Representation of Models for Expert Problem Solving in Physics

Hyung Joon Kook, *Member, IEEE,* and Gordon S. Novak, Jr., *Member, IEEE*

Abstract— **Expertise in solving physics problems is characterized by the ability to *set up* or *represent* a problem. An expert's representation employs idealized formal models, such as a point mass or the principle of uniform circular motion, for modeling objects and relationships in the problem. A computer program, APEX, has been developed to investigate such models. Two types of models are defined: *canonical physical objects* and *physical models*. During problem solving, the problem is represented as a data connection network, which is progressively augmented by these models in the form of additional network elements. APEX employs *views* as a representational framework for connecting the initially informal objects to the formal models of the domain. The view framework supports multiple representations (e.g., viewing many objects as a single canonical physical object), handling of incompletely specified problems, and *invertibility* of the views. This computational framework provides a powerful representational mechanism that allows a finite set of physical principles to be applied to a potentially infinite variety of problems. As a knowledge engineering technique, views allow general principles to be applied to a variety of objects whose representations differ.**

Index Terms— **Canonical objects, knowledge engineering, knowledge representation, physical models, physics problem solving, views.**

I. Introduction

PHYSICS problems are often presented in terms of informal, real-world objects and relationships among them. An important task of a problem solver is to obtain a formal representation of the problem by interpreting these objects and relationships into formal, abstract models of physics. Competence in solving a physics problem derives mainly from the skill of *setting up* the problem in terms of these models, rather than from mere mathematical skill [13]. An expert in physics is believed to possess a finite set of such models that may be applied to solve a large number of complex problems.

We have been investigating a machine problem solver in physics with representation and model-building as the primary research issues [11]. APEX (*A Physics Expert*) is a computer program built in an object-oriented programming environment [21] for solving physics problems in a model-based representational framework. Two types of models are employed in APEX: *canonical physical objects* and *physical models.* Canonical physical objects are idealized objects that are meaningful in physics or mathematics, such as a point mass, a circle, or an ideal rope. Physical models are data structures for representing the laws and principles of physics.

APEX extends the work in ISAAC [18], [19], a program that solves physics problems stated in English, with the main research issue shifted from natural language understanding to representational aspects of physics problem solving. Many underlying ideas in APEX are supported by psychological research [5], in which experts and novices were compared in categorizing a given set of problems. Other studies of performance differences between experts and novices [14], [15] suggest the importance of model-building behavior of experts. [1] and [17] describe the use of graphs that relate physical models, with edges of a graph representing differences in the assumptions of the models. [16] describes abstraction models in the domain of programming. [2] includes descriptions of several approaches to qualitative reasoning about physical systems, including qualitative process theory [7], confluences [6], and qualitative simulation [12]. Whether the models used are purely quantitative or are qualitative, a central issue is the model formulation process that we address; it is crucial when dealing with problems that are presented in the informal terms of the real world, often incompletely specified.

Manuscript received March 2, 1990; revised August 20, 1990. This work was supported by the U.S. Army Research Office under Contract DAAG29-84-K-0060. Computer equipment used in this research was donated by Xerox Corporation and Hewlett Packard.

H. J. Kook is with the Department of Computer Science, King Sejong University, Seoul, Korea 133-747.

G. S. Novak Jr. is with the Artificial Intelligence Laboratory, Department of Computer Sciences, University of Texas at Austin, Austin, TX 78712.

IEEE Log Number 9042112.

II. Overview of APEX

The organization of the program and its data is shown in Fig. 1. In the diagram, programs are represented by boxes with double lines, and data structures and rules by plain boxes; links between boxes represent data flow.

A physics problem is input as a semantic network describing features of objects and their relationships. The inputs used in testing APEX were hand-simulations of the output of a hypothetical English parser similar to those of ISAAC [18] and BEATRIX [22], [3], [4]. In one case, the output of BEATRIX was mechanically reformatted to serve as input to APEX.

