# A TRIDENT SCHOLAR PROJECT REPORT

NO. 256

**"Development of Angular Motion, Angular Momentum, and Torque Knowledge Bases for an Intelligent Physics Tutoring System"**

19990121 139


UNITED STATES NAVAL ACADEMY
ANNAPOLIS, MARYLAND

This document has been approved for public release and sale; its distribution is unlimited.

Reproduced From
Best Available Copy

# REPORT DOCUMENTATION PAGE

Form Approved
OMB no. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspects of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson

| 1. AGENCY USE ONLY (Leave blank) | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|

4. TITLE AND SUBTITLE "Development of angular motion, angular momentum, and torque knowledge bases for an intelligent physics tutoring system"

5. FUNDING NUMBERS

6. AUTHOR(S)
Michael Damon Eason

7 PERFORMING ORGANIZATIONS NAME(S) AND ADDRESS(ES)
U.S. Naval Academy, Annapolis, MD

8. PERFORMING ORGANIZATION REPORT NUMBER USNA Trident report; no. 256 (1998)

9 SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10 SPONSORING/MONITORING AGENCY REPORT NUMBER

11 SUPPLEMENTARY NOTES
Accepted by the U.S. Trident Scholar Committee

12a DISTRIBUTION/AVAILABILITY STATEMENT
This document has been approved for public release: its distribution is UNLIMITED.

12b DISTRIBUTION CODE

13 ABSTRACT (Maximum 200 words) This research involved the design and development of a physics knowledge base which allows an intelligent tutoring system, ANDES, to more effectively assist students in physics problem-solving. ANDES is a Office of Naval Research funded collaborative effort between the U.S. Naval Academy and the Learning Research and Development Center at the University of Pittsburgh. The system tutors Newtonian physics via coached problem-solving, a method of teaching cognitive skills where the tutor and the student work together to solve problems. The knowledge base developed in this project provides the physics backbone for the rest of the tutoring system by generating the necessary equations and solution graphs to solve selected angular motion physics problems. These mathematical outputs are used by other ANDES components to provide help to the students in a variety of ways. In order for ANDES to be an effective tutoring system, the knowledge base developed had to fulfill certain criteria. It needed to model a teacher's approach to problem-solving using planning and decision-making strategies to find the solution path efficiently. It also had to be robust enough to generate several solution paths for problems that have more than one possible solution method. Since this research was the first attempt to develop a knowledge base for the angular motion area of physics, encoding these concepts presented unique challenges. For example, the concept of normal force which is easily understood in the classic "object on top of a surface problem" becomes more difficult in the classic "roller coaster at the top of a loop" problem. However, both visualizations had to be modeled by the same implementation. The knowledge base designed and developed in this project successfully accomplished this modeling and is presently being incorporated into the larger ANDES project.

14 SUBJECT TERMS
Intelligent tutoring systems, Cognitive modeling, Circular motion, Angular motion, Torque

15. NUMBER OF PAGES

16. PRICE CODE

| 17 SECURITY CLASSIFICATION OF REPORT | 18 SECURITY CLASSIFICATION OF THIS PAGE | 19 SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATATION OF ABSTRACT |
|---|---|---|---|

U.S.N.A. --- Trident Scholar project report; no. 256 (1998)

# "Development of Angular Motion, Angular Momentum, and Torque Knowledge Bases for an Intelligent Physics Tutoring System"

by

Midshipman Michael Damon Eason, Class of 1998
United States Naval Academy
Annapolis, Maryland

Damon Eason
(signature)

Certification of Advisor Approval

Associate Professor Kay G. Schulze
Computer Science Department

Kay G. Schulze
(signature)

1 May 1998
(date)

Accepted for Trident Scholar Committee

Professor Joyce E. Shade
Chair, Trident Scholar Committee

J E Shade
(signature)

8 May 1998
(date)

## Abstract

This research involved the design and development of a physics knowledge base which allows an intelligent tutoring system, ANDES[1], to more effectively assist students in physics problem-solving.

ANDES is an Office of Naval Research funded collaborative effort between the U. S. Naval Academy and the Learning Research and Development Center at the University of Pittsburgh. The system tutors Newtonian physics via coached problem-solving, a method of teaching cognitive skills where the tutor and the student work together to solve problems.

The knowledge base developed in this project provides the physics backbone for the rest of the tutoring system by generating the necessary equations and solution graphs to solve selected angular motion physics problems. These mathematical outputs are used by other ANDES components to provide help to the students in a variety of ways. In order for ANDES to be an effective tutoring system, the knowledge base developed had to fulfill certain criteria. It needed to model a teacher's approach to problem-solving using planning and decision-making strategies to find the solution path efficiently. It also had to be robust enough to generate several solution paths for problems that have more than one possible solution method.

Since this research was the first attempt to develop a knowledge base for the angular motion area of physics, encoding these concepts presented unique challenges. For example, the concept of normal force which is easily understood in the classic "object on top of a surface problem" becomes more difficult in the classic "roller coaster at the top of a loop" problem. However, both visualizations had to be modeled by the same implementation. The knowledge base designed and developed in this project successfully accomplished this modeling and is presently being incorporated into the larger ANDES project.

Keywords: intelligent tutoring systems, cognitive modeling, circular motion, angular motion, torque

---

1

This research is supported at the U.S. Naval Academy by the Office of Naval Research (ONR) Cognitive Sciences Division Grant No. N00014-98-WR30124. It is supported at the University of Pittsburgh by Advanced Research Projects Agency (ARPA) Computer Aided Education and Training Initiative (CAETI), Grant N660001-95-C-8367, Office of Naval Research (ONR) Cognitive Sciences Division, Grant N00014-96-1-0260, and Air Force Office of Scientific Research (AFOSR) Artificial Intelligence Division, Grant F49620-96-1-0180.

DTIC QUALITY INSPECTED 3

## VIII. Acknowledgments

I would like to thank Dr. Kay Schulze first and foremost. As my faculty advisor, she guided me and helped tremendously from the outset of this project through to its completion. Dr. Schulze encouraged me every step of the way, and presented me with challenges that helped me to grow both professionally and personally. And she spent countless hours reading my papers, constructing slides for presentations, and doing so many other things that made this research project possible.

But Dr. Schulze also went far beyond what is expected of any advisor. She took the time to listen when family illnesses pulled me from my work and shared her experiences of dealing with the serious illness of a loved one. She opened her home when I had visitors in town. She showed me friendship as well as leadership, and set an example with her work ethic, patience, and positive attitude that I hope to follow.

Everyone on the ANDES team made sacrifices to contribute to this research project. Dr. Shelby and Dr. Wintersgill, as physics expert advisors, consistently made themselves available to resolve design issues, explain physics topics, check equations, and help out in many ways. Both also went to great length to ensure I felt welcome as a new member of the team. And Dr. Shelby even gave up class time on several occasions to allow me to conduct the classroom help evaluation. Members of the ANDES team at the University of Pittsburgh made sure that a demonstration version of ANDES was available both for the SIGCSE '98 conference in Atlanta, GA, and for the Trident Scholar Conference. All deserve recognition and my sincere thanks.

## Table of Contents

## I. Literature Review

The computer has significant potential for use as a tool to aid in learning. It not only has great number-crunching ability, but by virtue of the way that a process is represented in a computer's memory, it can naturally keep track of the actions used to carry out a task (Collins and Brown 1990). If the computer can encode the problem-solving process of an expert in a given field, it can compare a student's progress towards finding a solution to the expert's problem-solving method in generating the same solution. By showing a student a comparison between his approach to a problem and that of an expert's problem-solving method, the computer can tutor students in almost any field. This type of tutoring is frequently called model tracing tutoring.

Model tracing tutoring, pioneered by John Anderson and his colleagues at Carnegie-Mellon University, is one of the best existing approaches to intelligent tutoring (Anderson, et al. 1995). In its simplest form, a model tracing tutor contains a cognitive model that is capable of correctly solving any problem assigned to the student. The model arrives at the correct answer using only a correct sequence of steps. Moreover, the tutor plans the solution prior to actually solving the problem.

Model tracing tutors employ a technique known as coached problem-solving, where a tutor and a student solve problems together. At times, the coach simply watches in support as the student solves the problem, indicating at each step whether the student is correct or not. For difficult sections of the problem, however, the coach leads the student through the problem step-by-step, using the same methods the student should use on his own. This is known as modeling the cognitive processes that the student is trying to learn. Throughout this process, the tutor and student remain focused on the overall goal of solving the problem.

In a cognitive model, being able to represent the expert's approach to solving a problem in the subject matter is an essential part of a model tracing tutor. In order to be effective, the knowledge that the expert uses to find the solution must be available for the tutoring system to employ in a similar manner. If a computer uses its computational ability, rather than the concept-directed approach of a subject expert, such a process trace would be useless to a student in learning how to solve the problem himself. This type of approach,

known as a "black box" methodology, is useful in other situations, where keeping track of the steps taken to find the solution is unimportant. A computer can take advantage of methods that are potentially more efficient than the human problem-solving method. For a computer tutor, however, the path to a solution is at least as important as the solution itself, and so it employs a "glass box" methodology. The reasoning and conceptual explanation behind each step in the problem's solution can be explained to the student.

A knowledge base is the set of representations that the computer uses to construct a cognitive model of the expert's approach, generally rules concerning facts about the task domain and plans on how to obtain specific goals. These rules generally take the form of an "if-then" statement; if certain conditions are true in the problem space, then either more facts are known about the problem or more goals along the solution path can be set. These rules in the knowledge base can be divided into three basic categories. Rules in the first category are those that cannot be proven by experiments or logic, but are simply arbitrary rules. An example rule of this type would be, 'If you want to write a C program, set the goal of entering "int main (void)." This is simply the correct syntax for a C program; no other explanation is necessary or possible. Rules of the second type are true by assumption; examples from an algebraic equation solving tutor would include rules for the distributive law or the commutative law that are based on fundamental postulates of mathematics (Schulze 1989). Tutors up to this point have primarily used these two type of rules. Rules that are based on scientific knowledge and have a conceptual explanation make up the final category. An example from a physics knowledge base would be, "If an object is resting on a surface, that surface exerts a normal force on the object." The truth of these rules is not always readily apparent to the students. The task domain being modeled is the main source for determining the type of rules contained in the knowledge base. For example, a C tutor's knowledge base would consist mainly of arbitrary rules that deal with the correct syntax of a C program.

Because a tutor's main goal is to help the student work through the given problem, the system's knowledge base is concerned primarily with procedural knowledge and facts that represent a cognitive skill. A cognitive skill is primarily comprised of units of goal-related

knowledge. Cognitive skill acquisition is the formulation of "production rules" relating task goals to actions and consequences. These production rules have several defining characteristics. They are procedural in that they set forth an action to be carried out as part of task completion. The rules are abstract and applicable to many different situations; they are problem-independent. These production rules are also goal-related, directed so that they fit into an overall strategy for solving problems. These production rules are collectively known as the procedural knowledge of a task domain.

Declarative knowledge (i.e., knowing the concepts of a subject matter) should be distinguished from procedural knowledge (i.e., the ability to use the concepts of a subject to solve problems). Declarative knowledge is goal-independent. Developing cognitive skills involves converting declared knowledge into production rules. It consists of making the leap, for example, from knowing the Pythagorean theorem in geometry, to using that theorem to find the length of a right triangle's side. The applications and possible uses of the declarative knowledge in a task domain are often numerous and varied. If all domain knowledge were represented through procedural rules, each piece of knowledge would have to be represented separately in each different way that it could be used. Similarly, if all knowledge were encoded in declarative structures, interpreting everything would be very inefficient and place an immense burden on the working memory of an intelligent tutoring system (ITS). An efficient knowledge base should be balanced between declarative and procedural knowledge.

The process of building a knowledge base is called knowledge engineering. An effective knowledge engineer should be trained in the representation of rules for implementation of the knowledge base (whether it be English rules, LISP statements, etc.) The knowledge engineer must also comprehend enough of the task domain to represent the important objects and relationships. He need not be an expert in the subject matter. However, the knowledge engineer will usually interview domain experts to learn about the domain and to extract the required knowledge. This process may be necessary for the creation of the formal representations of the domain knowledge.

The knowledge engineering process can be divided into three main steps. First, the decision must be made as to which facts and objects need to be covered in the knowledge base and which can be ignored. If building a knowledge base about cars, one might want to pay attention to such attributes as car weight and engine size, but ignore price and manufacturer. Next, the knowledge engineer must decide on a vocabulary of constants, variables, functions, and predicates for his production rules. This translates important domain-level concepts into names used in the knowledge base implementation. In so doing, one creates the templates that determine how information about objects is encoded; this is known as the ontology of the domain. The concept of a car door, for example, might be represented as having attributes such as color, weight, number of hinges, and so on. A pick-up truck door might have different attributes, but would still be represented by the same basic concept as a car door. The tailgate of a truck, however, may or may not be able to be represented using the same attributes as a door. These decisions affect the functionality of the knowledge base and the relationships that inherently exist between domain's concepts. Having a good ontology, or method of expression of facts about the domain, generally means that few general rules need to be specified. This ontology must be sufficient to differentiate between the basic concepts of a task domain, but not so rigid as to distinguish between ideas that need not be separated. A physics knowledge base would probably have different templates for scalars and vectors, for example, but a fundamental distinction between a force vector and a velocity vector would most likely be extraneous.

Once this framework has been determined, the knowledge engineer can move on to the third step: encoding general knowledge about the domain. This involves formalizing the interrelationships between objects and declaring the factual knowledge of the domain. An effective knowledge base contains statements that are logically self-contained and independent of context; this contrasts with the functionality of a normal programming statement, which depends heavily on its context. For these reasons, a knowledge base's "code" does not resemble that of a typical program. The order of statements/rules is irrelevant; a rule that sets a goal or asserts a fact in the problem space relies on its preconditions, not on its physical location in the code, to know when to execute. With a

knowledge base constructed such that it can solve the necessary problems of the task domain, one can now look at how the ITS uses this expert cognitive model to help students learn.

Cognitive skill acquisition consists of three phases. It starts with a beginning period during which students acquire initial knowledge of the skill, usually by reading descriptions of it. They do not actually try to solve problems during this relatively short stage. The middle phase consists of learning how to solve problems using the knowledge encountered in the first phase. This application phase often lasts a long time. Students frequently need considerable help at the beginning of this stage, and refer to the textbook, examples, teachers, peers, or other sources of information about the skill. By the end of this phase, they can solve most problems without help. Occasionally, a third phase occurs wherein students who can already perform the task continue to practice it until it becomes nearly automatic. During this phase, their speed and accuracy continue to increase, but show diminishing returns as time passes (Siegler and Jenkins 1989).

Coached problem-solving is focused on the intermediate phase of cognitive skill acquisition, where the student acquires predominantly procedural knowledge. This is because the learning of declarative knowledge, phase one, is generally assumed to be problem-free. This acquisition of declarative knowledge occurs in situations where students learn, either by listening to a lecture, watching a video, reading a book, or similar activities. By itself, though, this declarative knowledge is considered inert and of limited use. Student must also learn how to use this knowledge to solve problems. Design of computer tutoring systems is based on the assumption that students are acquiring declarative knowledge from other sources; the tutors provide an environment for exercising this knowledge and acquiring and strengthening procedural knowledge.

VanLehn (1995a) theorizes that an ITS stimulates learning by triggering "rule learning events", which interrupt the main process. In his study, the main process was either example-studying or problem-solving. During a rule learning event, a student formulates a new rule about the subject matter or revises a previously existing rule. These events are frequently the result of impasses, where a student's current level of knowledge is insufficient

to get him to the next step in the problem solution, or some other manifestation of incomplete or incorrect knowledge.

These impasses, the events in which a problem solver gets stuck, occur because he has no operator for achieving the current goal or no means of selecting between several applicable operators (Newell 1990). For example, one student believed that mass and weight were the same thing, but was unwilling to make this inference on a problem that gave the students the mass of a space traveler and asked them to calculate her weight on various planets. If she applied her "mass-is-weight" rule to this problem, all the answers would be the same as the given mass. The student refused to believe a problem could have such a trivial solution, so she reached an impasse and sought another relationship between weight and mass. In another example, a student drew a force diagram of a stationary block, but incorrectly left out the normal force between the table and the block. The conflict between a stationary block (i.e., no acceleration) and unbalanced forces placed the student at an impasse, and he proceeded only after reading the textbook, learning the "normal force" rule, and correcting his force diagram (VanLehn 1995). These impasses take many different forms, but create similar situations for the student in which their current knowledge does not allow them to solve the problem at hand.

An ITS on a subject matter such as physics contains a significant number of concept-based rules and should be able to explain the concepts behind its rules in order to be a more effective tutor. Because the coached problem-solving approach limits the activity to solving problems, providing help with the non-arbitrary rules of a subject matter goes beyond the capabilities of a model tracing tutor. This is similar to human tutors, even in procedural task domains such as LISP coding, who interrupt coached problem-solving with discussions about the concepts of the subject matter (Anderson, et. al. 1985).

One way to eliminate misunderstandings from a student's representation of the subject matter is to provide qualitative problems as well as quantitative ones. In the example of a physics tutor, a student could conceivably be successful in a physics course by having a strong background in algebra and memorizing the necessary physics equations. Such a

student would retain many of the misconceptions about physics with which he entered the course.

The ITS must also decide what type of feedback to give the student and when to do so. Model-tracing tutors are capable of providing immediate feedback. As the student solves the problem step-by-step, the tutor can agree and tell the student to proceed if his input matches the expert model, or tell the student to try again if he has made an incorrect input. There is evidence that immediate feedback of errors is very effective, because it is easy for the student to localize the mental step that led to the error and identify the discrepancy in his knowledge (Mitrovic and Djordjevic 1995). Used incorrectly, though, immediate feedback can become a crutch for the student's problem-solving ability. There is evidence that a certain load must be placed on the student in order for learning to occur and too much help can interfere with learning (Stern et al. 1989).

One way to help determine when and how to provide feedback is to maintain a student model, keeping track of what he does and does not know about the task domain. Information useful to constructing a student cognitive model is available from a variety of sources. Explicit questionnaires about the student's previous experience with the subject matter, related courses taken, and so on, can provide some initial information. But the bulk of a model of the student's understanding and knowledge about the task domain is created and updated dynamically (Conati, Gertner, VanLehn, and Druzdel 1997). An ITS analyzes the steps that a student takes to solve a problem, the mistakes that he makes, and the help the student explicitly requests. All these provide clues about his knowledge concerning the task domain.

In an ITS, this student cognitive model is typically implemented using probabilities related to a knowledge network. In this structure, the domain of the network is a collection of topics, and each node represents some piece of conceptual or skill knowledge which the student should eventually learn. This network structure that represents the task domain is also useful in other ways beyond student modeling. For example, this type of knowledge representation is used in adaptive testing, which uses knowledge interdependencies to choose test items that most quickly differentiate between a student's mastery or non-mastery of the

selected topics. For an ITS, however, this domain representation is typically utilized only as part of a student modeler. In the knowledge network of interrelated topics, a factor which represents the probability that the student knows this piece of knowledge is bound to each topic. This is known as a belief or Bayesian network. Many interrelationships exist between these pieces; often a particular concept or skill is a prerequisite of a more advanced topic. For instance, a student must understand the concept of multiplication before moving on to study the topic of exponents. This is just one of the many interrelationships that typically exist between pieces of knowledge in a task domain. These relationships become important both for modeling student beliefs at a particular point in time and for determining how to adjust the probabilities of those beliefs over a time period. The adjustment of the probability at one node usually propagates through the network, causing an adjustment to the student knowledge probability in several other nodes. If a student successfully solves a problem involving exponents, this reinforces the belief that he understands exponents themselves, and also multiplication as a sub-skill.

This type of dynamic network models a "snapshot" of the student's knowledge. It does not keep track of the evolution itself, or a history of the development or acquisition of such knowledge. Such an analysis may be useful, but is not carried out by the student modeler in an ITS. Analyzing the learning trends of a group of students, for example, might highlight specific concepts in the task domain that deserve more attention in the curriculum and classroom. However, the usage of this historical data is beyond the scope of an ITS.

In ordinary belief networks, it is assumed that the properties of the external world modeled by the network are unchanging. Even though the student may gather information from the situation which causes the network to modify its measures of belief about items in that environment, those items remain either true or false. Such a belief network might be used to examine medical data on a hospital patient, for example, and based on its analyses generate a list of possible diseases or ailments that the patient might have. Belief networks used as student modelers are unique in that the domain they are modeling (i.e., the student's proficiency with and knowledge about the task domain's topics) changes over time.

Coached problem-solving pairs well with this type of student-modeling. Problem-solving consists of a sequential application of several different concepts and skills on the way to a solution. Coached problem-solving tracks the student's steps in the process, examining each one for correctness, rather than checking only his answer. In this way, the student modeler can more easily determine the reasoning that the student is applying to solve the problem. Looking only at a student's final response is not sufficient, since several different types of conceptual errors can result in the same incorrect answer. The coached problem solver, however, allows the belief network to make more fine-grained updates on its model of the student's knowledge.

The student model can also be used by the ITS to determine what type of help to give the student. The more proficient a student is at the skill being exercised the more subtle the tutor's hint. A good student may simply be instructed to re-examine his answer, while a student having a considerable amount of difficulty will be presented with a more obvious hint--to consult a specific theorem, for example. This is superior to requiring students to wade through several levels of hints before being presented with material that is appropriate to their level of knowledge (Lajoie 1993).

The tutor can also guide the student in practicing good study habits. Good students often use different techniques when studying, rather than just studying longer hours than their lower-scoring counterparts. An instance of such a technique occurs when a student explains an example in the textbook to himself, stepping through it and explaining each line of the solution to himself. Students who use examples in this way have been found to profit more from example studying than those who read through it in a cursory manner knowing that the example will always be available for reference if necessary (Chi et. al. 1989). VanLehn and colleagues analyzed another naturally occurring studying strategy related to example problem usage(VanLehn, et. al. 1992). When working through problems, some students recognize that the current problem is similar to a previous problem or example, but do not immediately refer to this precedent. Instead, they use this example only if they reach an impasse while working the current problem. These students learned more during problem-solving than

those who referred to the example as soon as they noticed its relevance and continually used it as they worked on the current problem.

An ITS can help foster good studying practices. To encourage such strategies as those mentioned above, the tutor would show its work through a step-by-step example, revealing to the student the reasoning and principles behind each step of the solution process. It would prevent the student from using the example as a crutch, constantly referring to it without trying to figure out what to do next. There is promising evidence that studying strategies such as these can be effectively taught (Bielaczyc, et. al. 1994). Ideally, learning strategies acquired while using an ITS will be applicable to areas beyond the current subject matter. These learned strategies should help the student learn how to learn, as well as instructing in the task domain. This process of acquiring better studying strategies is known as meta learning (VanLehn 1996b).

In conclusion, ITSs may be capable of more effective tutoring by extending the capabilities of current model tracing tutors. An ideal ITS would intersperse example studying and problem-solving, which more closely resembles the teaching methods of human tutors. It would provide focused feedback, based on the current problem, and on the student's level of proficiency. This ideal ITS would also help encourage conceptual and meta learning, rather than solely stimulating procedural knowledge acquisition. The ITS's knowledge base would be flexible enough to allow students to pursue any correct path, rather than restricting them to the one that the tutor recognizes. The ideal ITS would provide human teachers with another asset to help them instruct students.

## II. Overview of the ANDES System

ANDES is an intelligent tutoring system that teaches Newtonian physics via coached problem-solving (VanLehn, 1996b, Conati, et al., 1997, Conati, et. al., 1997, Gertner, et. al., in press, Schulze, et. al., 1998), a method of teaching cognitive skills where the tutor and the student work collaboratively to solve problems. In coached problem-solving, the initiative in the interactions between the student and the tutor changes as progress is being made. The tutor simply indicates agreement as long as the student is proceeding along a correct solution path. When the student stumbles with a portion of the problem, the tutor provides tailored hints to lead the student back in the correct direction. In this setting, the critical issue for the tutor is interpreting the student's actions and the line of reasoning the student is using. To perform this task the tutor needs a model of the problem-solving process the student is attempting to produce.

ANDES has a modular architecture composed of a graphical Workbench with which the student solves physics problems, an Assessor that maintains a probabilistic model of the student's knowledge and goals over time, an Action Interpreter that provides immediate feedback to student actions, a Help System that provides procedural and conceptual help, and a Problem Solver that plans and generates the solutions to the problems.

The ANDES project is an on-going collaboration between the United States Naval Academy and the Learning Research and Development Center at the University of Pittsburgh. Between the two sites, there are approximately ten researchers and programmers involved in the project. The researchers at the Naval Academy are developing the Problem Solver while the faculty researchers and graduate students at the University of Pittsburgh are developing the Workbench, Assessor, Action Interpreter, and Help System modules. ANDES is implemented in Allegro Common Lisp, CLIPS, and Microsoft Visual C++ on a Pentium PC running Windows 95.

ANDES represents the solutions to physics problems in a graphical structure called a 'solution graph'. The nodes in the solution graph represent three different kinds of information: 1) problem-specific knowledge (e.g. Block A is on the table.), 2) general physics knowledge (e.g. If an object is near a planet, the planet exerts a gravitational force

on the object.), and 3) strategic and procedural knowledge about how to solve physics problems (e.g. If you are looking for the total force acting on an object, try applying Newton's second law to that object.). The dependencies between these different kinds of information are represented by links in the solution graph structure. A list of facts and equations necessary for each physics problem the students will solve is generated by a rule-based Problem Solver prior to his solving the problem. This information is then used as input to a graph generating program that produces the solution graph used by ANDES.

The remainder of this section narrates a walkthrough of the workbench, the portion of ANDES that interacts with the student. It describes the different components the student can use, and provides a feel for how the behind-the-scenes actions of the physics knowledge base help the student to learn physics in various ways.

ANDES runs as an application in a Microsoft Windows environment. As such, many of its basic operations, e.g. resizing/minimizing windows, opening files, accessing pull-down menus, etc., are performed in the same way as in any other Windows application. This provides a familiar "look and feel" to ANDES and promotes ease of use for the student. Upon starting the program, the student can log into the system. This allows ANDES to save data about each student from session to session and permits multiple students to use the same terminal at different times. After logging in, the student can choose to work on a problem, study an example, or view ANDES help topics. Since the physics knowledge base developed during this research is involved only with problem-solving, this walkthrough will concentrate on that portion of ANDES.

