Explanation of Mechanical Systems Through Qualitative Simulation

Stuart Laughton

AITR85-19 Dec 1985

This research was sponsored by the US Army Research Office under contract DAAG20-84-K-0060. Computer equipment used in this research was furnished by a grant from the Xerox Corporation.
# TABLE OF CONTENTS

Chapter 1: Introduction ........................................... 1

Chapter 2: An Overview of Qualitative Reasoning ................... 3
  2.1 Quantitative vs. Qualitative Simulation .......................... 3
  2.2 Simulation vs. Envisionment ..................................... 4
  2.3 Quantities and Quantity Spaces .................................. 4
  2.4 Time .......................................................... 6
  2.5 Qualitative Relations ............................................. 7
  2.6 Prediction and propagation of qualitative values ............... 10
  2.7 Ambiguity .................................................... 14

Chapter 3: Related Work ............................................ 17
  3.1 The Naive Physics Manifesto ...................................... 17
  3.2 Device Centered Perspective: DeKleer and Brown ............... 19
  3.3 Process Centered Perspective: Forbus ............................ 22
  3.4 A Neutral Perspective: Kuipers ................................. 28
  3.5 The Steamer Project ............................................. 30

Chapter 4: Mechanical Device Simulation ............................. 32
  4.1 Implementation of the Device Simulation System ............... 32
  4.2 The QSIM Based Simulation Engine .............................. 32
  4.3 DSS Input and Output ......................................... 42
  4.4 The Special Status of Objects in DSS .......................... 46

Chapter 5: Modelling a Pressure Activated Valve .................... 48
  5.1 First Valve Model ............................................. 48
  5.2 Second Valve Model ............................................ 51
  5.3 Third Valve Model ............................................. 53

Chapter 6: A Piston Pump Model ..................................... 61
  6.1 The Slider and Crank Subsystem ................................ 61
  6.2 Piston and Cylinder Subsystem ................................ 61
  6.3 Valves and Flows .............................................. 64
  6.4 Additional Constraints ....................................... 66

Chapter 7: Animation and Explanation ................................ 67
  7.1 Animation ................................................... 67
  7.2 Narration ..................................................... 73
  7.3 Question Answering ........................................... 75
    7.3.1 Questions about Structure ............................... 75
7.3.2 Questions about Behavior ........................................... 76
7.3.3 "What if?" Questions ................................................ 81

Chapter 8: Conclusions and Future Work ......................... 83

Appendix: Piston Pump Model and Simulation Output ........... 85
A.1 Piston Pump Model Variables ........................................ 85
A.2 Piston Pump Model Constraints .................................... 86
A.3 Selected Piston Pump Behavior Path ............................. 88
A.4 Ambiguity in the Piston Pump Simulation ...................... 96

Bibliography ................................................................. 97
Chapter 1: Introduction

Representing and reasoning about the physical world in a qualitative or common sense manner has recently become an area of much interest in artificial intelligence. This thesis is an experiment testing the feasibility of using qualitative simulation as the foundation of a computerized system that is capable of displaying and explaining the function of simple mechanical devices.

The target application of this research was inspired by the format of the book The Way Things Work: An Illustrated Encyclopedia of Technology (1). This volume contains a collection of short descriptions of various mechanical devices. These descriptions are not intended to provide technical reference but instead, in the words of the publishers, "...to give the layman an understanding of how things work." For each device there is a page of text opposite a page of diagrams. The simple diagrams and brief text in each of these descriptions work together to minimize the effort that the reader must invest in order to acquire a casual understanding of the device. Thus, the book is entertaining as well as informative and it is open it to a non-technical audience. It is the author's belief that these features may be amplified by transferring this format to a computer based system.

Existing technologies of natural language understanding and generation, including question answering, suggest obvious advantages offered by a computer implementation of the text component of device descriptions like those found in The Way Things Work. Likewise, bit mapped graphics and interface devices such as the "mouse" will allow the user to interact with the diagram rather than remaining a passive observer. Zooming in on components and soliciting textual information by pointing are examples of the possibilities. Furthermore, such a system can move beyond explanation and present a simulation of the device behavior. This simulation may be reflected to the user as animated graphics, accompanied by English narration.

The use of qualitative simulation in this project was suggested by a recent interdisciplinary research effort investigating the use of mental models in human reasoning (2). This topic, which is relevant to the cognitive science, artificial intelligence and education communities, investigates the hypothesis that much of human reasoning about the physical world involves constructing and manipulating mental models. This reasoning style may be contrasted with the alternative technique of reasoning by way of inference rules ( as in production systems ). The prin-
principles of qualitative simulation provide a formalism in which a some types of mental model reasoning may be represented.

The hypothesis of this thesis was that qualitative simulation may be utilized as the basis of a computerized explanation system. The merit of the qualitative simulation formalism was judged by its adequacy for simulating device behavior and the ease with which both graphic and textual explanation could be generated. To test this hypothesis an experiment was conducted in which a prototype simulation system was constructed and a specific device, a piston driven pump, was modelled. Although several problems were encountered in this process, a satisfactory simulation was accomplished and rudimentary techniques of animation and explanation were developed. The basic conclusion drawn from this experiment was that qualitative simulation is an appropriate knowledge representation for this type of application but that several difficult problems have yet to be solved before real-world implementations will be possible.

The organization of this thesis is as follows. Chapter 2 presents an overview of the concepts of qualitative reasoning that are common to most of the current theories in this field. Chapter 3 briefly describes the particulars of several of these theories. Chapter 4 discusses the current implementation of the mechanical device simulation system. Chapter 5 discusses the iterative refinement of a valve sub-device model. Chapter 6 presents the complete model for the piston pump device. Chapter 7 discusses the design of the natural language and animation based user interface for this system. Finally, chapter 8 discusses unsolved problems, possible enhancements and the conclusions of the experiment.
Chapter 2: An Overview of Qualitative Reasoning

This chapter presents an overview of the basic concepts of qualitative reasoning and simulation. These basic concepts are for the most part common ground for the related work discussed in chapter 3.

2.1 Quantitative vs. Qualitative Simulation

Dynamic systems may be modelled quantitatively by a set of state variables (the essential parameters of the system) and constraints between them in the form of differential equations. Numerical or analytic solution methods then yield the behavior of the model as real valued time varying functions. The states of the system and their relationships to one another are embedded within these mathematical entities. This technique is often able to produce quite accurate simulations and serves well to describe what the behavior of a system is. However, when the goal is to explain how behavior comes about, and numerical precision is not important, a more qualitative approach is warranted.

Unlike quantitative simulation, qualitative methods do not assign numerical values to the state variables in the behavioral description of a system. The weak nature of state variable values in the qualitative domain means that a qualitative description of behavior will apply to a class of systems rather than a particular system instance. Similarly, because precise numeric values are not required as input to describe the structure of the system, qualitative reasoning can proceed with limited knowledge. However, an unavoidable consequence of the fuzzy nature of qualitative values is the introduction of ambiguity into the reasoning process. This ambiguity causes qualitative simulation to become at times a non-deterministic process producing multiple behaviors for a single model. Particular behaviors may correspond to different instances within a class of systems. In some cases the modelled behavior may not be realizable at all. An example of such a case is given by Kuipers (3 p.39).

Although there are qualitative analogs of differential equations involved in some qualitative reasoning theories, the "solution" methods are quite different from those of the quantitative domain. Qualitative simulation proceeds through local propagation of effect and its output explicitly represents a set of discrete system states and the transitions between them. Output of this form leaves causality more apparent than that of quantitative modelling.
2.2 Simulation vs. Envisionment

The details of qualitative reasoning differ greatly among the various theories. These differences will be discussed in chapter 3. However, almost all of them are based on either simulation or a process called envisionment. Although the distinction between these terms is rather blurred in the literature, the following definitions are used in this paper. Simulation is the process of running a model. Starting with a given initial state, successive states are generated locally and in temporal order. If ambiguity arises while generating a successor state it may be resolved either heuristically, arbitrarily, or with the aid of an external knowledge source (perhaps the user). Alternatively, the simulation might search more than one of the multiple behavior paths produced by ambiguity. The exploration of a path terminates when a cycle is discovered or a quiescent state is reached. In contrast, in envisionment the generation of system states needs not proceed in temporal order and the transitions between states need not be determined at the same time that the states are generated.¹ The principles of qualitative reasoning are discussed in this chapter primarily using the simulation paradigm. Furthermore, the work discussed in chapters 4 through 7 is based on a simulation algorithm.

2.3 Quantities and Quantity Spaces

The state variables of a real system can be modelled quantitatively as a set of real valued functions of time. In the qualitative domain all except a set of landmark values in the range of each function are mapped into open intervals which contain them. The boundaries of these intervals are defined by the landmark values that are retained. Therefore, the range of each qualitative state variable becomes a discrete finite set of intervals and their boundary values, see figure 2.1. Boundary values may be numeric, such as 0, or symbolic, such as TOP-OF-CONTAINER-2. As an example consider a state variable representing the temperature of a liquid. In the quantitative domain the range of values for this state variable might be the set of real numbers. However, in the qualitative domain this range might be abstracted to a discrete set

¹ Note that by this definition simulation is a special case of envisionment.
Figure 2.1

Real Numbers

* *
0 5 10 15 20 25 30 35

-∞ (-∞, 0) 0 (0, 32) 32 (32, +∞) +∞

Qualitative Values

This figure illustrates a mapping from the infinite continuous space of real numbers to a discrete qualitative quantity space with 7 members. In this example, the qualitative space includes the two "landmark" values 0 and 32.
consisting of the boiling point B, the freezing point F and the intervals (MINF,F), (F,B) and (B,INF).²

Information is necessarily lost in this mapping of quantity values from a real to a discrete space. However, the boundary values are chosen such that the information essential to describing the qualitative behavior of the system is retained. Obviously, this choice depends on the level of abstraction of the analysis as well as the specific system being modelled. An example of a quantity space where the abstraction is extreme consists of the single boundary point 0 separating the intervals (MINF,0) and (0,INF). Many interesting predictions can be made for several systems using only this simple representation (4,5).

The quantity space associated with a state variable may be unique to it or may be shared with other variables measured in the same units. In the latter case, these variables may be compared using qualitative analogs of the relations >, =, and < (see qualitative arithmetic below). Forbus (6) utilizes partial orderings among constant quantities and state variable values.

The derivatives of the system's state variables are the impetus for change when running a simulation. Some current theories include the entire derivative, Forbus (6), or the sign of the derivative, Kuipers (3), directly in the representation of the quantity. The role of the derivative in prediction is discussed below.

2.4 Time

Mapping the representation of time from the quantitative to the qualitative domain is related to the way quantities are mapped between these domains, as just described. The interest in simulating the behavior of dynamic systems is in how the state variables of the system change with time. Since the value of each of these quantities varies over a discrete set, the simulation is driven by discrete events. Each event consists of one or more quantities changing value. Therefore, the units of

² Throughout this paper, the symbols MINF and INF will be used to represent negative and positive infinity respectively.
time used in qualitative simulation need be of no finer granularity than is necessary to order these events.\(^3\)

An adequate representation of time is as a series of distinguished instants and the time intervals between them.\(^4\) Each of the qualitative states comprising the output of a simulation corresponds to either an instant or an interval of time. An instant corresponds to a state in the simulation where at least one variable changes qualitative value in one or both of the following ways:

- The value of the quantity is an interval in the preceding state (a time interval) and it reaches a boundary value in the present state (the instant).

- The value of the quantity is a boundary value in the present state (the instant) and moves into an interval in the following state (a time interval).

A time interval corresponds to a state in the simulation where all quantities remain qualitatively constant (i.e., their qualitative values remain in an interval or at a boundary value). Note that if a quantity has a boundary point as its value in a state representing a time interval then it does also at the states representing the preceding and following instants.

### 2.5 Qualitative Relations

In a qualitative simulation the characteristics of a specific system are represented by a set of qualitative relationships between its state variables. The organization of these relations as a description of the system's structure varies among qualitative theorists (see chapter 3). However, the types of relations utilized by each are quite similar.

An analog to ordinary arithmetic is defined by qualitative multiplication and addition relations as follows. Assume that A, B and C repre-

---

\(^3\) A total ordering over events may not be required, see Williams (5 p.342).

\(^4\) Forbus (6 p.40) allows consecutive instants.
Basic Qualitative Arithmetic

In the tables above + indicates a quantity value in the interval \((0, +\infty)\) and - indicates a value in the interval \((-\infty, 0)\). The symbol ? indicates ambiguity.
sent three qualitative state variables of a modelled system and that the structure of that system defines the relation \( C = A + B \) ( \( + \) indicates qualitative addition). Figure 2.2a illustrates the possible values of \( C \) given values for \( A \) and \( B \) assuming that they all exist in a quantity space with a single boundary point at zero. The symbol  in the upper right and lower left cells of the table indicate ambiguity. In these situations the value of \( C \) cannot be determined because of a lack of knowledge about the relative magnitudes of \( A \) and \( B \). Using the same three qualitative variables and the relation \( C = A \cdot B \) ( \( \cdot \) indicates qualitative multiplication) Figure 2.2b illustrates the possible values for \( C \) given \( A \) and \( B \).

The above discussion is concerned with how the values of quantities are constrained by qualitative arithmetic relations. However, as indicated by analogy to the infinitesimal calculus the qualitative derivatives of the quantities are also constrained. For example, if the relation \( c = a \cdot b \) holds for the quantitative model of some system then we know from the differential calculus that \( c' = a' \cdot b + b' \cdot a \) also holds. Thus, moving to the qualitative domain, we know that \( C = A \cdot B \) and that \( C' = A' \cdot B + B' \cdot A \). Hypothetically, if \( A = (0, \infty) \) and \( \text{sign}(A') = + \) and if \( B = (\text{MINF}, 0) \) and \( \text{sign}(B') = - \) then the above relation dictates that \( \text{sign}(C') = - \). Notice however, that for some combinations of \( A, B \) and their derivatives \( \text{sign}(C') \) is ambiguous.

Often in the model of a system one state variable will represent the derivative of another. A common example of this situation is found in the linear motion of an object through space. In this case the velocity of the object is the derivative of its position. This qualitative derivative relationship is fundamental in predicting the future values of qualitative variables and will be discussed in the next section.

The final basic type of qualitative relation is that of qualitative proportionality. If two qualitative variables are specified as directly proportional (monotonic) to each other then the signs of their derivatives must match. On the other hand, if they are specified as inversely proportional (inversely monotonic) then either both derivatives are zero or the signs of their derivatives are opposite.

In addition to monotonicity, a set of correspondences may be specified between landmark values for two variables. For example suppose that two variables, \( A \) and \( B \), are monotonic and that a correspondence is specified for their respective landmarks, \( A^* \) and \( B^* \). This implies that in

\[ f'(F') \] represents the first quantitative (qualitative) partial derivative of \( f(F) \) with respect to time.
Figure 2.3

X = M+(Y)  Correspond (X1, Y1) (X2, Y3)

This figure illustrates to views of hypothetical curves which are compatible with the above monotonicity relationship. (The curves in the two plots are not necessarily compatible with each other.)
any valid state of the system \( A = A^* \) if and only if \( B = B^* \). Furthermore, 
\( A < A^* \) iff \( B < B^* \) and \( A > A^* \) iff \( B > B^* \). See figure 2.3.

It is often the case that qualitative relationships hold only for a subset of the possible states of the modelled system. As will be shown in chapter 3 these subsets may be viewed in terms of operating regions for the component devices of the system and/or in terms of the status of the constituent processes of the system. In either case they may be defined in terms of the values of qualitative state variables. For this reason, most qualitative reasoning systems provide a mechanism to validate and devalidate qualitative relations based on the system state.

The role played by the qualitative relations in the simulation process is to define which possible combinations of values for the state variables represent valid states of the modelled system. In this light the relations are viewed as constraints and the simulation process partly as a constraint satisfaction algorithm. As we shall see, this is indeed a significant component of the existing theories. However, equally important is the manner in which the simulator predicts the transitions of the system between these valid states.

2.6 Prediction and propagation of qualitative values

There are two related subproblems involved in determining the behavior of a system from its qualitative model. Both of these must be addressed whether the algorithm used is an instance of simulation or an instance of envisionment. One determines which transitions from one value to another are legal for each state variable. The other is a constraint satisfaction problem: given sets of possible values for some or all of the parameters describing a system, determine all assignments of values to parameters such that the qualitative relations of the system are satisfied. In the following discussion these tasks and the relationship between them are discussed in the context of a simulation based algorithm. An envisionment algorithm is discussed in the description of DeKleer and Brown's work in chapter 3.

The input to a qualitative simulation includes a specified initial state. The first step in determining the successor state(s) of a given state is to determine the set of possible next state values for each state variable. This determination is local to each variable (and the sign of its qualitative derivative) and is independent of the qualitative relationships between variables.