Given an initial representation of a problem, the process of solving it occurs in several steps. First, guided by an inference system that reasons about the objects and their interactions, APEX transforms the initial problem into a representation in terms of *canonical physical objects,* such as a point mass, selected from a library. This transformation is performed by constructing a data connection network between the two objects that provides a computational facility for *viewing* an object as another (canonical physical) object. Therefore, the problem representation resulting from this stage (labeled *canonical representation* in the diagram) actually consists of a group of *views* that inherit all the properties of corresponding canonical physical objects.

Then, the canonical representation is again used as input for reasoning by the inference system, which guides APEX in building a *physical representation* of the problem in which the underlying physical principles and laws involved in the problem are explicitly represented. APEX's physical models can be divided into two subgroups based on their operational goals: those specialized for making inferences about implicit forces, and those specialized for writing equations. Accordingly, this stage of problem solving is performed in two distinct steps; the first step is to select and instantiate physical models involving forces,

1041-4347/91/0600-0048$01.00

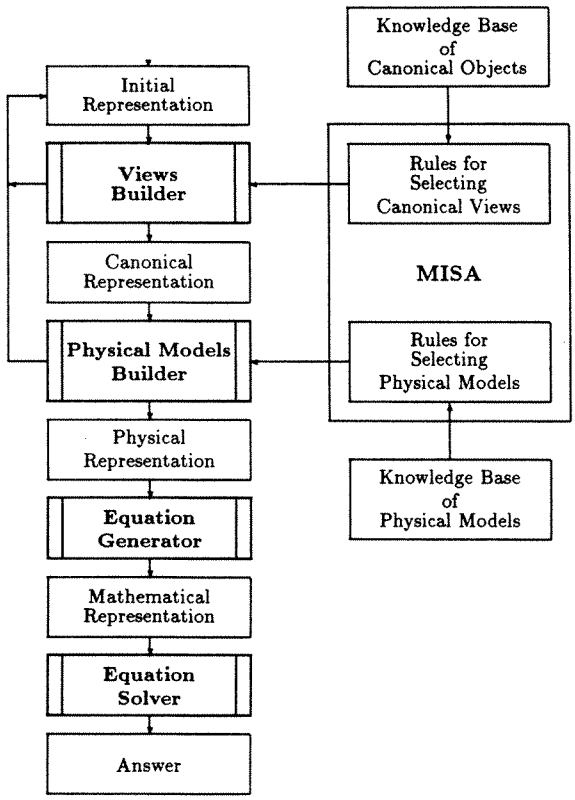

Fig. 1. Overall program organization.

and the second step is to select and instantiate physical models involving equations.

Next, a *mathematical representation* of the problem is formulated by collecting equations from the physical model instances in the physical representation. Finally, the equations, which may contain symbolic constants as well as numbers (with units) and variables, are sent to a symbolic manipulation package designed to solve simultaneous equations.

As outlined above, APEX solves a physics problem by a series of stepwise rerepresentation processes involving invocations of models to the initial problem representation. This process augments the previous problem representation rather than replacing it. Therefore, the final problem representation may be considered a representation of the problem with full specification of its underlying structures that were initially implicit. At each stage, APEX must be guided to select proper canonical objects or physical models. This guidance may be provided by a human, through a user-friendly interface, or by an expert system. MISA (*Mini Inference System for APEX*) is a small expert system that has been developed to demonstrate APEX in a limited number of problem-solving sessions without human intervention. MISA is comprised of a set of inference rules that a human physics expert might consider when selecting proper models.

III. Canonical Objects and Views

A. Canonical Objects

Canonical objects such as a point mass, a lever, or an ideal rope serve as the vocabulary for physics; problem solving requires interpreting the objects in the initial problem into these canonical objects. A group of canonical objects has been predefined in APEX's permanent knowledge base; each is an *object class* in an object-oriented programming environment. A canonical object class is defined with intrinsic attributes and methods for computing properties from the attributes. Fig. 2 shows the definitions for some canonical object classes.

```
class:       PtMass
attributes:  mass
superclass:  CanonicalObj

class:       Rope
attributes:  rotation
             tension
superclass:  CanonicalObj

class:       Circle
attributes:  radius
properties:  diameter ← 2·radius
             area ← π·radius²
             perimeter ← π·diameter
superclass:  CanonicalObj
```

Fig. 2. Example canonical object classes.