After choosing to solve a problem, the student selects and opens it using a file dialog box which automatically appears. At this point, the problem-solving interface, or workbench, opens (See Figure 1.). It includes a toolbar along the left border containing the various drawing tools needed for the construction of a free body diagram, the problem display, a drawing area, a variable window, and an a equation window. The workbench also has various pull-down menus, and scroll bars that allow the student to change the relative size of each piece of the workbench.

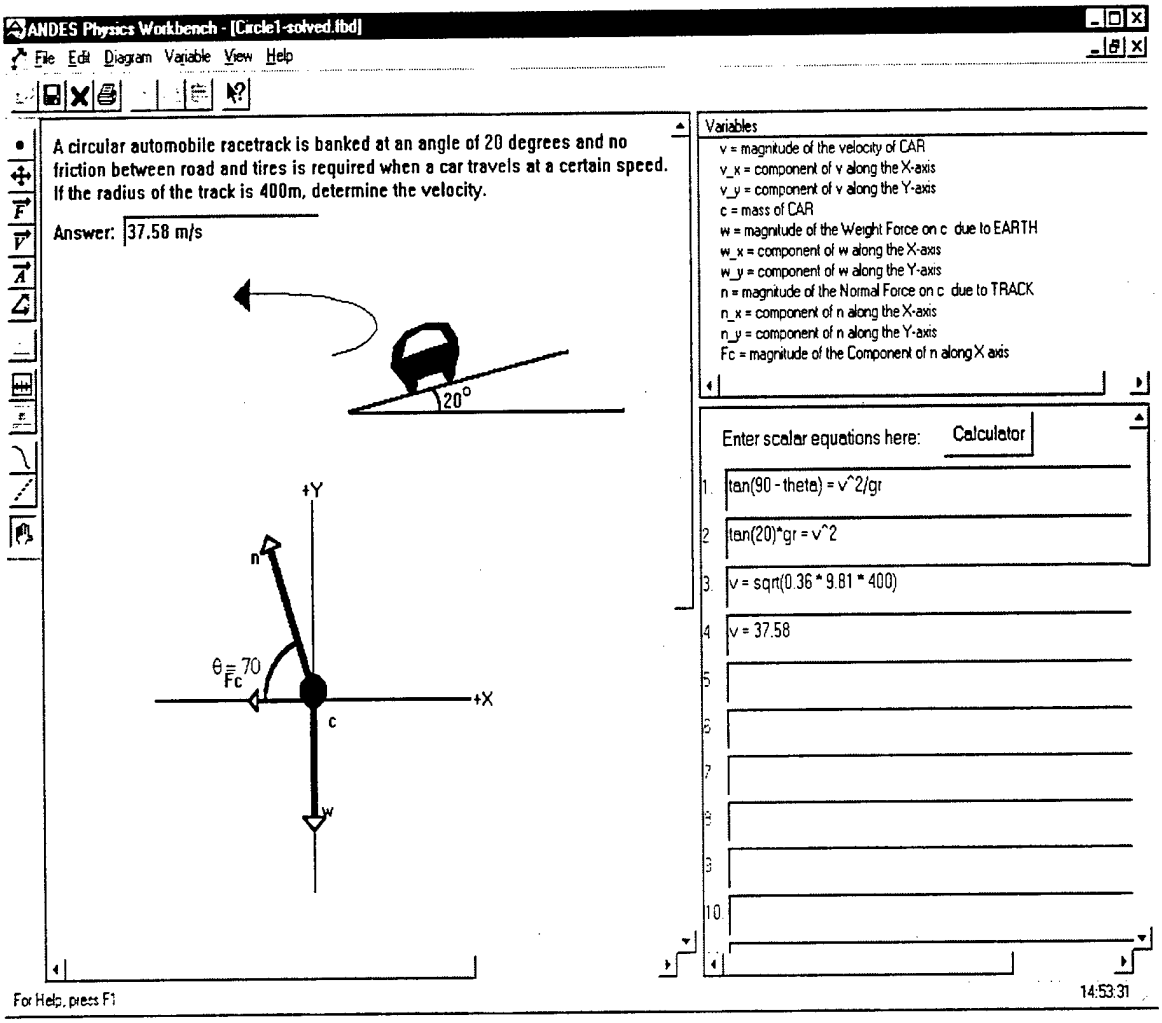


Figure 1. A view of ANDES Workbench.

The drawing tools available for student use operate similarly to those found in an art application. For example, with the vector tool selected, a student (when using the mouse in the free body diagram (FBD) drawing area) can click the left mouse button to place the tail of the vector, and releases it to place the head of the vector. Once the vector is drawn, a dialog box appears that the student uses to define attributes of this vector. The student assigns it a variable name, determines which body it is applied to, and at what time the vector exists. The student can also change the orientation (direction) of the vector by typing in the correct angle for the vector if it is drawn incorrectly. Other tools act in a similar way, each with a specific type of dialog box that appears to define the newly-drawn object.

Once a student adds an object to the free body diagram, its variable shows up in the variable box, along with a description of the variable, and a check-mark or X. For instance, after defining a mass "m" for a meter stick, "m = mass of meter stick" appears along with a check mark. The marks are used to provide visual feedback (green = correct and red = incorrect) to the student as to whether the defined variable is necessary for the problem's solution. The knowledge base produces a set of variables that are used in the correct problem solution; the student's variable is compared against this list to decide if the student's variable deserves a green or red X. This stored list does not determine what the variables should be named, but what the variables should define. It also creates a one-to-one mapping between the variables the student is using and the corresponding variable name the knowledge base generated. In this example, the list would contain "mass of meter stick" rather than "m" as the part to be matched to the student's variable and it would be associated with the system's internal variable name "mass_b1_exang1_1-1".

The pull-down menus located at to top of the workbench allow the student to perform many of these same tasks in different ways, and also access other options. For instance, the student can also use the "Variable" pull-down menu to define variables, or to alter the properties of a variable that he has already defined. This method is also used to define variables that represent undrawn parts of the problem, such as a time interval. The "Edit" pull-down menu facilitates the familiar cut/copy/paste operations found in a wide variety of other Windows applications. The student can also choose which toolbars are visible, save

an altered version of the problem so that he can continue work later, view help on using the workbench, etc. There is also a help tool that can be selected and used to display help on a certain facet of the workbench depending on where the student clicks the tool.

Once the student has defined a variable, it can be used in equations entered in the equation window. The student's equations turn red or green to indicate error or correctness. The student's equations are matched against a set of equations that are derived from the output of the Problem Solver. First , the algebraic manipulator takes the equations generated by the knowledge base and determines all the possible "rearrangements". For example, if the knowledge base output contains "F = m * a", then "m = F / a" and "a = F / m" would also be acceptable. ANDES's equations are in prefix notation so the algebraic processor manipulates the student's equation into canonical form and then matches it against the ones in the set of possible equations it has generated. The variables in the knowledge base equations are more complex than indicated above since they also denote their associated body, axes and method, as well as the time during which they exist.

Within the equation area is a "Calculator" button that pulls up the Windows calculator application. The student can perform mathematical operations and cut and paste the answers from "number-crunching" directly into the equation boxes. When the student has determined the final answer, he types it into the "Answer" text box in the problem display area. Once again, this answer will turn green or red to indicate correctness or otherwise.

Throughout the entire process of drawing a diagram, defining variables, and entering equations, another source of help is available to the student. Right-clicking on any drawn object, variable, or equation summons a pop-up menu which provides several options to the student. For an object in the FBD, the student can change its properties, delete it, duplicate it, or perform other similar options. If the student has clicked on an answer that is red (incorrect), a "What's wrong with that?" dialog box appears. Choosing this option will provide the student with a context-sensitive hint about why the student's answer, vector, variable assignment, etc., is not correct.

Viewing ANDES' help topics currently provides the student mainly with help in working with the ANDES system, rather than physics (subject matter) help. Here the student can learn how to use the various tools to draw a free body diagram, how to uncover the sketch of an example used in self-explanation, and other information on how to efficiently use ANDES' features. This help is hyper linked, meaning that if a student comes across a cross-reference to another help topic, he can click on it and jump directly to the related topic.

Mini-lessons on conceptual topics will automatically appear when ANDES determines that a student does not understand a physics concept. The various help components of ANDES are currently dissertation topics and are under various stages of development.

## III. Circular Motion

The development of the knowledge base dealing with rotary motion began with circular motion. This area of physics introduces some new concepts to students, but still relies on previous topics, such as forces and kinematics (Schulze, et al., 1998). It also lays the foundation for later rotary topics, such as torques and angular momentum.

### Circular Motion 1

The first step in the development was to select representative problems in the selected area of physics that the knowledge base should be able to solve. For example, the first circular motion problem involved an object swinging from a string in a horizontal circle (See Figure 2).

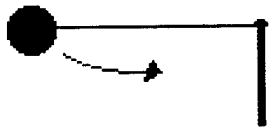

Figure 2: A 10.0 kg mass is tied to a 3/16-in. Manila line, which has a breaking strength of 1800 N. What is the maximum speed the mass can have if it is whirled around in a horizontal circle with a 1.0 m radius and the rope is not to break?

This problem is a simple one since it neglects the weight of the object, and the tension force is the only force in the problem. The problem involves circular motion and little else.

Once the word problem was selected, it had to be translated into the language of the knowledge base implementation, the CLIPS language in the case of ANDES. Knowledge can be represented as templates or simple facts. An example of a simple fact is (block B1), which declares the existence of a block in the system.

Most concepts, however, were defined in terms of templates. A template encapsulates the different parts of a certain concept. It contains a 'slot' for each distinct part of the concept to be defined. For example, the concept of one object being tied to another

is defined by the template *tied-to*, which has a slot for the first object, a slot for the second object, and a slot for the time period during which the two objects are connected. Templates allow for a standard representation of concepts. When two objects are tied together, the objects which are tied together and the time period when they are connected is always important. In general, templates should contain the minimum number of slots necessary to fully represent the concept being considered. These templates provide the structure for facts added to the knowledge base.

In the case of first problem, although it deals with circular motion, several properties about the system could be represented using existing templates from the development of the force and kinematic knowledge bases. Fundamental concepts, such as mass, are obviously the same across many areas of physics. Once all the information about the problem that could be represented by the existing templates was entered, a preliminary run was executed to determine how much the knowledge base knew about the problem from those givens. Since several fundamental concepts, such as the idea of circular motion, were not included, the knowledge base could not derive much new information. Two templates were designed to represent the new knowledge needed for this problem. The first template covered the concept of circular motion and the second dealt with the radius of the revolution of the object:

*circular-motion*

Slots: obj= the name of the body experiencing circular motion
angle= angle between the plane of the body's motion and the horizontal plane
Time= time period during which body is moving circularly
Ex: (circular-motion (obj b1) (angle 0) (Time exm1))

*revolution-radius*

Slots: obj= name of the body experiencing circular motion
value= magnitude of the radius distance
units= units of magnitude
Time= time period during which radius has this value
Ex: (revolution-radius (obj b1) (value 1)(units m)(Time exm1))

These templates allowed the new concepts of circular motion to be represented properly in the knowledge base. Originally, the radius of the revolution was part of the circular motion template, but it soon became apparent that since the radius might be unknown and/or be the goal of the problem, it needed to be a separate concept. A template called strength was also created, which dealt with the breaking strength of the string to which the object is tied. However, it soon became evident that this strength concept could be dealt with as a tension force, which already had a template from the forces knowledge base. Therefore, the strength template was deleted.

Now the problem was fully represented through templates and simple facts, but the knowledge base still did not progress in solving the problem. This was because no new goals or rules had been added to the system to deal with these new templates.

First, planning rules were added. Planning rules consist of goals for the knowledge base to work towards. These direct the flow of the knowledge base in deciding what properties of the problem it needs to define and solve for in order to solve the problem. First, since the goal of the problem was to find the velocity, a planning rule that asserted a subgoal of finding the magnitude of the instantaneous velocity was added. This tied into the planning network already in place for the previously developed sections. A planning rule which decided to use Newton's laws when circular motion and a force are involved was also necessary. Newton's laws are the same for forces, kinematics, and rotary motion, so new planning on lower levels was unnecessary.

A few basic rules were needed to deal with the given information about the problem. For example, a rule was developed that converts the revolution-radius template into another template, named scalar, that is used for all scalar variables. From this scalar template, an equation (for use by the algebraic component of the ANDES system) was generated; this generation is an already generated generic process for all scalars, and did not require a new rule. For example, if the scalar template contained (scalar (m_var b1_mass) (value 10)) then the equation (eq (= b1_mass 10)) was generated and added to the fact base.

The concept of circular motion implies that the body also has an acceleration and a velocity, so rules were created to assert the existence of these properties when there is circular motion. Also, since the direction of acceleration, force, and velocity vectors are continually changing, a design decision was made to choose an arbitrary position for the object and solve for the direction values at that point.

At this point, the basic solution path was complete. However, the knowledge base was using the wrong equation for acceleration; it was still using the linear definition of acceleration, velocity divided by time, rather than the rotary definition of acceleration, velocity squared divided by radius. To correct this, another rule that produced the proper formula was added. This also required a revision to the acceleration equation rule in the kinematics knowledge base to insure it asserted the linear definition of acceleration only in the situations involving linear motion.

Any new rule addition can potentially impact old rules in the same way. These errors are relatively easy to find when the knowledge base asserts incorrect or simply unnecessary facts about the current problem. But it is also possible that the new knowledge base will generate errors that did not previously exist when trying to solve the old problems. Therefore, verifying that the knowledge base is sufficient and correct for all problems is an iterative process.

Now the knowledge base could successfully develop the solution path for the first problem. This means that it would, using the givens, create all the equations necessary to solve for the goal of the problem. The actual algebraic manipulation of the equations to obtain the numerical solution is the responsibility of other components of the ANDES tutoring system. However, to verify that the equations generated by the knowledge base are correct, two preliminary steps occur. First, the problem is solved manually using the traditional approach of pencil, paper, and calculator to arrive at an expected answer. Then the generated equations are used to solve the problem on paper. The problem solution found using the equations from the knowledge base must be consistent with the expected answer.

**Circular Motion 2**

At this point, the process, from stating the problem to verifying the solution path's correctness, is repeated with the next problem. The second problem takes the object's weight into account, causing the string to make some angle with the horizontal plane. Other than that, it is identical to the first problem (See Figure 3).

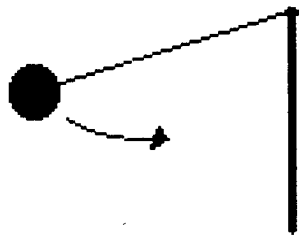

Figure 3: A 20.0 kg mass is tied to a 3/16-in. Manila line, on which is a tension of 341 N. What is the speed the mass has if it is being whirled around in a horizontal circle with a 1.0 m radius, with the Manila line at a 35-degree angle below horizontal?

The concept of gravity was already known by ANDES due to the force knowledge base. Therefore, if the first circular motion problem had represented the new circular motion knowledge correctly, no additional knowledge should be necessary for the second problem to execute correctly. The givens of the second problem were encoded into the appropriate simple facts and templates and the system was executed.

The force knowledge base already handled multiple forces and possessed the knowledge required to combine them into a resultant force. The correct equations were generated, and these were verified to be correct by hand.

**Circular Motion 3**

The third problem also involves the same system as the previous two problems, but instead the goal of the problem is to find the mass of the block, and the given information was changed accordingly (See Figure 4).

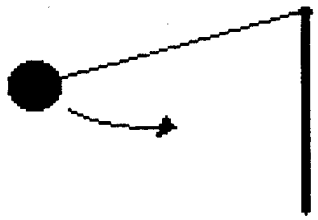

Figure 4: A block is tied to a 3/16-in. Manila line, on which is a tension of 1800 N. It moves in a horizontal circle of radius 1.0 m and speed of 5.19 m/s. The Manila line is at a 20-degree angle below horizontal. What is the mass of the block?

The similarity of the problems presented for the knowledge base to solve may seem to limit its robustness and ability to solve a wide variety of problems. Changing the goal of the problem, however, ensures that the knowledge base can handle a problem from any point of view. For example, having a knowledge base that could only find the velocity of the object in a circular motion system is unacceptable. Any of the quantities, such as mass, velocity, radius, or tension could be unknowns in the student's homework problems, and therefore the knowledge base should be capable of solving for each quantity.

None of the previously solved force or kinematics problems dealt with mass as a problem goal, so a new planning rule was needed to plan the solution path. Another new rule was added to assert a mass scalar of unknown magnitude, if the problem's goal was to find mass. This equation is identical, from the knowledge base's perspective, to equations with actual numbers in them. The algebraic manipulator solves for the unknowns and eliminates the unnecessary variables.

This is all the new information the knowledge base needed to handle the third problem. The solution paths turned out to be fairly similar, once the planning goals gave an initial direction for the problem's solution.

**Circular Motion 4**

The next problem combined the concepts of normal force and circular motion to determine the angle of a racetrack. A decision was made to simplify this problem into one which did not introduce any new concepts into the knowledge base, but for the first time combined concepts from previously separate problems (See Figure 5).

Figure 5: A circular automobile racetrack is banked at an angle of 20 degrees and no friction between road and tires is required when a car travels at a certain speed. If the radius of the track is 400m, determine the velocity.

This situation combined the concepts of circular motion and of a surface exerting a normal force on an object. Since these individual concepts were handled correctly by the knowledge base, combining them into one problem should not, in theory, have caused much difficulty for the system. All the necessary templates were in place: the knowledge base already knew the concepts of a surface, and of contact (and subsequently normal force) between two objects.

The knowledge base was able to generate some equations once the problem was converted into given simple facts and templates, but not enough to solve the problem. The mass of the car in the problem is not necessary to find the solution. However, rules that generated equations about normal and weight forces required the mass of the object experiencing these forces. In order to add the existence of a mass, the rule added for problem three was updated to add the unknown mass equation in any problem where the mass was not given, rather than only when the mass is the problem goal. This allowed the normal and weight forces to be generated; the unknown mass quantity is then canceled by the algebraic manipulator.

**Circular Motion 5**

Once it was verified that combining normal force with circular motion executed correctly, work on the original problem resumed. This problem involves the same situation except that the velocity is given, and determining the angle of the track is the goal of the problem (See Figure 6).

Figure 6: A circular automobile racetrack is banked at an angle such that no friction between road and tires is required when a car travels at a speed of 30 m/sec. If the radius of the track is 400m, determine the angle of the track.

At first glance, it seemed that this problem needed only a new planning rule dealing with the angle as a problem goal to guide the solution, and a domain rule to assert an equation for the unknown angle. Otherwise, the equations necessary to solve this problem were identical to those of the previous problem so, theoretically, very little new knowledge should be necessary.

When translating the homework problem into CLIPS, however, it was realized that the template for contact being two objects (the track and the car in this problem) required a contact angle. A new template was created to embody the concept of contact, without the angle of contact as part of it.

*touching*

Slots: obj1= name of the first body
obj2= name of the second body
Time= time period during which the two objects are touching
Ex: (touching (obj1 b1) (obj2 b2)(Time exm8))

For previous problems angles were always assumed to be numerical values, rather than unknowns (variable names) as in this case. Functions which attempted to evaluate the variable name as a number crashed the system; the rules which called these functions were rewritten to differentiate between variables and numbers. It was also realized that the angle values depend upon the arbitrary point chosen for the object's position in its revolution (i.e. car could be drawn 'going into' or 'coming out of' the page). Another rule was required to determine and assert the relationship between angles and which values should be used.

Once equations for the contact and incline angles were asserted, the contact template could be asserted with the variable name in the angle slot. This template was then used by the knowledge base to derive the existence of a normal force, and from there the solution path was identical to that of the previous problem.

**Circular Motion 6**

The next problem was chosen because it not only dealt with normal and weight forces, but also added the concept of defining the magnitude of one force in relation to the magnitude of another force (See Figure 7).

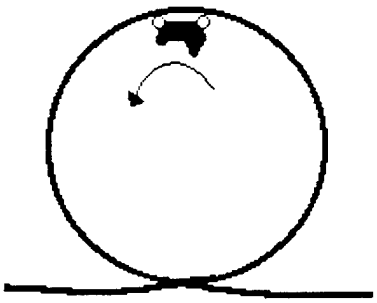

Figure 7: A roller coaster car goes through a loop-the-loop which has a radius of 20 m. If the normal force on the car is equal to its weight when the car is at the top of the loop, how fast is the car traveling?

Since this concept of establishing a relationship between the magnitudes of two variables was new, a template was created.

*ratio*

Slots: quantity1= name of the first quantity (variable name)
quantity2= name of the second quantity (variable name)
value= ratio of first quantity to second quantity
Time= time period during which this ratio exists
Ex: (ratio (quantity1 b1_N_mag) (quantity2 b2_W_mag)(value 2)(Time exm1))

Since other rules already asserted the equations of these force vectors, the concept of ratio simply needed a rule to assert an equation of the form "quantity1 = ratio times quantity2". This relationship provided the final piece of knowledge necessary for the knowledge base to solve this problem.

**Circular Motion 7**

The seventh problem simply provided a test case to see if a typical homework problem encoded into CLIPS would allow the knowledge base to solve the problem without any modifications(See Figure 8).

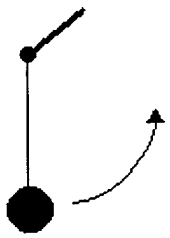

Figure 8: A 5 kg ball is attached to a 2 m rope which will break if the tension exceeds 100 N. If the ball is made to swing in a vertical circle, what is the maximum speed with which the ball can pass through the lowest point?

As expected, the knowledge base was able to generate the correct equations in order to solve the problem.

**Circular Motion 8 & 9**

The eighth and ninth problems encoded into CLIPS givens were closely related to the second and third problems, respectively. Upon closer inspection, these problems provided 'extra' information to the student. The magnitude of tension force was not essential to the second problem, nor was the speed necessary in the third problem. Since the students would be given more than enough information in their homework problems, revised versions of the earlier problems were written (See Figures 9 & 10).

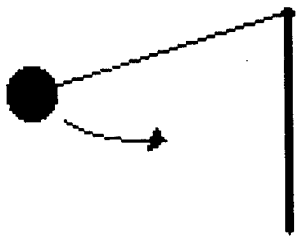

Figure 9: A 20.0 kg mass is tied to a 3/16-in. Manila line. What is the speed the mass has if it is being whirled around in a horizontal circle with a 1.0 m radius, with the Manila line at a 35-degree angle below horizontal?

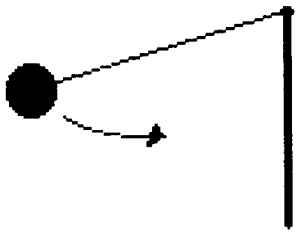

Figure 10: A block is tied to a 3/16-in. Manila line, which has a tension of 1800 N. It moves in a horizontal circle of radius 1.0 m. The Manila line is at a 20-degree angle below horizontal. What is the mass of the block and its speed?

These versions were also correctly solved by the knowledge base without any new knowledge required.

At this point, the portion of the knowledge base handling circular motion was considered sufficient. Additions may be required in the future due to later developments in other areas of the knowledge base, but the foundation has been laid for progress into the torques and angular momentum sections.

## IV. Torque

The next section of physics problems dealt with torques, the rotational equivalent of forces. Torques can be caused by forces acting on a body at a point other than the body's center of gravity. Previously, forces were assumed to act on the center of gravity of a body. An unbalanced torque causes a body to rotate, just as an unbalanced force causes a body to accelerate.

### Torque 1

The first problem is this section has two forces acting on a horizontal bar, and asks the student to find the torque about the left end resulting from these two forces(See Figure 11).

Figure 11: A horizontal 5m bar experiences a 8 N force straight down at its right end. It also experiences a 12 N force up and to the left, 30 degrees above the horizontal, at 2m from its left end. What is the resultant torque of the bar about its left end?

In this problem, the causes of these forces are not given--these forces are simply stated to exist. In previous problems, forces were implied by other attributes of the system; a weight force, for example, was implied by the existence of gravity and a body with a mass. Although the concept of an "arbitrary" force is fairly straightforward when compared to deriving the existence of a force from other system facts, it simply had not been dealt with before, and therefore a new template was required to encapsulate this knowledge.

*given-force*

Slots: applied-to= the name of the body experiencing the force
value=magnitude of the force
angle= direction of the force
units= units of force
applied-by= cause of the force (can be nil)
Time= time period during which the force acts on the body
Ex: (given-force (applied-to b1)(angle 270)(units N)(applied-by nil)(Time extor1))

Another new concept in this problem was the selection of specific, distinct points on a single body. Using existing rules to handle this new situation was difficult because many rules relied on several facts being associated with the same body in order to trigger the rule. In these cases, however, the goal of a problem might be associated with the body, while the force is associated with a certain point, and a displacement vector is associated with a third point.

Solving this difficulty required developing a standard method of how the associations between vectors of a system (forces, displacement, torques, etc.) would be made. A function was also needed that identified when given points were on the same body. Finally, many existing rules were modified to work both when a single body was used, and when different points on the same body were specified.

This torque problem obviously had a problem goal unlike previous problems. Rules were written to assert the existence of a resultant torque based on the goal of the problem, and to assert its magnitude as unknown. After that, this problem could follow the same strategy path of previous problems by setting a goal to find all vectors acting on the body with the resultant torque.

In the process of writing rules to calculate the torques produced, it was realized that two different equations were possible for computing a torque's magnitude. One equation uses components and the other uses the magnitudes of the force and displacement vectors, and the angle between them. Once the goal was set to use components, this problem could take advantage of existing rules which handled the generation of the vector components. However, since torque vectors did not need to be broken into components, this action had

to be blocked. The second method used a new equation and therefore required a new rule to assert it.