For now it is assumed that the real valued functions which describe the physical system and their corresponding qualitative state
variables all vary continuously with time (in chapter 5 some disadvantages of strict continuity and an adjunctive approach are discussed). This assumption teamed with qualitative interpretations of the intermediate value and mean value theorems of the infinitesimal calculus impose constraints on how state variables may change value.

Recall that the quantity space of a qualitative variable is composed of an alternating sequence of boundary points and open intervals. Therefore, a qualitative version of the intermediate value theorem stipulates that the only possible next state values for a variable whose current value is:

- an open interval is either one of the two boundary points defining the interval or that interval again.
- a boundary point is either one of the two neighboring open intervals or that boundary point again.

A qualitative interpretation of the mean value theorem together with the continuity restriction described above defines a relationship between the value of a variable and the sign of its derivative in two temporally adjacent states. This relationship is expressed in the following rules.

**Generalized Qualitative Integration Rules**

1. If a variable's value is a quantity interval and its qualitative derivative is positive (negative) over some time interval then at the end of that time interval it will have moved towards the upper (lower) boundary point of the quantity interval and will possibly have reached that point at the following instant. If no upper (lower) boundary point exists then the variable's value will have moved toward positive (negative) infinity.

2. If a variable's qualitative derivative is zero over some time interval then at the instant following that time interval the variable's value remains unchanged.

3. If a variable's value is a boundary point at some time instant and its qualitative derivative is positive (negative) over the following time interval then the variable's value over this time
interval will be the quantity interval above (below) the boundary point.

4. If a variable's value is a boundary point at some time instant and its qualitative derivative is zero over the following time interval then the variable's value remains at the boundary point over that time interval (and the following instant, by rule 2).

5. If a variable's value is a quantity interval at some time instant then it will remain in that quantity interval over the following time interval.

The first four rules are based on the "Qualitative Integration Rule" of Williams (5) but generalized to allow an arbitrary number of intervals and boundary points in quantity spaces. Rules 3 and 5 state that transitions to a boundary point require some interval of time while a departure from a boundary value may occur instantaneously. Kuipers (3) presents a formal derivation of a set of transition rules that include these concepts and in addition allow the discovery of new landmark values in quantity spaces. This set of rules is divided into transitions from time instants and transitions from time intervals. Kuipers QSIM algorithm will be discussed in detail in chapter 4.

The effects of higher order derivatives propagate through the qualitative derivative relationships under control of the rules of transition. To illustrate, consider the position (X), velocity (V) and acceleration (A) of a particle moving through space. Let X = 0, V = 0, A = 0, and \( \text{sign}(dA/dt) = + \) at some time instant t1. We know the following relationships:

\[
\frac{dX}{dt} = V \quad \text{and} \quad \frac{dV}{dt} = A
\]

Rule 5 tells us that \( \text{sign}(dA/dt) = + \) in the following time interval \((t1,t2)\). Therefore, rule 3 says that \( A = (0, \text{inf}) \) over \((t1,t2)\). Because \( A = dV/dt \) rule 3 also says that \( V = (0, \text{inf}) \) over \((t1,t2)\). Similarly, rule 3 implies \( X = (0, \text{inf}) \) over \((t1,t2)\).

Utilizing the integration rules just described a set of possible next state values may be determined for each state variable of the system. The qualitative relationships between variables are then used as a constraint network and all valid assignments of values to variables are determined through some constraint satisfaction algorithm. If there are

---

\(^6\) In some theories e.g. Williams (5) and DeKleer & Brown (4) the transition rules do not predict values for all variables. In this case
multiple valid assignments of values to the state variables then the simulation is faced with multiple behavior paths. This ambiguity in qualitative reasoning is discussed in the next section.

2.7 Ambiguity

The ambiguity in qualitative simulation originates in its weak representation of quantities. This defines an incomplete ordering among the values and the magnitudes of the derivatives of the state variables and consequently among the events in time. Ambiguity reveals itself as multiple transitions from one qualitative state to those succeeding it in time. In general there are three ways to deal with this phenomenon:

1. Prevent ambiguity at its roots i.e. move to a stronger more quantitative representation for quantity and/or the relations between them.

2. Use general device independent knowledge to choose among the multiple paths (qualitative physics intuition).

3. Explore all paths.

In order for more than a single state to be predicted as a successor to some state S1 there must be at least one state variable with multiple transitions from S1 that are consistent with the transitions for the other variables and the qualitative relations of the model. Reviewing the integration rules of the previous section two situations become evident where multiple transitions are predicted for a single variable.

1. At a time instant, a state variable has a boundary point as its value and a zero derivative. In this case the state variable may move into the upper quantity interval with a positive derivative, move into the lower quantity interval with a negative derivative, or remains constant in the following state (a time interval).

the constraint satisfaction algorithm also involves value propagation.
2. In a time interval a state variable has a quantity interval as its value and a non-zero derivative. Here the possible transitions, as stated in Integration Rule 1, are that the variable moves closer to the boundary point (upper or lower depending upon the sign of its derivative) but remains in the quantity interval or that the variable reaches the boundary point in the next time instant.

As an example of how multiple transitions for a single variable can be consistent with the qualitative relations consider three qualitative variables A, B and C and the the qualitative relationship \( C = A \pm B \). Now consider a time instant \( t_1 \) with \( A = B = C = 0 \), \( \text{sign}(dA/dt) = + \), \( \text{sign}(dB/dt) = - \) and \( dC/dt = 0 \). By integration rule 3, in the following time interval \( (t_1, t_2) \), \( A = (0, A^*) \) and \( B = (B^*, 0) \) where \( A^* \) is the least positive boundary value in the quantity space of \( A \) and \( B^* \) is the greatest negative boundary value in the quantity space of \( B \). However, integration rules 3 and 4 allow \( C \) to be any of \((C^-, 0), (0, C^+), \) or 0 in \((t_1, t_2)\) where \( C^- \) and \( C^+ \) are defined analogously to \( B^* \) and \( A^* \) respectively. Since in \((t_1, t_2)\) \( \text{sign}(A) = + \) and \( \text{sign}(B) = - \) the basic qualitative arithmetic cannot eliminate any of the three possible transitions for \( C \) based on the constraint \( C = A \pm B \).

A second example where ambiguity in the local transitions for state variables may not be resolved by the qualitative relations is as follows. Consider a situation where two valves of a physical system are both partially open but are closing during a time interval. If we let \( V_1 \) and \( V_2 \) be qualitative variables representing the openness of each valve we have \( V_1 = (0, V_1\text{max}) \), \( V_2 = (0, V_2\text{max}) \) and \( \text{sign}(dV_1/dt) = \text{sign}(dV_2/dt) = - \). Integration rule 1 predicts two possible transitions for each valve in the following time instant: \( V_1 \) either 0 or \((0, V_1\text{max})\) and for \( V_2 \) either 0 or \((0, V_2\text{max})\). In a likely physical situation the variables \( V_1 \) and \( V_2 \) may be completely unrelated except that the derivatives of both depend upon a common pressure variable (in chapter 5 a system with a similar configuration is described). In such a case the qualitative relations will not eliminate any of these transitions and the result is four possible next states defined by the cross product of the two sets of possible next state values. In English these states correspond to both valves closing completely at the same time, neither valve closing completely or one valve closing but not the other.

A possible method to prevent both of the instances of ambiguity above is to represent the relationships among the magnitudes of the qualitative variables involved. In the example with the relation \( C = A \pm B \) if it is known that \( \text{abs}(A) > \text{abs}(B) \) then all the possible transitions for \( C \) must be eliminated except \((0, C^+)*\). In the second example the relative
magnitudes of V1 and V2 (distance) as well as relation between the magnitudes of dV1/dt and dV2/dt (speed) would be required to determine which valve would close first.

In general it would be difficult to maintain knowledge of the relationships between the magnitudes of all values at all times without moving back to a single quantitative quantity space. However, Forbus (6) utilizes knowledge of relative magnitudes between quantities to resolve some cases of ambiguity in qualitative addition.

A second way to reduce ambiguity, as mentioned at the beginning of this section, is to allow multiple next states to be generated and then eliminate some of them based on some criteria. Kuipers (3) uses the term global filter to describe a set of such criteria. These rules include pruning successor states which duplicate previously generated states\(^7\) and pruning states which allow a variable to transition to positive or negative infinity. DeKleer and Brown (4) use external knowledge of the function of the device being modelled to introduce assumptions into the simulation process that resolve some ambiguities. Another source of external knowledge might be to present the alternatives offered by ambiguity to the user of the simulation system and allow her/him to decide.

Finally, if ambiguity prevails and the simulation is left with a branch in the transitions from some state a search strategy may be employed to expand more than one branch. Happily it is sometimes the case, as perhaps in the valve example above, that it does not matter which branch is explored. In this case whether either valve closes first and the other soon afterwards or if they both close at the same time the subsequent simulation will produce the same behavior. In such a situation the multiple branches produced by ambiguity would later merge in due course of the simulation. A breadth first search would resolve the ambiguity in this case.

---

\(^7\) Actually his system and the system used in the work described in chapter 4 allow a generated time interval state to duplicate the immediately preceding time instant state but not vice versa.
Chapter 3: Related Work

This chapter reviews the work of several groups in the area of qualitative reasoning. While each of these is specifically concerned with reasoning about physical phenomenon they vary in both their scope and perspective. By way of introduction, the "Naive Physics Manifesto" of P.J. Hayes (7) is discussed. Secondly, the device centered perspective of qualitative reasoning is presented. This point of view is taken in the theory of DeKleer and Brown (4). An alternative perspective proposed by Forbus (6) is then introduced in which processes rather than devices are the central elements of organization. Next, a formalism developed by Kuipers (3) is investigated in which the perspective is neutral with respect to the device/process dichotomy. Here I will demonstrate by way of example that a formalization from either the process or device centered schools may be represented in this neutral mode and also that a representation of a mechanism in the neutral formalism has both a device and process centered interpretation. Finally, the simulation based training system STEAMER, Hollins, Hutchins, Weitzman, (8), is described.

3.1 The Naive Physics Manifesto

In the introduction to his "Naive Physics Manifesto" P.J. Hayes emphasizes that his proposal is concerned with more than just a "toy problem", which he says Artificial Intelligence is full of. Indeed his proposal is to formalize a large part of common sense knowledge about the physical world in a relatively unconstrained environment. While conscientiously avoiding any implementation details, this proposal seeks to identify the essential implementation independent features that should be exhibited by a robust and thorough formalization of common sense physics.

As a metaphor for his discussion Hayes refers to a formalization of common sense knowledge as an axiom-concept graph (AC graph). The nodes of this graph are concepts of the domain and an arc between nodes represents their inclusion in an axiom of the domain. Using this model Hayes proceeds to identify the desirable attributes of a formalization.

First he points out that the ratio of axioms to concepts in the graph should be high (i.e. the graph should be highly connected). In order to preserve this density concepts should only be defined if they are generally useful i.e. they appear in many axioms. However, Hayes also suggests that there will be regions of higher and lower densities in the
graph. The higher density regions he refers to as clusters. Clusters correspond to the subareas of common sense physics. Hayes considers the identification and formalization of clusters to be an important methodology in constructing a naive physics. Examples of clusters given in the paper include measuring scales, energy and effort, forces and movement, etc. Forbus (6 p.16) claims that his Qualitative Process Theory is a cluster in this sense.

Two additional attributes Hayes considers important are fidelity and thoroughness. Hayes uses the concept of models of a first order axiomatization to define a high fidelity formalization as one for which its simplest model closely resembles the intended model. A formalization is said to be thorough if as new concepts are defined the majority of other concepts needed for the definition are already included in the formalization.

Having defined these criteria for a robust non-toy knowledge base of common sense physics, Hayes makes the following conclusions. First, he observes that in a dense formalization weak domain independent methods of inference will be computationally infeasible. Therefore, he claims, formalization of knowledge must precede formalization of inference methods as these methods must exploit particulars of the knowledge structure.

This discussion of the "Naive Physics Manifesto" is included mainly for contrast to the methodologies of common sense physics that follow. The work of Hayes predates the latest reports of the following theories by some five years. It is interesting that these theories are more alike each other than they are like Hayes' proposal. In particular, the QSIM algorithm of Kuipers (which is the basis of my own system (see chapter 4)) and much of the material in the preceding chapter) bears little resemblance to the knowledge intensive methodology advocated by Hayes. On the other hand the Qualitative Process Theory of Forbus seems to be the closest to the naive physics paradigm and Forbus does borrow Hayes' notion of history.

Histories are one of the clusters defined by Hayes. The concept of a history as a piece of space time was proposed as an alternative to the situational calculus where change is represented by a series of global state snapshots of the world. Situational calculus must deal with the frame problem. This may be done by supplying a frame axiom for each variable-action pair. These axioms specify whether or not the variable changes as a result of the action. The frame problem is handled more efficiently using histories because unlike a global state a history is bounded spatially as well as temporally. When working with histories change may be evolved independently for objects (or processes) that are separated from interaction either spatially (e.g. by a wall or by a great distance) as well as temporally.
Histories will be very important in a general theory of common sense physics because they allow reasoning about a sub-space of the world without regard to the infinite number of unrelated phenomenon that are happening at the same time. However, in the domain of mechanical devices explored in this research so far all the parameters representing a mechanism are interrelated either because they belong to the same object (or process) or because they are parameters of interacting objects (or processes). Therefore, a single history is evolved which is identical to a situation calculus representation with the exception that intervals of time are allowed to be states as well as instants. The spatial extent of this history is simply that of the mechanism as apart from the rest of the world.

3.2 Device Centered Perspective: DeKleer and Brown

As mentioned above the qualitative physics theory of DeKleer and Brown (4) is a device oriented one. Figure 3.1, from (4 p.10) diagrams their ENVISION algorithm for determining the behavior of a mechanism from its structure. ENVISION takes two inputs: the device topology and a library of predefined component models. The algorithm builds a composite device model using these inputs. Qualitative behavioral predictions and causal explanations are then produced from this model.

The structure of a mechanism is defined by a topology in terms of components, the connections between them called conduits, and the type of materials transported through conduits between the components. Examples of components include valves, capacitors, switches, etc. Conduits may represent pipes, wires, etc with corresponding materials fluids, electric currents, etc.

Each model in the component library specifies the laws of behavior for a class of component devices. A primary emphasis of this theory is to automate the modelling process, that is, to construct a composite model from component models, as well as to solve the resulting model (predict its behavior). With this goal in mind DeKleer and Brown dictate several maxims for defining component models. The first of these they

---

8 Williams (5 p.342) remarks that the concept of history allows a representation of time which imposes only a partial ordering on events. Thus under some circumstances the temporal relationship between events may be ignored.
The ENVISION Algorithm of DeKleer & Brown

ENVISION constructs a composite model from a library of component models and a device topology. It then solves this model producing a description and a causal explanation of system behavior.
call the no function in structure principle. This rules out any presumption of the function of a composite mechanism in the rules governing the behavior of its components. An example which violates this meta-rule, due to DeKleer and Brown (4 p.16), is a model of an electric switch which states simply that current flows iff the switch is closed. As they point out this may be false for two switches connected in series.

Two additional directives given for constructing component models are class-wide assumptions and locality. These ensure that the no function in structure principle is not violated. The first of these states that only assumptions applicable to the entire class of components may be included in a component's rules. The second allows a component model to reference only variables (quantities) that are either local to it or available through direct connections (conduits) to its immediate neighbors.

The component laws in this qualitative physics are called confluences. As an example, the confluences for a valve component might be:

\[
dP/dt + dA/dt = dQ/dt \quad \text{and} \quad P = Q
\]

where \( P \) is the pressure across the valve,
\( A \) is the area available for flow,
and \( Q \) is the rate of flow (all qualitative values).

DeKleer and Brown describe the first confluence as follows, "The confluence represents multiple competing tendencies: the change in area positively influences flow rate and negatively influences pressure, the change in pressure positively influences flow rate, etc." As the authors observe, this confluence is invalid when the valve is closed and stationary. In this case the above confluence implies that \( dA/dt = dQ/dt = Q = 0 \) and therefore that \( dP/dt = P = 0 \). However in reality the pressure should be unconstrained in this situation. This discrepancy is removed by introducing the notion of device state. With this addition the confluence for the valve can be stated as follows.

<table>
<thead>
<tr>
<th>STATE</th>
<th>CONFLUENCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A_{max} &gt; A &gt; 0:</td>
<td>( P = Q, \ dP/dt + dA/dt = dQ/dt )</td>
</tr>
<tr>
<td>A = A_{max}:</td>
<td>( P = 0, \ dP/dt = 0 )</td>
</tr>
<tr>
<td>A = 0:</td>
<td>( Q = 0, \ dQ/dt = 0 )</td>
</tr>
</tbody>
</table>

The concept of device state is fundamental to the part of the envisioning algorithm that predicts device behavior from a composite
mechanism model. This algorithm does not fit the definition of simulation given in chapter 2. It operates by first determining all valid assignments of qualitative values to each possible composite mechanism state. This is accomplished by a combination of constraint satisfaction and propagation. This set of possible states is determined as the cross product of the sets of component states for all components in the composite mechanism. For example, if the valve above (having three states) and a two state switch comprise the components of a composite mechanism there will be six possible states of that composite. The second stage of the algorithm then determines the possible transitions between these states based on the principles of qualitative continuity and integration as discussed in chapter 2.

