is found with both set of conditions satisfied, or it announces failure if none can be found. The selected models are returned to the motion handler for features update.

4.2.3 Model Features Update

We have seen in section 3.1.1 that each canonical model has a features component. After a model has been selected, the system tries to fill in as many of the feature-slots as possible. Since the features set for different models is different, each model has its own procedure for filling the feature-slots. We consider how this is done for the Linear-Motion-Object model.

After the motion handler has determined the two reference points demarcating the relevant motion segment, and the Linear-Motion-Object model has been selected, the motion handler proceeds to fill the corresponding feature-slots.

The motion handler begins this process by determining which of the two reference points is encountered earlier by the moving object. This is done by first retrieving the time corresponding to the object's visit to each point, and then determining which is the earlier time by checking the *precedent* link. The earlier time is used for filling the staring-time slot in the model features, and the corresponding point for filling the starting-point slot. Any velocity information at that point is used to fill the initial-velocity slot. Similarly, the later time occupies the ending-time slot, with the corresponding point and velocity taking up the ending-point slot and the final-velocity slot respectively.

Finally, pieces of information that are available either directly from the problem statement or via default are inserted into the respective slots in the features component of the model. All updated models are returned to the motion handler.

4.2.4 Default Information

When information is found missing either during model selection or model features update, default information may be used. Default information comes in two forms: 1) numerical values, e.g. the default acceleration due to gravity on earth is $9.8\,\mathrm{m\,s^{-2}}$, and 2) descriptive values, e.g. the default material of a car is steel.

The default information about a certain object or quantity is usually the value that it assumes under *normal conditions*. The information may be a reflection of some natural laws of physics, e.g. the speed of sound in air is $340\,\mathrm{m\,s^{-1}}$ under standard atmospheric pressure and temperature, or, a result of human technology or convention, e.g. books are made of paper, cars are made of steel.

Default values for token nodes are obtained from the corresponding type nodes in the system by looking up the AKO links – the standard notion of inheritance. Consider the following problem:

> Problem Scenario:
>
> Given two cars, car1 and car2. The material of car2 is copper.

Query:

What is the coefficient of linear expansion of (the material of) each car?

For car1, the system tries to access information about its material, fails, then follows the AKO link to access the material of the type object CAR as the default. The default material of car1 is found to be steel. Since the coefficient of linear expansion of steel is not given in the problem statement, the system retrieves the default value from the type object STEEL.

For car2, the system successfully accessed the material of the car, it's copper. Again, since the coefficient of linear expansion of copper is not given in the problem statement, the system retrieves the default value from the type object COPPER.

Updated models are used by the system in the equation generation process.

4.2.5 Equation Generation

Using the updated canonical models, the motion handler tries to generate a set of simultaneous equations sufficient for solving the problem. Figure 4.3 shows the equation generation process.

We go through each step of the equation generation process in this section.

Retrieve Equations

The canonical model selected for an object determines the set of equations the system retrieves to solve for some attribute of that object. For example, if the canonical model chosen is the Linear-Motion-Object model, then the set of linear motion equations is retrieved.

Select Equation

From the set of equations in the model, the system selects those equations containing the desired attribute, and from those then chooses the one with the fewest unknowns.

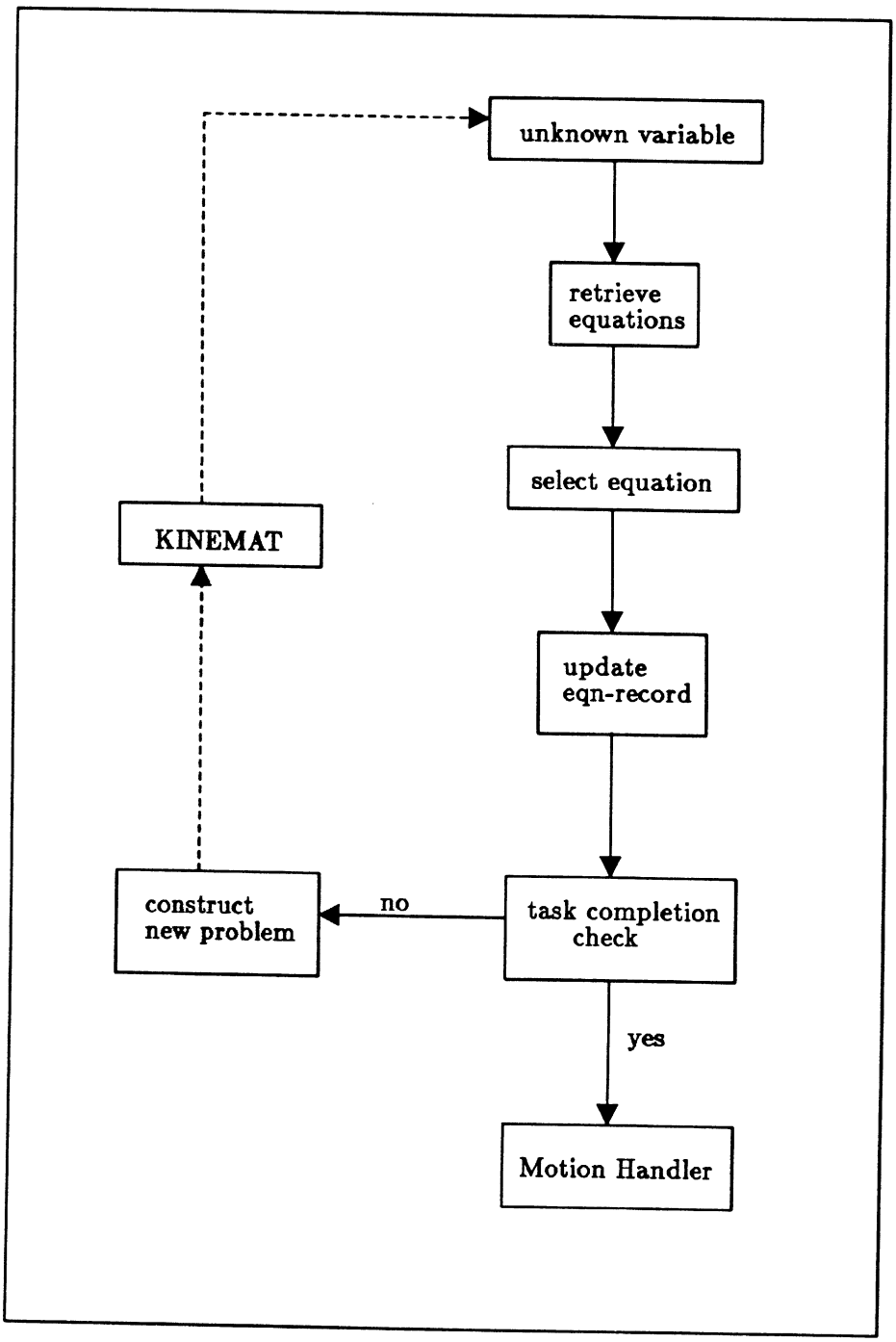

Figure 4.3: The Equation Generation Process

To select the one with the fewest unknowns, the system first determines what is *known* by checking for variable values in the feature-slots of the canonical model, and for known variables in the current problem-record structure (described in the next step). It then chooses the equation with the fewest unknowns.

Update Problem-Record

The problem-record structure keeps track of the extent to which a problem has been solved up to the current time. More specifically, it keeps track of: 1) the equations that have been selected for solving the problem (the equation-record), 2) the known variables (the known-variable-record), and 3) the outstanding variables (the outstanding-variable-record).

When the system is first asked to solve a problem, the equation it comes out with may contain more than one unknown. In this case, we need to generate simultaneous equations to solve the problem. The equation-record keeps track of the simultaneous equations that have been generated up to the current time.

A variable may be "known" in three ways: 1) its value is supplied by the problem statement, 2) its value is supplied by the system via a default, 3) its value is made known through the solving of one or more equations. The first case is straightforward. The second case has been explained in section 4.2.4. The third case requires more explanation.

Whenever a new equation is to be added to the problem-record structure, the system first obtains the list of unknown variables that are common to both the new equation and the outstanding-variable-record (the common-unknown-list). The outstanding-variable-record contains variables from equations in the equation-record whose values remain unknown up to the current time. The remaining unknown variables associated with the new equation are kept in the exclusive-unknown-list. The system then chooses any one variable from the common-unknown-list and adds it to the known-variable-record. It also removes that chosen variable from the outstanding-

variable-record. Next, variables in the exclusive-unknown-list are added to the outstanding-variable-record. Finally, the new equation is added to the equation-record.

The above process can be thought of as expressing the chosen variable from the new equation in terms of the other variables in the equation, and then substituting the resultant expression for all occurrences of that variable in the problem-record structure. Hence, the variables in the outstanding-variable-record are really the unknown variables remaining as all the equations in the equation-record are merged to yield a single equation.

Task Completion Check

As pointed out in the last step, the outstanding variables in the outstanding-variable-record are the remaining unknown variables as all the equations in the equation-record are expressed as a single equation. Hence, when there is only one outstanding variable, we have effectively one equation and one unknown, and the problem can be trivially solved.