B. Views

We regard the process of abstracting an initial object as a canonical physical object as a data conversion process, accomplished by *views*. A view serves as a hybrid data structure facilitating bidirectional mapping of the features of the objects in the two representations. *Viewing* an input object as a canonical object is achieved by constructing a collection of network elements within the object-oriented programming environment of APEX. The network consists of five kinds of data elements:

object class: The real-world object class.

object instance: An instance of the *object class* defined with a set of property–value pairs.

canonical object class: The target canonical object class that the *object instance* is modeled as.

view class: A hybrid data object class that makes the *object class* look like the *canonical object*. It is defined with a set of methods for deriving the attributes of the target canonical object class in terms of the properties of the *object instance*.

view instance: An instance of the *view class* created for viewing the *object instance*.

To illustrate view networks, consider the following problem ([10, p. 114, #27]).

A car is moving on a circular racetrack with a curvature-radius of 60 m. If the banking of the racetrack is 30 degrees from the horizontal, what is the uniform speed of the car to keep it moving in the circle without friction?

In solving the problem, MISA (Mini Inference System for APEX) suggests a circle view of the track, among other views. Fig. 3 shows the view network constructed by APEX for that effect. In the diagram, a real-world object instance, `Racetrack1`, is abstracted as a canonical object class, `Circle`. A new view class, `CircleView1`, is constructed with an `originalobj` attribute and a `radius` property. The mapping instruction for deriving the feature of the target canonical object, i.e., `radius`, is given by MISA as `(GETV Racetrack1 curvature-radius)`.[1] The mapping instruction is

[1] `(GETV <obj> <slot>)` retrieves the value of `<slot>` from `<obj>`.

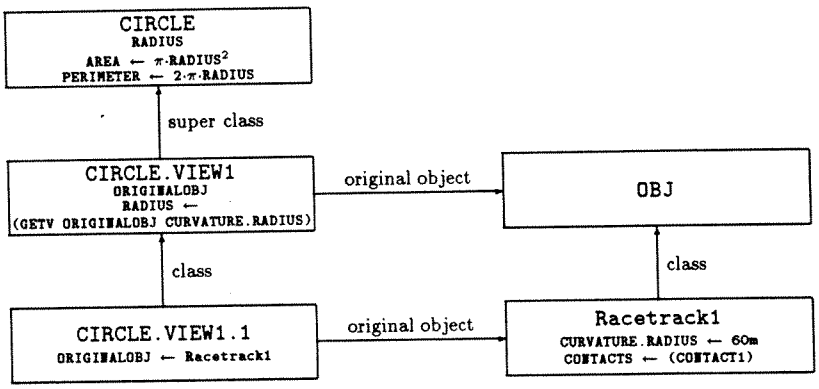

Fig. 3. An example view network structure.

```
class:       CircleView1
attributes:  originalobj
             viewdescription
properties:  radius ← (GETV originalobj curvature-radius)
superclass:  Circle
             View
```

Fig. 4. Data structure for `CircleView1`.

then stored, in a more general form, `(GETV originalobj curvature-radius)`, in the `CircleView1` class as the method for computing its `radius` property. The internal data structure for the `CircleView1` class is shown in Fig. 4. Finally, an instance of the view class, `CircleView1.1`, is created with its `originalobj` attribute defined as `Racetrack1`.

The resulting view instance, once constructed, enables APEX to treat it as if it were a "real" `Circle`, i.e., properties of a `Circle` (e.g., `area`, `perimeter`) are directly accessible to the view instance thanks to the property inheritance mechanism of the object-oriented programming environment. In APEX, the final physical representation of a problem is obtained in terms of a collection of such view instances.