3.3 Process Centered Perspective: Forbus

Qualitative Process Theory (QP theory) is an ontology for qualitative physics proposed by Forbus (thesis). This theory takes a process centered perspective. QP theory is based on a vocabulary of process models analogous to the library of component models in the theory of DeKleer and Brown. Typical process models include heat-flow, fluid-flow, motion, boiling etc. Simulation in QP theory operates by predicting the effect of active process instances on the parameters of objects and the status of other processes.

As an example the model for a heat-flow process is given in figure 3.2 (p.32). According to QP theory a process is a 5-tuple consisting of:

1. A set of Individuals (instances of objects) that are involved in the process. The quantities referenced in a process model are parameters of the individuals.

2. A set of preconditions on the individuals. These are necessary conditions for the process to be active. These preconditions are considered to be external to QP theory in the sense that their truth values must be supplied from some external source e.g. the user.

3. A set of quantity conditions. These are also necessary conditions for the process to be active. They are predicates and ine-
**Figure 3.2**

**Process:** heat-flow

**Individuals:**
- src an object, Has-Quantity(src, heat)
- dst an object, Has-Quantity(dst, heat)
- path a Heat-Path, Heat-Connection(path, src, dst)

**Preconditions:**
- Heat-Aligned(path)

**Quantity Conditions:**
- $A[temperature(src)] > A[temperature(dst)]$

**Relations:**
- Let flow-rate be a quantity
- $A[flow-rate] > ZERO$
- flow-rate $\propto (temperature(src) - temperature(dst))$

**Influences:**
- $I-(heat(src), \ A[flow-rate])$
- $I+(heat(dst), \ A[flow-rate])$

---

**A Process Model for Heat-Flow**

Individuals are the objects involved in the process. Preconditions and Quantity conditions specify the conditions under which the process is active. The notation $A[\text{quant}(\text{ob})]$ indicates the amount of quantity "quant" for object "ob". The symbol, $\propto$, indicates a positive qualitative proportionality relationship. $I+$ and $I-$ indicate positive and negative direct influence, respectively. The Relations and Influences provide the constraint for determining situation consistency and the tendency for change used in predicting next situations.
qualities involving quantities pertaining to the individuals. They also may include predicates involving the status of other processes. These conditions are considered internal to QP theory: determining their truth values is part of the QP reasoning program.

4. A set of qualitative relationships between individuals which are dictated when the process is active. These include qualitative proportionalities discussed below as well as Let statements which specify individuals whose existence is dependent on the status of the process. These relationships supply constraints to filter and propagate quantity values in a simulation.

5. A set of influences imposed on individuals by an active process. These influences are primary to "Limit Analysis" discussed below.

An individual view is a structure that describes the state of an object or a configuration of objects. An example of the individual view Contained-Stuff is shown in figure 3.3. The structure of an individual view model is like that of a process without the set of influences.

The sets of individuals, preconditions, and quantity conditions in a process or view instance are used to determine the status of that instance. If all the quantity and preconditions hold in a situation then the instance is said to be active. Otherwise, it is inactive. It will be shown in the next section that the influences (also called direct influences) of a process instance correspond to derivative relationships between quantities. The qualitative proportionalities are analogous to those described in chapter 2. However, in QP theory proportionalities (or indirect influences) work in only one direction. This direction reflects the causality in the relationship. In the example of figure 3.2 the qualitative proportionality

\[
\text{flow-rate is-proportional-to (temperature(src) - temperature(dst))}
\]

specifies that the difference in temperature causes heat to flow. Both direct and indirect influences on a specific quantity may exist in more than one process. Referring again to the example in figure 3.2, there could be another process, perhaps another instance of heat flow, that also directly influences the amount of heat in the source. These influ-
**Figure 3.3**

**Individual-View: **Contained-Stuff

**Individuals:**
- c a container
- s a substance

**Preconditions:**
- Contains-Substance(c,s)

**QuantityConditions:**
- A[amount-of-in(c,s)] > ZERO

**Relations:**
- There is p ∈ piece-of-stuff
- amount-of-in(c,s) = amount-of(p)
- s = made-of(p)
- inside(c) = location(p)

**A Model for Contained Stuff**

This model describes the conditions under which stuff may exist inside a container.
ences are combined using a qualitative arithmetic like that described in chapter 2.\(^9\)

Forbus asserts that QP theory is capable of several reasoning tasks. These include prediction, "postdiction", diagnosis, measurement interpretation etc. He outlines algorithms for some **basic deductions** which can be made within QP theory and then describes how these may be utilized in higher level programs to accomplish such reasoning tasks. These basic deductions are:

- **Elaboration of a situation:** This procedure determines the set of all possible view and process instances (given an initial set of individuals).

- **Determination of process structure:** A process structure in QP theory refers to the set of active process and view instances.

- **Influence resolution:** This algorithm determines the sign of the derivative (Ds value) for each quantity in the process structure. If a quantity is directly influenced, qualitative arithmetic is used to calculate its Ds value from all the direct influences. Qualitative proportionalities are then utilized to propagate influence to the other quantities.

- **Limit analysis:** This basic deduction is important in the simulation algorithm used in QP theory. Here the possible changes in a situation are predicted. Changes may be in the status of processes and/or the existence of individuals, both of which are based on changes in the quantity spaces.

Quantity spaces are represented in QP theory as sets of inequalities between quantity values.\(^{10}\) A **quantity hypothesis** (QH) is generated for each possible change in a single inequality relation. Determination of these possible changes is based on the signs of the derivatives of the involved quantities and on the continuity and inte-

---

\(^9\) Qualitative arithmetic in QP theory utilizes inequality information to resolve some cases of ambiguity in qualitative addition.

\(^{10}\) Quantity spaces also include "landmark" constant values.
gration constraints described in chapter 2. ( may need an example here )
The set of all the conjunctions ( the power set ) of these single QH's are then generated. Each of these conjunctive quantity hypotheses ( CQH's ) is then annotated with the changes in the process structure that it implies ( if any ).

Forbus' simulation algorithm proceeds, given an initial situation, by performing limit analysis and then generating a set of consistent next-situations for each CQH generated. The process of generating the next-situations is called temporal inheritance and is based on assumptions which, for a given CQH and previous situation, specify which features of that situation change and which do not. These assumptions are listed below.

- All static individuals in the previous situation exist in the new situation.
- Facts supporting preconditions in the previous situation remain true in the new situation.
- All changes in inequalities specified by the given CQH hold in the new state. All inequalities mentioned in other CQH's and not mentioned in the given CQH must remain the same in the new situation as in the previous one.
- When consistent, dynamic individuals that existed in the previous state remain in existence.
- When consistent, inequalities not mentioned in any CQH that held in the previous state remain true.

If any assumption introduces an inconsistency then the CQH is declared inconsistent.\textsuperscript{11} The result of this envisioning algorithm is a directed graph. The nodes of the graph are situations and the arcs represent transitions found through limit analysis and temporal inheritance.

\textsuperscript{11} QP is implemented in a truth maintenance system
3.4 A Neutral Perspective: Kuipers

The preceding two sections have described the process centered QP theory of Forbus and the device centered envisionment theory of DeKleer and Brown. These theories are duals of each other in the sense that information that is local in one (local to a process or device) is distributed in the other. According to DeKleer and Brown (9) both of these methodologies are attempts at a true qualitative physics. They assert that a theory of qualitative physics should include techniques for constructing a model of the world as well as being able to subsequently solve this model. The distinction here is analogous to that between physics and mathematics in the quantitative domain. The theory discussed in this section is neutral concerning the device vs process issue. Furthermore, it is not a true qualitative physics, in the sense above, rather it is a qualitative mathematics.

The QSIM (3), and ENV (10), programs of Kuipers both deduce the possible behavior(s) of a physical system from a description of its structure that is defined solely in terms of constraints between its state variables. The QSIM algorithm is used as the underlying inference engine in the present research and will be discussed in some detail in the next chapter. The purpose of this present section is to demonstrate that process and device based representations may be mapped into neutral representations based on constraints.

Such a mapping might provide QSIM, which is a relatively efficient and tractable algorithm, as a solution method (a qualitative mathematics) for a model constructed within a qualitative physics. However, the goals of the present work are primarily to simulate and explain the functioning of a device based on an existing model. The construction of the model may be carried out a priori, "by hand" and with these goals in mind. Therefore, the qualitative mathematics of QSIM provides an adequate simulation algorithm. On the other hand, a coherent explanation of a physical mechanism certainly must include both the concepts of device and of process. Therefore, it must be possible to generate device and process interpretations of both the qualitative model and its output.

Figure 3.4a shows a QP representation for the process fluid-flow. A corresponding QSIM like representation in terms of state variables and constraints is depicted in figure 3.4b.

The quantity condition of the QP model in figure 3.4a is implicit in the constraint based model. If this condition is false, and therefore the process instance is inactive, the variable Press-Rel must either be zero or negative in the constraint representation. If it is zero so must be the variable Flow-Rate, and therefore no flow occurs. If it is negative then the variable Flow-Rate is also negative and flow is from desti-
Figure 3.4a

Process: fluid-flow

Individuals:
src a contained-liquid
dst a contained-liquid
path a fluid-path, Fluid-Connected(src, dst, path)

Preconditions:
Aligned(path)

QuantityConditions:
A[pressure(src)] > A[pressure(dst)]

Relations:
Let flow-rate be a quantity
flow-rate \propto (A[pressure(src)] - A[pressure(dst)])

Influences:
\(1+(\text{amount-of(dst)}, A[\text{flow-rate}])\)
\(1-(\text{amount-of(src)}, A[\text{flow-rate}])\)

A process model of fluid flow

Figure 3.4b

State Variables:
Amount-src: amount of fluid at source
Amount-dst: amount of fluid at destination
Flow-rate: rate of flow of liquid from source to destination
Press-src: pressure at source
Press-dst: pressure at destination
Press-rel: relative pressure from source to destination

Constraints:
d\text{Amount-dst}/dt = \text{Flow-rate}
d\text{Amount-src}/dt = - \text{Flow-rate}
\text{Flow-rate} = \text{Press-rel}
\text{Press-rel} = \text{Press-src} - \text{Press-dst}
nation to source. In QP theory this would result in an instantiation of the fluid-flow model with the bindings of Src and Dst reversed.

Forbus defines the precondition, Aligned(path), as follows, "A fluid path is aligned if and only if it has no valves or every valve is open". Although preconditions are defined to be outside of QP theory (i.e. they are not normally utilized in QP simulation) this one may be handled easily in terms of constraints.

A valve in the fluid path may be modelled by changing the third constraint in figure 3.4b. The required change is to predicate this constraint on the device state of a valve as follows.

\[
\text{If Valve-Area > 0} \\
\text{then Flow-Rate = Press-Rel} \\
\text{else Flow-Rate = 0}
\]

where Valve-Area is a state variable for the area available for flow in the valve.

This type of constraint is an instance of an operating region as discussed in chapter 2. It also corresponds to a device state as described in the theory of DeKleer and Brown. However, Kuipers has analogous constructs for the constraint based ontology in both his QSIM and ENV programs.

The device model of a pressure activated valve, which is designed to allow flow from source to destination only, may be completed with the following constraint.

\[
\frac{d\text{Valve-Area}}{dt} = dVA \\
\text{Press-Rel} = dVA
\]

3.5 The Steamer Project

The STEAMER project Hollan, Hutchins, Weitzman (8) and Williams, Hollan, Stevens (11) is an experiment in the use of AI hardware and software technologies in an explanation and training system. The domain of this work has so far been limited to steam propulsion systems.
of naval vessels. However, the authors emphasize that their goal is to develop general technologies for training through simulation.

STEAMER is related to this thesis in that it bases its explanation capabilities on a simulation which is equipped with both graphical and language interfaces. The simulation model itself is of the quantitative variety and was developed independently of and previous to the STEAMER project. Therefore STEAMER is not relevant to the discussion of qualitative reasoning in this and the prior chapter. However, the simulation technique itself is not the focus of STEAMER. Rather, the goal is to produce an "interactive inspectable simulation" which can assist a student in forming a mental model of how the simulated system operates.

In STEAMER, great emphasis is placed on the graphical representation of the modelled system. This interface allows the user to observe representations of the controls and indicators of a power plant which are normally visible to the operator of a real plant. In addition, the user of the inspectable simulation is allowed to view phenomenon which is opaque in the real system. An example is the rate of flow of fluid through pipes. This additional information helps the user (student) to understand the causality in the system. The STEAMER program also allows the user to manipulate components and see the results of their actions.

Within the graphical interface there are representations for more than just the tangible components and indicators found in the real system. The visual explanation in STEAMER also makes use of graphic objects called "signal" or "derivative icons". These represent qualitatively the sign and magnitude of the first derivatives of significant quantities.

A general purpose graphics editor was developed in STEAMER which allows the graphic interface to be developed by a non-programmer. A domain expert creates a graphic object, such as a gauge or pump icon, through an interactive iterative refinement process. An icon initially has a generic set of default characteristics which may be changed and augmented. In addition to a visual representation (a bit map) the editor also generates LISP code that links the object to the simulator. The user of the editor specifies the variables (or functions of variables) in the simulation model that the parameters of the object are connected to.
Chapter 4: Mechanical Device Simulation

As stated in chapter 1, the present research effort investigates how to engineer a computer program capable of simulating a simple mechanical device and explaining its behavior. Such a system would have obvious applications in education and training. Additionally, this project may be viewed as testing the utility of qualitative models and simulation as a deep representation of the structure and function of a mechanical device. The extent to which such a representation may be used to "drive" both visual and linguistic accounts of this information will reflect its epistemological adequacy. This project is also an experiment to test the practicality of qualitative simulation when used to model a reasonably complex system. A following chapter describes such an application in detail (a model of a piston driven pump). The present chapter discusses the implementation of the simulation system itself.

4.1 Implementation of the Device Simulation System

As mentioned above a program has been developed which can simulate the behavior of a simple mechanical device, qualitatively. This program will be referred to as DSS for Device Simulation System. The input and output of this program will be used in the QA and animation interfaces. DSS is based primarily on the QSIM algorithm of Kuipers (3). This section will first present a relatively detailed discussion of the QSIM algorithm as utilized in DSS. The following section describes the input and output of a DSS simulation.

4.2 The QSIM Based Simulation Engine

The QSIM ontology is neutral with regard to whether a device is represented by its constituent components or by its constituent processes. Instead QSIM models a device with a set of state variables (parameters) and a set of constraints among them. QSIM is a true simulation algorithm by the definition given in chapter 2. The top level operation of this algorithm constitutes a search through the space of the
feasible states of a device. A feasible state is an instantiation of the model's state variables that satisfies the model's constraints. The search proceeds, given an initial state, by generating a set of possible next (temporally following) states. If this generated set contains more than a single next state then one is selected and the process recurs. Expanding a state, that is selecting it and generating its possible successors, will be referred to as a simulation step.

It is possible for a successor of a parent state to be identical to a previously generated state, perhaps even the parent. If a state selected for expansion has no successors or if all its successors have been generated previously then that behavior path is abandoned. The result of the search is the construction of a directed graph (see figure 4.1). Nodes in this graph are states of the device and paths through the graph represent feasible behaviors of the device. Because the behavior of most mechanical devices is periodic the search may terminate upon the discovery of a cycle in the graph. The following pseudocode illustrates the search algorithm currently implemented in DSS. This is a depth-first search where the selection of the next state to expand (in the function Expand-one-from) is performed by the user.

```
Procedure DSS(States)
    if Not-empty(States) and Not-cycle-found
        then DSS(Expand-one-from(States))
        DSS(Remaining(States));
```

An inspection of the simulation algorithm at a finer level of detail reveals that each simulation step consists of four parts. These are:

**Parts of a Simulation Step**

1. Selection of a state to expand.

2. Prediction of the set of possible transitions (next state values) for each parameter from the current values supplied by the state selected in step 1.

3. Local and pair-wise constraint filtering of the predictions generated in step 2.
This figure illustrates the general structure of the behavior network of a hypothetical simulation. The levels of this network correspond to alternating points and intervals of time.
4. Generation of a set of globally consistent next states based on the results of step 3.

