

**Qualitative Behavior Prediction for Simple Mechanical Systems**

by

Jonathan P. Pearce

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degrees of

Bachelor of Science in Computer Science and Engineering

and Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2001

© Massachusetts Institute of Technology 2001. All rights reserved.

Author.....  
Department of Electrical Engineering and Computer Science  
May 23, 2001

Certified by.....  
Randall Davis  
Thesis Supervisor

Accepted by.....  
Arthur C. Smith  
Chairman, Department Committee on Graduate Theses

# Abstract

Programs that can reason about a particular domain of problems can be helpful to designers and engineers working in those areas. In this vein, several attempts have been made to create a qualitative mechanical simulator – a program that can reason about mechanical systems and describe their possible behavior without solving a set of equations. However, these systems often rely on information such as sizes, shapes and distances that must be precisely specified to produce an accurate result. This thesis demonstrates a program called ARCHIMEDES that can qualitatively predict the behavior of a simple mechanical system using knowledge about the components that make up the system as well as the system's topology. It takes as input a list of the system's components and interconnections and outputs a text description of the system's behavior when an external force acts upon the system. It also makes inferences about the effects and purpose of the system. ARCHIMEDES'S use of topology to analyze mechanical systems allows for a high tolerance of ambiguity in the description of the system and its components.

# Chapter 1

## Introduction

When people want to understand how a mechanism works, they often study a diagram of it. The diagram simplifies the device by eliminating details that are unrelated to its function, while emphasizing its structure and kinematics. Looking at the diagram, people can visualize how different parts of the mechanism will move under certain conditions and how the combination of these movements reveals the purpose of the device itself.



**Figure 1-1:** A simple pulley system.  
(Halliday, p. 182)

Consider, for example, the simple pulley system in Figure 1-1. If someone were asked to describe its behavior when the rope is pulled, he might say something like “The rope gets pulled down, turning the first

pulley counterclockwise, then turning the second pulley clockwise, and pulling it up as well. Then the pulley pulls up on the second rope, which also pulls up the weight.”

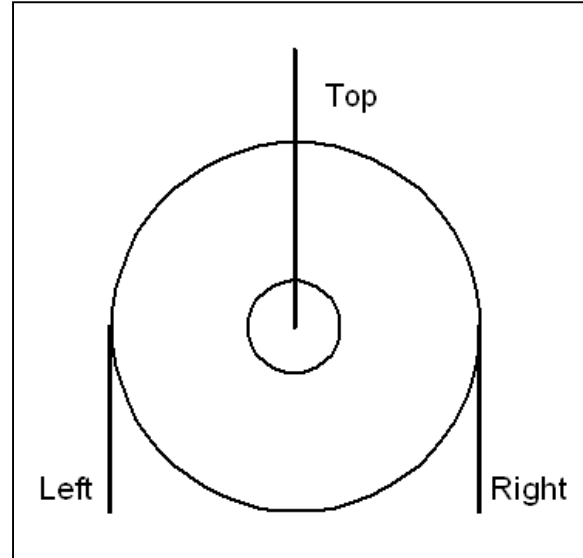
This small bit of reasoning implicitly includes a number of other inferences that are based on people’s knowledge about the properties of ropes and pulleys, as well as the conventions of the diagram. For example, people would assume that one end of the long rope is attached to the ceiling, which is immovable. The diagram would likely filter out variations in friction and tensile strength. They would know that a pulling force on one end of a straight segment of rope will cause a pulling force on whatever the other end is attached to or supporting. And, they might know that the downward force from the weight is divided evenly between the two rope segments coming out of the lower pulley. Some of these inferences are necessary to determine the basic operation of the device, that a strong enough pull on the rope will raise the weight. Others allow for the discovery that a pull weaker than the gravitational force on the weight (one half as strong) is sufficient to raise the weight.

## **1.1 The task**

While this sort of qualitative reasoning is easy for people to do, exactly how it is done is not obvious. The task in creating ARCHIMEDES is to formalize the qualitative reasoning process over a specific domain of mechanical systems. Before we can think about how to do this, we must decide what we mean by “reasoning” about a mechanical system. In this paper, a program can be said to reason about a mechanical system if it receives a description of the system and its inputs, manipulates a model of the system according to well-defined principles, and outputs a correct description of its behavior.

As input, ARCHIMEDES receives a text file that lists all the components of the system and their relevant properties, all the connections between the components, and the initial input force to the system.

ARCHIMEDES has built-in models of an assortment of common mechanical components. These components act according to simple rules, transmitting forces to each other and moving a68



**Figure 1-2:** A simplified model of a pulley.

According to those forces. Consider some of the rules followed by a simplified pulley, shown in Figure 1-2.

- If something is pulling on the left, then pull equally on the right.
- If something is pulling on the right, then pull equally on the left.
- If something is pulling on the top, then pull half as much on each of the left and right sides.
- If something is pulling on the left and the right, then pull twice as hard as the weaker of the two on the top.
- If the right side is being pulled and the left side is free, turn clockwise.
- If the left side is being pulled and the right side is free, turn counterclockwise.

Ropes are simplified by dividing them into segments, so that each segment interacts only with other components at its two ends. The long rope in the example would be divided into three segments.

ARCHIMEDES analyzes systems like this pulley system by applying each component's rules, propagating information about forces and motion around the system. First, it looks at the system in a static state, with the rope being held still. It transmits the weight's gravitational force to the rope it is hanging from, then to the pulley that rope hangs from, and to the rest of the components in the system. Then, it examines the effect of tugging on the rope in much the same way, transmitting the impulse from component to component according to the rules of each one, until it reaches the weight, which has no component to pass the impulse to, and must therefore move up.

## **1.2 Motivation**

The purpose of this research is twofold – first, to enrich the design process by providing qualitative information about the dynamics of mechanical systems, and second, to explore the qualitative physics problem in a new way, in an attempt to better understand human reasoning.

### **1.21 Conceptual design assistance**

The current design tools available to mechanical engineers are not well suited for the early, conceptual stages of design. Inputting a mechanical system into a CAD program and examining its behavior quantitatively is time-consuming and requires entering

precise parameters for every component. Changing the structure of the system once it has been entered into the computer is also tedious. For this reason most engineers do their initial designs on paper, based on a mixture of equations and intuition.

A qualitative simulator like ARCHIMEDES, coupled with a sketch recognizer, could provide instant feedback about the changes in a system's behavior as components are added, moved, or altered.

## **1.22 Design rationale capture**

When designing mechanical devices, it is often helpful to capture information about why certain design decisions were made, in order to create future versions that will not break old functionalities.

Designing a device with ARCHIMEDES can help, if the designer saves versions of the device as it is designed, along with the descriptive information that results from predicting the behavior of the device in ARCHIMEDES with various input forces.

## **1.23 A new approach**

ARCHIMEDES is also an attempt to look at the qualitative physics problem at the level of mechanical components. Most research in the field, with respect to mechanical systems, has focused on the shape, size, and location of components like gears, pulleys, and levers. ARCHIMEDES sees components as "black boxes," and makes inferences based on knowledge about the typical behavior of these components, as well as the interconnections between them (the topology of the overall system).

# Chapter 2

## Goals of ARCHIMEDES:

The objective of ARCHIMEDES is to qualitatively predict the behavior of a mechanical system. The program does this by examining a description of the system and generating a textual explanation of the prediction. ARCHIMEDES is inspired by a few observations about the way people mentally simulate mechanical systems, and follows a set of guiding principles about how these observations should be used.

### 2.1 Assumptions and observations

Fundamental to the design of ARCHIMEDES are several observations of how people simulate these systems mentally, as well as some common-sense assumptions. These are:

- All reasoning should be tolerant of the qualitative nature of the diagram. People do not use the exact size or position of objects in a diagram to predict the qualitative behavior of mechanical systems.
- Mechanical systems are constructed out of simple components whose properties are easily understood.

- To reduce complexity and the chance of error, systems should be understood in a static state before introducing motion.

## **2.11 All reasoning should be tolerant of the qualitative nature of diagrams.**

ARCHIMEDES is intended to be a qualitative reasoning engine. It predicts the behavior of mechanical systems that are described with the same level of detail as might be present in a simple, hand-drawn diagram. Thus, ARCHIMEDES has no notion of the quantitative position or scale of the components in the systems it analyzes, just as a human would not rely on a ruler or protractor to qualitatively describe the behavior of a system that has been sketched on a piece of paper. Instead, it relies primarily on the topology of the system to make its inferences.

ARCHIMEDES takes as input a description of each device in the system and the connections between them. The description of a device does not include its absolute position, and so the program does not use a coordinate system, eliminating the possibility of using vector arithmetic and trigonometric calculations to analyze how forces propagate through the system.

Nor does the input description include any notion of scale. For ease of implementation, some components are given a value for their size or weight, but this value exists only for comparison purposes, for example to distinguish larger and smaller gears in order to determine whether they produce a downshift or upshift.

Rather than relying on arithmetic values, which are often unknown in a simple diagram, ARCHIMEDES uses its knowledge of component behavior, as well as qualitative

properties of the forces acting on the components, in order to generate a prediction of the system's behavior. Certain combinations of motion-inducing forces and reaction forces will generate certain types of behavior in different components, regardless of the exact direction or strength of the forces.

## **2.12 Mechanical systems are constructed from simple components.**

ARCHIMEDES focuses on the connections between the components in a system to predict its behavior. On a mechanical component, there are usually only a few discrete areas that interact with other components, e.g. on a gear, the axis and a few of the teeth. For our purposes, the shape of the gear is not essential to understanding its behavior; the essential part is what is occurring at these areas; what forces is the component receiving and transmitting at these areas.

