QUALITATIVE REASONING ABOUT FLUIDS AND MECHANICS

Hyeonkyeong Kim

November 1993

The Institute for the Learning Sciences
Northwestern University
Evanston, IL 60201

This research was supported by the Office of Naval Research under contract N00014-85-K-0225. The Institute for the Learning Sciences was established in 1989 with the support of Andersen Consulting. The Institute receives additional support from Ameritech and North West Water, Institute Partners.

QUALITATIVE REASONING ABOUT FLUIDS AND MECHANICS

BY

HYEONKYEONG KIM

B.H.E., Seoul National University, 1982
M.S., Seoul National University, 1984

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1993

Urbana, Illinois

QUALITATIVE REASONING ABOUT FLUIDS AND MECHANICS

Hyeonkyeong Kim, Ph.D.
Department of Computer Science
University of Illinois at Urbana-Champaign, 1993
Kenneth D. Forbus, Advisor

Understanding people's commonsense knowledge about physical world is a fundamental problem in building intelligent systems. If this knowledge can be represented and used by computers, they can duplicate people's ability to understand and interact with the world. Qualitative physics is the attempt to capture and formalize this knowledge. An important aspect of qualitative reasoning is the ability to derive the possible behaviors of a given physical system from the structure of the system, using minimal initial information.

This thesis investigates qualitative domain theories and reasoning techniques which will enable computers to analyze the qualitative behaviors of physical systems which include both mechanical mechanisms and fluids, such as internal combustion engines and hydraulic lift pumps. We have developed a domain theory which integrates richer models of mechanics, fluids, and geometry than previous research in qualitative physics. These theories and inference techniques are embodied in `QSA`, a program that produces possible behaviors of physical systems.

To My Parents

ACKNOWLEDGMENTS

I would like to thank:

- Professor Kenneth Forbus, my advisor, for initiating and directing this research, for stimulating me to think creatively and to stand on my own legs, and for giving me a great example of researcher.
- Professors Gerald DeJong, Seth Hutchinson, Jean Ponce, and Marianne Winslett for serving on my final committee and providing helpful comments.
- Past and present members of the Qualitative Reasoning Group at Beckman Institute for useful discussions and for being friends. Especially to:
  - Janice and Gordon Skorstad for comments and discussions about modeling of thermodynamics, for stimulating me to hack Macintosh, and for providing me a warm friendship.
  - John Collins for comments and discussions about modeling of mechanical mechanisms and `QPE`.
  - Dennis DeCoste for discussions about many interesting problems in qualitative physics.
  - Paul Nielsen for discussions in early stage of this work.
- Professor Kyekyoon Kim for his encouragement and personal support.
- My parents, my sisters, my brother, and especially my husband Seungdo for their love, support, and encouragement throughout the years.

This research was supported by the Office of Naval Research, Contract No. N00014-85-K-0225.

TABLE OF CONTENTS

Chapter

Chapter 1

Introduction

Understanding people's commonsense knowledge about physical world is a fundamental problem in building intelligent systems. If this knowledge can be represented and used by computers, they can duplicate people's ability to understand and interact with the world. Qualitative physics is the attempt to capture and formalize this knowledge. It has developed qualitative representations of the world and inference mechanisms based on these representations.

An important aspect of qualitative reasoning is the ability to derive the possible behaviors of a given physical system from the structure of the system, using minimal initial information. The structure of the system is described in terms of components and the relations between them. The behavior is described by the changes of the components and the system as a whole over time (Bobrow, 1985).

The goal of this work is to develop qualitative domain theories and reasoning techniques which will enable computers to analyze the qualitative behaviors of physical systems which include both mechanical mechanisms and fluids, such as internal combustion engines and hydraulic lift pumps (Figure 1.1). In terms of representations, this requires capturing the interaction between dynamics and geometry. In terms of reasoning, this requires developing inference techniques for using this knowledge for our reasoning tasks. Our domain theories represent fluids, rigid objects, and the interactions between them. Our reasoning techniques include methods for handling angular changes, discontinuous changes, and global properties of behavior.

The kind of analysis we are interested in is exemplified by the CLOCK system (Faltings, 1986; Nielsen, 1988; Forbus et al., 1991). CLOCK derives the behaviors of mechanical mechanisms, composed of gears and escapements, by using spatial reasoning techniques. CLOCK assumes as input a set of objects and the external forces applied to them. The configuration space representations for each pair of objects are constructed by Faltings' program. Once the possible configurations of the system are found by analyzing the geometric constraints, Nielsen's program predicts possible motions in each configuration by analyzing the constraints arising from the surface contacts and forces acting on the system. The result of the analysis is the set of qualitative states and state transitions which explain the possible behaviors of the given system. A qualitative state consists of the union of kinematic state and dynamic state; a kine-

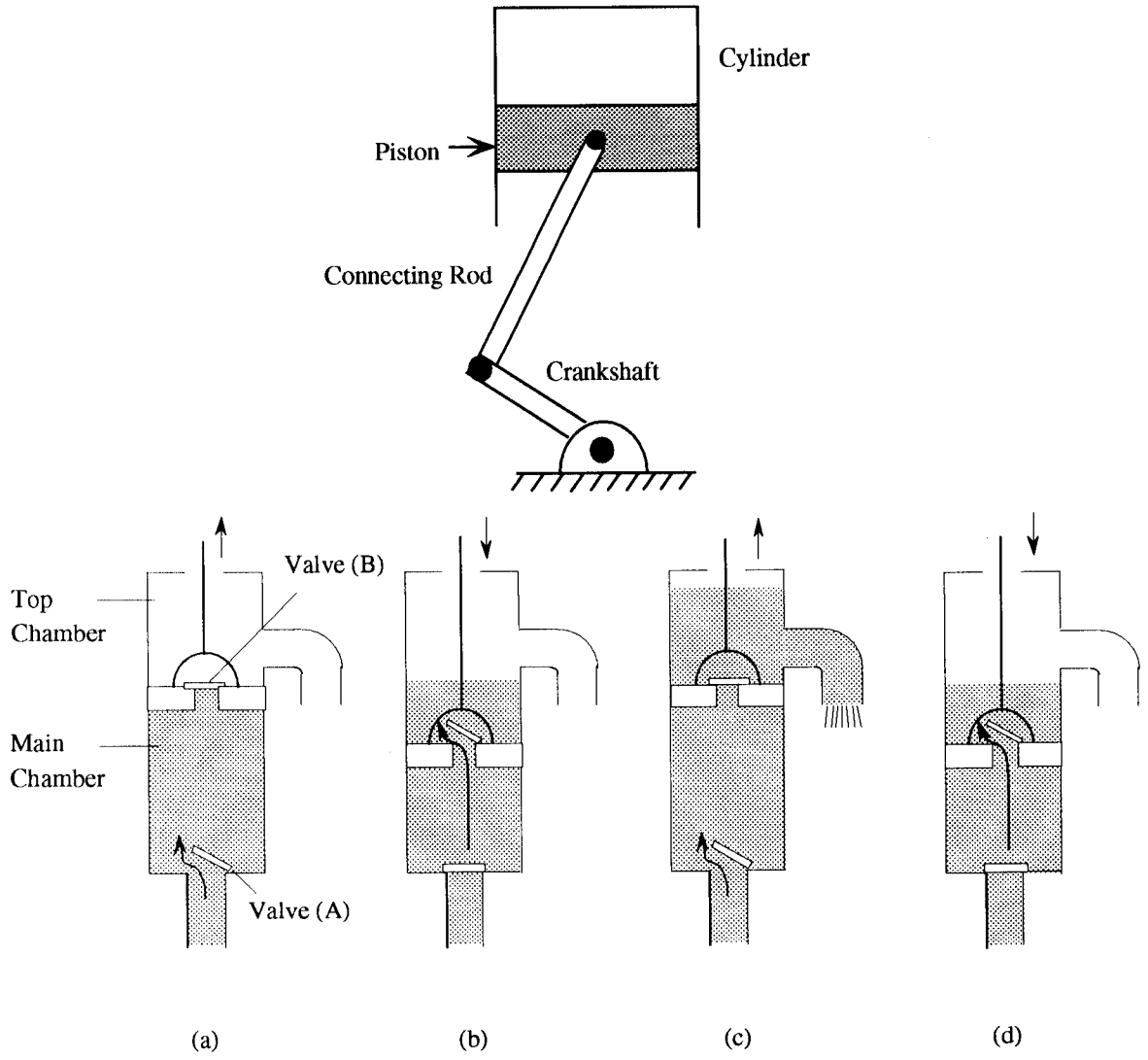


Figure 1.1: Some physical systems analyzed in this thesis. The top figure shows a simplified internal combustion engine. The bottom half illustrates the operation of a lift pump.

matic state represents a particular configuration of the system while a dynamic state describes the motions of the system. CLOCK has been successfully tested on several examples, including mechanical clocks.

However, CLOCK, like many other studies dealing with mechanical mechanisms, suffers from several important limitations: (1) It focuses on fixed axis mechanisms, and thus it is unable to analyze many important systems. For example, it cannot analyze the slider-crank mechanism which transmits the vertical motion of the piston to the rotation of the crankshaft in an automobile. (2) It focuses on only geometric interactions between rigid objects, ignoring fluid. (3) Since it only considers motions and gravity, other physical phenomena, such as the combustion occurring in internal combustion engines, cannot be analyzed. (4) The configuration space construction is inherently intractable for general mechanisms.

There has been much research on representing fluids in qualitative physics (Hayes, 1985; Collins & Forbus, 1987; Amador & Weld, 1990; Rajamoney & Koo, 1990; Skorstad, 1992). Unfortunately, this work has been insufficient to fully explain fluid behavior. For example, none of this work suffices to explain the operation of the lift pump in Figure 1.1, which requires the analysis of the geometric interaction between fluids and rigid objects.Geometry was rarely considered in previous research. Consequently, these representations are not adequate for situations where the geometric interactions between objects, both solid and fluid, play a key role in understanding the behavior of a system.

This chapter begins by briefly explaining why qualitative physics has been chosen for our analysis. Then our domain theories and inference mechanisms are described in the context of three major research areas of qualitative physics—qualitative dynamics, qualitative kinematics, and styles of reasoning, especially qualitative simulation (Section 1.2, Section 1.3, and Section 1.3 respectively). Finally, Section 1.6 provides an overview of the organization of this thesis.

1.1 Why Qualitative Physics?

The goal of qualitative physics is to develop theories that characterize human understanding of physical world in order to explain a broad class of physical phenomena as well as people can. It requires one to formalize human knowledge about the world and to develop inference techniques to use this knowledge. For hundreds of years, physicists have attempted to build models that can describe and predict the physical world as precisely as possible. While both qualitative and traditional physics are concerned about understanding physical phenomena, they differ in several ways.

The models of traditional physics basically consist of a set of mathematical equations. Using theories expressed in equations requires either analytic solution or numerical simulation. While this approach can often provide precise predictions, it has some limitations with regard to enabling computers to make qualitative judgments as people do. First, it requires more information than people have in everyday situations. Detailed information is not always available. In addition, precise predictions are often not even needed: sometimes quick and approximate predictions are preferable. For instance, we do not want our household robots to spend hours

solving partial differential equations to make a pot of coffee. Furthermore, many computers using traditional physics still requires qualitative physics (de Kleer, 1975; Skorstad & Forbus, 1990). The informal qualitative knowledge of physicists that determines what equations are appropriate for a situation, and how to interpret numerical results, requires formalization..

Traditional physics is not the only way with which people understand everyday physical phenomena. For instance, an average person, who might know little about physics, rarely has problems in dealing with everyday physical occurrences. Qualitative models, which capture the reasoning processes of an average person, describe the world by mapping the continuous world into a discrete representation for symbolic reasoning. This *quantization* of detailed information is crucial to support many reasoning tasks. A qualitative model typically loses precision, and thus leads to ambiguous predictions. However, generating solutions requires much less information. Furthermore, when complete numerical information is not available, qualitative analysis is the only available method. While ambiguity can lead to extra computation time, it allows one to explicitly find possible alternatives. Lastly, it is easier to understand the analyses performed by qualitative models than by numerical methods since qualitative models concentrate on fundamental processes while suppressing unnecessary details.

1.2 Qualitative Dynamics

Qualitative physics is subdivided into dynamics and kinematics just as traditional physics is. While qualitative dynamics is concerned with the changes that can be described by ordinary differential equations, qualitative kinematics deals with the spatial reasoning aspects of qualitative physics.

In this work, *Qualitative Process* (QP) theory (Forbus, 1984) is used as the primary framework for qualitative dynamics. Unfortunately, like most accounts of qualitative dynamics, QP theory focuses only on continuous changes and ignores discontinuous changes. We investigate how the technique of limit analysis can be extended to discontinuous models. We show this technique by introducing discrete processes in the framework of QP theory. The next section provides an introduction to QP theory for the convenience of the reader.

1.2.1 Qualitative Process Theory

A key tenet of QP theory is that *processes* are central to human understanding of the physical world; all physical changes are caused by processes. Examples of processes are liquid flow, motion, acceleration, boiling, compression, and expansion. In QP theory, a domain theory consists of a set of objects, a set of relationships between them, and a set of processes that provide mechanisms for changes. All objects are described by their quantities, such as `pressure` and `temperature`. The value of a quantity is described by its *quantity space*. A quantity space represents a continuous parameter by a set of comparisons with other relevant parameters. This quantization is useful to capture changes since processes start and stop when ordinal relationships between quantities change. Changes caused by processes are captured by the changes in particular inequality relations. Suppose flow of a liquid occurs between two containers. As long

```
(defProcess (Liquid-Flow ?src ?dst ?path)
  Individuals
    (?src  :type contained-liquid)
    (?dst  :type contained-liquid)
    (?path :type fluid-path
           :conditions (fluid-connections ?path ?src ?dst))
  Preconditions
    (Aligned ?path)
  QuantityConditions
    (Greater-than (pressure ?src) (pressure ?dst))
  Relations
    (Quantity flow-rate)
    (Q= flow-rate (- (pressure ?src) (pressure ?dst)))
  Influences
    (I- (amount-of ?src) flow-rate)
    (I+ (amount-of ?dst) flow-rate) )
```

Figure 1.2: Process Definition for simple Liquid Flow

as there is difference in their levels, liquid flow will continue. It will stop when their levels reach the same height.

A process is defined by five parts: *individuals*, *preconditions*, *quantity conditions*, *relations*, and *influences*. Figure 1.2 shows a simple liquid flow process. Whenever all individuals in the individuals field exist and all conditions in the preconditions and quantity conditions are satisfied, the statements in the relations and influences fields hold. Quantity conditions include ordinal relations between quantities which can be predicted by physics. Preconditions describe the conditions outside physics, for instance, that two containers are connected by an unblocked path (e.g., the aligned statement in Figure 1.2). The relations describe what will be true while the process is active. The direct effects of the process are specified in the influences. To describe direct influences, $I+$ and $I-$ are used. They contain the derivative of their first argument. The total derivative is computed by combining them, using closed-world assumptions made over the set of influences. While the process is active, its flow rate influences the source and the destination of the flow. The flow rate negatively influences the amount of the liquid at the source while it positively influences the amount of the liquid at the destination.

QP theory provides another type of conditionalized description besides process: the *view*. Views are used to capture various states of objects and dynamic relationships. For instance, the existence of a contained-liquid in a container can be described in a view: it exists if the amount of the liquid is greater than zero. Otherwise it does not exist. Views are defined in the same manner as processes but with no influences field since only processes can have direct influences.

The Qualitative Process Engine (QPE) is an envisioner which uses QP theory to provide possible behaviors (Forbus, 1986). Given a scenario, i.e., a particular situation being modeled, QPE automatically instantiates the domain theory to form a scenario model. From the scenario model all possible behaviors are derived. Besides simulation, QPE has been used for various kinds of reasoning such as measurement interpretation (DeCoste, 1992) and planning (Hogge, 1987; Forbus, 1989).

1.2.2 Discontinuous Changes

Dynamics can be divided into the study of continuous changes and discontinuous changes. Continuous changes occur gradually, while discontinuous changes occur abruptly.

Reasoning about discontinuous change has mostly been ignored in qualitative physics. Discontinuity has been studied in constrained domains such as electronic circuits (Nishida & Doshita, 1987) and cyclic behavior (Weld, 1984). There has been attempt to formalize discontinuity in motion (Forbus, 1980, 1981) and using logical formula (Sandwell, 1989; Rayner, 1991) and to adapt the approach in planning for reasoning about discontinuity (Dean & Siegle, 1990).

The alternatives for capturing discontinuity are: (1) model a discontinuous change as rapid continuous change and (2) model a discontinuous change as an abrupt change. The first approach might be needed for reasoning tasks which require a microscopic view. This can be done by an existing continuous qualitative simulator with infinitesimal calculus (Weld, 1988; Davis, 1987). On the other hand, in many cases the second approach—to understand discontinuity as what it is—is more intuitive and simplifies reasoning. The techniques developed for actions might be adopted for this. The simplest such approach is to model discontinuous change using the STRIPS representation: a discontinuous change is represented by specifying the state before change and the state after change, using add and delete lists of a STRIPS operator (Fikes, 1972; Weld, 1985, 1986; Forbus, 1989). This method is based on the similarity of action and discontinuous changes: both can lead to new discontinuous states in an instant, if an action is assumed to occur in an instant.

The STRIPS representation is inadequate for reasoning about discontinuous changes. Unlike continuous changes, which are reasoned about by influence resolution and limit analysis during simulation, discontinuous changes are postulated rather than inferred during simulation. For instance, it is inappropriate to capture the combustion in internal combustion engines via add and delete lists. It is difficult to postulate all possible pairs of add and delete lists for combustions in all possible situations. This will be shown in Chapter 6.

In this work, discontinuous changes are expressed as influences like continuous changes. Discontinuous changes are predicted from the current values of quantities and the effects of the change in each state. We show how limit analysis of continuous models is extended to discrete models in the framework of QP theory. A discrete process is defined by the necessary conditions and effects in the same way that continuous processes are defined. A discontinuous change is the result of a discrete process in the same way that a continuous change is the result of a continuous processes.

1.3 Qualitative Kinematics

Another important issue in qualitative physics is to understand how dynamics and geometry interact. This is crucial to reasoning about many common systems. Dynamics alone cannot fully explain the physical world. For example, applying the same force to different points on an object can cause dramatically different behaviors. Without geometric information, these distinct behaviors would be impossible to predict.

The second primary research topic of the present work is to investigate the interaction between qualitative spatial reasoning and dynamics. In order for an object to affect another, there must be some kind of contact between them. In this work, we deal with only the kinematic interaction which occurs by surface contact. We shall not inquire into field interactions, such as between an electron and a large uniformly charged sphere or the interaction of two bar magnets.

The first task in the analysis of kinematic interaction is to partition space at an appropriate level of detail. Just as quantities are partitioned into distinct intervals, space and shape must also be broken up into qualitatively distinct parts. Since kinematic interactions between objects depend on how they are in contact, qualitative distinctions must be based on configurations of contact. How to find an appropriate level of representation for space is a crucial problem in qualitative kinematics. The notion of *place* was introduced by Forbus (1980) to refer to qualitatively equivalent points of space. Space is divided into places and the behaviors in any point within a place must be qualitatively homogeneous. The criteria for quantization depends critically on the nature of reasoning task.

This work explores several problems related to geometric reasoning in two dimensional space: (1) an extension of the qualitative vector representation used in (Nielsen, 1988) with angle, (2) a qualitative analysis of motion for mechanical linkages, (3) the introduction of *bounded-stuff* ontology for reasoning about the behaviors of fluid with complex surface geometry, and (4) the analysis of the direction of fluid flow.

1.3.1 Mechanical Mechanisms

The general spatial reasoning problem is too unconstrained to be analyzed in general. Thus, research about rigid objects has focused on more constrained problems such as motion in limited domains (de Kleer, 1975; Forbus, 1981) and mechanical mechanisms (Faltings, 1986; Gelsey, 1987; Joskowicz, 1987; Nielsen, 1988).

We use the definition of a mechanism given by (Cowie, 1961): "an assemblage of resistant and *relatively constrained* parts connected together to transmit and modify force and motion". This work focuses on the physical systems whose rigid parts constitute mechanism(s). Here the term 'physical system' refers to any device consisting of rigid parts or fluids. For example, on an automobile engine, the pistons, connecting rods, crankshaft, and cylinder block constitute a mechanism that converts the reciprocating motion of the pistons to the rotary motion of the crankshaft. This mechanism and gases inside the cylinders make the engine do its intended work.

We focus on mechanical mechanisms for several reasons:

- The behaviors of mechanisms are in general highly constrained, yet they exhibit interesting and complex behaviors.
- People are good at reasoning about mechanisms.
- Mechanisms have practical applications in mechanical design, repair, and analysis.

The examples used in this work focuses on mechanical mechanisms in two dimensions for rigid objects.

1.4 Qualitative Simulation

Various styles of reasoning have been investigated in qualitative physics, including qualitative simulation, comparative analysis (Weld, 1988), measurement interpretation (Forbus, 1984; DeCoste, 1991), recognition of the function of a system (de Kleer, 1979), and planning. This work focuses on deriving behavior via qualitative simulation.

Qualitative simulation predicts the behavior of a given system over time from its structural description. The behaviors are represented by a set of qualitative states and the transitions between them. Qualitative simulation differs from traditional simulations in several respects. First, while traditional methods compute the state of the system in every fixed time dt, qualitative simulation generates new states only when an interesting event happens. Second, since qualitative simulations are based on less complete information than numerical simulation, they tend to produce alternative possible behaviors, rather than unique predictions.

Qualitative simulation can be done in two different ways: *envisioning* and *history generation.* Envisioning is a process of deriving all possible behaviors—all possible qualitative states and all transitions between them. History generation predicts a specific behavior from a given initial state. Since the present work is interested in finding all possible behaviors for the given system, envisioning is used here.

1.4.1 Global Filtering

In qualitative simulation, a change in the given system is expressed by a transition between two states. Each transition should reflect a correct change both locally and globally. Conceptually, we can view determining correct transitions as a process of filtering illegal state transitions. However, due to the nature of qualitative simulation techniques, they cannot avoid losing some information which is useful for finding unambiguous global behavior. Simulation itself works only by local filtering without taking global behavioral constraints into account: the transition from one state to another is determined only by the constraints between the two states. This technique can predict accurate behavior on the assumption that each state carries enough information so that local filtering is sufficient to get globally correct behaviors. Unlike numerical simulation, which uses precise metric information, qualitative simulation sometimes cannot avoid losing some useful information during suppression of details. We cannot always expect the local filtering to guarantee the global filtering in qualitative simulation. Thus, understanding

how to automate global filtering and how to integrate it with local filtering is crucial to deriving correct behaviors of systems.

Recent work in global constraints has focused on applying the idea of qualitative theory of dynamic systems to qualitative simulations (Lee & Kuiper, 1988, Struss, 1988). After states and transitions are computed by a qualitative simulator, each state is converted to a point in a phase space[1] and each transition is converted to the segment which connects its predecessor and successor state. Once the behavior of a system is expressed as a trajectory in phase space, then geometric constraints are applied to this trajectory to filter behaviors. This approach was useful for some cases such as the stability of a cycle. Unfortunately this approach can be applied only to a limited class of systems. Basically, this method is not adequate for reasoning tasks where thresholds play an important role (Lee & Kuiper, 1988). Furthermore, understanding this approach is not easy for the people who do not have mathematical background. People do not seem to use phase space representation for filtering behaviors.

Our work describes a new technique for representing and manipulating global constraints during simulation. The basic idea is to automatically generate additional information for maintaining global constraints in the form of *events*. This provides sufficient information to filter global behaviors automatically, by introducing variables and controlling their values to guide correct transitions between the behaviors. Chapter 7 shows how this technique is used in reasoning about power cycles of internal combustion engines.

1.5 Understanding Fluids

Understanding the behavior of fluids is crucial in reasoning about the physical world. The key issue in reasoning about fluids is partitioning the fluid into individuals at an appropriate level of detail. Unlike solids, fluids deform as long as shear stresses exist; their shapes are determined by the container. These properties of fluids make it difficult to individuate them in a reasoning system.

Two basic approaches to fluid ontology are the *contained-stuff* ontology and the *piece-of stuff* ontology (Hayes, 1985; Collins & Forbus, 1987; Amador & Weld, 1990; Rajamoney & Koo, 1990; Skorstad, 1992). The contained-stuff ontology views the fluid in a container as a single object. For instance, a river is viewed as a single object with this ontology. On the other hand, the piece-of stuff ontology views particular pieces of the fluid as a single individual object. With this ontology a river is viewed as a collection of water molecules. Both of these ontologies are appropriate for particular reasoning tasks. For example, the contained-stuff ontology can explain global aspects of fluid behavior, (e.g, the change in the pressure at a portal and the changes of the amount of fluid in a container). On the other hand, the piece-of-stuff ontology allows reasoning which the contained-stuff ontology cannot provide, such as details of a thermodynamic behavior.

[1] A phase space for a system is a Cartesian product of state variables.

The present work explores the interactions between fluids and rigid objects through their surface contacts; it focuses on reasoning about spatial aspects of fluids, rather than thermodynamic behaviors.

First, we introduce the *bounded-stuff* ontology for reasoning about behaviors of fluids and their effects on boundaries at the macroscopic level. A bounded fluid is individuated by the geometry of the surface contacts between it and rigid objects since it is the geometry that determines its behavior. Bounded fluids may disappear and reappear as the contact configuration changes during fluid motion. Fluid motion is captured by changes of free surfaces in the fluid, thus showing how the contact configuration changes. The motion of free surfaces is predicted by dynamically analyzing the interaction of surface geometry and pressure disturbances in fluid. Chapter 4 shows how this new ontology can be used to derive the behaviors of fluid with complex surface geometry which cannot be captured in other ontologies.

Second, a qualitative analysis of the direction of fluid flow is presented. This analysis depends on qualitative descriptions of the surface geometry of rigid bodies in contact with the fluid and a pressure change in fluid. We do this by dynamically analyzing the interaction of geometry and a pressure disturbance.

1.6 Overview

This thesis develops theories of fluids and rigid objects, and the inference techniques necessary to understand a broad class of systems involving them. These theories and inference techniques are embodied in *Qualitative System Analyzer* (`QSA`), a program that produces possible behaviors of physical systems. Figure 1.3 shows the block diagram of `QSA`.

Chapter 2 describes a qualitative vector representation. This representation forms the starting point for the rest of our spatial representations. Qualitative arithmetic on the representation is presented. The chapter also shows how rigid objects are described and how kinematic constraints of rigid objects are determined using this representation.

Chapter 3 describes a theory of linkages in two-dimensional space. It explains how possible behaviors of linkages are computed from their qualitative structural description. It also describes how angular changes are represented, introducing *inclination* as a qualitative concept.

Chapter 4 introduces the *bounded-stuff* ontology, a spatially extended contained-stuff ontology. It describes how the interaction of fluid and its neighbors is captured with this ontology.

Chapter 5 describes a qualitative analysis of the direction of fluid flow. It shows the interaction of the pressure of a fluid and the shape of its boundaries.

Chapter 6 describes a method for treating discontinuous changes as a new kind of influence rather than action-like changes.

Chapter 7 describes how global constraints are represented and manipulated, using a model of an internal combustion engine for illustration.

Chapter 8 describes the implementation of `QSA` and shows how its independent modules are combined to derive behaviors of physical systems, using our motivating examples.

Finally, chapter 9 provides a summary, reviews related work, and discusses future research.

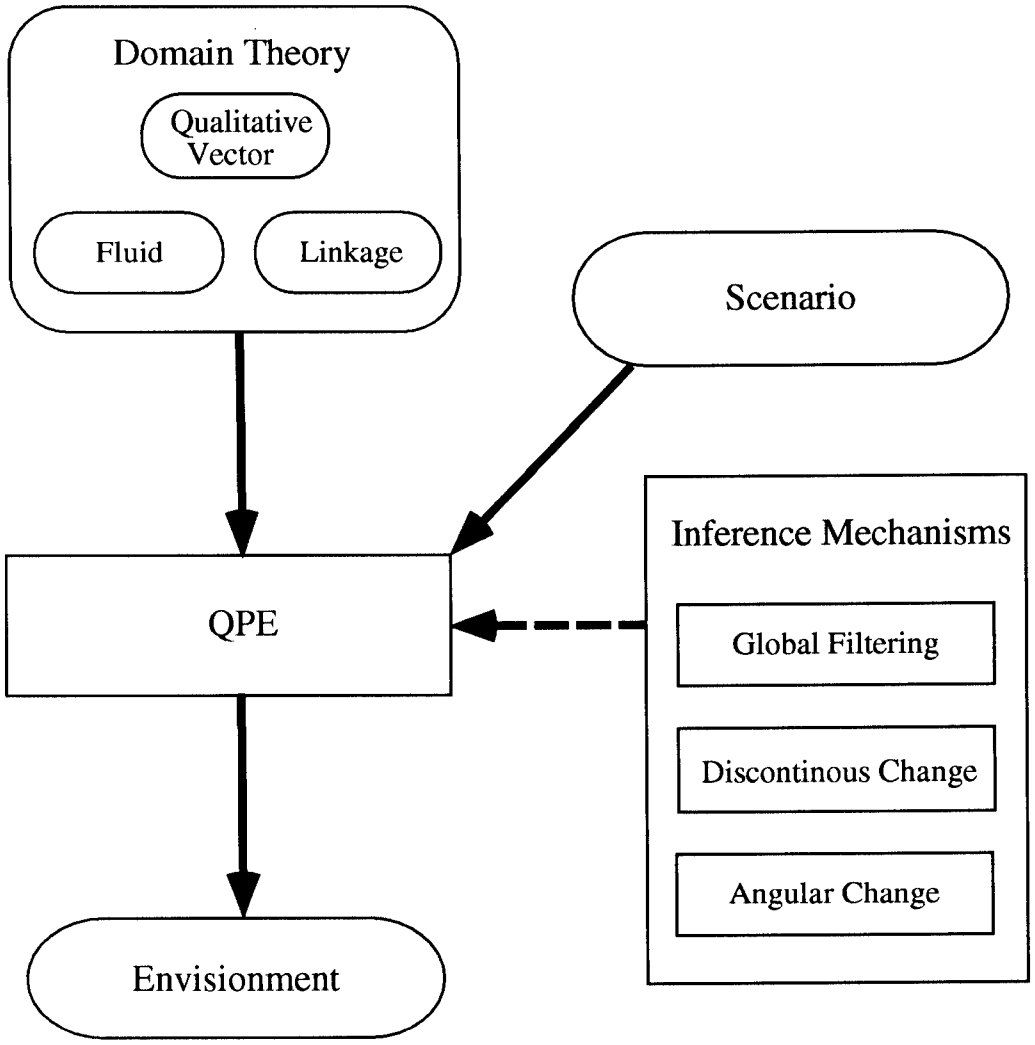


Figure 1.3: Block Diagram of QSA.

Chapter 2

Qualitative Vectors

In spatial reasoning, the concept of vector is essential in describing position, force, and motion. This chapter begins by describing our qualitative vector representation and its arithmetic. We generalize the qualitative vector representation used in (Nielsen, 1988) by adding a richer notion of angle. Qualitative vector arithmetic, which is used to compute directions, is extended to include this new representation. It then shows how they are used to describe rigid objects and find their geometric constraints.

2.1 Qualitative Vector Extended with Angle

In our theory, vectors are represented by *sense* and *inclination.* Sense indicates where the direction is pointing relative to some frame of reference. Sense is represented by an ordered tuple of sign values (Nielsen, 1988). We assume a single global reference frame. This reference frame can be translated but not rotated. In our two-dimensional space representation, the first component of a qualitative vector represents the qualitative direction along the x-axis and the second component represents the y-axis. To represent the x-axis direction, we use "+" for "right" and "−" for "left" and "0" for center. For the y-axis, "+" is used for "up" and "−" for "down" and "0" for center. For example, $(--)$ indicates the vector lies to the lower left of a reference frame. Figure 2.1 shows all possible translational directions in two-dimensional space. This notion of qualitative vector can be easily augmented to include z-axis as the third component to represent the directions in three-dimensional space.

Sense itself is not enough to represent direction. For example, Figure 2.2a and b shows two different kinematic states of a four-bar linkage even though their links have the same senses. Inclination is used to distinguish between these states.