Part 1

Since the simulator will usually be supplied with a unique initial state the first execution of part 1 is trivial. After this however, selecting one from a set of alternative next states is one of the most difficult problems in the simulation. ( the reasons why more than a single next state can be generated were discussed in the section on ambiguity in chapter 2 ). The current implementation of DSS leaves this choice to the user ( currently the user is the developer ). However, domain specific constraints, discussed in a following section, are utilized to narrow the field of choices as much as possible. Eventually heuristics and/or models with stronger constraints must be developed that will enable the program to make this decision itself. A heuristic best-first search algorithm might be employed in which the Expand-one-from function would choose the globally optimal state among all partially explored behavior paths. An alternative is to simply perform a breadth first search of the behavior graph. This would elaborate all feasible behaviors of the class of devices represented by the constraints. This is the approach actually taken in Kuipers' QSIM.

Part 2

This is a much more straightforward operation than part 1. The state selected in step 1 supplies a set of value instantiations for the parameters of the modelled system. Based on these "current" values of the state variables, including the sign of their derivatives, and on qualitative versions of the mean and intermediate value theorems ( see the section of chapter 2: Prediction and Propagation of Qualitative Value ) a set of possible next state values may be generated for each of the state variables. Note that this prediction is local to each individual parameter and is not affected by the current or predicted values for the other parameters. Figure 4.2 lists the transition rules used in QSIM to predict next state values. QSIM models time as a strict alternating series of instants and intervals. Thus there is a different set of transition rules for states representing instants and states representing intervals of time. Looking again at figure 4.1, notice that states are tagged as ITYPE, for intervals, and PTYPE, for points ( instants ), on alternating levels of the behavior graph.

The transition rules of figure 4.2 are based on the assumption that all state variables are continuously differentiable with respect to
Figure 4.2

Qsim Transition rules

Transitions from a PTYPE state:

\[(V \ (L1 \ STD) \ *) \Rightarrow (V \ (L1 \ STD) \ P1)\]
\[(V \ (L1 \ STD) \ *) \Rightarrow (V \ ((L1 \ L2) \ INC) \ P2)\]
\[(V \ (L1 \ STD) \ *) \Rightarrow (V \ ((L0 \ L1) \ DEC) \ P3)\]
\[(V \ (L1 \ INC) \ *) \Rightarrow (V \ ((L1 \ L2) \ INC) \ P4)\]
\[(V \ ((L1 \ L2) \ INC) \ *) \Rightarrow (V \ ((L1 \ L2) \ INC) \ P5)\]
\[(V \ (L1 \ DEC) \ *) \Rightarrow (V \ ((L0 \ L1) \ DEC) \ P6)\]
\[(V \ (L1 \ L2) \ DEC) \ *) \Rightarrow (V \ ((L1 \ L2) \ DEC) \ P7)\]

Transitions from an ITYPE state

\[(V \ (L1 \ STD) \ *) \Rightarrow (V \ (L1 \ STD) \ I1)\]
\[(V \ ((L1 \ L2) \ INC) \ *) \Rightarrow (V \ (L2 \ STD) \ I2)\]
\[(V \ ((L1 \ L2) \ INC) \ *) \Rightarrow (V \ (L2 \ INC) \ I3)\]
\[(V \ ((L1 \ L2) \ INC) \ *) \Rightarrow (V \ ((L1 \ L2) \ INC) \ I4)\]
\[(V \ ((L1 \ L2) \ DEC) \ *) \Rightarrow (V \ (L1 \ STD) \ I5)\]
\[(V \ ((L1 \ L2) \ DEC) \ *) \Rightarrow (V \ (L1 \ DEC) \ I6)\]
\[(V \ ((L1 \ L2) \ DEC) \ *) \Rightarrow (V \ ((L1 \ L2) \ DEC) \ I7)\]
\[(V \ ((L1 \ L2) \ INC) \ *) \Rightarrow (V \ (L* \ STD) \ I8)\]
\[(V \ ((L1 \ L2) \ DEC) \ *) \Rightarrow (V \ (L* \ STD) \ I9)\]

These are the transition rules adopted from QSIM, Kuipers (B4). The rules shown are for a hypothetical state variable V with quantity space (... L0 L1 L2 ...). The syntax on each side of the \( \Rightarrow \) indicates an instantiation of the variable V. The syntax is (Name (Value Direction) Rule) where Value may be a landmark or interval, Direction represents the sign of the derivative (w.r.t. time) of the variable and Rule identifies the transition rule which predicts the transition. A * indicates that the rule identification is irrelevant.
time. While at a precise quantitative level this assumption will almost always be true for the parameters of a real physical device, there are cases where strict continuity forces QSIM to generate states that may not be required to display the qualitative behavior of a device. A particular example of such a situation and the way it is handled in DSS will be discussed in chapter 5.

Most of the transition rules in figure 4.2 follow directly from the Generalized Qualitative Integration Rules given in chapter 2. I8 and I9 however, represent a part of QSIM that is unique among the theories of qualitative physics discussed in chapter 3. These rules allow the simulation to predict new landmark values for state variables. Consequently, the quantity spaces of the model parameters may be modified as the simulation proceeds with the addition of new landmarks between those given in the model input. New landmark values may also result in the discovery of new correspondences between variables involved in monotonic proportionality constraints.

Part 3

The third and fourth parts of a simulation involve a constraint satisfaction algorithm called Waltz filtering. This algorithm is due to Waltz (12) and was originally developed for labelling line drawing junctions in a vision system.

Mackworth and Freuder (13) describe a constraint satisfaction problem (CSP) as follows: "Given a set of n variables each with an associated domain and a set of constraining relations each involving a subset of the variables, find all possible n-tuples such that each n-tuple is an instantiation of the n variables satisfying the relations." In an earlier paper, Mackworth (14), Mackworth discusses the naive CSP solution method consisting of a depth-first backtracking search through the instantiation space of the variables. He points out that this can be a very inefficient technique because the same inconsistent instantiations for two variables may be tried and rejected several times in the course of backtracking. The Waltz algorithm works as a preprocessor for backtracking search. This algorithm reduces the size of the domains of the individual variables by applying pairwise constraints among them. Reducing the size of these domains in turn reduces the amount of backtracking required in the ensuing search.

The ultimate constraint satisfaction problem in QSIM is to find all assignments of values to the state variables which are consistent

---

12 For a formal derivation of these rules see Kuipers (3).
with the constraints defining the modelled device. However, the Waltz algorithm may be applied directly in QSIM by first formulating a different CSP. In this context the "variables" in Mackworth's definition of a CSP are the constraints which describe the modelled mechanical device. To reduce confusion, in the following discussion these "variables" (the device constraints) will be referred to as nodes. The "domain" of each node is a set of tuples of state variable instantiations. These tuples are generated in the following way:

Node Labeling

- Each node involves a particular set of state variables. A set of candidate tuples is generated for each node by forming the cross product between the sets of possible next state values of each state variable (recall that these sets of possible next state values were generated in step 2 above).

- Each of these candidate tuples is then either accepted or rejected as locally consistent with the node (itself a constraint) according to the rules of qualitative relationships discussed in chapter 2.

The "constraining relations" in Mackworth's definition correspond to the following simple rule for pair wise consistency between nodes.

Pair-Wise-Consistency Rule

Two nodes are said to be pair-wise consistent if and only if for each tuple attached to one node there exists at least one tuple attached to the other such that all the value instantiations of the state variables in both tuples are consistent (that is the instantiations for common variables match). Note that two nodes that do not have any common state variables are trivially pair-wise consistent.

The Waltz algorithm in QSIM is used to achieve pair-wise consistency among the labelings of all the nodes in the device description. The initial labeling is achieved as described above. In the following let NODES be a set of constraints which define a mechanical device. These
constraints are part of the model input. At the top level the Waltz algorithm makes a single iterative pass "Visiting" each element of NODES.

Procedure Waltz(NODES)
    for-each N an-element-of NODES
    Visit(N);

Visiting a node is a 3 step procedure.

Procedure Visit(NODE)
    Label(NODE);
    Constrain-By(NODE, Neighbors-of(NODE));
    Propagate-To(NODE, Neighbors-of(NODE));

The Label procedure attaches the set of locally consistent tuples (the domain) to NODE. At the outset all nodes are unlabeled. The Constrain-By procedure removes all tuples from the domain of NODE which are not pair-wise consistent with all its labeled neighbors (the neighbors of a node are all other nodes which share one or more state variables with it).

Procedure Constrain-By(NODE, NEIGHBORS)
    for-each NBR an-element-of NEIGHBORS
    if Labeled(NBR)
        then Pair-Wise-Constrain-By(NODE, NBR);

The Propagate-To procedure uses the tuples of NODE which survived Constrain-By to likewise constrain (again by the Pair-Wise-Consistency rule) the domain of each of its labeled neighbors. If the domain of a neighbor is reduced as a result of achieving pair wise consistency with NODE then this change must be propagated to all of its labeled neighbors (except NODE of course). This is accomplished in the recursive call inside Propagate-To.

Procedure Propagate-To(NODE, NEIGHBORS)
    for-each NBR an-element-of NEIGHBORS
    if Labeled(NBR)
        then do
Pair-Wise-Constrain-By(NBR, NODE)
if Changed(Domain(NBR))
    then Propagate-To(NBR, Remove(NODE, Neighbors(NBR)));
end;

As stated above the Waltz algorithm terminates after visiting each node exactly once. However, because it is possible for the propagation of domain changes to be cyclic in Propagate-To one might worry about non-termination. It may be seen that this is not a problem by noticing that propagation continues only so long as tuples are deleted. Since the number of tuples attached by the Label procedure is finite and since each node is labeled only once the algorithm is guaranteed to terminate.

Part 4

Once the Waltz filtering stage is complete, each node is left with a set of surviving tuples. The elements of each set are locally consistent with their containing node and pair-wise consistent with all other nodes. To generate the possible next states we need to find all global instantiations of the state variables that are consistent with the constraints (the nodes). Any collection of tuples, one from the surviving set of each constraint, such that all instantiations of any one state variable are consistent, defines a valid feasible next state.

Forming such collections of consistent tuples is performed through a depth-first search. Assuming an arbitrary ordering over the nodes and the tuples within nodes, this search begins by selecting the first tuple of the first node. At the second node, the search selects the first tuple that is consistent with the tuple selected at node 1. Such a tuple is guaranteed to exist because Waltz filtering assures pair-wise consistency and by the Pair-Wise-Consistency rule we know that for any tuple at one node of a pair there exists a consistent tuple at the other node. However, we are not guaranteed to find a tuple at the third node that is consistent with both tuples selected previously. To see this consider the following example involving three nodes each of which is a constraint involving two state variables.

<table>
<thead>
<tr>
<th>Node</th>
<th>State Variables</th>
<th>Tuples</th>
</tr>
</thead>
<tbody>
<tr>
<td>N1</td>
<td>A B</td>
<td>(A.1 B.1) (A.2 B.2)</td>
</tr>
<tr>
<td>N2</td>
<td>B C</td>
<td>(B.2 C.3) (B.1 C.2)</td>
</tr>
<tr>
<td>N3</td>
<td>C A</td>
<td>(C.3 A.1) (C.2 A.2)</td>
</tr>
</tbody>
</table>
Within a tuple \( V.N \) indicates that state variable \( V \) is instantiated with value \( N \).

Although these three nodes are pair-wise consistent there is no combination of one tuple from each that is globally consistent. Hence, it is possible that the search for a global instantiation may at times have to backtrack, even though pair-wise consistency is assured. It is also possible that no instantiation can be found. In this case, the current path in the behavior network is abandoned.

Mackworth (14) demonstrates a phenomenon of backtracking search in constraint satisfaction called "thrashing" in which the same inconsistency is discovered many times along different paths of the search. Pair-wise consistency filtering reduces thrashing. Mackworth also describes an algorithm to achieve path-consistency which he claims will assure that thrashing does not occur. The incorporation of this algorithm into DSS is a possible future enhancement.

Having elaborated the major components of a simulation step, the pseudocode for the best-first top level algorithm presented earlier may be rewritten as follows:

```
Procedure DSS(States)
  if Not-empty(States) and Not-Cycle-Found
    then State\(^*\) <= Select-One-from(States);
      Set-Context(State\(^*\));
      Generate-Transitions(State\(^*\));
      Waltz(Constraints);
      DSS(Generate-Next-States(Constraints));
      Clear-Context(State\(^*\));
      DSS(Remove(State\(^*\),States));
```

The procedure Set-Context and Clear-Context are necessary because the QSIM algorithm generates new landmarks values and correspondences. When a path leading from a state which includes new landmarks (and therefore possibly new correspondences) is to be explored these landmarks (and correspondences) must be asserted in the quantity space of the involved state variables. If the search later backtracks past this state these assertions must be retracted.
4.3 DSS Input and Output

Objects in both the input and output of the current implementation of DSS are represented with frames. For the sake of portability\textsuperscript{13} the frame system is hand coded in LISP rather than using dialect dependent features such as flavors or structures. A simple algorithm from Winston and Horn (15) was adopted for this purpose. This algorithm implements frames as generalized association lists. A frame in DSS is stored as the value of an atom whose PNAME is the name of the object that the frame represents. In ZETALISP the definition of a DSS frame appears as follows:

\[
\text{(DEFVAR Frame1 '((Slot1 (VALUE Slot1-Val))}
\text{(Slot2 (VALUE Slot2-Val))}
\text{...}
\text{(SlotN (VALUE SlotN-Val))})
\]

The operations defined on these frames include FGETV, retrieve the value of a slot, FRPLACV, replace the value of a slot, FPUTV, add to the value of a slot.

Several of the input and output objects contain instantiated state variables in the value of one or more slots. The general syntax of an instantiated state variable is:\textsuperscript{14}

\[
\text{(Vname (Value Direction) Trule)}
\]

where:

- Vname is the name of the state variable.
- If the current value of the variable is a landmark then Value is the name of that landmark (an atom). If the current value is an interval then Value is a list of two elements, (L-bound

---

\textsuperscript{13} The system currently runs on a DEC-20 in ELISP and on a Symbolics 3600 in ZETALISP. The Apple Macintosh is a target machine of the future.

\textsuperscript{14} This syntax is very similar to that used in QSIM.
U-bound), where L-bound and U-bound are the landmarks comprising the upper and lower boundaries of the interval.

- Direction is one of STD, INC, or DEC representing the sign 0, +, - respectively, of the derivative of the variable.

- Trule is the name of the transition rule which predicted this instantiation.

Input Objects

The input to DSS contains objects that collectively describe a mechanical device. Currently the object types required in these descriptions are System, State, Variable and Constraint. The syntax and semantics of each of these objects will be discussed in turn. Two slot types, that are common to all object types in DSS, will be omitted from the lists below. These types are INST, which always contains the object type as its value, and NAME, which contains the object name as its value.

1. System: There is a single system object in a device description. It serves as a root to which all other objects are connected, perhaps indirectly.

- VARIABLES - a list of the names of all state variables included in the device description.

- CONSTRAINTS - a list of the names of all constraints included in the device description.

- CORRESPONDENCES - a list of all the correspondences between state variables (see chapter 2).

- RANGES - a partial ordering over the lengths of the "normal" range of motion for all physical objects in the modelled device. The range of motion of a physical object is defined by the quantity space of the state variable representing the position of that object. This lattice is used heuristically by the simulator to prune possible next states. A discussion of this heuristic will follow in the next section.

- INITIAL-STATE - The name of the frame representing the initial state of the device. This is the state used in the first step of the simulation.
2. **State:** There is usually only one State object in a device
description: the initial state. However, the output of a simulation
is composed almost entirely of State frames. Therefore, the structure
of this these objects will be discussed in the next section describ-
ing the simulation output.

3. **Variable:** These objects represent each of the state variables of
the modelled device.

   - **TYPE** - the type(s) of this state variable. These currently
     include POSITION, DERIVATIVE and AMOUNT. More types may be
     introduced in the future.

   - **DERIVATIVE** - points to the state variable representing the
     derivative of this variable. This value may be NIL if the deriva-
     tive of the containing variable is not represented in the device
     model.

   - **ANTIDERIVATIVE** - if the containing variable has type DERIVATIVE
     this value points to the variable it is the derivative of.

   - **LANDMARKS** - a list of landmarks representing the quantity space
     of the containing variable.

4. **Constraint:** There is a frame to represent each constraint included
in the device description. These data structures are used in the
Waltz filtering algorithm.

   - **FORM** - this is the definition of the constraint. Currently in
     DSS the following constraint forms have been implemented ( see
     chapter 2 for the qualitative semantics of each type of relation-
     ship ):

     a. \((D\text{-}Dt \ V1 \ V2) \Rightarrow V2 \text{ is the derivative of } V1.\)

     b. \((M+/- \ V1 \ V2) \Rightarrow V1 \text{ and } V2 \text{ are directly ( } M+ \text{ ) or inversely}
       ( M- ) qualatatively proportional.

     c. \((ADD \ V1 \ V2 \ V3) \Rightarrow V3 = V1 + V2, ( + \text{ indicates qualitative}
       addition ).\)

     d. \((MULT \ V1 \ V2 \ V3) \Rightarrow V3 = V1 * V2, ( * \text{ indicates qualitative}
       multiplication ).\)
e. \((\text{QEQ} \ V \ C) \Rightarrow V \text{ is equal to the constant } C.\)

f. \((\text{COND} ((c11 \ c12 \ \ldots \ ) \ \text{form1})
   ((c21 \ c22 \ \ldots \ ) \ \text{form2})
   \ldots
   ((cM1 \ cM2 \ \ldots \ ) \ \text{formM}))\)

This conditional form is used to specify operating regions for a constraint. The \(cij\)'s are conditions on state variables. If \(\text{True}(cij)\) for all \(j\) in the \(i\)'th conjunction then form\(i\) is asserted. The form\(i\)'s are standard constraint forms as above. An example of the motivation for and use of this type of form appears later.