ARCHIMEDES views a mechanical component as an primitive black box that interacts with other components through certain points of interaction, defined for each component. Behavior is generated by the propagation of forces and motion through these points from component to component. ARCHIMEDES relies on simple rules about how standard components work (a tug on one end of a pulley also pulls the other end) in order to extrapolate the behavior of all the components in the system.

## **2.13 Systems should be understood in a steady state before introducing motion.**

When predicting the behavior of a system, ARCHIMEDES first examines the forces running through the system at steady state, when no motion or other physical changes to the system are occurring. The initial motion-inducing impulse on the system is introduced only once all other forces are accounted for.

This strategy helps to prevent the propagation of erroneous information around the system. For example, an object might be prematurely considered to be moving if all the forces on it are not known yet, causing other incorrect inferences to be made about components, resulting in an incorrect analysis and prediction of the system's behavior. Waiting for all the forces acting on a component to be accounted for before propagating the effects of motion improves the accuracy and simplicity of the program.

## **2.2 Flexibility**

ARCHIMEDES has been designed to be highly modular to facilitate the expansion of the ARCHIMEDES architecture itself and possible interfaces with other programs, like a sketch recognizer or a graphical simulator,

### **2.21 Modularity**

ARCHIMEDES analyzes mechanical systems using knowledge about the behavior of the systems' components. All the mechanical components ARCHIMEDES knows about

interact through a common interface that allows them to pass information about forces and motion between them without regard to what specific kind of components they are. In other words, each component does not have to contain any knowledge about any other component. This common interface allows new components to be added to the program, and existing objects modified, without having to redo the whole program.

ARCHIMEDES is also designed to be combined easily with other programs. It takes as input a single text file with a few pieces of information about each component and their relationships. A sketch-recognizing program could be made to generate this kind of file in order to provide a more intuitive simulation experience.

## **2.22 Tolerance of ambiguity**

ARCHIMEDES does not assume that the system it is analyzing has been described to accurate scale. When people look at diagrams that are not drawn to scale, it is acceptable for them not to be able to resolve certain questions. For example, in a complicated gear system, with many different sizes of gears, it may be impossible to tell whether the overall result has been a downshift or an upshift.

By not considering the shape, absolute size or position of the components in a system, ARCHIMEDES does take the chance of eliminating a significant amount of information that could conceivably be useful in resolving ambiguities that may arise, such as a question of the relative strengths of two forces acting on a component. However, incorrectly resolving an ambiguity is much worse than recognizing that an ambiguity cannot accurately be resolved.

## **2.3 Limitations of ARCHIMEDES**

This section identifies some of the things outside the scope of ARCHIMEDES in its current implementation. It was necessary to place some limits on the task of qualitative behavioral prediction in order to achieve results on a well-defined set of sample systems.

### **2.31 Limited understanding of devices**

ARCHIMEDES understands a component in a way that is different from understanding the complete physical behavior of that device. Instead, it has a very abstract characterization of the behavior of each component type, based on common-sense guidelines about how each component is commonly used. For example, gears commonly interact only with other gear-toothed objects and with axles and do not commonly roll on smooth surfaces or fall through the air. ARCHIMEDES would not be able to predict the behavior of a system that relied on that or other unusual uses for standard components.

### **2.32 Restricted domain of mechanical systems**

ARCHIMEDES can successfully analyze a limited range of mechanical systems. First, of course, it can only analyze systems containing the component types about which it has specific knowledge.

Second, because ARCHIMEDES relies primarily on the connections between devices in order to predict behavior, it can analyze only systems whose components are

connected to each other in a way that it explicitly understands. This does not mean that all components must be attached; a rope going around a pulley is considered a “connection” by ARCHIMEDES. However, the program is not capable of solving problems in which connections are dynamically created and destroyed, when the topology of the system changes. For example, ARCHIMEDES would not currently be well suited to solving a problem involving an object flying through the air and hitting another object.

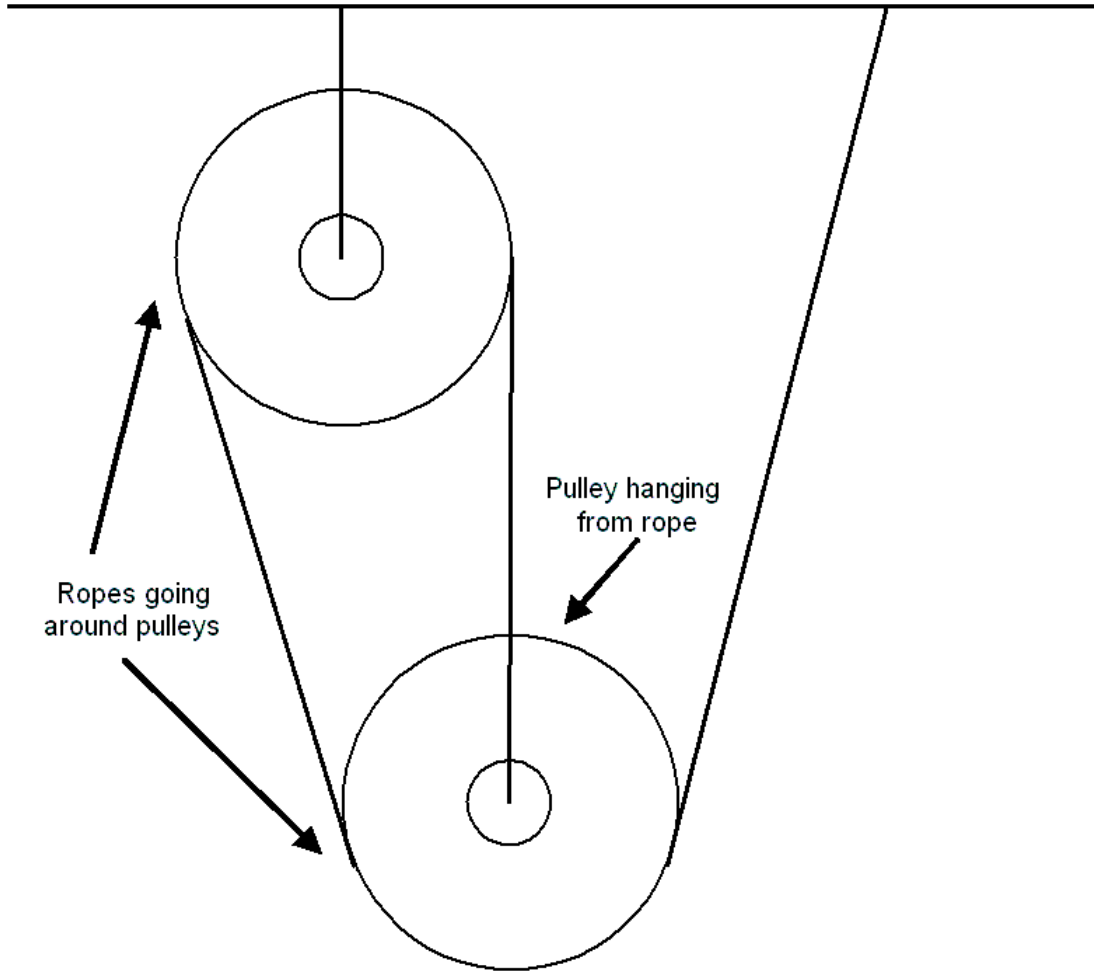
# Chapter 3

## ARCHIMEDES at work

This section explains in detail the reasoning that ARCHIMEDES uses predicts the behavior of mechanical systems. The implementation of this reasoning is laid out in chapter 4.

### 3.1 ARCHIMEDES' universe

ARCHIMEDES reasons qualitatively about a mechanical system based on its topology. To do this, it views these systems as a collection of simple components. These components are connected to each other at certain connection points: the axis of a gear, for example, or the end of a length of rope. A connection point represents a part of a component that connects to another component, but does not always have an exact physical equivalent. For example, a connection point might represent a point on the teeth of a gear that mesh with the teeth of another gear. This connection point between the two gears stays constant even though the individual teeth that form the physical connection change as the gears turn.



**Figure 3-1:** An example of the same components having different connection relationships.

When two components are connected at their respective points, this connection is assigned a type, which represents the relationship that is occurring at this connection point. Sometimes the same two components can be connected in different ways. The relationship between a rope that goes around a pulley is different from the relationship between a pulley that is hanging from a rope. See Figure 3-1.

When the system is held in a static state, the components exert forces on each other. Then, the system receives a motion impulse, the theoretically infinitely small push or pull that makes it move. There are two types of motion impulses – motion-inducing impulses (impulses that cause motion) and reaction impulses (impulses created when a motion-inducing impulse bounces off a fixed object)

## **3.2 Input to the system**

The input source to ARCHIMEDES is a text file that details the topology of the mechanical system to be analyzed. This input file is divided into three sections:

- A list of the components in the system and their relevant properties
- A list of the connections between the components – the specific points at which they are connected and the connection relationship that exists.
- The input force on one of the components.

When ARCHIMEDES is run, it parses this file and creates an internal representation of the system, instantiating component objects and codifying their relationships. Once this representation has been created it begins its analysis.

## **3.3 Library of components**

This section details the selection of mechanical components that ARCHIMEDES currently knows and can reason about. For convenience, they have been divided into four

categories of increasing complexity, based on the number of connection points each component contains.