Definition 2.1 (Relative-Angle) `Relative-Angle`*(v1,v2) is the angle measured counter-clockwise from qualitative vector v1 to v2.*

Definition 2.2 (Inclination) *If* `Relative-Angle`*((+0),v) is less than* π*, then* `Inclination`*(v) is equal to* `Relative-Angle`*((+0),v). Otherwise, if* `Relative-Angle`*((+0),v) is greater than or equal to* π*,* `Inclination`*(v) is equal to* `Relative-Angle`*((+0),v)* $- \pi$.

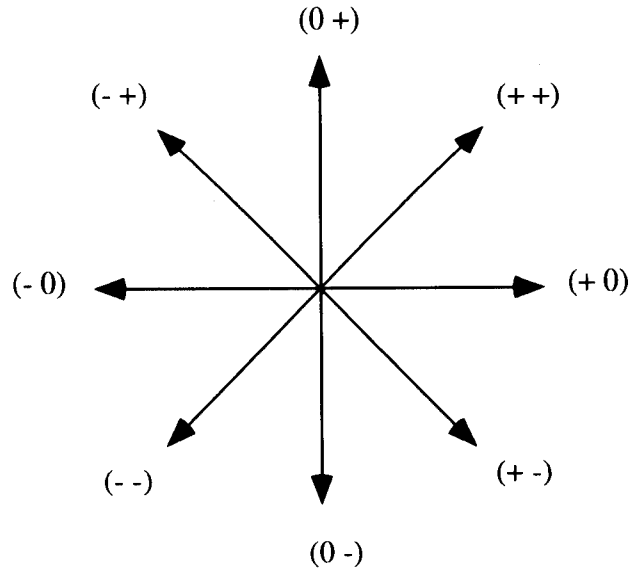


Figure 2.1: Qualitative directions in two-dimensional space

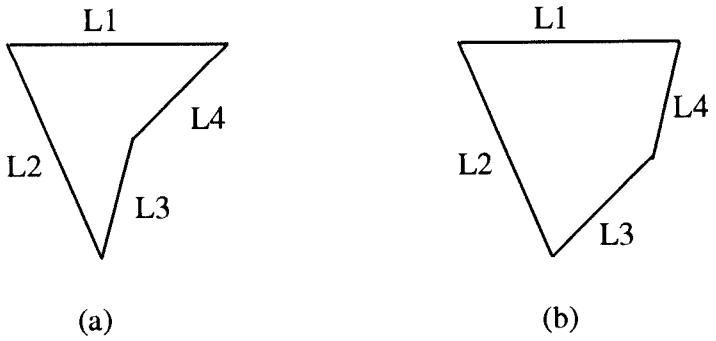


Figure 2.2: Two example configurations with same senses but different inclinations

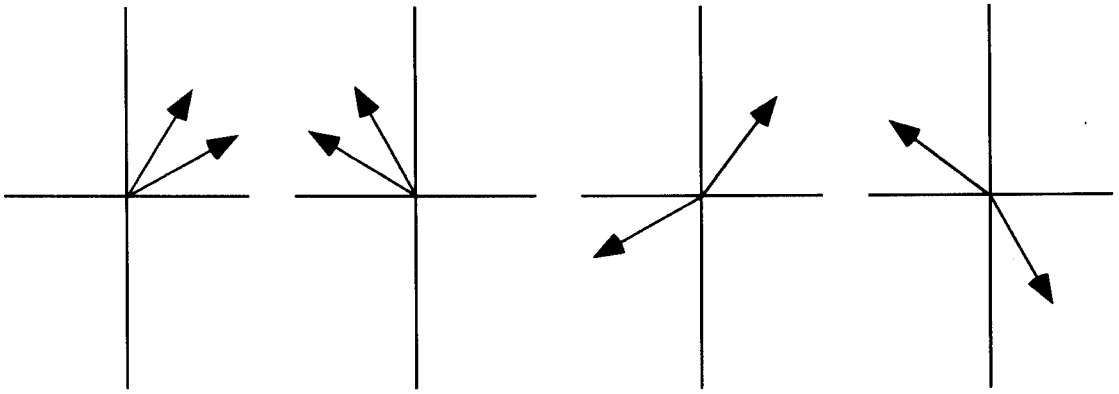


Figure 2.3: Examples of two vectors which have similar inclinations.

Inclination represents how much the vector is inclined relative to x-axis. For our analysis, the exact angle is not needed; instead the comparisons between inclinations are sufficient. Thus we can qualitatively distinguish the states in Figure 2.2 since in (a) `Inclination(L3)` > `Inclination(L4)`, and in (b) `Inclination(L4)` > `Inclination(L3)`.

The following definitions are fundamental to our theory:

Definition 2.3 (Inversion) `Inversion`*(v) is the qualitative vector v rotated by 180 degrees.*

`Inclination(Inversion(`v`))` is same as `Inclination(`v`)`.

Definition 2.4 (Acute Inclination) *A vector v has acute inclination if* `Inclination`*(v) is acute. Thus, a link has a sense of* $(++)$ *or* $(--)$*, exactly when it has acute inclination.*

Definition 2.5 (Obtuse Inclination) *A vector v has obtuse inclination if the* `Inclination`*(v) is obtuse. Thus, a link has a sense of* $(-+)$ *or* $(+-)$*, exactly when it has obtuse inclination.*

Definition 2.6 (Similar Inclinations) *Two vectors have similar inclinations if both have acute inclinations or both have obtuse inclinations.*

Figure 2.3 shows some examples of similar inclinations. If two links have similar inclinations, they can be distinguished by three inequalities of inclinations between them ($>$, $<$, $=$).

Since a global reference frame is used for representing directions, the angular relationship between the directions of two vectors can be defined as follows:

Definition 2.7 (CW) `CW`*(v1,v2) is true if* `Relative-Angle`*(v1,v2) is greater than zero and less than* π.

Definition 2.8 (CCW) `CCW`*(v1,v2) is true if* `Relative-Angle`*(v1,v2) is greater than* π.

Figure 2.4 shows an example for `CW` and an example for `CCW`.

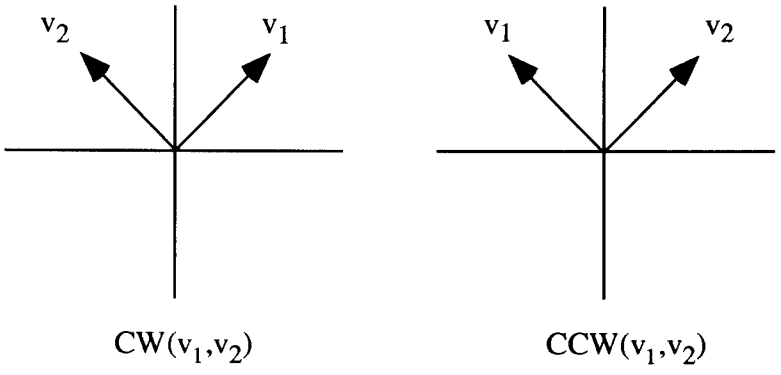


Figure 2.4: Examples of `CW` and `CCW`

2.2 Qualitative Vector Arithmetic

We extend the qualitative vector arithmetic with relative angles and relative lengths to enable us to filter more ambiguities. The sense of the sum of two vectors is the qualitative sum of their senses. When we add two qualitative vectors, the sense is computed by adding their senses. The inclination of the summed vector is constrained by the inclinations and lengths of the two vectors.

Law 2.1 (Addition) *If* `CW`*(v1,v2) and* $v = v1 + v2$*, then* `CCW`*(v,v1) and* `CW`*(v,v2) are true. If* $v1$ *and* $v2$ *have the same inclinations, then* v *has the same inclination.*

Figure 2.5 shows some examples. The possible summation of $v1$ and $v2$ is within the hatched region. In 2.5a, v has the same sense as $v1$ and $v2$ but `Inclination`$(v1)$ is between the inclinations of $v1$ and $v2$. In 2.5b, v has the sense whose y-direction is $+$ and its inclination is constrained by $v1$ and $v2$. In 2.5c, when we add $(++)$ and $(--)$, our answer can be any of the qualitative vectors. Such cases of ambiguous additions provide no useful information. Our vector arithmetic avoids such ambiguities by using relative angles and lengths. The summation of $v1$ and $v2$ of 2.5c cannot be $(+0)$, $(+-)$, $(0-)$, $(++)$ with less inclination than that of $v1$, or $(--)$ with greater inclination than that of $v2$. If we have information about the inequalities between the magnitude of vectors, more constraint can be imposed. For example, if we know magnitude of $v1$ is greater than that of $v2$ in 2.5c, then the y value of $v1 + v2$ should be "+".

We now generalize `Open-Half-Plane` and vector rotations used in (Nielsen, 1988) with inclinations.

Definition 2.9 (Rotate-90) `Rotate-90`*(v,d) is the vector which is perpendicular to* v *by rotation in direction* d*. Rotational direction is represented as "+" for counter-clockwise and "−" for clockwise rotation.*

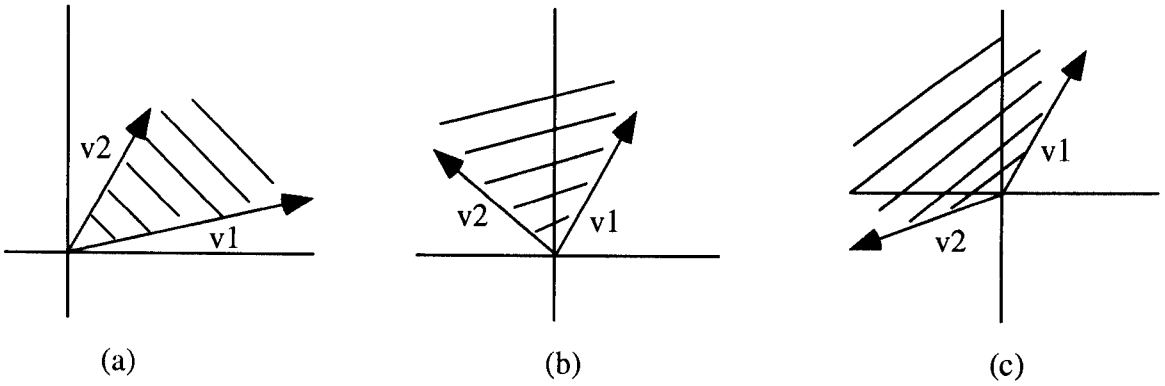


Figure 2.5: Vector Arithmetic

Law 2.2 (Rotation-90) *If the vectors* $v1$ *and* $v2$ *have similar inclinations and* `Inclination` $(v1) >$ `Inclination`$(v2)$*, then inclination of* `Rotate-90`$(v1,d)$ *is greater than that of* `Rotate-90` $(v2,d)$ *(Figure 2.6).*

Definition 2.10 (Open-Half-Plane) `Open-Half-Plane`(v) *are the vectors whose vector dot product with* v *is "+".*

Vectors having the same sense as `Rotate-90`$(v,-)$ and whose inclinations are greater than that of `Rotate-90`$(v,-)$ are also included in `Open-Half-Plane`(v) even though its dot product is qualitatively ambiguous. Similarly, vectors having the same sense as `Rotate-90`$(v,+)$ and whose inclination is less than that of `Rotate-90`$(v,+)$ are included in `Open-Half-Plane`(v).

2.3 Rigid Object Representation

To analyze a system, we first need a way to describe it. A system is described by representing each part and how it is related to others. As the notion of a part changes the configuration of the mechanism, new kinematic interactions between parts takes place. Thus each part continuously transmits and modifies both force and motion through the system. A part of a system is either solid or fluid. In this section we focus on solids, especially rigid objects. Fluids will be explained in Chapters 4 and 5.

Since we are focusing on mechanical forces, interactions between two objects can happen only when they are in contact. The interaction depends on how they come into contact. Thus a rigid object should be described in terms of its surfaces. The shape of a rigid object is represented by its surfaces and its geometric relationship with other objects is described by contacts between their surfaces.

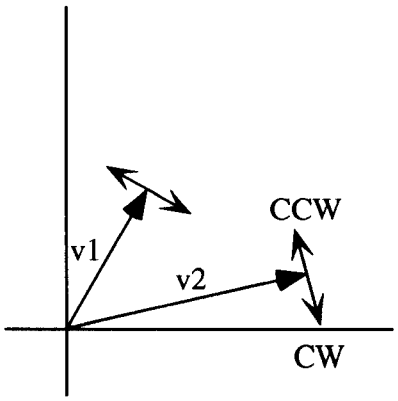


Figure 2.6: Directions of `Rotate-90` of two links with similar inclinations

In our two-dimensional space, a surface is represented as a line segment with surface normal. For each surface, we represent the direction of the line segment as the position of one end-point relative to the other. (We define end-points as the points where the line segment meets its neighbors.)

In general, the relative position can be defined for any two points:

Definition 2.11 (Relative-Position) `Relative-Position(p1,p2)` *is the qualitative vector which represents the direction from point* `p2` *to point* `p1`.

Consider a surface with end-points `p1` and `p2`. The direction of the surface is represented by `Relative-Position(p2,p1)`. By representing line segments in terms of end-points, the geometric relationships between surfaces are described.

Information about the relative position can be propagated using transitivity:

Definition 2.12 (Transitivity of Relative-Position) *For any points* `p1`, `p2`, *and* `p3`, `Relative-Position(p1,p3)` *is computed by adding the given values* `Relative-Position(p1,p2)` *and* `Relative-Position(p2,p3)`.

The relative direction of two adjacent surfaces is defined for the two surfaces:

Definition 2.13 (Surf-Rel-Pos) *For any two adjacent surfaces* `s1` *and* `s2` *whose end-points are* `(p1,p2)` *and* `(p2,p3)`, `Surf-Rel-Pos(s1,s2)` *represents* `Relative-Position(p1,p3)`.

As shown by (Nielsen, 1988), the surface normal of a surface plays a key role in determining how forces and motions are transmitted through the surface.

Definition 2.14 (Surface Normal) `Surface-Normal(s)` *is the qualitative vector which represents the surface normal of the surface* `s`.

2.3.1 Finding Constraints

Given a structural description of a system, the motion of each part is determined by the forces applied to the part and geometric configuration of the system. Certain paths of motion may be prevented by contacts between rigid objects while other paths of motion are not prevented at all. For example, a piston in a cylinder is free to move in up and down directions while it is constrained to the other directions. When a rigid object is free to move in some direction and force is applied to that direction, the object will move in that direction. On the other hand, when a object is constrained against moving in some direction, the object will not move that direction even though forces are applied to that direction.

How the motions of an rigid object are constrained by contact between rigid objects is identified in (Nielsen, 1988). We use Nielsen's law of contact constraint to determine constraints of rigid objects in each configuration. The following definitions and Figure 2.7 are from (Nielsen, 1988):

Definition 2.15 (Constraint) `TransConstraint(o,t)` *is true when object is absolutely prevented from moving in direction* `t`. `RotConstraint(o,r)` *is true when object* `o` *is absolutely prevented from rotating in direction* `r`.

Definition 2.16 (Freedom) `TransFreedom(o,t)` *is true when object is not prevented from moving in direction* `t`. `RotFreedom(o,r)` *is true when object* `o` *is not prevented from rotating in direction* `r`.

An object is free to move in some direction unless it is constrained in that direction. The following motions are prevented in an object, if the object which is in contact with the object is *sufficiently constrained.*

- Translational motion into the open half plane centered on the object's surface normal at the point of contact.
- Rotational motion clockwise about any point which lies in the open half plane centered ninety degrees clockwise from the object's surface normal at the point of contact.
- Rotational motion counter-clockwise about any point which lies in the open half plane centered ninety degrees counter-clockwise from the object's surface normal at the point of contact.

An object is sufficiently constrained if it is unable to move in any of the directions mentioned above. Figure 2.7 shows the constraints of a block by wall.

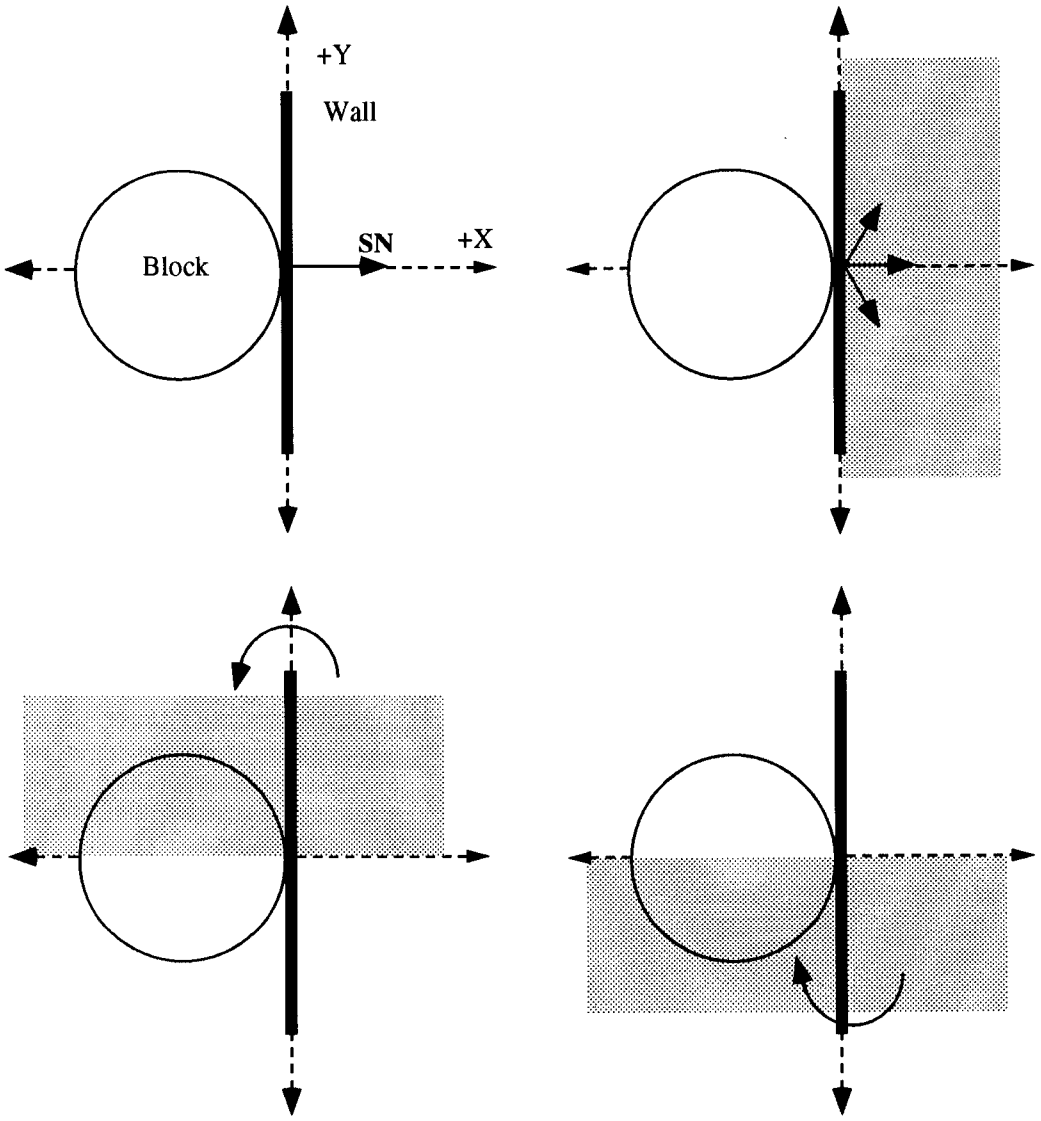


Figure 2.7: Constraints imposed by surface contact.

Chapter 3

Qualitative Kinematics of Linkages

Many important machines include linkages. Linkages are used to transfer motion (or force or torque) from one part of a system to another. For example, the slider-crank mechanism which transmits the vertical motion of the piston to the rotation of the crankshaft is a crucial part of automobile engines (Figure 3.1). Unfortunately, previous techniques focused on fixed axis mechanisms (Faltings, 1986; Joskowicz, 1987; Nielsen, 1988), and thus could not analyze many important systems, including linkages.

In this work, we develop a qualitative theory for linkages in two-dimensional space. A linkage is defined as "an assemblage of rigid bodies or links in which each link is connected with at least two other links by either pin connections or sliding blocks" (Cowie, 1961). If the linkage is not sufficiently constrained, it is not considered to be a mechanism. A four-bar mechanism is a four-bar linkage with one link fixed, so that its motions become relatively constrained. The analysis of four-bar mechanisms is important since it is the most commonly used linkage and it provides a basis for more complicated linkages. Its motion is constrained to have a single degree of freedom, given a driver. In general, if we have an n-bar mechanism with m degrees of freedom and m links are driven, every remaining link has a corresponding and predictable motion because of constraints. Our analysis can be generalized to more complicated linkages by propagating constraints across pairs of adjacent links.

We assume as input the connection between the links and their relative lengths. The result of the analysis is the qualitative states and state transitions which characterize the possible behaviors of the linkage. The basic idea of our approach is to represent *relative motions* of each link in terms of quadrants (qualitatively representing the direction relative to a global reference frame) and *relative inclinations* relative to the x-axis, as described in the previous chapter. Using this representation, we can derive all the possible motions of linkages.

3.1 Finding States

This section shows how to generate all possible states of a given linkage. Each state is qualitatively distinct from others; all of these states and transitions between them describe the possible behaviors of the linkage. A state consists of a static component representing the orien-

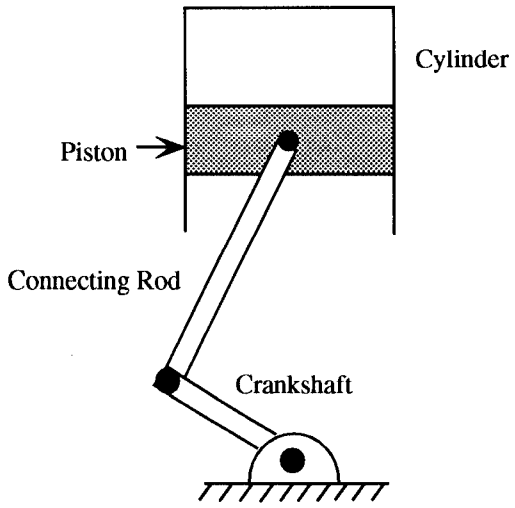


Figure 3.1: The slider-crank mechanism of a reciprocating engine

tations of each link, and a dynamic component representing the motions. This section begins with the representations of positions and motions of links. Then it describes how geometric constraints are used to filter illegal configurations of linkages. Position, motion, and constraints are expressed by relative terms.

3.1.1 Relative Position for Links

In linkages, the contact relationship between links does not change in time. The behavior of a linkage is determined by the connection and relative lengths between links, not by the shape of each link. Thus, the surface normal of each link can be ignored.

For each link, we represent the orientation of that link as the position of one end relative to the other. If the other end-point is fixed to the ground, that relative position represents the absolute position. As defined in Chapter 2, `Relative-Position(P1,P2)` is the qualitative vector which represents the direction from point `P2` to point `P1`. Information about the relative position can be propagated using transitivity.

Consider a link with end-points `P1` and `P2`, which are pin-jointed to other adjacent links. To get the possible orientations of the link, imagine `P1` is fixed and `P2` rotates about `P1`. The direction of the link is represented by `Relative-Position(P2,P1)`.

Since each link is connected to the other by either a pin joint or a sliding block carrying a pin joint, connected links can only rotate relative to each other. Quadrants are used to represent the changes of angular positions. Since a qualitative vector in each quadrant and at the quadrant boundaries all have unique senses, there are 8 possible qualitative angular positions for a link.

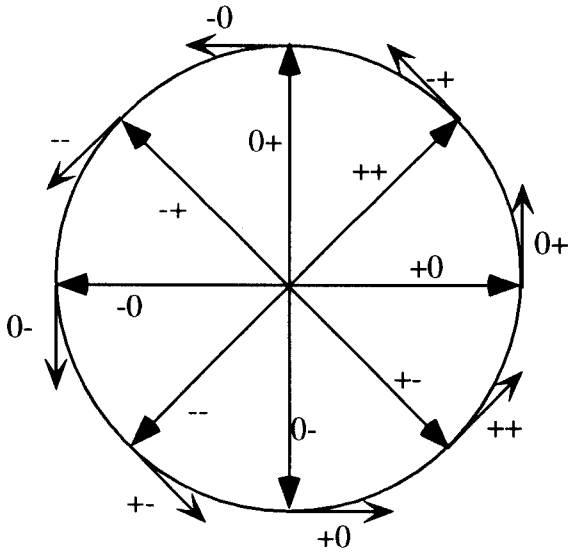


Figure 3.2: Angular positions and motion vectors in quadrants

3.1.2 Relative Motions

To represent the direction of motion, we also use qualitative direction. Just as the orientation of a link is represented by the relative position of two end points of that link, the motion of a link is represented by the relative motion of its two end points.

Definition 3.1 (Rotation) `Rotation(P1,P2)` *represents the rotational direction of the line connecting* `P1` *and* `P2`. *Possible values are + (CCW), − (CW), and 0 (no motion).*

The rotation of a link with end-points `P1` and `P2` can be represented by `Rotation(P1,P2)`.

Definition 3.2 (Relative-Motion) `Relative-Motion(P1,P2)` *is the qualitative vector which represents the direction of the motion of point* `P1` *relative to point* `P2`.

Given vector v `Relative-Position(P2,P1)` for link-i, its the motion is represented by `Relative-Motion(P2,P1)`. The possible values of `Relative-Motion(P2,P1)` are `Rotate-90(`v`,+)`, `Rotate-90(`v`,−)` and `(00)` for no motion. We call these non-zero relative motions *CCW rotation vector* and *CW rotation vector*, respectively. Figure 3.2 shows the 8 angular positions and corresponding unique `CCW` rotation vectors.

Figure 3.3 shows the orientations of the clockwise and counter-clockwise rotation of `L2`. Since `Relative-Position(B,A)` is (++), `Relative-Motion(B,A)` can be (+−) for clockwise rotation or (−+) for counter-clockwise or (00) for no motion. In this case the directions of these motions are absolute since pin joint A is fixed to the ground.

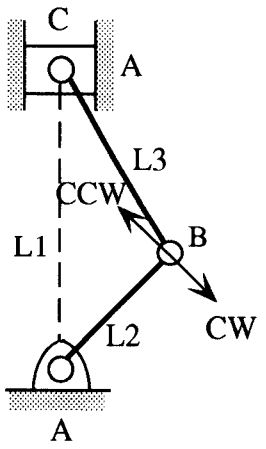


Figure 3.3: Slider-crank mechanism

When two links have similar inclinations, their rotation vectors will also have similar inclinations. Let $v1$ and $v2$ represent the directions of `link1` and `link2` respectively and have similar inclinations. From the Law of `Rotation-90`, it follows that if `Inclination`($v1$) > `Inclination`($v2$), then inclination of the motion vector of $v1$ is greater than that of $v2$.

`Relative-Motion`, like `Relative-Position` is transitive:

Law 3.1 (Transitivity of Relative-Motion) `Relative-Motion(P1,P3)` *is the sum of* `Relative-Motion(P1,P2)` *and* `Relative-Motion(P2,P3)`.

3.1.3 Coupled Vectors

To compute the states from the possible relative positions and motions of every pair of links, an adaptation of the *method of coupled vectors* (Cowie, 1961) is used. The idea of this traditional method is to express a vector as the sum of two vectors. We now apply it to our qualitative representations.

Consider the slider-crank mechanism of Figure 3.3. By the law of `Transitivity of Relative-Position`, we can compute `Relative-Position(C,A)` by summing the `Relative-Position(B,A)` and `Relative-Position(C,B)` since `L2` and `L3` are connected by the pin joint `B`. Since `Relative-Position(C,A)` is constrained to move only in vertical direction, the illegal configurations of `L2` and `L3` are filtered out. Similarly, `Relative-motion(C,B)` can be computed by this method according to the the law of `Transitivity of Relative-Motion`.

Computing absolute motions or positions in a movable mechanism is complicated unless the links are directly connected to the ground. Our method finds the absolute motions and positions by summing the relative ones of adjacent links. During this process, illegal configurations and

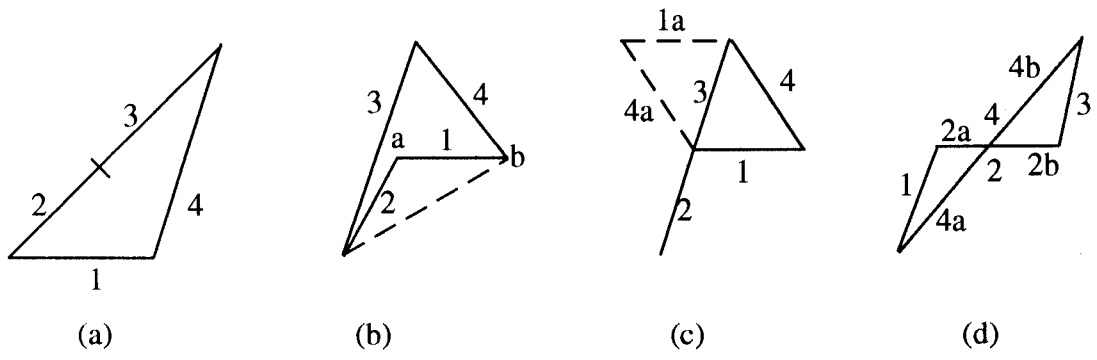


Figure 3.4: Triangles

motions are filtered out by given geometric constraints, namely qualitative vector arithmetic with relative angles and relative lengths.

If a vector is represented as the sum of two vectors, the change in the angle and magnitude (i.e., the change in direction of the motion) can also be expressed as sums. Since the `Relative-Position` for the link in a linkage cannot change magnitude, the computation of the motion of the summed vector of two links can be simplified. In the following section we give our method for using this simplification to maximally exploit the qualitative geometry to find angular constraint between two connected links.

3.1.4 Triangle Constraints

To produce consistent global kinematic states from the local information about the position of each link, we need geometric constraints in addition to vector arithmetic. The following laws have rational interpretations in qualitative terms and are used to filter out illegal kinematic states of linkages given the relative lengths of the links.

Law 3.2 *In a triangle, the lengths of any two sides is longer than the remaining side.*

Law 3.3 *In the case of a right triangle, the hypotenuse is always longer than either leg. Similarly for an obtuse triangle, the side opposite the obtuse angle is longer than either side adjacent to the obtuse angle(Figure 3.4a).*

Law 3.4 *Given four-sided polygon 1-2-3-4 with a reflex angle a, we can draw a segment 5 from d to b (Figure 3.4b). By observation, we can see that for the smaller triangle 1-2-5 to be "inside" the larger triangle 3-4-5, $3 + 4 > 1 + 2$.*

Law 3.5 *Now consider a special case of the Law 3.4 where the side 2 is folded onto side 3 (Figure 3.4c). In agreement with the Law 3.4, $3 + 4 > 1 + 2$. Furthermore we can show $1 + 3 > 2 + 4$. First we consider a parallelogram where $1a = 1$ and $4a = 4$. By examining the new 4-sided polygon, we can see $1a + 3 > 2 + 4a$ which implies $1 + 3 > 2 + 4$.*

		Angular Position							
		(++)	(0+)	(−+)	(−0)	(−−)	(0−)	(+−)	(+0)
Motion	CCW	(0+)	(−+)	(−0)	(−−)	(0−)	(+−)	(+0)	(++)
	CW	(+0)	(++)	(0+)	(−+)	(−0)	(−−)	(0−)	(+−)
Duration		int	inst	int	inst	int	inst	int	inst

Table 3.1: Angular changes. Duration represents the change from an angular position occurs for an instant (inst) or for an interval of time (int).

Law 3.6 *In the four-sided polygon 1-2-3-4 with 2 and 4 crossed, we can show 2 + 4 > 1 + 3 since 2a + 4a > 1 and 2b + 4b > 3 by Law 3.2 (Figure 3.4d).*

These constraints help filter illegal kinematic states which vector arithmetic cannot. By these laws and the method of coupled vectors, we can find every possible kinematic states for all kinds of four-bar mechanisms. To handle linkages which have more than 4 links, we may need more geometric constraints about polygons. But these constraints about triangles and four-sided polygons will be still applicable since adjacent links can be summed into one link.

3.2 Angular Changes

The motion of a link can change its angular position. For example, if a link is placed at the first quadrant and motion of the link is clockwise, that link will point directly "right", then the forth quadrant and so on. But due to the nature of angles, where discontinuous changes occur, angular changes cannot be captured by the usual laws governing limit analysis. However, we can formulate new laws for angles which extend the notion of limit analysis to cover them. How the angular position of a rotor changes by its motion is shown in Table 3.1.