Since the calculation of certain variables (e.g. the magnitude of each generated torque) can be accomplished by two independent methods, and certain equations may only be necessary for one of the methods, the knowledge base needs to be able to differentiate between these two methods. At a result the system needed to establish a strategy decision point to choose a method. Rules were written which asserted this decision point, and made use of it in discerning between the different methods of calculating torque magnitudes. At this point, some variables, like the magnitude of a force, are always the same regardless of the method used, and therefore their names do not include information about the method choice made at that point. However, a problem might have "the magnitude of torque #1 by components" and "the magnitude of torque #1 by vector magnitudes," and so their variable names include information about the choice made at the appropriate decision point.

Computing the resultant torque involved an approach similar to the method of calculating a resultant force. The torque magnitudes from one method were added together in one equation and set equal to the resultant magnitude, and those from the second method were added in another equation. In this way, the student would not need to write both sets of equations in order to find the resultant torque and solve the problem. At this point, the first torque problem was solved.

**Torque 2**

The next problem involved a slightly more complex system than the first one, with the torque arms being non-horizontal and distinct from each other, but otherwise held few differences from the first problem(See Figure 12).

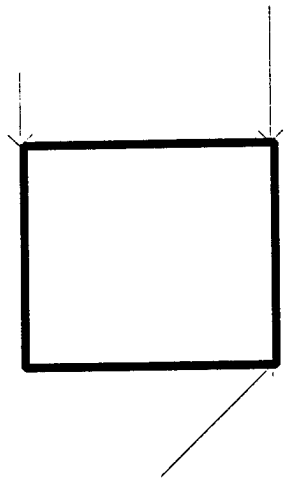

Figure 12: A square metal plate 0.18 m on each side is pivoted about its central axis, perpendicular to the plate. Calculate the net torque about this axis due to three forces:16 N down, applied to top left corner, 24 N down, applied to top right corner, and 18 N, applied at 45 degrees to bottom right corner.

This problem does require a minor geometric calculation to determine the length of the torque arm. Upon discussion with the physics advisors, it was decided that the knowledge necessary to handle even basic geometry could quickly become a non-trivial component of the knowledge base, and that the knowledge base should remain as close as possible to "just physics." The wording of the problem was left the same, however. The student would be required to compute the torque arm length on his own; ANDES can check the student's value against the given one, but need not offer any help other than to tell the student to check his math. Just as a human physics tutor would not be expected to review geometric fundamentals with students, neither would ANDES need the knowledge necessary to provide help for errors in these types of calculations.

Once this determination was made, this second problem was solved, with the torque arm length hard-coded into the given facts of the problem. These two problems were originally solved with the standard axes being the only choice of axes. Because of concurrent work on the kinematics knowledge base, it was realized that choosing the axis in the direction of the torque arm was also a possible student approach. Differentiating

between the possible axis choices was handled similarly to the calculation method choices, but with a different decision point denoting the axis choice being made.

### Torque 3

The third problem was written to test this new capability of choosing a set of axes besides the standard ones. It is identical to the first problem with the exception of the orientation of the torque arm (See Figure 13).

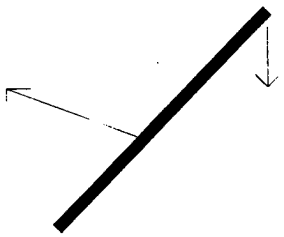

Figure 13: A 5m bar at 50 degrees above horizontal experiences a 8 N force straight down at its right end. It also experiences a 12 N force to the left, 30 degrees above horizontal, at 2m from its left end. What is the resultant torque of the bar about its left end

As expected, this problem ran correctly once all the axis knowledge and rules were in place. As an example of the knowledge base's ability to handle multiple solution paths, it generated four solutions since there was the possibility of two axis choices and each could be used with two solution methods.

### Torque 4

The next problem was very similar to the second problem, except that the resultant torque was given, and the student was tasked with calculating the magnitude of the one of the forces on the square. This required the knowledge base to use essentially the same logic as before, but to work in reverse (See Figure 14).

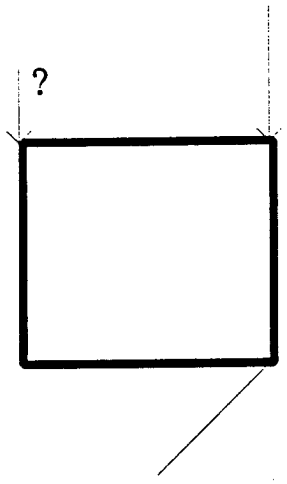


Figure 14: A square metal plate 0.18 m on each side is pivoted about its central axis, perpendicular to the plate. Calculate the net torque about this axis due to three forces: a force of unknown magnitude down, applied to top left corner, 24 N down, applied to top right corner, and 18 N, applied at 45 degrees to bottom right corner. If net torque is 5.88 N * m, what is the magnitude of the force applied to the top left corner?

Rules were written that allowed a torque that was a given to be recognized as the resultant torque on an object. With these in place, the knowledge base used the same goals as the last problem; only the unknowns in the problem were different, and these are treated by the algebraic manipulator as known variables. This problem ran correctly with the new torque and planning rules.

**Torque 5**

The next problem involved several concepts very different from those encountered in previous problems. It dealt with a rotating object, and with the properties of its rotation (e.g. speed of rotation, rotational inertia, etc.) which did not need examination in previous problems, and so it seemed that several new templates would be necessary (See Figure 15).

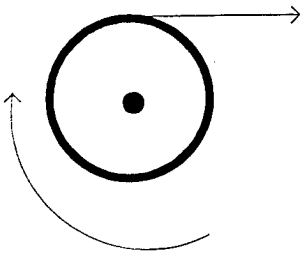

Figure 15:A cord is wrapped around the rim of a flywheel 0.4 m in radius, and a steady pull of 50.0 N is exerted on the cord. The wheel is mounted in frictionless bearings on a horizontal shaft through its center. The moment of inertia of the wheel is 4.0 kg m^2. What is the angular acceleration of the wheel?

Moment of inertia (I) is similar to the concept of mass, but with some important differences. Although the mass of an object usually stays constant throughout a problem, its moment of inertia can change. For example, an ice skater's moment of inertia changes when she goes from spinning with her arms outstretched to spinning with her arms above her head. A new template was added to encapsulate these characteristics.

*moment-of-inertia*

Slots: obj= the name of the body
value= magnitude of moment of inertia
units= units of magnitude
Time= time period during which body has this moment-of-inertia
Ex: (moment-of-inertia (obj b1)(value 30)(units kg m^2)(Time extor5))

In attempting to produce given facts about this problem, the template describing the concept of an axis of rotation was also created. This template helped to simplify some of the associations between different attributes of the system, and so it was added to the given facts of the previous torque problems, and the rules were altered to take advantage of this new encapsulation of knowledge.

*rotation-axis*

Slots: obj= the name of the rotating body
point= point on the body through which the axis passes
Ex: (rotation-axis (obj b1)(point p0))

Once again, this problem had a new goal, finding the angular acceleration of the body. The rotation of a body has many parallels with the movement of a body, in that there is a rotary equivalent for mass, force, acceleration, distance, velocity, momentum, and other linear quantities. Therefore the techniques used to work with these rotary quantities should be comparable to those used to deal with linear motion.

The method for determining angular acceleration was based on the way in which linear acceleration is calculated in the kinematics section of the knowledge base. These rules were fairly easy to construct since they so closely paralleled many rules already written. This problem worked correctly without any further difficulties.

Having completed these problems, the torque knowledge base was considered sufficient to start the angular momentum knowledge base.

## V. Angular Momentum

The final section of physics problems dealt with angular momentum, the rotational equivalent of linear momentum. A key concept of angular momentum is the fact that if there are no net external torques on a system, angular momentum is conserved. Since linear momentum had not yet been added to the knowledge base, there was no precedent as to how to embody this "conservation" concept. However, a correct implementation of this concept for angular momentum will serve as a good framework when linear momentum is added later.

### Angular Momentum 1

The first angular momentum problem simply involved calculating the angular momentum of a spinning object, given certain attributes of the object (See Figure 16).

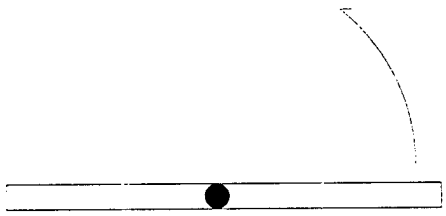

Figure 16: A meter stick of mass 0.5 kg is rotated at 3 rev/sec about its midpoint. What is its angular momentum?

Angular momentum (L) is defined as the product of an object's moment of inertia and its angular velocity. This problem also involved calculating the moment of inertia from the given information about the object, rather than having the value explicitly given as in previous problems. An object's moment of inertia is calculated differently depending on the shape of the body and its axis of rotation. Most equations involve multiplying mass, one dimension (radius or length) squared, and a certain coefficient; this coefficient is determined

by the shape and the axis around which it is spinning. For instance, a stick spinning about its endpoint uses the formula ($I=^1/_3 m l^2$). while one spinning around its midpoint uses ($I=^1/_{12} m l^2$). This situation called for a function which determined the proper value of the coefficient. Because the equation for determining moment of inertia is necessary, rather than just the value itself, the function does not handle any further calculations. The actual equation is determined and printed by the knowledge base rules.

The rotation radius of the spinning object is also a given in these problems, and needed a template to contain this concept. As previously noted, this value is used to calculate the moment of inertia of the object.

*rotation-radius*
Slots: obj= the name of the rotating body
value=magnitude of rotation radius
units=unit of length used to measure rotation radius
Time=time during which rotation radius has given value
Ex: (rotation-radius (obj b1)(value 0.5)(units m)(Time exang1))

With this new concept now standardized in templates, rules were written that calculated the moment of inertia from the appropriate givens. The calculated value of moment of inertia is a scalar and can be stored as such. Now the magnitude of the angular momentum vector could be calculated as the product of moment of inertia and angular velocity, and the problem was solved.

**Angular Momentum 2**

The second angular momentum problem resembled the previous one, with a different type of object spinning (See Figure 17).

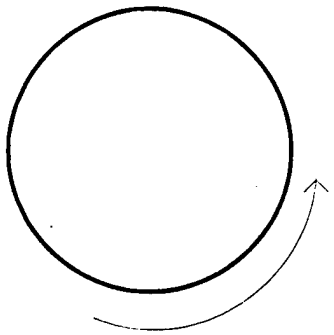

Figure 17: A hoop of mass 2 kg, with radius 0.75 m, is spinning at 4 rev/sec around an axis through its center and perpendicular to the hoop. What is its angular momentum?

This problem was picked simply as a test for the function that calculates the necessary coefficient in the moment of inertia equation discussed earlier. The knowledge base solved this problem correctly without any modifications.

**Angular Momentum 3**

The third angular momentum problem involved calculating angular momentum using the same methods as previous problems, but it also dealt with the conservation of angular momentum over a time interval (See Figure 18).

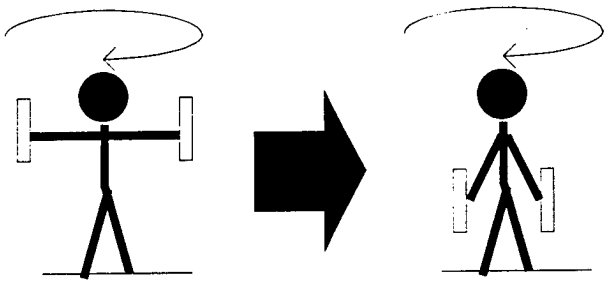

Figure18: A man stands on a frictionless platform that is rotating at an angular speed of 1.2 rev/sec; his arms are outstretched and he holds a weight in each hand. With his arms in this position, the rotational inertia of the entire man-weights-platform system is 6 kg*m^2. If the man moves his arms and the weights so that the rotational inertia of the system decreases to 2 kg*m^2, what is his new angular speed?

The concept of the conservation of angular momentum states that a rotating object's angular momentum remains constant as long as the object does not experience angular acceleration. Since angular momentum is angular velocity times moment of inertia, when one value changes, the other changes as well to compensate, leaving the overall value of angular momentum the same.

Several rules were required to determine if this concept applied, ensuring that the knowledge base would only attempt to conserve angular momentum when no torques or angular accelerations were acting on the body. The method of calculation of angular momentum was identical to that in the first problem. The knowledge base generated, among others, an equation calculating the angular momentum at the beginning of the problem, another for the end of the problem, and one equating these two values of angular momentum. Angular velocity as an unknown had already been covered by a previous problem, so when the conservation concept was correctly modeled by the knowledge base, this problem ran correctly.

**Angular Momentum 4**

The last angular momentum problem also involved the conservation of angular momentum, but in this problem two separate bodies combined into a compound body in the course of the problem (See Figure 19).

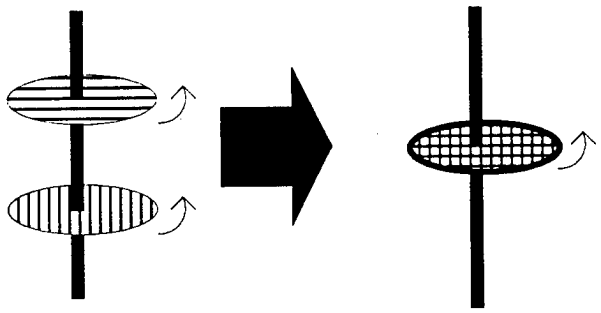

Figure 19:Two disks are mounted on low friction bearings on the same axle and can be brought together so that they couple and rotate as one unit. The first disk has a moment of inertia of 3.3 kg * m^2 and is rotating at 450 rev/min. The second disk, with moment of inertia of 6.6 kg * m^2, is spinning at 900 rev/min in the same direction as the first disk. If the two disks are then coupled together, what would their angular speed be after coupling?

The concept of a compound body consisting of two separate bodies was already covered in the force knowledge base. This problem presented a slight different situation, however, since the two bodies come together in mid-problem. Several rules were written to properly model this concept, and to determine information about the compound body based on the attributes (mass, moment of inertia, etc.) of its parts.

Once the compound body concept was properly dealt with, only one more rule was needed to handle the conservation of angular momentum of a compound body. With these additions, the knowledge base generate the necessary equations to solve this problem.

After solving the four angular momentum problems, every problem in all three sections was re-run to ensure that the knowledge base could still solve all problem correctly. In some cases, minor changes were necessary to 'fix' an old problem, but no significant changes were made. At this point, the knowledge base was considered sufficient to handle physics homework problems dealing with circular motion, torque, and angular momentum.

## VI. Help Evaluation

During February and March 1998, classroom sessions were conducted with two sections of Dr. R.N. Shelby's Physics I courses. The purpose of the sessions was to get a general idea of the type of questions that students ask as they work through rotary motion problems. The results of these sessions were provided to the developers of the ANDES help system at the University of Pittsburgh.

During these sessions, students worked in groups on circular motion, torque, and angular momentum problems that were used to develop ANDES. The students were split into groups with the hopes that this would foster discussion during problem-solving, and help to vocalize the questions that a student working alone would ask internally. A member of each group was designated as a recorder to note the questions asked among the group. The students had approximately twenty minutes to solve two problems, to take pressure off simply solving the physics problem and to encourage more discussion. These sessions involved only about forty students: this study was not intended as research in itself, but just to serve as a reinforcement that the help system would be providing answers to the type of questions that the students would be asking.

The questions that students asked during problem-solving fell into a few main categories. Although some questions will certainly fall outside of these generalizations, these types of questions constituted the bulk of those asked, and suggest the minimum requirements for a help system. In other words, if students cannot readily find answers to these questions, many will reach an impasse in their physics problem-solving. Statistical measurements on how these questions divided into the various categories were not done because of the small sample size.

*"What is the equation/formula for _____ ?"*

This was one of the most popular question types, for good reason. Since the students were engaged in solving a physics problem, their approach was most likely centered around

determining the equations necessary for solving the problem, and figuring out how to use these equations together to work toward a solution.

*"What is the equation which connects _____ and _____ ?"*
*"How does _____ affect _____ ?"*

These questions are similar to those of the first category. This type, however, shows the students had understood that two separate quantities were necessary in finding the problem solution, but they were unsure how to use them together.

In some cases, the information about the problem that the students were trying to use was more abstract; in one question, for example, a ball was swung in a vertical circle, and the students did not know how this trait of the system would affect their calculations.

*"What does _____ have to do with this problem?"*

This kind of question seemed to come up in two different types of circumstances. Sometimes the students would be presented with a quantity in the givens that was unnecessary in solving the problem. Upon discovering this unused fact, the students would wonder if their solution should have included this information in some way. In the other case, the students were uncertain to how tie in a quantity with the rest of the problem. This second type of situation resemble those reflected in the previous question type.

*"What are all the forces present in this problem?"*

Although this question seemed fairly specific, it was asked frequently. The repetition of this question suggests that the students were attempting to solve the problems in the way that they were taught, i.e. drawing a force body diagram (The question "What would a force body diagram look like?" also occurred several times.). While this question may be specific

to a certain type of physics problem, a help system should provide quality guidance to students who are attempting to use the correct methods.

## VII. Conclusions

The circular motion, torque, and angular momentum knowledge bases developed provide the physics backbone to ANDES. The equations and solution graphs the knowledge bases generate allow ANDES to more effectively tutor the student because the student assessor, workbench feedback system, and help systems develop at the University of Pittsburgh depend on the knowledge bases to function correctly.

Physics professors examined the equations and list of facts that the knowledge base produced for each problem and determined them to be correct. This suggests that the cognitive strategies that physics professors commonly use to solve these problems were sufficiently modeled by the knowledge base. This does not imply that all cognitive processes involved in circular motion, torque, and angular momentum were completely modeled. Since the knowledge base was developed as part of a tutoring system, however, only the commonly used problem-solving strategies were needed.

The research will also prove useful in other ways. The ANDES system as a whole will also serve as a test of the theories concerning the type of instruction that it employs: computer-based coached problem-solving. If students respond well to ANDES tutoring, similar tutoring systems in other subject matters may be beneficial as well. The data gained from observing students' interaction with ANDES and any improvements in their physics problem-solving skills will point the way for future research in learning and cognitive skill acquisition.

Experience gained from developing these knowledge bases will also be useful in future knowledge base development and refinement of the ANDES system. A key part of the development included using the rules written to solve previous problems as a guide for new rules. This methodology helped preserve a continuity and consistency within the knowledge base and closely matched the cognitive processes of an expert; a physicist knows and uses the concept that torques sum together into a resultant torque in a way similar to forces summing together into a resultant force. While this research took advantage of this similarity principle at times, practicing it more strictly, especially during the design and planning process, would greatly improve the knowledge base's flexibility and ease of use.

Unforeseen design questions continued to appear throughout the process of expanding the knowledge base to solve problems. For example, the use of variables for angle values originally caused quite a few snags, since angles were assumed to be given as numerical values. The knowledge base needed to work with angles either as variables or values, and use either form in solving the problem. The method for determining if conservation of momentum exists and the subsequent calculations also became an issue. After deciding what conditions must exist to have conservation, an appropriate method for determining when momentum values were equivalent was needed. This process was also designed in such a way that addition of the concept of linear momentum to the knowledge base can be based on the method of conceptualizing angular momentum. These are a few examples of the type of challenges faced when developing this knowledge base.

The lessons learned from this effort to develop a conceptual knowledge base will provide valuable experience for similar projects to come. The ability to standardize and encapsulate the concepts of physics into a computer knowledge base indicates that knowledge bases for a wide variety of other complex subjects are also feasible. In addition, these knowledge bases can have practical applications, whether in tutoring or other areas. In the course of this research several ideas have surfaced on how to improve the structure of this knowledge base, and perhaps guide the development of future conceptual knowledge bases. For example, our model of the physics experts' problem-solving methods could have been developed to refine the solution path into smaller steps. The next phase of the ANDES project will incorporate an improved planning module to develop high-level solutions and our knowledge base will need to be revised to reflect this improved planning. It is believed that more detailed planning will be necessary to solve more difficult and complex problems. These types of problems need to be incorporated into the system as an incentive for better students to use and value the system.

In addition, unresolved design issues still exist. For example, a method of handling a third dimension has not been established, nor has a way of representing vectors associated with an object that have no absolute direction, only direction relative to each other. These types of issues were expected, but the problems encountered during this research and their

resolutions should serve as guideposts to the future attempts to develop or refine the knowledge bases when problems involving these concepts are introduced into the ANDES system.

## IX. Bibliography

Anderson, J. R., Boyle, C. F., and Reiser, B. J. (1985). Intelligent tutoring systems. *Science*, 228:456-462.

Anderson, J. r., Corbett, A. T., Koedinger, K. R., and Pelletier, R. (1995). Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences*, 4(2): 167-207.

Bielaczyc, K., Pirolli, P., and Brown, A. L. (1994). Training in self-explanation and self-regulation strategies: Investigating the effects of knowledge acquisition activities on problem-solving. *Cognition and Instruction.*

Burton, R. R. and Brown, J. S. (1982). An investigation of computer coaching for informal learning activities. In Sleeman, D. and Brown, J. S., eds., Intelligent Tutoring Systems, pp. 79-98. Academic Press, Orlando, FL.

Chi, N., Bassok, M., Lewis, M., Reimann, P., and Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science*, 15: 145-182.

Clancey, W. J. (1987). Knowledge-based Tutoring: The GUIDON Program, The MIT Press, Cambridge, MA.

Collins, A. and Brown, J. S. (1990). The computer as a tool for learning through reflection. In Mandl, H. and Lesgold, A., eds., Learning Issues for Intelligent Tutoring Systems. Springer, New York, NY.

Collins, A. and Brown, J. S., and Newman, S. E. (1989). Cognitive apprenticeship: Teaching the craft of reading, writing and mathematics. In Resnick, L. B., Ed., Knowing, Learning and Instruction: Essays in Honor of Robert Glaser, pp. 543-594. Lawrence Erlbaum Associates, Hillsdale, NJ.

Conati, C., Gertner, A. S., VanLehn, K., and Druzdel, M. J. (1997). On-line student modeling for coached problem solving using Bayesian networks, *Proceedings of the Sixth International Conference on User Modeling UM-97.* Sardinia, Italy, 231-242.

Conati, C., Larkin, J. and VanLehn, K. (1997) A computer framework to support self-explanation. in *Proceedings of 8th World Conference on Artificial Intelligence in Education: Knowledge and Media in Learning Systems*. B. du Boulay and R. Mizoguchi (eds). IOS Press.

Gertner, A, Conati, C, and VanLehn, K. (in press) Procedural help in Andes: Generating hints using a Bayesian network student model, *Proceedings of the 15th National Conference on Artificial Intelligence*. Madison, WI.

Koedinger, K. and Anderson, J. R. (1993). Reifying implicit planning in geometry: Guidelines for model-based intelligent tutoring system design. In Lajoie, S. P. and Derry, S. J., editors, *Computers as Cognitive Tools*. Lawrence Erlbaum Associates, Hillsdale, NJ as sited in VanLehn, K. (1996b). Conceptual and meta learning during coached problem solving. In Frasson, C., Gauthier, and G., Lesgold, A., editors, *Intelligent Tutoring Systems*, pp. 29-47. Springer, New York, NY.

Lajoie, S. P. (1993). Computer environments as cognitive tools for enhancing learning. In Lajoie, S. P. and Derry, S. J., eds., Computer as Cognitive Tools, Lawrence Erlbaum, Hillsdale, NJ.

Lesgold, A., Lajoie, S., Bunzo, M. and Eggan, G. (1992). Sherlock: A coached practice environment for an electronics troubleshooting job. In Larkin, J. and Chabay, R., editors, *Computer Assisted Instruction and Intelligent Tutoring Systems: Shared Goals and Complementary Approaches*, pp. 201-238. Lawrence Erlbaum Associates, Hillsdale, NJ.

Mark, M. A. and Greer, J. E. (1995). The VCR tutor: Effective instruction for device operation. *The Journal of the Learning Sciences*, 4(2):209-246.

Mitrovic, A. and Djordevic, S. (1995). Interactive reconstructive student modeling: A machine learning approach, *International Journal Human-Computer Interaction*, 7(4):385-401.

Novak, G. and Araya, A. (1980). Research on expert problem solving in physics. *In Proceedings of the First National Conference on Artificial Intelligence*, AAAI Press, Menlo Park, CA.

Newell, A. (1990). Unified Theories on Cognition. Harvard University Press, Cambridge, Massachusetts.

Russell, S. and Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*, Morgan Kaufman, Los Altos, CA.

Schulze, K. G. (1989). Preceptor: An algebraic word problem tutor. *Proceedings of the Sixth Canadian Symposium on Instructions Technology*, pp. 64-67. Halifax, Canada.

Schulze, K. G. (1990). The use of multiple explanations in an open-ended intelligent tutor system. *Journal of Educational Technology System*, 10(3):235-243.

Schulze, K. G., Correll, F. D., Shelby, R. N., Wintersgill, M. C., and Gertner, A. S. (1998). A CLIPS problem solver of Newtonian physics force problems. In Giarratano, C. and Riley, G., Expert Systems Principles and Programming, 3rd edition, PWS Publishing Company.

Siegler, R. S. and Jenkins, E. (1989). How Children Discover New Strategies. Lawrence Erlbaum, Hillsdale, NJ.

Shute, V. J. (1991). Who is likely to learn programming skills? *Journal of Educational Computing Research*, 6:1-24.

Shute, V. J. and Psotka, J. (1996). Intelligent tutor systems: Past, present and future. In Joanne, D., editor, *Handbook of Research on Educational Communications and Technology*. Scholastic Publications.

Soloway, E., Woolf, B., Rubin, E. and Barth, P. (1981). MEMO-II: An intelligent tutoring system for novice programmers. *Proceedings of the Seventh IJCAI* (vol. 2), Vancouver, pp. 975-977.

Stern, M., Beck, J., and Woolf, B. P. (1996). Adaptations of problem presentation and feedback in an intelligent mathematics tutor, In Frasson, C. and Lesgold, A., eds. *Proceedings of the 3rd International Conference on Intelligent Tutoring Systems*. Springer, Berlin, 605-613.

VanLehn, K., Jones, R. M., and Chi, M. T. H., (1992). A Model of self-explanation effect. *The Journal of the Learning Sciences*, 2(1):1-59.

VanLehn, K. (1995). Rule-learning events in the acquisition of a complex skill. (Unpublished manuscript).

VanLehn, K. (1995a). Looking in the book: The effects of example-exercise analogy on learning. (Unpublished manuscript).