The task completion check computes the cardinality of the outstanding-variable-record. If the answer is 1, then the system announces that a solution is found. Otherwise, the system continues with the next step.

Construct New Problem

If there is more than one variable in the outstanding-variable-record, the system has to look for more simultaneous equations. The system chooses a variable from the outstanding-variable-record and sets up a sub-problem to solve for that variable.

The sub-problem is solved as before, with the resultant equation added to the problem-record structure. Note that this is the same problem-record as before.

4.3 Chapter Summary

Given a problem statement, KINEMAT first determines the problem type. If the problem concerns computing the distance between two reference points, the system determines the path that connects the two reference points, and then identifies those segments of unknown length within the path. Next, the system calls the motion handler to compute each of the unknown segment-lengths. Finally, the system sums over all the segments along the path. Computing the time interval between two reference times is done in a similar manner. The system first determines the path (this time it is made up of time segments) that connects the two reference times, and then it calls the motion handler to compute each of the unknown time segments. Finally, the system sums over all the time segments within the path.

If the problem involves computing velocity, acceleration, duration of motion or length of motion, it is passed on to the motion handler directly. The motion handler is responsible for computing the motion attribute of a moving object. Canonical models are chosen to model the objects in the problem. The system then tries to fill as many of the model feature-slots as possible, using both given and default information. Following that, equations for solving the problem are generated, and where necessary, sub-problems are setup to seek more simultaneous equations. The problem is solved when there is one variable in the outstanding-variable-record.

Chapter 5

System in Action

We have successfully tested KINEMAT on fifteen simple kinematics problems, each of which is selected to represent a large class of problems. In this chapter, we show how Kinemat solves three problems. The first problem is straightforward, and is presented to illustrate the basic problem solving mechanisms of the system. The second problem shows how the system makes use of order-of-magnitude reasoning to simplify the problem and how it handles the case where the length of an object is relevant to solving the problem. The last problem illustrates the case where a sub-problem has to be set up in order to generate the necessary simultaneous equations.

5.1 Canonical Models

For ease of reference, the Linear-Motion-Object canonical model and the Point-Object canonical model are reproduced here:

Linear-Motion-Object Model

1. Features

    - initial velocity
    - final velocity

- acceleration
- starting point
- ending point
- length
- starting time
- ending time
- duration
- reference frame

2. Motivating Conditions (oneof):

   - Interested in some motion attribute of an object.
   - Been requested to treat the object as a Linear-Motion-Object.

3. Permitting Conditions (all):

   - The object is in linear motion.
   - The object has zero or one dimension.

Point-Object Model

1. Features

   - Nil

2. Motivating Conditions (oneof):

   - Been requested to treat the object as a Point-Object.

3. Permitting Conditions (all):

- The object is a rigid body.
- The object's dimensions remain constant.
- The dimensional properties of the object do not affect the value we are trying to compute.

5.2 A Simple Problem

Consider the following problem:

> Problem Scenario:
> A car starts from rest with a constant acceleration of $15\,\mathrm{m\,s^{-2}}$, and covers a distance of 48km in 80 s.
>
> Query:
> What is the velocity of the car at the end of the 80 s?

We are asked to determine the velocity of the car, which is a motion attribute. Hence, the problem is passed on to the motion handler. The motion handler suggests that the relevant motion segment might be between the point where the car starts moving and the point where it reaches the velocity of $80\,\mathrm{m\,s^{-1}}$.

The next step is model selection. Since we are interested in a motion attribute, one of the motivating conditions of the Linear-Motion-Object canonical model is satisfied. The system then checks if the corresponding permitting conditions are satisfied.

The first permitting condition requires that the car be in linear motion. Since the car starts from rest with a constant acceleration, the system deduces that it is indeed in linear motion.

The second condition requires that the car has zero or one dimension. To check that, the system first determines if the car can be treated as an object with zero dimension, i.e. a point object. This motivates the system to check if the car satisfies the permitting conditions of the Point-Object model.

The first permitting condition of the Point-Object model requires that the car be a rigid body, which is satisfied by our global assumption. The second and the third permitting conditions are trivially satisfied since we are not given any information on the dimensions of the car. All the permitting conditions of the Point-Object model are now satisfied, and the car is successfully modeled as a point object. As a consequence, all the permitting conditions of the Linear-Motion-Object model are satisfied, and the car is successfully modeled as a linear-motion object. This means that the car is effectively modeled as a point in linear motion, one of our hybrid models.

Next, the system updates the feature-slots in the Linear-Motion-Object model with information given in the problem statement. In particular, the initial velocity is $0\,\mathrm{m\,s^{-1}}$, the acceleration is $15\,\mathrm{m\,s^{-2}}$, the duration of motion is 80 s, and the length of motion is 48000 m. We are interested in the final velocity of the motion.

For the equation generation step, each linear motion equation is examined for the desired attribute, i.e. the final velocity. The following equations are found to be relevant:

$$v = u_{known} + a_{known} t_{known}$$

$$v^2 = u_{known}^2 + 2 a_{known} s_{known}$$

$$s_{known} = 0.5(u_{known} + v) t_{known}$$

Each of the three equations has one unknown, v, which is the desired attribute. The system arbitrarily chooses one of the equations as the solution to the problem.

5.3 The Race-Car Problem

The modified race-car problem is reproduced here for easy reference:

Problem Scenario:

A red car of length 4m starts from rest with its front just behind the starting line and finishes when its rear passes the finishing line. It moves at a constant acceleration of $10\,\mathrm{m\,s^{-2}}$ during the race. The distance between the starting and the finishing lines is 1000 m. The temperature of the car goes up by 10 K during the race. The car is made of steel, which has a coefficient of linear expansion of $10.5 \times 10^{-6}\,\mathrm{K^{-1}}$.

Query:

What is the velocity of the car at the point when it finishes the race?

Since we are interested in finding the velocity of the car at the end of the race, KINEMAT invokes the motion handler. The motion handler suggests that the relevant motion segment might be between the starting and the finishing lines.

The system tries to select an appropriate canonical model for the car. One of the motivating conditions of the Linear-Motion-Object is satisfied since we are interested in a motion attribute, namely the velocity of the car. The corresponding permitting conditions are checked.

The first permitting condition requires the car to be in linear motion, which is satisfied since the car starts from rest with a constant acceleration. For the second condition, the system first checks if the car can be modeled as a point object. This means that the Point-Object model's motivation condition is satisfied, and the system continues by checking the corresponding permitting conditions.

The car is a rigid body by our global assumption, hence the first condition is satisfied. To see if the the car's dimension remains the same, the system checks if the other two properties mentioned in the problem, namely color and temperature change of the car, affect the car's length. The system tries to relate color and length using both derivational-relevance detection scheme and situational-relevance detection scheme, fails, and infers that they are unrelated; that is, color does not affect length. The system then tries to relate temperature change and car length. The following derivation

chain is obtained using the derivation-relevance detection scheme:

1. ΔT
2. $\Delta l = \alpha\, l_i\, \Delta T$
3. $l_f = l_i + \Delta l$

Since a derivation chain is found, the system concludes that temperature change does affect the length of the car. In fact, an increase in temperature causes the length of the car to increase through thermal expansion.

To check if the increase in length due to thermal expansion is significant, the system uses the order-of-magnitude analysis described in section 2.4. The thermal expansion of the car is found to be 5 orders of magnitude smaller than the original length of the car. Given a default order-of-magnitude threshold of 2, this increase in car length is considered insignificant and is therefore ignored. Since the system has determined that color does not seem to affect length at all, and the effect of temperature change is not significant, it concludes that the length of the car remains constant. Hence, the second permitting condition is satisfied.

For the third condition, the system checks if the length of the car matters to the computation of the car's final velocity. No derivation chain is found using the derivational-relevance detection scheme, but using the situational-length procedure described in section 2.2.2, the system discovers that the length of the car does matter. Since the third condition is not satisfied, the attempt to model the car as a point object fails. This brings us back to the second permitting condition of the Linear-Motion-Object model.

Since the car cannot be treated as a point (i.e. having no dimensions), the system checks if the car can be modeled as having just one dimension. This motivates the mapping of the car to the One-Dimensional-Object model:

One-Dimensional-Object Model

1. Features

- object length

2. Motivating Conditions (oneof):

   - Been requested to treat the object as a One-Dimensional-Object.

3. Permitting Conditions (all):

   - The object is a rigid body.
   - The object's dimensions remain constant.
   - One of the dimensions of the object is relevant to the value we are trying to compute.

The permitting conditions of the One-Dimensional-Object model are similar to that of the Point-Object model. We have seen earlier that the first two conditions are satisfied. Moreover, the checking done by the situational-length procedure shows that the length of the car matters in this case. Since *length* is the only dimension given in the problem, the third condition is satisfied. The car is successfully modeled as an object with one dimension: its length. As a consequence, all the permitting conditions of the Linear-Motion-Object model are now satisfied, and the car is effectively modeled as a one-dimensional object in linear motion.

Having selected the models, the system proceeds to update the feature-slots in the Linear-Motion-Object model and the One-Dimensional-Object model. In particular, the initial velocity is $0\,\mathrm{m\,s^{-1}}$, the acceleration is $10\,\mathrm{m\,s^{-2}}$, and the length of motion of the car is 1004 m, adjusted for the car's length.