The duration of a transition is determined by the law of Equality Change Law (ECL) (Forbus, 1984). The duration is either an instant or an interval of time.

Law 3.7 (Equality Change) *Transitions from equality occur in an instant. In all other cases transitions last for an interval of time.*

Since our angular position is expressed by an ordered tuple of sign values, we can determine the duration by checking all sign values. For example, the change from (0+) occurs in an instant because of the x direction. On the other hand, the change from (++) requires for an interval of time.

3.2.1 Angle between two Links

To predict transitions between states, changes in relative positions and motions are not sufficient. In Figure 3.5, for example, reasoning about angular change between two links is needed.

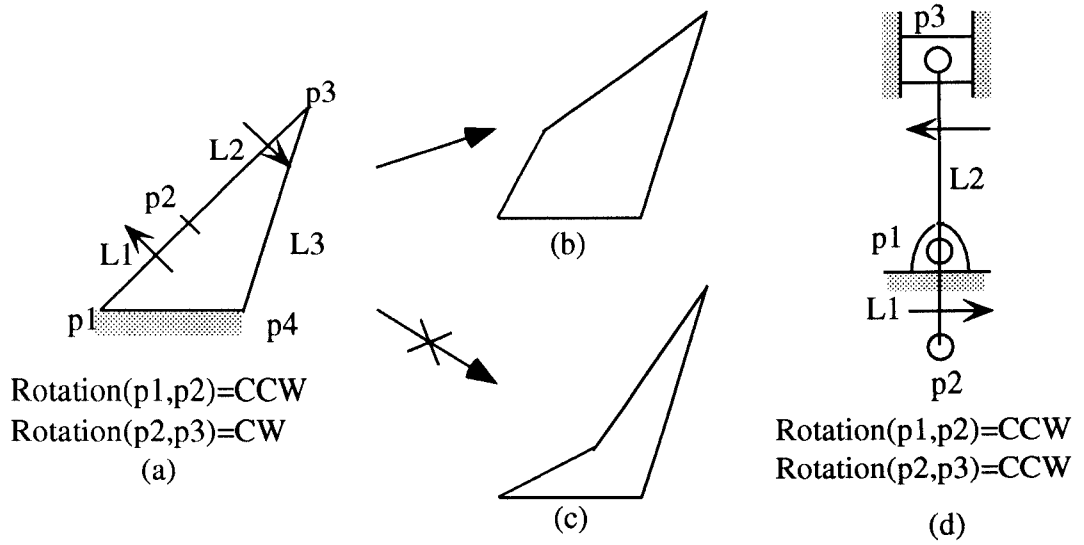


Figure 3.5: Examples of Angular Change between two moving links

Angular relationships with other links can change if a link is moving, as shown in Table 3.2. This table filters the illegal transition from the state of Figure 3.5a to 3.5c. But for the ambiguous cases (i.e., N1 - N6 in the table), we need to know the inequalities of the angular velocities between the links. In Figure 3.5d, for example, linear velocities of P1 and P3 relative to P2 are the same. Since the linear velocity of the end point of a rotating link is its length times its angular velocity and L2 is longer than L1, the angular velocity of L1 is greater than that of L2. Let $v1$ and $v2$ represent Relative-Position(P2,P1) and Relative-Position(P3,P2) respectively. We can eliminate Inclination($v1$) < Inclination($v2$) and Inclination($v1$) = Inclination($v2$) from the next possible state since it will be (CW $v1$ $v2$). Therefore, the only next possible state for Figure 3.5d has $v1 = (+-)$, $v2 = (-+)$, with Inclination($v1$) > Inclination($v2$).

We may not always be able to determine inequalities between angular velocities. For those cases, another method finds transitions that exploit the available constraints. That method focuses on the angular change between two adjacent links. The analysis of this change is important since it can intuitively show how the shape of a linkage changes with motion (e.g., two adjacent links are approaching or departing).

Law 3.8 (Angular Change) *Suppose* L1 *and* L2 *are two adjacent links with different inclinations. If their end points are (*P1 *&* P2*) and (*P2 *&* P3*) respectively, then by the Law of Transitivity of* Relative-Motion *we can compute* Relative-Motion(P3,P1) *by adding* Relative-Motion(P3,P2) *and* Relative-Motion(P2,P1). *If* Relative-Motion(P3,P1) *is contained in* Open-Half-Plane(Relative-Position(P3,P1)), *then* L1 *and* L2 *are departing. If* Relative-

For the sake of convenience, CW(A,B), CCW(A,B) are represented as A < B and A > B, respectively. When A and B have the same inclinations, A =o B indicates A and B have the opposite sense and A =s B indicates A and B have the same sense. Ds[A] and Dm[A] represent the sign and magnitude of the angular velocity of A.

For A > B:

Ds[A] \ Ds[B]	-1(CW)	0(No Motion)	1(CCW)
-1(CW)	N1	=s	=s
0	=o	>	=s
1(CCW)	=o	=o	N2

```
N1:Dm[A] > Dm[B] then =s, otherwise =o
N2:Dm[A] < Dm[B] then =o, otherwise =s
```

For A =s B:

Ds[A] \ Ds[B]	-1(CW)	0(No Motion)	1(CCW)
-1(CW)	N3	<	<
0	>	=s	<
1(CCW)	>	>	N4

```
N3:Dm[A] > Dm[B] then <        N4:Dm[A] > Dm[B] then >
   Dm[A] < Dm[B] then >           Dm[A] < Dm[B] then <
   Dm[A] = Dm[B] then =s          Dm[A] = Dm[B] then =s
```

For A =o B:

Ds[A] \ Ds[B]	-1(CW)	0(No Motion)	1(CCW)
-1(CW)	N5	>	>
0	<	=o	>
1(CCW)	<	<	N6

```
N5:Dm[A] > Dm[B] then >        N6:Dm[A] > Dm[B] then <
   Dm[A] < Dm[B] then <           Dm[A] < Dm[B] then >
   Dm[A] = Dm[B] then =o          Dm[A] = Dm[B] then =o
```

Table 3.2: Changes in Angular Relationship for given Motions

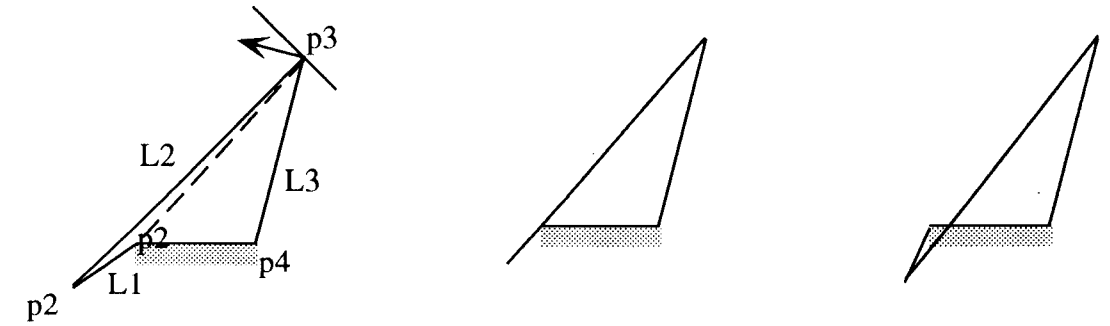


Figure 3.6: State Transition in Crank-Rocker Mechanism

`Motion(P3,P1)` *is contained in* `Open-Half-Plane(Inversion(Relative-Position(P3,P1)))`, *then* `L1` *and* `L2` *are approaching. Otherwise, there is no angular change between* `L1` *and* `L2`.

Using this law for two adjacent links with different inclinations yields the same result as Table 3.2. In the table, the next state having `=s` means two links are approaching while `=o` means they are departing.

We use `approaching` or `departing` instead for the decreasing or increasing of the relative angle since it is not clear whether the angle is the reflexive angle or the obtuse one. Whether the angle is decreasing or increasing depends on the shape of the linkage. Our strategy is especially useful for finding additional constraints when two adjacent links move in same the direction and the angular velocities are not known. If `Relative-Motion(P3,P1)` is computed by adding the motions of two adjacent links, it may have several qualitative vectors rather than one. But this vector is constrained to have an unique value since every link in a mechanism is constrained to have unique motions in a given kinematic state.

Figure 3.6 shows three qualitative states of the crank-rocker mechanism that we explain later. Table 3.2 and the law predict the transition from Figure 3.6a to b and to c by reasoning about the angular relationship between `L1` and `L2`. In Figure 3.6a, `L1` and `L2` are rotating `CCW` and `L3` is rotating `CCW`. If we add the relative motions of `L1` and `L2` to compute `Relative-Motion(P3,P1)`, there are several possible vectors. But `Relative-Motion(P3,P1)` is constrained to be $(-+)$ perpendicular to `Relative-Position(P3,P4)` since `Relative-Motion (P3,P4)` is constrained to move in that direction and both `P1` and `P4` are fixed to ground. If we consider triangle `P1-P2-P3`, the length of `P1-P2` and `P2-P3` are fixed but `P1-P3` is decreasing, which implies angle `P1-P2-P3` is decreasing. Since the angle `P1-P2-P3` is decreasing, `L1` and

L2 are approaching each other. This implies that L1 and L2 may be folded at the next state (Figure 3.6b). In Figure 3.6b, both L1 and L2 rotate CCW. Since the angular velocity of L1 is greater than that of L2 in Figure 3.6b, the table for A =s B shows that the inclination of L2 will be less than that of L1 in the next state (Figure 3.6c). Since L1 and L2 are predicted to be departing from Figure 3.6c, they are not folded in the next state. If L1 is driven to rotate in the CW direction, the motions of the remaining links will be opposite. Then using the same analysis, we can predict that the direction of the transition will be reversed.

3.3 Envisioning Linkages

A qualitative state consists of the union of kinematic state and dynamic state. A kinematic state represents a particular configuration of links. The dynamic state describes the motions of the links. Motions can change the kinematic state of the link. If the linkage is a mechanism, each link has only one motion in a given kinematic state. Otherwise, each link may have more than one motion. Given a description of a linkage in terms of relative lengths of links and the connections between them, the following algorithm computes an envisionment for that system.

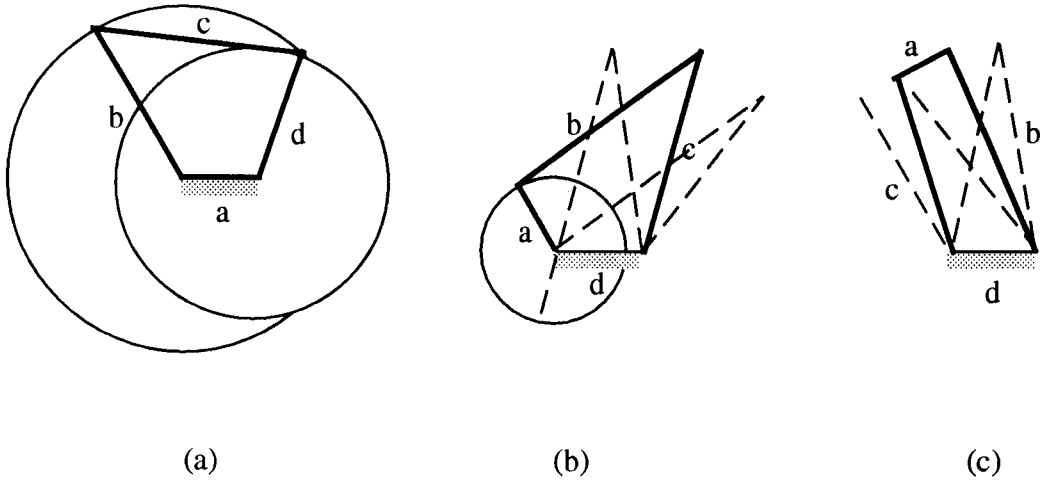
1. Compute the kinematic states for the given mechanism.
   (a) Generate all possible positions of the links in terms of the relative positions of their two end points.
   (b) Filter out all illegal combinations using the coupled vector method based on the extended vector arithmetic and triangle constraints.
2. Compute the dynamic state for each kinematic state.
   (a) Generate all motions for the given position of each link.
   (b) Filter out all illegal motions in every kinematic state using the coupled vector method based on the extended vector arithmetic and triangle constraints.
3. Compute all possible transition between the states.
   (a) For each state, find all kinematic transitions by using directions of motion and information associated with quadrant and angular change.
   (b) For each state, find all dynamic transitions.
   (c) Find all consistent combinations of dynamic and kinematic transitions.

3.4 Implementation

LINKAGE is a program that implements the algorithm in Section 3.3. It is an independent program that produces a qualitative analysis of linkages, given minimal geometric information. It consists of three major parts: The theory of linkages module (Section 3.1), angular change

```
(link-pins-connection L2 p1 p2)
(link-pins-connection L3 p2 p3)
(grounded L1 p1 p3 (:ZERO :PLUS))
(sliding-block L1 p1 p3 vertical)
(greater-than (length L3) (length L2))
```

Figure 3.7: Description of a slider-crank mechanism



(a) (b) (c)

Figure 3.8: Four-Bar Mechanisms

module (Section 3.2), and envisioning module (Section 3.3). The first is used as a domain model for linkages and the second is used during simulation in QSA.

LINKAGE has analyzed slider-crank mechanisms, four-bar mechanisms, a mechanism where two slider-crank mechanisms are connected by sliding blocks, and linkages which were not fully constrained. Figure 3.7 shows an input description of the slider-crank mechanism shown in Figure 3.3. For each `link-pins-connection`, relative-positions in x and y, and relative motion are defined for the end-points of a link.

Taking the four-bar mechanism as an example, there are three possible mechanisms to consider. In a four-bar mechanism, the link that is attached to ground may rotate completely or may oscillate. We call the former a crank and the latter a rocker. The three possible mechanisms are (1)two links attached to the ground rotate completely (*drag-link* mechanism) (2)one grounded link rotates and the other oscillates (*crank-rocker* mechanism) (3)both grounded links oscillate (*double-rocker* mechanism). In kinematics textbooks, these different mechanisms are categorized based on the relative lengths between the links. Suppose `a,b,c,` and `d` represent

the links. We stipulate that link a is the shortest link and b the longest and a + b < c + d. If a is fixed, this becomes the drag-link (Figure 3.8a). If a is used as the driver, this becomes crank-rocker (Figure 3.8b). If a connects the two grounded links, it becomes the double-rocker (Figure 3.8c). When **LINKAGE** is given the connections and relative lengths between the links, it can produce the behaviors of the corresponding mechanism.

Complete envisionments for these four-bar mechanisms and a slider-crank mechanism are given in appendices.

For every kinematic state, there is a unique and corresponding motion for each link. However, since only relative inclinations are used to represent the kinematic state instead of the exact value of the angle between the links, several qualitatively consistent dynamic state are sometimes computed for a single kinematic state. For n-bar linkages which are not constrained enough, every possible dynamic state is also computed for each kinematic state.

Our analysis extends to 2-dimensional motions even when not all parts of the linkages lie in the same plane. For example, in an actual engine the piston, block, crank-shaft, and the connecting rod do not lie in the same plane. However, since all parts move in parallel planes, 2-dimensional analysis suffices. Linkages which require three-dimensional space are very rare. For instance, (Cowie, 1961) contains no such 3D linkages. It is a reasonable first step to develop a 2-dimensional theory and then use this to attack the 3-dimensional problem. Consider a 3-dimensional linkage. Its possible configurations and motions can be analyzed in xy-plane, yz-plane, and zx-plane and then the analyses in each plane could be combined to give the positions or motions in three-dimensional space. Compared to the two-dimensional domain, the more ambiguities are expected, due to more ambiguity about the relative lengths in each plane. In some cases, the ambiguities in relative lengths in each plane cannot be avoidable without the exact numerical values for positions on that plane.

Chapter 4

Reasoning About Fluids Using Surface Boundaries

The behaviors of a physical system are explained in terms of the interactions of its parts. Understanding the kinematic interactions between parts is thus very important. Many systems include both rigid objects and fluids, yet qualitative physics has focused on rigid solids and fluids separately. Most researchers avoid the integration problem by either ignoring the interactions or only allowing very simple ones.

Suppose we want to explain the motion of the liquid in an U-tube whose left tube is connected to a gas tank and right tube is open (Figure 4.1). At first, the pressure in the tank is equal to the atmospheric pressure and thus the level of liquid in both sides are the same. Next, assume the pressure in the tank is increasing. Most qualitative physics research dealing with fluid systems requires viewing the contained fluid in a container as a single lumped-parameter object. Thus it is impossible to consider the motion in each side of the tube independently. Similarly, it is intractable to consider the motion of every minute piece of fluid. However, the downward motion of the liquid-gas surface in the left tube and the upward motion of the liquid surface in the right tube can be easily predicted by simple geometric analysis.

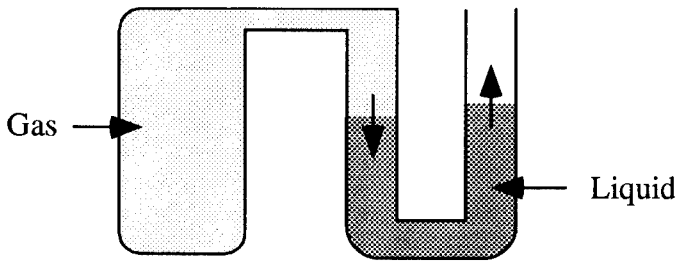


Figure 4.1: An example of an U-tube with liquid.

This chapter describes the model and implementation of the theory of fluids used in QSA. (The exception is direction of fluid flow, in Chapter 5.) It presents a technique for reasoning about the behaviors of fluids and their subsequent effects on the surfaces of rigid bodies in contact with the fluids. The behaviors of fluids are derived by analyzing interactions at their surfaces. We focus on kinematic behaviors of fluids rather than the details of the thermodynamics of fluids. Our goal is to reason qualitatively about how forces are transmitted between fluids and other parts, and how their motions change in a system.

Since interactions between fluids and other objects occur only when they are in contact, our theory individuates fluids based on the geometry of surfaces in contact with them. Our theory dynamically predicts how bounded fluids change as the contact configuration changes during fluid motions. Fluid motions are described by changes of free surfaces in the fluids, thus showing how the contact configurations change. During this analysis, the changes of fluids inside containers are easily determined by relating the geometric structures of surfaces to containers.

Section 4.1 outlines the nature of fluids and shows the problems and desiderata in designing a reasoning system for fluids. Section 4.2 shows the drawbacks of the contained-stuff ontology in terms of these desiderata, and shows the need for spatial reasoning based on surface contacts between fluids and their neighbors. In Section 4.3, we describe how the interaction of pressure disturbances and surface geometry captures the behaviors of fluids in our theory.

4.1 The Nature of Fluids

Unlike solids, fluids move and deform continuously as long as shear stresses exist. The shape of a fluid is determined by its container. This property makes it difficult to individuate fluids in a reasoning system. In physics, the following physical theory about dynamics and geometries is used to capture the behaviors of fluids (Halliday & Resnick, 1974).

- *Pressure transmission:* Pressure is transmitted to solid boundaries or across arbitrary sections of fluid at right angles to these boundaries or sections at every point.
- *The law of pressure change:* As elevation increases, pressure decreases. If $p1$ is the pressure at elevation $y1$ and $p2$ the pressure at elevation $y2$, we have (using a force balance) $p2 - p1 = -\rho g(y2 - y1)$, where ρ is the density of a fluid and g is the acceleration due to gravity. Thus, $p1 = p2 + \rho g(y2 - y1)$.
- *Pascal's principal:* Pressure applied to an enclosed fluid is transmitted undiminished to every portion of the fluid and to the walls of the containing vessel.

To reason with these laws, we must first individuate fluid entities and then define the relationships between each type of individual (i.e., between fluid and fluid, and between fluid and solid). The key problem in reasoning about fluids is how to partition the fluid at an appropriate level of detail. Fluids should be divided into qualitatively different parts based on the reasoning task, and unnecessary partitions should be avoided. In addition, the fluid theory should be easily applied to each individual. Since interaction between fluids and other parts in

a system occurs only when they are in contact, the individuation of fluid should be based on the geometry of its surface contact with rigid objects, i.e., contact configuration of the system.

As a solid part moves it changes the configuration of the system. A fluid entity also changes the configurations of contact by its motion. Such changes, whether caused by solids or fluids, lead to new kinematic interactions between the parts. Thus each part continuously transmits and modifies both force and motion through the system.

4.2 Limitations of the Contained-Stuff Ontology

Most qualitative physics research has used the contained-stuff ontology for fluids. It individuates fluids using the natural boundaries provided by containers. A contained-liquid or contained-gas may disappear and reappear as the amounts of mass in a container changes from positive to zero or back again. The contained-stuff ontology provides a useful and intuitive notion for reasoning about the overall behaviors of fluids. However, it has been used in very restricted way: the fluid in a container is viewed as a single object.[1] Unfortunately this approach is not rich enough to capture certain important aspects of fluid behaviors. In this section, we describe some problems with the contained-stuff ontology and illustrate them with the lift pump example.

4.2.1 Sources of Problems

Two main sources of problems with the contained-stuff ontology are:

1. The concept of a container is hard to define. Consider a leaky container and a channel. Hayes (1985) claimed the former is a container but the latter is not. He also wrote "Unfortunately to describe these adequately requires metric ideas. (For example, a tin with a small hole in the base is a leaky tin, but a tin with the bottom missing isn't a container at all, although it could be a channel.)" However, even metric information does not seem to solve this problem. For example, how can we define"a small hole"? Can we define a small hole as a hole whose diameter is less than half, for instance, of the diameter of the base? Can we say Figure 4.2a is a container but Figure 4.2b is not? In the case of Figure 4.2c, a channel is connected with a tin. In this case, the upper channel contains the liquid, which means the channel is a container. We can find more examples which confuse the concept of a container. These arguments show that there might be no general rule to define a container upon which everyone can agree.

2. While the contained-stuff ontology provides a good conceptual division, individuating fluids only according to its container does not include sufficient geometry. Thus this ontology fails to represent fluids in all but the simplest geometries. A contained space, which can enclose fluid, is not necessarily the inside of a single container. In many situations, the inside of a container can be either split into distinct contained spaces or included as part of a contained space.

[1] From now on, in this chapter the contained-stuff ontology refers to this restricted view in order to distinguish it from the bounded-stuff ontology.

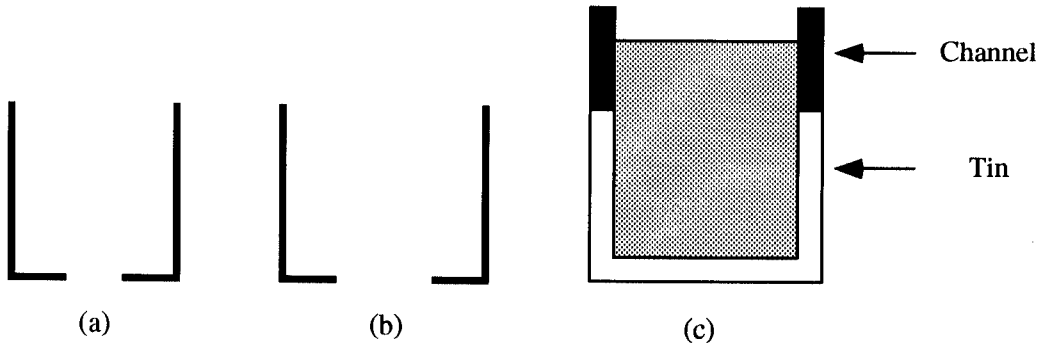


Figure 4.2: Container Examples

4.2.2 Lift Pump Example

We begin by describing a lift pump. Figure 4.3 depicts the sequence of states of a lift pump in operation. The pump's manual explains its behavior as follows:

> **Step 1**: Lifting up the piston opens valve(A), sucking water is up into the main chamber (Figure 4.3a).
>
> **Step 2**: Pushing the piston down closes valve(A) by force of the water. Water is lifted up to top chamber as valve(B) opens (Figure 4.3b).
>
> **Step 3**: Lifting the piston up again shuts valve(B) and pushes water out of the top chamber. Also water is sucked up into the main chamber as valve(A) opens (Figure 4.3c).
>
> **Step 4**: Pushing the piston down again opens valve(B) by force of water. Also water flows into the top chamber (Figure 4.3d).

As this explanation shows, the directions seemingly refer to the contained-liquid ontology to explain the behavior of the lift pump. However, we find that the strict contained-liquid ontology fails to explain the physical phenomena in this example. Let us look closer.

Step 1: As the piston is raised, the pressure exerted by the air in the main chamber to the water in the pump is decreased. This causes a pressure disturbance in the water in the main chamber, inlet pipe, and reservoir: the pressure difference is $(p_{air-main} + \rho g y_{main})$ minus $(p_{air-rsv} + \rho g y_{rsv})$, where $p_{air-main}$, $p_{air-rsv}$, y_{main}, and y_{rsv} represent the air pressure in the main chamber, in the water reservoir, the elevation of the top surface in the main chamber, and in the water reservoir, respectively. This pressure difference causes the top water surface in the main chamber to move up while that in the connected

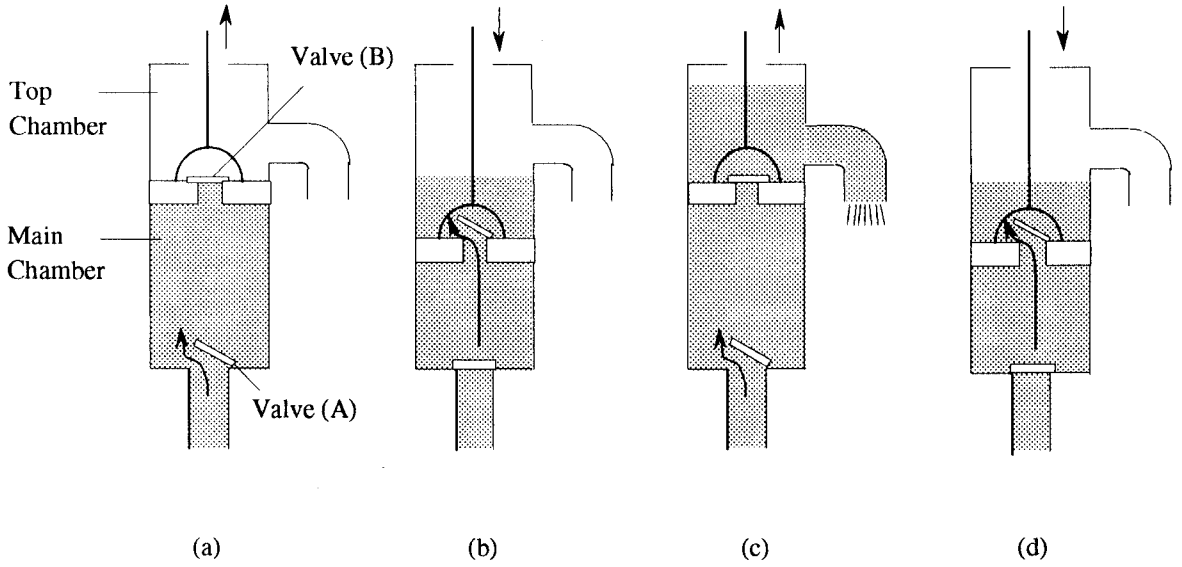


Figure 4.3: Lift Pump Example

water reservoir moves down. In the standard domain theory using the contained-stuff ontology, fluid motion is limited to fluid flow between two containers regardless of how the geometry of fluid containment changes. With the standard approach, it is difficult to reason about changes in configurations by pressure disturbances.

Step 2: To explain why pushing the piston closes valve(A), we need a way to compute pressures on the surfaces of valve(A). From the principles mentioned in Section 4.1, we can derive the pressure on the top surface p_{top}: $p_{top} = p_{water} + p_{air} + p_{piston}$, where p_{water}, p_{air}, and p_{piston} represent the pressure exerted by the water inside of both chambers, the pressure by the air inside of the top chamber, and the pressure by the force applied to the piston, respectively. This shows P_{top} is determined by the interaction with the water inside of both chambers; while the air and the piston are not in contact with the valve, p_{air} and p_{piston} are transmitted through the water since the water is in contact with them. In short, the fluid theory is applied to the water in both chambers during process. However, using the contained-stuff ontology the water is split into two distinct individuals. Step 4 is the same as step 2.

Step 3: Reasoning about net pressures on valve(A) and valve(B) also requires partitioning the water based on its containment, rather than the simple division by each container.

These problems are caused by lack of geometric information. In particular, the contained-stuff ontology does not represent (1) the surface geometry which constrains the fluid and (2) the effects of changes in geometric configurations on the fluid.

4.3 The Bounded-Stuff Ontology

Here we introduce a new ontology, the *bounded-stuff* ontology, or reasoning about fluids. The bounded-stuff ontology is a spatial generalization of the contained-stuff ontology. A fluid transmits forces and motions to its neighbors through physical contact. These interactions are determined by surface contact, and therefore by the configuration of the fluid within the container.

4.3.1 Extending Contained-Space with Boundaries

This section begins by examining the directions that the physical principles, mentioned in Section 4.1, suggest for representing fluids.

- *Pressure transmission:* Fluids influence other objects only through surface contact. Thus we need a way to represent a fluid in terms of the surface configuration of the objects which contact the fluid.
- *The law of pressure change:* The motions of adjacent rigid objects and the motions of fluids are derived by comparing all pressures applied to them. Since the pressures rely on the elevation of fluids, the geometric configuration plays a key role.
- *Pascal's principal:* This suggests viewing an enclosed fluid as an individual since a pressure disturbance applied to any part of the fluid is transmitted to every part of the fluid and its boundaries. Each enclosed fluid forms a qualitatively distinct place.

The bounded-stuff ontology is designed to address these requirements. It individuates fluids based on surface contacts with their boundaries. For example, the two contained-liquids inside the pump body in Figure 4.3 (b) are treated as a single liquid in our representation.

The bounded-stuff ontology assumes that rigid objects are described by sets of surfaces related via mechanical contacts. A piece of fluid is introduced by a contiguous set of boundaries in contact with some subset of these surfaces. For each surface, we represent the surface normal using a qualitative vector. The pressure exerted by the fluid is applied to a boundary in the opposite direction of the surface normal. This direction of the pressure is crucial to determining the motion of the boundary.

While the contained-stuff focuses on contained spaces formed by each container, ours covers a more general contained space: "some connected volume of three-dimensional-space which has a contiguous boundary (at least) below it and around it" (Hayes, 1985). A contained space which is full of fluid is treated as one object. Notice that the boundaries need not be the boundaries of a container. They might consist of either the boundaries of many containers or part of the boundaries of a single container. Each boundary may be the surface of a rigid object or even of another fluid.

As the parts of a system (either rigid objects or fluids) change their positions, the configuration of the system changes, and thus the contained spaces changes. For example, in the lift pump the motions of the valves and piston and water cause bounded-stuffs to disappear and

reappear. Our reasoning process derives the changes in the configuration, the bounded-stuff in each configuration, and their interaction with boundaries by applying formal, qualitative renderings of the principles above.

4.3.2 Finding Bounded-Stuff

Here we give some useful definitions for describing contained spaces and bounded-fluids.

Definition 4.1 (Connected) `Connected(pos1(`*c1*`),pos2(`*c2*`))` *is true if a position* `pos1` *of a container c1 is joined with* `pos2` *of c2.*

Position describes the part of an object such as the bottom of the main-chamber. For instance, `Connected(bottom(main-chamber),top(inlet-pipe))` means that the bottom of the main-chamber is connected with the top of the inlet-pipe in the liquid pump. Here we assume only two possible positions for a container: the *bottom* and the *top* of the container. While a lower part position is approximated by a bottom position, an upper part position is approximated by a top position. If two containers are connected, they can be either *aligned* or *not aligned.*

Definition 4.2 (Aligned) `Aligned(pos1(`*c1*`),pos2(`*c2*`))` *is true if* `Connected(pos1(`*c1*`),pos2(`*c2*`))` *is true and fluids can pass through a position* `pos1` *of a container c1 and* `pos2` *of c2.*

If there is a valve between two connected containers, its position determines whether or not they are aligned. Otherwise, they are always either aligned or not aligned. The inlet pipe and the reservoir connected to the pump, for example, are always aligned.

Definition 4.3 (Containers) `Containers(`*x*`)` *is the set of containers which form the boundaries of a contained space x.*

Definition 4.4 (Bottoms) `Bottoms(`*x*`)` *is a subset of* `Containers(`*x*`)` *whose bottoms are not aligned to any container in* `Containers(`*x*`)`.