VanLehn, K. (1996a). Cognitive skill acquisition. In Spence, J., Darly, J., and Foss, D. J., editors, *Annual Review of Psychology*, Vol. 47, pp. 513-539. Annual Reviews, Palo Alto, CA.

VanLehn, K. (1996b). Conceptual and meta-learning during coached problem solving. In Frasson, C. and Lesgold, A., eds. *Proceedings of the 3rd International Conference on Intelligent Tutoring Systems*. Springer, Berlin, 29-47.

## Appendix A: Circular Motion Problems

### Circular Motion Problem 1

A 10.0 kg mass is tied to a 3/16-in. Manila line, which has a breaking strength of 1800 N. What is the maximum speed the mass can have if it is whirled around in a horizontal circle with a 1.0 m radius and the rope is not to break?

### Givens

```
(deffacts exm1 "Facts for exercise rotary motion problem m1"
  (problem exm1)
  (mass (obj b1) (value 10.0) (units kg))
  (string s1)
  (taut s1)
  (tied-to (object1 b1)(object2 s1)(Time exm1))
  (tension (magnitude 1800)(units N)(applied-to b1)(applied-by s1)(Time exm1))
  (relative-position (obj s1)(wrt b1)(direction 0)(Time exm1)) ;; gives dir of tension
  (circular-motion (obj b1)(angle 0)(Time exm1)) ;; give dir of accel
  (revolution-radius (obj b1)(value 1.0) (units m) (Time exm1))
  (goal-problem (is find-inst-vel) (applied-to b1) (Time exm1)))
```

### Equations Generated

```
(= mass_b1_exm1_1-1 10.0)
(= r_b1_exm1_1-1 1.0)
(= dir_a_b1_exm1_1-1 0)
(= dir_t_b1_s1_exm1_1-1 0)
(= mag_t_b1_s1_exm1_1-1 1800)
(= t_xc_b1_s1_exm1_2-1 (* mag_t_b1_s1_exm1_1-1 (cos (- 0 0))))
(= T_b1_s1_exm1_1-1 t_xc_b1_s1_exm1_2-1)
(= t_yc_b1_s1_exm1_2-1 (* mag_t_b1_s1_exm1_1-1 (sin (- 0 0))))
(= mag_v_b1_exm1_1-1 unknown1)
(= mag_a_b1_exm1_1-1
        (/ (* mag_v_b1_exm1_1-1  mag_v_b1_exm1_1-1 ) r_b1_exm1_1-1 ))
(= A_b1_exm1_1-1_2-1 a_xc_b1_exm1_2-1)
(= a_xc_b1_exm1_2-1 (* mag_a_b1_exm1_1-1 (cos (- 0 0))))
(= a_yc_b1_exm1_2-1 (* mag_a_b1_exm1_1-1 (sin (- 0 0))))
(= t_xc_b1_s1_exm1_2-1(* mass_b1_exm1_1-1 a_xc_b1_exm1_2-1))
```

**Circular Motion Problem 2**

A 20.0 kg mass is tied to a 3/16 in. Manila line, which has a tension of 341 N. What is the speed the mass has if it is being whirled around in a horizontal circle with a 1.0 m radius and the Manila line is at a 35-degree angle below horizontal?

**Givens**

```
(deffacts exm2 "Facts for exercise rotary motion problem m2"
  (problem exm2)
  (mass (obj b1) (value 20.0) (units kg))
  (string s1)
  (taut s1)
  (tied-to (object1 b1)(object2 s1)(Time exm2))
  (tension (magnitude 341)(units N)(applied-to b1)(applied-by s1)(Time exm2))
  (relative-position (obj s1)(wrt b1)(Time exm2)(direction 35))
  (circular-motion (obj b1)(angle 0)(Time exm2)) ;; give dir of accel
  (near-planet (planet earth) (Time exm2))
  (revolution-radius (obj b1)(value 1.0) (units m) (Time exm2))
  (goal-problem (is find-inst-vel) (applied-to b1) (Time exm2)))
```

**Equations Generated**

```
(= mass_b1_exm2_1-1 20.0)
(= r_b1_exm2_1-1 1.0)
(= dir_a_b1_exm2_1-1 0)
(= dir_t_b1_s1_exm2_1-1 35)
(= mag_t_b1_s1_exm2_1-1 341)
(= t_xc_b1_s1_exm2_2-2 (* mag_t_b1_s1_exm2_1-1 (cos (- 35 35))))
(= T_b1_s1_exm2_1-1 t_xc_b1_s1_exm2_2-2)
(= t_xc_b1_s1_exm2_2-1 (* mag_t_b1_s1_exm2_1-1 (cos (- 35 0))))
(= T_b1_s1_exm2_1-1 t_xc_b1_s1_exm2_2-1)
(= t_yc_b1_s1_exm2_2-2 (* mag_t_b1_s1_exm2_1-1 (sin (- 35 35))))
(= t_yc_b1_s1_exm2_2-1 (* mag_t_b1_s1_exm2_1-1 (sin (- 35 0))))
(= mag_w_b1_earth_exm2_1-1  (* mass_b1_exm2_1-1 g))
(= dir_w_b1_earth_exm2_1-1 270)
(= w_xc_b1_earth_exm2_2-2 (* mag_w_b1_earth_exm2_1-1 (cos (- 270 35))))
(= w_xc_b1_earth_exm2_2-1 (* mag_w_b1_earth_exm2_1-1 (cos (- 270 0))))
(= w_yc_b1_earth_exm2_2-2 (* mag_w_b1_earth_exm2_1-1 (sin (- 270 35))))
(= W_b1_earth_exm2_1-1 w_yc_b1_earth_exm2_2-2)
(= w_yc_b1_earth_exm2_2-1 (* mag_w_b1_earth_exm2_1-1 (sin (- 270 0))))
(= W_b1_earth_exm2_1-1 w_yc_b1_earth_exm2_2-1)
(= mag_v_b1_exm2_1-1 unknown1)
(= mag_a_b1_exm2_1-1
        (/ (* mag_v_b1_exm2_1-1  mag_v_b1_exm2_1-1 ) r_b1_exm2_1-1 ))
(= a_xc_b1_exm2_2-2 (* mag_a_b1_exm2_1-1 (cos (- 0 35))))
```

```
(= a_yc_b1_exm2_2-2 (* mag_a_b1_exm2_1-1 (sin (- 0 35))))
(= A_b1_exm2_1-1_2-1 a_xc_b1_exm2_2-1)
(= a_xc_b1_exm2_2-1 (* mag_a_b1_exm2_1-1 (cos (- 0 0))))
(= a_yc_b1_exm2_2-1 (* mag_a_b1_exm2_1-1 (sin (- 0 0))))
(= g 9.8)
(= (+ t_yc_b1_s1_exm2_2-1 w_yc_b1_earth_exm2_2-1)
        (* mass_b1_exm2_1-1 a_yc_b1_exm2_2-1))
(= (+ t_xc_b1_s1_exm2_2-1 w_xc_b1_earth_exm2_2-1)
        (* mass_b1_exm2_1-1 a_xc_b1_exm2_2-1))
(= (+ t_yc_b1_s1_exm2_2-2 w_yc_b1_earth_exm2_2-2)
        (* mass_b1_exm2_1-1 a_yc_b1_exm2_2-2))
(= (+ t_xc_b1_s1_exm2_2-2 w_xc_b1_earth_exm2_2-2)
        (* mass_b1_exm2_1-1 a_xc_b1_exm2_2-2))
```

**Circular Motion Problem 3**

A block is tied to a 3/16-in. Manila line, which has a tension of 1800 N. It moves in a horizontal circle of radius 1.0 m and has a speed of 5.19 m/s. The Manila line is at a 20-degree angle below horizontal. What is the mass of the block?

**Givens**

```
(deffacts exm3 "Facts for exercise rotary motion problem m3"
  (problem exm3)
  (string s1)
  (taut s1)
  (tied-to (object1 b1)(object2 s1)(Time exm3))
  (speed (obj b1)(type inst)(value 5.19) (units m/s)(Time exm3))
  (tension (magnitude 1800)(units N)(applied-to b1)(applied-by s1)(Time exm3))
  (relative-position (obj s1)(wrt b1)(direction 20)(Time exm3))
  (circular-motion (obj b1)(angle 0)(Time exm3)) ;; give dir of accel
  (near-planet (planet earth) (Time exm3))
  (revolution-radius (obj b1)(value 1.0) (units m) (Time exm3))
  (goal-problem (is find-mass) (applied-to b1) (Time exm3)))
```

**Equations Generated**

```
(= mass_b1_exm3_1-1 unknown1)
(= speed_inst_b1_exm3_1-1 5.19)
(= r_b1_exm3_1-1 1.0)
(= dir_a_b1_exm3_1-1 0)
(= dir_t_b1_s1_exm3_1-1 20)
(= mag_t_b1_s1_exm3_1-1 1800)
(= t_xc_b1_s1_exm3_2-2 (* mag_t_b1_s1_exm3_1-1 (cos (- 20 20))))
(= T_b1_s1_exm3_1-1 t_xc_b1_s1_exm3_2-2)
(= t_xc_b1_s1_exm3_2-1 (* mag_t_b1_s1_exm3_1-1 (cos (- 20 0))))
(= T_b1_s1_exm3_1-1 t_xc_b1_s1_exm3_2-1)
(= t_yc_b1_s1_exm3_2-2 (* mag_t_b1_s1_exm3_1-1 (sin (- 20 20))))
(= t_yc_b1_s1_exm3_2-1 (* mag_t_b1_s1_exm3_1-1 (sin (- 20 0))))
(= mag_w_b1_earth_exm3_1-1  (* mass_b1_exm3_1-1 g))
(= dir_w_b1_earth_exm3_1-1 270)
(= w_xc_b1_earth_exm3_2-2 (* mag_w_b1_earth_exm3_1-1 (cos (- 270 20))))
(= w_xc_b1_earth_exm3_2-1 (* mag_w_b1_earth_exm3_1-1 (cos (- 270 0))))
(= w_yc_b1_earth_exm3_2-2 (* mag_w_b1_earth_exm3_1-1 (sin (- 270 20))))
(= W_b1_earth_exm3_1-1 w_yc_b1_earth_exm3_2-2)
(= w_yc_b1_earth_exm3_2-1 (* mag_w_b1_earth_exm3_1-1 (sin (- 270 0))))
(= W_b1_earth_exm3_1-1 w_yc_b1_earth_exm3_2-1)
(= mag_v_b1_exm3_1-1 5.19)
(= mag_a_b1_exm3_1-1
        (/ (* mag_v_b1_exm3_1-1 mag_v_b1_exm3_1-1 ) r_b1_exm3_1-1 ))
```

```
(= a_xc_b1_exm3_2-2 (* mag_a_b1_exm3_1-1 (cos (- 0 20))))
(= a_yc_b1_exm3_2-2 (* mag_a_b1_exm3_1-1 (sin (- 0 20))))
(= A_b1_exm3_1-1_2-1 a_xc_b1_exm3_2-1)
(= a_xc_b1_exm3_2-1 (* mag_a_b1_exm3_1-1 (cos (- 0 0))))
(= a_yc_b1_exm3_2-1 (* mag_a_b1_exm3_1-1 (sin (- 0 0))))
(= g 9.8)
(= (+ t_yc_b1_s1_exm3_2-1 w_yc_b1_earth_exm3_2-1)
        (* mass_b1_exm3_1-1 a_yc_b1_exm3_2-1))
(= (+ t_xc_b1_s1_exm3_2-1 w_xc_b1_earth_exm3_2-1)
        (* mass_b1_exm3_1-1 a_xc_b1_exm3_2-1))
(= (+ t_yc_b1_s1_exm3_2-2 w_yc_b1_earth_exm3_2-2)
        (* mass_b1_exm3_1-1 a_yc_b1_exm3_2-2))
(= (+ t_xc_b1_s1_exm3_2-2 w_xc_b1_earth_exm3_2-2)
        (* mass_b1_exm3_1-1 a_xc_b1_exm3_2-2))
```

**Circular Motion Problem 4**

A circular automobile racetrack is banked at an angle of 20 degrees and no friction between the road and the tires is required when a car travels at a certain speed. If the radius of the track is 400 m, determine the velocity.

**Givens**

```
(deffacts exm4 "Facts for exercise rotary motion problem m4"
  (problem exm4)
  (view horizontal)
  (surface (obj b2)(fixed yes)(incline 20))
  (touching (obj1 b1)(obj2 b2) (Time exm4))
  (circular-motion (obj b1)(angle 180)(Time exm4))
  (near-planet (planet earth) (Time exm4))
  (revolution-radius (obj b1)(value 400.0) (units m) (Time exm4))
  (goal-problem (is find-inst-vel) (applied-to b1) (Time exm4)))
```

**Equations Generated**

```
(= mass_b1_exm4_1-1 unknown1)
(= r_b1_exm4_1-1 400.0)
(= dir_a_b1_exm4_1-1 180)
(= dir_n_b1_b2_exm4_1-1 theta_n_b1_b2_exm4_1-1)
(= mag_n_b1_b2_exm4_1-1 unknown2)
(= theta_n_b1_b2_exm4_1-1 (+ 20 90))
(= theta_n_b1_b2_exm4_1-1 (+ 20 90))
(= n_xc_b1_b2_exm4_2-2
        (* mag_n_b1_b2_exm4_1-1 (cos (- theta_n_b1_b2_exm4_1-1 0))))
(= n_xc_b1_b2_exm4_2-1
        (* mag_n_b1_b2_exm4_1-1 (cos (- theta_n_b1_b2_exm4_1-1 20))))
(= n_yc_b1_b2_exm4_2-2
        (* mag_n_b1_b2_exm4_1-1 (sin (- theta_n_b1_b2_exm4_1-1 0))))
(= n_yc_b1_b2_exm4_2-1
        (* mag_n_b1_b2_exm4_1-1 (sin (- theta_n_b1_b2_exm4_1-1 20))))
(= mag_w_b1_earth_exm4_1-1  (* mass_b1_exm4_1-1 g))
(= dir_w_b1_earth_exm4_1-1 270)
(= w_xc_b1_earth_exm4_2-2 (* mag_w_b1_earth_exm4_1-1 (cos (- 270 0))))
(= w_xc_b1_earth_exm4_2-1 (* mag_w_b1_earth_exm4_1-1 (cos (- 270 20))))
(= w_yc_b1_earth_exm4_2-2 (* mag_w_b1_earth_exm4_1-1 (sin (- 270 0))))
(= W_b1_earth_exm4_1-1 w_yc_b1_earth_exm4_2-2)
(= w_yc_b1_earth_exm4_2-1 (* mag_w_b1_earth_exm4_1-1 (sin (- 270 20))))
(= W_b1_earth_exm4_1-1 w_yc_b1_earth_exm4_2-1)
(= mag_v_b1_exm4_1-1 unknown3)
```

```
(= mag_a_b1_exm4_1-1
        (/ (* mag_v_b1_exm4_1-1 mag_v_b1_exm4_1-1 ) r_b1_exm4_1-1 ))
(= A_b1_exm4_1-1_2-2 a_xc_b1_exm4_2-2)
(= a_xc_b1_exm4_2-2 (* mag_a_b1_exm4_1-1 (cos (- 180 0))))
(= a_yc_b1_exm4_2-2 (* mag_a_b1_exm4_1-1 (sin (- 180 0))))
(= a_xc_b1_exm4_2-1 (* mag_a_b1_exm4_1-1 (cos (- 180 20))))
(= a_yc_b1_exm4_2-1 (* mag_a_b1_exm4_1-1 (sin (- 180 20))))
(= g 9.8)
(= (+ n_yc_b1_b2_exm4_2-1 w_yc_b1_earth_exm4_2-1)
        (* mass_b1_exm4_1-1 a_yc_b1_exm4_2-1))
(= (+ n_xc_b1_b2_exm4_2-1 w_xc_b1_earth_exm4_2-1)
        (* mass_b1_exm4_1-1 a_xc_b1_exm4_2-1))
(= (+ n_yc_b1_b2_exm4_2-2 w_yc_b1_earth_exm4_2-2)
        (* mass_b1_exm4_1-1 a_yc_b1_exm4_2-2))
(= (+ n_xc_b1_b2_exm4_2-2 w_xc_b1_earth_exm4_2-2)
        (* mass_b1_exm4_1-1 a_xc_b1_exm4_2-2))
```

**Circular Motion Problem 5**

A circular automobile racetrack is banked at an angle theta such that no friction between the road and the tires is required when a car travels at 30.0 m/s. If the radius of the track is 400 m, determine theta.

**Givens**

```
(deffacts exm5 "Facts for exercise rotary motion problem m5"
  (problem exm5)
  (view horizontal)
  (track b2)
  (speed (obj b1) (type inst) (value 30.0) (units m/s) (Time exm5))
  (touching (obj1 b1)(obj2 b2) (Time exm5))
  (circular-motion (obj b1)(angle 180)(Time exm5))
  (near-planet (planet earth) (Time exm5))
  (revolution-radius (obj b1)(value 400.0) (units m) (Time exm5))
  (goal-problem (is find-incline) (applied-to b1) (Time exm5)))
```

**Equations Generated**

```
(= speed_inst_b1_exm5_1-1 30.0)
(= theta unknown1 )
(= mass_b1_exm5_1-1 unknown2)
(= r_b1_exm5_1-1 400.0)
(= dir_a_b1_exm5_1-1 180)
(= dir_n_b1_b2_exm5_1-1 theta_n_b1_b2_exm5_1-1)
(= mag_n_b1_b2_exm5_1-1 unknown3)
(= theta_n_b1_b2_exm5_1-1 (+ theta 90))
(= n_xc_b1_b2_exm5_2-1
        (* mag_n_b1_b2_exm5_1-1 (cos (- theta_n_b1_b2_exm5_1-1 0))))
(= n_yc_b1_b2_exm5_2-1
        (* mag_n_b1_b2_exm5_1-1 (sin (- theta_n_b1_b2_exm5_1-1 0))))
(= mag_w_b1_earth_exm5_1-1  (* mass_b1_exm5_1-1 g))
(= dir_w_b1_earth_exm5_1-1 270)
(= w_xc_b1_earth_exm5_2-1 (* mag_w_b1_earth_exm5_1-1 (cos (- 270 0))))
(= w_yc_b1_earth_exm5_2-1 (* mag_w_b1_earth_exm5_1-1 (sin (- 270 0))))
(= W_b1_earth_exm5_1-1 w_yc_b1_earth_exm5_2-1)
(= mag_v_b1_exm5_1-1 30.0)
(= mag_a_b1_exm5_1-1
        (/ (* mag_v_b1_exm5_1-1  mag_v_b1_exm5_1-1 ) r_b1_exm5_1-1 ))
(= A_b1_exm5_1-1_2-1 a_xc_b1_exm5_2-1)
(= a_xc_b1_exm5_2-1 (* mag_a_b1_exm5_1-1 (cos (- 180 0))))
(= a_yc_b1_exm5_2-1 (* mag_a_b1_exm5_1-1 (sin (- 180 0))))
(= g 9.8)
(= (+ n_yc_b1_b2_exm5_2-1 w_yc_b1_earth_exm5_2-1)
```

```
        (* mass_b1_exm5_1-1 a_yc_b1_exm5_2-1))
(= (+ n_xc_b1_b2_exm5_2-1 w_xc_b1_earth_exm5_2-1)
        (* mass_b1_exm5_1-1 a_xc_b1_exm5_2-1))
```

**Circular Motion Problem 6**

A roller coaster car goes through a loop-de-loop which has a radius of 20 m. If the normal force on the car is equal to its weight when the car is at the top of the loop, how fast is the car traveling?

**Givens**

```
(deffacts exm6 "Facts for exercise rotary motion problem m6"
  (problem exm6)
  (view horizontal)
  (surface (obj b2) (fixed yes) (incline 180))
  (circular-motion (obj b1)(angle 270) (Time exm6))
  (touching (obj1 b1) (obj2 b2) (Time exm6))
  (ratio (quantity1 N)(quantity2 W) (value 1) (Time exm6))
  (near-planet (planet earth) (Time exm6))
  (revolution-radius (obj b1)(value 20)(units m)(Time exm6))
  (goal-problem (is find-inst-vel) (applied-to b1) (Time exm6)))
```

**Equations Generated**

```
(= mass_b1_exm6_1-1 unknown1)
(= r_b1_exm6_1-1 20)
(= dir_a_b1_exm6_1-1 270)
(= dir_n_b1_b2_exm6_1-1 theta_n_b1_b2_exm6_1-1)
(= mag_n_b1_b2_exm6_1-1 unknown2)
(= theta_n_b1_b2_exm6_1-1 (+ 180 90))
(= n_xc_b1_b2_exm6_2-1
        (* mag_n_b1_b2_exm6_1-1 (cos (- theta_n_b1_b2_exm6_1-1 0))))
(= n_yc_b1_b2_exm6_2-1
        (* mag_n_b1_b2_exm6_1-1 (sin (- theta_n_b1_b2_exm6_1-1 0))))
(= mag_w_b1_earth_exm6_1-1  (* mass_b1_exm6_1-1 g))
(= mag_n_b1_b2_exm6_1-1 (* 1 mag_w_b1_earth_exm6_1-1))
(= dir_w_b1_earth_exm6_1-1 270)
(= w_xc_b1_earth_exm6_2-1 (* mag_w_b1_earth_exm6_1-1 (cos (- 270 0))))
(= w_yc_b1_earth_exm6_2-1 (* mag_w_b1_earth_exm6_1-1 (sin (- 270 0))))
(= W_b1_earth_exm6_1-1 w_yc_b1_earth_exm6_2-1)
(= mag_v_b1_exm6_1-1 unknown3)
(= mag_a_b1_exm6_1-1
        (/ (* mag_v_b1_exm6_1-1  mag_v_b1_exm6_1-1 ) r_b1_exm6_1-1 ))
(= a_xc_b1_exm6_2-1 (* mag_a_b1_exm6_1-1 (cos (- 270 0))))
(= A_b1_exm6_1-1_2-1 a_yc_b1_exm6_2-1)
(= a_yc_b1_exm6_2-1 (* mag_a_b1_exm6_1-1 (sin (- 270 0))))
(= g 9.8)
(= (+ n_yc_b1_b2_exm6_2-1 w_yc_b1_earth_exm6_2-1)
        (* mass_b1_exm6_1-1 a_yc_b1_exm6_2-1))
```

```
(= (+ n_xc_b1_b2_exm6_2-1 w_xc_b1_earth_exm6_2-1)
        (* mass_b1_exm6_1-1 a_xc_b1_exm6_2-1))
```

**Circular Motion Problem 7**

A 5 kg ball is attached to a 2 m rope which will break if the tension exceeds 100 N. If the ball is made to swing in a vertical circle, what is the maximum speed with which the ball can pass through the lowest point?

**Givens**

```
(deffacts exm7 "Facts for exercise rotary motion problem m7"
  (problem exm7)
  (mass (obj b1) (value 5.0) (units kg))
  (view horizontal)
  (string s1)
  (taut s1)
  (tied-to (object1 b1) (object2 s1) (Time exm7))
  (relative-position (obj s1) (wrt b1) (Time exm7) (direction 90))
  (tension (magnitude 100) (units N) (applied-to b1) (applied-by s1) (Time exm7))
  (circular-motion (obj b1)(angle 90) (Time exm7))
  (near-planet (planet earth) (Time exm7))
  (revolution-radius (obj b1)(value 2)(units m)(Time exm7))
  (goal-problem (is find-inst-vel) (applied-to b1) (Time exm7)))
```

**Equations Generated**

```
(= mass_b1_exm7_1-1 5.0)
(= r_b1_exm7_1-1 2)
(= dir_a_b1_exm7_1-1 90)
(= dir_t_b1_s1_exm7_1-1 90)
(= mag_t_b1_s1_exm7_1-1 100)
(= t_xc_b1_s1_exm7_2-1 (* mag_t_b1_s1_exm7_1-1 (cos (- 90 0))))
(= t_yc_b1_s1_exm7_2-1 (* mag_t_b1_s1_exm7_1-1 (sin (- 90 0))))
(= T_b1_s1_exm7_1-1 t_yc_b1_s1_exm7_2-1)
(= mag_w_b1_earth_exm7_1-1  (* mass_b1_exm7_1-1 g))
(= dir_w_b1_earth_exm7_1-1 270)
(= w_xc_b1_earth_exm7_2-1 (* mag_w_b1_earth_exm7_1-1 (cos (- 270 0))))
(= w_yc_b1_earth_exm7_2-1 (* mag_w_b1_earth_exm7_1-1 (sin (- 270 0))))
(= W_b1_earth_exm7_1-1 w_yc_b1_earth_exm7_2-1)
(= mag_v_b1_exm7_1-1 unknown1)
(= mag_a_b1_exm7_1-1
        (/ (* mag_v_b1_exm7_1-1  mag_v_b1_exm7_1-1 ) r_b1_exm7_1-1 ))
(= a_xc_b1_exm7_2-1 (* mag_a_b1_exm7_1-1 (cos (- 90 0))))
(= A_b1_exm7_1-1_2-1 a_yc_b1_exm7_2-1)
(= a_yc_b1_exm7_2-1 (* mag_a_b1_exm7_1-1 (sin (- 90 0))))
(= g 9.8)
(= (+ t_yc_b1_s1_exm7_2-1 w_yc_b1_earth_exm7_2-1)
        (* mass_b1_exm7_1-1 a_yc_b1_exm7_2-1))
```

**Circular Motion Problem 8**

A 20.0 kg mass is tied to a 3/16-in. Manila line. What is the speed the mass has if it is being whirled around in a horizontal circle with a 1.0 m radius, with the Manila line at a 35-degree angle below horizontal?