- TUPLES - the current labelling of the constraint in the Waltz filtering algorithm (see the preceding section).

**Output Objects**

The output of a DSS simulation consists of State objects. These states are the nodes in the behavior network of the device. The form of a State object frame is as follows:

**State**

- NAME - The name of the state. For all states other that the initial state this name is generated from a numeric sequence.

- STATUS - Each state has a status of either active or inactive. A state is active if and only if it lies on the path through the behavior network which is currently being explored. Otherwise, the state is inactive.

- PREDECESSORS - A list of all states for which the containing state has been found to be a feasible next state.

- SUCCESSORS - A list of all states that have been found to be feasible next states for the containing state.

- FORM - A list containing a value instantiation for each state variable in the device description. These instantiations were found to be globally consistent in part 4 of the simulation step which generated this state (see the preceding section). This
list provides the current values for the states variables when and if the containing state is expanded.

- **TYPE** - Designates whether the state represents an interval of time, **ITYPE**, or a point in time, **PTYPE**.

- **NEW-LMARKS** - A list of all new landmark values that were discovered in the process of generating this state. Each new landmark is stored with information that allows it to be inserted into its respective quantity space when the containing state becomes activated and retracted when the state becomes deactivated. (preceding section).

- **NEW-CORRESPONDS** - A list of new correspondences discovered while generating the containing state. These are asserted and retracted in the same manner as **NEW-LMARKS**.

### 4.4 The Special Status of Objects in DSS

The DSS system is intended to simulate the operation of simple mechanical devices. This domain involves most of the concepts of qualitative physics e.g. fluid flow, pressures, mechanical linkages etc. However, the movement of discrete physical objects is especially relevant in such devices. Therefore, objects and the motion of objects are given special status in this system. State variables of type position are treated differently from amounts and derivatives in several ways. For example, a heuristic used in predicting possible next state values (Part 2 of a simulation step as discussed in the preceding section) says that new landmarks should not be generated for position variables unless no possible next state is found otherwise. Support for this rule comes from the nature of device component motion. Typically it is the case that the range of motion of a component is constrained by limits that are known a priori. Likewise, the landmark positions between these extrema may be specified beforehand. In other words the quantity space representing the normal range of motion of a component is defined by the device structure and may often be fully specified as input.

\[15\] Recall that position, amount and derivative are the only variable types currently defined.
The quantity spaces of position variables differ from those of
the other two variable types in another way. Variables of type deriva-
tive and amount always include zero in their quantity space. The status
of these variables as either negative, positive or zero is important in
satisfying the qualitative constraints they are involved in. The value
zero is consistent and absolute for amounts and derivatives but arbitrary
and relative for positions. Therefore, zero is not used in the quantity
spaces of these variables. Rather, the quantity space of a position vari-
able consists of symbolic names for its extrema and significant interme-
diate points.

The input for a device model in DSS includes a partial ordering
over the lengths of the ranges of motion for all position variables. This
ordering is used heuristically to eliminate some ambiguity when predict-
ing the next state in a simulation. An example of this sort of ambiguity
was given in chapter 2 involving two valves both of which were open and
in the process of closing. In this case, the possible next states must
include (a) both valves having closed, (b and c) one having closed and
the other still open, and (d) both still open. In general the number of
next states is exponential in the number of moving objects. In the case
where the relative speeds and distances traveled by the two ( or more )
objects are comparable this ambiguity is valid. However, if the
speed-distance product for one object is of a different order of magni-
tude than the other, the simulator should utilize this information to
prune away the ambiguity. In this sense the partial ordering of relative
lengths in DSS is actually a relation among orders of magnitude of
speed-distance products. This heuristic is somewhat ad hoc. Future work
is needed to develop a general and formal theory of the relative time
granularity of events.
Chapter 5: Modelling a Pressure Activated Valve

The next chapter discusses the DSS model for a particular device, a piston pump. One of the components (sub-devices) of this piston pump is a pressure activated valve. This chapter describes the evolution of the valve model which is used in the current piston pump model.

Figure 5.1 shows a graphic rendering of a possible valve design. The valve allows fluid to flow through it in one direction but prevents flow in the other. In figure 5.1 flow is allowed from side A to side B. The device is activated by the pressure difference across the valve. In this example, if the pressure on side A is greater than that on side B (press(A) > press(B)) the valve will open and fluid will flow from A to B. If the converse is true, press(B) > press(A), then the valve will close. The range of motion of the valve is constrained by the valve housing when closed and the valve stem guide when opened maximally.

5.1 First Valve Model

A first attempt at modelling this device defines three state variables Vx, dVx and Prel as follows:

(Defvar Vx
 ;; The position of the Valve
 '((Inst (VARIABLE))
   (Type (POSITION))
   (Derivative (dVx))
   (Landmarks (VxMin VxMax)))

The landmarks VxMin and VxMax refer to the fully closed and the fully open positions respectively.

(Defvar dVx
 ;; The velocity of the valve. The motion of the valve is
 ;; defined to be linear.
 '((Inst (VARIABLE))
   (Type (DERIVATIVE))

48
Figure 5.1

A Pressure Activated Valve
(Antiderivative (Vx))
  (Landmarks (MINF 0 INF)) ))

(Defvar Prel
  ;; The pressure difference from side A to side B of the
  ;; valve. This represents the net force on the valve in
  ;; this model.
  '((Inst (VARIABLE))
    (Type (AMOUNT))
    (Landmarks (MINF 0 INF)) ))

This model also defines two constraints as follows.

(Defvar C1
  ;; dVx is the derivative of Vx
  '((Inst (CONSTRAINT))
    (Form (D-dt Vx dVx))
    (Neighbors (C2)) ))

(Defvar C2
  ;; The valve opens under a positive difference and
  ;; closes under a negative difference.  
  '((Inst (CONSTRAINT))
    (Form (M+ dVx Prel))
    (Neighbors (C1)) ))

With associated correspondence (( dVx 0 ) ( Prel 0 )).

Unfortunately this simple model will not work. The problem is
that the valve will not be able to close, i.e. reach VxMin and become
steady,\footnote{In this simple model inertia is ignored.} while the pressure difference is negative. To see this
observe that, by C2 and the associated correspondence, as long as Prel
is negative so must be dVx. Furthermore, by C1, as long as dVx is nega-
tive Vx must be decreasing. Therefore, Vx cannot become steady while
Prel is negative.

\footnote{A symmetric situation exists for the valve reaching its fully open
position but will be ignored here.}
5.2 Second Valve Model

The problem above stems from the fact that the only represented force on the valve is the pressure difference, $P_{rel}$. In order to allow the valve to be shut under a negative pressure a second force imparted by the valve housing must be represented which can equalize the force of the pressure on the valve. To do this we introduce a state variable, $SF$, to represent that force and revise constraint C2.

(Defvar SF
  ;; Structural force of the pump housing
  '((Inst (VARIABLE))
    (Type (AMOUNT))
    (Landmarks (MINF 0 INF)))
)

(Defvar C2.1
  '((Inst (CONSTRAINT))
    (Form (ADD Press SF dVx))
    (Neighbors (C1)))
)

The total force on the valve is now defined as the sum of the force from the pressure difference and the force from the housing structure.

The problem now is to constrain the value of the structural force, $SF$, in relation to the rest of the system. One approach is to model the elasticity of the valve housing. To do this the quantity space for the valve position variable $Vx$ is changed to the following:

(Landmarks ($Vx_{Min}$- $Vx_{Min}$ $Vx_{Max}$ $Vx_{Max}$+))

When the value of $Vx$ is in the interval ($Vx_{Min}$- $Vx_{Min}$) the valve is closed and the valve housing has "bent in" some under the force of Press.

With this revised quantity space we can define the variable $SF$ in terms of $VX$. However, this requires a new form of constraint, a conditional constraint.

(Defvar C3
  '((Inst (CONSTRAINT))
    (Form (COND
      ( ((VxMin <= Vx <= VxMax+))
        (EQ SF ZERO) )
      ( (VxMin- <= Vx < VxMin))
    ))
)
\[(M \text{- SF } Vx) \)  
\[(\text{Neighbors (C1 C2.1))})\]

The following correspondence is also required:
\[(\text{Correspond (SF 0) (Vx VxMin)))}\]

The first condition of this constraint states that when the valve is open (in the interval \((VxMin \text{ VxMax})\) or when it is in contact with but not pushing in on the valve housing (at the landmark \(VxMax\)), the structural force is zero and steady. The second condition specifies that when \(Vx\) is at landmark \(VxMin-\) or in the interval \((VxMin- \text{ VxMin})\) the relation \((M- \text{ SF } Vx)\) holds i.e. the farther the valve bends the housing (the more \(Vx\) decreases) the greater the resistant force the housing exerts.

This model exhibits the desired behavior. If we consider a situation where the pressure difference is negative and steady the behavior is straightforward. In this case, the valve closes (reaches \(VxMin\)) and continues past this point, pressing in on the valve housing, until the structural force, SF, is equal and opposite the Press. At that time, \(Vx\) reaches a landmark and becomes steady.

In a second situation, the pressure difference is negative and constantly decreasing (i.e. the negative force, Press, is getting stronger) without bound. Again in this case, the valve closes and begins to press in on the housing. By C3, this causes SF to become positive and increasing, opposing Press. Unfortunately, at this point the ambiguity of qualitative addition rears its ugly head via constraint C2.1 and the simulation begins to periodically branch in two directions. These directions correspond to the case where the valve motion is slowing down because SF is increasing faster than Press is decreasing and the converse: the valve motion is speeding up because Press is decreasing faster than SF is increasing.\(^{18}\) A purely local simulation (one in which predictions are based only on the current state) could even select the two cases alternately or at random resulting in a prolific branching of behaviors. With Press constantly decreasing, the constraints will not allow a consistent state where the motion terminates.

---

\(^{18}\) The second case could predict the possible, though abnormal, situation where the device fails because the valve housing collapses. This could be represented in the simulation as \(Vx\) moving below \(VxMin-\).
The second situation above is artificial. In most devices, like a piston pump, the pressure might be negative and decreasing for a time, causing the valve to move into the interval \((VxMin - VxMin)\), but eventually it would stabilize, begin increasing, reach zero and finally become positive (these changes being due to events in other areas of the device). However, during any period when the pressure is negative and decreasing the simulation could embark on a wild goose chase like that described above. To prevent this, heuristics and/or stronger model constraints would have to be developed which could distinguish the proper behavior path from the myriad of erroneous possibilities. Even if a deterministic behavior is found the value of including the states which represent the settling behavior of the valve is dubious. If the valve is a sub-device of some larger mechanism (as it is in this case) this settling behavior may be microscopic in relation to the behavior of the overall mechanism.

5.3 Third Valve Model

The considerations above motivate yet another model of the pressure activated valve for use in the simulation of the piston pump. When the valve model is embedded in the larger device it is desirable that the number of states representing the behavior of the valve be held to the minimum necessary to reveal its function with relation to the overall device. A succinct way to describe the motion of the valve when it reaches either end of its range of motion is that it terminates discontinuously. In this manner, modelling the microscopic settling behavior of the valve may be avoided. Such a model operates at a higher level of abstraction: a series of states that represent a smooth transition at a finer level of granularity are represented as a discontinuous change at this more coarse level. This abstraction reduces the computational burden on the simulation system: it need consider fewer states and make fewer decisions to resolve ambiguous branching. But perhaps more importantly, the output represents the device behavior more concisely and at a more consistent level. At the same time however, it is gratifying to know that the qualitative simulation principles are capable of reasoning at the finer level of detail if called upon to do so. A simulation system capable of reasoning at multiple levels of precision seems superior for the task of explanation. This hypothesis is bolstered by evidence, see Williams, Holohan and Stevens (16), that people utilize multiple complementary mental models in understanding physical phenomenon.
The final valve model does not explicitly represent the elasticity of the valve housing. Therefore, the quantity space for the valve position state variable, Vx, may be reverted to its original form:

\[(\text{Landmarks} \ (Vx_{\text{Min}} \ Vx_{\text{Max}}))\]

The explicit representation of the structure force, SF, is also no longer required. This means that constraint C3 may be abandoned and constraint C2.1 must be replaced by the following:

\[(\text{Defvar} \ C2.2)\]

\[\text{;; The valve opens under a positive pressure difference}\]

\[\text{;; and closes under negative difference. However, if the}\]

\[\text{;; valve is shut it will not respond to a negative}\]

\[\text{;; difference.}\]

\[((\text{Inst} \ (\text{CONSTRAINT}))\]

\[\text{(Form} \ (\text{COND})\]

\[\text{((Vx = Vx_{\text{Min}}) \text{ and } (Press < 0))}\]

\[\text{ (\text{EQ} \ dVx \ \text{ZERO})}\]

\[\text{ (T (M+ dVx \ Pre1))})\]

The problem is how to predict the discontinuous behavior of the valve motion when it closes. Recall that the transition rules in figure 4.2 were derived based on the assumption that the state variables represent continuously differentiable functions. Herein lies the problem. In order for the above model to work as desired, the state variable, dVx, must make a discontinuous leap in value in a specific situation. This situation is as follows. Assume that the valve is open but is closing under a negative and decreasing (strengthening) pressure. The Interval type state representing this situation has the following form:

\[\text{Istate-1}\]

\[(Vx ((Vx_{\text{Min}} \ Vx_{\text{Max}}) \text{ DEC}) *)\]

\[(dVx ((\text{MINF O}) \text{ DEC}) *)\]

\[(\text{Press} ((\text{MINF O}) \text{ DEC}) *)\]

The direction of dVx must necessarily be decreasing in Istate-1 because of constraint C2 and the value of Press. The state we would like to reach has the following form:
(Vx (VxMin STD) *)
(dVx (O STD) *)
(Press ((MINF O) DEC) *)

This state is consistent with the constraints of the valve model. Unfortunately, however, its values for Vx and dVx are never predicted by the basic transition rules from the values in Istate-1. The desired transition for dVx represents a discontinuity and the desired transition for Vx implies a non-differentiable change. Figure 5.2 a graphs the general curves that the desired predictions define. In order to abstract out the settling behavior of the valve we are effectively cutting out some intermediate states of the simulation. It is this abbreviation of the behavior network which causes the above conflict with the prediction rules of QSIM.

To resolve these conflicts we must somehow augment the transitions predicted by the transition rules to include those desired. This could be accomplished by loosening the constraint on the local next state value predictions imposed by the transition rules. In other words, we could add more rules. However, this could potentially increase the number of predicted transitions for all of the state variables in each step of the simulation. The result could be a dramatic increase in computation as well as predicting unwanted and meaningless discontinuity in the model's behavior. So the question is how to recognize those specific situations and variables where discontinuous predictions are warranted.

Two possible methods have been investigated for predicting discontinuities. Both have an interpretation in which the concept of process is used to predict when discontinuities should be introduced. In the valve example a motion process is relevant. As was mentioned earlier, the motion modeled here corresponds to the Aristotelian motion process defined in the QP theory of Forbus (6), see figure 5.3. The preconditions slot represents the knowledge we require. In the context of the valve model, this condition states that the its motion may occur only within its structural bounds. In DSS we would like to predict that when this condition is violated the motion must end and hence that the discontinuity should be predicted. The Let clause in the Relations slot of the QP model indicates that the existence of velocity depends on the motion process. In DSS this corresponds to the discontinuous prediction
Discontinuous Closing of a Valve

The graphs in the left column represent the qualitative shape of the values for the position, $V_x$, and the velocity, $dV_x$, of the valve as well as the pressure difference, $P_{rel}$. The right hand column shows the position of the valve before, after and at distinguished time point $P_1$. 
Process: Motion

Individuals:
  B an object, Mobile(B)
  dir a direction

Preconditions:
  Free-Direction(B,dir)
  Direction-Of(dir,net-force(B))

Quantity Conditions:
  A[net-force(B)] > ZERO

Relations:
  Let velocity be a quantity
  velocity $\propto$ net-force(B)
  velocity $\propto$ mass(B)

Influences:
  I+(position(B),A[velocity])

A Process Model for Aristotelian Motion
to (0 STD) that is required for dVx when the motion process becomes inactive.  

Figure 5.4 depicts an abstract flow diagram for the QSIM (and DSS) simulation algorithm. First, prediction rules generate a set of possible next state values for each state variable. Recall that these predictions are made locally for each state variable. These predictions then become input for the constraint satisfaction procedure which produces a set of possible next states.