C. Bidirectionality of Data Conversion

Physics problems are often incompletely specified. Such underspecification is effectively dealt with by the proposed representational framework. APEX's canonical representations provide *bidirectional* data conversion between the initial and canonical representations of a problem. In the forward direction (from initial to canonical, and to physical representation), the derivation methods are stored only in symbolic forms specifying where and how to derive data. Actual values are not retrieved until after building the final physical representation of the problem, at which point values are retrieved by *evaluating* these forms in order to write equations involving only constants and variables. In the backward direction, intermediate results obtained from the canonical (or physical) representation are transmitted into the missing slots of the original representation, eventually resulting in full specification of the problem. An example of backward data propagation would be when external forces acting on the car (viewed as a point mass) in the previous example are considered. In order to derive the gravitational force exerted on the car, it is necessary to consider the gravitational field in which the car resides. For this, APEX decides to create a field instance with default gravitational constant; the newly created instance is attached to the car in the initial representation.[2] In an analogous way, backward data propagations are also effectively used for specifying an object's structural attributes (e.g., the tension of a rope), or even attributes of an environment (e.g., the acceleration of a motion).

D. Multiple views

[20] suggested the importance of multiple representations in solving physics problems. In the previous section, we discussed viewing a single object as a single canonical object. As problem complexity increases, however, there are cases in which a one-to-one view strategy alone is not adequate. This occurs when an object plays multiple roles in the problem; for example, in the racetrack problem, the track's geometry must be abstracted as a canonical circle and viewed as a canonical incline as well. This type of multiple view (one-to-many view) is handled in APEX without any difficulty, since once a canonical view representation is constructed, its internal data connections to initial objects are transparent to the problem solver.

Another type of multiple view is viewing many objects as a single canonical object. Representation of many-to-one views requires additions to the view framework we have presented. We will describe the extensions to the proposed view framework by considering the following problem [15].

> What constant horizontal force F must be applied to the large cart in Fig. 5 (of mass M) so that the smaller carts (masses m_1 and m_2) do not move relative to the large cart? Neglect friction.

In solving this problem, the problem solver must take a single point mass view of all objects in motion (three carts, pulley, and string) as a preliminary step to writing a force equation about the total mass with respect to the external force (F). When advised by the inference system to view these objects as a point mass, APEX's canonical representation builder first recognizes

[2] A field is an *environment* of an object. In APEX, there are four kinds of environments: field, contact, motion, and external force.

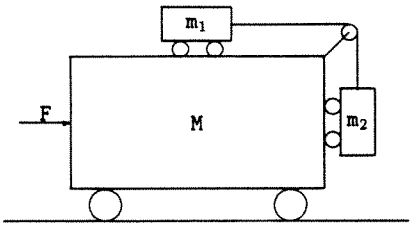

Fig. 5. Sample problem.

this as advice to build a many-to-one view network and makes an instance of `CompositeObj` class in the current context. `CompositeObj` is a special object class that is designed to bridge between a target canonical view and multiple objects in the initial representation. An instance of the `CompositeObj` class is essentially an encapsulation of multiple objects, with pointers to those objects involved in the view. Once such an instance is set up, it is treated by APEX as a single object instance for the later stages of problem solving. Fig. 6 shows the problem representation after a multiple view is taken (the initial problem representation is shown in the dashed box). Fig. 7 shows the data structures for the added nodes.

IV. Physical Models

A. Representation of Physical Principles

A physics textbook presents a set of physical principles. Expertise in problem solving requires the ability to apply a proper set of physical principles, in order to obtain a set of equations relevant to the problem. APEX employs *physical models* as the data structures for modeling principles of physics. A physical model is an encapsulation of a single principle or law of physics, stored in a permanent knowledge base of APEX. Whereas a canonical physical object is an abstraction of one or more objects, a physical model is a physical interpretation of a situation involving objects and their *relationships*. Each physical model contains a chunk of physical principle such as:

- A point mass placed on an incline without friction is subjected to a force exerted in a direction normal from the incline.
- The net force exerted on a point mass (of mass m) moving in a circular path (of radius r) with uniform speed (v) equals mv^2/r.