From the set of linear motion equations, the one with the desired attribute and fewest unknowns is chosen:

$$v^2 = u^2_{known} + 2a_{known}s_{known}$$

Since we have one equation and one unknown, v, we are done.

5.4 The Overtake Problem

The overtake problem from section 1.1 is reproduced here for easy reference:

> Problem Scenario:
> A dark green truck of license plate 007A and length 5 m starts with a constant acceleration $6\,m\,s^{-2}$. At the same instant a blue convertible car, traveling with a constant speed of $30\,m\,s^{-1}$, overtakes and passes the truck.
>
> Query:
> What is the distance between the two overtaking points?
> (This problem is a modified version of [Halliday&Resnick78], P50, Q25.)

Since we are interested in computing the distance between the two overtaking points (say point1 and point2, point1 being the earlier overtaking point), the system first tries to locate the path between the two points. Using the bi-direction search described in section 4.1, the system locates two paths linking the two source points: one given by the motion of the car, the other given by the motion of the truck. The system arbitrarily chooses one of the paths, say that associated with the motion of the truck. The problem of computing the distance between the two overtaking points now reduces to that of computing the length of motion of the truck, delimited by point1 and point2. The motion handler is invoked.

Since the relevant motion segment is given, the system proceeds with the model selection process. The system is motivated to check the permitting conditions of the Linear-Motion-Object model. By default, the directions of the truck's velocity and acceleration are the same. This observation, and the fact that the truck's acceleration is constant, lead the system to conclude that the truck is in linear motion. Hence, the first permitting condition is satisfied. For the second condition, the system checks if the truck can be treated as a point object, which motivates the system to map the truck to the Point-Object model.

The first permitting condition of the Point-Object model is satisfied since the truck is rigid by our global assumption. Since the system is unable to relate color and license plate number of the truck to its length, these properties are considered irrelevant. Hence, the second condition is satisfied. The last permitting condition is satisfied through the use of the situational-length procedure (the location of the truck relative to each overtaking point is made explicit by the Overtake event structure described in section 3.2.2). The truck is successfully modeled as a point object, and as a consequence, a linear-motion object. The feature-slots of the Linear-Motion-Object model are appropriately filled: initial velocity is $0\,\mathrm{m\,s^{-1}}$, acceleration is $6\,\mathrm{m\,s^{-2}}$.

Since we are interested in the length of motion, the linear motion equation chosen is:

$$s_1 = u_{known1}t_1 + 0.5a_{known1}t_1^2$$

This is an equation with 2 unknowns, namely the motion length s_1 and the duration of motion t_1. To look for another equation, a sub-problem looking for one of the two unknowns is set up.

Suppose the sub-problem is to find s_1, which is the distance between point1 and point2. The bi-directional search is again carried out, and the same two paths (one associated with the motion of the truck, the other with the motion of the car) are obtained. Since the path associated with the truck has been explored, the system examines the other path.

Going through a similar process as in the truck's case, the system successfully modeled the car as point, and then as a linear-motion object. The appropriate feature-slots are filled: initial velocity is $30\,\mathrm{m\,s^{-1}}$, acceleration is $0\,\mathrm{m\,s^{-2}}$.

Since we are interested in the motion length, the following equation is chosen:

$$s_1 = u_{known2}t_2 + 0.5a_{known2}t_2^2$$

Once again, we have an equation with two unknowns, s_1 and t_2. Since t_2 is the time duration between the two overtaking events, the system deduces that it must

have the same value as t_1. Hence, we now have two equations and two unknowns, and the problem is solved.

5.5 Chapter Summary

In this chapter we go through how KINEMAT solves three problems. The first problem illustrates the working of the system's basic problem solving mechanism. The motion handler is invoked since we are interested in computing the acceleration of a car. The system successfully modeled the car as a point in linear motion. An equation with one unknown is generated as the solution to the problem.

In the second problem, the motion handler is again invoked since we are interested in computing the velocity of a car. The system tries but fails to model the car as a point object because the car's length is found to be relevant. However, the system successfully modeled the car as a one-dimensional object in linear motion. In the process, the problem is simplified using an order-of-magnitude reasoning process (which renders the information concerning the thermal expansion of the car irrelevant). Models are updated and an equation with one unknown is obtained.

In the third problem, the system begins by locating a path joining the two overtaking points. Using the path, the motion handler goes through the model selection process in which the car in the problem is successfully modeled as a point in linear motion. An equation with two unknowns is obtained, and a sub-problem is set up to look for the other simultaneous equation. Going through a similar process with the truck in the problem, another equation with the same two unknowns is obtained. The two equations, each with the same two unknowns, are then returned as the solution.

Chapter 6

Review of Previous Work

In this chapter, we review some of the related work done on solving physics problems and on qualitative reasoning with physical systems.

6.1 Solving Physics Problems

Our work is primarily motivated by [Novak76, 77]. While Novak's effort was directed toward natural language understanding, he realized that in order to solve the underlying statics problems, his system, ISAAC, has to decide how to model the entities in the problem. A person, for example, is modeled as a *point mass* when he is standing on a ladder, but is modeled as a *pivot* if he is carrying the ladder. Following Novak, we called them canonical models. Novak's work however, did not address the crucial issues of how these models are defined and selected.

[deKleer75, 77] implemented a system, NEWTON, for solving simple physics problems involving the kinematics of objects moving on surfaces. NEWTON possessed both qualitative and quantitative knowledge. The system starts by trying to solve a problem using just the qualitative knowledge. If the attempt fails, it proceeds with the quantitative analysis. [McDermott&Larkin78] implemented a system, PH-632, for solving simple mechanics problems. The system starts with a pictorial represen-

tation of the problem, re-represents it by inserting the relevant physics principles, and then generates the set of equations for solving the problem. MECHO, another mechanics problem solver described in [Bundy78], [Bundy,Luger,Mellish&Palmer78], [Bundy,Byrd,Luger,Mellish&Palmer79] and [Lugar81], used a GPS-like means-end analysis [Newell&Simon63, 72] (the Marples algorithm [Marples74]) to derive the equations for the problem. The system starts with the unknown in a problem and searches backwards for the set of simultaneous equations sufficient to solve the problem. The search is guided by the use of means-end analysis. All three systems, NEWTON, PH-632 and MECHO, however, assumed that canonicals models for the objects in the problem are available at the outset. That is, they took as inputs idealized objects like blocks, point-mass, pulleys, lines etc, and hence completely by-passed the problem of selecting appropriate canonical models.

Insight from studies on how humans solve physics problems provide the starting framework for our problem solving process. [Novak79] and [Novak&Araya81] discussed how humans solve physics problems by using multiple views, each giving rise to a different level of analysis. For example, an object propelled upward at an angle from the surface of the earth may be viewed as traveling in a straight line if we are interested in a sufficiently small segment of the motion, but may be viewed as a projectile motion if we are considering the entire motion. This is obviously the result of some order-of-magnitude reasoning process.

[Novak&Araya80], [Larkin,McDermott,Simon&Simon80] and [Larkin83] discussed how human experts represent and solve physics problems, while [Larkin&McDermott80] and [Chi,Feltovich&Glaser81] contrasted the differences in approach between expert and novice physics problem solvers. Although we do not try to emulate the expert physics problem solver's behavior, these studies have nonethesis influenced our problem solving approach and our choice of canonical models.

6.2 Qualitative Reasoning

There has been much interest in applying qualitative reasoning to physical systems, e.g. [deKleer&Brown84], [Forbus81], [Forbus84], [Hayes79], [Kuipers86], [Kuipers87], [Weld88]. The main emphasis of this line of research is on developing an algebra for describing and reasoning with the qualitative aspects of a system. It does not, however, deal with how models of different complexities are defined and selected to handle different problems.

The work on PROMPT [Murthy&Addanki87] introduced the idea of a *Graph of Models*. Each node in the graph represents a model, and the edge between two nodes is labeled with the set of simplifying assumptions that have to be changed in getting from one model to another. [Addanki,Cremonini&Penberthy89] described the methods used in the Graph of Models paradigm for model switching. More recently, [Weld90] introduced a theory of qualitative model switching based on *approximation reformulation*.

Both [Weld90] and [Addanki,Cremonini&Penberthy89] presumed that there are available both a description of the original problem and some measurement from the real world. It is the discrepancy between the assumptions made from the description and values taken from the real world that drives the model switching. Hence, their work is set in a laboratory environment. We are solving a different problem. For our case, there are no observed values. We are solving the problem of what to do when all we have is a description of the world, as we get in problems from physics textbooks. Hence, our work is set in the textbook environment.

6.3 Chapter Summary

In this chapter, we discussed some of the related work in the field. Novak introduced the notion of canonical models in his early work on solving simple statics problems. However, his effort was directed toward natural language understanding, and did not

address the issues of how canonical models are defined and selected.

We looked at some of the other previous physics problem solving systems, namely NEWTON, PH-632 and MECHO, and noted that they didn't consider the problem of model selection. Appropriate canonical models were assumed to be available to the system at the outset.