### **3.31 One-point components**

These three components – the weight, the anchor, and the input – each contain only one connection point at which they can interact with other components.

#### **3.311 *Hanging Weight***

The hanging weight generates a gravitational force from its only connection point. When it is defined, it must be given a value for its weight (usually an arbitrary value of 1). See Figure 4.

##### **Static rules:**

- Output a pulling force equal to the weight of the weight.

##### **Motion rules:**

- If a motion impulse is received

Then move in the direction of the impulse.

#### **3.312 *Anchor***

The Anchor is used to fix one connection point of a component in place. For example, to attach one end of a rope to a wall or ceiling, it should be connected to an anchor.

**Static rules:**

- If a force is received,  
Then output an equal, opposite force.

**Motion rules:**

- If a motion impulse is received  
Then output an equal, reaction motion impulse.

**3.313 Input**

The input is the object that produces the motion impulse – the imaginary hand that pulls on a rope or turns a gear to set a system in motion. In the static state, when it is holding the system still, it acts like the anchor.

**Static rules:**

- If a force is received,  
Then output an equal, opposite force.

**Motion rules:**

- Output a motion impulse in the opposite direction of the force being received.

**3.32 Two-point Components**

These three components interact with others at two points and exhibit more complex behavior than the one-point components.

### 3.321 Axle

ARCHIMEDES's implementation of an axle contains two connection points, Start and End, at which gears or other components can be attached. Axles can transmit and receive only clockwise and counterclockwise forces (torques). See Figure 3-2.

#### Static rules:

- If a force is received at one point  
Then propagate that force to the other point.
- If a force is received at both points  
Then propagate each force to the opposite point.

(The forces must be equal in the static state)

#### Motion rules:

- If a motion impulse is received at one point  
And the other point is free to move  
Then turn in the direction of the motion impulse.
- If a motion impulse is received at one point  
And the other point is not free to move  
Then propagate that impulse to the other point
- If equal-strength motion impulses are received at both Start and End  
Then propagate the impulse received at Start out from End  
And propagate the impulse received at End out from Start

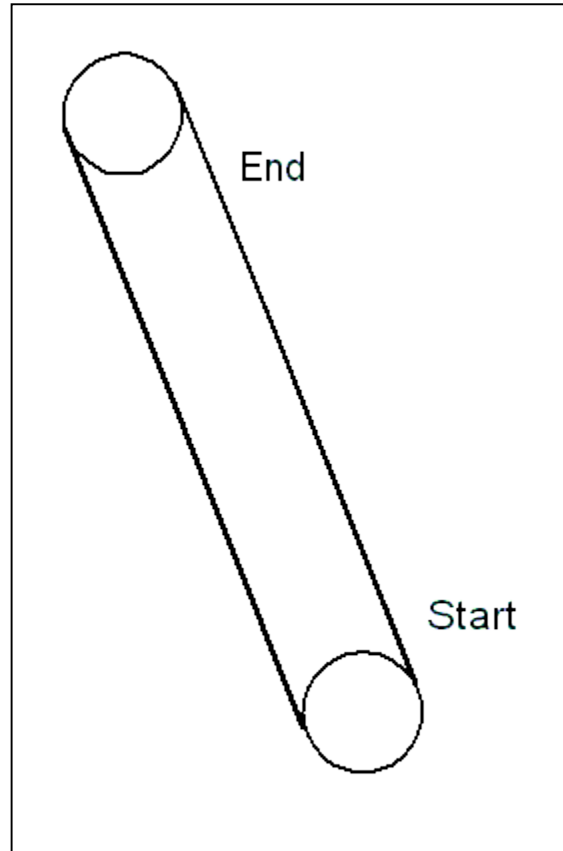
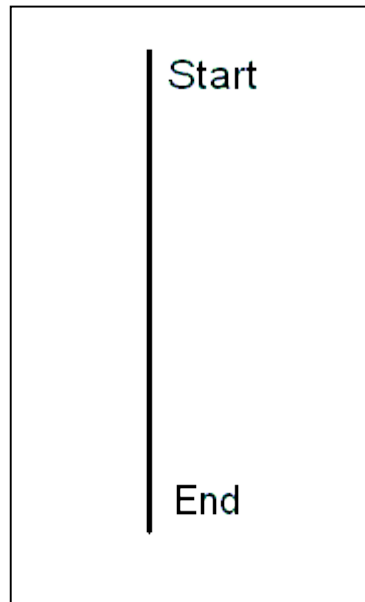


Figure 3-1: The Axle. (3D representation)

- If unequal-strength motion impulses are received at both points  
Then propagate the stronger one out from the opposite point.

### 3.322 Rope Segment

The rope segment is a straight segment of rope that contains two connection points, Start and End. It can connect to other objects at either point with connection relationships of Attached, Wound Clockwise, Wound Counterclockwise, and Pulley System. It can transmit and receive only pulling forces. A pushing force on a rope segment is ignored because pushing on the end of a rope is qualitatively the same as simply not pulling on it. See Figure 3.3.



**Figure 3-3:** The Rope.

#### Static rules:

- If a pulling force is received at one point  
Then propagate that force to the other point.
- If a pulling force is received at both points  
Then propagate each force to the opposite point.

#### Motion rules:

- If a motion impulse is received at one point  
And the other point is free to move  
Then move in the direction of the motion impulse.
- If a motion impulse is received at one point

And the other point is not free to move  
Then propagate that impulse to the other point

- If equal-strength motion impulses are received at both Start and End  
Then propagate the impulse received at Start out from End  
And propagate the impulse received at End out from Start
- If unequal-strength motion impulses are received at both points  
Then propagate the stronger one out from the opposite point.

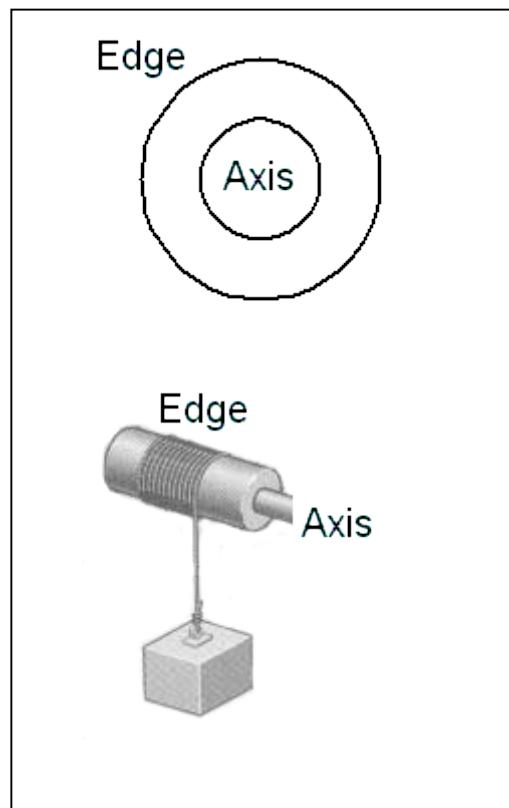
### 3.323 Winch

The Winch is used to transform a torque into a pulling force on a rope, or vice versa. It has two connection points – Edge (the surface the rope is wrapped around) and Axis. Edge accepts only the connection relationships Wound Clockwise and Wound Counterclockwise. Axis only accepts connection type Axis. See Figure 3-4.

#### Static rules:

- If a force is received at one point  
Then propagate that force to the other point.
- If a force is received at both points  
Then propagate each force to the other point.

(The forces must be equal in the static state.)



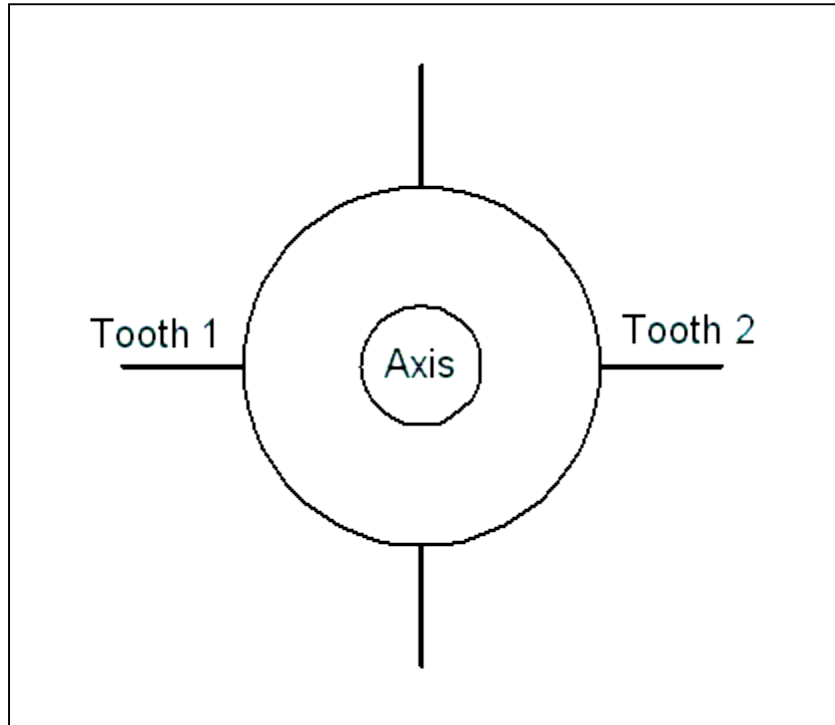
**Figure 3-4:** A diagram of the Winch model and a picture of a winch.