B. Contents of Physical Models

Physical models are implemented as a set of object classes in an object-oriented programming environment. The following is the description of the contents of a physical model.

class name: The name of a physical model.
attributes:
  components: A list of (canonical physical) objects and their environments that the physical model is about.
  sysvar: An optional system variable whose value influences subsequent computation of properties.
properties:
  locals: Each of the local properties defines the computation of an intermediate quantity.
  OUT: Every physical model has an `OUT` property whose method reflects the net effect of instantiating a physical model.

When the inference system (e.g., MISA) examines the canonical representation of a problem for selecting physical models, the *components* of physical models are tested for match with a certain group of views and their environments. If a physical model involving a *sysvar* (e.g., a coordinate system) is selected, the inference system first computes it by invoking a library procedure specialized for this task.

After matching contexts (and *sysvar,* if any) are passed to programs for instantiating the selected physical models, these programs first construct an object class called a *model class* by copying from the attributes of the selected physical model with an additional attribute for *model description.* After making the new model class a subclass of the selected physical model, the model-building programs instantiate it in the given context. The *locals* of the physical model (which is now a super class of the model class) are computed in a predefined order using the methods given in the *locals* property definitions; this may involve inference of unspecified data. Suppose, for instance, that the physical model in Fig. 8 has been selected for instantiation, and that the `obj.` in question is in contact (`Contact1`) with the `incline` in question. If the normal force acting at `Contact1` is initially unknown, the effect of computing

```
(GETV contact normalforce
   (CREATE Vector WITH dir=(normalto in-
     cline)))
```

is to create a vector as prescribed by the (`CREATE` . . .) procedure (i.e., with direction = (`normalto incline`) and magnitude = a symbolic constant), and to store it as the value of the `normalforce` property of `Contact1`.

The last property of a physical model is always an `OUT` property. Computation of an `OUT` property results in generating either a set of equations (through an `EQNADD` operation) or a new environment for an object (through an `ENVADD` operation). When the `OUT` property is an `EQNADD` operation, the equations are stored in a global equation set. An example use of an `ENVADD` operation is given in Fig. 8; instantiation of this model will result in attaching the force (an environment) to the `obj.` Hence, the net effect of selecting and instantiating a physical model is the formulation of a solution model which either explicates an implicit environment of an object or writes one or more constraint equations about an underlying structure of the problem. In APEX, a problem is finally solved by manipulating a set of constraint equations obtained from a selected set of physical model instances.

V. Inference

Selecting proper physical models requires what is sometimes called *physical intuition* [24]; the problem solver, human or machine, may not rely merely on the surface features of the problem statement but often is required to reason about the underlying structures and principles of the given system [5]. In APEX, such expertise is captured in a collection of inference rules, MISA (*Mini Inference System for APEX*), that reasons about the features and the contexts (e.g., environments) of objects in order to guide APEX in selecting views and physical models. MISA is a relatively small-scale rule base designed to work on a small number of selected physics problems for the purpose of demonstrating the ideas of APEX's representation [11].

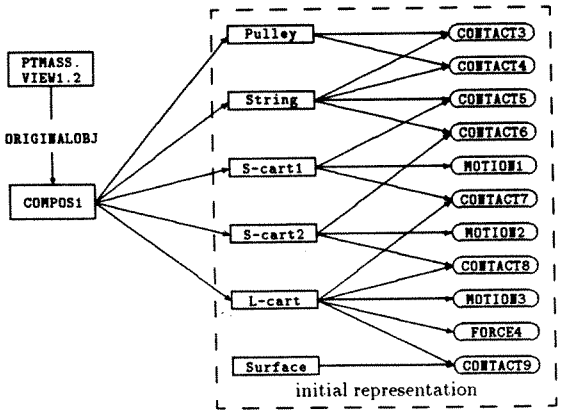

Fig. 6. Multiple view.

```
instance:    PtMassView1.2
attributes:  originalobj ← CompositeObj1
             viewdescription ← ViewDesc1
properties:  mass ← (GETV originalobj mass)
superclass:  PtMass
             View

instance:    CompositeObj1
attributes:  fromobjs ← (S-cart1 S-cart2 L-cart Pulley String)
properties:  mass ← M + m1 + m2
```