**Givens**

```
(deffacts exm8 "Facts for exercise rotary motion problem m8"
  (problem exm8)
  (mass (obj b1) (value 20.0) (units kg))
  (string s1)
  (taut s1)
  (tied-to (object1 b1)(object2 s1)(Time exm8))
  (relative-position (obj s1)(wrt b1)(direction 35)(Time exm8))
  (circular-motion (obj b1)(angle 0)(Time exm8)) ;; give dir of accel
  (near-planet (planet earth) (Time exm8))
  (revolution-radius (obj b1)(value 1.0) (units m) (Time exm8))
  (goal-problem (is find-inst-vel) (applied-to b1) (Time exm8)))
```

**Equations Generated**

```
(= mass_b1_exm8_1-1 20.0)
(= r_b1_exm8_1-1 1.0)
(= dir_a_b1_exm8_1-1 0)
(= dir_t_b1_s1_exm8_1-1 35)
(= mag_t_b1_s1_exm8_1-1 unknown1)
(= t_xc_b1_s1_exm8_2-2 (* mag_t_b1_s1_exm8_1-1 (cos (- 35 35))))
(= T_b1_s1_exm8_1-1 t_xc_b1_s1_exm8_2-2)
(= t_xc_b1_s1_exm8_2-1 (* mag_t_b1_s1_exm8_1-1 (cos (- 35 0))))
(= T_b1_s1_exm8_1-1 t_xc_b1_s1_exm8_2-1)
(= t_yc_b1_s1_exm8_2-2 (* mag_t_b1_s1_exm8_1-1 (sin (- 35 35))))
(= t_yc_b1_s1_exm8_2-1 (* mag_t_b1_s1_exm8_1-1 (sin (- 35 0))))
(= mag_w_b1_earth_exm8_1-1  (* mass_b1_exm8_1-1 g))
(= dir_w_b1_earth_exm8_1-1 270)
(= w_xc_b1_earth_exm8_2-2 (* mag_w_b1_earth_exm8_1-1 (cos (- 270 35))))
(= w_xc_b1_earth_exm8_2-1 (* mag_w_b1_earth_exm8_1-1 (cos (- 270 0))))
(= w_yc_b1_earth_exm8_2-2 (* mag_w_b1_earth_exm8_1-1 (sin (- 270 35))))
(= W_b1_earth_exm8_1-1 w_yc_b1_earth_exm8_2-2)
(= w_yc_b1_earth_exm8_2-1 (* mag_w_b1_earth_exm8_1-1 (sin (- 270 0))))
(= W_b1_earth_exm8_1-1 w_yc_b1_earth_exm8_2-1)
(= mag_v_b1_exm8_1-1 unknown2)
(= mag_a_b1_exm8_1-1
        (/ (* mag_v_b1_exm8_1-1  mag_v_b1_exm8_1-1 ) r_b1_exm8_1-1 ))
(= a_xc_b1_exm8_2-2 (* mag_a_b1_exm8_1-1 (cos (- 0 35))))
(= a_yc_b1_exm8_2-2 (* mag_a_b1_exm8_1-1 (sin (- 0 35))))
```

```
(= A_b1_exm8_1-1_2-1 a_xc_b1_exm8_2-1)
(= a_xc_b1_exm8_2-1 (* mag_a_b1_exm8_1-1 (cos (- 0 0))))
(= a_yc_b1_exm8_2-1 (* mag_a_b1_exm8_1-1 (sin (- 0 0))))
(= g 9.8)
(= (+ t_yc_b1_s1_exm8_2-1 w_yc_b1_earth_exm8_2-1)
        (* mass_b1_exm8_1-1 a_yc_b1_exm8_2-1))
(= (+ t_xc_b1_s1_exm8_2-1 w_xc_b1_earth_exm8_2-1)
        (* mass_b1_exm8_1-1 a_xc_b1_exm8_2-1))
(= (+ t_yc_b1_s1_exm8_2-2 w_yc_b1_earth_exm8_2-2)
        (* mass_b1_exm8_1-1 a_yc_b1_exm8_2-2))
(= (+ t_xc_b1_s1_exm8_2-2 w_xc_b1_earth_exm8_2-2)
        (* mass_b1_exm8_1-1 a_xc_b1_exm8_2-2))
```

**Circular Motion Problem 9**

A block is tied to a 3/16-in. Manila line, which has a tension of 1800 N. It moves in a horizontal circle of radius 1.0 m. The Manila line is at a 20-degree angle below horizontal. What is the mass of the block and its speed?

**Givens**

```
(deffacts exm9 "Facts for exercise rotary motion problem m9"
  (problem exm9)
  (string s1)
  (taut s1)
  (tied-to (object1 b1)(object2 s1)(Time exm9))
  (tension (magnitude 1800)(units N)(applied-to b1)(applied-by s1) (Time exm9))
  (relative-position (obj s1)(wrt b1)(direction 20)(Time exm9))
  (circular-motion (obj b1)(angle 0)(Time exm9))
  (near-planet (planet earth) (Time exm9))
  (revolution-radius (obj b1)(value 1.0) (units m) (Time exm9))
  (goal-problem (is find-mass) (applied-to b1) (Time exm9))
  (goal-problem (is find-inst-vel) (applied-to b1) (Time exm9)))
```

**Equations Generated**

```
(= mass_b1_exm9_1-1 unknown1)
(= r_b1_exm9_1-1 1.0)
(= dir_a_b1_exm9_1-1 0)
(= dir_t_b1_s1_exm9_1-1 20)
(= mag_t_b1_s1_exm9_1-1 1800)
(= t_xc_b1_s1_exm9_2-2 (* mag_t_b1_s1_exm9_1-1 (cos (- 20 20))))
(= T_b1_s1_exm9_1-1 t_xc_b1_s1_exm9_2-2)
(= t_xc_b1_s1_exm9_2-1 (* mag_t_b1_s1_exm9_1-1 (cos (- 20 0))))
(= T_b1_s1_exm9_1-1 t_xc_b1_s1_exm9_2-1)
(= t_yc_b1_s1_exm9_2-2 (* mag_t_b1_s1_exm9_1-1 (sin (- 20 20))))
(= t_yc_b1_s1_exm9_2-1 (* mag_t_b1_s1_exm9_1-1 (sin (- 20 0))))
(= mag_w_b1_earth_exm9_1-1  (* mass_b1_exm9_1-1 g))
(= dir_w_b1_earth_exm9_1-1 270)
(= w_xc_b1_earth_exm9_2-2 (* mag_w_b1_earth_exm9_1-1 (cos (- 270 20))))
(= w_xc_b1_earth_exm9_2-1 (* mag_w_b1_earth_exm9_1-1 (cos (- 270 0))))
(= w_yc_b1_earth_exm9_2-2 (* mag_w_b1_earth_exm9_1-1 (sin (- 270 20))))
(= W_b1_earth_exm9_1-1 w_yc_b1_earth_exm9_2-2)
(= w_yc_b1_earth_exm9_2-1 (* mag_w_b1_earth_exm9_1-1 (sin (- 270 0))))
(= W_b1_earth_exm9_1-1 w_yc_b1_earth_exm9_2-1)
(= mag_v_b1_exm9_1-1 unknown2)
(= mag_a_b1_exm9_1-1
        (/ (* mag_v_b1_exm9_1-1  mag_v_b1_exm9_1-1 ) r_b1_exm9_1-1 ))
(= a_xc_b1_exm9_2-2 (* mag_a_b1_exm9_1-1 (cos (- 0 20))))
```

```
(= a_yc_b1_exm9_2-2 (* mag_a_b1_exm9_1-1 (sin (- 0 20))))
(= A_b1_exm9_1-1_2-1 a_xc_b1_exm9_2-1)
(= a_xc_b1_exm9_2-1 (* mag_a_b1_exm9_1-1 (cos (- 0 0))))
(= a_yc_b1_exm9_2-1 (* mag_a_b1_exm9_1-1 (sin (- 0 0))))
(= g 9.8)
(= (+ t_yc_b1_s1_exm9_2-1 w_yc_b1_earth_exm9_2-1)
        (* mass_b1_exm9_1-1 a_yc_b1_exm9_2-1))
(= (+ t_xc_b1_s1_exm9_2-1 w_xc_b1_earth_exm9_2-1)
        (* mass_b1_exm9_1-1 a_xc_b1_exm9_2-1))
(= (+ t_yc_b1_s1_exm9_2-2 w_yc_b1_earth_exm9_2-2)
        (* mass_b1_exm9_1-1 a_yc_b1_exm9_2-2))
(= (+ t_xc_b1_s1_exm9_2-2 w_xc_b1_earth_exm9_2-2)
        (* mass_b1_exm9_1-1 a_xc_b1_exm9_2-2))
```

## Appendix B: Torque Problems

### Torque Problem 1

A horizontal 5m bar experiences a 8 N force straight down at its right end. It also experiences a 12 N force up and to the left, 30 degrees above the horizontal, at 2m from its left end. What is the resultant torque of the bar about its left end?

### Givens

```
(deffacts extor1 "Facts for exercise rotary motion problem tor1"
  (problem extor1)
  (view horizontal)
  (same-body b1 p0 p1 p2)
  (rotation-axis (obj b1)(point p0))
  (relative-position (obj p1)(wrt p0)(direction 0)(Time extor1))
  (relative-position (obj p2)(wrt p0)(direction 0)(Time extor1))
  (distance-btw-pts (pt1 p0)(pt2 p1)(value 2)(units m)(Time extor1))
  (distance-btw-pts (pt1 p0)(pt2 p2)(value 5)(units m)(Time extor1))
  (given-force (applied-to p1)(value 12)(angle 150)(units N)(applied-by nil)(Time extor1))
  (given-force (applied-to p2)(value 8)(angle 270)(units N)(applied-by nil)(Time extor1))
  (goal-problem (is find-torque)(applied-to b1)(Time extor1)))
```

### Generated Equations

```
(= dir_F_GEN1_b1_p2_extor1_1-1 270)
(= mag_F_GEN1_b1_p2_extor1_1-1 8)
(= dir_F_GEN2_b1_p1_extor1_1-1 150)
(= mag_F_GEN2_b1_p1_extor1_1-1 12)
(= mag_disp_p0_p1_extor1_1-1 2)
(= dir_disp_p0_p1_extor1 0)
(= mag_Trq_GEN2_b1_p1_extor1_1-1_2-1_3-1
(* mag_F_GEN2_b1_p1_extor1_1-1 mag_disp_p0_p1_extor1_1-1
(sin (- dir_disp_p0_p1_extor1 dir_F_GEN2_b1_p1_extor1_1-1 0 ))))
(= mag_disp_p0_p2_extor1_1-1 5)
(= dir_disp_p0_p2_extor1 0)
(= mag_Trq_GEN1_b1_p2_extor1_1-1_2-1_3-1
        (* mag_F_GEN1_b1_p2_extor1_1-1 mag_disp_p0_p2_extor1_1-1
        (sin (- dir_disp_p0_p2_extor1 dir_F_GEN1_b1_p2_extor1_1-1 0 ))))
(= dist_btw_p0_p2_extor1 5)
(= dist_btw_p0_p1_extor1 2)
(= gen2_xc_b1_extor1_2-1 (* mag_F_GEN2_b1_p1_extor1_1-1 (cos (- 150 0))))
(= gen1_xc_b1_extor1_2-1 (* mag_F_GEN1_b1_p2_extor1_1-1 (cos (- 270 0))))
(= DISP_p0_p1_extor1_1-1 disp_xc_b1_p1_extor1_2-1)
(= disp_xc_b1_p1_extor1_2-1 (* mag_disp_p0_p1_extor1_1-1 (cos (- 0 0))))
```

```
(= DISP_p0_p2_extor1_1-1 disp_xc_b1_p2_extor1_2-1)
(= disp_xc_b1_p2_extor1_2-1 (* mag_disp_p0_p2_extor1_1-1 (cos (- 0 0))))
(= gen2_yc_b1_extor1_2-1 (* mag_F_GEN2_b1_p1_extor1_1-1 (sin (- 150 0))))
(= F_GEN1_b1_p2_extor1_1-1 gen1_yc_b1_extor1_2-1)
(= gen1_yc_b1_extor1_2-1 (* mag_F_GEN1_b1_p2_extor1_1-1 (sin (- 270 0))))
(= disp_yc_b1_p1_extor1_2-1 (* mag_disp_p0_p1_extor1_1-1 (sin (- 0 0))))
(= mag_Trq_GEN2_b1_p1_extor1_1-1_2-1_3-2
      (* disp_xc_b1_p1_extor1_2-1 gen2_yc_b1_extor1_2-1))
(= disp_yc_b1_p2_extor1_2-1 (* mag_disp_p0_p2_extor1_1-1 (sin (- 0 0))))
(= mag_Trq_GEN1_b1_p2_extor1_1-1_2-1_3-2
      (* disp_xc_b1_p2_extor1_2-1 gen1_yc_b1_extor1_2-1))
(= mag_restrq_b1_extor1_1-1_2-1 unknown1)
(= mag_restrq_b1_extor1_1-1_2-1 (+ mag_Trq_GEN2_b1_p1_extor1_1-1_2-1_3-2
      mag_Trq_GEN1_b1_p2_extor1_1-1_2-1_3-2))
(= mag_restrq_b1_extor1_1-1_2-1 (+ mag_Trq_GEN2_b1_p1_extor1_1-1_2-1_3-1
      mag_Trq_GEN1_b1_p2_extor1_1-1_2-1_3-1))
```

**Torque Problem 2**

A square metal plate 0.18 m on each side is pivoted about its central axis, perpendicular to the plate. Calculate the net torque about this axis due to three forces: 16 N down, applied to top left corner, 24 N down, applied to top right corner, and 18 N, directed to the right at 45 degrees above the horizontal, applied to bottom right corner.

**Givens**

```
(deffacts extor2 "Facts for exercise rotary motion problem tor2"
  (problem extor2)
  (view horizontal)
  (same-body b1 p0 p1 p2 p3)
  (rotation-axis (obj b1)(point p0))
  (relative-position (obj p1)(wrt p0)(direction 135)(Time extor2))
  (relative-position (obj p2)(wrt p0)(direction 45)(Time extor2))
  (relative-position (obj p3)(wrt p0)(direction 315)(Time extor2))
  (distance-btw-pts (pt1 p0)(pt2 p1)(value .127)(units m)(Time extor2))
  (distance-btw-pts (pt1 p0)(pt2 p2)(value .127)(units m)(Time extor2))
  (distance-btw-pts (pt1 p0)(pt2 p3)(value .127)(units m)(Time extor2))
  (given-force (applied-to p1)(value 16)(angle 270)(units N)(applied-by nil)(Time extor2))
  (given-force (applied-to p2)(value 24)(angle 180)(units N)(applied-by nil)(Time extor2))
  (given-force (applied-to p3)(value 18)(angle 45)(units N)(applied-by nil)(Time extor2))
  (goal-problem (is find-torque)(applied-to b1)(Time extor2)))
```

**Equations Generated**

```
(= dir_F_GEN1_b1_p3_extor2_1-1 45)
(= mag_F_GEN1_b1_p3_extor2_1-1 18)
(= dir_F_GEN2_b1_p2_extor2_1-1 180)
(= mag_F_GEN2_b1_p2_extor2_1-1 24)
(= dir_F_GEN3_b1_p1_extor2_1-1 270)
(= mag_F_GEN3_b1_p1_extor2_1-1 16)
(= mag_disp_p0_p1_extor2_1-1 0.127)
(= dir_disp_p0_p1_extor2 135)
(= mag_Trq_GEN3_b1_p1_extor2_1-1_2-1_3-1 (* mag_F_GEN3_b1_p1_extor2_1-1
        mag_disp_p0_p1_extor2_1-1 (sin (- dir_disp_p0_p1_extor2
        dir_F_GEN3_b1_p1_extor2_1-1 0 ))))
(= mag_Trq_GEN3_b1_p1_extor2_1-1_2-2_3-1 (* mag_F_GEN3_b1_p1_extor2_1-1
        mag_disp_p0_p1_extor2_1-1 (sin (- dir_disp_p0_p1_extor2
         dir_F_GEN3_b1_p1_extor2_1-1 45 ))))
(= mag_disp_p0_p2_extor2_1-1 0.127)
(= dir_disp_p0_p2_extor2 45)
(= mag_Trq_GEN2_b1_p2_extor2_1-1_2-2_3-1 (* mag_F_GEN2_b1_p2_extor2_1-1
        mag_disp_p0_p2_extor2_1-1 (sin (- dir_disp_p0_p2_extor2
         dir_F_GEN2_b1_p2_extor2_1-1 45 ))))
```

```
(= mag_Trq_GEN2_b1_p2_extor2_1-1_2-1_3-1 (* mag_F_GEN2_b1_p2_extor2_1-1
        mag_disp_p0_p2_extor2_1-1 (sin (- dir_disp_p0_p2_extor2
        dir_F_GEN2_b1_p2_extor2_1-1 0 ))))
(= mag_disp_p0_p3_extor2_1-1 0.127)
(= dir_disp_p0_p3_extor2 315)
(= mag_Trq_GEN1_b1_p3_extor2_1-1_2-2_3-1 (* mag_F_GEN1_b1_p3_extor2_1-1
        mag_disp_p0_p3_extor2_1-1 (sin (- dir_disp_p0_p3_extor2
        dir_F_GEN1_b1_p3_extor2_1-1 45 ))))
(= mag_Trq_GEN1_b1_p3_extor2_1-1_2-1_3-1 (* mag_F_GEN1_b1_p3_extor2_1-1
        mag_disp_p0_p3_extor2_1-1 (sin (- dir_disp_p0_p3_extor2
        dir_F_GEN1_b1_p3_extor2_1-1 0 ))))
(= dist_btw_p0_p3_extor2 0.127)
(= dist_btw_p0_p2_extor2 0.127)
(= dist_btw_p0_p1_extor2 0.127)
(= gen3_xc_b1_extor2_2-2 (* mag_F_GEN3_b1_p1_extor2_1-1 (cos (- 270 45))))
(= gen3_xc_b1_extor2_2-1 (* mag_F_GEN3_b1_p1_extor2_1-1 (cos (- 270 0))))
(= F_GEN2_b1_p2_extor2_1-1 gen2_xc_b1_extor2_2-2)
(= gen2_xc_b1_extor2_2-2 (* mag_F_GEN2_b1_p2_extor2_1-1 (cos (- 180 45))))
(= F_GEN2_b1_p2_extor2_1-1 gen2_xc_b1_extor2_2-1)
(= gen2_xc_b1_extor2_2-1 (* mag_F_GEN2_b1_p2_extor2_1-1 (cos (- 180 0))))
(= F_GEN1_b1_p3_extor2_1-1 gen1_xc_b1_extor2_2-2)
(= gen1_xc_b1_extor2_2-2 (* mag_F_GEN1_b1_p3_extor2_1-1 (cos (- 45 45))))
(= F_GEN1_b1_p3_extor2_1-1 gen1_xc_b1_extor2_2-1)
(= gen1_xc_b1_extor2_2-1 (* mag_F_GEN1_b1_p3_extor2_1-1 (cos (- 45 0))))
(= disp_xc_b1_p1_extor2_2-2 (* mag_disp_p0_p1_extor2_1-1 (cos (- 135 45))))
(= disp_xc_b1_p1_extor2_2-1 (* mag_disp_p0_p1_extor2_1-1 (cos (- 135 0))))
(= DISP_p0_p2_extor2_1-1 disp_xc_b1_p2_extor2_2-2)
(= disp_xc_b1_p2_extor2_2-2 (* mag_disp_p0_p2_extor2_1-1 (cos (- 45 45))))
(= DISP_p0_p2_extor2_1-1 disp_xc_b1_p2_extor2_2-1)
(= disp_xc_b1_p2_extor2_2-1 (* mag_disp_p0_p2_extor2_1-1 (cos (- 45 0))))
(= disp_xc_b1_p3_extor2_2-2 (* mag_disp_p0_p3_extor2_1-1 (cos (- 315 45))))
(= disp_xc_b1_p3_extor2_2-1 (* mag_disp_p0_p3_extor2_1-1 (cos (- 315 0))))
(= F_GEN3_b1_p1_extor2_1-1 gen3_yc_b1_extor2_2-2)
(= gen3_yc_b1_extor2_2-2 (* mag_F_GEN3_b1_p1_extor2_1-1 (sin (- 270 45))))
(= F_GEN3_b1_p1_extor2_1-1 gen3_yc_b1_extor2_2-1)
(= gen3_yc_b1_extor2_2-1 (* mag_F_GEN3_b1_p1_extor2_1-1 (sin (- 270 0))))
(= gen2_yc_b1_extor2_2-2 (* mag_F_GEN2_b1_p2_extor2_1-1 (sin (- 180 45))))
(= gen2_yc_b1_extor2_2-1 (* mag_F_GEN2_b1_p2_extor2_1-1 (sin (- 180 0))))
(= gen1_yc_b1_extor2_2-2 (* mag_F_GEN1_b1_p3_extor2_1-1 (sin (- 45 45))))
(= gen1_yc_b1_extor2_2-1 (* mag_F_GEN1_b1_p3_extor2_1-1 (sin (- 45 0))))
(= DISP_p0_p1_extor2_1-1 disp_yc_b1_p1_extor2_2-2)
(= disp_yc_b1_p1_extor2_2-2 (* mag_disp_p0_p1_extor2_1-1 (sin (- 135 45))))
```

```
(= mag_Trq_GEN3_b1_p1_extor2_1-1_2-2_3-2 (* disp_yc_b1_p1_extor2_2-2
        gen3_xc_b1_extor2_2-2))
(= DISP_p0_p1_extor2_1-1 disp_yc_b1_p1_extor2_2-1)
(= disp_yc_b1_p1_extor2_2-1 (* mag_disp_p0_p1_extor2_1-1 (sin (- 135 0))))
(= mag_Trq_GEN3_b1_p1_extor2_1-1_2-1_3-2 (* disp_xc_b1_p1_extor2_2-1
        gen3_yc_b1_extor2_2-1))
(= disp_yc_b1_p2_extor2_2-2 (* mag_disp_p0_p2_extor2_1-1 (sin (- 45 45))))
(= mag_Trq_GEN2_b1_p2_extor2_1-1_2-2_3-2 (* disp_xc_b1_p2_extor2_2-2
        gen2_yc_b1_extor2_2-2))
(= disp_yc_b1_p2_extor2_2-1 (* mag_disp_p0_p2_extor2_1-1 (sin (- 45 0))))
(= mag_Trq_GEN2_b1_p2_extor2_1-1_2-1_3-2 (* disp_yc_b1_p2_extor2_2-1
        gen2_xc_b1_extor2_2-1))
(= DISP_p0_p3_extor2_1-1 disp_yc_b1_p3_extor2_2-2)
(= disp_yc_b1_p3_extor2_2-2 (* mag_disp_p0_p3_extor2_1-1 (sin (- 315 45))))
(= mag_Trq_GEN1_b1_p3_extor2_1-1_2-2_3-2 (+ (* disp_xc_b1_p3_extor2_2-2
        gen1_yc_b1_extor2_2-2) (* disp_yc_b1_p3_extor2_2-2
        gen1_xc_b1_extor2_2-2)))
(= DISP_p0_p3_extor2_1-1 disp_yc_b1_p3_extor2_2-1)
(= disp_yc_b1_p3_extor2_2-1 (* mag_disp_p0_p3_extor2_1-1 (sin (- 315 0))))
(= mag_Trq_GEN1_b1_p3_extor2_1-1_2-1_3-2 (+ (* disp_xc_b1_p3_extor2_2-1
        gen1_yc_b1_extor2_2-1) (* disp_yc_b1_p3_extor2_2-1
        gen1_xc_b1_extor2_2-1)))
(= mag_restrq_b1_extor2_1-1_2-2 unknown1)
(= mag_restrq_b1_extor2_1-1_2-2 (+ (+ mag_Trq_GEN3_b1_p1_extor2_1-1_2-2_3-2
        mag_Trq_GEN2_b1_p2_extor2_1-1_2-2_3-2)
        mag_Trq_GEN1_b1_p3_extor2_1-1_2-2_3-2) )
(= mag_restrq_b1_extor2_1-1_2-2 (+ (+ mag_Trq_GEN3_b1_p1_extor2_1-1_2-2_3-1
        mag_Trq_GEN2_b1_p2_extor2_1-1_2-2_3-1)
        mag_Trq_GEN1_b1_p3_extor2_1-1_2-2_3-1) )
(= mag_restrq_b1_extor2_1-1_2-1 unknown2)
(= mag_restrq_b1_extor2_1-1_2-1 (+ (+ mag_Trq_GEN3_b1_p1_extor2_1-1_2-1_3-2
        mag_Trq_GEN2_b1_p2_extor2_1-1_2-1_3-2)
        mag_Trq_GEN1_b1_p3_extor2_1-1_2-1_3-2) )
(= mag_restrq_b1_extor2_1-1_2-1 (+ (+ mag_Trq_GEN3_b1_p1_extor2_1-1_2-1_3-1
        mag_Trq_GEN2_b1_p2_extor2_1-1_2-1_3-1)
        mag_Trq_GEN1_b1_p3_extor2_1-1_2-1_3-1) )
```

**Torque Problem 3**

A 5m bar at 50 degrees above horizontal experiences a 8 N force straight down at its right end. It also experiences a 12 N force at 150 degrees standard, at 2m from its left end. What is the resultant torque of the bar about its left end?