**Motion rules:**

- If a motion impulse is received at one point  
And the other point is free to move  
Then turn in the direction of the motion impulse.
- If a motion impulse is received at one point  
And the other point is not free to move  
Then propagate that impulse to the other point
- If equal-strength motion impulses are received at both Start and End  
Then propagate the impulse received at Start out from End  
And propagate the impulse received at End out from Start
- If unequal-strength motion impulses are received at both points  
Then propagate the stronger one out from the opposite point.

**3.33 Three-point Components****3.331 Gear**

The gear contains three connection points, Tooth1, Tooth2, and Axis. Tooth1 and Tooth 2 do not represent specific locations on the edge of the gear, but simply two different points at which other geartoothed objects can be attached. Tooth1 and Tooth2 accept only connections of type Geartooth. Axis accepts only connections of type Axis. See Figure 3-5.



**Figure 3-5:** The Gear

**Static rules:**

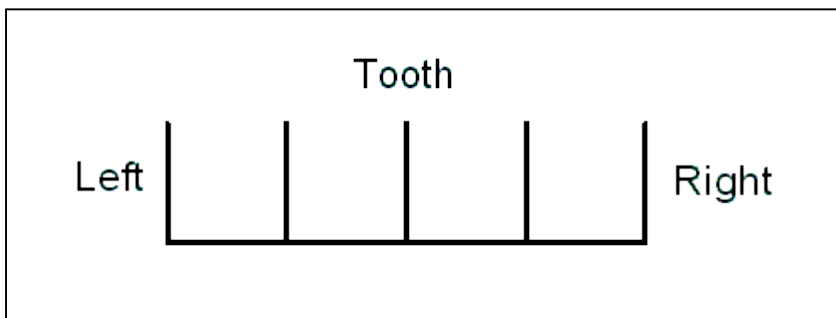
- If a force is received at one point  
And only one other point is connected to another component  
Then propagate that force to that point.
- If forces are received at two points  
And the third point is not connected to another point  
Then propagate each force to the opposite point that is receiving a force.

**Motion rules:**

- If a motion impulse is received at one point  
And the other points are free to move  
Then turn in the direction of the motion impulse.
- If a motion impulse is received at one point  
And one other point is not free to move  
Then propagate that impulse to that point.

- If a motion-inducing impulse is received at one Tooth point  
And a reaction impulse is received at the other Tooth point in the other direction  
And Axis is free to move  
Then roll in the direction of the motion-inducing impulse.
- If a motion-inducing impulse is received at one Tooth point  
And a reaction impulse is received at the other Tooth point in the other direction  
And Axis is not free to move  
Then propagate the motion-inducing impulse out from Axis.

### 3.332 Flat Gear



**Figure 3-6:** The Flat Gear

The flat gear is a flat surface with gear teeth on it. It has three connection points, Left, Right and Tooth. Tooth accepts only connections of type Gear tooth; Start and End accept several different connection types. For the flat gear, the clockwise direction is considered to be from left to right, as if the flat gear were a part of a very large round gear. See Figure 3-6.

#### Static rules:

- If a force is received at Left or Right  
Then propagate that force to Right or Left, respectively  
And propagate the appropriate force from Edge (clockwise or counterclockwise).
- If a clockwise force is received at Edge  
Then propagate a pulling force from Left

And propagate a pushing force from Right.

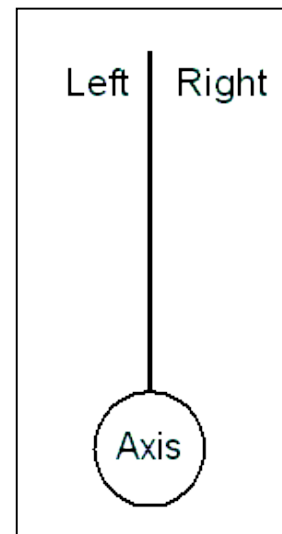
- If a counterclockwise force is received at Edge  
Then propagate a pushing force from Left  
And propagate a pulling force from Right.
- If a force is received at both Left and Edge  
Then propagate each force to the other point.
- If a force is received at both Right and Edge  
Then propagate each force to the other point.

**Motion rules:**

- If a motion impulse is received at one point  
And the other points are free to move  
Then move in the direction of the motion impulse.
- If a motion impulse is received at one point  
And one other point is not free to move  
Then propagate that impulse to that point.

**3.333 Lever Segment**

The lever segment is one half of a lever. A complete lever can be constructed from two lever segments connected by an axle. Using lever segments allows the two ends of the lever to have different lengths to increase or decrease torque. Levers have three connection points, Left, Right and Axis. Axis can accept only connections of type Axis; Left and Right can accept several different connection types. See Figure 3-7.



**Figure 3-7:** The Lever Segment

**Static rules:**

- If a clockwise force is received at Axis  
Then propagate a pulling force from Left

And propagate a pushing force from Right.

- If a counterclockwise force is received at Axis  
Then propagate a pushing force from Left  
And propagate a pulling force from Right.
- If a force is received at Left or Right  
Then propagate that force to Right or Left, respectively.  
And propagate the appropriate force from Axis (clockwise or counterclockwise).
- If a force is received at both Left and Edge  
Then propagate each force to the other point.
- If a force is received at both Right and Edge  
Then propagate each force to the other point.

**Motion rules:**

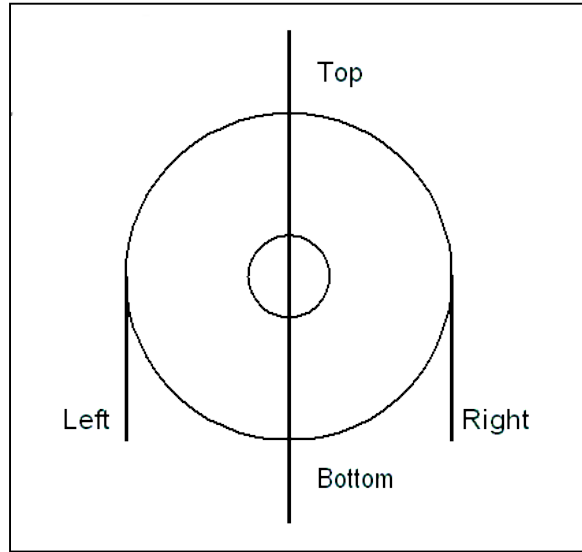
- If a motion impulse is received at one point  
And the other points are free to move  
Then move in the direction of the motion impulse.
- If a motion impulse is received at one point  
And one other point is not free to move  
Then propagate that impulse to that point.

### **3.34 Four-point Components**

#### **3.341 Pulley**

The pulley exhibits the most complex behavior of all components and is the only component with four connection points. Its connection points are Left, Right, Top and Bottom. For ease of explanation, we will refer to Left and Right as rope points, and Top and Bottom as body points. See Figure 3-8.

A pulley can be assigned a state of Fixed or Free, depending on whether it is attached to a ceiling or wall, or supported on the bottom by a rope and thus able to move with the rope. Of course, the pulley is able to turn no matter which state it is in. If the pulley is not connected to other components at either Top or Bottom, it is assumed to be a fixed pulley, since its motion would not accomplish anything.



**Figure 3-8:** The Pulley

**Static rules:**

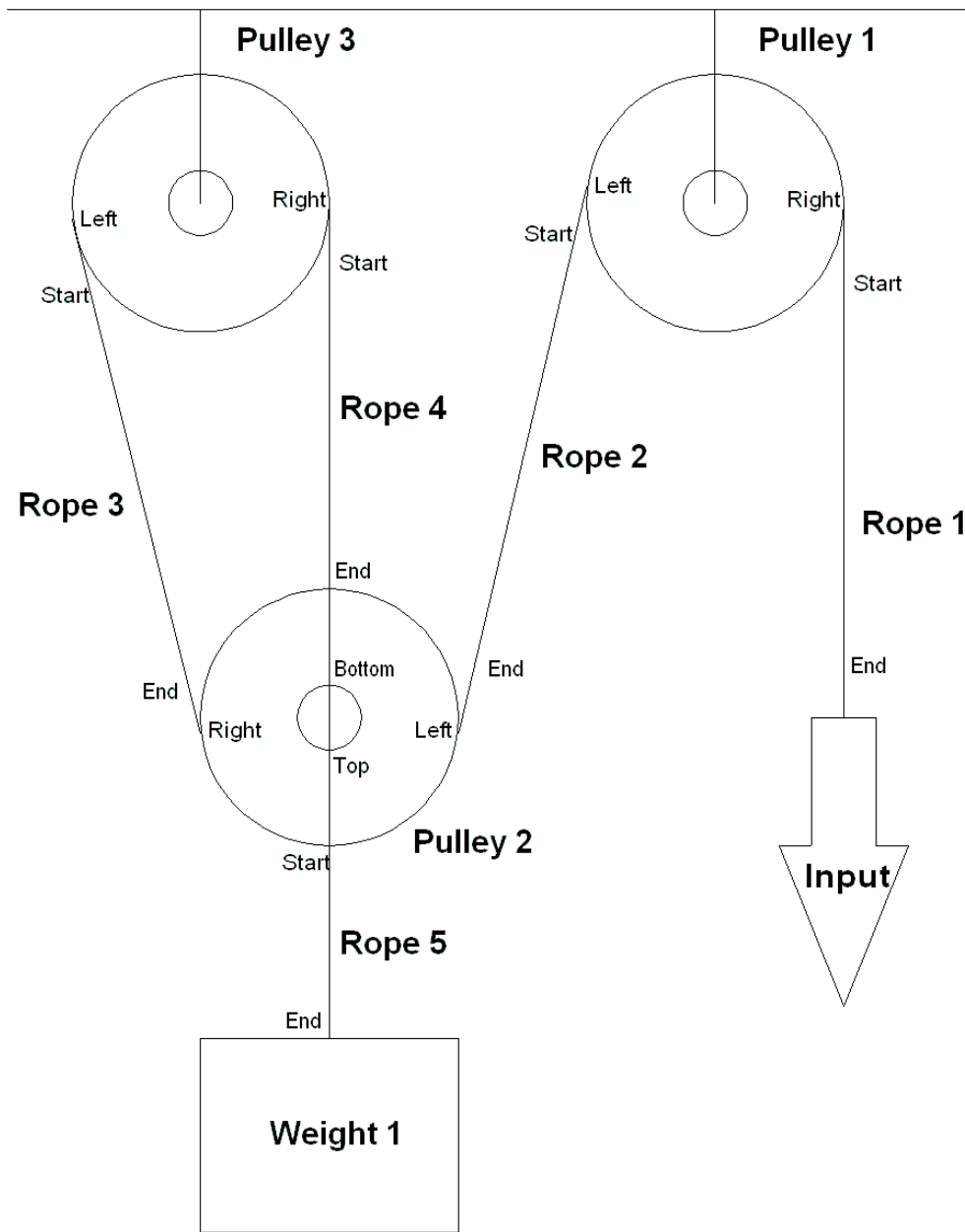
- If a pulling force is received at one rope point  
Then propagate that force to the other rope point.
- If a pulling force is received at Top  
And Bottom is unattached  
Then output pulling forces half as strong from both rope points.
- If pulling forces are received at both rope points  
(These two forces must be equal in the static state)  
And Bottom is unattached  
Then output a pulling force twice as strong from Top.
- If a pulling force is received at Top  
And Bottom is attached  
Then output pulling forces one-third as strong from both rope points and Bottom.
- If pulling forces are received at both rope points and Bottom  
(These three forces must be equal in the static state)  
Then output a pulling force three times as strong from Top.