We looked at the work on qualitative physics and found that the main emphasis there is in developing an algebra for describing and reasoning with the qualitative aspects of a system. It too does not deal with how models are selected.

The work on Graph of Models and approximation reformulation addressed the problem of automatic model switching. Both pieces of work are set in the laboratory environment as they need some observed value to help guide the model switching process. We are solving a different problem. Our work is set in a textbook environment where no observed value is available.

Chapter 7

Future Work

In the preceding chapters we have shown how KINEMAT models a problem. In particular, we have: 1) presented three ways of determining the relevance of a piece of information, 2) defined a set of four basic canonical models and two hybrid models, and 3) showed how they may be selected and used. In this chapter, we discuss some possible extensions to our work, both in the short-term and in the long-term.

7.1 Short-Term Extensions

7.1.1 Include Other Problem Classes

One natural extension is to define and build canonical models for solving other sub-classes of problems within the kinematics domain, in particular, those involving two dimensional motions like circular motion, projectile motion, satellite motion etc. With the inclusion of these sub-classes of problems, we can then investigate the solving of hybrid problems, that is, problems that span more than one sub-class. For example, we might have two moving objects in the problem, one in circular motion and the other in linear motion.

More interestingly, having multiple sub-classes of problems will allow us to investigate the conditions under which problems in one sub-class might approximate those

in a simpler sub-class. Consider the case where an object is propelled upward at an angle from the surface of the earth. Though the object's trajectory is parabolic (a projectile motion sub-class problem), it approximates that of linear if a sufficiently small segment of the motion is considered (a linear motion sub-class problem). This is likely to require some order-of-magnitude analysis at the problem-class level.

A related extension is to include the ability to deal with dynamics problems. In dynamics problems, we are interested in the forces that caused the motion of an object. We believe that our work can be naturally extended to include the dynamics domain since solving a dynamics problem often also involves computing some kinematics property.

In this case, the problem solving process can be seen as having two levels. The base level consists of models and principles for solving kinematics problems (the kinematics level), and the upper level consists of models and principles for computing forces (the force level). For example, to find the magnitude of the force needed to push a 30 kg mass along a frictionless floor, we may need to determine the acceleration of the mass – a kinematics property. We can compute the value of the acceleration at the kinematics level, and then pass the result up to the force level for further consideration.

7.1.2 Combined Derivational and Situational Relevance

To determine the relevance of a source property S to a target property T, KINEMAT currently checks if S is derivational-relevant or situational-relevant to T. In some cases however, the relationship between the source property and the target property may not be purely derivational or situational. For example, we might have a case where the source property S is situational-relevant to some intermediate property I, and I in turn is derivational-relevant to the target property T.

In general, the relationship between two given properties may be made up of a series of intermediate derivational-relevant and situational-relevant properties, and

we would like KINEMAT to be able to detect such a connection. This is an easy extension, given our existing framework, if we simply try all possible combinations of situational-relevant procedures and derivation chains derivable from the source property. The interesting problem here, however, is to figure out a way to accomplish the task without resorting to an exhaustive enumeration process.

7.1.3 Indirect Relevance

Currently, the system is only able to check for the relevance of properties that have direct impacts on the object itself, e.g. the increase in temperature of an object causes its length to increase via thermal expansion. The system is unable to infer relevance if the effect of a property on an object is *indirect*, that is, the effect is felt only through one or more intermediate mediums. Consider the following example:

> Problem Scenario:
> A steel bar of length 2 m is in a room with a thermostat setting of 20degC.
> The thermostat setting is adjusted to 30degC.
>
> Query:
> What is the length of the steel bar when the temperature in the room stabilizes?

For this problem, we would like the system to infer that the change in thermostat setting would bring about a rise in the room temperature from 20degC to 30degC over time, and stabilizes at the latter temperature. This would in turn lead to the temperature of every object in the room, including object A, to increase by 10degC. Object A's length would therefore be increased by an amount given by the thermal expansion due to its rise in temperature.

To detect indirect relevance, the straightforward equation tracing mechanism currently used to determine derivational-relevance is not sufficient: we need to include in

the detection mechanism knowledge about how the source and target objects might relate to each other through one or more intermediate mediums mentioned in the problem. What is needed then, is some functional description of each object in the problem, and a vocabulary for describing and reasoning with the relationship between objects. To relate the change in thermostat setting to the length of the steer bar for example, the system needs to take into account that both are in the same room, and that the change in the thermostat setting changes the temperature of the air in the room, and through it that of every object in the room, over a period of time. This is likely to be a fairly simple extension.

7.1.4 Incremental Model Generation

With our current system, a model is selected only if all the corresponding conditions are satisfied. If the system has exhausted its repertoire of models without finding an exact match, it announces failure.

Clearly, the system would have gathered a substantial amount of information about the object during the model selection process, even if the process fails. In particular, the system would be able to identify why a match fails, i.e. which conditions have not been satisfied. Using this information, we might be able to dynamically create a new model for the object by either removing existing constraints from or adding new ones to a chosen base model. The logical choice for the base model would be one with the fewest discrepancies.

Since the removal of an existing constraint or the adding of a new constraint might lead to unexpected side effects, the process should probably be an incremental one; that is, dealing with the discrepancies one at a time, and making sure at each iteration that both the list of features and the set of equations associated with the base model are appropriately updated.

Proper handling of the modifications to the feature-list and the equation-set are probably the two most challenging parts of this extension. Among other things,

we have to figure out which attributes are affected by the added/deleted constraint, identify the equations that are affected by those attributes, determine how those equations are affected (possibly through some sort of simulation), and compensate for the magnitude and effect of those changes. The new model so created can then be added to the system's collection of models.

7.2 Some Long-Term Extensions

7.2.1 Variable Order-of-Magnitude Threshold

With the current implementation, the order-of-magnitude threshold has a default value of 2, and the user has to change the value if it is deemed inappropriate for a particular problem.

We would like to investigate how the threshold can be dynamically chosen by the system for a given problem. To do that, we might need to go beyond the facts given in a problem and speculate on how the computed result might be used, whether, for example, it is an intermediate step in a larger process. This understanding would then provide some basis for assessing the error margin allowed in the computation. For example, the threshold value for linear expansion of a steel bar is likely to be smaller (i.e. allowing a larger error margin) if we were using the bar for hanging clothes, than if we were using it as a measuring device.

7.2.2 Create Novel Models

The incremental model generation process described in section 7.1.4 may only be feasible if some existing model can serve as the basis for the change, that is, where there is sufficient resemblance between the desired model and an existing model. In some problem however, a completely new way of modeling it might be needed in order to make it tractable.

Devising a novel model is particularly difficult since even when humans manage to do that, it is often attributed to a stroke of brilliance, an inspiration of sort. As a start however, we might investigate the problem of creating new models out of the combination of *partial* existing models. That is, instead of creating a new model by combining *complete* existing models, or by incremental debugging of a particular base model, we create it by combining only relevant properties of a selected set of models. Deciding which models to select and what properties of a selected model to include in the new model are the two main problems here.

7.3 Chapter Summary

In this chapter, we discussed some short-term and long-term extensions to our work. The first short-term extension involves building and defining models for new problem classes. With the inclusion of more sub-classes of kinematics problems, we can investigate both the solving of hybrid problems and the conditions under which problems in one sub-class might approximate those in a simpler sub-class. We also consider the extension to handle dynamics problems.

The second short-term extension deals with combining derivational-relevance detection and situational-relevance detection. This extension is intended to handle the case where the relationship between a source property and the target property is not purely derivational or situational. For example, we might have a case where the source property S is situational-relevant to some intermediate property I, and I in turn is derivational-relevant to the target property T.

The third short-term extension is concerned with finding ways to determine indirect relevance. To detect indirect relevance, the straightforward equation tracing mechanism currently used to determine derivational-relevance is not sufficient: we need to include in the detection mechanism knowledge about how the source and target objects might relate to each other through one or more intermediate mediums

mentioned in the problem.

The fourth short-term extension deals with devising an approach to do incremental model debugging. Here, we want to be able to dynamically create a new model for an object by either removing existing constraints from or adding new ones to a chosen base model. The logical choice for the base model would be one with the fewest discrepancies.

The first long-term extension discussed concerns the dynamic generation of the order-of-magnitude threshold value. To do that, we might need to go beyond the facts given in a problem and speculate on how the computed result might be used.

The second long-term extension deals with the creation of novel models. Devising a novel model is particularly difficult since even when humans manage to do that, it is often attributed to a stroke of brilliance, an inspiration of sort. As a start however, we might investigate the problem of creating new models out of the combination of partial existing models.

Appendix A

Sample Kinematics Problems

Here are the fifteen kinematics problems solved by the system. The problems are chosen to test the different aspects of the system.

A.1 Problems with Just Enough Information

We tested the basic problem solving mechanism of KINEMAT on four problems with just sufficient information. Each problem is chosen to solve for one motion attribute, namely acceleration, motion length, duration of motion, and velocity.

Compute Acceleration

A car accelerates at a constant rate from $10\,\mathrm{m\,s^{-1}}$ to $70\,\mathrm{m\,s^{-1}}$ in $100\,\mathrm{s}$. What is the car's acceleration?