Fig. 7. Data structures for the elements of multiple view.

```
class:       NormalforceModel
components:  obj
             incline
property:    contact ← (CONTACTBETWEEN obj incline)
             loc ← (LOCATIONOF obj contact)
             force.vect ←
               (GETV contact normalforce
                     (CREATE Vector WITH
                       dir = (normalto incline)))
             force.env ← (CREATE ExtForce WITH
                                 force = force.vect
                                 obj = obj
                                 loc = loc
                                 agent = incline)
             OUT ← (ENVADD obj force.env)
superclass:  PhysicalModel
```

Fig. 8. Physical model class `NormalforceModel`.

In addition to automatic selection of models, a user-friendly interface to APEX has been developed to allow manual selection. When APEX is running in human-guided inference mode, MISA is turned off, and the representation choices for a problem are made according to instructions from the human user. The interface package also provides graphical presentations of the current state of APEX's internal representation of the problem, in which a graph node (e.g., a view or a physical model instance) may be further inspected in a menu-driven way. Fig. 9 shows the screen image while solving the racetrack problem.

As APEX solves a problem, the representation does not explicitly reflect the system's inference history that brought about the present state of problem representation. APEX employs *view descriptions* and *model descriptions* as data structures for keeping track of the system's inference history. View and model descriptions are defined as object classes, which are instantiated for specific *viewing* or *modeling* operations and attached to the resulting view or physical model. They record what rule has been applied to what kind of situations, and what initially-unspecified quantities have been derived and how. During or after solving a problem, they allow APEX to answer questions about the system's reasoning behavior. For instance, a user might want to ask the system why a specific view has been taken; through the menu-driven graphic interface, the user can obtain the answer in the form of a rule, instantiated in its matching context and translated into English.

VI. Conclusion

This paper presents an advanced representation mechanism for physics problems. By isolating representation from inference, it allows the problem solver to choose which representations to use; these choices can be made by the human user or by an expert system. Accomplishments of this research include development of formal representations for the *conceptual entities* [9] and relations of physics. The proposed representation separates *views* from actual objects in a way that facilitates *invertibility* of the view and multiple representations.

In this scheme, the internal representation of a problem is obtained as a collection of physical models which are constructed from the views and which are, in turn, connected to the informal representations of the original objects. Due to the transparency of the linkage between the physical models and the original objects, the physical models are made independent of the representations of the actual objects, which might contain a variety of informal