**Motion rules:**

- If a motion impulse is received at one rope point  
And the other rope point is free to move  
Then turn in the direction of the motion impulse  
(clockwise if right, counterclockwise if left)
- If a motion impulse is received at one rope point  
And the other rope point is not free to move  
Then propagate that impulse to the other rope point
- If a motion-inducing impulse is received at one rope point  
And a reaction impulse is received at the other rope point  
And the body points are free to move  
Then turn in the direction of the motion impulse  
(clockwise if right, counterclockwise if left)  
And move in the direction opposite to the side of the pulley that has the rope around it.  
(This usually happens when a rope is being used to raise a pulley)
- If a motion-inducing impulse is received at one rope point  
And a reaction impulse is received at the other rope point  
And the body points are not free to move  
Then output a motion impulse from Top and/or Bottom.

### **3.4 An example: ARCHIMEDES predicts the behavior of a pulley system.**

This example of a pulley system illustrates the steps that ARCHIMEDES takes to predict physical behavior. A diagram of the system is shown in Figure 3-9 on the next page. This system is constructed of a weight, three pulleys and five segments of rope. We will call the segments “ropes” for simplicity, but it is important to remember that these rope segments have properties different from actual ropes; in fact four of them are segments of a single rope.



**Figure 3-9:** The pulley system example.

The first thing ARCHIMEDES does is to identify the force-generating components in the system; in this case the hanging weight that introduces a gravitational force to the system.

1. Weight 1 outputs a pulling force equal to its weight.
2. Rope 5 receives a pulling force from Weight 1 at End, and then it propagates that force to Start.
3. Pulley 2 receives a pulling force from Rope 5 at Top. Please note that since this pulley is inverted, its connection point Top is actually towards the bottom of the pulley in the diagram, and Left and Right are on the sides opposite their names. Pulley 2 is connected to other components at points Left, Right, and Bottom, and so it propagates a force with  $1/3$  the strength to each of these connection points.
4. Rope 2 receives a pulling force from Pulley 2 at End and propagates it to Start.
5. Rope 3 receives a pulling force from Pulley 2 at End and propagates it to Start.
6. Rope 4 receives a pulling force from Pulley 2 at End and propagates it to Start.
7. Pulley 1 receives a pulling force from Rope 2 at Left and propagates it to Right.
8. Pulley 3 receives a pulling force from Rope 3 at Left and another pulling force from Rope 2 at Right. It propagates each force to the opposite side, outputting equal forces to both Left and Right.
9. Rope 1 receives a pulling force from Pulley 1 at Start and propagates it to End.
10. Rope 4 receives a pulling force from Pulley 3 at Start and propagates it to End.
11. Rope 3 receives a pulling force from Pulley 3 at Start and propagates it to End.
12. Input receives a pulling force from Rope 1 and returns an equal, opposite pulling force.
13. Pulley 2 receives a pulling force from Rope 4 at Bottom and does nothing because it has no rule for this situation.
14. Pulley 2 receives a pulling force from Rope 3 at Right and does nothing because it has no rule for this situation.
15. Rope 1 receives a pulling force from Input at End and propagates it to Start.
16. Pulley 1 receives a pulling force from Rope 1 at Right and propagates it to Left.
17. Rope 2 receives a pulling force at point Start and propagates it to End.
18. Pulley 2 is now receiving three pulling forces: one from Rope 2 at Left, one from Rope 3 at Right, and one from Rope 4 at Bottom. Pulley 2 propagates the sum of these equal forces to Top.
19. Rope 5 receives a pulling force from Pulley 2 at Start and propagates it to End.
20. Weight 1 receives a pulling force from Rope 5 and has nowhere to propagate it.

Weight 1 has no attached components to pass this force to (It is already outputting a force to Rope 5), and so this ends the examination of the system in the steady state.

Now ARCHIMEDES enters the second phase of its analysis, in which it predicts the motion of the components in the system. This is done by sending a motion-inducing impulse along the path created by the propagation of the static forces in the model. The direction of the impulse is the same as the direction of the force from one component to the next. In this example, the motion-inducing impulse is the tug on Rope 1, so the program begins there.

1. Input releases a motion-inducing impulse.
2. Rope 1 receives a motion-inducing impulse from Input at End and propagates it to Start.
3. Pulley 1 receives a motion-inducing impulse from Rope 1 at Right and propagates it to Left.
4. Rope 2 receives a motion-inducing impulse from Pulley 1 at Start and propagates it to End.
5. Pulley 2 receives a motion-inducing impulse from Rope 2 at Left and propagates it to Right.
6. Rope 3 receives a motion-inducing impulse from Pulley 2 at End and propagates it to Start.
7. Pulley 3 receives a motion-inducing impulse from Rope 3 at Left and propagates it to Right.
8. Rope 4 receives a motion-inducing impulse from Pulley 3 at Start and propagates it to End.
9. Pulley 2 receives a motion-inducing impulse from Rope 4 at Bottom and propagates it to Top.
10. Rope 5 receives a motion-inducing impulse from Pulley 2 at Start and propagates it to End.
11. Weight 1 receives a motion-inducing impulse from Rope 5 and has no connected components to pass the impulse to. Therefore, it must move in the direction of the impulse (towards Rope 5).
12. Rope 5 receives a message that Weight 1 has moved. For this instant (Weight 1 has moved, Rope 5 has not yet), Rope 5 is still receiving a motion impulse at Start and is no longer constrained at End because Weight 1 has moved. So Rope 5 now moves in the direction of the motion impulse, towards Start.
13. Pulley 2 receives a message that Rope 5 has moved, so it moves towards Bottom.
14. Rope 4 receives a message that Pulley 2 has moved, so it moves towards Start.

15. Pulley 3 receives a message that Rope 4 has moved, so it turns counterclockwise.
16. Rope 3 receives a message that Pulley 3 has turned, so it moves towards End.
17. Pulley 2 receives a message that Rope 3 has moved, so it turns counterclockwise.
18. Rope 2 receives a message that Pulley 2 has moved, so it moves towards Start.
19. Pulley 1 receives a message that Rope 2 has moved, so it turns clockwise.
20. Rope 1 receives a message that Pulley 1 has turned, so it moves towards End.
21. Input receives a message that Rope 1 has moved, which ends this phase of the program.

ARCHIMEDES notes from its motion analysis that the initial motion impulse on Rope 1 directly causes the motion of Weight 1.

It also sees from its earlier static analysis that the force coming out of Input is one-third the strength of the gravitational force on Weight 1. This value is possible to determine qualitatively because it depends only on the number of pulleys present in the system, and the topology of their connections. However, it is also based on the lack of any friction, and several other basic assumptions, and should not be taken to be exact in the final analysis.

Therefore, by the static and dynamic analyses of the system, the overall effect of the pulley system has been found to reduce the force necessary to move the weight. Consequently, ARCHIMEDES knows that the weight will move less than the distance that Rope 1 is pulled.