Compute Motion Length

A car accelerates at a constant rate from $10\,\mathrm{m\,s^{-1}}$ to $50\,\mathrm{m\,s^{-1}}$ in $80\,\mathrm{s}$. What is the distance covered by the car during that time?

Compute Duration of Motion

How long does it take for a car traveling at the constant velocity of $30\,\mathrm{m\,s^{-1}}$ to cover a distance of 1000 m?

Compute Velocity

A car starts from rest with a constant acceleration of $10\,\mathrm{m\,s^{-2}}$. What is it's velocity after traveling for 120 m?

A.2 Problem with Multiple Solutions

There are two possible solutions to this problem. One solution makes use of the given motion length, the other makes use of the given duration of motion:

A car starts from rest with a constant acceleration of $15\,\mathrm{m\,s^{-2}}$, and covers a distance of 48 km in 80 s. What is the velocity of the car at the end of the 80 s?

A.3 Derivational and Situational Relevance

The solving of each of the four problems here involves using the derivational-relevance and the situational-relevance detection schemes.

Derivational-Relevance Detection

A white rabbit of weight 1.2 kg accelerates from $1\,\mathrm{m\,s^{-1}}$ to $3\,\mathrm{m\,s^{-1}}$ with a constant acceleration in 30 s. What is the distance traveled by the rabbit during that time?

Situational-Relevance Detection – Length Doesn't Matter

A red race car of length 4 m starts with its front just behind the starting line and finishes when its front passes the finishing line. It moves at a constant acceleration of $10\,\mathrm{m\,s^{-2}}$ during the race and completed the race in 100 s. What is the distance

between the starting line and the finishing line?

Situational-Relevance Detection – Length to be Added

A red race car of length 4 m starts with its front just behind the starting line and finishes when its rear passes the finishing line. It moves at a constant acceleration of $10\,\mathrm{m\,s^{-2}}$ during the race and completed the race in 100 s. What is the distance between the starting line and the finishing line?

Derivational-Relevance Detection – Length to be Subtracted

A red race car of length 4 m starts with its rear just behind the starting line and finishes when its front passes the finishing line. It moves at a constant acceleration of $10\,\mathrm{m\,s^{-2}}$ during the race and completed the race in 100 s. What is the distance between the starting line and the finishing line?

A.4 Order-of-Magnitude Reasoning

This problem illustrates the use of an order-of-magnitude reasoning process to render the linear expansion of the car irrelevant:

A red race car of length 4m starts with its front just behind the starting line and finishes when its rear passes the finishing line. It moves at a constant acceleration of $10\,\mathrm{m\,s^{-2}}$ during the race and completed the race in 100 s. The car is made of an alloy which has a coefficient of linear expansion of $10.5 \times 10^{-6}\,\mathrm{K^{-1}}$. The temperature of the car increases by 10 K during the race. What is the distance between the starting line and the finishing line?

A.5 Getting Default Values

The three problems here illustrate how default values may be obtained during problem solving:

Default Length for a Car

A red race car starts with its front just behind the starting line and finishes when its rear passes the finishing line. It moves at a constant acceleration of $10\,\mathrm{m\,s^{-2}}$ during the race and completed the race in 100 s. What is the distance between the starting line and the finishing line?

Default Coefficient of Linear Expansion

A red race car of length 4 m starts with its rear just behind the starting line and finishes when its rear passes the finishing line. It moves at a constant acceleration of $10\,\mathrm{m\,s^{-2}}$ during the race and completed the race in 100 s. The car is made of copper. The temperature of the car increases by 10 K during the race. What is the distance between the starting line and the finishing line?

Default Material of a Car

A red race car of length 4 m starts with its rear just behind the starting line and finishes when its rear passes the finishing line. It moves at a constant acceleration of $10\,\mathrm{m\,s^{-2}}$ during the race and completed the race in 100 s. The temperature of the car increases by 10 K during the race. What is the distance between the starting line and the finishing line?

A.6 Reasoning with Event and Multiple Objects

This problem illustrates the use of an event structure to help in understanding the problem, and the generation of simultaneous equations based on the motion of differ-

ent objects:

A dark green truck of license plate number 007A and length 5 m starts with a constant acceleration $6\,\mathrm{m\,s^{-2}}$. At the same instant a blue convertible car, traveling with a speed of $30\,\mathrm{m\,s^{-1}}$, overtakes and passes the truck. How far beyond the starting point will the truck overtake the car?

A.7 Exploring Different Canonical Models

This problem illustrates that successful mapping to a canonical model may not guaranteed an answer. When that happens, mapping based on other potentially relevant motion segment is tried: (Note that for the following problem, an object passes a point when its trailing edge reaches that point.)

A truck of length 10 m starts with its front just behind point0. It moves at a constant acceleration of $15\,\mathrm{m\,s^{-2}}$. The truck passes point1 and point2, where point1 is 50 m from point2, and point2 is 100 m away from point0. What is the velocity of the truck as it passes point2?

Appendix B

System Outputs to Selected Problems

Problem 1:

A car accelerates at a constant rate from $10\,\mathrm{m\,s^{-1}}$ to $70\,\mathrm{m\,s^{-1}}$ in 100 s. What is the car's acceleration?

Solution:

```
(KINEMAT 'quest1)
(ATTEMPT TO MAP CAR TO THE LINEAR-MOTION-OBJECT MODEL...)
(CAR IS IN LINEAR MOTION.)
(ATTEMPT TO MAP CAR TO THE POINT-OBJECT MODEL...)
(CAR IS A RIGID BODY BY OUR GLOBAL ASSUMPTION)
(MAPPING OF CAR TO THE POINT-OBJECT MODEL IS SUCCESSFUL.)
(MAPPING OF CAR TO THE LINEAR-MOTION-OBJECT MODEL IS SUCCESSFUL.)
(EQN GENERATION NEXT)
(THE ACCELERATION OF MOTION-CAR CAN BE FOUND USING
     #S(EQNS :A_EQNS
               ((EQUATION (V = U + AT)
```

```
        KNOWN-VARIABLES ((DURATION 100)
                                 (FINAL-VELOCITY 70)
                                 (INIT-VELOCITY 10))
                UNKNOWN-VARIABLES ((ACCELERATION MOTION-CAR))
                MOTION MOTION-CAR))
            :A_MODELS (#:|mv-obj9|)
            :A_MOTIONS (MOTION-CAR)
            :A_KNOWN-VAR-VALUES ((VELOCITY 70)
                                 (VELOCITY 10)
                                 (DURATION 100))
            :A_OUTSTANDING-VAR-VALUES ((ACCELERATION MOTION-CAR))))
```

Problem 2:

How long does it take for a car traveling at the constant velocity of $30\,\mathrm{m\,s^{-1}}$ to cover a distance of 1000 m?

Solution:

```
(KINEMAT 'quest2) % 3
(ATTEMPT TO MAP CAR TO THE LINEAR-MOTION-OBJECT MODEL...)
(CAR IS IN LINEAR MOTION.)
(ATTEMPT TO MAP CAR TO THE POINT-OBJECT MODEL...)
(CAR IS A RIGID BODY BY OUR GLOBAL ASSUMPTION)
(MAPPING OF CAR TO THE POINT-OBJECT MODEL IS SUCCESSFUL.)
(MAPPING OF CAR TO THE LINEAR-MOTION-OBJECT MODEL IS SUCCESSFUL.)
(EQN GENERATION NEXT)
(THE DURATION OF MOTION-CAR CAN BE FOUND USING
     #S(EQNS :A_EQNS
               ((EQUATION (S = 0.5 * (U + V) * T)
                 KNOWN-VARIABLES ((MOTION-LENGTH 1000)
```

```
                                (FINAL-VELOCITY 30)
                                (INIT-VELOCITY 30))
                    UNKNOWN-VARIABLES ((DURATION MOTION-CAR))
                    MOTION MOTION-CAR))
                :A_MODELS (#:|mv-obj3|)
                :A_MOTIONS (MOTION-CAR)
                :A_KNOWN-VAR-VALUES ((VELOCITY 30)
                                (VELOCITY 30)
                                (MOTION-LENGTH 1000)
                                (ACCELERATION 0))
                :A_OUTSTANDING-VAR-VALUES ((DURATION MOTION-CAR))))
```

Problem 3:

A car starts from rest with a constant acceleration of $15\,\mathrm{m\,s^{-2}}$, and covers a distance of 48 km in 80 s. What is the velocity of the car at the end of the 80 s?