Definition 4.5 (Tops) `Tops(`*x*`)` *is a subset of* `Containers(`*x*`)` *whose tops are not aligned to any container in* `Containers(`*x*`)`.

For instance, the main chamber belongs to $\texttt{Bottoms}(x)$ while the top chamber belong to $\texttt{Tops}(x)$, where x is the contained space formed by the surfaces of both chambers.

Suppose a system has n containers, c_1, c_2, ..., and c_n. Since the system keeps changing its configuration, a number of contained spaces are possible. They are found by incremental generation as follows.

1. For all c_i in $\{c_1, c_2, \ldots, \text{and } c_n\}$

   Generate *ContainedSpace*(x),
   where $\texttt{Containers}(x) = \{c_i\}$,
   $\texttt{Bottoms}(x) = \{c_i\}$, and $\texttt{Tops}(x) = \{c_i\}$

2. For every pair of contained spaces, $x1$ and $x2$, such that [$c1 \in$ **Containers**($x1$) $\bigwedge c2 \in$ **Containers**($x2$) $\bigwedge$ **Connected**(**pos1**($c1$))(**pos2**($c2$))] is true,

   Generate *ContainedSpace*($x3$),
   where **Containers**($x3$) = **Containers**($x1$) $\cup$ **Containers**($x2$)
   If **pos1** = *bottom* & **pos2** = *bottom*,
   Tops($x3$) = **Tops**($x1$) $\cup$ **Tops**($x2$)
   Bottoms($x3$) = **Bottoms**($x1$) $\cup$ **Bottoms**($x2$)
   If **pos1** = *bottom* & **pos2** = *top*,
   Tops($x3$) = **Tops**($x1$) $\cup$ **Tops**($x2$) $-$ $c2$
   Bottoms($x3$) = **Bottoms**($x1$) $\cup$ **Bottoms**($x2$) $-$ $c1$

The above algorithm produces every possible contained space. Suppose this is applied to the lift pump and top, main, pipe, and reservoir represent the top chamber, the main chamber, the inlet pipe, and the water reservoir, respectively. Then contained spaces whose containers are {top}, {main}, {pipe}, {reservoir}, {top, main}, {main, pipe}, {pipe, reservoir}, {top, main, pipe}, {main, pipe, reservoir}, and {top, main, pipe, reservoir} are found.

Actually, we are interested in finding a set of bounded-stuffs in each configuration. While a bounded-liquid refers to a contained space filled with liquid, a bounded-gas refers to a contained space filled with gas. Following definitions are used to find the bounded-stuffs.

Definition 4.6 (Length) **Length**(*x*) *is defined as the number of elements in a set x.*

Definition 4.7 (Full) **Full**(*c,sub*) *is true iff a container c is filled with substance sub.*

Definition 4.8 (Empty) **Empty**(*c,sub*) *is true iff there is no substance sub in a container c.*

Since gas molecules are widely spaced with negligible cohesive forces, a gas is free to expand until it encounters confining walls. Thus whenever a container c has a gas *sub*, **full**(c,*sub*) is true:

$\forall$*sub* $\in$ *gas*, $\forall c \in$ *containers*
Taxonomy (**Full**(c,*sub*), **Empty**(c,*sub*))

However, a liquid, being composed of relatively close-packed molecules with strong cohesive forces, tends to retain its volume and will form a free surface in a gravitational field if unconfined from above:

$\forall$*sub* $\in$ *liquid*, $\forall c \in$ *containers*
Taxonomy (**Full**(c,*sub*), **Empty**(c,*sub*), [$\sim$**Full**(c,*sub*) $\bigwedge$ $\sim$**Empty**(c,*sub*)])

Table 4.1 and Table 4.2 define bounded-liquids and bounded-gases in each configuration when more than two containers are connected. When there is only one container, a single bounded-stuff is defined for a substance.

$$
\begin{aligned}
&(\forall x, sub)\ [ContainedSpace(x) \bigwedge liquid(sub)\\
&\quad \bigwedge (\forall c1, c2)\ [c1 \in Containers(x) \bigwedge c2 \in Containers(x) \bigwedge Connected(top(c1), bottom(c2))\\
&\qquad \Rightarrow [Aligned(top(c1), bottom(c2)) \bigwedge Full(c1, sub)]]\\
&\quad \bigwedge (\forall c1, c2)\ [c1 \in Containers(x) \bigwedge c2 \in Containers(x) \bigwedge Connected(bottom(c1), bottom(c2))\\
&\qquad \Rightarrow [Aligned(bottom(c1), bottom(c2)) \bigwedge \sim [Empty(c1, sub) \bigwedge Empty(c2, sub)]]]\\
&\quad \bigwedge (\forall c1, c2)\ [c1 \in Containers(x) \bigwedge c2 \in Containers(x) \bigwedge Connected(top(c1), top(c2))\\
&\qquad \Rightarrow [Aligned(top(c1), top(c2)) \bigwedge Full(c1, sub) \bigwedge Full(c2, sub)]]\\
&\quad \bigwedge (\forall c1, c2)\ [c1 \in Containers(x) \bigwedge c2 \notin Containers(x) \bigwedge Connected(top(c1), bottom(c2))\\
&\qquad \Rightarrow [\sim Empty(c1, sub) \bigwedge\\
&\qquad\quad \sim [Aligned(top(c1), bottom(c2)) \bigwedge Full(c1, sub) \bigwedge \sim Empty(c2, sub)]]]\\
&\quad \bigwedge (\forall c1, c2)\ [c1 \in Containers(x) \bigwedge c2 \notin Containers(x) \bigwedge Connected(bottom(c1), top(c2))\\
&\qquad \Rightarrow [\sim Empty(c1, sub) \bigwedge \sim [Aligned(bottom(c1), top(c2)) \bigwedge Full(c2, sub)]]]\\
&\quad \bigwedge (\forall c1, c2)\ [c1 \in Containers(x) \bigwedge c2 \notin Containers(x) \bigwedge Connected(bottom(c1), bottom(c2))\\
&\qquad \Rightarrow [\sim empty(c1, sub) \bigwedge \sim Aligned(bottom(c1), bottom(c2))]]\\
&\quad \bigwedge (\forall c1, c2)\ [c1 \in Containers(x) \bigwedge c2 \notin Containers(x) \bigwedge Connected(top(c1), top(c2))\\
&\qquad \Rightarrow [\sim Empty(c1, sub) \bigwedge\\
&\qquad\quad \sim [Aligned(top(c1), top(c2)) \bigwedge Full(c1, sub) \bigwedge Full(c2, sub)]]]]\\
&\quad \Longleftrightarrow Bounded - Liquid(x, sub)
\end{aligned}
$$

Table 4.1: The definition of bounded-liquid

$$
\begin{aligned}
&(\forall x, sub)\ [ContainedSpace(x) \bigwedge gas(sub)\\
&\bigwedge (\forall c1, c2, pos1, pos2)\\
&\quad [c1 \in Containers(x) \bigwedge c2 \in Containers(x) \bigwedge Connected(pos1(c1), pos2(c2))\\
&\quad \Rightarrow [Full(c1, sub) \bigwedge Full(c2, sub) \bigwedge Aligned(pos1(c1), pos2(c2)) \bigwedge\\
&\qquad Equal(pressure(c1), pressure(c2))]]\\
&\bigwedge (\forall c1, c2, pos1, pos2)\\
&\quad [c1 \in Containers(x) \bigwedge c2 \notin Containers(x) \bigwedge Connected(pos1(c1), pos2(c2))\\
&\quad \Rightarrow [Full(c1, sub) \bigwedge [\sim Aligned(pos1(c1), pos2(c2)) \bigvee \sim Equal(pressure(c1), pressure(c2))]]]]\\
&\Longleftrightarrow Bounded - Gas(x, sub)
\end{aligned}
$$

Table 4.2: The definition of bounded-gas

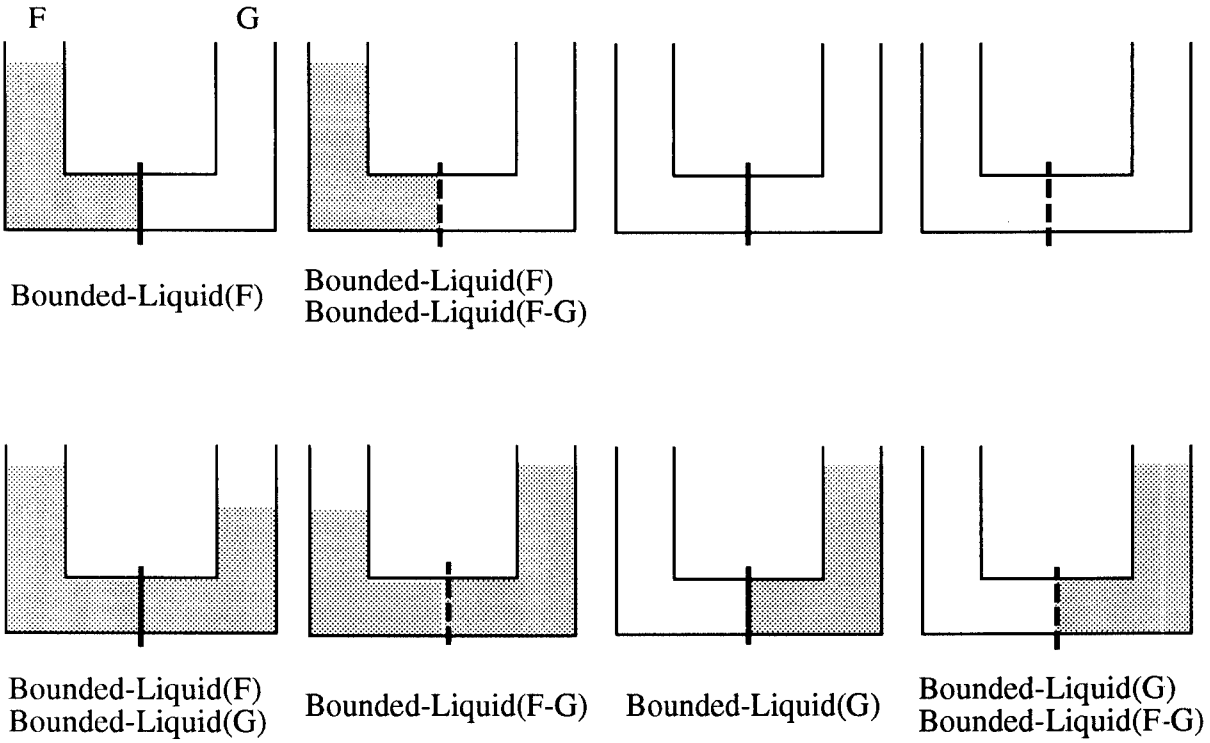


Figure 4.4: Bounded-liquids Examples: Parallel aligned containers

Figure 4.4 and Figure 4.5 shows all possible bounded-liquids when two containers are connected with a valve between them. Bounded-gases are similarly illustrated. While the bottoms of two containers are connected in Figure 4.4, two container are vertically connected in Figure 4.5. In the figures, a dashed line represents the valve is open while a straight line means it is closed. In Figure 4.4, when both F and G contain liquid, three configurations are possible depending on the ordinal relations between the heights of the liquid in F and G (i.e., $>$, $=$, and $<$). Since the same bounded-liquids are generated regardless of the ordinal relations, the figure shows only one of the three configurations. In Figure 4.5, when G contains liquid and the valve is closed, the bounded-liquids in that situation are not relevant to whether G is `Full` or not does not. The figure includes only when G is not `Full`.

When the boundary of a bounded-liquid coincides with the boundary between containers, that is a container $c1$ is full of liquid *sub*, a container $c2$ is aligned with $c1$, and $c2$ is empty, the liquid is viewed in two ways: (1) the bounded-liquid whose boundaries does include $c2$, (2) the bounded-liquid whose boundaries does not include $c2$ (see the cases where two views are active in Figure 4.4 and Figure 4.5). These cases are analyzed further in the following section.

If a system has n containers, `Length(Containers(`x`))`, where x is any contained space in the system, can vary from 1 to n. Then the maximum number of possible contained spaces

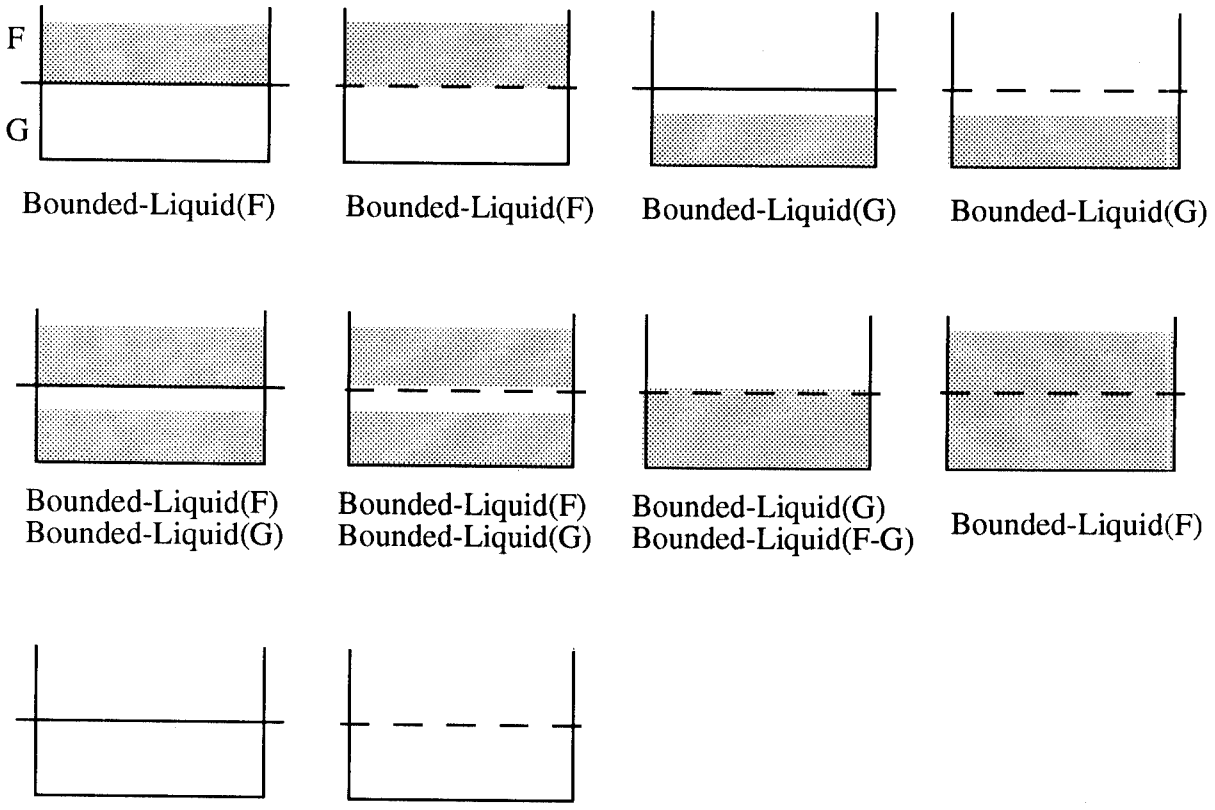


Figure 4.5: Bounded-liquids Examples: Vertically aligned containers

is ${}_nC_1 + {}_nC_2 + \ldots {}_nC_n$, approximately 2^n. This happens when every container in a system is connected to all of the containers, which is is very hard to imagine, especially for large n. However, in many system the number of connections is linear in n. If we consider systems with $n-1$ connections, i.e., the liquid pump, the maximum number is $n + (n-1) + \ldots$ 1, i.e, $n(n-1)/2$. If two containers are always aligned (e.g., the inlet pipe and the water reservoir in the lift pump), this number is further reduced. For instance, contained spaces which include either the inlet pipe or the water reservoir but not both are impossible. Any contained space includes both of them or neither of them. Conceptually, we can view the two containers as a single container in counting the number of possible contained spaces, since the two always belong to the same contained space.

4.3.3 Reasoning about Fluid

After identifying the bounded-stuffs in each configuration of a system, the fluid behavior and subsequent effects on boundaries are predicted by analyzing how fluid and boundaries interact along with any external forces applied to the system. In our theory, we focus on transmissions of pressures and motions between fluid and its boundaries. We concentrate on how fluid motions bring about changes in the configuration of a system.

Since gas has a very small density, pressure changes due to height can be neglected. Thus for gases, pressure differences due to geometry can be considered insignificant. In our representation, the pressure at any point of a bounded-gas is assumed the same; the pressure exerted by the gas on any boundary is thus assumed identical regardless of the location of the boundary. These pressures change when external pressure is applied or gas flow occurs between aligned gases (or to an empty container). Gas flow occurs from the bounded-gas with higher pressure to the bounded-gas with lower pressure until their pressure becomes equal. If the two gases are still aligned after the flow, they are combined into a new bounded-gas.

In the case of liquids, which have much larger densities than gases, pressure differences due to geometry are significant. Thus in our bounded-stuff ontology both the pressure inside a liquid and the pressure exerted on its boundary are computed based on its geometry.

To capture how the surface contact configuration in a system changes during liquid motions, we focus on the motion of top surfaces. (A top surface will be a free surface unless confined from above.) Once we know the position of a bounded-liquid top surface, the contained space below its top surface and around the surface can be determined from the geometric structure of the system.

For each top surface, we determine

- is the top surface prevented from moving in an up or down direction?
- what is the net force on the top surface?

The top surfaces of a bounded-liquid are located inside the containers in `Tops`(cs), where cs is the contained-space containing the liquid. `Top-surface`(c,bl) refers to the top surface of a bounded-liquid bl at a container c.

4.3.3.1 Translational Freedom of Top Surfaces

We extend the representation of translational freedom of Nielsen(1988) from rigid bodies to include liquids. By default, a top surface is free to move in some direction unless it is constrained. In the following definition, direction *dir* is either up or down.

Definition 4.9 (TransConstraint) `TransConstraint`*(t_i,dir) is true when a top surface t_i is prevented from moving in direction dir.*

Definition 4.10 (TransFreedom) `TransFreedom`*(t_i,dir) is true when a top surface t_i is not prevented from moving in direction dir.*

Definition 4.11 (Open-top) `Open-top`*(c) is true iff the top of a container c is open and is not connected to any container.*

We only consider the translational freedom of either up or down motion in a gravitational field. The translational constraint of top surfaces is found as follows. Figure 4.6 illustrates each case graphically.

For each top surface t_i, where t_i is `top-surface`(c,`bounded-liquid`(cs,sub))

1. If `Empty`(c,sub), `Transconstraint`(x,$down$) is true.
2. If `Bottom-of`(c,t_i) is not aligned to any container and the bottom is closed, `Transconstraint`(t_i,$down$) is true.
3. If `Full`(c,sub) is true and c is aligned to some container, `Transconstraint`(t_i,up) is true.
4. If `Full`(c,sub) is true and c has an open top, `Transconstraint`(t_i,up) is true.
5. If `Full`(c,sub) is true and the rigid body which is in contact with t_i is sufficiently constrained to move in the up direction, `Transconstraint`(t_i,up) is true.
6. If `Transconstraint`(t_j,up) is true for every t_j, where t_j is a top surface of `bounded-liquid` (cs,sub) and $t_i \neq t_j$, `Transconstraint`(t_i,$down$) is true.
7. If `Transconstraint`(t_j,$down$) is true for every t_j, where t_j is a top surface of `bounded-liquid` (cs,sub) and $t_i \neq t_j$, `Transconstraint`(x,up) is true.

4.3.3.2 Net Pressure on Top Surface

The net force on a top surface is determined by comparing the downward and upward pressures at the top surface. These pressures are described by quantities `down-pressure`(top-surface ?top ?BL) and `up-pressure`(top-surface ?top ?BL). By the pressure change principle, we can find the upward pressure on a top surface by comparing with the pressure on the other top surface in the fluid (Figure 4.7). We ignore the effect of velocity of the fluid on the pressure. If the net pressure (i.e., net force) is non-zero and the top surface is free to move the direction of the net force, then it moves in that direction.

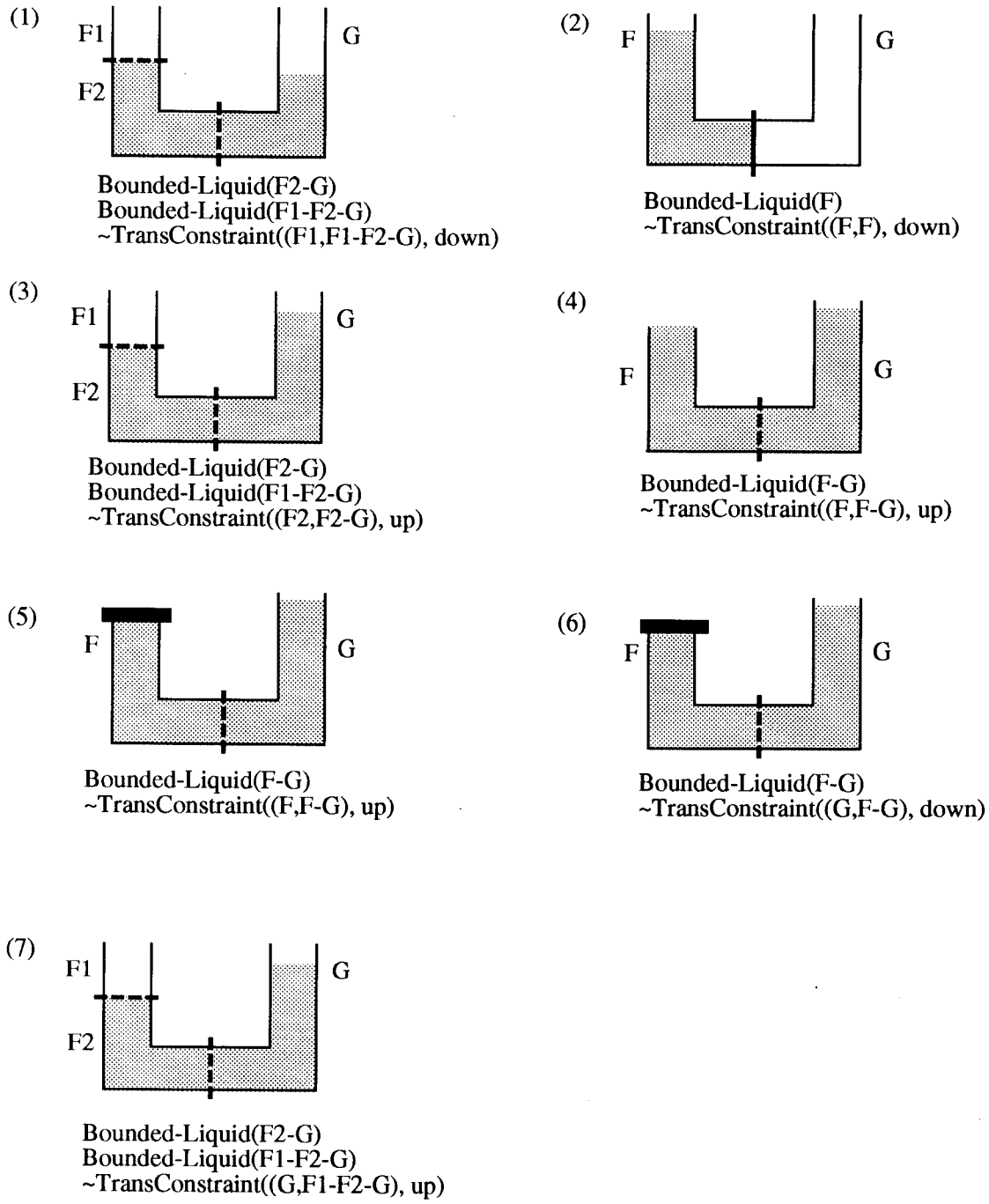


Figure 4.6: `TransConstraint` of Top Surface. For the sake of convenience, substances are omitted in representing bounded-liquids. In describing top-surfaces, bounded-liquids are represented by contained-spaces. Dashed lines and straight lines represent the two containers are aligned and not aligned, respectively.

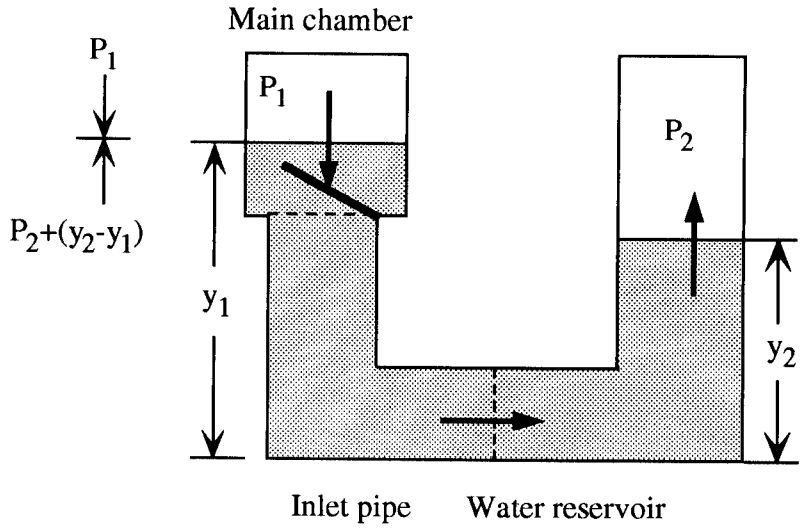


Figure 4.7: Liquid Pump (Top chamber is not shown here.) The water in the main chamber, inlet pipe, and water reservoir is viewed as one bounded liquid. The motion of the free surface in the main chamber is determined by comparing its downward pressure and the upward pressure from the reservoir. The motion of the free surface in the reservoir is similarly computed.

The effects of a fluid on its boundaries are determined by computing the net forces applied to them. Since fixed boundaries are assumed not to move, only the motions of movable boundaries are determined. The net force on a movable object by a liquid is computed by comparing the pressure at the object with the pressures at the free surfaces of the liquid. For instance, the upward force on the valve Figure 4.7 is proportional to $p2 + y2$ while the downward force is proportional to $p1 + y1$. When the valve is closed, the downward force is computed from the pressure and height of the free surface of the upper bounded-liquid. The upward force is computed from the free surface of the lower bounded-liquid.

In addition to comparing the pressures and the heights of top surfaces, the force applied to an enclosed fluid should be also considered. In Figure 4.3b the additional downward force of the piston is added to the force on the upper part of valve(a) due to the Pascal's principle.

Since our contained spaces are described in terms of boundaries and the boundaries are easily related to their containers, the fluid motion is still understood as changes in containers.

The `net-pressure` of a top surface influences the `velocity` of the surface. The `velocity` in turn changes the `position` of the surface. The amount of liquid in a container changes as the `position` of the top surface in the container changes. Figure 4.8 shows the process for a top surface motion and the relation between its motion to the change of contained-stuff. `(B-S ?sub liquid ?bnd)` express `Bounded-Liquid (?bnd,?sub)`. If a container `?top` belong to `Tops(?bnd)`, then `(top-surface ?top (B-S ?sub liquid ?bnd))` is true. `Bottom-of(?top ?bnd)` provides a container which is below `?top` in `?bnd` and belongs to `Bottoms(?bnd)`.

```
(Defprocess (Surface-Up-motion ?top ?sub ?bnd)
  Individuals
    ((?top  :type container)
     (?sub  :type substance)
     (?bnd  :type contained-space
            :conditions
            (top-surface ?top (B-S ?sub liquid ?bnd)) )
     (?top2 :type container
            (top-surface ?top2 (B-S ?sub liquid ?bnd))
            (connected (bottom (bottom-of ?top ?bnd))
               (bottom (bottom-of ?top2 ?bnd)))) )
  PreConditions
    ((transfreedom (top-surface ?top (B-S ?sub liquid ?bnd)) up))
  QuantityConditions
    (;;; The downward-pressure is less than the upward-pressure
     (less-than
      (A (pressure (top-surface ?top (B-S ?sub liquid ?bnd))))
      (A (pressure (top-surface ?top2 (B-S ?sub liquid ?bnd)))))))
  Relations
    ((quantity flow-rate)
     (Q= flow-rate ;;; (- upward-pressure downward-pressure)
         (- (pressure (top-surface ?top2 (B-S ?sub liquid ?bnd)))
            (pressure (top-surface ?top (B-S ?sub liquid ?bnd))))) )
  Influences
    ((I+ (velocity (top-surface ?top (B-S ?sub liquid ?bnd)))
         (A flow-rate)) )


;;; Relation for connecting the top-surface motion to the change
;;; in the amount of the container which include the top-surface
(Qprop (amount-of-in ?sub LIQUID ?top)
       (velocity (top-surface ?top ?B-L)))
```

Figure 4.8: A process about free surface motion in bounded-liquid.

4.3.4 The Lift Pump Example: Reprise

This theory has been implemented and tested for several examples, including the lift pump. This implementation has been incorporated into QSA as a domain theory of fluids. Now, we show how the problems of the contained-stuff ontology are solved, using the lift pump example: The simulation process starts with a scenario describing the lift pump, which is given in Figure 4.9. In Chapter 8, the envisionment of this example will be described in detail.

Step 1: The upward fluid motion in the inlet pipe and the downward motion in the reservoir are easily captured by computing the pressure disturbances in the bounded water in the main chamber, inlet pipe, and water reservoir. This net pressure is computed by applying the law of pressure change to the water Figure 4.10 shows part of state description for Figure 4.3 (a).

Step 2: The water in the main chamber and the fluid in the top chamber belong to the same bounded-stuff in Figure 4.3b. Since the valve(A) is at the bottom of the fluid, the downward force by the water is proportional to level of the fluid, i.e., the sum of the fluid levels in the main chamber and the top chamber. Since the downward force applied to the piston is transmitted to the water by Pascal's principal, the force is also added to the top of the valve(A). At the same, since the bottom of the valve is in contact with the water in the inlet-pipe and reservoir, the upward force on the valve is proportional to the level of the water in the reservoir.