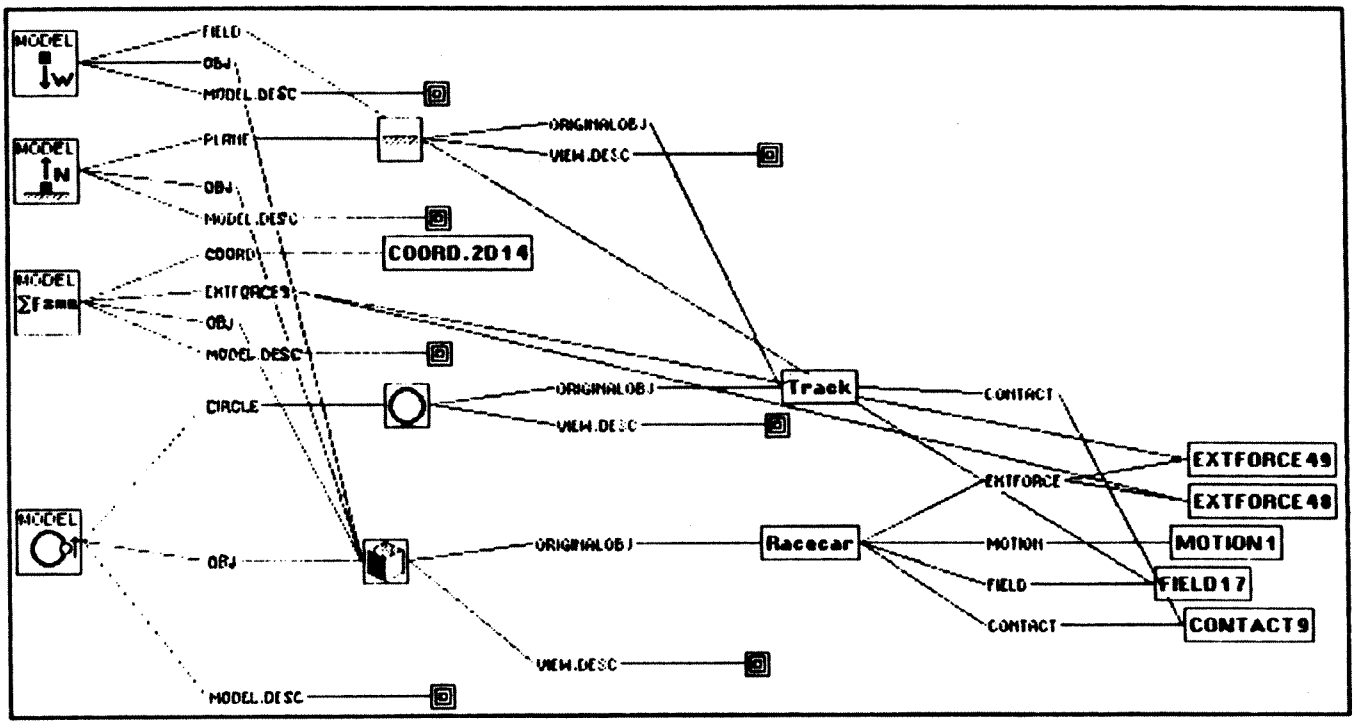

Fig. 9. Screen image while solving the racetrack problem.

features and relations of the real world. This scheme achieves an important goal of a computational model for solving physics problems: how can a *finite* set of physical principles suffice to solve an *infinite* set of problems?

The APEX system has been tested on five physics problems; details are given in [11]. We believe that the representational framework of APEX would be sufficient for a variety of physics problems and for engineering domains in which problem solving is based on interaction of discrete canonical models of real-world objects; it would not be suitable for problems involving, e.g., fields that vary over space or time and that could not be modeled as discrete interactions among objects. The MISA expert system used to choose the models to be used for a given problem is a small one, and it would need considerable expansion to handle a wide variety of problems; this is an area of current research.

The representational methodologies and system design principles of this research can be useful for knowledge engineering in many application areas. As knowledge-based systems become larger and more widespread, the problem of reuse of knowledge becomes critical: How can general knowledge developed for one application be used for a different application, which may involve different representations for the objects? The *view* mechanism we describe allows the separation of the canonical view taken in coding the general principles from the representations of the objects to which those principles are applied. Because the views are bidirectional, it is possible not only to derive properties of objects as needed for reasoning, but to return the results of the reasoning to the original objects in the appropriate form. Views can therefore allow the development of expert knowledge to be decoupled from the applications to which it is applied.

References

[1] S. Addanki, R. Cremoni, and J. S. Penberthy, "Reasoning about assumptions in graphs of models," in *Proc. IJCAI-89,* pp. 1432–1438.

[2] D. Bobrow, Ed., *Qualitative Reasoning About Physical Systems.* Cambridge, MA: MIT Press, 1985.

[3] W. Bulko, "Understanding text with an accompanying diagram," in *Proc. First Int. Conf. Industrial Eng. Appl. AI Expert Syst.* Tullahoma, TN, 1988, pp. 894–898.

[4] ——— "Understanding coreference in a system for solving physics word problems," Ph.D. dissertation, available as AI-89-102, CS Dep., Univ. of Texas at Austin, 1989.