Solution:

```
(KINEMAT 'quest3) %5
(ATTEMPT TO MAP CAR TO THE LINEAR-MOTION-OBJECT MODEL...)
(CAR IS IN LINEAR MOTION.)
(ATTEMPT TO MAP CAR TO THE POINT-OBJECT MODEL...)
(CAR IS A RIGID BODY BY OUR GLOBAL ASSUMPTION)
(MAPPING OF CAR TO THE POINT-OBJECT MODEL IS SUCCESSFUL.)
(MAPPING OF CAR TO THE LINEAR-MOTION-OBJECT MODEL IS SUCCESSFUL.)
(EQN GENERATION NEXT)
(THE VELOCITY OF MOTION-CAR CAN BE FOUND USING
     #S(EQNS :A_EQNS
               ((EQUATION (S = 0.5 * (U + V) * T)
                 KNOWN-VARIABLES ((MOTION-LENGTH 48000)
```

```
                                (DURATION 80)
                                (INIT-VELOCITY 0))
            UNKNOWN-VARIABLES ((FINAL-VELOCITY VEL-CAR-TIME1))
            MOTION MOTION-CAR))
        :A_MODELS (#:|mv-obj3|)
        :A_MOTIONS (MOTION-CAR)
        :A_KNOWN-VAR-VALUES ((VELOCITY 0)
                                (DURATION 80)
                                (MOTION-LENGTH 48000))
        :A_OUTSTANDING-VAR-VALUES ((VELOCITY VEL-CAR-TIME1))))
```

Problem 4:

A white rabbit of weight 1.2 kg accelerates from $1\,\mathrm{m\,s^{-1}}$ to $3\,\mathrm{m\,s^{-1}}$ with a constant acceleration in 30 s. What is the distance traveled by the rabbit during that time?

Solution:

```
(KINEMAT 'quest4) %6
(ATTEMPT TO MAP RABBIT TO THE LINEAR-MOTION-OBJECT MODEL...)
(RABBIT IS IN LINEAR MOTION.)
(ATTEMPT TO MAP RABBIT TO THE POINT-OBJECT MODEL...)
(RABBIT IS A RIGID BODY BY OUR GLOBAL ASSUMPTION)
((COLOR) OF (RABBIT) IS NOT RELATED TO (LENGTH) OF (RABBIT)
 BY ANY GENERAL PHYSICS EQUATION KNOWN TO THE SYSTEM.)
((WEIGHT) OF (RABBIT) IS NOT RELATED TO (LENGTH) OF (RABBIT)
 BY ANY GENERAL PHYSICS EQUATION KNOWN TO THE SYSTEM.)
((LENGTH) OF (RABBIT) IS NOT RELATED TO (MOTION-LENGTH) OF (RABBIT)
 BY ANY GENERAL PHYSICS EQUATION KNOWN TO THE SYSTEM.)
(MAPPING OF RABBIT TO THE POINT-OBJECT MODEL IS SUCCESSFUL.)
(MAPPING OF RABBIT TO THE LINEAR-MOTION-OBJECT MODEL IS SUCCESSFUL.)
```

```
(EQN GENERATION NEXT)
(THE MOTION-LENGTH OF MOTION-RABBIT CAN BE FOUND USING
     #S(EQNS :A_EQNS
               ((EQUATION (S = 0.5 * (U + V) * T)
                 KNOWN-VARIABLES ((DURATION 80)
                                  (FINAL-VELOCITY 3)
                                  (INIT-VELOCITY 1))
                 UNKNOWN-VARIABLES ((MOTION-LENGTH MOTION-RABBIT))
                 MOTION MOTION-RABBIT))
             :A_MODELS (#:|mv-obj9|)
             :A_MOTIONS (MOTION-RABBIT)
             :A_KNOWN-VAR-VALUES ((VELOCITY 3)
                                  (VELOCITY 1)
                                  (DURATION 80))
             :A_OUTSTANDING-VAR-VALUES ((MOTION-LENGTH MOTION-RABBIT))))
```

Problem 5:

A red race car of length 4 m starts with its front just behind the starting line and finishes when its rear passes the finishing line. It moves at a constant acceleration of $10\,m\,s^{-2}$ during the race and completed the race in 100 s. What is the distance between the starting line and the finishing line?

Solution:

```
(KINEMAT 'quest5) %8
(ATTEMPT TO MAP RACE-CAR TO THE LINEAR-MOTION-OBJECT MODEL...)
(RACE-CAR IS IN LINEAR MOTION.)
(ATTEMPT TO MAP RACE-CAR TO THE POINT-OBJECT MODEL...)
(RACE-CAR IS A RIGID BODY BY OUR GLOBAL ASSUMPTION)
((COLOR) OF (RACE-CAR) IS NOT RELATED TO (LENGTH) OF (RACE-CAR)
```

```
 BY ANY GENERAL PHYSICS EQUATION KNOWN TO THE SYSTEM.)
((LENGTH) OF (RACE-CAR) IS NOT RELATED TO (MOTION-LENGTH) OF (RACE-CAR)
 BY ANY GENERAL PHYSICS EQUATION KNOWN TO THE SYSTEM.)
(MAPPING OF RACE-CAR TO THE POINT-OBJECT MODEL FAILS BECAUSE
 ((THE DIMENSIONAL PROPERTY OF RACE-CAR IS SIT-RELEVANT TO SOLVING
  THE PROBLEM.)))
(ATTEMPTING TO MAP RACE-CAR TO THE ONE-DIMENSIONAL OBJECT MODEL....)
(MAPPING OF RACE-CAR TO THE ONE-DIMENSIONAL OBJECT MODEL IS SUCCESSFUL.)
(MAPPING OF RACE-CAR TO THE LINEAR-MOTION-OBJECT MODEL IS SUCCESSFUL.)
(EQN GENERATION NEXT)
((DISTANCE BETWEEN (STARTING-LINE ENDING-LINE) CAN BE SOLVED WITH
  #S(EQNS :A_EQNS
          ((EQUATION (S = UT + 0.5AT**2)
            KNOWN-VARIABLES ((DURATION 100)
                             (ACCELERATION 10)
                             (INIT-VELOCITY 0))
            UNKNOWN-VARIABLES ((+ (MOTION-LENGTH MOTION-RACE-CAR) 4))
            MOTION MOTION-RACE-CAR))
          :A_MODELS (#:|mv-obj6|)
          :A_MOTIONS (MOTION-RACE-CAR)
          :A_KNOWN-VAR-VALUES ((VELOCITY 0)
                               (DURATION 100)
                               (ACCELERATION 10))
          :A_OUTSTANDING-VAR-VALUES ((MOTION-LENGTH MOTION-RACE-CAR)))))
```

Problem 6:

A red race car of length 4m starts with its front just behind the starting line and finishes when its rear passes the finishing line. It moves at a constant acceleration of $10\,\mathrm{m\,s^{-2}}$ during the race and completed the race in 100 s. The car is made of an alloy

which has a coefficient of linear expansion of 10.5×10^{-6} K. The temperature of the car increases by 10 K during the race. What is the distance between the starting line and the finishing line?

Solution:

```
(KINEMAT 'quest6) %10
(ATTEMPT TO MAP RACE-CAR TO THE LINEAR-MOTION-OBJECT MODEL...)
(RACE-CAR IS IN LINEAR MOTION.)
(ATTEMPT TO MAP RACE-CAR TO THE POINT-OBJECT MODEL...)
(RACE-CAR IS A RIGID BODY BY OUR GLOBAL ASSUMPTION)
((COLOR) OF (RACE-CAR) IS NOT RELATED TO (LENGTH) OF (RACE-CAR)
 BY ANY GENERAL PHYSICS EQUATION KNOWN TO THE SYSTEM.)
((TEMP-CHANGE) OF (RACE-CAR) HAS NEGLIGIBLE EFFECT ON
 (LENGTH) OF (RACE-CAR) -- IGNORE IT.)
((LENGTH) OF (RACE-CAR) IS NOT RELATED TO (MOTION-LENGTH) OF (RACE-CAR)
 BY ANY GENERAL PHYSICS EQUATION KNOWN TO THE SYSTEM.)
(MAPPING OF RACE-CAR TO THE POINT-OBJECT MODEL FAILS BECAUSE
 ((THE DIMENSIONAL PROPERTY OF RACE-CAR IS SIT-RELEVANT TO SOLVING
 THE PROBLEM.)))
(ATTEMPTING TO MAP RACE-CAR TO THE ONE-DIMENSIONAL OBJECT MODEL....)
(MAPPING OF RACE-CAR TO THE ONE-DIMENSIONAL OBJECT MODEL IS SUCCESSFUL.)
(MAPPING OF RACE-CAR TO THE LINEAR-MOTION-OBJECT MODEL IS SUCCESSFUL.)
(EQN GENERATION NEXT)
((DISTANCE BETWEEN (STARTING-LINE ENDING-LINE) CAN BE SOLVED WITH
  #S(EQNS :A_EQNS
          ((EQUATION (S = UT + 0.5AT**2)
            KNOWN-VARIABLES ((DURATION 100)
                             (ACCELERATION 10)
```

```
                (INIT-VELOCITY 0))
            UNKNOWN-VARIABLES ((+ (MOTION-LENGTH MOTION-RACE-CAR) 4))
            MOTION MOTION-RACE-CAR))
        :A_MODELS (#:|mv-obj6|)
        :A_MOTIONS (MOTION-RACE-CAR)
        :A_KNOWN-VAR-VALUES ((VELOCITY 0)
                (DURATION 100)
                        (ACCELERATION 10))
        :A_OUTSTANDING-VAR-VALUES ((MOTION-LENGTH MOTION-RACE-CAR)))))
```

Problem 7:

A red race car of length 4 m starts with its rear just behind the starting line and finishes when its rear passes the finishing line. It moves at a constant acceleration of $10\,\mathrm{m\,s^{-2}}$ during the race and completed the race in 100 s. The temperature of the car increases by 10 K during the race. What is the distance between the starting line and the finishing line?