Step 3: First, the closing of the valve(A) is explained by computing the net pressure on the valve exerted by the water in both chambers. Since the downward external force of the piston does not exist any more, the valve is closed due to its weight. Then the opening of the valve(B) is explained by comparing the downward pressure by the water in the main chamber and the upward pressure by the water in the inlet pipe and reservoir.

```
(open-container reservoir)
(container main-chamber)
(container top-chamber)

;;;----- Piston
(wall piston)
(mobile piston)
(part-of (bottom piston) main-chamber)
(part-of (top piston) top-chamber)
(pump-handle handle piston)

;;;----- Portal
(portal-of P1 top-chamber)
(less-than (height portal1) (top-height top-chamber))
(less-than (bottom-height top-chamber) (height portal1))


;;;----- Valve(a)
(valve va (bottom main-chamber) (bottom reservoir))
(valve va (bottom reservoir) (bottom main-chamber))
(part-of (right va) reservoir)
(part-of (left va) main-chamber)
(rot-constraint va ccw)
(connected (bottom reservoir) (bottom main-chamber))
(connected (bottom main-chamber) (bottom reservoir))

;;;----- Valve(b)
(connected (bottom top-chamber) (top main-chamber))
(connected (top main-chamber) (bottom top-chamber))
(valve vb (bottom top-chamber) (top main-chamber))
(valve vb (top main-chamber) (bottom top-chamber))

;;;----------
(substance water)
(substance air)

(Can-Contain-Substance reservoir water liquid)
(Can-Contain-Substance main-chamber water liquid)
(Can-Contain-Substance main-chamber air gas)
(Can-Contain-Substance top-chamber water liquid)
(Can-Contain-Substance top-chamber air gas)

;;;----- Water reservoir always has water inside
(greater-than  (amount-of-in water liquid reservoir) ZERO)
```

Figure 4.9: The lift pump scenario description.

```
~(aligned (bottom top-chamber) (top main-chamber))
~(aligned (bottom main-chamber) (bottom reservoir))
(TransFreedom (top-surface (main-chamber
                    (B-S water liquid main-pipe-reservoir))) up)
(TransFreedom(top-surface (main-chamber
                    (B-S water liquid main-pipe-reservoir))) down)
(Bounded-Liquid (B-S water liquid main-pipe-reservoir))

Ds[Flow-rate(PI0)] = -1
Ds[position(piston)] = 1
Ds[velocity(piston)] = 1
Ds[velocity(top-surface (main-chamber
                   (B-S water liquid main-pipe-reservoir)))] = 1
Ds[pressure(top-surface (main-chamber
                   (B-S water liquid main-pipe-reservoir)))] = 1
Ds[pressure(top-surface (reservoir
                   (B-S water liquid main-pipe-reservoir)))] = -1
Ds[pressure(top-chamber)] = 1
Ds[fluid-level(main-chamber)] = 1
Ds[amount-of-in(water liquid main-chamber)] = 1
Ds[amount-of-in(water liquid reservoir)] = -1

(Less-than (pressure main-chamber) (pressure reservoir))
(Greater-than (fluid-level main-chamber) (fluid-level reservoir))
(less-than
   (pressure (top-surface (main-chamber
                    (B-S water liquid main-pipe-reservoir))))
   (pressure (top-surface (reservoir
                    (B-S water liquid main-pipe-reservoir)))) )
(Greater-than (velocity (top-surface (main-chamber
                    (B-S water liquid main-pipe-reservoir))))
              ZERO)
(Greater-than (velocity piston) ZERO)
(less-than (pressure main-chamber) (pressure top-chamber))

Active (PI0)
Active (PI1)

PI0: SURFACE-UP-MOTION(main-chamber,water,main-pipe-reservoir)
;;; upward force applied from the handle to piston
PI1: ACCELERATION(piston,handle)
```

Figure 4.10: Part of state description for Figure 4.3 (a). Ds denotes the sign of the derivative.

```
(aligned (bottom top-chamber) (top main-chamber))
~(aligned (bottom main-chamber) (bottom reservoir))
(TransFreedom (top-surface (top-chamber
                    (B-S water liquid main-pipe-reservoir))) up)
~(TransFreedom(top-surface (main-chamber
                    (B-S water liquid main-pipe-reservoir))) down)
(Bounded-Liquid (B-S water liquid top-main))
(Bounded-Liquid (B-S water liquid pipe-reservoir))

Ds[position(piston)] = -1
Ds[velocity(piston)] = -1
Ds[pressure(top-surface (top-chamber
                   (B-S water liquid main-pipe-reservoir)))] = 1
Ds[fluid-level(top-chamber)] = 0
Ds[amount-of-in(water liquid top-chamber)] = 1
Ds[amount-of-in(water liquid main-chamber)] = -1

(Greater-than (fluid-level top-chamber) (fluid-level reservoir))
(less-than (fluid-level top-chamber) (height portal1))
(equal-to (external-force (B-S water liquid pipe-reservoir))
          ZERO)
(greater-than (external-force (B-S water liquid top-main)) ZERO)
(greater-than
   (pressure (top-surface (top-chamber
                    (B-S water liquid top-main))))
   (pressure (top-surface (reservoir
                    (B-S water liquid pipe-reservoir)))) )
(equal-to
   (velocity (top-surface (top-chamber
                    (B-S water liquid top-main)))) zero)
(less-than (velocity piston) ZERO)
(less-than (pressure top-chamber) (pressure reservoir))
```

Figure 4.11: Part of state description for Figure 4.3 (b).

Chapter 5

Geometry of Fluid Flow

In this chapter, we focus on reasoning about directions of fluid flows. Understanding how fluid flows change their directions around solid objects is important in explaining the motions of objects. While our main examples can be explained without this due their structures,[1], some interesting physical phenomena, such as the lift of airfoil, cannot be explained without understanding flow directions. (The lift will be illustrated in this chapter later.)

Consider the physical situation in Figure 5.1. The valve in the middle of the right side is open and the pressure inside the cylinder is greater than the pressure outside. Suppose we want to explain the motion of the gas in the cylinder. People can conclude that the gas near the top surface moves downward to the right and the gas near the bottom surface moves upward to the right, and so on. This chapter describes a formalism which can derive such inferences.

Fluid flow is determined by pressure differences in a fluid and the surface geometry of rigid bodies in contact with the fluid. The previous chapter described how fluid motion changes the contact configuration of a fluid. Once a pressure disturbance occurs in a bounded-fluid, every point in the fluid starts to move (assuming the fluid is free to move or is compressible).

[1] Most rigid parts are fixed and the effects due to the changes of fluid flow directions to mobile parts are negligible.

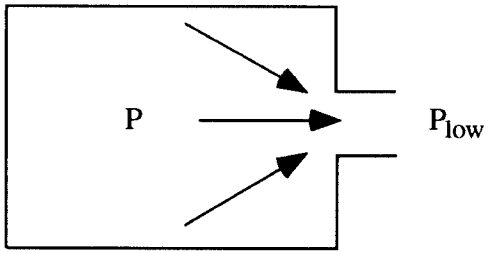


Figure 5.1: Gas flow inside a cylinder

The direction of the net pressure at each point determines the direction of flow at that point. We present a technique for reasoning about directions of fluid flows in two-dimensions using incremental generation of the *places* in a bounded-fluid.

We extend the work on places of (Forbus et al., 1991; Nielsen, 1988; Forbus, 1981) from the rigid body domain to the fluid domain. The theory defines a *place* as a region of space in which every point has the same qualitative direction of flow and each place has a qualitatively distinct direction of flow relative to its neighbors. While there is a single place in a bounded-fluid at rest, there might be many places in a moving fluid. Bounded-fluids are divided into places by analyzing the contact configurations of fluids and pressure disturbances in the fluids. The fluid is partitioned so each part has the same qualitative fluid motion.

Section 5.1 presents the theory for reasoning about flow direction in qualitative and geometric terms, given a pressure change. Section 5.2 explains how to qualitatively simulate the behaviors of fluid in each part. Finally, we show how our flow analysis explains the lift of airfoil from its curvature.

5.1 A Qualitative Theory of Fluid Motion

We begin in Section 5.1.1 by describing the two properties of fluids which are central to this part of our theory. Section 5.1.2 shows how places are incrementally generated. In Section 5.1.3, we describe a supplementary method which is used when the method shown in Section 5.1.2 is not sufficient to generate places.

5.1.1 Pressure Wave Propagation and Continuous Change

There are two central properties of fluids used in our theory of fluid flows. We describe each in turn.

Pressure Wave Propagation (PWP): When a pressure disturbance occurs in a compressible fluid, the disturbance travels with the velocity of sound. If the disturbance is due to lower pressure, then an *expansion* wave is propagated. If it is due to higher pressure, then a *compression* wave occurs. The pressure wave moves radially outward from the starting point. As the pressure wave is propagated through a still fluid, the fluid properties (i.e., pressure, temperature, and density and so on) change and it starts to move. As a compression wave is propagated, the fluid molecules have a velocity which has the same direction as the wave propagation. On the other hand, when an expansion wave travels, the fluid has a velocity which has the opposite direction of the wave (i.e., toward the source of the disturbance). The induced velocity of the fluid by wave propagation is much slower than the wave propagation.

Definition 5.1 (PWP-Constraint) *Suppose a surface* `s` *is in contact with fluid.* `PWP-Constraint(s,d)` *is true when PWP is prevented in direction d near* `s`.

Law 5.1 (Surface-Constraint) *Suppose an immovable surface* `s` *is contact with fluid and its surface normal is sn. Then pressure waves cannot propagate from the surface to the fluid. Thus for every d which belongs to* `Open-Half-Plane(sn)`, `PWP-Constraint(s,d)` *is true.*

Continuous Change: We assume the flow is smooth and steady (*laminar*). When flow is not laminar but fluctuating and agitated (*turbulent*), it is impossible to explain the behavior. Even in fluid mechanics, no general analysis of fluid motion in turbulence yet exists and there may never be. People also have difficulty explaining the direction of the turbulent flow. In laminar flow, the changes of properties are continuous. To make the assumption of laminar flow reasonable, we assume surfaces are smooth and the changes in surface are not abrupt.

5.1.2 Place Generation

To see how we might generate places, let us compare our problem to an already solved problem. FROB (Forbus, 1981) was built to understand motion in space. Given a geometric description of surfaces, it generates the places needed to envision the possible motions of a ball. Space is split into places by the geometric constraints of surface and gravity. Since gravity is the same everywhere, space can be divided without regard to neighbors. The situation with fluids is substantially more complex. Since the direction of `PWP` can change because of the surface geometry of rigid bodies as waves propagate, places cannot be generated without considering the interactions between pressure waves and surfaces. Even though two given fluids have the same geometry, they can be partitioned in completely different ways with different direction pressure waves.

Since the direction of a pressure wave determines the direction of flow, places in our reasoning problem should be distinguished by the pressure wave direction in each part. The continuous interaction of pressure waves and geometry suggests our place generation should be incremental as the pressure wave propagates from the source of a disturbance.

Definition 5.2 (PWP-Place) *A* `PWP-Place` *is defined by its boundaries, the direction of the pressure wave, and the direction of the induced velocity. Given a* `PWP-Place` *P,* `Pres-Wave(P)` *refers to the the direction of pressure wave in P. The boundaries represent the adjacent* `PWP-Places` *or adjacent surfaces of rigid bodies.*

Definition 5.3 (Places) `Places(s)` *maps from a surface* `s` *to the* `PWP-Places` *in which* `s` *is a boundary.*

The following function is defined to select the place around an end-point of a surface `s` among `Places(s)`.

Definition 5.4 (Place) `Place(s,p)` *maps from a surface* `s` *and its end-point* `p` *to the* `PWP-Place` *around* `p` *in which* `s` *is a boundary.*

Law 5.2 (Connectivity of Place) *Given two adjacent surfaces* `s1` *and* `s2`, `Place(s1,p)` *and* `Place(s2,p)`, *where* `p` *is their common end-point, are also adjacent. Furthermore,* `Places(s1)` *lie to the* `Surf-Rel-Pos(s1,s2)` *direction of* `Places(s2)`.

Since fluid motion is caused by pressure disturbances, we assume a pressure disturbance as input. In our approach, places are generated incrementally from the initial pressure change in

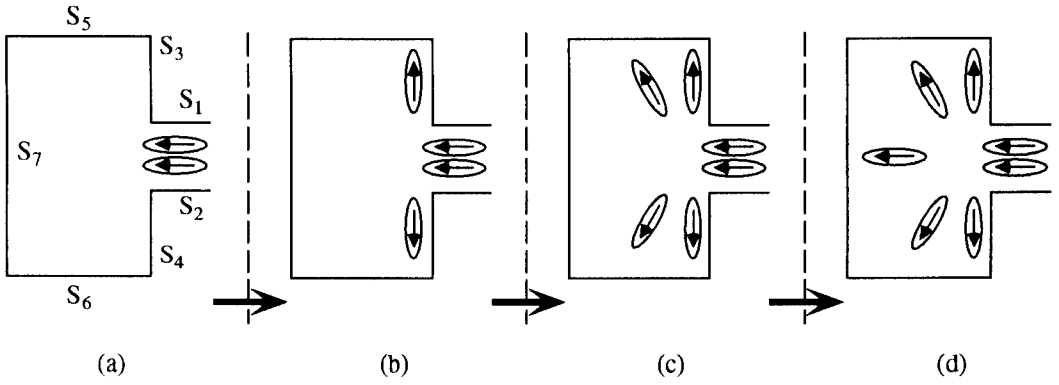


Figure 5.2: Incremental place generation in a cylinder. An oval represents the a place generated.

the direction of PWP. Since the surfaces of rigid bodies can be the only explicit boundaries of fluids, our approach first generates the places near the surfaces by propagating the pressure wave across pairs of adjacent surfaces. The places of the space which are not bounded by the surface are not generated at first since it is impossible to trace every point and give the boundary of the place. But as places around the surfaces are generated, the whole space can be covered by the continuous change property of fluid.

Figure 5.2 graphically shows this process in a cylinder. When the cylinder valve opens, the pressure inside the cylinder is greater than the outside pressure. The expansion wave is propagated from the outside to `s1` and `s2` first since they are closest to the outside. Then places are generated around these surfaces (Figure 5.2a). After that, places near `s3` and `s4` are generated as the pressure wave continues traveling (Figure 5.2b). Figures 5.2c and d show the remaining places.

To propagate a pressure wave across pairs of adjacent surfaces, the first step is to determine the propagation order between the adjacent surfaces. Given a newly generated place, our system checks how the pressure wave can propagate toward the adjacent surface.

Definition 5.5 (Forward-Propagation?) *Suppose* `s1` *and* `s2` *are two adjacent surfaces and their end-points are* `(p1,p2)` *and* `(p2,p3)`. *If the* `Pres-Wave(Place(s1,p2))` *belongs to* `Open-Half-Plane(Relative-Position(p2,p1))`, *then* `Forward-Propagation?(s2,s1)` *is true. Otherwise, it is false.*

As Figure 5.3 shows, when `Forward-Propagation?(s2,s1)` is true, we can infer that the source of the disturbance is not closer to `s2`. Thus the following law is introduced.

Law 5.3 (Forward Propagation) *Suppose* `Forward-Propagation?(s2,s1)` *is true and the end-points of* `s1` *and* `s2` *are* `(p1,p2)` *and* `(p2,p3)`. *Then there are two cases in which* `s2` *belongs*

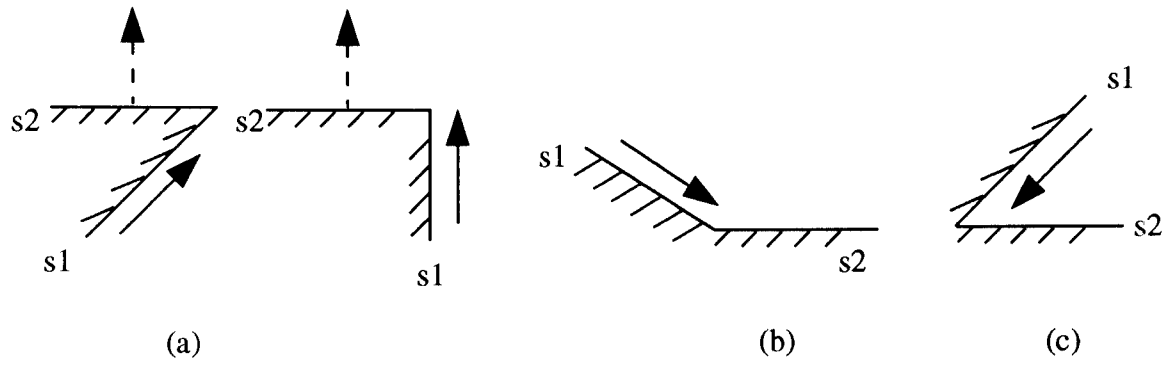


Figure 5.3: Examples of `Forward-Propagation?(s2,s1)`, where `Surface-Normal(s2)` = (0+)

to next wave front of `s1`*: (1) blocked:* `Pres-Wave(Place(s1,p2))` *belongs to* `Open-Half-Plane (Surface-Normal(s2))` *(Figure 5.3a), (2) further:* `s2` *is not blocked and* `Pres-Wave-Place (s1,p2)` *belongs to* `Open-Half-Plane(Relative-Position(p3,p2))` *(Figure 5.3b).* `s2` *belongs to same wave front of* `s1` *if* `s2` *is not blocked and* `Pres-Wave(Place(s1,p2))` *belongs to* `Open-Half-Plane(Inverse(Relative-Position(p3,p2)))` *(Figure 5.3c).*

Law 5.4 (Free Propagation) *Suppose* `Forward-Propagation?(s2,s1)` *is true and the end-points of* `s1` *and* `s2` *are* `(p1,p2)` *and* `(p2,p3)`. *If* `s2` *is not blocked, then* `Pres-Wave(Place(s2, p2))` *from the source belongs to* `Open-Half-Plane(Pres-Wave(Place(s1,p2)))`.

A pressure wave travels from the source of a disturbance in all directions unless it is blocked by a rigid body surface. Unless the direction of `PWP` is changed by a surface, then `Pres-Wave` of any point can be simply inferred as the direction from the source to that point. Suppose `Forward-Propagation?(s2,s1)` is true and `PWP` is not *blocked.* If `s2` belongs to the *next wave front* of `s1`, then a single place is generated around `s2` since `PWP` proceeds along the surface without being blocked by the surface. The `Pres-Wave` of the place, the direction from the source of a disturbance to the surface `s2` is computed by adding `Pres-Wave(Place(s1,p2))` and `Relative-Position(p3,p2)` by the law of Transitivity of Relative-Position. In the case of *same wave front*, `Pres-Wave(Place(s2,p2))` has the same direction as `Pres-Wave(Place(s1,p2))`. Thus these two places are merged since they have the same `PWP` direction.

The law of free propagation shows that if `PWP` is not blocked, then it smoothly changes its direction across the adjacent surfaces. However, if `PWP` is blocked by surface(s), it might abruptly change the direction across adjacent surfaces (Figure 5.2b). It starts to proceed as if a new pressure disturbance is generated. Figure 5.4 shows how a new source is generated when wave arrives at point `A` from the source.
We identify instances when a new source is generated.

Law 5.5 (New Source) *Suppose* `Forward-Propagation?(s2,s1)` *is true and the end-points of* `s1` *and* `s2` *are* `(p1,p2)` *and* `(p2,p3)`. *Then there are two cases in which a new source is*

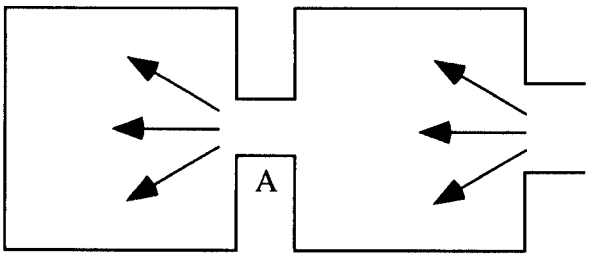


Figure 5.4: New Source

generated: (1) Blocking: If `s2` *is blocked, then the new source is generated around* `p2`. *(2) First-Moving: If* `s2` *is further than* `s1` *from the source, then the new source might be generated in* `Places(s1)`. *Since the fluid in* `Places(s1)` *starts to move earlier than that of* `s2`, *this may cause a pressure disturbance.*

When a new source is generated, the pressure wave propagates in the direction parallel to the adjacent surface. The `Relative-Position` (p3,p2) becomes `Pres-Waves` of `Places(s2)`.

5.1.3 Inferring Propagation in Backward Direction

Since our approach propagates a pressure wave across connected surfaces and divides the space based on the places near the surface, it may not suffice given a more complicated geometry. For example, in Figure 5.5 when `Places(s1)` is generated, a new place cannot be generated since there is no surface adjacent to `s1` in the forward direction of `PWP`.

However, by using backward reasoning with forward `PWP`, this problem can be solved. As we can expect, this reverse inference may lead to ambiguities. Since our technique does not use any metric information, several possibilities may be introduced. But by using some characteristics of fluids we can formulate constraints which eliminate many ambiguities.

Definition 5.6 (Backward-Propagation?) *Suppose* `s1` *and* `s2` *are two adjacent surfaces and their end-points are* `(p1,p2)` *and* `(p2,p3)`. *If the* `Pres-Wave(Place(s1,p2))` *belongs to* `Open-Half-Plane(Inverse(Relative-Position(p2,p1)))`, *then* `Backward-Propagation?(s2, s1)` *is true.*

If `Backward-Propagation?(s2,s1)` is *true*, then `Forward-Propagation?(s1,s2)` is *true*. Thus `s1` belongs to the next wave front or the same wave front of `s2`. We identified how to infer whether `s2` belongs to previous or same wave front of `s1` from the geometric analysis for all possible cases.

Law 5.6 (Backward Propagation) *If* `Backward-Propagation?(s2,s1)` *is true, the relationship between* `s1` *and* `s2` *in the propagation of a pressure wave is: (1) same wave front:*

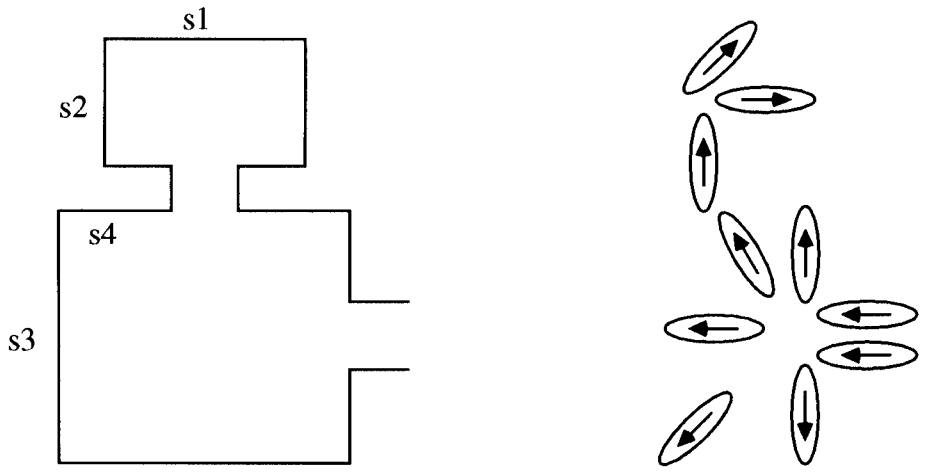


Figure 5.5: Complex Geometry

`Pres-Wave(Place(s1,p2))` *belongs to* `Open-Half-Plane(Inverse(Surface-Normal(s2)))` *(Figure 5.6a); In addition, if* `Relative-Position(p1,p2)` *is equal to the* `Pres-Wave(Place(s1, p2))`, *then* `Forward-Propagation?(s1,s2)` *and that* `s1` *is blocked may be true in this geometry (Figure 5.6b), (2) previous wave front: unless* `s2` *is the same wave front (Figure 5.6c).*

In the case of *same wave front*, `Pres-Waves` of `Places(s2)` are computed by adding `Pres-Wave (Place(s1,p2))` and `Relative-Position(p3,p2)`. If `Forward-Propagation?(s1,s2)` is true and `s1` is blocked, `Pres-Waves` of `Places(s2)` can be any d which belongs to `Open-Half-Plane( Surface-Normal(s1))`. For example, $(--)$ is a possible `PWP` direction around `s2` in Figure 5.6a and b. $(+0)$ and $(+-)$ are also possible directions for `s2` in Figure 5.6b. $((++)$ is eliminated by the surface constraint.)

In the case of *previous wave front*, we cannot compute the possible directions but can give constraints which filter the illegal ones.

Law 5.7 (Source-Constraint) *Suppose* `Backward-Propagation?(s2,s1)` *is true,* `s2` *belongs to the previous wave front, and the end-points of* `s1` *and* `s2` *are* `(p1,p2)` *and* `(p2,p3)`. *Then, (1) the pressure wave cannot propagate from* `p2` *to* `s2`. *Thus for every d which belongs to* `Open-Half-Plane(Relative-Position(p3,p2))`, `PWP-Constraint(s2,`*d*`)` *is true. (2) the pressure wave cannot propagate from* `s1` *to* `s2`. *Thus by the law of Free Propagation, for every d which belongs to* `Open-Half-Plane(Inverse(Pres-Wave(Place(s1,p2))))`, `PWP-Constraint (s2,`*d*`)` *is true.*

By giving surface and source constraints, we can get (-0) and $(--)$ as possible directions for `PWP` near `s2` in Figure 5.6c.

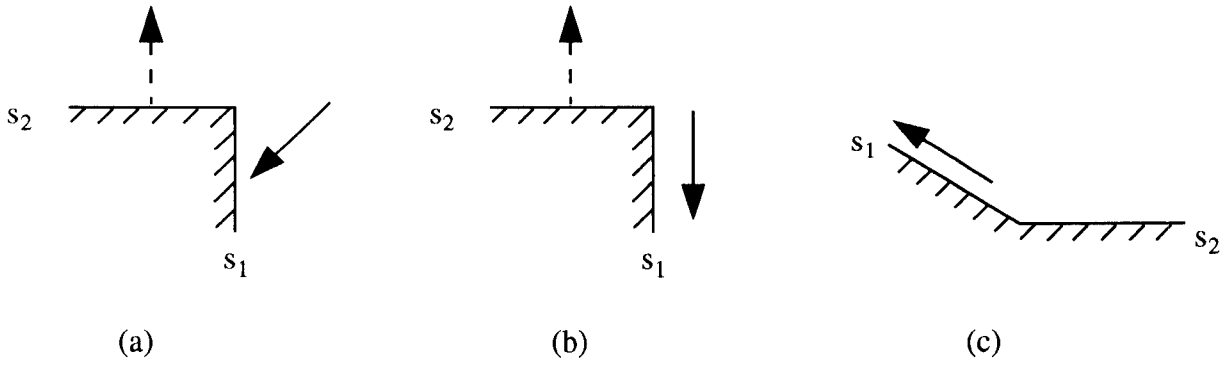


Figure 5.6: Examples of `Backward-Propagation?(s2,s1)`, where `Surface-Normal(s2)` = (0+)

5.2 Envisioning Flow Direction

Given an external disturbance of pressure, the space of interest is incrementally divided by connected places. Then the direction of induced force in each place is determined: it is the same as `Pres-Wave` of the place if `PWP` is due to a compression wave; it is the opposite of `Pres-Wave` of the place if `PWP` is due to an expansion wave. The induced force in each place changes the velocity to that direction.

Once we know the direction of flow in each place, we can predict the next place where the fluid will go by the connectivity of the places. For example, if the induced velocity of a place is (+−), then the fluid in that place will flow into the places which are located on the right and down from the place. When fluid flows into a place, the fluid does not immediately change to the direction of the applied force in the place if those directions are different. This happens since the flow already has momentum when it enters.

For example, in Figure 5.7 when the fluid from `s2`'s place arrives at `s3`'s place around with the velocity (+0), the flow does not immediately change its direction to (+−) due to its momentum even though the induced force (+−) around `s3` is applied to the fluid. There are fewer fluid molecules near `s3` than in other parts of `s3`'s place.

Compared to place generations due to `PWP`, this inertial effect is limited to the inside of a place; a *local* place is generated inside a place when this happens. Even though we can infer this region exists near a surface, it seems to be impossible to explicitly give its boundaries inside of a place.

Since the local place is also generated by pressure change, two kinds are possible:

Definition 5.7 (N-Local-Place) *Suppose P is a place and belongs to* `Places(s)`. *If the pressure adjacent to* `s` *is lower than the other part of inside of P, then* `N-Local-Place(P,s)` *is generated near* `s`.

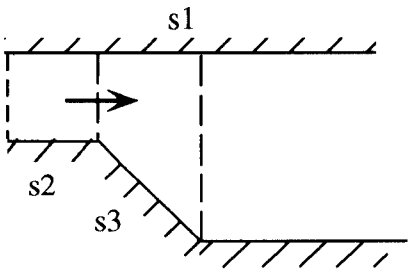


Figure 5.7: Flow and Surface Interaction. The pressure adjacent `s3` is below the pressure of the other part of the fluid. This is caused by the velocity of the fluid arriving at `s3` and the surface geometry of `s3`.

Definition 5.8 (P-Local-Place) *Suppose P is a place and belongs to* `Places(s)`. *If the pressure adjacent to* `s` *is higher than the other part of inside of P, then* `P-Local-Place(P,s)` *is generated near* `s`.

We identified the interaction between flow and geometry as follows:

Law 5.8 (Pulling) *Suppose P is a place belonging to* `Places(s)` *and the flow with the velocity v is entering into P from P's adjacent place. If* `Surface-Normal(s)` *belongs to* `Open-Half-Plane(v)`, *then* `N-Local-Place(P,s)` *is formed. This* `N-Local-Place` *gives the force in direction of* `Inverse(Surface-Normal(s))` *to the flow in P (i.e., it pulls the flow into the surface).*

Law 5.9 (Pushing) *Suppose P is a place belonging to* `Places(s)` *and the flow with the velocity v is entering into P from P's adjacent place. If* `Surface-Normal(s)` *belongs to* `Open-half-Plane(Inverse(v))`, *then* `P-Local-Place(P,s)` *is formed. This* `P-Local-Place` *gives the force in direction of* `Surface-Normal(s)` *to the flow in P (i.e., it pushes the flow into the surface).*

Pushing happens since fluid molecules hit a wall and those collisions increase the pressure near the wall.

Thus fluid direction in each place can be envisioned by starting from the source of disturbance and traveling into the adjacent places with possible changes of its direction due to surface interactions. The flow will stop if its pressure disturbance disappears.

We have only dealt with the velocity change of fluid here. Reasoning about the other important properties of fluid, such as pressure, temperature, and density so on is left as future work.

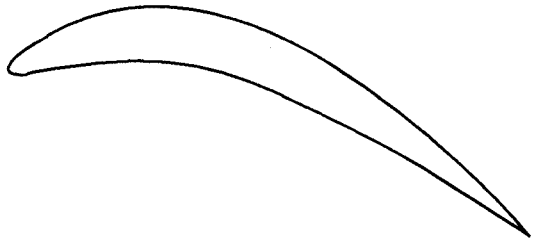


Figure 5.8: An airfoil

5.3 Airfoil Example

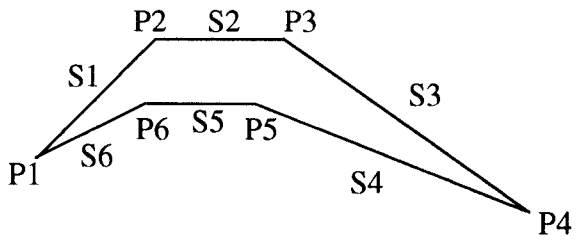
Our technique will now be demonstrated with an airfoil example. A lifting force is the result of an increase of pressure on the lower surface or a reduction below atmospheric pressure on the upper surface or a combination of the two effects (Warner, 1927). In order to induce a lift, an airfoil is designed to force sudden changes of the directions of motion of the particles of air passing below and above the airfoil (Figure 5.8): at least one of the upper and lower surfaces should be continuously concave downward. This shape of the lower surface induces the increase of the pressure on the lower surface since the particles of air are being pushed into the surface. On the other hand, this shape of the upper surface decreases the pressure on the upper surface since the diverted flow from the front creates a vacuum around the top surface.

In our example, we show how the increase and decrease of pressure on both surfaces are explained from the shape. We assume a compression pressure wave is coming from the left of the airfoil. The complete scenario description is shown in Figure 5.9.

Figure 5.10 illustrates envisioning process graphically. F and B represent further and blocked, which are introduced in the forward propagation law, respectively.

At first the pressure wave propagates to `s1` and `s6` (Figure 5.10(1)). Applying forward propagation law shows that both `s1` and `s6` belong to the next wave front of the initial wave front; while the initial pressure wave is blocked by the airfoil around `s6` , it is not around `s1` (i.e., further). Thus the direction of `PWP` around `s1` is $(+0) + (++)$ (i.e., $(++)$). Since a new source is generated around `p1`, the direction of `PWP` around `s6` is $(++)$.

At this point, `PWP` continues to `s2` and `s5` (Figure 5.10(2)). Since `Forward-Propagation?(s2, s1)` is true and `s2` is blocked, `PWP` changes the direction to $(+0)$ around `s2`. In the case of `s5`, `Forward-Propagation(s5,s6)` is true and `s5` is further. Thus `PWP` proceeds in the direction of $(++) + (+0)$. Repeating this process, `s3` and `s4` are now found to have $(+-)$ and $(+*)$ `PWP`, respectively (Figure 5.10(3)). By continuous change property of fluids, the direction around `s4` smoothly changes from $(++)$ to $(+-)$ (Figure 5.10(4)).




```
(airfoil wing)
(flowing-air air)
(enveloped air wing)

(Surface (surf wing s1))   (Surface (surf wing s2))
(Surface (surf wing s3))   (Surface (surf wing s4))
(Surface (surf wing s5))   (Surface (surf wing s6))

;;;----------
(KIN (surf-norm (surf wing s1) (:MINUS :PLUS)) )
(KIN (surf-norm (surf wing s2) (:ZERO :PLUS)) )
(KIN (surf-norm (surf wing s3) (:PLUS :PLUS)) )
(KIN (surf-norm (surf wing s4) (:MINUS :MINUS)) )
(KIN (surf-norm (surf wing s5) (:ZERO :MINUS)) )
(KIN (surf-norm (surf wing s6) (:PLUS :MINUS)) )

(KIN (end-point (surf wing s1) (p1 p2)) )
(KIN (end-point (surf wing s2) (p2 p3)) )
(KIN (end-point (surf wing s3) (p3 p4)) )
(KIN (end-point (surf wing s4) (p4 p5)) )
(KIN (end-point (surf wing s5) (p5 p6)) )
(KIN (end-point (surf wing s6) (p6 p1)) )

(KIN (rel-pos wing (p1 p2) (:MINUS :MINUS)) )
(KIN (rel-pos wing (p2 p3) (:MINUS :ZERO)) )
(KIN (rel-pos wing (p3 p4) (:MINUS :PLUS)) )
(KIN (rel-pos wing (p4 p5) (:PLUS :MINUS)) )
(KIN (rel-pos wing (p5 p6) (:PLUS :ZERO)) )
(KIN (rel-pos wing (p6 p1) (:PLUS :PLUS)) )

;;;---------
;;; Compression wave propagation toward to p1
(pwp comp forward (:PLUS :ZERO) p1)
```

Figure 5.9: Scenario description of an airfoil.