# Chapter 4

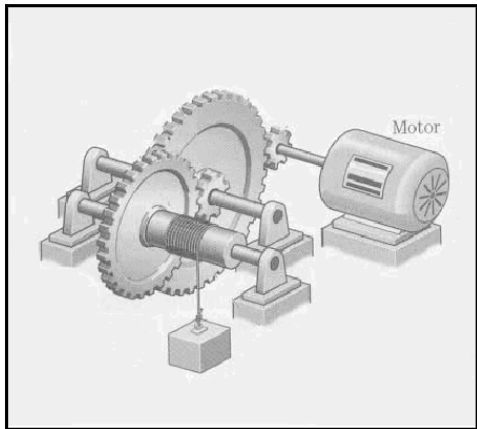
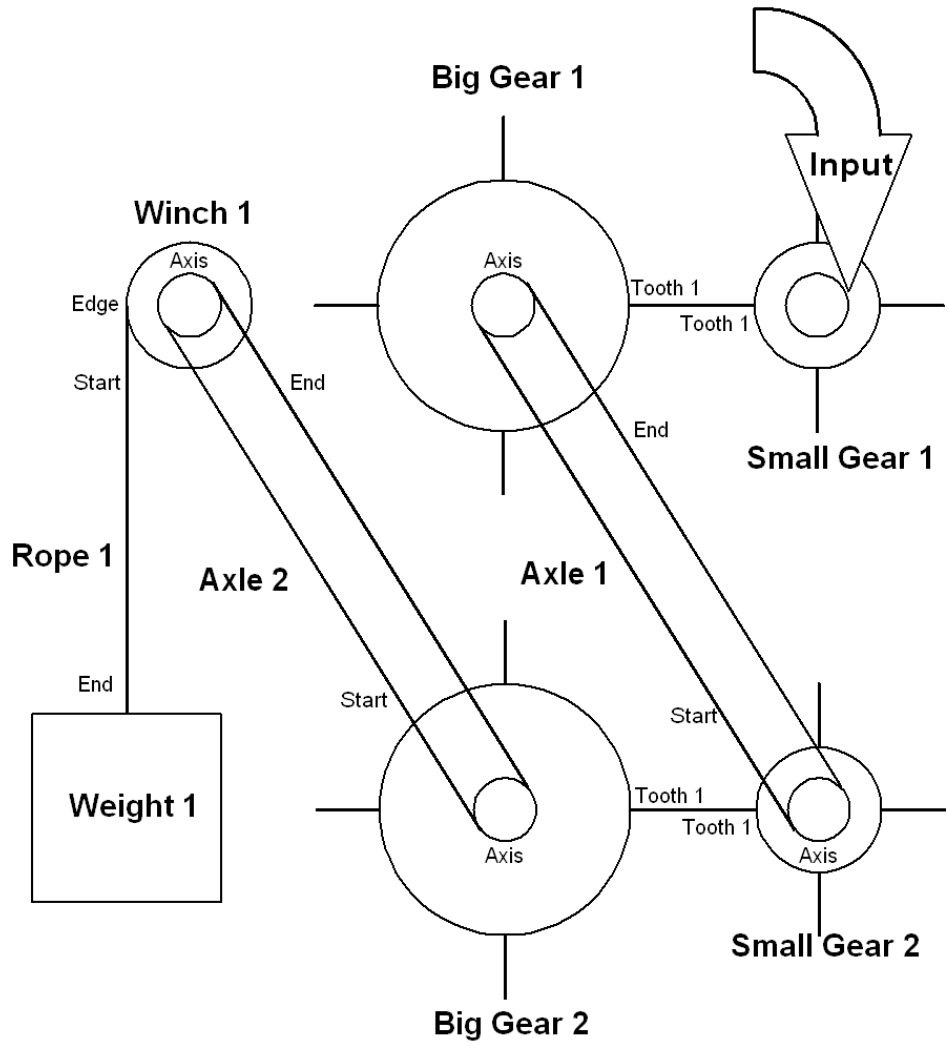
## More Examples

This section presents two more examples of ARCHIMEDES at work. The first is a gearbox system designed to increase torque. The second is a system of a lever and two pulleys that is intended to show distinctions between action and reaction impulses as well as an ambiguous result.

### **4.1 ARCHIMEDES predicts the behavior of a gear system.**

This system is constructed of four gears, two axles, a winch, and a rope. A diagram of the system is shown in Figure 4-1 on the next page. The gears have been given different radii so that the program can make inferences about the effects generated by pairs of different-sized gears. Again, ARCHIMEDES begins the static-state analysis by examining the weight:

1. Weight 1 outputs a pulling force equal to its weight.
2. Rope 2 receives a pulling force from Weight 1 at End and propagates it to Start.
3. Winch 1 receives a pulling force from Rope 1 at Edge and propagates it to Axis.



**Figure 4-1:** The gear system example. The diagram is analogous to the picture at the bottom left with the input force acting as the motor.

When a linear force is transformed into a torque, a record of that transformation stays with the force as it travels through the system. We will say that a force transformed into a torque by a component has been “torqued” by that component, and “untorqued” if transformed from a torque into a linear force. Therefore, this force has been torqued by Winch 1. Since the connection between Rope 1 and Winch 1 is the relationship “wound counterclockwise,” the pulling force from Rope 1 becomes a counterclockwise torque.

4. Axle 2 receives a counterclockwise torque from Winch 1 at End and propagates it to Start.
5. Big Gear 2 receives a counterclockwise torque from Axle 2 at Axis and propagates it to Tooth 1, transforming it back into a linear force. This force has now been torqued by Winch 1 and untorqued by Big Gear 2.
6. Small Gear 2 receives the force from Big Gear 2 at Tooth 1 and propagates it to Axis in a clockwise direction. Now, the force has been torqued by Winch 1 and Small Gear 2 and untorqued by Big Gear 2.
7. Axle 1 receives a clockwise torque from Small Gear 2 at Start and propagates it to End
8. Big Gear 1 receives a clockwise torque from Axle 1 at Axis and propagates it to Tooth 1. This force has now been torqued by Winch 1 and Small Gear 2 and untorqued by Big Gear 2 and Big Gear 1.
9. Small Gear 1 receives the force from Big Gear 1 at Tooth 1 and propagates it to Axis in a counterclockwise direction. Now, the force has been torqued by Winch 1, Small Gear 2 and Small Gear 1 and untorqued by Big Gear 2 and Big Gear 1.
10. Input receives a counterclockwise torque from Small Gear 1 and outputs a new, equal clockwise torque.
11. Small Gear 1 receives a clockwise torque from Input at Axis and propagates it to Tooth 1. This force has now been untorqued by Small Gear 1.
12. Big Gear 1 receives the force from Small Gear 1 at Tooth 1 and propagates it to Axis in a counterclockwise direction. This force has now been torqued by Big Gear 1 and untorqued by Small Gear 1.
13. Axle 1 receives a counterclockwise torque from Big Gear 1 at End and propagates it to Start
14. Small Gear 2 receives a counterclockwise torque from Axle 1 at Axis and propagates it to Tooth 1. This force has now been torqued by Big Gear 1 and untorqued by Small Gear 1 and Small Gear 2.
15. Big Gear 2 receives the force from Small Gear 1 at Tooth 1 and propagates it to Axis in a clockwise direction. This force has now been torqued by Big Gear 1 and Big Gear 2 and untorqued by Small Gear 1 and Small Gear 2.

16. Axle 2 receives a clockwise torque at from Big Gear 2 at Start and propagates it to End.
17. Winch 1 receives a clockwise torque from Axle 2 at Axis and propagates it to Edge, transforming it into a linear force. This force has now been torqued by Big Gear 1 and Big Gear 2 and untorqued by Small Gear 1, Small Gear 2 and Winch 1.
18. Rope 1 receives a pulling force from Winch 1 at Start and propagates it to End.
19. Weight 1 receives a pulling force from Rope 1 and has nowhere to propagate it.

Now, ARCHIMEDES examines the effects of the motion-inducing impulse.

1. Input releases a motion-inducing impulse.
2. Small Gear 1 receives a motion-inducing impulse from Input at Axis and propagates it to Tooth 1.
3. Big Gear 1 receives a motion-inducing impulse from Small Gear 1 at Tooth 1 and propagates it to Axis.
4. Axle 1 receives a motion-inducing impulse from Big Gear 1 at End and propagates it to Start.
5. Small Gear 2 receives a motion-inducing impulse from Axle 1 at Axis and propagates it to Tooth 1.
6. Big Gear 2 receives a motion-inducing impulse from Small Gear 2 at Tooth 1 and propagates it to Axis.
7. Axle 2 receives a motion-inducing impulse from Big Gear 2 at Start and propagates it to End.
8. Winch 1 receives a motion-inducing impulse from Axle 2 at Axis and propagates it to Edge.
9. Rope 1 receives a motion-inducing impulse from Winch 1 at Start and propagates it to End.
10. Weight 1 receives a motion-inducing impulse from Rope 1 at Start and cannot propagate it. Therefore, it must move in the direction of the force that Rope 1 is applying to Weight 1 (towards Rope 1).
11. Rope 1 receives a message that Weight 1 has moved, so it moves toward Start.
12. Winch 1 receives a message that Rope 1 has moved, so it turns clockwise.
13. Axle 2 receives a message that Winch 1 has turned, so it turns clockwise.
14. Big Gear 2 receives a message that Axle 1 has turned, so it turns clockwise.
15. Small Gear 2 receives a message that Big Gear 2 has turned, so it turns counterclockwise.
16. Axle 1 receives a message that Small Gear 2 has turned, so it turns counterclockwise.
17. Big Gear 1 receives a message that Axle 1 has turned, so it turns counterclockwise.
18. Small Gear 1 receives a message that Big Gear 1 has turned, so it turns clockwise.