Solution

```
(KINEMAT 'quest7) %13
(ATTEMPT TO MAP RACE-CAR TO THE LINEAR-MOTION-OBJECT MODEL...)
(RACE-CAR IS IN LINEAR MOTION.)
(ATTEMPT TO MAP RACE-CAR TO THE POINT-OBJECT MODEL...)
(RACE-CAR IS A RIGID BODY BY OUR GLOBAL ASSUMPTION)
((COLOR) OF (RACE-CAR) IS NOT RELATED TO (LENGTH) OF (RACE-CAR)
 BY ANY GENERAL PHYSICS EQUATION KNOWN TO THE SYSTEM.)
(MATERIAL OF RACE-CAR IS NOT GIVEN. ASSUME USE OF DEFAULT MATERIAL -- STEEL)
((TEMP-CHANGE) OF (RACE-CAR) HAS NEGLIGIBLE EFFECT ON
 (LENGTH) OF (RACE-CAR) -- IGNORE IT.)
```

```
((LENGTH) OF (RACE-CAR) IS NOT RELATED TO (MOTION-LENGTH) OF (RACE-CAR)
 BY ANY GENERAL PHYSICS EQUATION KNOWN TO THE SYSTEM.)
(MAPPING OF RACE-CAR TO THE POINT-OBJECT MODEL IS SUCCESSFUL.)
(MAPPING OF RACE-CAR TO THE LINEAR-MOTION-OBJECT MODEL IS SUCCESSFUL.)
(EQN GENERATION NEXT)
((DISTANCE BETWEEN (STARTING-LINE ENDING-LINE) CAN BE SOLVED WITH
  #S(EQNS :A_EQNS
          ((EQUATION (S = UT + 0.5AT**2)
            KNOWN-VARIABLES ((DURATION 100)
                     (ACCELERATION 10)
                     (INIT-VELOCITY 0))
            UNKNOWN-VARIABLES ((MOTION-LENGTH MOTION-RACE-CAR))
     MOTION MOTION-RACE-CAR))
          :A_MODELS (#:|mv-obj6|)
          :A_MOTIONS (MOTION-RACE-CAR)
          :A_KNOWN-VAR-VALUES ((VELOCITY 0)
                     (DURATION 100)
                     (ACCELERATION 10))
          :A_OUTSTANDING-VAR-VALUES ((MOTION-LENGTH MOTION-RACE-CAR)))))
```

Problem 8:

A dark green truck of license plate number 007A and length 5 m starts with a constant acceleration $6\,\mathrm{m\,s^{-2}}$. At the same instant a blue convertible car, traveling with a speed of $30\,\mathrm{m\,s^{-1}}$, overtakes and passes the truck. How far beyond the starting point will the truck overtake the car?

Solution:

```
(KINEMAT 'quest8) %14
(ATTEMPT TO MAP CAR1 TO THE LINEAR-MOTION-OBJECT MODEL...)
```

```
(CAR1 IS IN LINEAR MOTION.)
(ATTEMPT TO MAP CAR1 TO THE POINT-OBJECT MODEL...)
(CAR1 IS A RIGID BODY BY OUR GLOBAL ASSUMPTION)
((COLOR) OF (CAR1) IS NOT RELATED TO (LENGTH) OF (CAR1)
 BY ANY GENERAL PHYSICS EQUATION KNOWN TO THE SYSTEM.)
((ID-NUM) OF (CAR1) IS NOT RELATED TO (LENGTH) OF (CAR1)
 BY ANY GENERAL PHYSICS EQUATION KNOWN TO THE SYSTEM.)
((LENGTH) OF (CAR1) IS NOT RELATED TO (MOTION-LENGTH) OF (CAR1)
 BY ANY GENERAL PHYSICS EQUATION KNOWN TO THE SYSTEM.)
(MAPPING OF CAR1 TO THE POINT-OBJECT MODEL FAILS BECAUSE
 ((THE DIMENSIONAL PROPERTY OF CAR1 IS SIT-RELEVANT TO SOLVING
  THE PROBLEM.)))
(ATTEMPTING TO MAP CAR1 TO THE ONE-DIMENSIONAL OBJECT MODEL....)
(MAPPING OF CAR1 TO THE ONE-DIMENSIONAL OBJECT MODEL IS SUCCESSFUL.)
(EQN GENERATION NEXT)
(ATTEMPT TO MAP CAR2 TO THE LINEAR-MOTION-OBJECT MODEL...)
(CAR2 IS IN LINEAR MOTION.)
(ATTEMPT TO MAP CAR2 TO THE POINT-OBJECT MODEL...)
(CAR2 IS A RIGID BODY BY OUR GLOBAL ASSUMPTION)
((COLOR) OF (CAR2) IS NOT RELATED TO (LENGTH) OF (CAR2)
 BY ANY GENERAL PHYSICS EQUATION KNOWN TO THE SYSTEM.)
((STYLE) OF (CAR2) IS NOT RELATED TO (LENGTH) OF (CAR2)
 BY ANY GENERAL PHYSICS EQUATION KNOWN TO THE SYSTEM.)
((LENGTH) OF (CAR2) IS NOT RELATED TO (MOTION-LENGTH) OF (CAR2)
 BY ANY GENERAL PHYSICS EQUATION KNOWN TO THE SYSTEM.)
(MAPPING OF CAR2 TO THE POINT-OBJECT MODEL FAILS BECAUSE
 ((THE DIMENSIONAL PROPERTY OF CAR2 IS SIT-RELEVANT TO SOLVING
  THE PROBLEM.)))
```

```
(ATTEMPTING TO MAP CAR2 TO THE ONE-DIMENSIONAL OBJECT MODEL....)
(MAPPING OF CAR2 TO THE ONE-DIMENSIONAL OBJECT MODEL IS SUCCESSFUL.)
(THE LENGTH OF CAR2 IS NOT KNOWN --- USE DEFAULT LENGTH.)
(EQN GENERATION NEXT)
((DISTANCE BETWEEN (POINT1 POINT2) CAN BE SOLVED WITH
  #S(EQNS :A_EQNS
          ((EQUATION (S = 0.5 * (U + V) * T)
            KNOWN-VARIABLES ((FINAL-VELOCITY 30)
                             (INIT-VELOCITY 30))
            UNKNOWN-VARIABLES ((- (MOTION-PTS (POINT1 POINT2)) 5)
                               (MOTION-TIMES (TIME1 TIME2)))
     MOTION MOTION-CAR2)
           (EQUATION (S = UT + 0.5AT**2)
            KNOWN-VARIABLES ((ACCELERATION 6)
                             (INIT-VELOCITY 0))
            UNKNOWN-VARIABLES ((+ (MOTION-PTS (POINT1 POINT2)) 5)
                                  (MOTION-TIMES (TIME1 TIME2)))
     MOTION MOTION-CAR1))
      :A_MODELS (#:|mv-obj14| #:|mv-obj22|)
      :A_MOTIONS (MOTION-CAR1 MOTION-CAR2)
      :A_KNOWN-VAR-VALUES ((ACCELERATION 0)
                           (VELOCITY 30)
                                   (MOTION-PTS (POINT1 POINT2))
                           (VELOCITY 0)
                                   (ACCELERATION 6))
      :A_OUTSTANDING-VAR-VALUES ((MOTION-TIMES (TIME1 TIME2))))))
```

Problem 9:

A truck of length 10 m starts with its front just behind point0. It moves at a constant

acceleration of $15\,m\,s^{-2}$. The truck passes point1 and point2, where point1 is 50 m from point2, and point2 is 100 m away from point0. What is the velocity of the truck as it passes point2?

Solution:

```
(KINEMAT 'quest9) %15
(ATTEMPT TO MAP TRUCK TO THE LINEAR-MOTION-OBJECT MODEL...)
(TRUCK IS IN LINEAR MOTION.)
(ATTEMPT TO MAP TRUCK TO THE POINT-OBJECT MODEL...)
(TRUCK IS A RIGID BODY BY OUR GLOBAL ASSUMPTION)
((LENGTH) OF (TRUCK) IS NOT RELATED TO (VELOCITY) OF (TRUCK)
 BY ANY GENERAL PHYSICS EQUATION KNOWN TO THE SYSTEM.)
(MAPPING OF TRUCK TO THE POINT-OBJECT MODEL IS SUCCESSFUL.)
(MAPPING OF TRUCK TO THE LINEAR-MOTION-OBJECT MODEL IS SUCCESSFUL.)
(EQN GENERATION NEXT)
(MAPPING OF TRUCK TO THE POINT-OBJECT MODEL DOES NOT YIELD ANY SOLUTION.)
(ATTEMPTING TO MAP TRUCK TO THE ONE-DIMENSIONAL OBJECT MODEL....)
(MAPPING OF TRUCK TO THE ONE-DIMENSIONAL OBJECT MODEL IS SUCCESSFUL.)
(EQN GENERATION NEXT)
(THE VELOCITY OF MOTION-TRUCK CAN BE FOUND USING
     #S(EQNS :A_EQNS
               ((EQUATION (V**2 = U**2 + 2AS)
                 KNOWN-VARIABLES ((MOTION-LENGTH 110)
                                  (ACCELERATION 15)
                                  (INIT-VELOCITY 0))
                 UNKNOWN-VARIABLES ((FINAL-VELOCITY VEL-TRUCK-TIME2))
                 MOTION MOTION-TRUCK))
             :A_MODELS (#:|mv-obj4|)
```

```
:A_MOTIONS (MOTION-TRUCK)
:A_KNOWN-VAR-VALUES ((VELOCITY 0)
                     (MOTION-LENGTH 110)
                     (ACCELERATION 16))
:A_OUTSTANDING-VAR-VALUES ((VELOCITY VEL-TRUCK-TIME2))))
```

Bibliography

[Addanki,Cremonini&Penberthy89] S. Addanki, R. Cremonini, and J. S. Penberthy. *Reasoning about Assumptions in Graph of Model.* In Proceedings of IJCAI-89, August 1989.