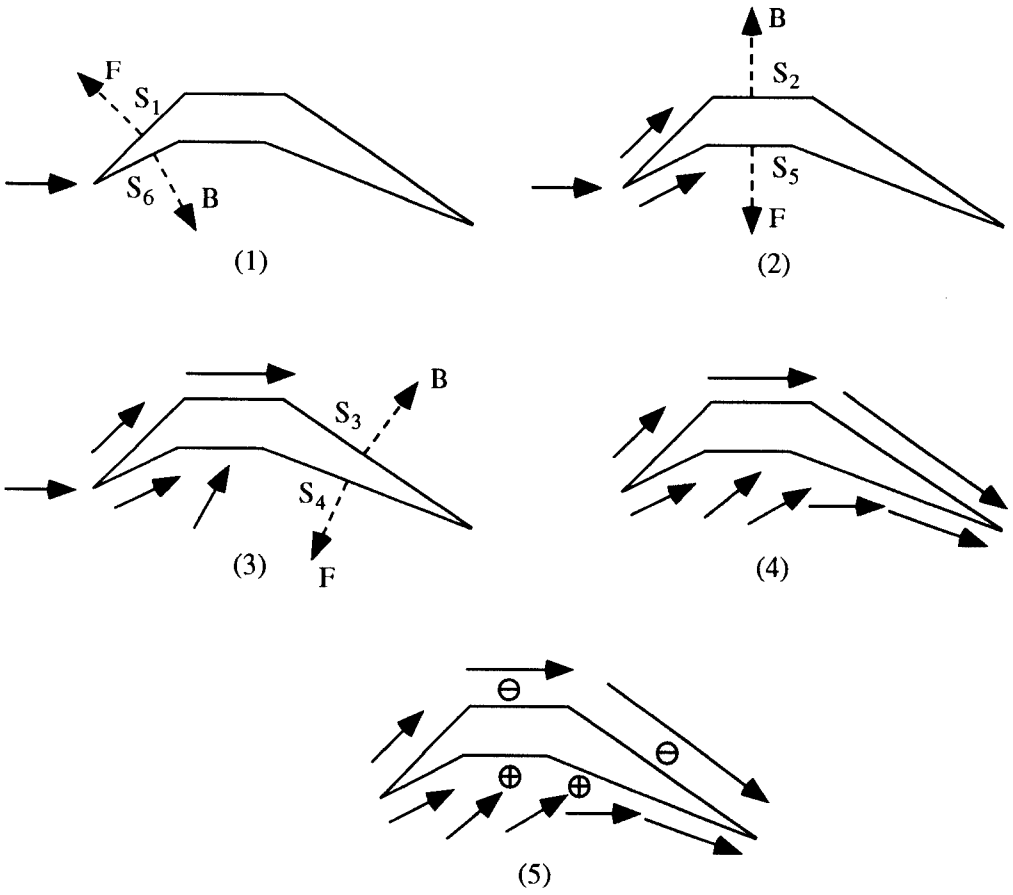


Figure 5.10: Envisioning air flow around an airfoil. F and B represent *further* and *blocked* on the surface, respectively. − and + represent `N-local-place` and `P-local-place` respectively.

Since the given `PWP` is a compression wave, the direction of flow is same as the the direction of `PWP` in each place. Applying the law of pulling and the law of pushing produces an unbalanced force on the airfoil: `N-local-place` on upper surface `s2` and `P-local-place` on lower surface `s5`. In Figure 5.10(5), − (i.e., N-local-place) and + (i.e., P-local-place) mean that the pressure is below and above atmospheric, respectively. Thus, this unbalanced force makes the airfoil lift; the interaction between the curvature of the airfoil and air flow around it brings a lift.

Chapter 6

Discontinuous Changes

Living in the physical world, we are constantly confronted with change. Continuous changes occur gradually, while discontinuous changes occur abruptly. Both kinds of change must be reasoned about for robust explanations of the physical world. For example, in order to predict the behaviors of internal combustion engines, the rapid heat rise from combustion—a discontinuous change—should be explained as well as the continuous changes in the compression cycle.

Qualitative simulation has been successfully and widely used in continuous models. Its main power comes from the method of limit analysis. Given a state, limit analysis identifies which of its defining conditions can change. By detecting possible state changes, the possible behaviors of a system can be predicted.

This chapter describes the model and implementation of discontinuous changes used in QSA. We express discontinuous changes as a new kind of influence. A discrete process is defined by the necessary conditions and effects in the same way that continuous processes are defined. Discontinuous change is caused by discrete processes in a similar way to how continuous change is caused by continuous processes. This chapter describes how limit analysis is extended to apply to discontinuous changes. Using this technique, state changes which show discontinuous changes in quantities can be predicted. The basic inference processes for both cases are conceptually the same: once what happens in a state is determined, the potential state changes are deduced by analyzing the current state and the kinds of possible changes. This similarity helps integrate reasoning about discontinuous and continuous changes.

Section 6.1 briefly reviews the relevant aspects of qualitative dynamics. Section 6.2 and Section 6.3 show how discrete processes are defined and how to predict the state changes they cause. Finally, we describe the implementation and examples in Section 6.4 and Section 6.5.

6.1 Qualitative Dynamics

Given the structural description of a system, envisioning produces a set of states and transitions between them. Each state represents a particular behavior of the system and is qualitatively distinct from others. The transitions describe the changes in behavior.

All objects in the system are described by their quantities, such as **pressure** and **amount-of**. State changes include changes of quantities and are caused by the processes which are acting in the state. There are two types of processes: continuous and discrete. If the value of a quantity gradually changes over time, this is caused by a continuous process. On the other hand, discrete processes cause abrupt changes.

A quantity can be both continuously and discontinuously influenced, depending on circumstances. (Here, we do not consider the discontinuous changes due to the characteristic of the quantity itself, such as angle.) For instance, the pressure inside of a cylinder in an internal combustion engine increases continuously during compression, but it abruptly changes via combustion.

When a quantity is influenced by a process, the change is represented by changes of the ordering relations in its quantity space. The first step in finding changes is to determine which processes are acting in each state. Once the set of active processes are determined, the changes they cause in the state are deduced based on the current values of quantities and the effects of the processes. For instance, for quantities `a` and `b`, if `a` < `b` and `a` is increasing while `b` is not changing, then next state may be `a` = `b`. This is called *limit analysis* and is the core of qualitative simulation. The theory of limit analysis has been well developed. While this powerful technique has been applied in continuous domains, it has not been applied to discrete processes.

It might seem at first that the STRIPS representation for actions (Forbus, 1989) might be adapted to reason about discrete processes. Unfortunately, the STRIPS model cannot be used to implement limit analysis.

Suppose we want to model the combustion in an internal combustion engine. Once combustion occurs, the quantities such as the pressure, the amount of fuel mist, and the amount of exhaust gas inside of the cylinder change abruptly. The pressure after combustion is significantly greater than the pressure before combustion. The amount of fuel mist after combustion is much less than before, and the amount of exhaust gas is greater. The common pattern is: whatever the quantity was before combustion, it will be increased or decreased by some amount once the discrete process occurs. It is difficult to represent these changes simply by add and delete lists.

In this work, discontinuous changes are expressed as influences, like continuous changes are. Discontinuous changes are predicted from the current values of quantities and the effects of the active discrete processes in each state. In the following sections, we show how such discrete processes are defined, and how add and delete lists are generalized to include the influences of discrete processes.

6.2 Discrete Process

A discrete process differs in several ways from a continuous process. First, it can result in a discontinuous ordering change in quantity spaces. However, this does not always occur. If there is no neighbor in the direction of change within the quantity space, then the change cannot affect anything. A discrete process might also lead to a continuous ordering change.

Second, the value and derivative of the influenced quantity are not defined during the time the discrete process is active. Thus, an active discrete process should be represented as a transition between the state before the change and the state after the change. Third, the state before a discontinuous change always lasts an instant, since the change is rapidly introduced by a discrete process as soon as the state is reached. There is no constraint on the duration of the state after the transition.

A discrete process is defined in four parts, similar to processes in QP theory: *individuals*, *preconditions*, *quantity conditions*, and *effects*. Whenever all individuals in *individuals* exist and all conditions in *preconditions* and *quantity conditions* are satisfied, the related quantities discontinuously change as described in *effects*. Quantity conditions include ordinal relations between quantities which can be predicted by physics. Preconditions describe the conditions outside physics, for instance, that two containers are connected by a path. Effects of discrete process correspond to influences of continuous process. To indicate an abrupt change of a quantity `qty`, we write

(*Increase* <qty>)
(*Decrease* <qty>) or
(*Increase-by* <qty> <amount>)
(*Decrease-by* <qty> <amount>)

This corresponds to $I\pm$(`qty1`, `qty2`), which means `qty2` directly influence the quantity `qty1` in a continuous process definition. Unlike $I\pm$, the derivative of `qty` is not defined during the process is active. The *Increase-by* and *Decrease-by* are provided to specify how much the quantity changes. The **amount** is a positive number, described qualitatively. The changes by the combustion described in the previous section are described in our representation as

```
Effects (Increase-by (pres ?c-g) combustion-pres)
        (Decrease (amount-of (fuel-mist ?c-g)) )
        (Increase (amount-of (exhaust-gas ?c-g)) )
```

`?c-g` represents the contained-gas in the cylinder. The complete process definition for the combustion is shown in Figure 6.1. Our representation also allows specifying the status of a quantity after an discontinuous change, like in the delete-lists in the STRIPS representation. To indicate a quantity `qty` is changed to a **value**, we write

(= <qty> <value>)

For example, if we assume the complete combustion of gas inside a cylinder, then the amount of fuel-mist will be decreased to `ZERO` after combustion, and this is represented by

```
Effects (= (fuel-mist ?c-g) ZERO)
```

```
DiscreteProcess CLTDC
;;; Combustion occurs at TDC
  Individuals
      (?pst   :type piston)
      (?cyl   :type container
              :conditions (part-of ?pst ?cyl))
      (?crankshaft :type crankshaft
                   :conditions (connected ?pst ?crankshaft)
                   (engine-cylinder ?pst ?cyl ?crankshaft))
      (?c-g   :type contained-gas
              :form (c-s ?sub GAS ?cyl))
  PreConditions (spark ?pst ?cyl)
  QuantityConditions
      (= (position ?crankshaft) TDC)
      (> (amount-of (fuel-mist ?c-g)) ZERO)
  Effects
      (Increase-by (pres ?c-g) combustion-pres)
      (Decrease (amount-of (fuel-mist ?c-g)) )
      (Increase (amount-of (exhaust-gas ?c-g)) )
```

Figure 6.1: An example of `DiscreteProcess`

6.3 Finding State Changes Caused by Discrete Processes

Here is how to infer the effects of discrete processes:

1. *Finding the active process in each state:* Instances of discrete processes are created for each combination of entities which satisfy the individuals specifications of the discrete processes. The set of active discrete processes consists of those instances whose precoditions and quantity conditions are satisfied.

2. *Determining the changes in each state:* Once the processes which apply to a state are ascertained, the net changes caused by these processes are determined by summing the effects of each process. The increases or decreases caused by the discrete processes are represented by changes in the quantity space orderings for the affected parameters. Table 6.1 shows these changes. When the amounts of changes are provided by increase-by's and decrease-by's, they can be used to disambiguate `N1` - `N4` in the table. Ambiguous next states in `N1` - `N4` are filtered out if the inequalities of the net changes between two relevant quantities are also provided.

6.4 Implementation

The implementation of discontinuous changes inference is based on the representations of `QPE`. Currently `QSA` does not reason about multiple discrete processes acting in a given instant.

`QPE`'s standard envisionments contain the set of states and transitions due to continuous processes. The set of states are generated by computing all consistent combinations of preconditions and quantity conditions of processes. Let Q_s and P_s represent the set of assumptions about the inequalities in quantities and the set of background assumptions for a scenario, respectively. Then each state consists of a set of assumptions drawn from S_A, where $S_A = Q_s \cup P_s$. Q_s is changed by the active processes in S_A and P_s can be changed by the actions in the state.

A *dp hypothesis* is defined for an discrete process instance. A process instance is created by instantiating a process in a domain model. For a dp hypothesis, finding in which states the dp hypothesis becomes active is simple since the temporal scoping of facts is implicit in their ATMS label. Checking whether its preconditions and quantity conditions are hold in a state is performed simply by seeing if they are implied by the assumptions defining the state. If the condition hold in a state S_1, then the possible transition from S_1 is found by checking the changes caused by active processes on S_1.

For each discrete process instance PI_i,

1. Find all states S_{PI_i} which are consistent with QC(PI_i) and PC(PI_i), where QC(PI_i) and PC(PI_i) represent the assumptions described in the quantity conditions and preconditions of PI_i, respectively.

2. For each state s in S_{PI_i},

For A > B:

Change[B]

Change[A]	Decrease	No Change	Increase
Decrease	N1	all	all
No Change	>	>	all
Increase	>	>	N2

```
N1:If Change[A] > Change[B] then all; > otherwise
N2:If Change[A] < Change[B] then all; > otherwise
```

For A = B:

Change[B]

Change[A]	Decrease	No Change	Increase
Decrease	N3	<	<
No Change	>	=	<
Increase	>	>	N4

```
N3:Change[A] > Change[B] then <
   Change[A] < Change[B] then >
   Change[A] = Change[B] then =

N4:Change[A] > Change[B] then >
   Change[A] < Change[B] then <
   Change[A] = Change[B] then =
```

Table 6.1: Change by discrete processes in quantity space. Change[A] and Change[B] represent the change of quantity A and B by discrete processes, respectively.

(a) Find the set of assumptions Q_s which describes the possible results of PI_i on s.

i. If the effect on a quantity is given by = <qty> <value>, the description is the next status for the quantity.

ii. Otherwise, the next status for a quantity is determined by Table 6.1. If the next status is ambiguous, it is not included in Q_s.

(b) Find all states $\{s_{next}\}$ which are consistent with $(s - \mathrm{QC}(PI_i)) + Q_s$. Make the transition from s to each state in $\{s_{next}\}$ with label PI_i.

In step 2 (a), if the next status for a quantity is ambiguous, it is not included in Q_s. Thus in step 2 (b) $\{s_{next}\}$ includes each state with every possible value of the quantity.

6.5 Examples

Our implementation has been tested on various examples, including molecular genetic models, shock waves in compressible fluid flow, and combustion.

Figure 6.2 shows some of possible transitions by `CLTDC` in Figure 6.1. The states in the left column represent some of the states where `CLTDC` will happen. They have common assumptions, which are the quantity conditions and precondition of the process. Each of them differs from others in the status of either `(pres c-g)` or `(amount-of (exhaust-gas c-g))`.

Since `(position crankshaft)` is not affected by the process, it will have same value after the process occurs. However, the quantities `(pres c-g)`, `(amount-of (exhaust-gas c-g))`, and `(amount-of (fuel-mist c-g))` will change their values since they are influenced by the process. Their new values depend on the types of effect and the current values when the process occurs. For example, if `(> (pres c-g) some-pres)` and `some-pres + combustion-pres = large-pres`, then `(> (pres c-g) large-pres)` will be true after the process occurs.

Our molecular genetic model replicates the examples in (Weld, 1985). In our representation, the model is modified so that every change by processes is captured by effects on quantities. Consider an exonuclease E and a dna DNA. If empty cleft of E is near the right end of free DNA, the right end of DNA will be bound to the cleft: this is called *bind* process. Once they are bound, the right end of DNA chain will be digested by E and DNA is freed with the length one shorter than before: this is called *snarf* process. Figure 6.3 shows how these processes are represented in our representation. By determining the changes of `stance-between` and `length`, our envisionment can be obtained. Figure 6.4 shows envisionment when the quantity space of `length` consists of the orderings with 0. The abrupt change caused by BIND or SNARF is represented by the transition with the label. If the quantity space includes more elements, the envisionment will be described by more (BIND,SNARF) transitions before arriving to `s2` in the figure.

before	after
s1:	s3:
(spark pst cyl)	(spark pst cyl)
(= (position crankshaft) TDC)	(= (position ?crankshaft) TDC)
(> (amount-of (fuel-mist c-g)) ZERO)	(> (amount-of (fuel-mist c-g)) ZERO)
(> (pres c-g) some-pres)	(> (pres c-g) large-pres)
(= (amount-of (exhaust-gas c-g)) ZERO)	(> (amount-of (exhaust-gas c-g)) ZERO)
s2:	s4:
(spark pst cyl)	(spark pst cyl)
(= (position crankshaft) TDC)	(= (position crankshaft) TDC)
(> (amount-of (fuel-mist c-g)) ZERO)	(= (amount-of (fuel-mist c-g)) ZERO)
(> (pres c-g) some-pres)	(> (pres c-g) large-pres)
(> (amount-of (exhaust-gas c-g)) ZERO)	(> (amount-of (exhaust-gas c-g)) ZERO)
s5:	s7:
(spark pst cyl)	(spark pst cyl)
(= (position crankshaft) TDC)	(= (position crankshaft) TDC)
(> (amount-of (fuel-mist c-g)) ZERO)	(> (amount-of (fuel-mist c-g)) ZERO)
(> (pres c-g) ZERO)	(> (pres c-g) combustion-pres)
(= (amount-of (exhaust-gas c-g)) ZERO)	(> (amount-of (exhaust-gas c-g)) ZERO)
s6:	s8:
(spark pst cyl)	(spark pst cyl)
(= (position crankshaft) TDC)	(= (position crankshaft) TDC)
(> (amount-of (fuel-mist c-g)) ZERO)	(= (amount-of (fuel-mist c-g)) ZERO)
(> (pres c-g) ZERO)	(> (pres c-g) combustion-pres)
(> (amount-of (exhaust-gas c-g)) ZERO)	(> (amount-of (exhaust-gas c-g)) ZERO)

Figure 6.2: Transitions by CLTDC. Both s1 and s2 can change to s3 and s4. Both s5 and s6 can change to s7 and s8. The description of each state is partial; other quantities not directly related to CLTDC are not shown here. (c-g represents the contained-gas inside a cylinder.)

```
DiscreteProcess SNARF
  Individuals (?DNA :type dna)
              (?E   :type exonuclease)
  PreConditions (At-Right ?DNA ?E)
  QuantityConditions (= (distance-between ?DNA ?E) ZERO)
  Effects (Increase (distance-between ?DNA ?E))
          (Decrease (length ?DNA) by (unit-length ?DNA))


DiscreteProcess BIND
  Individuals (?DNA :type dna)
              (?E   :type exonuclease)
  PreConditions (At-Right ?DNA ?E)
  QuantityConditions (> (distance-between ?DNA ?E) ZERO)
  Effects (= (distance-between ?DNA ?E) ZERO)
```

Figure 6.3: Discrete processes: SNARF and BIND

```
s0:(At-Right DNA E)              BIND   s1:(At-Right DNA E)
(> (distance-between DNA E) 0) ------> (= (distance-between DNA E) 0)
(> (length DNA) 0)             <------ (> (length DNA) 0)
          |                     SNARF
          | SNARF
          |
          V
s2: ??(At-Right DNA E)           ;;; This cannot be defined
(?? (distance-between DNA E) 0) ;;; Distance cannot be defined
(= (length DNA) 0) ;;; DNA does not exist any more
```

Figure 6.4: Envisionment of BIND and SANRF model.

Chapter 7

Global Filtering

In the preceding chapters, we described the domain theories and analysis methods necessary to explain possible behaviors of the physical systems that combine fluids and mechanisms. All of these techniques are integrated into a qualitative simulator (i.e., QPE) to generate the behavior of a given physical system.

Conceptually, we view determining possible behaviors as a process of filtering illegal states and illegal state transitions (Struss, 1988). Transition filtering should be done at two different levels: *local* and *global* filtering. Local filtering focuses on whether a transition between two states is legal or not, independent of other states. In qualitative simulation, this is driven by continuity.On the other hand, global filtering concerns finding correct sequences of behaviors.

Simulation processes, whether numerical or qualitative, find behaviors via local computations. In spite of the lack of global filtering, numerical simulation can find correct behaviors since it uses precise metric information. However, in qualitative simulation, the localized nature of simulation combined with qualitative description is not sufficient to infer completely accurate global behaviors. Thus, understanding how to automate global filtering and how to integrate it with local filtering is crucial to designing better qualitative simulators.

This chapter discusses how global constraints are represented and manipulated during the process of qualitative simulation. The basic idea is to automatically generate additional information for enforcing global constraints. This is done by automatically introducing variables and controlling their values to guide correct transitions between the behaviors. We express this idea within the framework of QP theory.

As our motivating example—combustion in an internal combustion engine—shows, our approach can provide the means to link geometric constraints with possible motions. In an internal combustion engine, for instance, the interaction between the behavior during power stroke, and the geometry of a a piston-cylinder combination were captured by this technique. We illustrate this using an implemented model of a simple internal combustion engine. We describe how subtly different expansion periods with and without combustion in the cylinder are captured.

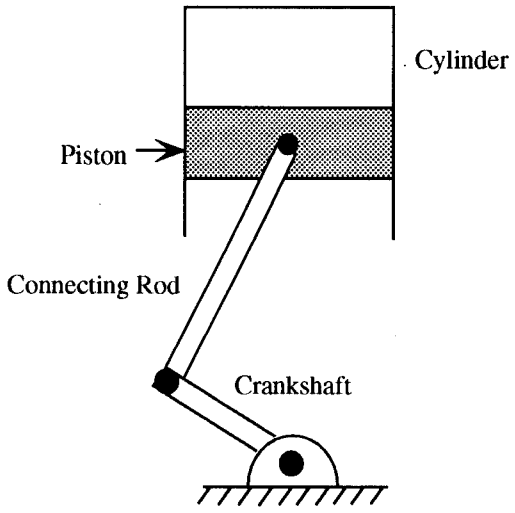


Figure 7.1: Piston cylinder geometry

7.1 The Problem

Envisioning is the process of deriving all possible behaviors of a system given the structural description of the system and a domain theory. The behavior is represented by a set of qualitative states and the transitions between them. The graph of states and transitions is called the envisionment for the system.

Like a finite state automaton, each state in an envisionment has an incomplete summary of information about its past path (Hopcroft & Ullman, 1979). The next transition from a state is determined only by the constraints between the information in the state and the information in another state. This reflects the local nature of simulation. The past behavior need not be traced since the current state has enough information to determine the next state(s).

We illustrate the need for global filtering during the qualitative simulation process by examining the derivation of possible behaviors for a simple internal combustion engine.

Figure 7.1 shows an abstract geometry of a cylinder and its assembly. A piston is connected to a crankshaft through a connecting rod. The slider crank mechanism transmits the vertical motion of the piston to the rotation of the crankshaft. When the crankshaft is at *Top dead center* (TDC) and *Bottom dead center* (BDC), the piston reaches its highest and lowest position, respectively. We can infer that the cylinder has the minimum volume at TDC and maximum at BDC.

A four-stroke engine, which is the most common engine, goes through four phases (i.e., intake, compression, power, and exhaust) to complete one cycle. The rapid heat rise by combustion is translated into pressure which acts on the piston to force it down. Then, by the geometry, positive torque is transmitted to the crankshaft by the connecting rod. The flywheel,

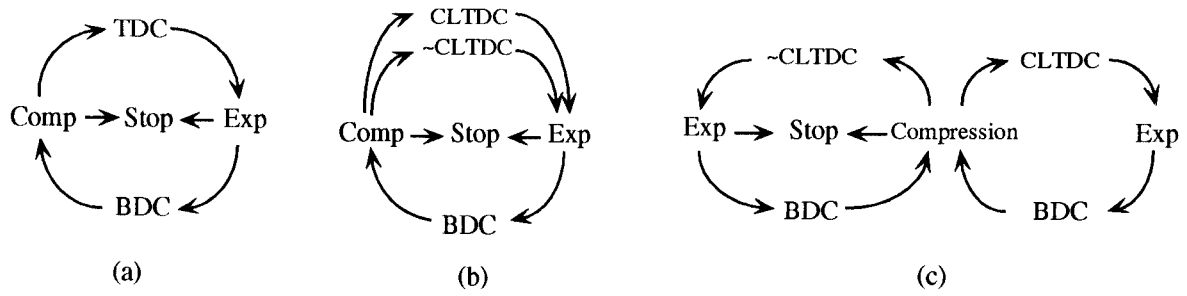


Figure 7.2: The behavior of a piston-cylinder system. These show abstract behaviors rather than actual states in envisionments. While `CLTDC` represents "combustion lasts at `TDC`", i.e., `(pressure ?c-g)` >= `combustion-pres`, ~`CLTDC` represents the state `(pressure ?c-g)` < `combustion-pres`. (`?c-g` represents the contained-gas inside of the cylinder.)

which is connected to the crankshaft, also gets positive torque during this power stroke. The heavy flywheel gives back to the crankshaft, during the three other strokes, the surplus energy it took during the power stroke. The interaction between the part motions and the pressure changes plays a key role in understanding this system.

In building our model we have focused on this interaction, ignoring features irrelevant to our motivating example, such as intake and exhaust flows. Our model is built based on the following assumptions: (1) The working fluid is ideal gas, (2) There is a fixed mass of working fluid through cycle, and (3) Combustion is modeled as heat addition from external source. These assumptions are same as those of the air-standard Otto cycle. The Otto cycle is an ideal cycle that closely approximates a spark-ignition internal-combustion engine (Ferguson, 1976). We also consider friction in the model.

With these assumptions, a piston-cylinder system repeats the cycle of compression and expansion as the piston moves upward and downward (Figure 7.2a). Each part eventually will stop moving due to friction. At `TDC`, the cylinder has the minimum volume and the maximum pressure. As the piston is forced down by the pressure, the pressure is decreasing since the volume is increasing. After passing `BDC`, the gas in the cylinder is compressed as the piston is moving up due to the force of the crankshaft and the flywheel. If the pressure or the upward force is not big enough, the engine might stop due to friction.

Suppose combustion happens at `TDC`. The expansion period after combustion (i.e., power stroke) is slightly different from the expansion without combustion: each part will not stop during the former while it might stop during the latter. Once the pressure is increased by combustion, say `(pressure ?c-g)` >= `combustion-pres`, the pressure remains high enough to accelerate the crankshaft even though the pressure is decreasing during following expansion period. The geometry of the engine is designed so that once the pressure reaches `combustion-pres` at `TDC`, the pressure until `BDC` is greater than `nonstop-pres`. The `nonstop-pres` is a pressure sufficient to overcome the friction of each part.

However, it is impossible to distinguish these different behaviors with current qualitative simulators (Figure 7.2b) since this kind of distinction requires filtering paths, i.e., sequences of states. In other words, correct analysis of the power cycle should prevent the transition from the combustion at TDC to the path ended in stop state. With current qualitative simulators, the different statuses at TDC are captured while the different expansions are not (Figure 7.2b).

Figure 7.2c shows the desirable envisionment which accurately captures the behavior of a piston-cylinder system. This has more states than Figure 7.2b and distinguishes the behaviors with and without combustion. Since we do not give any constraint during the compression period after combustion, the behavior during the period is the same as the behavior of compression period without combustion.

7.2 Modeling Global Constraints

The problem just shown can be solved by adding more information to each state, representing the global information needed to filter transitions from a particular state.

7.2.1 Event

We introduce a notion of *event* to model the constraints in the behavior after some point. It allows the effect of the state to be explicitly reflected in following paths. To filter out the transitions to illegal paths, we add more information representing global constraints by introducing an extra variable. Intuitively, this variable is used as the tag which informs afterwards whether or not the event happened or not at some point. Since each state cannot always include sufficient information to predict the next state, as shown in combustion example, the explicit information about the past is automatically included by specifying an event in that case.

Events are defined by three parts: *individuals*, *quantity conditions*, and *relations*. Whenever all individuals in *individuals* exist and all conditions in *quantity conditions* are satisfied, the event is active. *Relations* contain the constraints about behavior after the event happens. Figure 7.3 shows how this is represented in the example of combustion at TDC.

Quantity conditions consists of a set of statements of the form

(<*conditions*> **when** <*configurations or states of objects*>).

It says the event occurs if <*conditions*> are true in some <*configurations states of objects*> (e.g., the pressure in a cylinder is increasing, decreasing, or not changing). For example, in Figure 7.3, the event CLTDC become active if the **pressure** of the cylinder is not less than the **combustion-pres** when the crankshaft reaches TDC. If the **pressure** is less than the **combustion-pres** at TDC, the event does not become active.

Relations explicitly represent the constrained behavior after an event. Each statement in the field is expressed by

(<*name*> <*constrained-behavior*> **when** <*configurations or states of objects*>) or
(<*name*> <*constrained-behavior*> **when** <*configurations or states of objects*>
:neg <*constrained-behavior*>)

```
Defevent CLTDC ;;; Combustion lasts at TDC
  Individuals
    (?pst  :type piston)
    (?cyl  :type cylinder
           :conditions (part-of ?cyl ?pst))
    (?crs  :type crankshaft
           :conditions (connected ?pst ?crs))
    (?c-g  :type contained-gas
           :form (c-s ?sub GAS ?cyl))
  QuantityConditions
    ((pressure ?c-g) >= (combustion-pres ?pst ?cyl ?crs)
       when (position ?crs) = TDC)
  Relations
    (CLTDC-EXP (pressure ?c-g) > (nonstop-pres ?pst ?cyl ?crs)
      when (not ((velocity ?pst) > 0)))
```

Figure 7.3: An example of `defevent`.

It says that the behavior is constrained to <*constrained-behavior*> when <*configurations or states of objects*> is true after the event occurs. The `:neg` option is provided to the constrained behavior if the event did not happen (i.e., "no event") in the point.

In `CLTDC` (Figure 7.3), the relation `CLTDC-EXP` represents the constrained behavior during the expansion period[1] after combustion—the `pressure` of the cylinder is greater than the `nonstop-pres`.

Suppose the following expansion cycle when the event does not occur is also constrained, say, that the `pressure` during the period is not greater-than `nonstop-pres`. Then `CLTDC-EXP` is described as follows:

```
(CLTDC-EXP
   (pressure ?c-g) > nonstop-pres
       when (not ((velocity ?pst) > 0))
   :neg (pressure ?c-g) <= nonstop-pres)
```

7.2.2 Filtering Behaviors Using Events

We assume the underlying simulator finds every correct set of states and local state transitions. Events are used to prevent the transitions to states which lead to impossible sequences of behavior (Figure 7.4). The remaining part of the envisionment should not be affected by this filtering. For example, in Figure 7.4, the path from $s1$ is a part of the behavior, even though the transition from s0 to s1 is illegal.

[1] When the velocity of the piston is non-negative, the gas in the cylinder is expanded.

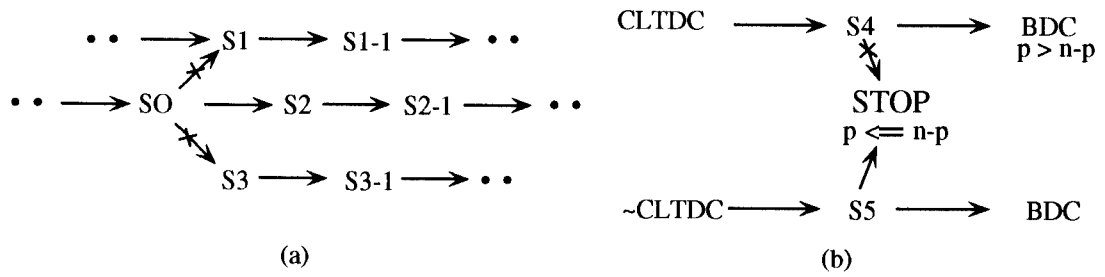


Figure 7.4: Filtering behaviors. (a) There are three possible paths from s0. Suppose only the path which starts with $s2$ should be selected from $s0$ due to nonlocal constraints. The transitions to $s1$ and $s3$ must be prevented. (b) The transition from $s4$ to `STOP` should be prevented. (`n-p` stands for `nonstop-pres`.)