19. Input receives a message that Small Gear 1 has moved, which ends this phase of the program.

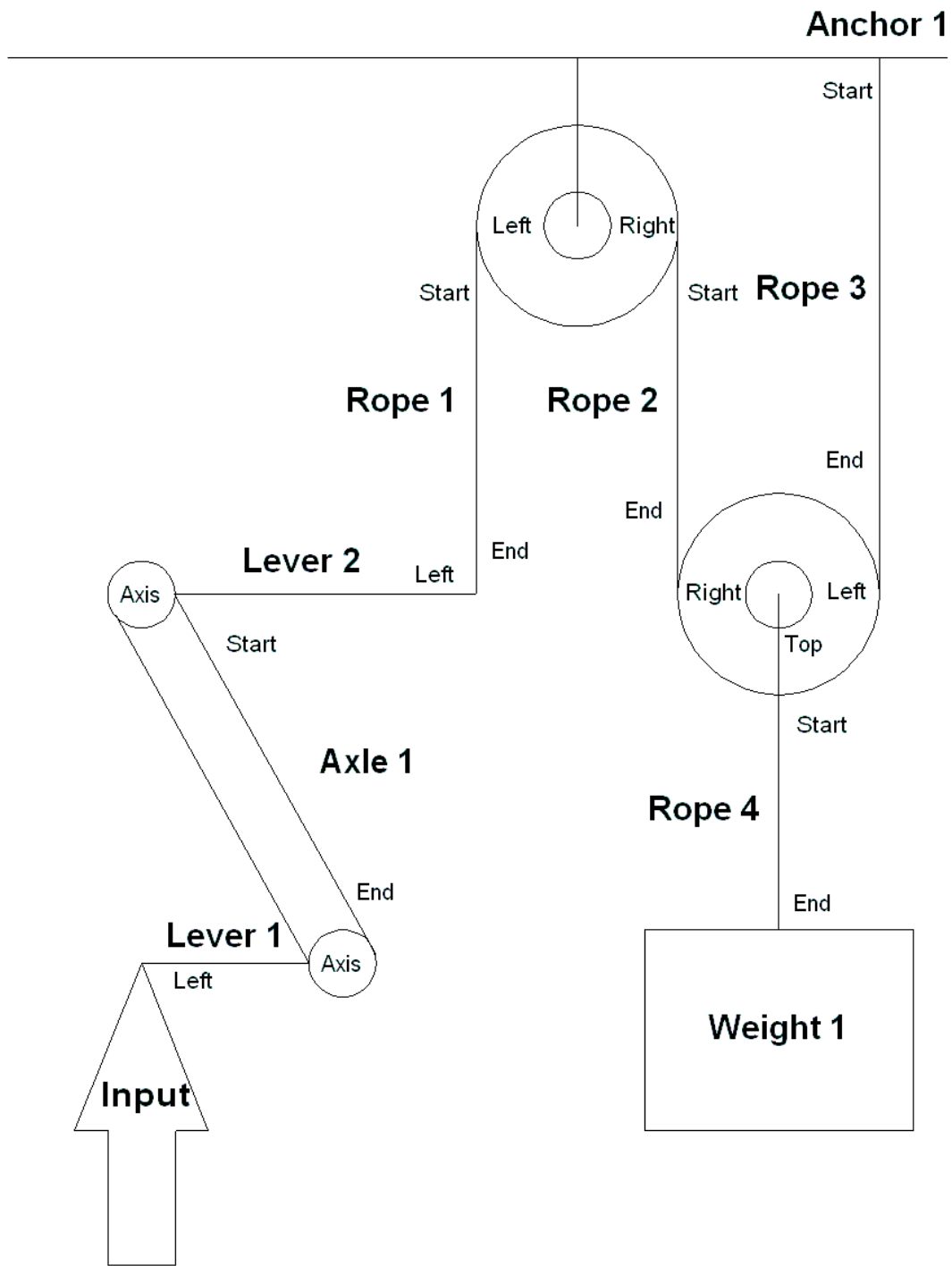
Archimedes notes from its motion analysis that the initial motion impulse on Rope 1 directly causes the motion of Weight 1.

Then, using the results of its static analysis, Archimedes compares the force coming out of Input with the force on Weight 1. It sees that the force on Weight 1 has been torqued by Big Gear 1 and Big Gear 2 and untorqued by Small Gear 1, Small Gear 2 and Winch 1. If each of the components that has torqued a force has a larger radius than each of the components that has untorqued it, then the result has been an increase in torque and a decrease in distance traveled. This condition is found to be true in this example, in which the input force represents a rapidly spinning motor using a gear system to slowly lift a weight. (The reverse case would have the opposite result.)

## **4.1 Another example: Reaction forces and ambiguity**

See Figure 4-2 on the next page for the diagram of this system. First, the static-state analysis:

1. Weight 1 outputs a pulling force equal to its weight.
2. Rope 4 receives a pulling force from Weight 1 at End and propagates it to Start.
3. Pulley 2 receives a pulling force from Rope 4 at Top and propagates a force half as strong to Left and Right.
4. Rope 3 receives a pulling force from Pulley 2 at End and propagates it to Start.
5. Rope 2 receives a pulling force from Pulley 2 at End and propagates it to Start.
6. Anchor 1 receives a pulling force from Rope 3 and returns an equal, opposite pulling force.
7. Pulley 1 receives a pulling force from Rope 2 at Right and propagates it to End.
8. Rope 3 receives a pulling force from Anchor 1 at Start and propagates it to End.
9. Rope 1 receives a pulling force from Pulley 1 at Start and propagates it to End.



**Figure 4-2:** A system containing a lever and two pulleys.

10. Pulley 2 receives a pulling force from Rope 3 at Left and does nothing because it has no rule for this situation.
11. Lever 2 receives a pulling force from Rope 1 at Left and propagates it to Axis in a counterclockwise direction. This force has now been torqued by Lever 2.
12. Axle 1 receives a counterclockwise torque from Lever 2 at End and propagates it to Start.
13. Lever 1 receives a counterclockwise torque from Axle 1 at Axis and propagates it to Left as a pushing force. This force has now been torqued by Lever 2 and untorqued by Lever 1.
14. Input receives a pushing force from Lever 1 at End and returns an equal, opposite pushing force.
15. Lever 1 receives a pushing force from Input at Left and propagates it to Axis in a clockwise direction. This force has now been torqued by Lever 1.
16. Axle 1 receives a clockwise torque from Lever 1 at Start and propagates it to End.
17. Lever 2 receives a clockwise torque from Weight 1 at Axis and propagates it to Left as a pulling force.
18. Rope 1 receives a pulling force from Lever 2 at End and propagates it to Start.
19. Pulley 1 receives a pulling force from Lever 2 at Left and propagates it to Right.
20. Rope 2 receives a pulling force from Pulley 1 at Start and propagates it to End.
21. Pulley 2 receives pulling forces at Left and Right from Rope 3 and Rope 2, respectively, and propagates a pulling force of double value to Top.
22. Rope 4 receives a pulling force from Pulley 2 at Start and propagates it to End.
23. Weight 1 receives a pulling force from Rope 4 and has nowhere to propagate it.

Then, the motion analysis:

1. Input releases a motion-inducing impulse.
2. Lever 1 receives a motion-inducing impulse from Input at Left and propagates it to Axis.
3. Axle 1 receives a motion-inducing impulse from Lever 1 at Start and propagates it to End.
4. Lever 2 receives a motion-inducing impulse from Axle 1 at End and propagates it to Left.
5. Rope 1 receives a motion-inducing impulse from Lever 2 at End and propagates it to Start.
6. Pulley 1 receives a motion-inducing impulse from Rope 1 at Left and propagates it to Right.
7. Rope 2 receives a motion-inducing impulse from Pulley 1 at Start and propagates it to End.
8. Pulley 2 receives a motion-inducing impulse from Rope 2 at Right and propagates it to Left.
9. Rope 3 receives a motion-inducing impulse from Pulley 2 at End and propagates it to Start.
10. Anchor 1 receives a motion-inducing impulse from Rope 3 and returns a reaction impulse.

11. Rope 3 receives a reaction impulse from Anchor 1 at Start and propagates it to End.
12. Pulley 2 receives a reaction impulse from Rope 3 at Left and is still receiving a motion-inducing impulse from Rope 2 at Right. It sends a motion-inducing impulse out from Top.
13. Rope 4 receives a motion-inducing impulse from Pulley 2 at Start and propagates it to End.
14. Weight 1 receives a motion-inducing impulse from Rope 4 and moves in the direction of the force acting on it from Rope 4 (towards Rope 4).
15. Rope 4 receives a message that Weight 1 has moved, so it moves toward Start.
16. Pulley 2 receives a message that Rope 4 has moved, so it turns clockwise and moves away from the rope (up).
17. Rope 3 receives a message that Pulley 2 has moved, so it does nothing because it is being pulled by a reaction impulse, not a motion-inducing impulse.
18. Rope 2 receives a message that Pulley 2 has turned, so it moves towards Start.
19. Pulley 1 receives a message that Rope 2 has moved, so it turns counterclockwise.
20. Rope 1 receives a message that Pulley 1 has turned, so it moves towards Start.
21. Lever 2 receives a message that Rope 1 has moved, so it turns counterclockwise.
22. Axle 1 receives a message that Lever 2 has turned, so it turns counterclockwise.
23. Lever 1 receives a message that Axle 1 has turned, so it turns counterclockwise.
24. Input receives a message that Lever 1 has turned, so the program ends.