**Givens**

```
(deffacts extor3 "Facts for exercise rotary motion problem tor3"
  (problem extor3)
  (view horizontal)
  (same-body b1 p0 p1 p2)
  (rotation-axis (obj b1)(point p0))
  (relative-position (obj p1)(wrt p0)(direction 50)(Time extor3))
  (relative-position (obj p2)(wrt p0)(direction 50)(Time extor3))
  (distance-btw-pts (pt1 p0)(pt2 p1)(value 2)(units m)(Time extor3))
  (distance-btw-pts (pt1 p0)(pt2 p2)(value 5)(units m)(Time extor3))
  (given-force (applied-to p1)(value 12)(angle 150)(units N)(applied-by nil)(Time extor3))
  (given-force (applied-to p2)(value 8)(angle 270)(units N)(applied-by nil)(Time extor3))
  (goal-problem (is find-torque)(applied-to b1)(Time extor3)))
```

**Equations Generated**

```
(= dir_F_GEN1_b1_p2_extor3_1-1 270)
(= mag_F_GEN1_b1_p2_extor3_1-1 8)
(= dir_F_GEN2_b1_p1_extor3_1-1 150)
(= mag_F_GEN2_b1_p1_extor3_1-1 12)
(= mag_disp_p0_p1_extor3_1-1 2)
(= dir_disp_p0_p1_extor3 50)
(= mag_Trq_GEN2_b1_p1_extor3_1-1_2-1_3-1
        (* mag_F_GEN2_b1_p1_extor3_1-1 mag_disp_p0_p1_extor3_1-1
        (sin (- dir_disp_p0_p1_extor3 dir_F_GEN2_b1_p1_extor3_1-1 0 ))))
(= mag_Trq_GEN2_b1_p1_extor3_1-1_2-2_3-1
        (* mag_F_GEN2_b1_p1_extor3_1-1 mag_disp_p0_p1_extor3_1-1
        (sin (- dir_disp_p0_p1_extor3 dir_F_GEN2_b1_p1_extor3_1-1 50 ))))
(= mag_disp_p0_p2_extor3_1-1 5)
(= dir_disp_p0_p2_extor3 50)
(= mag_Trq_GEN1_b1_p2_extor3_1-1_2-2_3-1
        (* mag_F_GEN1_b1_p2_extor3_1-1 mag_disp_p0_p2_extor3_1-1
        (sin (- dir_disp_p0_p2_extor3 dir_F_GEN1_b1_p2_extor3_1-1 50 ))))
(= mag_Trq_GEN1_b1_p2_extor3_1-1_2-1_3-1
        (* mag_F_GEN1_b1_p2_extor3_1-1 mag_disp_p0_p2_extor3_1-1
        (sin (- dir_disp_p0_p2_extor3 dir_F_GEN1_b1_p2_extor3_1-1 0 ))))
(= dist_btw_p0_p2_extor3 5)
(= dist_btw_p0_p1_extor3 2)
(= gen2_xc_b1_extor3_2-2 (* mag_F_GEN2_b1_p1_extor3_1-1 (cos (- 150 50))))
```

```
(= gen2_xc_b1_extor3_2-1 (* mag_F_GEN2_b1_p1_extor3_1-1 (cos (- 150 0))))
(= gen1_xc_b1_extor3_2-2 (* mag_F_GEN1_b1_p2_extor3_1-1 (cos (- 270 50))))
(= gen1_xc_b1_extor3_2-1 (* mag_F_GEN1_b1_p2_extor3_1-1 (cos (- 270 0))))
(= DISP_p0_p1_extor3_1-1 disp_xc_b1_p1_extor3_2-2)
(= disp_xc_b1_p1_extor3_2-2 (* mag_disp_p0_p1_extor3_1-1 (cos (- 50 50))))
(= DISP_p0_p1_extor3_1-1 disp_xc_b1_p1_extor3_2-1)
(= disp_xc_b1_p1_extor3_2-1 (* mag_disp_p0_p1_extor3_1-1 (cos (- 50 0))))
(= DISP_p0_p2_extor3_1-1 disp_xc_b1_p2_extor3_2-2)
(= disp_xc_b1_p2_extor3_2-2 (* mag_disp_p0_p2_extor3_1-1 (cos (- 50 50))))
(= DISP_p0_p2_extor3_1-1 disp_xc_b1_p2_extor3_2-1)
(= disp_xc_b1_p2_extor3_2-1 (* mag_disp_p0_p2_extor3_1-1 (cos (- 50 0))))
(= gen2_yc_b1_extor3_2-2 (* mag_F_GEN2_b1_p1_extor3_1-1 (sin (- 150 50))))
(= gen2_yc_b1_extor3_2-1 (* mag_F_GEN2_b1_p1_extor3_1-1 (sin (- 150 0))))
(= F_GEN1_b1_p2_extor3_1-1 gen1_yc_b1_extor3_2-2)
(= gen1_yc_b1_extor3_2-2 (* mag_F_GEN1_b1_p2_extor3_1-1 (sin (- 270 50))))
(= F_GEN1_b1_p2_extor3_1-1 gen1_yc_b1_extor3_2-1)
(= gen1_yc_b1_extor3_2-1 (* mag_F_GEN1_b1_p2_extor3_1-1 (sin (- 270 0))))
(= disp_yc_b1_p1_extor3_2-2 (* mag_disp_p0_p1_extor3_1-1 (sin (- 50 50))))
(= mag_Trq_GEN2_b1_p1_extor3_1-1_2-2_3-2
        (+ (* disp_xc_b1_p1_extor3_2-2 gen2_yc_b1_extor3_2-2)
        (* disp_yc_b1_p1_extor3_2-2 gen2_xc_b1_extor3_2-2)))
(= disp_yc_b1_p1_extor3_2-1 (* mag_disp_p0_p1_extor3_1-1 (sin (- 50 0))))
(= mag_Trq_GEN2_b1_p1_extor3_1-1_2-1_3-2
        (+ (* disp_xc_b1_p1_extor3_2-1 gen2_yc_b1_extor3_2-1)
        (* disp_yc_b1_p1_extor3_2-1 gen2_xc_b1_extor3_2-1)))
(= disp_yc_b1_p2_extor3_2-2 (* mag_disp_p0_p2_extor3_1-1 (sin (- 50 50))))
(= mag_Trq_GEN1_b1_p2_extor3_1-1_2-2_3-2
        (+ (* disp_xc_b1_p2_extor3_2-2 gen1_yc_b1_extor3_2-2)
        (* disp_yc_b1_p2_extor3_2-2 gen1_xc_b1_extor3_2-2)))
(= disp_yc_b1_p2_extor3_2-1 (* mag_disp_p0_p2_extor3_1-1 (sin (- 50 0))))
(= mag_Trq_GEN1_b1_p2_extor3_1-1_2-1_3-2
        (* disp_xc_b1_p2_extor3_2-1 gen1_yc_b1_extor3_2-1))
(= mag_restrq_b1_extor3_1-1_2-2 unknown1)
(= mag_restrq_b1_extor3_1-1_2-2 (+ mag_Trq_GEN2_b1_p1_extor3_1-1_2-2_3-2
        mag_Trq_GEN1_b1_p2_extor3_1-1_2-2_3-2) )
(= mag_restrq_b1_extor3_1-1_2-2 (+ mag_Trq_GEN2_b1_p1_extor3_1-1_2-2_3-1
        mag_Trq_GEN1_b1_p2_extor3_1-1_2-2_3-1) )
(= mag_restrq_b1_extor3_1-1_2-1 unknown2)
(= mag_restrq_b1_extor3_1-1_2-1 (+ mag_Trq_GEN2_b1_p1_extor3_1-1_2-1_3-2
        mag_Trq_GEN1_b1_p2_extor3_1-1_2-1_3-2) )
(= mag_restrq_b1_extor3_1-1_2-1 (+ mag_Trq_GEN2_b1_p1_extor3_1-1_2-1_3-1
        mag_Trq_GEN1_b1_p2_extor3_1-1_2-1_3-1) )
```

**Torque Problem 4**

A square metal plate 0.18 m on each side is pivoted about its central axis, perpendicular to the plate. Calculate the net torque about this axis due to three forces: a force of unknown magnitude down, applied to top left corner, 24 N down, applied to top right corner, and 18 N, applied at 45 degrees to bottom right corner. If the net torque is 5.88 N * m, what is the magnitude of the force applied to the top left corner?

**Givens**

```
(deffacts extor4 "Facts for exercise rotary motion problem tor4"
  (problem extor4)
  (view horizontal)
  (same-body b1 p0 p1 p2 p3)
  (rotation-axis (obj b1)(point p0))
  (relative-position (obj p1)(wrt p0)(direction 135)(Time extor4))
  (relative-position (obj p2)(wrt p0)(direction 45)(Time extor4))
  (relative-position (obj p3)(wrt p0)(direction 315)(Time extor4))
  (distance-btw-pts (pt1 p0)(pt2 p1)(value .127)(units m)(Time extor4))
  (distance-btw-pts (pt1 p0)(pt2 p2)(value .127)(units m)(Time extor4))
  (distance-btw-pts (pt1 p0)(pt2 p3)(value .127)(units m)(Time extor4))
  (given-force (applied-to p1)(value nil)(angle 270)(units N)(applied-by nil)(Time extor4))
  (given-force (applied-to p2)(value 24)(angle 270)(units N)(applied-by nil)(Time extor4))
  (given-force (applied-to p3)(value 18)(angle 45)(units N)(applied-by nil)(Time extor4))
  (given-torque (applied-to b1)(value 5.88)(units Nm)(applied-at p0)(Time extor4))
  (goal-problem (is find-force-mag)(applied-to p1)(Time extor4)))
```

**Equations Generated**

```
(= mag_restrq_b1_extor4_1-1_2-1 5.88)
(= dir_F_GEN1_b1_p3_extor4_1-1 45)
(= mag_F_GEN1_b1_p3_extor4_1-1 18)
(= dir_F_GEN2_b1_p2_extor4_1-1 270)
(= mag_F_GEN2_b1_p2_extor4_1-1 24)
(= mag_F_GEN3_b1_p1_extor4_1-1 unknown1)
(= dir_F_GEN3_b1_p1_extor4_1-1 270)
(= mag_disp_p0_p1_extor4_1-1 0.127)
(= dir_disp_p0_p1_extor4 135)
(= mag_restrq_b1_extor4_1-1_2-2 5.88)
(= mag_disp_p0_p2_extor4_1-1 0.127)
(= dir_disp_p0_p2_extor4 45)
(= mag_disp_p0_p3_extor4_1-1 0.127)
(= dir_disp_p0_p3_extor4 315)
(= mag_Trq_GEN3_b1_p1_extor4_1-1_2-1_3-1 (* mag_F_GEN3_b1_p1_extor4_1-1
      mag_disp_p0_p1_extor4_1-1  (sin  (-  dir_disp_p0_p1_extor4
      dir_F_GEN3_b1_p1_extor4_1-1 0 ))))
```

```
(= mag_Trq_GEN3_b1_p1_extor4_1-1_2-2_3-1 (* mag_F_GEN3_b1_p1_extor4_1-1
      mag_disp_p0_p1_extor4_1-1  (sin  (-  dir_disp_p0_p1_extor4
      dir_F_GEN3_b1_p1_extor4_1-1 45 ))))
(= mag_Trq_GEN2_b1_p2_extor4_1-1_2-1_3-1 (* mag_F_GEN2_b1_p2_extor4_1-1
      mag_disp_p0_p2_extor4_1-1  (sin  (-  dir_disp_p0_p2_extor4
      dir_F_GEN2_b1_p2_extor4_1-1 0 ))))
(= mag_Trq_GEN2_b1_p2_extor4_1-1_2-2_3-1 (* mag_F_GEN2_b1_p2_extor4_1-1
      mag_disp_p0_p2_extor4_1-1  (sin  (-  dir_disp_p0_p2_extor4
      dir_F_GEN2_b1_p2_extor4_1-1 45 ))))
(= mag_Trq_GEN1_b1_p3_extor4_1-1_2-1_3-1 (* mag_F_GEN1_b1_p3_extor4_1-1
      mag_disp_p0_p3_extor4_1-1  (sin  (-  dir_disp_p0_p3_extor4
      dir_F_GEN1_b1_p3_extor4_1-1 0 ))))
(= mag_Trq_GEN1_b1_p3_extor4_1-1_2-2_3-1 (* mag_F_GEN1_b1_p3_extor4_1-1
      mag_disp_p0_p3_extor4_1-1 (sin (- dir_disp_p0_p3_extor4
      dir_F_GEN1_b1_p3_extor4_1-1 45 ))))
(= mag_restrq_b1_extor4_1-1_2-1 (+ (+ mag_Trq_GEN3_b1_p1_extor4_1-1_2-1_3-1
      mag_Trq_GEN2_b1_p2_extor4_1-1_2-1_3-1)
      mag_Trq_GEN1_b1_p3_extor4_1-1_2-1_3-1) )
(= dist_btw_p0_p3_extor4 0.127)
(= dist_btw_p0_p2_extor4 0.127)
(= dist_btw_p0_p1_extor4 0.127)
(= gen3_xc_b1_extor4_2-2 (* mag_F_GEN3_b1_p1_extor4_1-1 (cos (- 270 45))))
(= gen3_xc_b1_extor4_2-1 (* mag_F_GEN3_b1_p1_extor4_1-1 (cos (- 270 0))))
(= gen2_xc_b1_extor4_2-2 (* mag_F_GEN2_b1_p2_extor4_1-1 (cos (- 270 45))))
(= gen2_xc_b1_extor4_2-1 (* mag_F_GEN2_b1_p2_extor4_1-1 (cos (- 270 0))))
(= F_GEN1_b1_p3_extor4_1-1 gen1_xc_b1_extor4_2-2)
(= gen1_xc_b1_extor4_2-2 (* mag_F_GEN1_b1_p3_extor4_1-1 (cos (- 45 45))))
(= F_GEN1_b1_p3_extor4_1-1 gen1_xc_b1_extor4_2-1)
(= gen1_xc_b1_extor4_2-1 (* mag_F_GEN1_b1_p3_extor4_1-1 (cos (- 45 0))))
(= disp_xc_b1_p1_extor4_2-2 (* mag_disp_p0_p1_extor4_1-1 (cos (- 135 45))))
(= disp_xc_b1_p1_extor4_2-1 (* mag_disp_p0_p1_extor4_1-1 (cos (- 135 0))))
(= DISP_p0_p2_extor4_1-1 disp_xc_b1_p2_extor4_2-2)
(= disp_xc_b1_p2_extor4_2-2 (* mag_disp_p0_p2_extor4_1-1 (cos (- 45 45))))
(= DISP_p0_p2_extor4_1-1 disp_xc_b1_p2_extor4_2-1)
(= disp_xc_b1_p2_extor4_2-1 (* mag_disp_p0_p2_extor4_1-1 (cos (- 45 0))))
(= disp_xc_b1_p3_extor4_2-2 (* mag_disp_p0_p3_extor4_1-1 (cos (- 315 45))))
(= disp_xc_b1_p3_extor4_2-1 (* mag_disp_p0_p3_extor4_1-1 (cos (- 315 0))))
(= F_GEN3_b1_p1_extor4_1-1 gen3_yc_b1_extor4_2-2)
(= gen3_yc_b1_extor4_2-2 (* mag_F_GEN3_b1_p1_extor4_1-1 (sin (- 270 45))))
(= F_GEN3_b1_p1_extor4_1-1 gen3_yc_b1_extor4_2-1)
(= gen3_yc_b1_extor4_2-1 (* mag_F_GEN3_b1_p1_extor4_1-1 (sin (- 270 0))))
(= F_GEN2_b1_p2_extor4_1-1 gen2_yc_b1_extor4_2-2)
(= gen2_yc_b1_extor4_2-2 (* mag_F_GEN2_b1_p2_extor4_1-1 (sin (- 270 45))))
```

```
(= F_GEN2_b1_p2_extor4_1-1 gen2_yc_b1_extor4_2-1)
(= gen2_yc_b1_extor4_2-1 (* mag_F_GEN2_b1_p2_extor4_1-1 (sin (- 270 0))))
(= gen1_yc_b1_extor4_2-2 (* mag_F_GEN1_b1_p3_extor4_1-1 (sin (- 45 45))))
(= gen1_yc_b1_extor4_2-1 (* mag_F_GEN1_b1_p3_extor4_1-1 (sin (- 45 0))))
(= DISP_p0_p1_extor4_1-1 disp_yc_b1_p1_extor4_2-2)
(= disp_yc_b1_p1_extor4_2-2 (* mag_disp_p0_p1_extor4_1-1 (sin (- 135 45))))
(= mag_Trq_GEN3_b1_p1_extor4_1-1_2-2_3-2 (* disp_yc_b1_p1_extor4_2-2
      gen3_xc_b1_extor4_2-2))
(= DISP_p0_p1_extor4_1-1 disp_yc_b1_p1_extor4_2-1)
(= disp_yc_b1_p1_extor4_2-1 (* mag_disp_p0_p1_extor4_1-1 (sin (- 135 0))))
(= mag_Trq_GEN3_b1_p1_extor4_1-1_2-1_3-2 (* disp_xc_b1_p1_extor4_2-1
      gen3_yc_b1_extor4_2-1))
(= disp_yc_b1_p2_extor4_2-2 (* mag_disp_p0_p2_extor4_1-1 (sin (- 45 45))))
(= mag_Trq_GEN2_b1_p2_extor4_1-1_2-2_3-2 (* disp_yc_b1_p2_extor4_2-2
      gen2_xc_b1_extor4_2-2))
(= disp_yc_b1_p2_extor4_2-1 (* mag_disp_p0_p2_extor4_1-1 (sin (- 45 0))))
(= mag_Trq_GEN2_b1_p2_extor4_1-1_2-1_3-2 (* disp_xc_b1_p2_extor4_2-1
      gen2_yc_b1_extor4_2-1))
(= DISP_p0_p3_extor4_1-1 disp_yc_b1_p3_extor4_2-2)
(= disp_yc_b1_p3_extor4_2-2 (* mag_disp_p0_p3_extor4_1-1 (sin (- 315 45))))
(= mag_Trq_GEN1_b1_p3_extor4_1-1_2-2_3-2 (+ (* disp_xc_b1_p3_extor4_2-2
      gen1_yc_b1_extor4_2-2) (* disp_yc_b1_p3_extor4_2-2
      gen1_xc_b1_extor4_2-2)))
(= DISP_p0_p3_extor4_1-1 disp_yc_b1_p3_extor4_2-1)
(= disp_yc_b1_p3_extor4_2-1 (* mag_disp_p0_p3_extor4_1-1 (sin (- 315 0))))
(= mag_Trq_GEN1_b1_p3_extor4_1-1_2-1_3-2 (+ (* disp_xc_b1_p3_extor4_2-1
      gen1_yc_b1_extor4_2-1) (* disp_yc_b1_p3_extor4_2-1
      gen1_xc_b1_extor4_2-1)))
(= mag_restrq_b1_extor4_1-1_2-2 (+ (+ mag_Trq_GEN3_b1_p1_extor4_1-1_2-2_3-2
      mag_Trq_GEN2_b1_p2_extor4_1-1_2-2_3-2)
      mag_Trq_GEN1_b1_p3_extor4_1-1_2-2_3-2) )
```

**Torque Problem 5**

A cord is wrapped around the rim of a flywheel 0.4 m in radius, and a steady pull of 50.0 N is exerted on the cord. The wheel is mounted in frictionless bearings on a horizontal shaft through its center. The moment of inertia of the wheel is 4.0 kg m^2. What is the angular acceleration of the wheel?

**Givens**

```
(deffacts extor5 "Facts for exercise rotary motion problem 10-8"
  (problem extor5)
  (flywheel b1)
  (rotation-axis (obj b1)(point p0))
  (same-body b1 p0 p1)
  (moment-of-inertia (obj b1) (value 4) (units kg*m^2)(Time extor5))
  (rotation-radius (obj b1) (value .4) (units m) (Time extor5))
  (relative-position (obj p1)(wrt p0)(direction 0)(Time extor5))
  (distance-btw-pts (pt1 p0)(pt2 p1)(value .4)(units m)(Time extor5))
  (given-force (applied-to p1)(value 50)(angle 270)(units N)(applied-by s1)(Time extor5))
  (goal-problem (is find-ang-accel)(applied-to b1)(Time extor5)))
```

**Generated Equations**

```
(= dir_F_GEN1_b1_p1_extor5_1-1 270)
(= mag_F_GEN1_b1_p1_extor5_1-1 50)
(= mag_disp_p0_p1_extor5_1-1 0.4)
(= dir_disp_p0_p1_extor5 0)
(= mag_Trq_GEN1_b1_p1_extor5_1-1_2-1_3-1 (* mag_F_GEN1_b1_p1_extor5_1-1
      mag_disp_p0_p1_extor5_1-1 (sin (- dir_disp_p0_p1_extor5
      dir_F_GEN1_b1_p1_extor5_1-1 0 ))))
(= mag_alpha_b1_extor5_1-1 unknown1)
(= I_b1_extor5_1-1 4)
(= dist_btw_p0_p1_extor5 0.4)
(= gen1_xc_b1_s1_extor5_2-1 (* mag_F_GEN1_b1_p1_extor5_1-1 (cos (- 270 0))))
(= DISP_p0_p1_extor5_1-1 disp_xc_b1_p1_extor5_2-1)
(= disp_xc_b1_p1_extor5_2-1 (* mag_disp_p0_p1_extor5_1-1 (cos (- 0 0))))
(= F_GEN1_b1_p1_extor5_1-1 gen1_yc_b1_s1_extor5_2-1)
(= gen1_yc_b1_s1_extor5_2-1 (* mag_F_GEN1_b1_p1_extor5_1-1 (sin (- 270 0))))
(= disp_yc_b1_p1_extor5_2-1 (* mag_disp_p0_p1_extor5_1-1 (sin (- 0 0))))
(= mag_Trq_GEN1_b1_p1_extor5_1-1_2-1_3-2 (* disp_xc_b1_p1_extor5_2-1
      gen1_yc_b1_s1_extor5_2-1))
(= mag_restrq_b1_extor5_1-1_2-1 unknown2)
(= mag_restrq_b1_extor5_1-1_2-1 mag_Trq_GEN1_b1_p1_extor5_1-1_2-1_3-2 )
(= mag_restrq_b1_extor5_1-1_2-1 mag_Trq_GEN1_b1_p1_extor5_1-1_2-1_3-1 )
(= mag_restrq_b1_extor5_1-1_2-1 (* I_b1_extor5_1-1 mag_alpha_b1_extor5_1-1 )
```

## Appendix C: Angular Momentum Problems

### Angular Momentum Problem 1

A meter stick of mass .5 kg is rotated at 3 rev/sec about its midpoint. What is its angular momentum?

### Givens

```
(deffacts exang1 "Facts for exercise rotary motion problem ang1"
  (problem exang1)
  (meter stick b1)
  (mass (obj b1)(value .5)(units kg))
  (rotation-axis (obj b1)(shape rod)(point p0)(spin major))
  (same-body b1 p0)
  (rotation-radius (obj b1) (value .5) (units m) (Time exang1))
  (speed (obj b1)(type ANG)(value 3)(units rev/sec)(Time exang1))
  (goal-problem (is find-ang-momentum)(applied-to b1)(Time exang1)))
```

### Equations Generated

```
(= mass_b1_exang1_1-1 0.5)
(= mag_omega_b1_exang1_1-1 3)
(= r_rot_b1_exang1_1-1 0.5)
(= I_b1_exang1_1-1 ( * 0.083 ( * mass_b1_exang1_1-1
        ( * r_rot_b1_exang1_1-1 r_rot_b1_exang1_1-1 ))))
(= mag_L_b1_exang1_1-1 unknown1)
(= mag_L_b1_exang1_1-1 (* I_b1_exang1_1-1  mag_omega_b1_exang1_1-1 ) )
```

**Angular Momentum Problem 2**

A hoop of mass 2 kg, with radius .75 m, is rotated at 4 rev/sec about an axis through its center and perpendicular to its radius. What is its angular momentum?

**Givens**

```
(deffacts exang2 "Facts for exercise rotary motion problem ang2"
  (problem exang2)
  (hoop b1)
  (mass (obj b1)(value 2)(units kg))
  (rotation-axis (obj b1)(shape hoop)(point p0)(spin major))
  (same-body b1 p0)
  (rotation-radius (obj b1) (value .75) (units m) (Time exang2))
  (speed (obj b1)(type ANG)(value 4)(units rev/sec)(Time exang2))
  (goal-problem (is find-ang-momentum)(applied-to b1)(Time exang2)))
```

**Equations Generated**

```
(= mass_b1_exang2_1-1 2)
(= mag_omega_b1_exang2_1-1 4)
(= r_rot_b1_exang2_1-1 0.75)
(= I_b1_exang2_1-1 ( * 1
        ( * mass_b1_exang2_1-1 ( * r_rot_b1_exang2_1-1 r_rot_b1_exang2_1-1 ))))
(= mag_L_b1_exang2_1-1 unknown1)
(= mag_L_b1_exang2_1-1 (* I_b1_exang2_1-1  mag_omega_b1_exang2_1-1 ) )
```

**Angular Momentum Problem 3**

A man stands on a frictionless platform that is rotating at an angular speed of 1.2 rev/sec. His arms are outstretched and he holds a weight in each hand. With his arms in this position, the rotational inertia of the entire man-weights-platform system is 6 kg*m^2. If the man moves his arms and the weights so that the rotational inertia of the system decreases to 2 kg*m^2, what is his new angular speed?

**Givens**

```
(deffacts exang3 "Facts for exercise rotary motion problem ang1"
  (problem exang3)
  (moment-of-inertia (obj b1) (value 6) (units kg*m^2)(Time 1))
  (moment-of-inertia (obj b1) (value 2) (units kg*m^2)(Time 2))
  (speed (obj b1)(type ANG)(value 1.2)(units rev/sec)(Time 1))
  (time-points (Time exang3)(intervals 1 2))
  (goal-problem (is find-ang-velocity)(applied-to b1)(Time 2)))
```

**Equations Generated**

```
(= mag_omega_b1_1_1-1 1.2)
(= mag_omega_b1_2_1-1 unknown1)
(= I_b1_1_1-1 6)
(= mag_L_b1_1_1-1 (* I_b1_1_1-1  mag_omega_b1_1_1-1 ) )
(= I_b1_2_1-1 2)
(= mag_L_b1_2_1-1 (* I_b1_2_1-1  mag_omega_b1_2_1-1 ) )
(= mag_L_b1_2_1-1 mag_L_b1_1_1-1 )
```

**Angular Momentum Problem 4**

Two disks are mounted on low friction bearings on the same axle and can be brought together so that they couple and rotate as one unit. The first disk has a moment of inertia of 3.3 kg * m^2 and is rotating at 450 rev/min. The second disk, with moment of inertia of 6.6 kg * m^2, is spinning at 900 rev/min in the same direction as the first disk. If the two disks are then coupled together, what would their angular speed be after coupling?