To filter out illegal paths, we need some means to express behaviors with an extra variable and to constrain the transitions between them by controlling its values. Views in QP theory can nicely capture the behaviors. Once the behaviors can be identified, filtering some behaviors after a particular state during limit analysis must satisfy the following restrictions:

- *If the behavior is constrained:* the transitions from the state should be made only to the path(s) which describe the behavior. The transitions to others should be prevented (Figure 7.5a).
- *If the behavior is not constrained:* the transitions to every possible path should be made (Figure 7.5b).

7.3 Implementation

In this section, we show how global constraints expressed by `defevent` are incorporated in the framework of QP theory, using the combustion model as an example. A `defevent` is translated into several views with an extra variable, and filtering is done based on the variable.

7.3.1 Generating Views

First, we need to distinguish whether or not an event happens. This is done by generating two views for each case, along with an extra variable. The name of the event and the name prefixed with $\sim$ are used for the names of the two views. For example, the views, `CLTDC` and `~CLTDC`, are introduced for the event `CLTDC`. The extra variable is set to positive when the event happens and set to non-positive otherwise. Figure 7.6 shows the views `CLTDC` and `~CLTDC`. The individuals of both views are same as the individuals of the event `CLTDC`. An extra variable `CLTDC-tag` is introduced with different values.

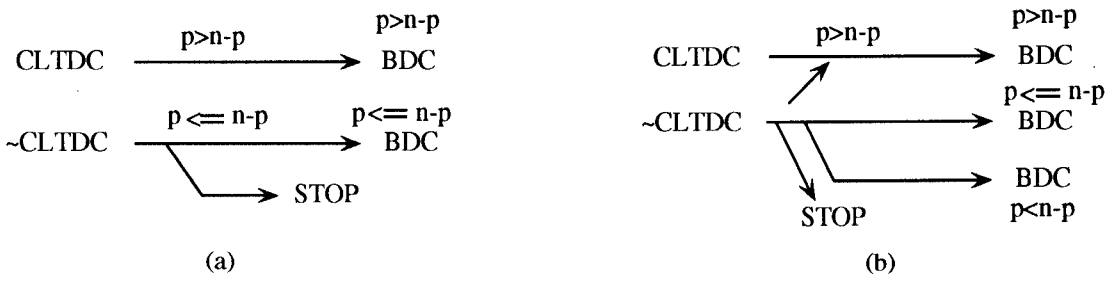


Figure 7.5: The behavior from TDC to BDC in a piston-cylinder system. In (a), expansion after both CLTDC and ~CLTDC are constrained and different from each other. In (b), the expansion after ~CLTDC is not constrained.

Two different views are also generated for each relation: one for the subsequent behavior after the event and the other for the behavior when the event does not occur(Figure 7.7). The quantity conditions of both include the statements specified after **when**, along with the statement about the tag. While the former includes the assumption that the tag is positive, the latter includes the assumption that the tag is non-positive. The relations of the views constrain the behavior as specified in the relation of the event. The name of the relation and the name prefixed with ~ are used for the names of the views. In Figure 7.7 since the behavior during expansion after ~CLTDC is not constrained, the relation field in ~CLTDC-EXP is left empty. If it is also constrained, the constrained behavior will be described in the field.

7.3.2 Transitions

Once these views are generated, the correct behaviors for each case are selected by controlling the sign of the extra variable.

The first requirement—when the behavior is constrained—is easily solved since the views generated by an event are manipulated to set the tag variable with this in mind. Continuity detection is used to purse possible transitions. (Figure 7.5a).

In the case of the second requirement—when the behavior is not constrained, deriving correct behaviors includes the transition between two states which have different tag values. Note that this case happens only to the behavior when the event does not occur. If the behavior after no event is not constrained, it implies transitions to every possible behavior, including the transition to the constrained path after the event (Figure 7.5b).

This transition requires to change the tag from **off** to **on**, i.e., from **tag** <= 0 to **tag** > 0. Thus, we need some means to connect the states which have different values of the tag. There are two approaches to handling this:

Model as a continuous change: In QP theory, changes are caused only by processes. Thus, legal transition can be done by generating a dummy process which changes the **tag** from **off**

```
defView CLTDC ;;; Combustion lasts at TDC
  Individuals
    (?pst  :type piston)
    (?cyl  :type cylinder
           :conditions (part-of ?cyl ?pst))
    (?crs  :type crankshaft
           :conditions (connected ?pst ?crs))
    (?c-g  :type contained-gas
           :form (c-s ?sub GAS ?cyl))
  QuantityConditions
    (position ?crs) = TDC
    (pres ?c-g) >= (combustion-pres ?pst ?cyl ?crs)
  Relations
    (CLTDC-tag ?pst ?cyl ?c-g) > 0

defView ~CLTDC ;;; Combustion does not last at TDC
  Individuals
    (?pst  :type piston)
    (?cyl  :type cylinder
           :conditions (part-of ?cyl ?pst))
    (?crs  :type crankshaft
           :conditions (connected ?pst ?crs))
    (?c-g  :type contained-gas
           :form (c-s ?sub GAS ?cyl))
  QuantityConditions
    (position ?crs) = TDC
    (pres ?c-g) < (combustion-pres ?pst ?cyl ?crs)
  Relations
    (CLTDC-tag ?pst ?cyl ?c-g) <= 0
```

Figure 7.6: QP theory definition of CLTDC and ~CLTDC views. These two views distinguish whether or not combustion happens at TDC.

```
defView CLTDC-EXP
  Individuals
    (?pst  :type piston)
    (?cyl  :type cylinder
           :conditions (part-of ?cyl ?pst))
    (?crs  :type crankshaft
           :conditions (connected ?pst ?crs))
    (?c-g  :type contained-gas
           :form (c-s ?sub GAS ?cyl))
  QuantityConditions
    (not ((vel ?pst) > 0))
    (CLTDC-tag ?pst ?cyl ?c-g) > 0
  Relations
    (pressure ?c-g) > (nonstop-pres ?pst ?cyl ?crs)

defView ~CLTDC-EXP
  Individuals
    (?pst  :type piston)
    (?cyl  :type cylinder
           :conditions (part-of ?cyl ?pst))
    (?crs  :type crankshaft
           :conditions (connected ?pst ?crs))
    (?c-g  :type contained-gas
           :form (c-s ?sub GAS ?cyl))
  QuantityConditions
    (not ((vel ?pst) > 0))
    (CLTDC-tag ?pst ?cyl ?c-g) <= 0
  Relations
    ()
```

Figure 7.7: QP theory definition of CLTDC-EXP and ~CLTDC-EXP views. These two views distinct behaviors during expansion period with and without combustion at TDC.

to on. The process consists of the same fields as the fields of the view for the unconstrained behavior. Thus, it becomes active during the path for the behavior. Its influence field describes the change of the tag variable. Then limit analysis finds the correct transition by checking the continuity of every state variable, including the tag.

Model as a discontinuous change: We can directly make the transition, as is done with discrete processes. The transition is made from off to on as far as the transition implies the legal changes of other variables.

In qualitative reasoning, avoiding unnecessary distinctions is important, since this reduces complexity and provides a more abstract analysis. Thus it is important to make a distinction only when the effects of an event result in qualitatively different behaviors. Unless the effects clearly make differences, such as the compression period after CLTDC, we do not consider whether or not the event happens. In other words, the extra tag variable is not used. CLTDC-tag, for instance, is not considered during the compression period. Since the transition between the state with tag and the state without tag does not violate the continuity due to the tag variable, no extra work is done to connect the influenced behavior to the subsequent uninfluenced behavior. For instance, the different paths during expansion are connected to one path for compression in CLTDC (Figure 7.2c) without extra manipulation. Continuity checking between the state variables is sufficient.

Though multiple events are defined, they can be handled without any difficulty since they are manipulated by independent additional variables. Thus, there is no interference between events.

7.4 Example

In this section we illustrate envisionments of a simple internal combustion engine produced by QPE. Figure 7.8 shows the scenario description.

Our model of a simple internal combustion engine is that of Section 7.1. It describes the phenomena related to understanding the system—motion, angular-motion, acceleration, friction, compression, expansion, and force applied to a piston from a flywheel. How a flywheel derives the vertical motion of a piston is modeled as follows:

```
(Q= (force-between ?flywheel ?pst)
    (- (const-force ?flywheel ?pst) (var-force ?flywheel ?pst)))
(Qprop (var-force ?flywheel ?pst) (position ?pst))
```

With this force relation, the basic total number of states is 13—TDC, BDC, 5 states for compression, 5 states for expansion, and a stop state. Figure 7.9 shows a partial description of states for expansion period, TDC, and BDC, focusing on the motion of a piston and force applied to the piston. The compression period is described in the opposite way to the expansion period. In an envisionment, these basic states will be split by the pressure inside a cylinder and the effects of friction.

Envisionments for the scenario in Figure 7.8 are shown in Figure 7.10 and Figure 7.11. Figure 7.10 shows the envisionment when our domain model does not include the event CLTDC.

```
(container cylinder1)
(piston piston1)       ;;; piston is object
(flywheel flywheel1)   ;;; flywheel is rotor
(part-of cylinder1 piston1)
(mobile piston1)
(flywheel-pst-pair flywheel1 piston1)

;;; Assume flywheel1 does not rotate in CCW direction
(not (greater-than (A (angular-velocity flywheel1)) ZERO))

;;;------
(state gas)
(substance fuel-mix)
(Can-Contain-Substance cylinder1 fuel-mix gas)

;;; Assume cylinder is filled with fuel-mix
(greater-than (a (amount-of-in fuel-mix gas cylinder1)) zero)
```

Figure 7.8: A scenario description of an internal combustion engine.

Quantity	TDC	s1	s2	s3	s4	s5	BDC
Ds[var-force(fw,pst)]	0	-1	-1	-1	-1	-1	0
Ds[force(fw,pst)]	0	1	1	1	1	1	0
Ds[velocity(pst)]	-1	-1	0	1	1	1	1
Ds[position(pst)]	0	-1	-1	-1	-1	-1	0
A[const-force(fw,pst)] A[var-force(fw,pst)]	$<$	$<$	$<$	$<$	$=$	$>$	$>$
A[force-between(fw,pst)]	< 0	< 0	< 0	< 0	$= 0$	> 0	> 0
A[velocity(pst)]	$= 0$	< 0	< 0	< 0	< 0	< 0	$= 0$

Figure 7.9: Partial state descriptions for expansion period (from TDC to BDC). An piston-cylinder system goes through TDC -> s1 -> s2 -> s3 -> s4 -> s5 -> BDC. (fw and pst represent flywheel1 and piston1, respectively.)

While TDC is split into two states—a state with pressure(c-g) > combustion-pres and a state with pressure(c-g) <= combustion-pres, the subsequent different expansion periods are not distinguished. On the other hand, Figure 7.11 shows an envisionment which distinguishes these expansion periods by including the event CLTDC in our domain model. Notice that the system does not stop during expansion period after CLTDC.

Figure 7.12 shows expansion periods with the event CLTDC in details. In addition to the transition from ~CLTDC to stop, there is a transition from ~CLTDC to CLTDC-EXP since ~CLTDC-EXP is not constrained. In other words, when pressure(c-g) <= combustion-pres at TDC, the system might either stop or keep moving during the ensuing expansion period. But there is no transition to stop after CLTDC, so it does not stop.

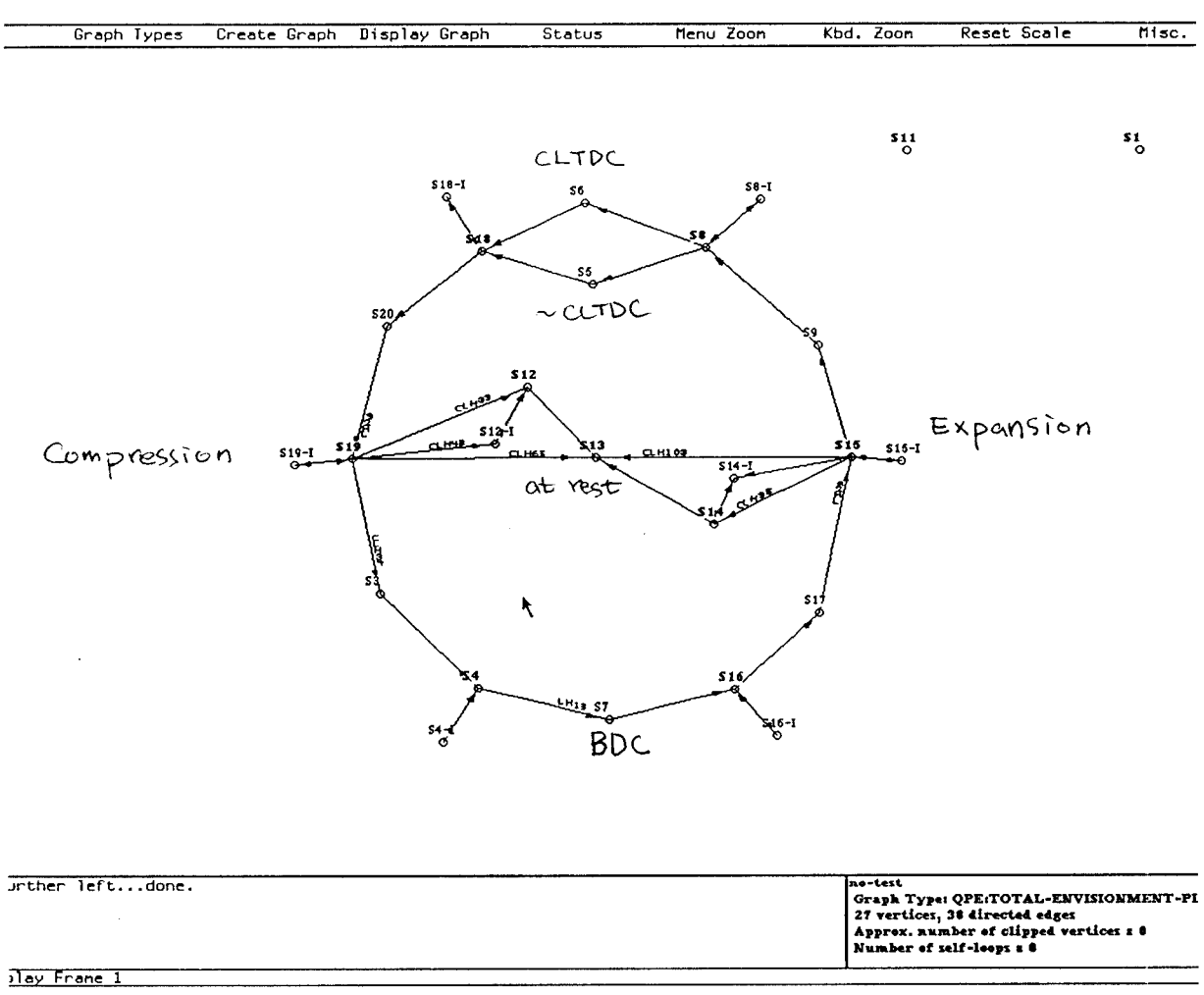


Figure 7.10: Envisionment of a piston-cylinder system without event CLTDC.

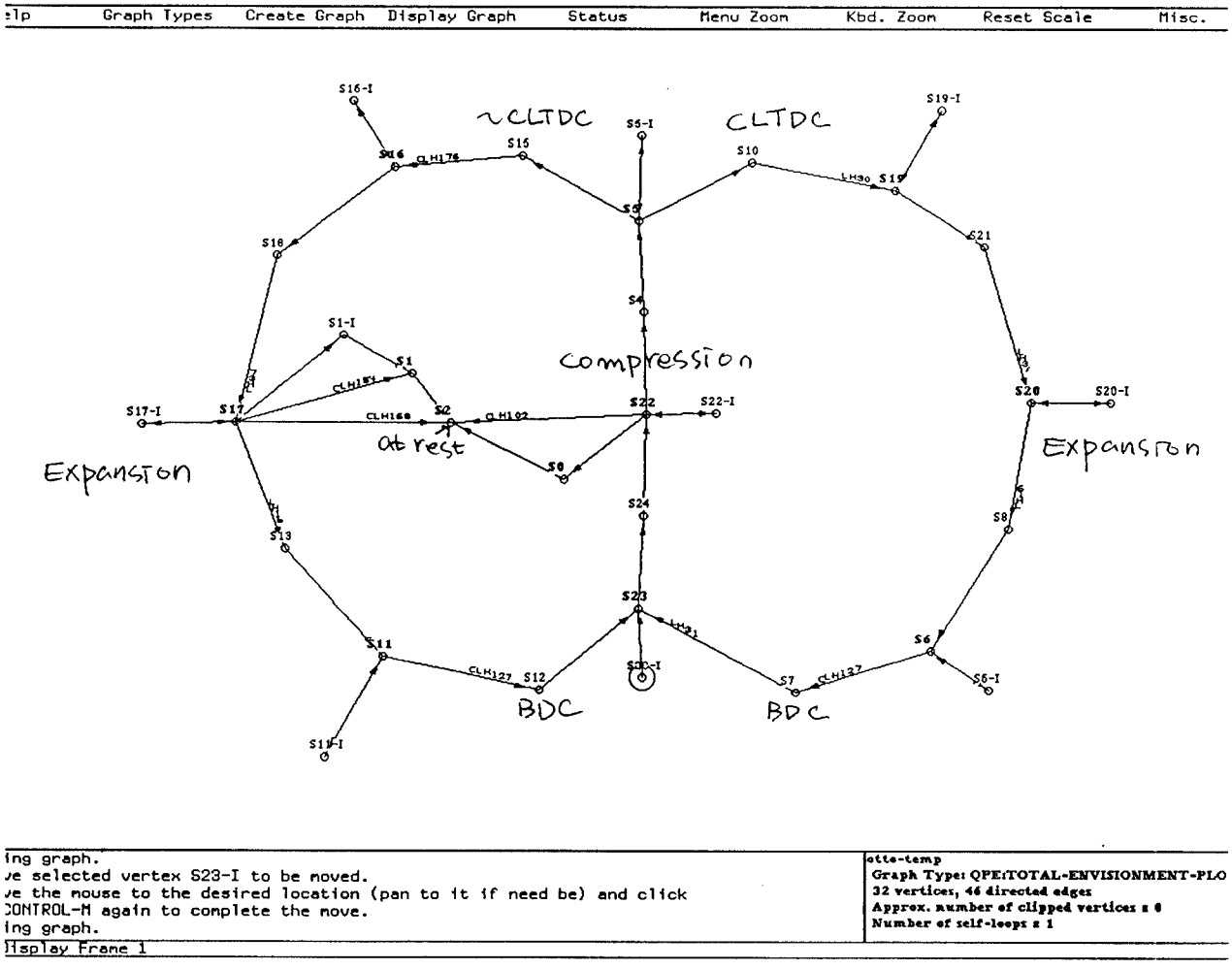


Figure 7.11: Envisionment of a piston-cylinder system with event CLTDC.

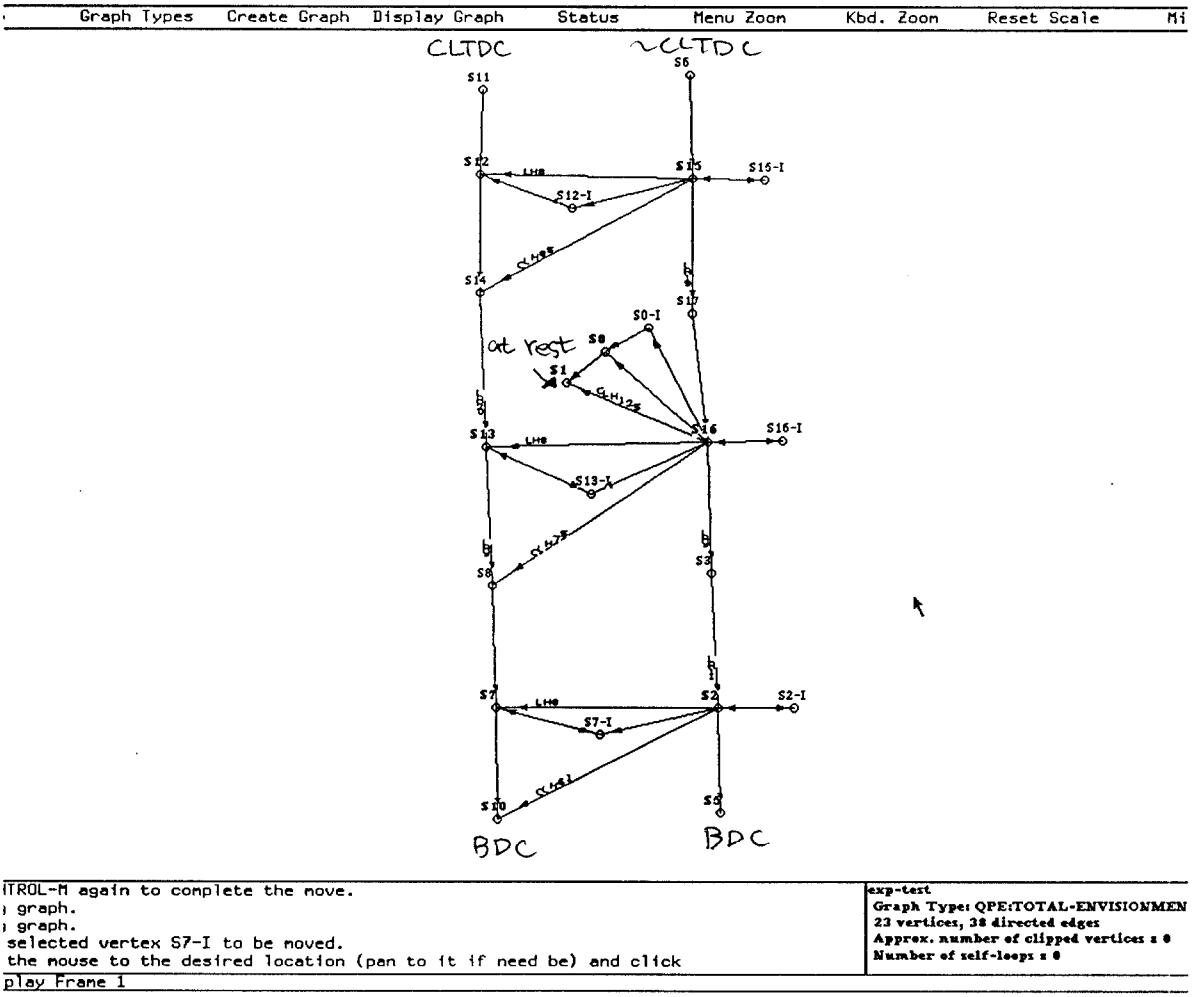


Figure 7.12: Expansion behavior of a piston-cylinder system with the event CLTDC.

Chapter 8

The QSA System

In this thesis, we have developed domain theories of fluids, linkages, and motion, focusing on geometric interactions between each kind of object. We have also explored techniques for reasoning about changes, such as global filtering, angular change, and discontinuous change. This chapter describes QSA, a system which demonstrates the utility of these theories by combining them to reason about systems involving both fluids and motions. Previous chapters described each of the domain theories and reasoning technique in isolation. Here we provide a unified description of the implementation.

8.1 Organization

Given a scenario description, QSA calculates a physical system's behaviors. QSA consists of three parts (Figure 8.1):

- Domain theory: A domain theory consists of quantities, individuals, relationships, individual views and processes. QSA's domain theories describe directions, fluids, and linkages. They also include angular quantities, representations of global constraints, and discontinuous processes.

    1. Qualitative vectors: Positions, forces, and motions are represented using the qualitative vector representation of Chapter 2.
    2. Linkages: The configurations and motions of linkages are represented using the qualitative kinematics of Chapter 3.
    3. Fluids: Fluids and their motions are represented using the bounded-stuff ontology on Chapter 4.
    4. Discrete processes: Discrete changes are represented as a new kind of influence using the discrete processes of Chapter 6.
    5. Events: Global behaviors are represented via events (Chapter 7).

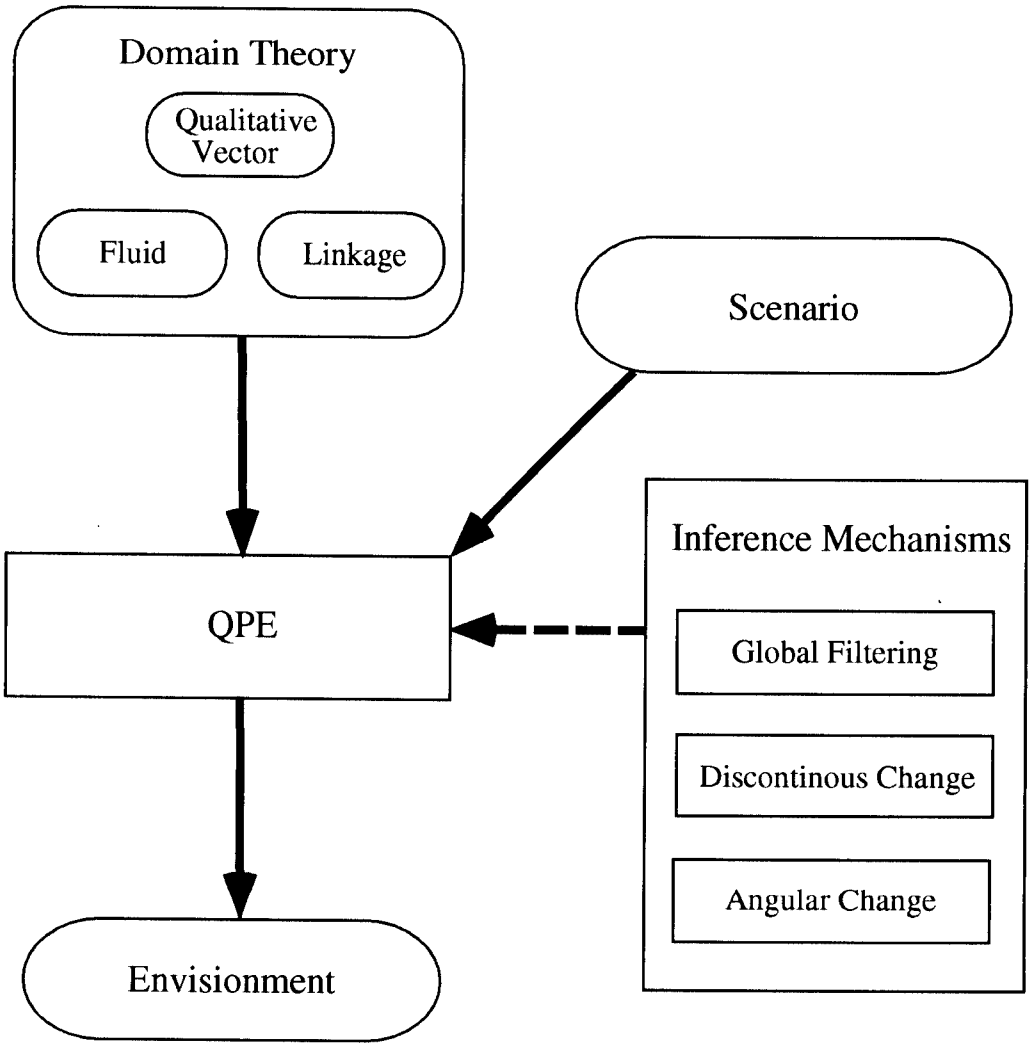


Figure 8.1: Overview of QSA.

- QPE: Forbus' qualitative process engine (QPE) is used as the qualitative simulation program. It produces envisionments by instantiating the domain theory given a scenario description. It identifies state transitions by continuous changes.
- Inference modules: These are used to make additional inferences about the behaviors of systems which cannot be predicted by QPE alone (Figure 8.2).

    1. Global filtering: This module produces sets of views and processes for events on which QPE can apply global filtering constraints (Chapter 7).
    2. Angular changes: This module finds transitions due to angular changes during the limit analysis stage of QPE (Chapter 3).
    3. Discontinuous changes: This module finds transitions due to discontinuous processes after the limit analysis stage of QPE (Chapter 6).

The domain theory and inference rules of QSA use the modeling language of QPE. They, like QPE, are implemented using an assumption based truth maintenance system (ATMS) (deKleer, 1986) and the ATMoSphere rule engine developed by Forbus and deKleer.

8.1.1 Algorithms

Figure 8.2 shows the information flow between QSA's modules. QSA begins with a scenario description of a system and the domain theories.

If we explain the flow with QPE in the center, we can classify the other modules into two categories:

1. preprocessing: producing sets of initial environments which are then fed to QPE. This generation phase happens via extra choice sets and constraints added to QPE. A choice set for a term is the potential range of values for the term. For example, every possible assumption about the relative magnitudes of a pair of numbers (>, <, =, ?) forms a choice set. Choice sets define the space of behaviors explored by QPE.
2. postprocessing of QPE envisionment.

Here we outline how these modules interact to produce the envisionment of a given system. The processing occurs in the following five major steps:

1. Load the domain theory.
2. Load the scenario.
3. Find all possible states.
4. Find all possible transitions due to continuous changes, including angular changes.
5. Find all possible transitions due to discontinuous changes.

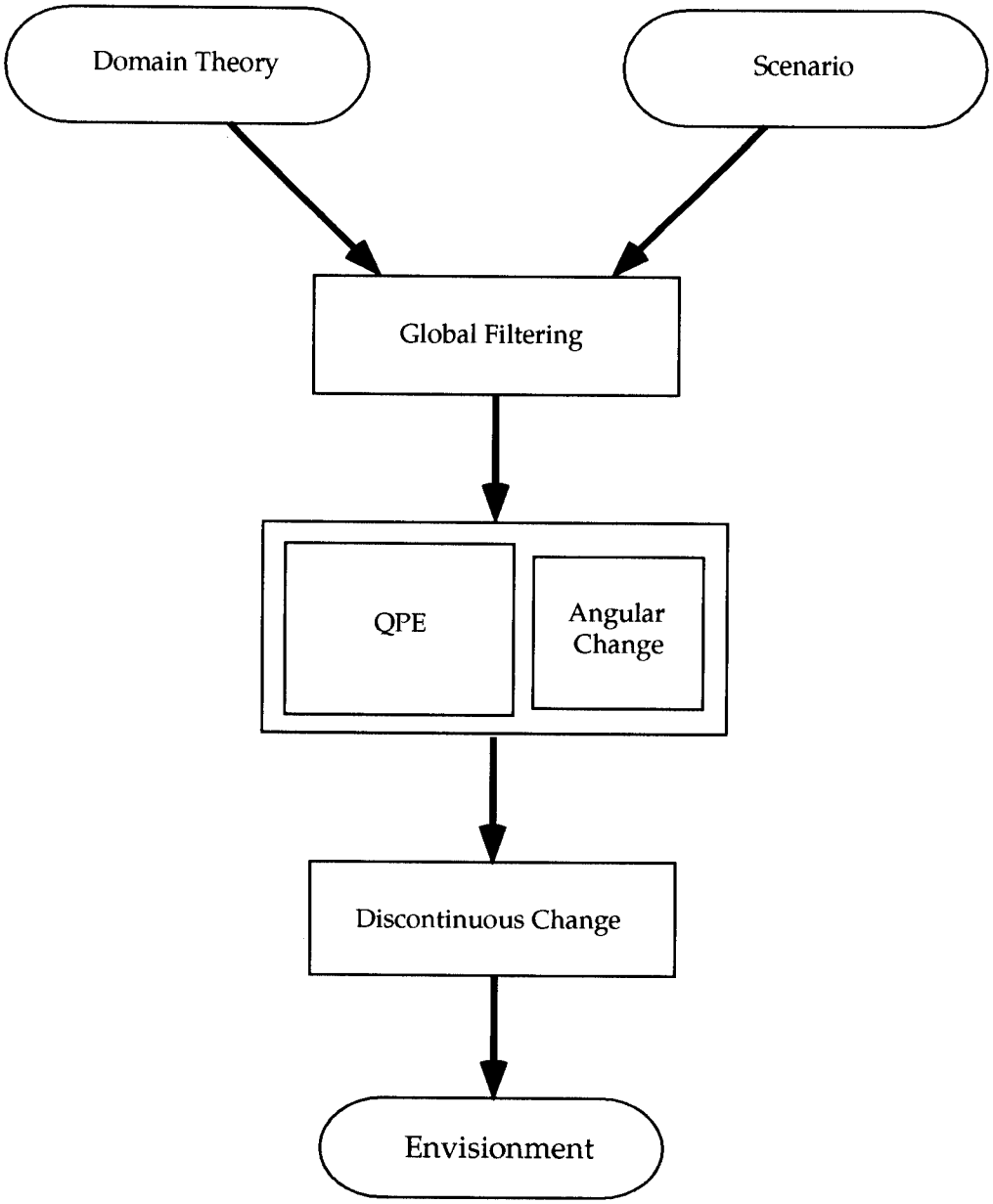


Figure 8.2: Information flow in QSA.