Two possible scenarios for introducing non-smooth transitions into this process correspond to the two stages of the algorithm. The first proceeds as follows. First, the standard prediction rules (those in figure 4.2) generate all the smooth predictions. Next, the set of active process models is scanned to see if any combinations of these predictions will make a process(s) become inactive. If so the process model is used to predict discontinuous transitions for relevant variables e.g. velocity in the motion process.

This technique has been implemented and works satisfactorily. However, in this scheme new predictions may be added for a variable based on assumptions of which predictions will ultimately be included in the next state for other variables. Therefore, the predictions must be partitioned so that in the constraint satisfaction phase the introduced transitions are only considered in conjunction with the predictions that brought them about. The number of partitions may be exponential with the number of processes and the constraint satisfaction process must be run independently for each partition.

A second algorithm for introducing non-differentiable transitions keys on the possible next states output from the constraint satisfaction procedure. Based on the current values for variables in each of these states, non-smooth predictions may be added to the local predictions normally generated when predicting its successors. This approach has the advantage that discontinuities are triggered by combinations of state variables observed in the isolation of a single state. Therefore, the partitioning has already been done by the basic QSIM algorithm. Furthermore, some combinations that would have had to be considered when following the previous algorithm may have already been ruled out because

---

19 For the sake of completeness, note that the second line in the QP Relations of figure 5.3 corresponds to constraint C2 (there is no constraint corresponding to the third clause because inertia, and hence mass, is ignored in this model). Note also that constraint C1 is equivalent to the QP models' influences slot.
Abstract Flow Chart for the QSIM Algorithm
they were inconsistent with the device constraints. Other heuristics, such as the detection of stutter or duplication of states, may further reduce the number of states to be considered.

This second approach is the one currently implemented in DSS. However, explicit process models are not represented. Rather, a set of production rules is applied to the values of the variables in a predicted state to generate discontinuities. These rules are similar to the standard QSIM prediction rules except that their antecedent refers to a specific variable and their consequent may make predictions for more than one variable. The rule used in the valve model has the following form:

\[(Vx \ (VxMin \ DEC) \ *) \Rightarrow (Vx \ (VxMin \ STD) \ D) \ (dVx \ (O \ STD) \ D)\]

For the sake of illustration the following rule could model the situation where the valve bounces against the pump housing:

\[(Vx \ (VxMin \ DEC) \ *) \Rightarrow (Vx \ (VxMin \ INC) \ D) \ (dVx \ ((0 \ INF) \ DEC) \ D)\]

The combination of discontinuity rules and the conditional constraints discussed earlier provide a powerful and general language for defining a simulation model for a device. A possible future enhancement to this scheme would be to explicitly represent the situation where a derivative does not exist. For example the rule:

\[(Vx \ ((VxMin \ VxMax) \ DEC) \ *) \Rightarrow (Vx \ (VxMin \ DEC-STD) \ D)\]

Here DEC-STD represents the situation when Vx is not differentiable at a point in time but was decreasing an infinitesimal time interval earlier and steady an infinitesimal time interval later. The qualitative relations would have to be redefined such that they could decide on the consistency of values involving this kind of derivative. Also some new representation for a variable representing a derivative that does not exist would have to be created e.g. what would be the value of dVx when Vx = (VxMin DEC-STD) ?.
Chapter 6: A Piston Pump Model

This chapter describes a complete DSS model for a piston pump mechanism, see figure 6.1. The piston pump device was selected for this experiment because it is complex enough to make simulation and explanation interesting while remaining tractable. Also, instances of a wide variety of physical phenomena are contained within this device including: both rotary and linear motion, relative pressures, fluid flows, and mechanical linkages. The remainder of this chapter describes how each subsystem of the device is modelled.

6.1 The Slider and Crank Subsystem

A piston pump operates by alternately creating positive and negative pressure (relative to the atmosphere) within the pump housing. This is accomplished by moving a piston inside a cylinder to decrease and increase the volume within the housing. The movement of the piston, in turn, is caused by a piston rod attached to a rotating crank. The crank may be turned manually or by a motor.

The motion of the crank in the slider-crank assembly is rotary. Because rotary motion involves two dimensions, two state variables are used to represent the position of the crank (actually the end of the crank where it attaches to the rod). The current model of this device assumes that the angular velocity of the crank is constant. This and the trigonometric relationships between the position variables, their derivatives and the position of the slider complete this subsystem model yielding constraints as shown in figure 6.2.

6.2 Piston and Cylinder Subsystem

The constraints on the position and the velocity of the piston may be taken directly from the slider-crank model by equating the slider with the piston. Furthermore, it is easy to see that the volume within the cylinder (and hence the entire pump housing) is proportional to
The position of the Crank is modelled in two dimensions as \((Cy, Cx)\). Simple trigonometry and calculus give the following relationships:

\[
\begin{align*}
Cy &= -R \cos \theta \\
Cx &= R \sin \theta \\
dCy/d\theta &= R \sin \theta \\
dCx/d\theta &= R \cos \theta
\end{align*}
\]

Assuming that \(d\theta/dt\) is a non-zero constant i.e. that the crank is rotating at a constant velocity we can write:

\[
\begin{align*}
Cx &\propto dCy/dt \\
Cy &\propto -dCx/dt
\end{align*}
\]

or in DSS

\[
(M+ Cx \; dCy) \\
(M- Cy \; dCx)
\]

A good quantitative approximation of the relationship between the displacement, \(\Delta s\), of the slider and the position of the crank is:

\[
\Delta s = R(1-\cos \theta) + R^2/2L \sin^2 \theta
\]

Holding \(R\) constant and increasing \(L\) without bound yields:

\[
\Delta s = R(1-\cos \theta)
\]

This leads to the qualitative constraint:

\[(M+ Sy \; Cy) \]

(\text{Correspond} (SyMin \; CyMin) (SyMid \; CyMid) (SyMax \; CyMax))

where \(Sy\) is the position of the slider within the quantity space (SyMin, SyMid, SyMax).
Piston Pump

The circular motion of the crank results in an up and down motion of the piston. This creates alternating negative and positive relative pressures inside the pump housing. A negative pressure draws fluid in through the inlet valve. A positive pressure forces the fluid out through the outlet valve.
the position of the piston. Therefore the following constraint is added:

\[(M+ \text{Vol} \text{ Cy})\]

where: Vol is the volume within the pump housing, and
Cy is the vertical component of the crank position.\(^2\)

The model of pressure within the pump housing comes from the following law of elementary physics:

\[\text{Pressure} = \frac{\text{Mass}}{\text{Volume}}\]

Producing the qualitative constraint:

\[(\text{MULT Vol Mass Press})\]

Above Press is the absolute pressure. The pressure relative to the input and output pipes (assuming that these pressures are both equal to that of the atmosphere) is defined as follows:

\[(M+ \text{Prel Press}) (\text{Correspond (Prel 0) (Press Pair)})\]

where Pair is the constant atmospheric pressure.

\[\text{6.3 Valves and Flows}\]

The amount of mass (uncompressed fluid) within the cylinder depends upon the flow rates through the inlet and outlet valves giving the constraints:

\[(D-\text{dt Mass Net-Flow})\]
\[(\text{ADD Flow1 Flow2 Net-Flow})\]

Above Flow1 and Flow2 represent the flow-rates through Valve1 and Valve2 respectively see figure 6.1.

The two flow rate variables are constrained by the relative pressure, Prel, and the area available for flow through their respective

\[\text{2}\] Vertical according to the orientation of the pump in figure 6.1.
valves. The quantitative expression for fluid flow through an orifice is:

\[ Q = C \cdot A \cdot \sqrt{2F/p} \]  (all quantitative operators)

Where \( Q \) is the flow rate, \( C \) is the discharge constant of the orifice, \( p \) is the density of the fluid (assumed constant here), \( F \) is the pressure, and \( A \) is the area available for flow.

Therefore, when expressed qualitatively, this leads to the following DSS constraints for the two valves:

(MULT Va1 Prel Flow1)
(MULT Va2 Prel Flow2)

Here \( Va1 \) and \( Va2 \) are the areas available for flow through the valves. These areas are related to the position variables for the two valves as follows:

(M+Vx1 Va1) (Correspond (Vx1Min O))
(M+Vx2 Va2) (Correspond (Vx2Min O))

The models for the valves themselves are derived as in the final valve model in chapter 5 except that operating regions are defined at both extremes of their range of motion. The resulting constraints are:

Valve1 - the outlet valve

(D-dt Vx1 dVx1)
(COND ([(Vx1Min < Vx1 < Vx1Max)  
(M+ dVx1 Prel)])
((Vx1 = Vx1Min) or (Vx1 = Vx1Max))
(QEQ dVx1 ZERO)))

with discontinuity prediction rules:
(Vx1 (Vx1Min DEC) *) => (Vx1 (Vx1Min STD) D)
(Vx1 (Vx1Max INC) *) => (Vx1 (Vx1Max STD) D)

Valve2 - the inlet valve

(D-dt Vx2 dVx2)
(COND ((Vx2Min < Vx2 < Vx2Max)
(M- dVx2 Prel))
((Vx2 = Vx2Min) or (Vx2 = Vx2Max))
(QEQ dVx2 ZERO)))

with discontinuity prediction rules:
(Vx2 (Vx2Min DEC *) => (Vx2 (Vx2Min STD) D) (dVx2 (0 STD) D)
(Vx2 (Vx2Max INC *) => (Vx2 (Vx2Max STD) D) (dVx2 (0 STD) D)

6.4 Additional Constraints

Two additional constraints are included in the model. Both of these constraints are heuristic in nature and both depend on the assumption that the device is operating normally. The first constrains the pressure in terms of the rate of change of volume:
M- Press dVol (Correspond (Pair 0))

This constraint assumes that the valves are operating normally (or at least that one is always open). Motivation for the constraint is to force Press (and therefore Prel) to reach landmarks only when Vol reaches its landmarks. The constraint is justified by the argument that as the volume changes the pressure is perturbed faster than the flows can compensate. Furthermore, when the rate of change of the volume, dVol, is increasing (decreasing) the lag time for the flow increases (decreases) and hence the pressure changes accordingly.

The second heuristic constraint assures that the valves will always reach their landmarks at the same time. This constraint has the following form:
M- Vx1 Vx2 (Correspond (Vx1Min Vx2Max) (Vx1Max Vx2Min))

This avoids predicting four possible next states whenever both valves are approaching a landmark. Because the structure of the two valves is symmetric, and since the movement of both is caused by the same variable, Prel, the result of this constraint is intuitively acceptable. However, this constraint implies causality that does not exist, i.e. direct causation between the positions of the valves, and it would have to be revoked it is ever specified that one of the valves is stuck shut (or open), see chapter 7. It is an open research problem to develop formal methods of applying this kind of causal abstraction.

The appendix contains a complete listing of the constraints used to simulate the piston pump and an illustrated listing of the simulation output.
Chapter 7: Animation and Explanation

As has been described above, the results of a qualitative simulation is a sequence\(^{21}\) of qualitative states. The ultimate usefulness of qualitative simulation depends on how amenable this output is to interpretation and the manner in which this interpretation may be applied in performing various tasks. Potential applications include expert systems like reasoning tasks such as diagnosis and prediction. This chapter explores how readily a QSIM style model and simulation output may be interpreted and applied to the tasks of driving animation and natural language explanation of the simulated system. It seems likely that many of the strengths and and the shortcomings of the QSIM ontology that are revealed in the course of this explanation will appear in other applications as well.

7.1 Animation

It is the author's belief that the information content in a well designed graphic will usually exceed that possible in a body of text occupying the same amount of space. The value of graphics in teaching and communication in general is self evident in the ubiquity of graphs, diagrams, illustrations, photographs etc. in every medium used for these purposes. Computers are no exception. Graphics are rapidly gaining greater importance in this once teletype dominated medium. An example of this trend may be seen in the great popularity of the Macintosh computer which is largely due to its emphasis on graphics. In fact text on this machine is simply a special case of graphics. Technologies such as bitmapped displays, high band-width processors and specialized graphics hardware have enabled computers to produce graphic output, including animation, with unprecedented quality.

Graphics are especially relevant in a program designed to simulate and explain a mechanical device. Since such systems are usually quite visible and because much of their function is apparent in the motion of their components, animation is a superlative medium for their exposition.

\(^{21}\) Actually a network when multiple behaviors are found.
A simple approach to producing animation from the output of a DSS simulation is as follows:

1. Generate a single animation frame for each qualitative state. Each such frame is a bit map illustrating (at least) the positions of each discrete visible object contained in the device.

2. Display each frame, in order, rapidly enough to give the illusion of motion.

Representation of less directly visible phenomenon such as flows, forces, pressures etc. should ultimately be included in an explanatory animation along with the motion of discrete rigid objects. However, in the present discussion of animation only the later are considered. Also for now, only objects with linear motion (1 dimensional) will be discussed.

Generating Animation Frames
It is quite simple to assemble an animation frame based on a qualitative state, given the following:

- A rectangular bitmap containing the image of each discrete object (the rectangle is assumed to be of minimal size that can enclose the object).

- For the position variable associated with each object, a mapping from each element of its quantity space (landmarks and intervals) to a point in the Cartesian space defined by the bitmap representing the frame.

Figure 7.1 illustrates how two states in the simulation output for the piston pump are assembled into animation frames. Again in this example, only the objects with linear motion, namely the piston and the valves, are considered. Repeating this process creates an array of bitmaps which is isomorphic to the array of qualitative states, generated by qualitative simulation, which represent a behavior of the device.

22 Here the term frame is used in the sense of a movie frame not a data structure.
**Figure 7.1**

**Piston:** Q-Space: (Min Mid Max)
Mappings: Min $\Rightarrow (225,177)$
(Min Mid) $\Rightarrow (250,177)$
Mid $\Rightarrow (275,177)$
(Mid Max) $\Rightarrow (300,177)$
Max $\Rightarrow (325,177)$

**Valve1:** Q-Space: (Min Max)
Mappings: Min $\Rightarrow (90,420)$
(Min Max) $\Rightarrow (70,420)$
Max $\Rightarrow (50,420)$

**Valve2:** Q-Space: (Min Max)
Mappings: Min $\Rightarrow (290,420)$
(Min Max) $\Rightarrow (270,420)$
Max $\Rightarrow (250,420)$

**State1:** (Piston (Min Std))  **State2:** (Piston ((Min Mid) Inc)
(Valve1 (Min Std))  (Valve1 ((Min Max) Inc)
(Valve2 (Max Std))  (Valve2 ((Min Max) Dec)

Simple Mapping from Qualitative States to Animation Frames
Running the Animation
After generating an array of animation frames, the device behavior may be animated by displaying each bitmap, in sequence, at a fixed location on the screen. Recalling that a qualitative behavior is an alternating sequence of instants and time intervals, it seems reasonable to display the frames for alternating shorter and longer durations (e.g., 10 counts for instants and 500 for intervals).

Improving the Animation Quality
There are several problems with this simple approach to animation beyond its omission of complicating factors such as higher dimensional motion, overlapping objects, etc. These problems originate in the local translation of qualitative states into animation frames. For illustration, a slice of the simulated piston pump behavior shown in appendix B is considered. This slice includes only the piston position and the position of one valve. Figure 7.2a illustrates the relationship between the global sequence of qualitative states for the entire system, the local sequences of quantity intervals and landmarks for the two state variables, and the frames of an animation, generated as described above.

Figure 7.2a provides examples of two types of problems. First, there is no distinction between the lengths of each time interval. Therefore, the animation frame associated with each is displayed for the same amount of time. This is a problem because all events in the animation will appear to happen in equal amounts of time even though some are orders of magnitude quicker than others in reality. This will be referred to as interval duration ambiguity. The information necessary to make this kind of distinction is inherently missing in a QSIM qualitative simulation. For example, the movement of the valve from Min to Max requires one time interval, namely S2, as does the movement of the piston from Mid to Max in interval S6. In reality, the former movement will occur in much less time than the later. However, if the animation is run based on the frames shown in figure 7.2a these two movements will appear to happen in an equal elapsed time. Fortunately in this case, the inconsistency is resolved in the process of correcting another anomaly as described below. In general however, it will be necessary to impose a partial ordering over the lengths of the time intervals in a behavior. This implies that some kind of temporal information must be given, along with the simulation model and output, as input to the animation generator. An important area for future research is how to include this temporal knowledge in a qualitative simulation.