**Givens**

```
(deffacts exang4 "Facts for exercise rotary motion problem ang4"
  (problem exang4)
  (rotation-axis (obj b1_b2)(point p0))
  (moment-of-inertia (obj b1) (value 3.3) (units kg*m^2)(Time 1))
  (moment-of-inertia (obj b2) (value 6.6) (units kg*m^2)(Time 1))
  (speed (obj b1)(type ANG)(value 450)(units rev/min)(Time 1))
  (speed (obj b2)(type ANG)(value 900)(units rev/min)(Time 1))
  (compound (object b1_b2)(parts b1 b2)(Time 2))
  (time-points (Time exang4)(intervals 1 2))
  (goal-problem (is find-ang-velocity)(applied-to b1_b2)(Time 2)))
```

**Equations Generated**

```
(= mag_omega_b2_1_1-2 900)
(= I_b2_1_1-2 6.6)
(= mag_L_b2_1_1-2 (* I_b2_1_1-2  mag_omega_b2_1_1-2 ) )
(= mag_omega_b1_1_1-3 450)
(= I_b1_1_1-3 3.3)
(= mag_L_b1_1_1-3 (* I_b1_1_1-3  mag_omega_b1_1_1-3 ) )
(= I_b1_b2_2_1-1 (+ I_b2_1_1-2 I_b1_1_1-3))
(= mag_omega_b1_b2_2_1-1 unknown1)
(= mag_L_b1_b2_2_1-1 (* I_b1_b2_2_1-1  mag_omega_b1_b2_2_1-1 ) )
(= mag_L_b1_b2_exang4_1-1 (+ mag_L_b1_1_1-3 mag_L_b2_1_1-2))
(= mag_L_b1_b2_2_1-1 mag_L_b1_b2_exang4_1-1 )
```

## Appendix D: Templates

There are a total of fifty-five templates currently in ANDES. The new templates created by this research are included below. Many of the remaining templates were modified for this project but have not been included.

```
(deftemplate circular-motion
 (slot obj
    (default ?DERIVE))
 (slot angle
    (default ?DERIVE))
 (multislot Time
   (default ?DERIVE)))

(deftemplate given-force
  (slot applied-to
    (default ?DERIVE))
  (slot value
    (default ?DERIVE))
  (slot angle
    (default ?DERIVE))
  (slot units
    (default ?DERIVE))
  (slot applied-by
    (default ?DERIVE))
  (multislot Time
    (default ?DERIVE)))

(deftemplate given-force-angle
  (slot applied-to
    (default ?DERIVE))
  (slot angle
    (default ?DERIVE))
  (slot units
    (default ?DERIVE))
  (slot applied-by
    (default ?DERIVE))
  (multislot Time
    (default ?DERIVE)))

(deftemplate given-torque
  (slot applied-to
    (default ?DERIVE))
```

```
  (slot value
    (default ?DERIVE))
  (slot units
    (default ?DERIVE))
  (slot applied-at
    (default ?DERIVE))
  (multislot Time
    (default ?DERIVE)))

(deftemplate moment-of-inertia
  (slot obj
    (type SYMBOL)
    (default ?DERIVE))
  (slot value
    (default ?DERIVE))
  (slot units
    (type SYMBOL)
    (default ?DERIVE))
 (multislot Time
   (default ?DERIVE)))

(deftemplate ratio
  (slot quantity1
    (default ?DERIVE))
  (slot quantity2
    (default ?DERIVE))
  (slot value
    (default 1))
  (multislot Time
    (default ?DERIVE)))

(deftemplate relative-position
  (slot obj
    (type SYMBOL)
    (default ?DERIVE))
  (slot wrt
    (type SYMBOL)
    (default ?DERIVE))
  (slot direction
    (type INTEGER)
    (default 0))
  (multislot Time
    (default ?DERIVE)))
```

```
(deftemplate revolution-radius
  (slot obj
    (type SYMBOL)
    (default ?DERIVE))
  (slot value
     (default ?DERIVE))
  (slot units
     (default ?DERIVE))
  (multislot Time
     (default ?DERIVE)))

(deftemplate rotation-axis
  (slot obj
    (type SYMBOL)
    (default ?DERIVE))
  (slot shape
    (default ?DERIVE))
  (slot point
    (default ?DERIVE))
  (slot spin
    (default ?DERIVE)))

(deftemplate rotation-radius
  (slot obj
    (type SYMBOL)
    (default ?DERIVE))
  (slot value
     (default ?DERIVE))
  (slot units
     (default ?DERIVE))
  (multislot Time
     (default ?DERIVE)))

(deftemplate touching
  (slot obj1
     (default ?DERIVE))
  (slot obj2
     (default ?DERIVE))
   (multislot Time
     (default ?DERIVE)))
```

## Appendix E: Procedural Rules

```
;; If there is circular motion, then the direction of the acceleration vector
;; is toward the center
(defrule acc_dir_known_cir_motion
  (logical (subgoal (is find-vector-properties)(body ?b)(v_var ?v1)(Time $?t))
        (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t))
        (vector (kind acceleration)(itsname A)(v_var ?v1)(applied-to ?b)
             (Time $?t))
        (circular-motion (obj ?b1)(angle ?a)(Time $?t)))
=>
  (bind ?v2 (sym-cat dir_a_ ?b _ (string-time $?t) _ ?n1 - ?n2))
  (bind ?z (assert (direction (v_var ?v1)(theta ?a)(theta_var ?v2)(Time $?t))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v2 crlf)))

;;angular acceleration = velocity squared / radius
(defrule acc_eq_vel^2_div_rad
  (logical
    (subgoal (is find-vector-properties)(body ?b)(v_var ?v1)(Time $?t))
    (vector (kind acceleration)(itsname A)(v_var ?v1)(applied-to ?b)(Time $?t))
    (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t))
    (vector (kind velocity)(itsname V)(v_var ?v2)(applied-to ?b)(Time $?t))
    (mag (v_var ?v2) (magnitude ?m1)(units ?u1)(m_var ?v3)(Time $?t))
    (scalar (kind radius)(body ?b)(magnitude ?m2)(units ?u2)(m_var ?v4)(Time $?t)))
=>
  (bind ?f (sym-cat "(/ (* " ?v3 "  " ?v3 " ) " ?v4 " )" ))
  (bind ?v5 (sym-cat mag_a_ ?b _ (string-time $?t) _ ?n1 - ?n2))
  (bind ?z (assert (mag (v_var ?v1)(magnitude ?f)(units FIX_ME)(m_var ?v5)
        (Time $?t))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v5 crlf)))

;;if the problem goal is to find ang. accel, then it exists in the problem
(defrule ang_accel_exists
  (logical (goal-problem (is find-ang-accel)(applied-to ?b)(Time $?t))
        (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t)))
=>
  (bind ?v1 (sym-cat alpha_ ?b _  (string-time $?t) _ ?n1 - ?n2 ))
  (bind ?z (assert (vector (kind ang-accel)(itsname ANG)(v_var ?v1)(applied-to ?b)
        (applied-by nil)(Time $?t))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v1 crlf)))
```

```
;; if the problem goal is to find angular accel, then its magnitude is unknown
(defrule ang_accel_unknown
  (logical (goal-problem (is find-ang-accel)(applied-to ?b)(Time $?t))
        (vector (kind ang-accel)(itsname ANG)(v_var ?v1)(applied-to ?b)(Time $?t)))
=>
  (bind ?v2 (sym-cat mag_ ?v1))
  (bind ?a (sym-cat unknown (inc-unkn)))
  (bind ?z (assert (mag (v_var ?v1)(magnitude ?a)(units rad/sec^2)(m_var ?v2)
        (Time $?t))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v2 crlf)))

;;if angular momentum is known at a certain time, and the problem body is not
;;experiencing a torque or angular acceleration, then momentum is conserved and
;; known at all times
(defrule ang_momentum_equal_at_different_times
  (logical (vector (kind ang-momentum)(itsname ?i)(v_var ?v3)(applied-to ?b)
              (Time $?t1))
        (mag (v_var ?v3)(units ?u1)(m_var ?v4)(Time $?t1))
        (vector (kind ang-momentum)(itsname ?i)(v_var ?v1&~?v3)(applied-to ?b)
              (Time $?t))
        (mag (v_var ?v1)(units ?u2)(m_var ?v2)(Time $?t))
        (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t)))
  (not (vector (kind torque)(applied-to ?b)))
  (not (vector (kind ang-accel)(applied-to ?b)))
=>
  (bind ?eq (sym-cat "(= " ?v2 " " ?v4 " )"))
  (assert (eq ?eq))
  (printout ?*file* ?eq crlf))

;;if a body has a moment of inertia and an angular velocity, then it also has
;;angular momentum
(defrule ang_momentum_exists
  (logical (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t1))
        (subgoal (is find-vector)(body ?b)(Time $?t1))
        (scalar (kind I)(body ?b)(Time $?t))
        (vector (kind ang-velocity)(itsname ANG)(applied-to ?b)(Time $?t)))
=>
  (bind ?v1 (sym-cat L_ ?b _ (string-time $?t) _ ?n1 - ?n2 ))
  (bind ?z (assert (vector (kind ang-momentum)(itsname ANG)(v_var ?v1)(applied-to ?b)
        (applied-by nil)(Time $?t))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v1 crlf)))
```

```
;;magnitude of angular momentum = moment of inertia * ang. velocity
(defrule ang_momentum_mag_known_by_Iw
  (logical ;(subgoal (is find-vector-properties)(body ?b)(v_var ?v1)(Time $?t))
        ;(body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t))
        (vector (kind ang-momentum)(itsname ANG)(v_var ?v1)(applied-to ?b)(Time $?t))
        (scalar (kind I)(body ?b)(units ?y)(m_var ?v2)(Time $?t))
        (vector (kind ang-velocity)(itsname ANG)(v_var ?v3)(applied-to ?b)(Time $?t))
        (mag (v_var ?v3)(units ?u)(m_var ?v4)(Time $?t)))
=>
  (bind ?v5 (sym-cat mag_ ?v1))
  (bind ?a (sym-cat "(* " ?v2 "  " ?v4 " ) " ))
  (bind ?z (assert (mag (v_var ?v1)(magnitude ?a)(units FIX)(m_var ?v5)(Time $?t))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v5 crlf)))

;;a compound body's separate pieces each contribute to the total angular
;;momentum of the compound
(defrule ang_momentum_part_of_compounds_ang_momentum
  (logical (compound (object ?b)(parts $?p)(Time $?t))
        (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t))
        (moment-of-inertia (obj ?b)(value nil)(units ?u)(Time $?t))
        (axes (theta_c ?a2)(decision-pt ?n3)(choice ?n4)(body ?b))
        (vector (kind ang-momentum)(v_var ?v1)(applied-to ?b1)(Time $?t1))
        (mag (v_var ?v1)(m_var ?v2)(Time $?t1)))
  (test (member$ ?b1 $?p))
=>
  (bind ?w (member$ ?b ?*bodies*))
  (bind ?*num-momentum* (replace$ ?*num-momentum* ?w ?w (length$ $?p)))
  (bind ?*sum-momentum* (add-comp ?b ?v2 ?n4 ?*sum-momentum*)))

;;if the problem goal is to find angular momentum, then
;;it is unknown at the time of goal
(defrule ang_momentum_unknown
  (logical (goal-problem (is find-ang-momentum)(applied-to ?b)(Time $?t))
      (vector (kind ang-momentum)(itsname ANG)(v_var ?v1)(applied-to ?b)(Time $?t)))
=>
  (bind ?v2 (sym-cat mag_ ?v1))
  (bind ?a (sym-cat unknown (inc-unkn)))
  (bind ?z (assert (mag (v_var ?v1)(magnitude ?a)(units FIX)(m_var ?v2)
       (Time $?t))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v2 crlf)))
```

```
;;angular velocity exists if a speed of type ANG is given
(defrule ang_velocity_exists
  (logical (speed (obj ?b)(type ANG)(value ?x)(units ?u)(Time $?t))
        (body (obj ?b)(decision-pt ?n1)(choice ?n2)))
=>
  (bind ?v1 (sym-cat omega_ ?b _  (string-time $?t) _ ?n1 - ?n2 ))
  (bind ?z (assert (vector (kind ang-velocity)(itsname ANG)(v_var ?v1)(applied-to ?b)
        (applied-by nil)(Time $?t))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v1 crlf)))

;;if finding angular velocity is the problem goal, then ang. vel. exists
(defrule ang_velocity_exists2
  (logical (goal-problem (is find-ang-velocity)(applied-to ?b)(Time $?t))
        (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t)))
=>
  (bind ?v1 (sym-cat omega_ ?b _  (string-time $?t) _ ?n1 - ?n2 ))
  (bind ?z (assert (vector (kind ang-velocity)(itsname ANG)(v_var ?v1)(applied-to ?b)
        (applied-by nil)(Time $?t))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v1 crlf)))

;;magnitude of angular velocity is known if ANG speed exists
(defrule ang_velocity_mag_given
  (logical (speed (obj ?b)(type ANG)(value ?x)(units ?u)(Time $?t))
        (vector (kind ang-velocity)(itsname ANG)(v_var ?v1)(applied-to ?b)(Time $?t)))
=>
  (bind ?v2 (sym-cat mag_ ?v1))
  (bind ?z (assert (mag (v_var ?v1)(magnitude ?x)(units ?u)(m_var ?v2)(Time $?t))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v2 crlf)))

;;if finding angular velocity is the problem goal, then its magnitude
;; is unknown
(defrule ang_velocity_unknown
  (logical (goal-problem (is find-ang-velocity)(applied-to ?b)(Time $?t))
        (vector (kind ang-velocity)(itsname ANG)(v_var ?v1)(applied-to ?b)(Time $?t)))
=>
  (bind ?v2 (sym-cat mag_ ?v1))
  (bind ?a (sym-cat unknown (inc-unkn)))
  (bind ?z (assert (mag (v_var ?v1)(magnitude ?a)(units rev/sec)(m_var ?v2)
        (Time $?t))))
  (if (neq ?z FALSE) then
```

```
    (printout ?*file-v* ?v2 crlf)))

;;if a body is touching a surface, then it has a certain contact angle
(defrule body_on_surface
  (logical
    (touching (obj1 ?b)(obj2 ?b2)(Time $?t))
    (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t))
    (surface (obj ?b2)(incline ?e)))
=>
  (bind ?v1 (sym-cat theta_n_ ?b _ ?b2 _ (string-time $?t) _ ?n1 - ?n2))
  (bind ?z (assert (contact (object1 ?b)(object2 ?b2)(contact_angle ?v1)
        (Time $?t))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v1 crlf)))

;;a compound body has a moment of inertia equal to
;;the sum of its pieces' moments of inertia
(defrule compound_has_moment_of_I
  (logical (compound (object ?x)(parts $?p)(Time $?t))
        (body (obj ?x)(decision-pt ?n1)(choice ?n2)(Time $?t))
        (moment-of-inertia (obj ?x)(value nil)(units ?u)(Time $?t)))
=>
  (bind ?v1 (sym-cat I_ ?x _ (string-time $?t) _ ?n1 - ?n2))
  (bind ?z (assert (scalar (kind I) (body ?x)
              (magnitude (nth$ (member$ ?x ?*bodies*) ?*cmpd-moments*))
              (units ?u)(m_var ?v1)(Time $?t))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v1 crlf)))

;;if a body is part of a compound, then its moment of inertia is
;;added to the other pieces' moments of inertia
(defrule compound_I_is_sum_of_parts
  (declare (salience 1)) ;; Nec so when compound-has-I fires ?*cmpd-moments* is full
  (logical (compound (object ?x)(parts $?p)(Time $?t))
        (body (obj ?x)(decision-pt ?n1)(choice ?n2)(Time $?t))
        (scalar (kind I)(body ?b)(units ?u)(m_var ?v1)))
  (test (neq FALSE (member$ ?b $?p)))
=>
  (bind ?*cmpd-moments* (insert-moment ?x ?v1)))
```

```
;;if a compound body has angular momentum, and all of
;;its pieces' momentums have been added together,
;;then the magnitude of the angular mometum of the compound body is known
(defrule compound_momentum_known
  (logical (vector (kind ang-momentum)(itsname ANG)(v_var ?v1)(applied-to ?b)
                   (Time $?t))
           (compound (object ?b)(parts $?p)(Time $?t2))
           (vector (kind ang-momentum)(itsname ANG)(v_var ?v2&~?v1)
                   (applied-to ?b1)(Time $?t1))
           (mag (v_var ?v2)(magnitude ?f&~nil)(m_var ?v3)(Time $?t1))
           (axes (theta_c ?a2)(decision-pt ?n1)(choice ?n2)(body ?b)))
  (test (member$ ?b1 $?p))
  (test (eq (nth$ (member$ ?b ?*bodies*) ?*num-momentum*)
            (count-plus (nth$ (- (+ (member$ ?b ?*bodies*) ?n2) 1) ?*sum-momentum*))))
  (test (same-method-axes ?v1 ?n1 ?n2))
  (test (same-method-axes ?v2 ?n1 ?n2))
  (test (tinside $?t1 $?t))
  (test (tinside $?t2 $?t))
=>
  (bind ?v4 (sym-cat mag_ ?v1))
  (bind ?a (nth$ (- (+ (member$ ?b ?*bodies*) ?n2) 1) ?*sum-momentum*))
  (bind ?z (assert (mag (v_var ?v1)(magnitude ?a)(units nil)(m_var ?v4)(Time $?t))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v4 crlf)))

;;if two bodies have a displacement between them, and their relative
;;position if defined, then the direction of displacement is known
(defrule disp_direction_known
  (logical (subgoal (is find-vector-properties)(body ?b)(v_var ?v1)(Time $?t))
           (vector (kind displacement)(itsname DISP)(v_var ?v1)(applied-to ?b1)
                   (applied-by ?b2)(Time $?t))
           (relative-position (obj ?b2)(wrt ?b1)(Time $?t)(direction ?d)))
=>
  (bind ?v2 (sym-cat dir_disp_ ?b1 _ ?b2 _ (string-time $?t)))
  (if (not (numberp ?d)) then (bind ?d (convert-compass ?d)))
  (bind ?z (assert (direction (v_var ?v1)(theta ?d)(theta_var ?v2)(Time $?t))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v2 crlf)))
```

```
;;if distance-btw-pts is given, then the magnitude of displacement
;;is known
(defrule disp_mag_known
  (logical (subgoal (is find-vector-properties)(body ?b)(v_var ?v1)(Time $?t))
        (body (obj ?b)(decision-pt ?n3)(choice ?n4)(Time $?t))
        (distance-btw-pts (pt1 ?b1)(pt2 ?b2)(value ?x)(units ?u)(Time $?t))
        (vector (kind displacement)(itsname DISP)(v_var ?v1)(applied-to ?b1)
             (applied-by ?b2)(Time $?t)))
=>
  (bind ?v3 (sym-cat mag_disp_ ?b1 _ ?b2 _ (string-time $?t) _ ?n3 - ?n4))
  (bind ?z (assert (mag (v_var ?v1)(magnitude ?x)(units ?u)(m_var ?v3)(Time $?t))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v3 crlf)))

;;if distance-btw-pts and relative position are given,
;;then a displacement vector exists
(defrule disp_vector_exists_torque
   (logical (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t))
         (distance-btw-pts (pt1 ?b1)(pt2 ?b2)(value ?x)(units ?u)(Time $?t))
         (relative-position (obj ?b2)(wrt ?b1)(direction ?d)(Time $?t)))
=>
  (bind ?v1 (sym-cat DISP_ ?b1 _ ?b2 _ (string-time $?t) _ ?n1 - ?n2))
  (bind ?q (assert (vector (kind displacement)(itsname DISP)(v_var ?v1)
             (applied-to ?b1)(applied-by ?b2)(Time $?t))))
  (if (neq ?q FALSE) then
        (printout ?*file-v* ?v1 crlf)))


;;If a given-force exists (with no source), assert it as a generic force
(defrule generic_force_exists
  (logical (subgoal (is find-vector)(body ?b1)(Time $?t))
        (given-force (applied-to ?b2)(value ?x)(angle ?a)(units ?y)(applied-by ?ab)
              (Time $?t))
        (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t)))
=>
  (bind ?in (sym-cat GEN (inc-gen)))
  (bind ?v1 (sym-cat F_ ?in _ ?b1 _ ?b2 _ (string-time $?t) _ ?n1 - ?n2))
  (bind ?z (assert (vector (kind force)(itsname ?in)(v_var ?v1)(applied-to ?b2)
        (applied-by ?ab)(Time $?t))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v1 crlf)))
```

```
;;if a force is given, then its direction is known
(defrule given_force_dir_known
  (logical (subgoal (is find-vector-properties)(body ?b)(v_var ?v1)(Time $?t))
        (vector (kind force) (itsname ?in)(v_var ?v1)(applied-to ?b1)
             (applied-by ?ab)(Time $?t))
        (given-force (applied-to ?b1)(value ?x)(angle ?a)(units ?y)(applied-by ?ab)
              (Time $?t))
        (body (obj ?b)(decision-pt ?n3)(choice ?n4)(Time $?t)))
  (test (or (eq ?b ?b1) (are-same-body ?*co-points* ?b ?b1)))
=>
  (bind ?v2 (sym-cat dir_F_ ?in _ ?b _ ?b1 _ (string-time $?t) _ ?n3 - ?n4))
  (bind ?z (assert (direction (v_var ?v1)(theta ?a)(theta_var ?v2)(Time $?t))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v2 crlf)))

;;if a force is given with a non-nil value, then its magnitude is known
(defrule given_force_mag_known
  (logical (subgoal (is find-vector-properties)(body ?b1)(v_var ?v1)(Time $?t))
        (vector (kind force)(itsname ?in)(v_var ?v1)(applied-to ?b2)(Time $?t))
        (body (obj ?b1)(decision-pt ?n3)(choice ?n4)(Time $?t))
        (given-force (applied-to ?b2)(value ?x&~nil)(angle ?a)(units ?y)
              (applied-by ?ab)(Time $?t)))
  (test (are-same-body ?*co-points* ?b1 ?b2)) ;(eq ?b1 ?b2)))
=>
  (bind ?v2 (sym-cat mag_F_ ?in _ ?b1 _ ?b2 _ (string-time $?t) _ ?n3 - ?n4))
  (bind ?z (assert (mag (v_var ?v1)(magnitude ?x)(units ?y)(m_var ?v2)(Time $?t))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v2 crlf)))

;;if a force is given with a nil value, then its magnitude is unknown
(defrule given_force_mag_unknown
  (logical (goal-problem (is find-force-mag)(applied-to ?b2)(Time $?t))
        (vector (kind force)(itsname ?in)(v_var ?v1)(applied-to ?b2)(Time $?t))
        (body (obj ?b1)(decision-pt ?n3)(choice ?n4)(Time $?t))
        (given-force (applied-to ?b2)(value nil)(angle ?a)(units ?y)(applied-by ?ab)
              (Time $?t)))
  (test (are-same-body ?*co-points* ?b1 ?b2)) ;(eq ?b1 ?b2)))
=>
  (bind ?v2 (sym-cat mag_F_ ?in _ ?b1 _ ?b2 _ (string-time $?t) _ ?n3 - ?n4))
  (bind ?x (sym-cat unknown (inc-unkn)))
  (bind ?z (assert (mag (v_var ?v1)(magnitude ?x)(units ?y)(m_var ?v2)(Time $?t))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v2 crlf)))
```

```
;;if the problem goal is to find the incline angle, then
;;then angle is unknown
(defrule incline_unknown
  (logical (goal-problem (is find-incline) (applied-to b1) (Time $?t))
        (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t)))
=>
  (bind ?v2 (sym-cat incline))
  (bind ?a (sym-cat unknown (inc-unkn)))
  (bind ?z (assert (angle theta ?a)))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v2 crlf)))


;; If the goal is to use Newton's laws, and a body has no declared mass,
;; then the mass is unknown.
(defrule mass_unknown
  (logical ;(subgoal (is use-newton) (body ?b1) (Time $?t))
        (near-planet (Time $?t))
        (body (obj ?b1)(decision-pt ?n1)(choice ?n2)(Time $?t)))
  (not (scalar (kind mass)(body ?b1)(Time $?t)))
  (not (mass (obj ?b1)))
=>
  (bind ?a (sym-cat unknown (inc-unkn)))
  (assert (mass (obj ?b1) (value ?a) (units kg))))

;;if a body has a rotation axis and no moment of inertia defined yet,
;;then moment of inertia exists but is unknown
(defrule moment_of_inertia_exists
  (logical (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t))
        (rotation-axis (obj ?b)(shape ?s)(spin ?p)))
  (not (moment-of-inertia (obj ?b) (value ?x2) (units ?u)(Time $?t)))
=>
  (assert (moment-of-inertia (obj ?b) (value nil) (units kg*m^2)(Time $?t))))


;; If the moment of inertia of an object exists create a variable for it.
(defrule moment_of_inertia_given
  (logical (moment-of-inertia (obj ?b) (value ?x&~nil) (units ?y)(Time $?t))
        (body (obj ?b)(decision-pt ?n1)(choice ?n2)))
=>
  (bind ?v1 (sym-cat I_ ?b _ (string-time $?t) _ ?n1 - ?n2))
  (bind ?z (assert (scalar (kind I)(body ?b)(magnitude ?x)(units ?y)
        (m_var ?v1)(Time $?t))))
```

```
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v1 crlf)))

;;magnitude of moment of inertia = mass * radius squared
(defrule moment_of_inertia_known_by_mr^2
  (logical (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t))
        (scalar (kind mass)(body ?b)(m_var ?v1)(Time $?t))
        (rotation-axis (obj ?b)(shape ?s)(spin ?p))
        (scalar (kind radius)(type rotation)(body ?b)(m_var ?v2)(Time $?t))
        (moment-of-inertia (obj ?b) (value nil)(Time $?t)))
=>
  (bind ?f (inertia-fraction ?s ?p))
  (bind ?a (sym-cat "( * " ?f " ( * " ?v1 " ( * " ?v2 " " ?v2 " )))"))
  (bind ?v3 (sym-cat I_ ?b _ (string-time $?t) _ ?n1 - ?n2))
  (bind ?z (assert (scalar (kind I)(body ?b)(magnitude ?a)(units kg*m^2)
        (m_var ?v3)(Time $?t))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v3 crlf)))


;; If goal is to find normal, then normal magnitude is unknown.
(defrule normal_mag_unknown
  (logical (subgoal (is find-vector-properties)(body ?b)(v_var ?v1)(Time $?t2))
        (vector (kind force) (itsname N)(v_var ?v1)(applied-to ?b)
             (applied-by ?s)(Time $?t1))
        (direction (v_var ?v1)(Time $?t2))
        (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t1)))
  (not (goal-problem (is find-normal)(applied-to ?b)(Time $?t1)))
  (test (tintersect $?t1 $?t2))
=>
  (bind $?t $?t1);(create$ (tintersect $?t1 $?t2)))
  (bind ?v2 (sym-cat mag_n_ ?b _ ?s _ (string-time $?t) _ ?n1 - ?n2))
  (bind ?a (sym-cat unknown (inc-unkn)))
  (bind ?z (assert (mag (v_var ?v1)(magnitude ?a)(units nil)(m_var ?v2)
        (Time $?t))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v2 crlf))
  (bind ?*num-mag* (add-one ?b ?*num-mag*)))

;;if points are defined as being on the same body, then
;;put them into a global variable for easier use of this info (by knowledge base)
(defrule points_are_same_body
  (declare (salience 1))
```

```
  (logical (same-body $?x))
=>
  (bind ?*co-points* (insert$ ?*co-points* (+ 1 (length$ ?*co-points*))
                (create$ (length ?x) ?x))))

;;if angular momentum is known at a certain ti me, and
;;no torques act on the body, then assert an angular momentum
;;for the entire problem
(defrule problem_has_ang_momentum_conserved
  (logical (goal-problem (applied-to ?b))
        (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t))
        (vector (kind ang-momentum)(itsname ANG)(v_var ?v1)(applied-to ?b)
                (Time $?t1))
        (mag (v_var ?v1)(units ?u)(m_var ?v2)(Time $?t1))
        (problem $?pb))
  (not (problemp $?t1))
  (not (vector (kind torque)(applied-to ?b)))
  (not (vector (kind ang-momentum)(applied-to ?b)(Time $?pb)))
=>
  (bind ?v3 (sym-cat L_ ?b _ (string-time $?pb) _ ?n1 - ?n2))
  (bind ?z (assert (vector (kind ang-momentum)(itsname ANG)
                    (v_var ?v3)(applied-to ?b)(Time $?pb))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v3 crlf)))

;;if a ratio between force magnitudes is given,
;;then equation involving the two can be asserted
(defrule ratio_of_force_mags_known
  (logical (subgoal (is find-vector-properties)(body ?b)(v_var ?v1)(Time $?t2))
        (vector (kind force) (itsname ?i1)(v_var ?v1)(applied-to ?b)
             (applied-by ?s1)(Time $?t1))
        (vector (kind force) (itsname ?i2)(v_var ?v2)(applied-to ?b)
             (applied-by ?s2)(Time $?t1))
        (mag (v_var ?v1) (magnitude ?m1) (units ?u1) (m_var ?v3) (Time $?t1))
        (mag (v_var ?v2) (magnitude ?m2) (units ?u2) (m_var ?v4) (Time $?t1))
        (ratio (quantity1 ?i1) (quantity2 ?i2)(value ?v)(Time $?t3)))
  (test (tintersect $?t1 $?t2 $?t3))
=>
  (bind ?f (sym-cat "(= " ?v3 " (* " ?v " " ?v4 "))"))
  (assert (eq ?f))
  (printout ?*file* ?f crlf))
```

```
;;if a given torque is applied to the rotation axis, then it is the
;; resultant torque and its magnitude is known
(defrule resultant_mag_given
  (logical (subgoal (is find-vector-properties)(body ?b)(v_var ?v1)(Time $?t))
        (given-torque (applied-to ?b)(value ?x)(units ?u)(applied-at ?aa)(Time $?t))
        (vector (kind torque)(itsname RTQ)(v_var ?v1)(applied-to ?b)(Time $?t))
        (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t))
        (axes (theta_c ?a2)(decision-pt ?n3)(choice ?n4)(body ?b)))
=>
  (bind ?v2 (sym-cat mag_ ?v1))
  (bind ?z (assert (mag (v_var ?v1)(magnitude ?x)(units nil)(m_var ?v2)(Time $?t))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v2 crlf)))

;; if all torques have been added up, then the magnitude
;;of the resultant torque is known
(defrule resultant_mag_known
  (logical (vector (kind torque)(itsname RTQ)(v_var ?v1)(applied-to ?b)(Time $?t))
        (mag (v_var ?v1)(magnitude ?m)(m_var ?v4)(Time $?t))
        (vector (kind torque)(itsname ?in&~RTQ)(v_var ?v2)(applied-to ?b1)(Time $?t))
        (mag (v_var ?v2)(magnitude ?f)(m_var ?v3)(Time $?t))
        (axes (theta_c ?a2)(decision-pt ?n1)(choice ?n2)(body ?b))
        (method (find-what torque)(decision-pt ?n3)(choice ?n4)))
  (test (eq (nth$ 1 ?*num-torques*)
            (count-plus (nth$ (slot-no ?n4 ?n2) ?*sum-torques*))))
  (test (are-same-body ?*co-points* ?b ?b1))
  (test (same-method-axes ?v3 ?n3 ?n4))
  (test (same-method-axes ?v1 ?n1 ?n2))
=>
  (bind ?eq (sym-cat "(= " ?v4 " " (nth$ (slot-no ?n4 ?n2) ?*sum-torques*) " )"))
  (bind ?g (assert (eq ?eq)))
  (if (neq ?g FALSE) then
    (printout ?*file* ?eq crlf)))
```

```
;;if moment of inertia and angular acceleration are known, then
;;the body's total torque is known
(defrule resultant_mag_known_by_I_alpha
  (logical (vector (kind torque)(itsname RTQ)(v_var ?v1)(applied-to ?b)(Time $?t))
        (mag (v_var ?v1)(magnitude ?m)(m_var ?v2)(Time $?t))
        (scalar (kind I)(body ?b)(magnitude ?x)(units ?y)(m_var ?v3) (Time $?t))
        (vector (kind ang-accel)(itsname ANG)(v_var ?v4)(applied-to ?b) (Time $?t))
        (mag (v_var ?v4)(magnitude ?f)(m_var ?v5)(Time $?t))
        (axes (theta_c ?a2)(decision-pt ?n1)(choice ?n2)(body ?b)))
=>
  (bind ?eq (sym-cat "(= " ?v2 " (* " ?v3 " " ?v5 " )"))
  (bind ?g (assert (eq ?eq)))
  (if (neq ?g FALSE) then
    (printout ?*file* ?eq crlf)))

;;if the problem goal is to find the resultant torque,
;;then its magnitude is unknown
(defrule resultant_mag_unknown
  (declare (salience -1))
  (logical (goal-problem (is find-torque)(applied-to ?b)(Time $?t))
        (vector (kind torque)(itsname RTQ)(v_var ?v1)(applied-to ?b)(Time $?t))
        (rotation-axis (obj ?b)(point ?b1)))
  (not (given-torque (applied-to ?b)(applied-at ?b1)(Time $?t)))
=>
  (bind ?v2 (sym-cat mag_ ?v1))
  (bind ?a (sym-cat unknown (inc-unkn)))
  (bind ?z (assert (mag (v_var ?v1)(magnitude ?a)(units nil)(m_var ?v2)
        (Time $?t))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v2 crlf)))

;;if the problem goalis to find the resultant torque,
;;then it exists
(defrule resultant_torque_exists
  (logical (goal-problem (is find-torque)(applied-to ?b)(Time $?t))
        (axes (theta_c ?a2)(decision-pt ?n3)(choice ?n4)(body ?b))
        (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t)))
=>
  (bind ?v1 (sym-cat restrq_ ?b _ (string-time $?t) _ ?n1 - ?n2 _ ?n3 - ?n4))
  (bind ?z (assert (vector (kind torque)(itsname RTQ)(v_var ?v1)
        (applied-to ?b)(applied-by nil)(Time $?t))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v1 crlf)))
```

```
;;if a given torque is applied at the rotation axis of the body,
;;then this given torque is the resultant torque
(defrule resultant_torque_given
  (logical (subgoal (is find-vector)(body ?b)(Time $?t))
        (given-torque (applied-to ?b)(value ?x)(units ?u)(applied-at ?aa)(Time $?t))
        (rotation-axis (obj ?b)(point ?aa))
        (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t))
        (axes (theta_c ?a2)(decision-pt ?n3)(choice ?n4)(body ?b)))
=>
  (bind ?v1 (sym-cat restrq_ ?b _  (string-time $?t) _ ?n1 - ?n2 _ ?n3 - ?n4))
  (bind ?z (assert (vector (kind torque)(itsname RTQ)(v_var ?v1)(applied-to ?b)
        (applied-by nil)(Time $?t))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v1 crlf)))


;;If a revolution radius exists, create a variable for it.
(defrule revolution_radius_given
  (logical (revolution-radius (obj ?b) (value ?x) (units ?y) (Time $?t))
        (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t)))
=>
  (bind ?v1 (sym-cat r_rev_ ?b _ (string-time $?t) _ ?n1 - ?n2))
  (bind ?z (assert (scalar (kind radius)(type revolution)(body ?b)(magnitude ?x)(units ?y)
        (m_var ?v1)(Time $?t))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v1 crlf)))

;;if rotation radius is given, create a variable for it
(defrule rotation_radius_given
  (logical (rotation-radius (obj ?b) (value ?x) (units ?y) (Time $?t))
        (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t)))
=>
  (bind ?v1 (sym-cat r_rot_ ?b _ (string-time $?t) _ ?n1 - ?n2))
  (bind ?z (assert (scalar (kind radius)(type rotation)(body ?b)(magnitude ?x)(units ?y)
        (m_var ?v1)(Time $?t))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v1 crlf)))

;;if a track with an angle is given, then it is a surface
(defrule surface_has_incline
 s (logical (track ?b2)
        (angle theta ?e))
```

```
=>
  (assert (surface (obj ?b2)(fixed yes)(incline theta))))


;; If goal is to find tension, then tension magnitude is unknown.
(defrule tension_mag_unk_cir_motion
  (logical (subgoal (is find-vector-properties)(body ?b)(v_var ?v1)(Time $?t2))
        (vector (kind force) (itsname T)(v_var ?v1)(applied-to ?b)(applied-by ?s)
             (Time $?t1))
        (circular-motion (obj ?b) (Time $?t1))
        (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t)))
  (not (tension (applied-to ?b) (Time $?t1)))
  (not (mag (v_var ?v1)))
  (test (tintersect $?t1 $?t2))
=>
  (bind $?t $?t1); (create$ (tintersect $?t1 $?t2)))
  (bind ?v2 (sym-cat mag_t_ ?b _ ?s _(string-time $?t) _ ?n1 - ?n2))
  (bind ?a (sym-cat unknown (inc-unkn)))
  (bind ?z (assert (mag (v_var ?v1)(magnitude ?a)(units nil)(m_var ?v2)(Time $?t))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v2 crlf))
  (bind ?*num-mag* (add-one ?b ?*num-mag*)))


;;if theta is known, then eq known.
(defrule theta_eq
  (logical (angle theta ?e))
=>
  (bind ?f (sym-cat "(= theta " ?e " )"))
  (assert (eq ?f))
  (printout ?*file* ?f crlf))
```

```
;;if a body is on a surface, then the incline angle has
;;  a fixed relationship with the contact angle
(defrule theta_n_known
  (logical
    (surface (obj ?b2) (incline ?e))
    (contact (object1 ?b1) (object2 ?b2) (contact_angle ?a) (Time $?t)))
    (circular-motion (obj ?b) (Time $?t))
    (vector (kind force) (itsname N) (v_var ?v2) (Time $?t))
    (direction (v_var ?v2) (theta ?a) (theta_var ?v3) (Time $?t))
    (vector (kind acceleration) (itsname A) (v_var ?v1) (Time $?t))
    (direction (v_var ?v1) (theta ?e2) (Time $?t))
=>
  (if (numberp ?e2) then
    (if (or (<= ?e2 90) (> ?e2 270)) then
      (bind ?f (sym-cat "(= " ?a " (- " ?e " 90))")))
    (if (and (> ?e2 90) (<= ?e2 270)) then
      (bind ?f (sym-cat "(= " ?a " (+ " ?e " 90))")))
    else (bind ?f (sym-cat "(= " ?a " (+ " ?e " 90))")))
  (assert (eq ?f))
  (printout ?*file* ?f crlf))

;;if a force is applied to a point on a body that is not on the rotation axis,
;;then that force causes a torque
(defrule torque_caused_by_force
  (logical (subgoal (is find-vector-properties)(body ?b)(v_var ?v1)(Time $?t1))
        (rotation-axis (obj ?b)(point ?b1))
        (vector (kind force)(itsname ?in)(v_var ?v1)(applied-to ?b2)
              (applied-by ?ab)(Time S?t1))
        (direction (v_var ?v1)(theta ?a1)(theta_var ?v2)(Time $?t1))
        (vector (kind displacement)(itsname DISP)(v_var ?v3)(applied-to ?b1)
              (applied-by ?b2)(Time S?t1))
        (direction (v_var ?v3)(theta ?a2)(theta_var ?v4)(Time $?t1))
        (body (obj ?b)(decision-pt ?n3)(choice ?n4)(Time $?t1)))
  (test (are-same-body ?*co-points* ?b1 ?b2))
  (test (not (inlinep ?a1 ?a2)))
=>
  (bind ?*num-torques* (add-one ?b ?*num-torques*))
  (bind ?v5 (sym-cat Trq_ ?in _ ?b _ ?b2 _ (string-time $?t1) _ ?n3 - ?n4))
  (bind ?z (assert (vector (kind torque)(itsname ?in)(v_var ?v5)(applied-to ?b)
        (applied-by ?ab)(Time $?t1))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v5 crlf)))
```

```
;; asserts torque equation of form T = r * F sin (theta)
;; this equation always placed in first position of ?*sum-torques*
(defrule torque_mag_known
  (logical (subgoal (is find-vector-properties)(body ?b)(v_var ?v1)(Time $?t))
        (body (obj ?b)(decision-pt ?n3)(choice ?n4)(Time $?t))
        (vector (kind torque) (itsname ?in)(v_var ?v1)(applied-to ?b)
              (applied-by ?ab1)(Time $?t))
        (vector (kind force) (itsname ?in)(v_var ?v2)(applied-to ?b2)
              (applied-by ?ab1)(Time $?t))
        (mag (v_var ?v2)(magnitude ?m1)(units ?u1)(m_var ?v3)(Time $?t))
        (direction (v_var ?v2)(theta ?a1)(theta_var ?v4)(Time $?t))
        (vector (kind displacement)(itsname DISP)(v_var ?v5)(applied-to ?b1)
              (applied-by ?b2)(Time $?t))
        (mag (v_var ?v5)(magnitude ?m2)(units ?u2)(m_var ?v6)(Time $?t))
        (direction (v_var ?v5)(theta ?a2)(theta_var ?v7)(Time $?t))
        (axes (theta_c ?a3)(decision-pt ?n1)(choice ?n2)(body ?b))
        (method (find-what torque)(decision-pt ?n9)(choice ?n10&1)))
=>
  (bind ?a (sym-cat "( - " ?v7 " ( - " ?v4 " " ?a3" ))" ))
  (bind ?f (sym-cat "( * " ?v3 " ( * " ?v6 " (sin " ?a ")))" ))
  (bind ?v8 (sym-cat mag_Trq_ ?in _ ?b _ ?b2 _
              (string-time $?t) _ ?n3 - ?n4 _ ?n1 - ?n2 _ ?n9 - ?n10))
  (bind ?z (assert (mag (v_var ?v1)(magnitude ?f)(units nil)(m_var ?v8)(Time $?t))))
  (if (neq ?z FALSE) then
    (printout ?*file-v* ?v8 crlf))
  (bind ?*num-mag* (add-one ?b ?*num-mag*))
  (bind ?*sum-torques* (add-comp ?b ?v8 ?n2 ?*sum-torques*)))
```

```
;; asserts torque equation of form T = rx * Fy + ry * Fx
;; this equation always placed in second position of ?*sum-torques*
(defrule torque_mag_known2
  (logical (subgoal (is find-vector-properties)(body ?b)(v_var ?v1)(Time $?t))
        (vector (kind torque) (itsname ?in)(v_var ?v1)(applied-to ?b)
              (applied-by ?ab1)(Time $?t))
        (vector (kind force) (itsname ?in)(v_var ?v2)(applied-to ?b2)
              (applied-by ?ab2)(Time $?t))
        (xc (v_var ?v2) (theta_c ?a1) (x_comp ?x1) (xc_var ?v3) (decision-pt ?n1) (choice
              ?n2) (Time $?t))
        (yc (v_var ?v2) (theta_c ?a2) (y_comp ?y1) (yc_var ?v4) (decision-pt ?n1)
              (choice ?n2) (Time $?t))
        (vector (kind displacement)(itsname DISP)(v_var ?v5)(applied-to ?b1)
              (applied-by ?b2)(Time $?t))
        (xc (v_var ?v5) (theta_c ?a3) (x_comp ?x2) (xc_var ?v6) (decision-pt ?n1) (choice
              ?n2) (Time $?t))
        (yc (v_var ?v5) (theta_c ?a4) (y_comp ?y2) (yc_var ?v7) (decision-pt ?n1)
              (choice ?n2) (Time $?t))
        (axes (theta_c ?a5)(decision-pt ?n1)(choice ?n2)(body ?b))
        (method (find-what torque)(decision-pt ?n9)(choice ?n10&2))
        (body (obj ?b)(decision-pt ?n3)(choice ?n4)(Time $?t)))
=>
   (if (or (perpendicularp ?a5 ?a1)(inlinep ?a5 ?a4)) then
        (bind ?f (sym-cat "(* " ?v6 " " ?v4 ")" ))
        (bind ?v8 (sym-cat mag_Trq_ ?in _ ?b _ ?b2 _
              (string-time $?t) _ ?n3 - ?n4 _ ?n1 - ?n2 _ ?n9 - ?n10))
        (bind ?z (assert (mag (v_var ?v1)(magnitude ?f)(units nil)(m_var ?v8)
              (Time $?t))))
        (if (neq ?z FALSE) then
              (printout ?*file-v* ?v8 crlf)
        (bind ?*num-mag* (add-one ?b ?*num-mag*))
        (bind ?*sum-torques* (add-comp ?b ?v8 (+ 2 ?n2) ?*sum-torques*)))
   else
   (if (or (inlinep ?a5 ?a2)(perpendicularp ?a5 ?a3)) then
        (bind ?f (sym-cat "(* " ?v7 " " ?v3 ")" ))
        (bind ?v8 (sym-cat mag_Trq_ ?in _ ?b _ ?b2 _
              (string-time $?t) _ ?n3 - ?n4 _ ?n1 - ?n2 _ ?n9 - ?n10))
        (bind ?z (assert (mag (v_var ?v1)(magnitude ?f)(units nil)(m_var ?v8)
              (Time $?t))))
        (if (neq ?z FALSE) then
              (printout ?*file-v* ?v8 crlf)
        (bind ?*num-mag* (add-one ?b ?*num-mag*))
        (bind ?*sum-torques* (add-comp ?b ?v8 (+ 2 ?n2) ?*sum-torques*)))
```

```
   else
        (bind ?f (sym-cat "(+ (* " ?v6 " " ?v4 ") (* " ?v7 " " ?v3 "))" ))
        (bind ?v8 (sym-cat mag_Trq_ ?in _ ?b _ ?b2 _
              (string-time $?t) _ ?n3 - ?n4 _ ?n1 - ?n2 _ ?n9 - ?n10))
        (bind ?z (assert (mag (v_var ?v1)(magnitude ?f)(units nil)(m_var ?v8)
              (Time $?t))))
        (if (neq ?z FALSE) then
              (printout ?*file-v* ?v8 crlf)
              (bind ?*num-mag* (add-one ?b ?*num-mag*))
              (bind ?*sum-torques* (add-comp ?b ?v8 (+ 2 ?n2) ?*sum-torques*))))))


;; If the velocity vector exists and inst speed exist, then mag of vector exists.
(defrule velocity_mag_known
  (logical
    (subgoal (is find-vector-properties)(body ?b)(v_var ?v1)(Time $?t))
    (vector (kind velocity)(itsname V)(v_var ?v1)(applied-to ?b)(Time $?t))
    (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t))
    (speed (obj ?b) (type inst) (value ?m) (units ?u) (Time $?t)))
=>
  (bind ?v2 (sym-cat mag_v_ ?b _ (string-time $?t) _ ?n1 - ?n2))
  (bind ?z (assert (mag (v_var ?v1)(magnitude ?m)(units ?u)(m_var ?v2)
       (Time $?t))))
  (if (neq ?z FALSE) then
     (printout ?*file-v* ?v2 crlf)))

;;if the problem goal is to find instant velocity, then is mag is unknown
(defrule velocity_mag_unknown
  (logical
    (goal-problem (is find-inst-vel)(applied-to ?b)(Time $?t))
    (subgoal (is find-vector-properties)(body ?b)(v_var ?v1)(Time $?t))
    (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t))
    (vector (kind velocity)(itsname V) (v_var ?v1)(applied-to ?b)(Time $?t)))
=>
  (bind ?v2 (sym-cat mag_v_ ?b _ (string-time $?t) _ ?n1 - ?n2))
  (bind ?a (sym-cat unknown (inc-unkn)))
  (bind ?z (assert (mag (v_var ?v1)(magnitude ?a)(units nil)(m_var ?v2)
       (Time $?t))))
  (if (neq ?z FALSE) then
     (printout ?*file-v* ?v2 crlf)))
```