ARCHIMEDES performs the same inferences as in the previous examples to establish that the input force on Lever 1 directly causes the motion of Weight 1. However, the program cannot determine whether the overall effect is to increase or decrease the force necessary to raise the weight. The existence of Pulley 2 caused the force necessary to raise Weight 1 to be halved, but the torque combination of Lever 1 and Lever 2 increased it. Therefore, the program can make no judgment on the overall effect of the system beyond the simple motions of components.

# Chapter 5

## Implementation

ARCHIMEDES was written in Java v.1.3. This section details the program's data structures and program flow and explains the decisions that were made in the implementation phase.

### 5.1 Objects

ARCHIMEDES uses several different Java classes to implement its model of a mechanical system and its components. The most important of those classes and interfaces are elaborated upon here.

#### 5.11 Components

Each type of mechanical component is represented by its own class. This class contains the logic by which the component outputs forces and motions based on its input of forces

and motions coming from other components. Each has at least one connection point, where the input and output takes place.

For example, an axle is defined to have two connection points at which gears or other components can be attached. One of the rules for an axle is, “If something is turning one end of the axle, then propagate that torque to whatever is attached to the other end.”

## **5.12 Connection points**

The same class represents all different kinds of connection points on different kinds of devices. The class keeps track of the following information:

- Its name.
- The parent component that this connection point is a part of.
- The relationship that exists at the connection point. This is the specific type of connection that exists between this component and another, such as “hangs from” or “wrapped around clockwise”.
- The point on another component that this point is connected to.
- The input action on this point, if one exists.
- The output action that is coming from this point, if one exists.

## **5.13 The device interface**

The Device interface used in ARCHIMEDES is the means by which different mechanical components in a system communicate, sending information to each other about forces

and motion (actions). All the component classes implement this interface. Using it allows the program to find out information about a component without regard to what particular kind of component it is.

The Device interface supports methods that do the following:

- Get the component's type (gear, pulley, etc.).
- Get the component's name.
- Return one of the component's connection points, given that point's name.
- Get the relationship that exists at one of the component's connection points, given the point's name.
- Tell the device that it is receiving a new action.
- Tell the device to apply its internal rules based on whatever actions it is currently receiving and return any actions that it is now outputting.

## 5.14 Actions

A single class stores information about all types of actions (forces and motion impulses) that are propagated around the system. This class keeps track of the following information:

- The connection point on a component that this action is coming from.
- The connection point that it is acting upon.
- What type of action it is – an action force, a reaction force, a motion impulse, etc.
- The action's "value." This is a measure of the strength of a force. It comes from the component that is the original source of the force; in ARCHIMEDES, this is usually an arbitrarily heavy weight. This value is only used when comparing one

force against another, when the value of both is known exactly; for example, when examining two differently weighted objects hanging from either side of a pulley.

- Whether or not the value is known exactly. If two or more forces are combined, or if a force is transformed in any way, then its value is no longer exactly known, and other information must be used to compare forces instead.
- The direction of the action. This is either a push or a pull, or a clockwise or counterclockwise turn.
- All the devices that have transformed this action and in what way. For example, a straight-line force may have been transformed into an angular torque by a gear.

Keeping track of an action's "lineage" in this way allows for more detailed qualitative comparison of forces, such as that seen in the gear system example in Chapter 4.

## **5.2 Program flow**

The data structure that controls the program flow of ARCHIMEDES is a queue. When actions are generated, the components that are receiving the actions are placed at the end of the queue, if they are not already in the queue. Then, the component at the front of the queue is removed, and runs its logic to see whether it outputs any actions given its inputs.

Sometimes a component does not have enough information to know what to do – it needs another input. Using the queue solves this problem without the need for a special case. When the component reaches the front, it just doesn't output any actions to other

components. Then, when another component generates the action that the first one needs, the first component gets added to the end of the queue and will come through again.

Having a single thread also reduces the complexity and chance of error that would be present in a multi-threaded system. Many mechanical systems contain loops and complex branching in their topology. While a branching, multi-threaded program flow for ARCHIMEDES is certainly possible, and may be more analogous to the way that humans examine mechanical systems, its added complexity is not necessary.

# Chapter 6

## Related Work

Qualitative physical reasoning has grown to become a large area of research in artificial intelligence. This chapter describes a small selection of works in the field, and their relationship and contribution to ARCHIMEDES.

### 6.1 Mental simulation

In order to develop effective qualitative simulation techniques, several researchers have examined how people mentally simulate mechanical systems. Hegarty performed a pair of experiments that observed subjects' accuracy, response latency and eye fixations as they examined diagrams of pulley systems in which one of the components (a rope or a weight) was given to be in motion. The subjects were asked to describe the motion of another component in the system.

The subjects' dominant mental animation strategy was first to look at the whole configuration of a system, and then to trace a causal chain from the component in motion to the component in question. Subjects also made inferences against the direction of

causality, but this took more time and effort. Hegarty concluded that people reason about these types of systems incrementally because their knowledge is limited to component-level behavior.

Naranayan, Suda, and Motoya performed a similar experiment in which subjects examined piston diagrams. Their results supported Hegarty's conclusions.

ARCHIMEDES uses this approach, examining systems incrementally at the component level and following a causal rule-chaining system. However, when it reaches a component whose behavior is uncertain, it examines other components in order to eliminate the ambiguity before proceeding.

## **6.2 Shape and space-based kinematic reasoning**

A number of researchers have approached the qualitative physics problem based on the geometries and positions of the components in a mechanical system. One approach focuses on kinematic analysis, examining the motion of components in the system without explicitly considering the forces that cause the motion. Joskowicz presents an algorithm for fixed-axis systems that takes the position and geometrical description of each component and finds the possible relative motion of each pair of components that are in contact with each other. Then, using a constraint propagation technique, the algorithm computes the actual motion of each component, given an initial input motion on some component.

Gelsey used a similar pair-based algorithm to produce kinematic analyses of a piston/crankshaft mechanism and a gear differential. Both Gelsey and Joskowicz use

Reuleaux's vocabulary of kinematic pairs, which classifies the possible ways two moving parts can come into contact.

While ARCHIMEDES explicitly does not make inferences based on components' shape or absolute position, it does use the idea of a vocabulary of "connection types" that describe the relationship between two connected components.

### **6.3 Qualitative dynamic reasoning**

In contrast to kinematics, qualitative dynamics does include the explicit representation of the forces acting on components in a system.

Stahovich, et al describe a qualitative simulation program for rigid-body systems which abstracts a diagram into a qualitative configuration space that captures the spatial constraints between components in the system. They also provide a new, qualitative representation for forces in order to reduce ambiguity when simulating the behavior of these systems. The force representation consists of four properties: direction, magnitude, type, and interface motion, all of which are expressed either in qualitative values (+, -, or 0) or as belonging to one of a small number of well-defined categories.

ARCHIMEDES uses a similar representation for forces in order to examine and describe system behavior on a purely qualitative level, without a coordinate system or vector arithmetic.

# Chapter 7

## Conclusions and Future Directions

ARCHIMEDES demonstrates the viability of a topology-based qualitative mechanical simulator. It can successfully predict the behavior of systems made from the components it understands, as well as analogous systems that can be simulated using those components. For example, the component shown in Figure 15 could be simulated as a gear, axle and lever segment.

The principal advantage ARCHIMEDES has over other qualitative mechanical simulators is its high tolerance for ambiguity in the description of the system it receives. The exact sizes, shapes, positions and orientations of components need not be specified in order to predict a system's behavior, just as is the case with a person reasoning about a basic diagram.

To widen the program's range, several steps could be taken in the future. The first and simplest would be to add more devices and connection relationships to the current system. Another would be to allow for changes in system topology caused by motion, for example, a lever swinging down and locking into place or a weight falling through the air and landing on a platform.

# References

- Forbus, Kenneth D. “Qualitative Physics: Past, Present, and Future.” *Readings in Qualitative Reasoning about Physical Systems*. Ed. Daniel Weld and Johan de Kleer. (San Mateo: Morgan Kaufman Publishers, 1990).
- Forbus, Kenneth D. “Qualitative Spatial Reasoning Framework and Frontiers.” *Diagrammatic Reasoning: Cognitive and Computational Perspectives*. Ed. Janice Glasgow, N. Hari Narayanan, and B. Chandrasekaran. (Menlo Park: The AAAI Press, 1995).
- Funt, Brian V. “Problem-Solving with Diagrammatic Representations.” *Diagrammatic Reasoning: Cognitive and Computational Perspectives*. Ed. Janice Glasgow, N. Hari Narayanan, and B. Chandrasekaran. (Menlo Park: The AAAI Press, 1995).
- Hegarty, Mary. “Mental Animation.” *Diagrammatic Reasoning: Cognitive and Computational Perspectives*. Ed. Janice Glasgow, N. Hari Narayanan, and B. Chandrasekaran. (Menlo Park: The AAAI Press, 1995).

Narayanan, N. Hari, Masaki Suwa and Hiroshi Motoda. "Hypothesizing Behaviors from Device Diagrams." *Diagrammatic Reasoning: Cognitive and Computational Perspectives*. Ed. Janice Glasgow, N. Hari Narayanan, and B. Chandrasekaran. (Menlo Park: The AAAI Press, 1995).

Rieger, Chuck and Milt Grinberg. "The Declarative Representation and Procedural Simulation of Causality in Physical Mechanisms." *Readings in Qualitative Reasoning about Physical Systems*. Ed. Daniel Weld and Johan de Kleer. San Mateo: Morgan Kaufman Publishers, 1990.

Stahovich, Thomas F., Randall Davis and Howard Shrobe. "Qualitative Rigid Body Mechanics." To be published in 2000.