The second type of problem occurs when a variable's value remains in the same quantity interval over several contiguous time
Figure 7.2a

Qualitative States

Piston

Valve

Frames

increasing \( \bullet \bullet \bullet \); decreasing \( \bullet \bullet \bullet \); steady \( \bullet \bullet \bullet \);

Figure 7.2b

Here, each stroke of the piston requires 4 time intervals. The valve changes state in 1 interval. This animation sequence captures the qualitative relationship between the timing of the two events.
intervals; and hence several contiguous animation frames. This situation occurs when another variable of the system reaches a landmark while the first remains in a quantity interval. In QSIM, if a variable's value is an interval then its derivative is nonzero. Therefore in the case of a position variable in DSS, if its value is a quantity interval then its corresponding object is moving. However, using the simple animation algorithm outlined above, each quantity interval is mapped into a single point in the animation frames. This implies that there are some situations in which a single interval value of a position variable needs to be mapped into multiple points in successive animation frames. The problem of deciding how to do this will be called the interval mapping problem.

An example of the interval mapping problem may be seen in figure 7.2a. Observe that the transition of the piston from position Max to Mid requires two time intervals, S8 and S10, i.e. the position variable for the piston has value ((Mid Max) DEC) for three frames. This situation occurs because the valve reaches the landmark Min, generating instant S9. There are actually two anomalous consequences of this situation. First, the piston makes the transition from position Max to position Mid in two time intervals, S8 and S10 while it makes the transition from position Mid to Max in one interval, S6. In reality these transitions take the same amount of time and the animation should reflect this. This is an example of interval duration ambiguity. Second, notice that the image of the valve moves in the transition from frame F8 to F9 while the image of the piston is motionless because its state variable's value is mapped into the same bitmap position in both frames. Again, this is in qualitative disagreement with reality and the semantics of the simulation output. This is an example of the interval mapping problem.

The interval mapping problem above may be corrected by mapping the single quantity interval, (Mid Max) in the quantity space of the piston, into three positions in the animation frames. For simplicity, assume that these positions are equally spaced 1/4, 1/2, and 3/4 the distance from Mid to Max. These new positions are used in frames F10, F9, and F8, respectively. These states now display an incremental movement of the piston from Max to Mid. The interval duration problem above may be eliminated by breaking the frame F6 into 3 new frames F6', F6'',

---

23 This follows from the constraint, implicit in the model input, that the crank is rotating at a fixed angular velocity throughout the simulation, see figure 6.2.
and F6"". The positions of the piston in these frames are the reverse of those in the frames F8, F9 and F10. These states will display an incremental movement of the piston from Mid to Max. Applying analogous transformations to the frames representing the piston's movements from Min to Mid in states S2 through S4, and from Mid to Min in state S12, produces the final sequence of frames enumerated in figure 7.2b.

The transformation that was applied above was based on the knowledge that the crank moves at a constant rate and therefore that each half stroke of the piston takes the same amount of time. This amounts to using the crank as a clock. Although many mechanical devices do exhibit cyclic behavior, it may not be the case that every device one wishes to simulate will have such an embedded clock.

The issue of two and three dimensional motion has been ignored above. Two dimensional motion of an object presents no particular problem unless the image (bitmap) of the object must be rotated as it moves. The problems involved in developing general algorithms for rotating raster images in two and three dimensions are interesting but not directly related to the issue of interpreting qualitative simulation output and thus are beyond the scope of this discussion.

One pertinent issue is uncovered by considering the addition of the crank and rod (see figure 6.1) to the piston pump animation. Recall from chapter 6 that the position of the piston is constrained through an M+ relationship with the Y component of the position of the crank. For the sake of running the qualitative simulation, this level of abstraction is appropriate. However in the animation, if the length of the rod connecting the crank to the piston is to remain constant then the relationship between their position must be more precise, i.e. quantitative (see the first equation for delta s in figure 6.2). Unfortunately, this would seem to be the case for any "tightly coupled" components of a simulated mechanism.

7.2 Narration

One use for natural language in a mechanical device simulation/explanation system is narration of the device's behavior. The following example illustrates the kind of information that must be added to the standard QSIM or DSS input to allow the generation of narration.

Given that the value of the state variable Py is (Py ((PyMid PyMax) DEC)) in some state of the simulation and that the following data structures are given in the DSS model input:
(Piston-1 (PNAME ("Piston")))
  (INST (Moving-Object))
  (ORIENTATION ((DEC "downwards")
              (INC "upwards")))
  (POSITION-VAR (Py)) ... 
)

(Py (INST (Variable))
  (TYPE (Position))
  (OBJECT (Piston-1))
  (LANDMARKS (PyMin PyMid PyMax)) ... )

(PyMid (INST (Landmark))
  (Pname ("middle position")) ... )

(Py Max (INST (Landmark))
  (Pname ("uppermost position")) ... )

then the following description template:

(Tem1 (INST (Object-Motion-Description-Template))
  (FORM ("The " Pname(Object(Py)) " is between its "
            Pname(PyMid) " and its " Pname(PyMax) " and is "
            "moving " Orientation(Object(Py), DEC)"."))

may be used to generate the following sentence:

The piston is between its middle position and its uppermost position and is moving downwards.

This is a rather trivial example but demonstrates that some information that is necessary for interpreting the output of the simulation is missing in the bare input constraints and output qualitative states alone. In this case, the information that Py is the position of a moving object and that this object moves in a linear up and down motion has been added as well as the corresponding English phrases.
7.3 Question Answering

The most interesting use of natural language in the explanation system will involve various types of question answering (QA). Three classes of QA are defined here:

Classes of QA

1. QA about device structure.
2. QA about device behavior, causality and function.
3. QA about device perturbation: "What if ... ?" questions.

These classes of QA differ in the sources of information that must be tapped to formulate answers.

7.3.1 Questions about Structure

The first of these categories involves rather straightforward questions. These include queries as to the names of various device components, about the nature of connections between components, whether components are fixed or mobile, etc. The information necessary to answer this type of question will be included in the description of the device's structure which is provided in the model input.

For example, information about the piston may be represented in the following object oriented style:

Instance: (Piston-1 (Pname ("piston"))
    (INST (Moving-Object))
    (Connections (Con-1 Con-2))
    (Position-Var (Py)) ... )

Type:  (Moving-Object (ISA (Object))
    (SLOTS (Position-Var))
    (PROPERTIES ("mobile")) ... )
Instance: (Con-1 (INST (Sliding-Connection))
   (COMPONENTS (Piston-1 Cylinder-Wall-1)) ... )

Instance: (Con-2 (INST (Hinged-Connection))
   (COMPONENTS (Piston-1 Rod-1)) ... )

Type: (Hinged-Connection (ISA (Connection))
   (Properties ("hinged")) ... )

Method: (Describe-Self (METHOD-FOR (Object))
   (PARMS ())
   (TEMPLATE ("This is the " Pname.
      (send Pname Describe-Properties("It"))
      (send Pname Describe-Connections("The " Pname))
   ... )))

Method: (Describe-Properties (METHOD-FOR (Object))
   (PARMS (Name))
   (TEMPLATE (Name "is a" (enumerate Properties)
      " object." )))

Description queries may be answered by sending a Describe message to the appropriate object instance. For example:
   (send Piston-1 Describe-Self)
would produce something like the following:
   This is the piston. It is a mobile object. The piston has the following connections ...

   This object oriented style of text generation is an adaptation of that described by Thompson (17).

7.3.2 Questions about Behavior

Examples of the second QA class are:
• What is the pressure in the cylinder relative to the atmosphere?

• Why is the outlet valve closing?

Both of these questions refer to the mechanism at a particular state(s) of its operation i.e. at a particular simulated time. Answering this type of question will require information from the simulation output as well as the model input. The user will only be able to ask such questions after suspending the simulation or while running it in a halt/step mode.

Answering the first question is similar to answering questions about structure in that the answer merely describes a state of affairs. In practice, questions of this type will be answered passively by the narration and/or animation. Direct questions of this type may be handled using techniques similar to those described above for generating narration and answering queries about structure.

The second of these questions may actually be interpreted in two ways. First, it may be considered a query concerning causality i.e. 'What is causing the outlet valve to close?'. The task of deriving causality from a QSIM style model and its simulation output is discussed below. A second interpretation of this question is in the following sense: "How does the closing of the outlet valve contribute to the function of the overall device?". The answer may be that the purpose of the valve closing is to prevent fluid from flowing back in through the outlet valve during the intake stroke. The key words here are function and purpose. The problem of extracting such teleological information from the model and its behavior is also touched upon below.

**Causal Explanation**

Constraints are a non-causal representation. In demonstrating this point, Forbus (6) uses the following relationship between heat, temperature and the amount of a substance as an example.

\[
\text{Heat} = \text{Temp} \times \text{Amount}
\]

An appropriate causal interpretation of this constraint is that increasing Heat increases Temp. On the other hand, the causal arguments that increasing Heat increases Amount or that increasing Temp decreases Amount seem implausible. The problem is that the constraint contains no indication of which causal interpretation is appropriate. To resolve this causal ambiguity Forbus distinguishes the variables in a constraint as independent or dependent. In his ontology, independent variables are those that may be directly influenced by processes. In the constraint given above Heat may be influenced by heat flow and/or fluid flow;
Amount may be influenced by fluid flow. On the other hand, Temp is not
directly influenced; it is a dependent variable.

As a second example consider the following constraint from the
piston pump model:

\[ M+ dVx1 Pre1 \]

This asserts that the rate of change of the position of Valve1 and the
relative pressure within the pump are monotonically increasing func-
tions of each other. Here the problem is perhaps even more obvious.
Clearly the correct causal interpretation is that a change in the rela-
tive pressure causes a change in the velocity of the valve and not vice
versa.

Following Forbus, the causality assignment problem may be han-
dled by typing each of the state variables involved in a constraint as
either independent or dependent i.e. as an input or an output. It is
important to note that this proposed typing of the state variables with-
in constraints will require no changes in the simulation algorithm dis-
cussed in chapter 4. The utility of this distinction is found when
generating causal explanations. Consider the following constraints from
the piston pump model and the corresponding types of their variables:

1. \[ M+ dVx1 Pre1 \] - Vx1: output, Pre1: input
2. \[ M- Pre1 dVol \] - Pre1: output, dVol: input
3. \[ M+ Vol Py \] - Vol: output, Py: input
4. \[ M+ Py Cy \] - Py: output, Cy: input
5. \[ D-Dt Vx1 dVx1 \] - non-causal
6. \[ D-Dt Vol dVol \] - non-causal

where: Vx1 - horizontal position of the outlet valve
dVx1 - rate of change of the position of outlet valve
Pre1 - relative pressure in pump
dVol - rate of change of volume in pump
Py - vertical position of the piston
Vol - amount of volume in pump
Cy - vertical component of crank position

Answering the question 'Why is the outlet valve closing?' may
now proceed by back-chaining through the constraints as outlined in the
following:

Why is the outlet valve closing?
Why is the position variable for Valve-1, Vx1, decreasing?24

Why is dVx1 negative? : constraint 5

This last question inquires about the cause of the value for a variable which is an output of a causal relation (namely constraint 1) and thus may be answered as follows:

Because the relative pressure is negative.

This answer may be augmented by attaching an explanation template to the constraint so that it reads:

Because a negative relative pressure across the valve is pushing it shut.

If the explanation program is to determine whether this level of explanation is sufficient it would have to utilize a user profile or some other heuristic. These issues are beyond the scope of the present discussion. However, it is easy to show how the explanation could be continued.

Why is Prel negative?

=> Because dVol is positive. : constraint 2

Why is dVol positive?

=> Why is Vol increasing? : constraint 6

=> Because Py is increasing. : constraint 3

Why is Py increasing?

=> Because Cy is increasing. : constraint 4

The propagation of questions will end here because Cy is defined to be externally driven i.e. the crank is being turned by a motor. If there was no externally driven component of the model it appears that a circular explanation would be inevitable. However, it seems that most all mechanical devices (with the exception of a perpetual motion machine) must include some external influence; this being a battery, combustible fuel, the wind, etc.

DeKleer and Brown (4) distinguish the task of formulating causal explanations for interstate and intrastate behavior in a qualitative

---

24 Notice here, as seen previously, that information must be provided which connects the syntax of the DSS model with the semantics of the modelled pump e.g. 'outlet valve' = Valve-1, 'closing' = Position-Var(Valve-1) is decreasing, etc.
simulation. The example above explains an intrastate phenomenon. A question like 'Why did the outlet valve close?' refers to interstate behavior. To answer this question the QA program must trace backwards along the path of qualitative states leading to the current state to find the transition that caused the valve to close. An analysis like that above would be required to determine why the valve was moving in the first place. Additionally, an explanation would be required for why the movement stopped. This later explanation would require reasoning with the transition rules (possibly including the discontinuity rules) that predicted the pertinent state changes.

Explanation of Purpose

Assuming that a device is operating the way it was designed to, it seems reasonable to equate the purpose of the events in its behavior with the ramifications of those events. Thus a straightforward, albeit crude, method of answering the question 'What is the purpose of the outlet valve closing?' is to chain forward through the constraints and determine the consequences of this event.\footnote{25 To illustrate how this might work, consider the following constraints:}

\begin{verbatim}
1  M+ Val Vx1 - Val: output, Vx1: input
2  Mult Val Pre1 Flow1 - Val: input, Pre1: input, Flow1: output
\end{verbatim}

where: Val - area available for flow through outlet valve
Vx1 - horizontal position of the outlet valve
Pre1 - relative pressure inside pump
Flow1 - rate of flow out of the outlet valve

What is the purpose of the valve closing?
=> What is the consequence of Vx1 decreasing?
  => Val is decreasing. : constraint 1
  What is the consequence of Val decreasing?
  => Flow1 is decreasing. : constraint 2

25 Of course pure side effects, like the generation of heat by an electric motor, may be problematic.
To produce a truly adequate answer e.g. 'The purpose of the valve closing is to prevent fluid from flowing in through the outlet valve during the intake stroke.' would require analysis of the limit of the process of the valve closing and consideration of the global device context. In addition, knowledge of the global purpose of the device, i.e. to transfer fluid from the inlet pipe to the outlet pipe, would be required. The preceding discussion has only scratched the surface of the very difficult problem of reasoning about purpose. Much future work remains to be done in this area.

7.3.3 "What if?" Questions

One of the most effective ways to learn about the functioning of a mechanical device is to manipulate it in some way and see what happens. The following are examples of QA which involve some perturbation of the device model.

- What if the inlet valve sticks closed?
- What if the crank is rotated faster?

A simulation based explanation system at its best should be able to present the user with an environment in which he may experiment with the modelled device. This is possible because the system is not bound to a canned explanations of device structure and behavior. However, in order for such a system to work the model upon which the simulation is based must be quite robust and any assumptions contained in the model design must be represented explicitly so that proposed perturbations that violate them may be recognized.

The basic strategy for answering questions of this type would be to manipulate the model, re-run the simulation, and interpret its output. The analysis of the behavior of the altered model should include comparison to the behavior of the original model. All of this is unimplemented at the represent time and suggests a great deal of possible future work.

It is interesting to note that the second question above would be particularly troublesome to answer from a QSIM simulation. A correct analysis of this change would require representations of both relative
durations and relative magnitudes neither of which is explicit in the QSIM ontology.
Chapter 8: Conclusions and Future Work

This brief and final chapter lists the major problems that were found with using qualitative simulation as the basis of an explanation system and the future work that they suggest. Before a system like the one proposed can be used in a real application these problems must be solved and the system enhanced in several ways.

Ambiguity: The often prolific branching in the behavior predicted by qualitative simulation is a serious impediment to its practical use. Both stronger device specific constraints and general knowledge-based heuristics have been employed to curtail ambiguity. Future work should concentrate on developing formal methods for constructing accurate device models as well as generally valid device independent heuristics.

Discontinuity: The uncompromising mathematical formality of the QSIM paradigm make it quite elegant in the domain of analytic functions. However, when simulating a complex system, especially when that simulation should be amenable to human comprehension, the ability to include discontinuities is essential. The DSS system with its conditional constraints and discontinuity prediction rules is only a first attempt at what needs to become a general theory of discontinuity in qualitative simulation.

Granularity: The animation experiment described in chapter 7 revealed problems in interpreting qualitative simulation output due to its weak ontology for time. The relative durations among states are important for determining the granularity of events. In the case of the pressure activated valve, chapter 5, discontinuous prediction was used to eliminate events that exist at a level of granularity below that being considered. Information must be included in the model that allows the simulation to differentiate the orders of magnitude of duration for the qualitative states.

Causality: The proposed algorithm for generating causal explanations described in chapter 7 indicates the lack of causal information in a constraint based model. The method of tagging variables as either dependent or independent seems promising but much future work will be required before this problem can be considered solved for the general case.
Performance: As larger more complex systems are modelled using QSIM type simulation performance will become a critical issue. The depth first search following Waltz filtering, as described in chapter 4, has exponential time complexity in the worst case. The path-consistency preprocessing algorithm of Mackworth, cited in chapter 4, may provide a significant speed up for large systems. An additional avenue for future is the investigating the feasibility of decomposing a large system into a set of smaller subsystems which may be simulated in a semi-independent manner (i.e. attack complexity with divide and conquer).

User Interface: The animation and natural language interface described in chapter 7 is for the most part just a proposal at this time. Work is currently under way to implement a prototype system of this kind.