## Appendix F: Planning Rules

```
; If the problem goal is to find force magnitude then make the
;; goal to find vector.
(defrule make_goal_find_force_mag
  (logical (goal-problem (is find-force-mag)(applied-to ?b1)(Time $?t1))
        (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t1)))
  (test (or (eq ?b ?b1)(are-same-body ?*co-points* ?b ?b1)))
=>
  (assert (subgoal (is find-vector)(body ?b)(Time $?t1))))


;; If the goal is to find the inst vel then make a goal of finding the mag of the vel
(defrule make_goal_find_mag_vel
 (logical (subgoal (is find-inst-vel)(body ?b)(Time $?t1))
        (vector (kind velocity)(itsname V)(v_var ?v1)(applied-to ?b)(Time $?t1)))
  (not (mag (v_var ?v1)))
=>
  (assert (subgoal (is find-inst-vel-mag)(body ?b)(Time $?t1))))

;;if goal is to find a force magnitude, and that force is causing
;;a torque, then set a goal to find torques
(defrule make_goal_find_torque_from_force
  (logical (goal-problem (is find-force-mag)(applied-to ?b1)(Time $?t))
        (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t))
        (rotation-axis (obj ?b)))
  (test (or (eq ?b ?b1)(are-same-body ?*co-points* ?b ?b1)))
=>
  (assert (goal-problem (is find-torque)(applied-to ?b)(Time $?t))))

;;if torques exists, and goal exists to find torques, then set
;;a goal to find torques
(defrule make_goal_find_torque_if_torques_exist
  (logical (goal-problem (is ?gp)(applied-to ?b)(Time $?t))
        (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t))
        (vector (kind torque)(applied-to ?b)(Time $?t)))
  (not (goal-problem (is find-torque)))
=>
  (assert (goal-problem (is find-torque)(applied-to ?b)(Time $?t))))
```

```
; If the problem goal is to find angular accel then make the
;; goal to find vector.
((defrule make_goal_find_vector_ang_accel
  (logical (goal-problem (is find-ang-accel)(applied-to ?b)(Time $?t1))
        (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t1)))
=>
  (assert (subgoal (is find-vector)(body ?b)(Time $?t1))))

; If the problem goal is to find angular momentum then make the
;; goal to find vector.
(defrule make_goal_find_vector_ang_momentum
  (logical (goal-problem (is find-ang-momentum)(applied-to ?b)(Time $?t1))
        (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t1)))
=>
  (assert (subgoal (is find-vector)(body ?b)(Time $?t1))))


;; If the problem goal is to find the avg vel then make the
;; goal to find vector.
(defrule make_goal_find_vector_avg_vel
  (logical (goal-problem (is find-avg-vel)(applied-to ?b)(Time $?t1))
        (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t1)))
=>
  (assert (subgoal (is find-vector)(body ?b)(Time $?t1))))


;; If the problem goal is to find theta then make the
;; goal to find vector.
(defrule make_goal_find_vector_incline
  (logical (goal-problem (is find-incline)(applied-to ?b)(Time $?t1))
        (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t1)))
=>
  (assert (subgoal (is find-vector)(body ?b)(Time $?t1))))


;; If the problem goal is to find a inst-vel
;; then make the goal to find vector.
(defrule make_goal_find_vector_inst_vel
  (logical (goal-problem (is find-inst-vel)(applied-to ?b)(Time $?t1))
        (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t1)))
=>
  (assert (subgoal (is find-vector)(body ?b)(Time $?t1))))
```

```
;;if the problem has a body, then find vectors on that body
(defrule make_goal_find_vector_on_body
  (logical (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t)))
  (not (subgoal (is find-vector)(body ?b)(Time $?t)))
=>
  (assert (subgoal (is find-vector)(body ?b)(Time $?t))))


;; If the problem goal is to find torque then make the
;; goal to find vector.
(defrule make_goal_find_vector_torque
  (logical (goal-problem (is find-torque)(applied-to ?b)(Time $?t1))
        (body (obj ?b)(decision-pt ?n1)(choice ?n2)(Time $?t1)))
=>
  (assert (subgoal (is find-vector)(body ?b)(Time $?t1))))


;; If a centripetal force exists make goal use newtons
(defrule make_goal_use_centri_newtons
  (logical (subgoal (is find-vector-properties)(body ?b)(v_var ?v)
          (Time $?t))
        (circular-motion (obj ?b) (Time $?t))
        (vector (kind force) (itsname ?i)(v_var ?v) (applied-to ?b)
          (applied-by ?s) (Time $?t)))
=>
  (assert (subgoal (is use-newton)(body ?b)(Time $?t))))

;;if a goal is to find torques, the use components
(defrule make_goal_use_torque_components
  (logical (goal-problem (is find-torque)(applied-to ?b)(Time $?t)))
=>
  (assert (subgoal (is find-sum-components)(body ?b)(Time $?t))))
```