[5] M. Chi, P. Feltovich, and R. Glaser. "Categorization and representation of physics problems by experts and novices," *Cognitive Sci.,* vol. 5, pp. 121–152, 1981.

[6] J. de Kleer and J. S. Brown, "A qualitative physics based on confluences," in *Qualitative Reasoning About Physical Systems,* Cambridge, MA: MIT Press, 1985, pp. 7–84.

[7] K. Forbus, "Qualitative process theory," in *Qualitative Reasoning About Physical Systems.* Cambridge, MA: MIT Press, 1985, pp. 85–168.

[8] D. Gentner and A. Stevens, Eds., *Mental Models.* Hillsdale, N.J: Lawrence Erlbaum, 1983.

[9] J. Greeno, "Conceptual entities," in *Mental Models.* Hillsdale, NJ: Lawrence Erlbaum, 1983.

[10] D. Halliday and R. Resnick, *Physics.* New York: Wiley, 1978.

[11] H. J. Kook, "A model-based representational framework for expert physics problem solving," Ph.D. dissertation, Tech. Rep. AI-89-103, Comput. Sci. Dep., Univ. of Texas at Austin, 1989.

[12] B. Kuipers, "Commonsense reasoning about causality: deriving behavior from structure," in *Qualitative Reasoning About Physical Systems.* Cambridge, MA: MIT Press, pp. 169–204.

[13] J. Larkin, J. McDermott, D. Simon, and H. Simon, "Expert and novice performance in solving physics problems problems," *Science,* vol. 208, pp. 1335–1342, 20 June 1980.

[14] ——— "Models of competence in solving physics problems," *Cognitive Sci.* vol. 4, pp. 317–345, 1980.

[15] J. Larkin, "The role of problem representation in physics," in *Mental Models.* Hillsdale, NJ: Lawrence Erlbaum, 1983.

[16] M. Lowry, "The abstraction/implementation model of problem reformulation," in *Proc. IJCAI-87,* pp. 1004–1010.

[17] S. Murthy and S. Addanki, "PROMPT: An innovative design tool," in *Proc. AAAI-87,* pp. 637–642.

[18] G. Novak, "Computer understanding of physics problems stated in natural language," *Amer. J. Computational Linguistics,* Microfiche 53, 1976. Also available as TR-NL-30, CS Dep., Univ. of Texas at Austin, 1976.

[19] ——— "Representations of knowledge in a program for solving physics problems," in *Proc. IJCAI-77,* 1977, pp. 286–291.
[20] G. Novak and A. Araya, "Physics problem solving using multiple views," TR-173, CS Dep., Univ. of Texas at Austin, 1981.
[21] G. Novak, "GLISP: A LISP-based programming system with data abstraction," *AI Mag.,* pp. 37–47, Fall, 1983.
[22] G. Novak and W. Bulko, "Understanding natural language with diagrams," in *Proc. AAAI-90,* pp. 465–470.
[23] R. Siegler, Ed., *Children's Thinking: What Develops?* Hillsdale, NJ: Lawrence Erlbaum Associates, 1978.
[24] D. Simon and H. Simon, "Individual differences in solving physics problems," in *Children's Thinking: What Develops?.* Hillsdale, NJ: Lawrence Erlbaum, 1978.

Hyung Joon Kook (S'87–M'89) is Assistant Professor at the Department of Computer Science, King Sejong University, Seoul, Korea. The APEX system was the subject of his doctoral dissertation at the University of Texas at Austin.

Gordon S. Novak, Jr. (S'75–M'76) is Associate Professor and Director of the Artificial Intelligence Laboratory in the Department of Computer Sciences, University of Texas at Austin. He is the author of the ISAAC program, which solves physics problems stated in English, and the GLISP programming language and compiler. His current research interests include physics problem solving and automatic programming based on reuse of generic algorithms. He is editor and co-author of a videotape course, *Introduction to Artificial Intelligence and Expert Systems* (San Mateo, CA: Morgan Kaufmann, 1988).