Although the problems enumerated above are certainly hard ones, it is the author's belief that they are solvable. Furthermore, it may be concluded from the experiment documented in this thesis that, assuming such solutions are found, qualitative simulation will provide an excellent nucleus for building a computerized explanation system for mechanical devices.
Appendix: Piston Pump Model and Simulation Output

A.1 Piston Pump Model Variables

- $C_y$ - vertical component of crank position
- $C_x$ - horizontal component of crank position
- $dC_y$ - derivative of $C_y$
- $dC_x$ - derivative of $C_x$
- $\text{Press}$ - absolute pressure inside pump
- $\text{Pres}_1$ - relative pressure inside pump
- $V_1$ - amount of volume inside pump
- $dV_1$ - derivative of $V_1$
- $M_1$ - amount of fluid inside pump
- $dM_1$ - derivative of $M_1$
- $\text{Flow}_1$ - flow into pump through outlet valve
- $\text{Flow}_2$ - flow into pump through inlet valve
- $V_x_1$ - horizontal position of outlet valve
- $dV_x_1$ - derivative of $V_x_1$
- $V_{a1}$ - area available for flow through outlet valve
- $V_x_2$ - horizontal position of inlet valve
- $dV_x_2$ - derivative of $V_x_2$
• Va2 - area available for flow through inlet valve

A.2 Piston Pump Model Constraints

1. \( (D-Dt \: C_y \: dC_y) \)
2. \( (D-Dt \: C_x \: dC_x) \)
3. \( (M+ \: C_x \: dC_y) \)
   Correspond \( (C_xMid \: 0) \)
4. \( (M- \: C_y \: dC_x) \)
   Correspond \( (C_yMid \: 0) \)
5. \( (M+ \: V_o1 \: C_y) \)
   Correspond \( (V_o1Min \: C_yMin) \) \( (V_o1Mid \: C_yMid) \) \( (V_o1Max \: C_yMax) \)
6. \( (M+ \: dV_o1 \: dC_y) \)
   Correspond \( (0 \: 0) \)
7. \( (M ult \: P ress \: V_o1 \: M ass) \)
8. \( (D-Dt \: M ass \: dM ass) \)
9. \( (M+ \: P rel \: P ress) \)
   Correspond \( (0 \: P air) \)
10. \( (A dd \: F l ow1 \: F l ow2 \: dM ass) \)
11. \( (M ult \: -P rel \: V a1 \: F l ow1) \)
12. \( (M ult \: -P rel \: V a2 \: F l ow2) \)
13. \( (D-Dt \: V x1 \: dV 1) \)
14. \( (D-Dt \: V x2 \: dV x2) \)
15. If \( (V x1 = (V x1Min \: S TD) \ and \ P rel < 0 \) \) or
    \( (V x1 = (V x1Max \: S TD) \ and \ P rel > 0 \) \)
then (QEQ dVx1 'O STD))
else (M+ dVx1 Pre1)
Correspond (O O)
with Discontinuity Prediction Rules
(Vx1 (Vx1Min DEC)) => (Vx1 (Vx1Min STD) D) (dVx1 (O STD) D)
(Vx1 (Vx1Max INC)) => (Vx1 (Vx1Max STD) D) (dVx1 (O STD) D)

16. If ( Vx2 = (Vx2Min STD) and Pre1 > 0 ) or
    ( Vx2 = (Vx2Max STD) and Pre1 < 0 )
    then (QEQ dVx2 'O STD))
    else (M- dVx2 Pre1)
Correspond (O O)
with Discontinuity Prediction Rules
(Vx2 (Vx2Min DEC)) => (Vx2 (Vx2Min STD) D) (dVx2 (O STD) D)
(Vx2 (Vx2Max INC)) => (Vx2 (Vx2Max STD) D) (dVx2 (O STD) D)

17. If Vx1 > Vx1Min
    then (M+ Vx1 Va1)
    else (QEQ Va1 'O STD))
Correspond (Vx1Min O) (Vx1Max Va1Max)

18. If Vx2 > Vx2Min
    then (M+ Vx2 Va2)
    else (QEQ Va2 'O STD))
Correspond (Vx2Min O) (Vx2Max Va2Max)

19. D-Dt Vol dVol

20. If VolMin < Vol < VolMax
    then (M- Press dVol) (Correspond (Pair O))
    else (QEQ Press 'Pair STD))

21. (M- Vx1 Vx2)
    Correspond (Vx1Min Vx2Min) (Vx1Max Vx2Min)
A.3 Selected Piston Pump Behavior Path

PP001
(CY ((CYMIN CYMID) INC) 10)
(CX ((CXMIN CXMAX) INC) 10)
(DCY ((0 INF) INC) 10)
(DCX ((0 INF) DEC) 10)
(DMASS ((0 INF) INC) 10)
(DVOL ((0 INF) INC) 10)
(DVX1 ((MINF 0) DEC) 10)
(DVX2 ((0 INF) INC) 10)
(FLOW1 ((0 INF) DEC) 10)
(FLOW2 ((0 INF) INC) 10)
(MASS ((0 INF) INC) 10)
(PREL ((MINF 0) DEC) 10)
(PRESS ((0 PAIR) DEC) 10)
(VA1 ((0 VA1MAX) DEC) 10)
(VA2 ((0 VA2MAX) INC) 10)
(VOL ((VOLMIN VOLMID) INC) 10)
(VX1 ((VX1MIN VX1MAX) DEC) 10)
(VX2 ((VX2MIN VX2MAX) INC) 10)
ITYPE

GO022
(CX ((CXMIN CXMAX) INC) 14)
(CY ((CYMIN CYMID) INC) 14)
(DCX ((0 INF) DEC) 17)
(DCY ((0 INF) INC) 14)
(DMASS ((0 INF) INC) 14)
(DVOL ((0 INF) INC) 14)
(DVX1 ((MINF 0) DEC) 17)
(DVX2 ((0 INF) INC) 14)
(FLOW1 (0 STD) 15)
(FLOW2 ((0 INF) INC) 14)
(MASS ((0 INF) INC) 14)
(PREL ((MINF 0) DEC) 17)
(PRESS ((0 PAIR) DEC) 17)
(VA1 (0 STD) 15)
(VA2 (VA2MAX INC) 13)
(VOL ((VOLMIN VOLMID) INC) 14)
(VX1 (VX1MIN DEC) 16)
(VX2 (VX2MAX INC) 13)
PTYPE
G0024
(CX ((CXMIN CXMAX) INC) P5)
(CY ((CYMIN CYMAX) INC) P5)
(DCX ((0 INF) DEC) P7)
(DCY ((0 INF) INC) P5)
(DMASS ((0 INF) INC) P5)
(DVOL ((0 INF) INC) P5)
(DVX1 (0 STD) D)
(DVX2 (0 STD) D)
(FLOW1 (0 STD) P1)
(FLOW2 ((0 INF) INC) P5)
(MASS ((0 INF) INC) P5)
(PREL ((0 MIN) DEC) P7)
(PRESS ((0 PAIR) DEC) P7)
(VA1 (0 STD) P1)
(VA2 (VA2MAX STD) D)
(VOL ((VOLMIN VOLMAX) INC) P5)
(VX1 (VX1MIN STD) D)
(VX2 (VX2MAX STD) D)
ITYPE

G0037
(CX (CXMAX STD) I2)
(CY (CYMIN INC) I3)
(DCX (0 DEC) I6)
(DCY (*DCY0029 STD) I8)
(DMASS (*DMASS0030 STD) I8)
(DVOL (*DVOL0031 STD) I8)
(DVX1 (0 STD) I1)
(DVX2 (0 STD) I1)
(FLOW1 (0 STD) I1)
(FLOW2 (*FLOW20032 STD) I8)
(MASS ((0 INF) INC) I4)
(PREL (*PREL0034 STD) I9)
(PRESS (*PRESS0035 STD) I9)
(VA1 (0 STD) I1)
(VA2 (VA2MAX STD) I1)
(VOL (VOLMIN INC) I3)
(VX1 (VX1MIN STD) I1)
(VX2 (VX2MAX STD) I1)
PTYPE

Midway through intake stroke.
Prel reaches global minimum
Flow2 reaches a minimum (negative) as Valve2 closes.
GO077
(CX ((CXMIN CXMIN) DEC) P7)
(CY ((CYMIN CYMIN) DEC) P7)
(DCX ((#DCX0043 O) INC) P5)
(DCY ((MINF O) DEC) P7)
(DMASS ((MINF O) DEC) P7)
(DVOL ((MINF O) DEC) P7)
(DVX1 ((O INF) INC) P5)
(DVX2 ((MINF O) DEC) P7)
(FLOW1 ((MINF O) DEC) P7)
(FLOW2 ((#FLOW0065 O) INC) P2)
(MASS ((#0 #MASS0048) DEC) P7)
(PREL ((O INF) INC) P5)
(PRESS ((PAIR INF) INC) P5)
(VA1 ((O VA1MAX) INC) P5)
(VA2 ((O VA2MAX) DEC) P7)
(VOL ((VOLMIN VOLMAX) DEC) P7)
(VX1 ((VX1MIN VX1MAX) INC) P5)
(VX2 ((VX2MIN VX2MAX) DEC) P7)
ITYPE

GO100
(CX ((CXMIN CXMIN) DEC) I7)
(CY ((CYMIN CYMIN) DEC) I7)
(DCX ((#DCX0043 O) INC) I4)
(DCY ((MINF O) DEC) I7)
(DMASS ((MINF O) DEC) I7)
(DVOL ((MINF O) DEC) I7)
(DVX1 ((O INF) INC) I4)
(DVX2 ((MINF O) DEC) I7)
(FLOW1 ((MINF O) DEC) I7)
(FLOW2 (O STD) I2)
(MASS ((O #MASS0048) DEC) I7)
(PREL ((O INF) INC) I4)
(PRESS ((PAIR INF) INC) I4)
(VA1 (VA1MAX INC) I3)
(VA2 (O STD) I5)
(VOL ((VOLMIN VOLMAX) DEC) I7)
(VX1 (VX1MAX INC) I3)
(VX2 (VX2MIN DEC) I6)
PTYPE
G0102
(CX ((CXMIN CX1MID) DEC) P7)
(CY ((CYMID CYMAX) DEC) P7)
(DCX ((*DCX0043 O) INC) P5)
(DCY ((MINF O) DEC) P7)
(DMASS ((MINF O) DEC) P7)
(DVOL ((MINF O) DEC) P7)
(DVX1 (O STD) D)
(DVX2 (O STD) D)
(FLOW1 ((MINF O) DEC) P7)
(FLOW2 (O STD) P1)
(MASS ((O *MASS0048) DEC) P7)
(PREL ((O INF) INC) P5)
(PRESS ((PAIR INF) INC) P5)
(VA1 (VA1MAX STD) D)
(VA2 (O STD) P1)
(VOL ((VOLMID VOLMAX) DEC) P7)
(VX1 (VX1MAX STD) D)
(VX2 (VX2MIN STD) D)
ITYPE

G0115
(CX (CXMIN STD) I5)
(CY (CYMID DEC) I6)
(DCX (O INC) I3)
(DCY (*DCY0107 STD) I9)
(DMASS (*DMASS0108 STD) I9)
(DVOL (*DVOL0109 STD) I9)
(DVX1 (O STD) I1)
(DVX2 (O STD) I1)
(FLOW1 (*FLOW10110 STD) I9)
(FLOW2 (O STD) I1)
(MASS ((O *MASS0048) DEC) I7)
(PREL (*PRELO112 STD) I8)
(PRESS (*PRESS0113 STD) I8)
(VA1 (VA1MAX STD) I1)
(VA2 (O STD) I1)
(VOL (VOLMID DEC) I6)
(VX1 (VX1MAX STD) I1)
(VX2 (VX2MIN STD) I1)
PTYPE

Piston is midway through outlet stroke.
Prei is at a global maximum.
GO117
(CX (CXMIN CXMIN) INC) P2
(CY (CYMIN CYMIN) DEC) P6
(DCX ((DINF DINF) INC) P4)
(DCY ((DCY010 DCY010) INC) P2)
(DMASS ((DMASS0108 DMAS0108) INC) P2)
(DVOL ((DVOL0109 DVOL0109) INC) P2)
(DVX1 (DSTD DSTD) P1)
(DVX2 (DSTD DSTD) P1)
(FLOW1 ((FLOW10110 FLOW10110) INC) P2)
(FLOW2 (DSTD DSTD) P1)
(MASS ((0 *MASS048 DEC) DEC) P7)
(PREL ((0 *PREL0112 DEC) DEC) P3)
(PRESS ((PAIR *PRESS0113 DEC) DEC) P3)
(VA1 (VA1MAX STD) P1)
(VA2 (DSTD DSTD) P1)
(VOL ((VOLMIN VOLMIN) DEC) P6)
(VX1 (VX1MAX STD) P1)
(VX2 (VX2MIN STD) P1)

GO130
(CX (CXMIN INC) I3)
(CY (CYMIN STD) I5)
(DCX (*DCX0121 STD) I8)
(DCY (DINC DINC) I3)
(DMASS (DSTD DSTD) I2)
(DVOL (DINC DINC) I3)
(DVX1 (DSTD DSTD) I1)
(DVX2 (DSTD DSTD) I1)
(FLOW1 (DSTD DSTD) I2)
(FLOW2 (DSTD DSTD) I2)
(MASS (*MASS0126 STD) I9)
(PREL (DSTD DSTD) I5)
(PRESS (PAIR STD) I5)
(VA1 (VA1MAX STD) I1)
(VA2 (DSTD DSTD) I1)
(VOL (VOLMIN STD) I5)
(VX1 (VX1MAX STD) I1)
(VX2 (VX2MIN STD) I1)

Flow1 = 0
Prel = 0
G0132
(CG CX (CXMIN CXMAX) (INC) P4)
(CY (CYMIN CYMID) (INC) P2)
(DCX ((O *DCX0121) (DEC) P3)
(DCY ((O *DCY0029) (INC) P4)
(DMASS ((O *DMASS0030) (INC) P2)
(DVOL ((O *DVOL0031) (INC) P4)
(DVX1 ((MINF O) (DEC) P3)
(DVX2 ((O INF) (INC) P2)
(FLOW1 ((O INF) (INC) P2)
(FLOW2 ((O *FLOW0032) (INC) P2)
(MASS ((O *MASS0126 *MASS0048) (INC) P2)
(PREL ((O *PRELO0034) (DEC) P3)
(PRESS ((O *PRESS0035 PAIR) (DEC) P3)
(VA1 ((O VA1MAX) (DEC) P3)
(VA2 ((O VA2MAX) (INC) P2)
(VOL ((VOLMIN VOLMID) (INC) P2)
(VX1 ((VX1MIN VX1MAX) (DEC) P3)
(VX2 ((VX2MIN VX2MAX) (INC) P2)
IETYPE

G0153
(CG CX (CXMIN CXMAX) (INC) I4)
(CY (CYMIN CYMID) (INC) I4)
(DCX ((O *DCX0121) (DEC) I7)
(DCY ((O *DCY0029) (INC) I4)
(DMASS ((O *DMASS0030) (INC) I4)
(DVOL ((O *DVOL0031) (INC) I4)
(DVX1 ((MINF O) (DEC) I7)
(DVX2 ((O INF) (INC) I4)
(FLOW1 (*FLOW10142 STD) I8)
(FLOW2 ((O *FLOW0032) (INC) I4)
(MASS ((O *MASS0126 *MASS0048) (INC) I4)
(PREL ((O *PRELO0034 (DEC) I7)
(PRESS ((O *PRESS0035 PAIR) (DEC) I7)
(VA1 ((O VA1MAX) (DEC) I7)
(VA2 ((O VA2MAX) (INC) I4)
(VOL ((VOLMIN VOLMID) (INC) I4)
(VX1 ((VX1MIN VX1MAX) (DEC) I7)
(VX2 ((VX2MIN VX2MAX) (INC) I4)
PCTYPE

Flow 1 reaches a maximum as Valve 1 closes
A.4 Ambiguity in the Piston Pump Simulation

The ambiguity at G0075 and G0077 concerns Flow2 as Valve2 closes.

G0076: is identical to G0075.
G0078: Flow2 begins to decrease again after reaching minimum in G0075.
G0099: Flow2 reaches a negative point of inflection after increasing from the minimum reached in G0075.
G0101: Dmass reaches a spurious minimum.

The ambiguity at G0153 and PP001 is caused by a symmetric situation involving Flow1 and Valve1.

The ambiguity at G0132 is caused by previously discovered landmarks for the three variables Flow2, Dmass, and Mass. 15 next states are generated reflecting all combinations of the following 4 events.

1) Flow1 reaches its MAX. (the desired event)
2) Flow2 exceeds its previous MAX.
3) Dmass exceeds its previous MAX.
4) Mass exceeds its previous MAX.

G0170 completes the cycle with PP001.
Bibliography