[Bundy78] A. Bundy. *Will it Reach the Top? Prediction in the Mechanics World.* Aritificial Intelligence, 10:111-122, 1978.

[Bundy,Luger,Mellish&Palmer78] A. Bundy, G. Luger, C. Mellish, and M. Palmer. *Knowledge about Knowledge: Making Decisions in Mechanics Problem Solving.* In Proceedings of AISB\GI Conference, 71-82, 1978.

[Bundy,Byrd,Luger,Mellish&Palmer79] A. Bundy, L. Byrd, G. Luger, C. Mellish, and M. Palmer. *Solving Mechanics Problems Using Meta-Level Inference.* In Proceedings of IJCAI-79, 1979.

[Chi,Feltovich&Glaser81] M. Chi, P. Feltovich, and R. Glaser. *Categorization and Representation of Physics Problems by Experts and Novices.* Cognitive Science, 5, 1981.

[Dague,Raiman&Deves87] P. Dague, O. Raiman, and P. Deves. *Troubleshooting: When Modeling is the Difficulty.* In Proceedings of AAAI-87, Seattle, WA, August 1987.

[Davis84] R. Davis. *Diagnostic Reasoning Based on Structure nad Behavior.* Artificial Intelligence 24(3), December 1984.

[deKleer75] J. deKleer. *Qualitative and Quantitative Knowledge in Classical Mechanics.* MIT AI Lab TR-352, 1975.

[deKleer77] J. deKleer. *Multiple Representations of Knowledge in a Mechanics Problem Solver.* IJCAI-5, 1977.

[deKleer87] J. de Kleer. *Diagnosing Multiple Faults.* Artificial Intelligence 32(1):97-130.

[deKleer&Brown84] J. de Kleer, and J. Brown. *A Qualitative Physics based on Confluences.* Artificial Intelligence, 24, 1984.

[Forbus81] K. Forbus. *A Study of Qualitative and Geometric Knowledge in Reasoning about Motion.* In Proceedings of the National Conference on AI, August 1980.

[Forbus84] K. Forbus. *Qualitative Process Theory.* MIT AI Lab TR-789, July 1981.

[Goh90] C. Goh. *TRSHOOT: A model-based Troubleshooter.* In Proceedings of IEA/AIE-90, Charlston, SC, July 1990.

[Halliday&Resnick78] D. Halliday, and R. Resnick. *Physics.* John Wiley & Sons, New York, 1978.

[Hayes79] P. Hayes. *The Naive Physics Manifesto.* Expert System in the Micro-Electronic Age, (D. Michie ed), Edinburgh University Press, May 1979.

[Kuipers86] B. Kuipers. *Qualitative Simulation.* Artificial Intelligence, 29, September 1986.

[Kuipers87] B. Kuipers. *Abstraction by Time-Scale in Qualitative Simulation.* In Proceedings of AAAI-87, July 1987.

[Larkin,McDermott,Simon&Simon80] J. Larkin, J. McDermott, D. Simon, and H. Simon. *Models of Competence in Solving Physics Problems.* Cognitive Science: 317-345, 1980.

[Larkin,McDermott,Simon&Simon80] J. Larkin, J. McDermott, D. Simon, and H. Simon. *Expert and Novice Performance in Solving Physics Problems.* Science, vol. 208, June 1980.

[Larkin83] J. Larkin. *The Role of Problem Representation in Physics.* Mental Models (D. Gentner and A. Stevens eds.), Hillsdale, NJ., Lawrence Erlbaum Associates, 1983.

[Lugar81] G. Lugar. *Mathematical Model Building in the Solution of MEchanics Problems: Human Protocals and the MECHO Trace.* Cognitive Science 5: 55-77, 1981.

[Macsyma83] The Mathlab Group. *Macsyma Reference Manual, Version 10.* Laboratory for Computer Science, MIT, January 1983.

[Marples74] D. Marples. *Argument and Technique in the Solution of Problems in Mechanics and electricity.* Department of Engineering, Cambridge University, 1974.

[Mathematica88] S. Wolfram. *Mathematica: A System for Doing Mathematics.* Addison-Wesley, MA 1988.

[McDermott&Larkin78] J. McDermott, and J. Larkin. *Re-representing Textbook Physics Problems.* CSCSI-2, 1978.

[Minsky75] M. Minsky. *A Framework for Representing Knowledge.* The Psychology of Computer Vision, McGraw-Hill, New York, 1975.

[Murthy&Addanki87] S. Murthy, and S. Addanki. *PROMPT: An Innovative Design Tool.* In Proceedings of AAAI-87, August 1987.

[Newell&Simon63] A. Newell, and H. Simon. *GPS: A Program that Simulates Human Thought.* Coputers and Thought (Feigenbaum & Feldman Eds.), New York: McGraw-Hill, 1963.

[Novak76] G. Novak. *Computer Understanding of Physics Problems Stated in Natural Language.* TR-NL-30, Computer Science Department, University of Texas at Austin, 1976.

[Novak77] G. Novak. *Representations of Knowledge in a Program for Solving Physics Problems.* In Proceedings of IJCAI-5, 1977.

[Novak79] G. Novak. *Goals and Methodology of Research on Solving Physics Problems.* TR-158, CS Dept., University of Texas at Austin, 1979.

[Novak&Araya80] G. Novak, and A. Araya. *Research on Expert Problem Solving in Physics.* In Proceedings of AAAI-80, 1980.

[Novak&Araya81] G. Novak, and A. Araya. *Physics Problem Solving Using Multiple Views.* TR-173, Computer Science Department, University of Texas at Austin, 1981.

[Pan84] J. Pan. *Qualitative Reasoning with Deep-level Mechanism Models for Diagnoses of Mechanical Failures.* In Proceedings of CAIA-84, Denver, Colorado, IEEE, December 1984.

[Patil,Szolovits&Schwartz81] R. Patil, P. Szolovits, and W. Schwartz. *Causal understanding of patient illness in medical diagnosis.* In Proceedings of IJCAI-81, Vancouver, BC, August 1981.

[Schank&Rieger74] R. Schank, and C. Rieger. *Inference and the Computer Understanding of Naturally Language.* Artifical Intelligence, vol. 5, no. 4, 1974.

[Schank75] R. Schank. *Conceptual Information Processing.* North-Holland Publishing Company, New York, 1975.

[Weld87] D. Weld. *Comparative Analysis.* Artificial Intelligence, 36(3), October 1988.

[Weld90] D. Weld. *Approximation Reformulation.* In Proceedings of AAAI-90, August 1990.

This blank page was inserted to preserve pagination.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AI-TR 1257	2. GOVT ACCESSION NO. AD-A228692	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Model Selection for Solving Kinematics Problems		5. TYPE OF REPORT & PERIOD COVERED technical report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Choon P. Goh		8. CONTRACT OR GRANT NUMBER(s) N00014-85-K-0124
9. PERFORMING ORGANIZATION NAME AND ADDRESS Artificial Intelligence Laboratory 545 Technology Square Cambridge, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Advanced Research Projects Agency 1400 Wilson Blvd. Arlington, VA 22209		12. REPORT DATE September 1990 13. NUMBER OF PAGES 91
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Office of Naval Research Information Systems Arlington, VA 22217		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution is unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

canonical models
model selection
lineau kinematics
determine relevance
equation generation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

There has been much interest in the area of model-based reasoning within the Artificial Intelligence community, particularly in its application to diagnosis and troubleshooting. The core issue in this thesis, simply put, is, model-based reasoning is fine, but whence the model? Where do the models come from? How do we know we have the right models? What does *the right model* mean anyway?

(continued on back)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Block 20 continued:

Our work has three major components. The first component deals with how we determine whether a piece of information is relevant to solving a problem. We have three ways of determining relevance: *derivational*, *situational* and an *order-of-magnitude* reasoning process. The second component deals with the defining and building of models for solving problems. We identify these models, determine what we need to know about them, and importantly, determine when they are appropriate. Currently, the system has a collection of four basic models and two hybrid models. This collection of models has been successfully tested on a set of fifteen simple kinematics problems. The third major component of our work deals with how the models are selected.

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**

Scanned

Date: 6/26/1995

M.I.T. Libraries
Document Services