We describe each in turn.

Step 1: Load the domain theory.
The domain theory of QSA is specified in the modeling language of QPE, extended with **defevent** and **discrete process**. For the extensions to be processed by QPE,

- the global filtering module turns every event into corresponding views and processes with an extra variable generated.
- the discontinuous change module generates extra choice sets including the preconditions and quantity conditions of discrete processes. This is done by generating views which include the conditions of each discrete process.

Step 2: Load the scenario.
QPE builds a scenario model of the scenario by instantiating the general descriptions from the domain theory on the particular entities described in the scenario.

Step 3: Find all possible states.
The normal envisioning algorithm of QPE is used to generate all possible states.

Step 4: Find all possible transitions caused by continuous changes, including angular changes.

1. QPE finds transitions by its limit analysis procedure, without considering angular changes.
2. The angular change module filters out transitions which violate the continuity of angular change.

Step 5: Find all possible transitions caused by discontinuous changes.
For each discrete process instance,

1. Find the set of states in which its preconditions and quantity conditions hold.
2. For each state found, find transitions to set of states which describe the possible result of the instance on the state.

8.2 Lift Pump Example

The total envisionment for the lift pump example of Chapter 4 will now be presented. A scenario of the lift pump in Figure 8.3 is given in Chapter 4.

8.2.1 Assumptions

In addition to the descriptions shown in Chapter 4, we added assumptions to our scenario in order to reduce the envisionment size. The assumptions are:

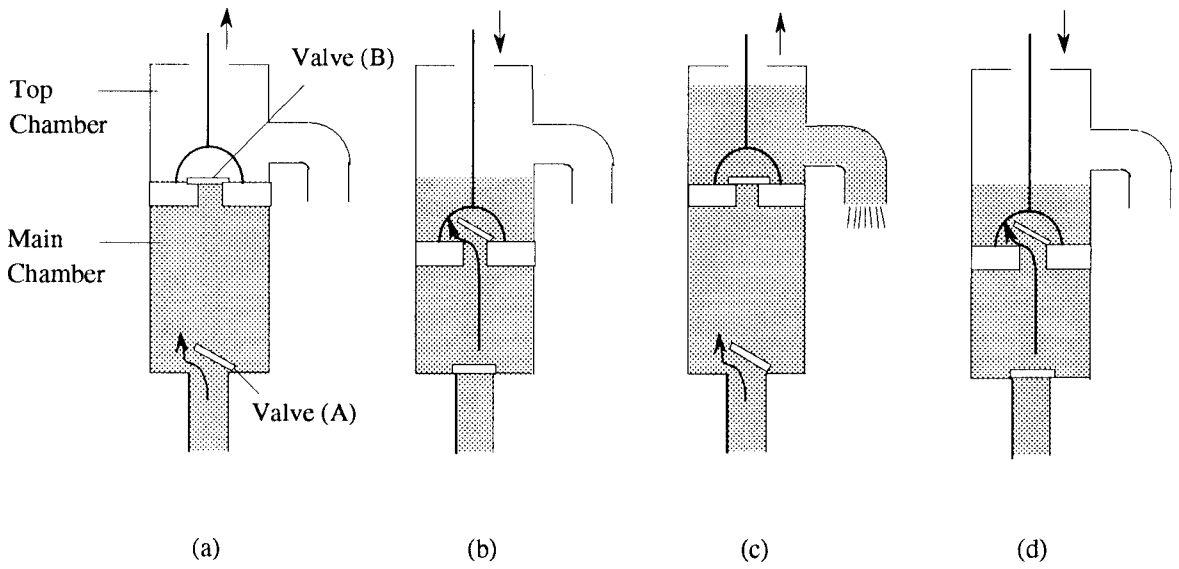


Figure 8.3: Lift Pump Example

- The pump is always primed with water and the water reservoir is always filled with water (Figure 8.4a).
- The level of water inside the pump is higher than the level inside the reservoir (Figure 8.4b). This is the reason why we use lift pumps. If the level is lower, the envisionment will be much simpler since it is basically water flow from the higher place (i.e., the water reservoir) to the lower place (i.e., the pump). In the case of same levels, the behavior will be the same as the scenario with the lower level.
- The downward external force applied to the piston is big enough to open the valve(B) when the piston touches the water in the main chamber. It is also big enough to close the valve(A) (Figure 8.4c). Otherwise, the behavior would be trivial since the external force applied to the piston could be ignored.

In addition to these assumptions, three assumptions about derivatives are made to focus on how the bounded-stuff plays a key role in capturing the behavior of the pump:

- When air flows from the main chamber to the top chamber and the piston is pushed down, the piston is pushed hard enough to increase the pressure of the air in the main chamber (Figure 8.5a).
- When the piston and the water in the main chamber move into the same direction (i.e., up or down), the volume of the main chamber changes in the same rate as the amount of the water in the main chamber changes (Figure 8.5b).

- The direction of the external force applied to the piston and of the motion of the piston are same (Figure 8.5c). The envisionment graph does not include transient acceleration of the piston.

Since QPE distinguishes states by the derivatives of quantities and the set of active processes and views, these assumptions make the envisionment smaller by avoiding unnecessary distinctions due to the ambiguous status of derivatives, processes, and views.

In our current system, gas diffusion through liquid is not modeled. Thus, we assume that the top chamber does not contain water when there is air inside the main chamber:

```
(adb:rule :intern
          (((greater-than (A (amount-of-in water liquid top-chamber))
                          ZERO) . :TRUE) :var ?a1
           ((greater-than (A (amount-of-in air gas main-chamber))
                          ZERO) . :TRUE) :var ?a2)
  (adb:rnogood (?a1 ?a2)) )
```

Modeling gas bubbles moving through liquid remains an interesting future research goal.

8.2.2 Envisionment of the Lift Pump

The QPE envisionment for the scenario is shown in Figure 8.6. Parts of the envisionment are shown in Figure 8.7 through Figure 8.14. The figures show the behavior graphically with partial state descriptions. While the manual for the pump does not mention the possibility of vacuum in the main chamber during its operation, our envisionment also exhibits such behaviors. Vacuum is produced when the piston is raised with the main chamber full of with water and the valve(B) closed as shown in Figure 8.9.

Notice there is no transition from s28 to s31 in Figure 8.10. While the transition represents a legal change, QPE filters it out since the transition violates continuity: external force applied to the water in the main chamber changes from zero to negative since the force is exerted when they are in contact. This changes prevents the changes from s4 to s29, from s9 to s29, from s3 to s30, and from s8 to s30 in Figure 8.10 and Figure 8.12. While the states in the cluster in the envisionment marked with * represent legal states, QPE could not find their transitions due to this problem or the assumptions made about the derivatives.

Let us examine how the envisionment captures the behavior of the lift pump, by comparing it to the explanation in its manual.

> **Step 1:** *Lifting up the piston opens valve(A), sucking water up into the main chamber.* $\Longleftrightarrow$ (s16 $\rightarrow$ s18 $\rightarrow$ s27), (s16 $\rightarrow$ s27), (s26 $\rightarrow$ s17)
>
> **Step 2:** *Pushing the piston down closes valve(A) by force of the water. Water is lifted up to top chamber as valve(B) opens.* $\Longleftrightarrow$ (s11 $\rightarrow$ s28 $\rightarrow$ s31 $\rightarrow$ s30), (s7 $\rightarrow$ s28 $\rightarrow$ s31 $\rightarrow$ s30) (While we did not show it here, s6 is the state same as s17 except the direction of the piston. While QPE did not find the transition due to the reason explained above, there is also a path staring s6: (s6 $\rightarrow$ s31 $\rightarrow$ s30).)

```
(assertq (greater-than (A (amount-of-in water liquid main-chamber)) ZERO))
(assertq (greater-than (A (amount-of-in water liquid reservoir)) ZERO))
                                    (a)

(adb:rule :intern
  (((quantity (fluid-level main-chamber)) . :TRUE) :var ?f1
   ((quantity (fluid-level reservoir)) . :TRUE)    :var ?f2)
  (adb:rule :intern
    (((equal-to (A (fluid-level main-chamber))
                (A (fluid-level reservoir) )) . :TRUE) :var ?e)
    (adb:rnogood (?f1 ?f2 ?e)) )
  (adb:rule :intern
    (((less-than (A (fluid-level main-chamber))
                         (A (fluid-level reservoir))) . :TRUE) :var ?l)
    (adb:rnogood (?f1 ?f2 ?l)) ))

(adb:rule :intern
  (((quantity (fluid-level top-chamber)) . :TRUE) :var ?f1
   ((quantity (fluid-level reservoir)) . :TRUE)   :var ?f2)
  (adb:rule :intern
    (((equal-to (A (fluid-level top-chamber))
                        (A (fluid-level reservoir))) . :TRUE) :var ?e)
    (adb:rnogood (?f1 ?f2 ?e)) )
  (adb:rule :intern
    (((less-than (A (fluid-level top-chamber))
                         (A (fluid-level reservoir))) . :TRUE) :var ?l)
    (adb:rnogood (?f1 ?f2 ?l)) ))
                                    (b)

(adb:rule :intern
          (((full main-chamber) . :TRUE) :var ?full
           ((less-than (A (force-between handle piston)) ZERO) . :TRUE) :var ?push)
  (adb:rule :intern
            (((aligned (bottom top-chamber) (top main-chamber)) . :FALSE)
             :var ?not-aligned)
    (adb:rnogood (?full ?push ?not-aligned)) )
  (adb:rule :intern
            (((aligned (bottom main-chamber) (bottom reservoir)) . :TRUE)
             :var ?aligned)
    (adb:rnogood (?full ?push ?aligned)) ))
                                    (c)
```

Figure 8.4: Assumptions made in the lift pump scenario.

```
(adb:rule :intern
  (((process-instance gas-flow ?x ?sub main-chamber top-chamber
            . ?rest) . :TRUE)
   ((active ?x) . :TRUE)   :var ?gflow
   ((less-than (A (force-between handle piston)) ZERO) . :TRUE) :var ?push)
  (adb:rule :intern
    (((equal-to (D (flow-rate ?x)) ZERO) . :TRUE)  :var ?e)
    (adb:rnogood (?gflow ?push ?e)) )
  (adb:rule :intern
    (((less-than (D (flow-rate ?x)) ZERO) . :TRUE) :var ?l)
    (adb:rnogood (?gflow ?push ?l)) ))
                                    (a)

(adb:rule :intern
          (((greater-than (A (force-between handle piston)) ZERO) . :TRUE)
           :var ?f-g
   ((equal-to (A (force-between handle piston)) ZERO) . :TRUE)
   :var ?f-e
   ((less-than (A (force-between handle piston)) ZERO) . :TRUE)
   :var ?f-l
   ((greater-than (A (velocity piston)) ZERO) . :TRUE)  :var ?v-g
   ((equal-to (A (velocity piston)) ZERO) . :TRUE)      :var ?v-e
   ((less-than (A (velocity piston)) ZERO) . :TRUE)     :var ?v-l)
    (adb:rnogood (?f-g ?v-e))
    (adb:rnogood (?f-g ?v-l))
    (adb:rnogood (?f-e ?v-g))
    (adb:rnogood (?f-e ?v-l))
    (adb:rnogood (?f-l ?v-g))
    (adb:rnogood (?f-l ?v-e)) )
                                    (b)

(adb:rule :intern
  (((greater-than (D (volume (C-S ?sub LIQUID main-chamber)))
            (D (volume main-chamber)) ) . :TRUE) :var ?g
     ((less-than (D (volume (C-S ?sub LIQUID main-chamber)))
                 (D (volume main-chamber)) ) . :TRUE) :var ?l)
  (adb:rule :intern
    (((less-than (A (velocity piston)) ZERO) . :TRUE) :var ?vel
     ((down-liquid-motion main-chamber water ?bnd) . :TRUE) :var ?mot)
    (adb:rnogood (?vel ?mot ?g))
    (adb:rnogood (?vel ?mot ?l)) )
  (adb:rule :intern
    (((greater-than (A (velocity piston)) ZERO) . :TRUE) :var ?vel
     ((up-liquid-motion main-chamber water ?bnd) . :TRUE)    :var ?mot)
    (adb:rnogood (?vel ?mot ?g))
    (adb:rnogood (?vel ?mot ?l)) ) )
                                    (c)
```

Figure 8.5: Assumptions made in the lift pump scenario (cont.).

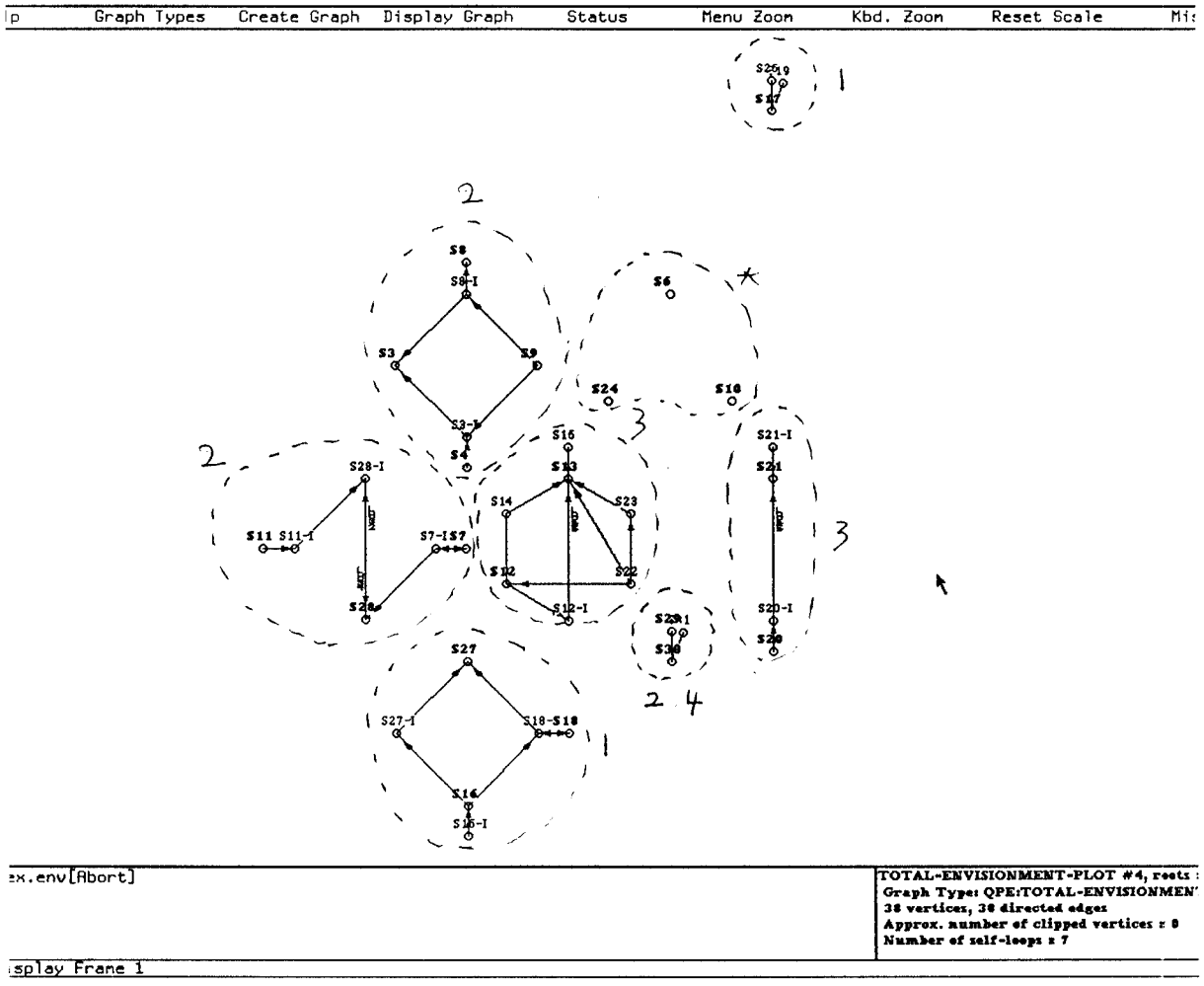


Figure 8.6: The envisionment of the lift pump. In the envisionment, there is no cycle for entire sequence of states as explained in Chapter 4 since each step is connected by a discontinuous change of the external force applied to the piston by an external agent and original QPE does not handle this.

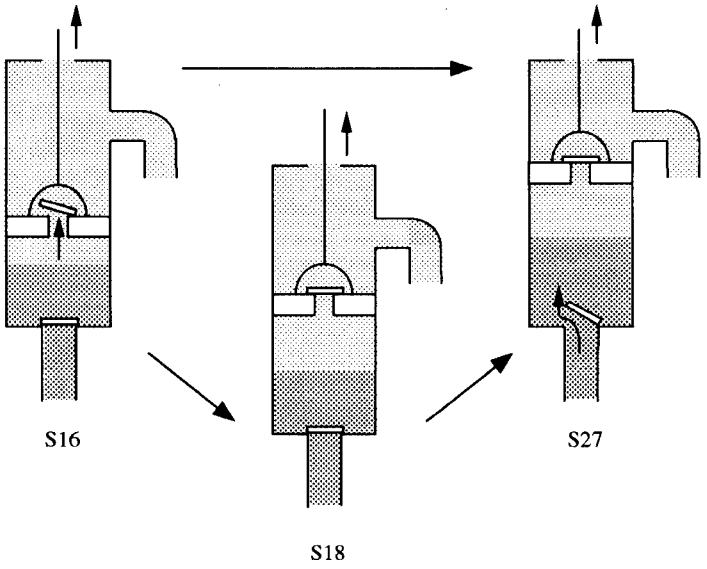


Figure 8.7: Part of the envisionment corresponding to step 1.

Step 3: *Lifting the piston up again shuts valve(B) and pushes water out of the top chamber. Also water is sucked up into the main chamber as valve(A) opens.* $\Longleftrightarrow$ (s22 $\rightarrow$ s23 $\rightarrow$ s13), (s22 $\rightarrow$ s12 $\rightarrow$ s13)

Step 4: *Pushing the piston down again opens valve(B) by force of water. Also water flows into the top chamber.* $\Longleftrightarrow$ (s29 $\rightarrow$ s30), (s9 $\rightarrow$ s4 $\rightarrow$ s3 $\rightarrow$ s30), (s9 $\rightarrow$ s8 $\rightarrow$ s3 $\rightarrow$ s30), (s9 $\rightarrow$ s3 $\rightarrow$ s30), (s9 $\rightarrow$ s4 $\rightarrow$ s29 $\rightarrow$ s30), (s9 $\rightarrow$ s8 $\rightarrow$ s30)

While our trace includes paths exactly corresponding to the explanation in the manual, our envisionment also includes other possible behaviors as we showed previously using the figures.

Quantity	s16	s18	s27
Ds[Amount-of-in(water,liquid,M)]	0	0	1
Ds[Flow-rate(PI4)]	-1	?	?
Ds[Flow-rate(PI7)]	?	?	-1
Ds[Fluid-level(M)]	0	0	1
Ds[Position(piston)]	1	1	1
Ds[Pressure(M)]	-1	-1	-1
Ds[Pressure(t-s(M,B-S(water,liquid,M)))]	-1	-1	?
Ds[Pressure(t-s(M,B-S(water,liquid,M-R)))]	?	?	1
Ds[volume(M)]	1	1	1
A[Pressure(T)] A[Pressure(M)]	<	>=	>=
A[Pressure(t-s(M,B-S(water,liquid,M)))] A[Pressure(t-s(R,B-S(water,liquid,R)))]	>	>=	?
A[Pressure(t-s(M,B-S(water,liquid,M-R)))] A[Pressure(t-s(R,B-S(water,liquid,M-R)))]	?	?	<
Active(PI2)	T(up)	T(up)	T(up)
Active(PI3)	T(up)	T(up)	T(up)
Active(PI4)	T	F	F
Active(PI7)	F	F	T

Processes:

```
PI2: Motion(piston)
PI3: Acceleration(piston,handle)
PI4: Gas-flow(air,M,T)
PI7: Liquid-surface-motion(M,R,liquid,M-R)
```

Bounded-liquids:

```
s16: Bounded-Liquid(M),Bounded-Liquid(R)
s18: Bounded-Liquid(M),Bounded-Liquid(R)
s27: Bounded-Liquid(M-R)
```

Figure 8.8: Partial state descriptions for the states of Figure 8.7.

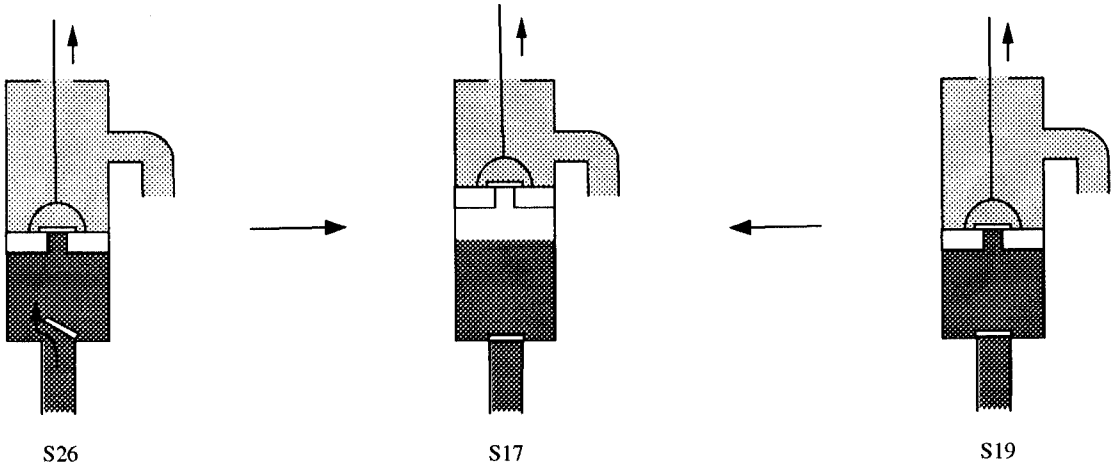


Figure 8.9: Part of the envisionment corresponding to step 1 with vacuum in the main chamber.

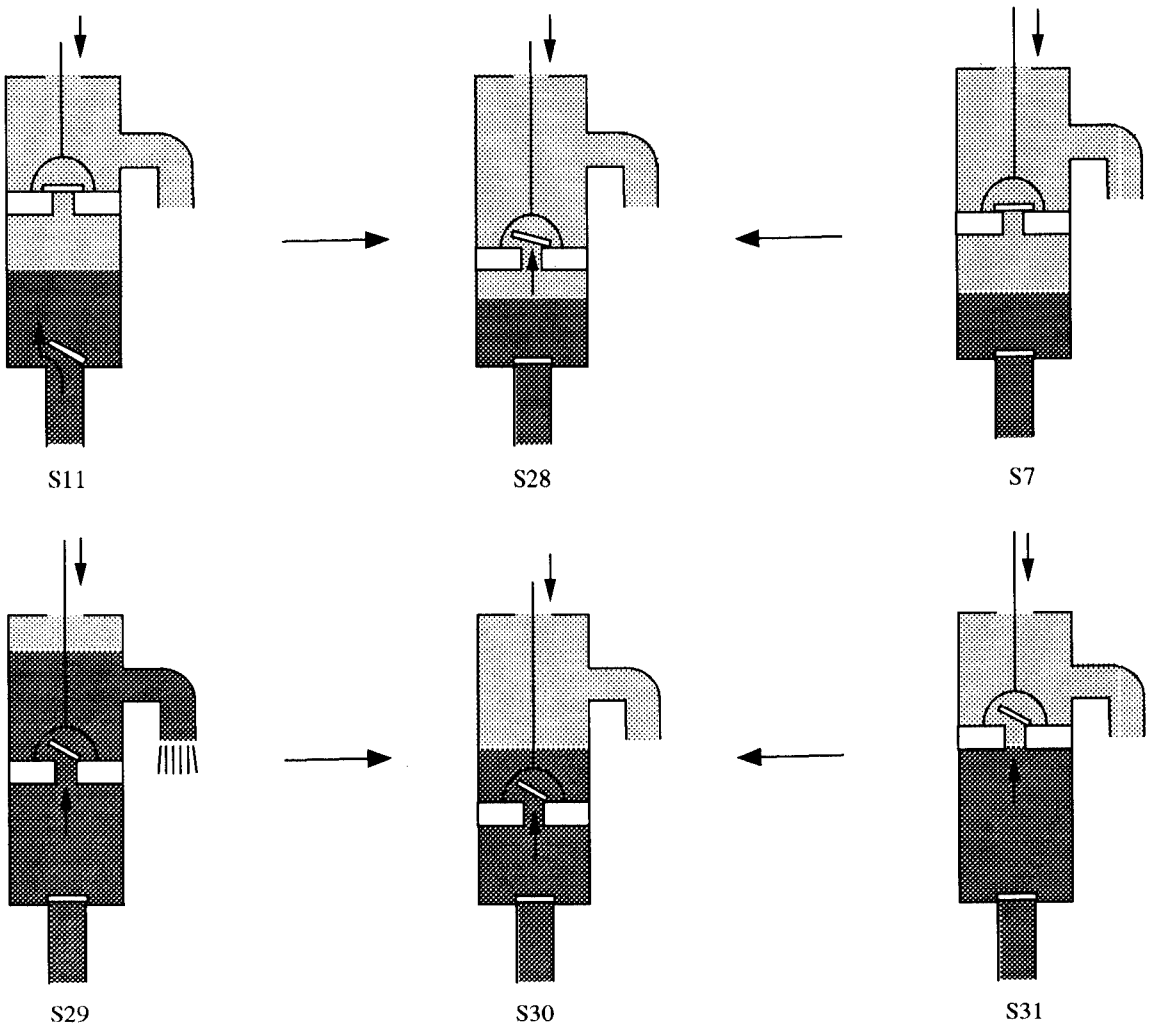


Figure 8.10: Part of the envisionment corresponding to step 2 and 4.

Quantity	s11	s28	s7	s29	s30	s31
Ds[Amount-of-in(water,liquid,M)]	1	0	0	-1	-1	-1
Ds[Amount-of-in(water,liquid,T)]	0	0	0	1	1	1
Ds[Flow-rate(VI8)]	?	?	?	-1	?	?
Ds[Flow-rate(PI4)]	?	1	?	?	?	?
Ds[Flow-rate(PI7)]	-1	?	?	?	?	?
Ds[Fluid-level(M)]	1	0	0	?	?	-1
Ds[Fluid-level(T)]	?	?	?	-1	0	0
Ds[Position(piston)]	-1	-1	-1	-1	-1	-1
Ds[Pressure(M)]	1	1	1	0	0	0
Ds[Pressure(t-s(M,B-S(water,liquid,M)))]	?	1	1	?	?	-1
Ds[Pressure(t-s(M,B-S(water,liquid,M-R)))]	1	?	?	?	?	?
Ds[Pressure(t-s(T,B-S(water,liquid,M-T)))]	?	?	?	-1	0	0
Ds[volume(M)]	-1	-1	-1	-1	-1	-1
A[Pressure(T)] A[Pressure(M)]	>=	<	>=	=	=	=
A[Pressure(t-s(M,B-S(water,liquid,M)))] A[Pressure(t-s(R,B-S(water,liquid,R)))]	?	>=	>=	?	?	>=
A[Pressure(t-s(M,B-S(water,liquid,M-R)))] A[Pressure(t-s(R,B-S(water,liquid,M-R)))]	<	?	?	?	?	?
A[Pressure(t-s(T,B-S(water,liquid,M-T)))] A[Pressure(t-s(R,B-S(water,liquid,R)))]	?	?	?	>=	>=	>=
Active(VI8)	F	F	F	T	F	F
Active(PI2)	T(dn)	T(dn)	T(dn)	T(dn)	T(dn)	T(dn)
Active(PI3)	T(dn)	T(dn)	T(dn)	T(dn)	T(dn)	T(dn)
Active(PI4)	F	T	F	F	F	F
Active(PI7)	T	F	F	F	F	F

Processes and Views:

```
PI2: Motion(piston)
PI3: Acceleration(piston,handle)
PI4: Gas-flow(air,M,T)
PI7: Liquid-surface-motion(M,R,liquid,M-R)
VI8: Pouring(water,P1,T)
```

Bounded-liquids:

```
s11: Bounded-Liquid(M-R)
s28: Bounded-Liquid(M),Bounded-Liquid(R)
s7:  Bounded-Liquid(M),Bounded-Liquid(R)
s29: Bounded-Liquid(M-T),Bounded-Liquid(R)
s30: Bounded-Liquid(M-T),Bounded-Liquid(R)
s31: Bounded-Liquid(M-T),Bounded-Liquid(M),Bounded-Liquid(R)
```

Figure 8.11: Partial state descriptions for the states of Figure 8.10.

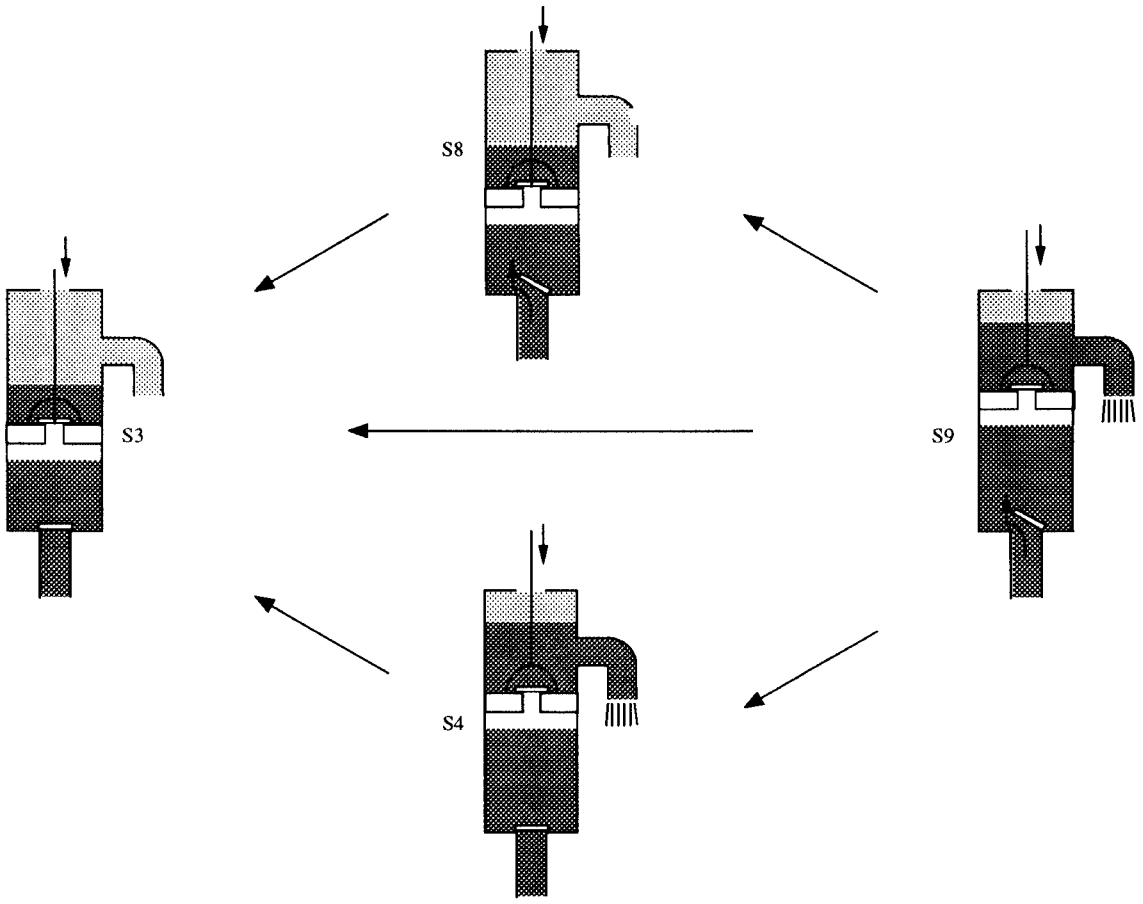


Figure 8.12: Part of the envisionment corresponding to step 4 with vacuum in the main chamber.

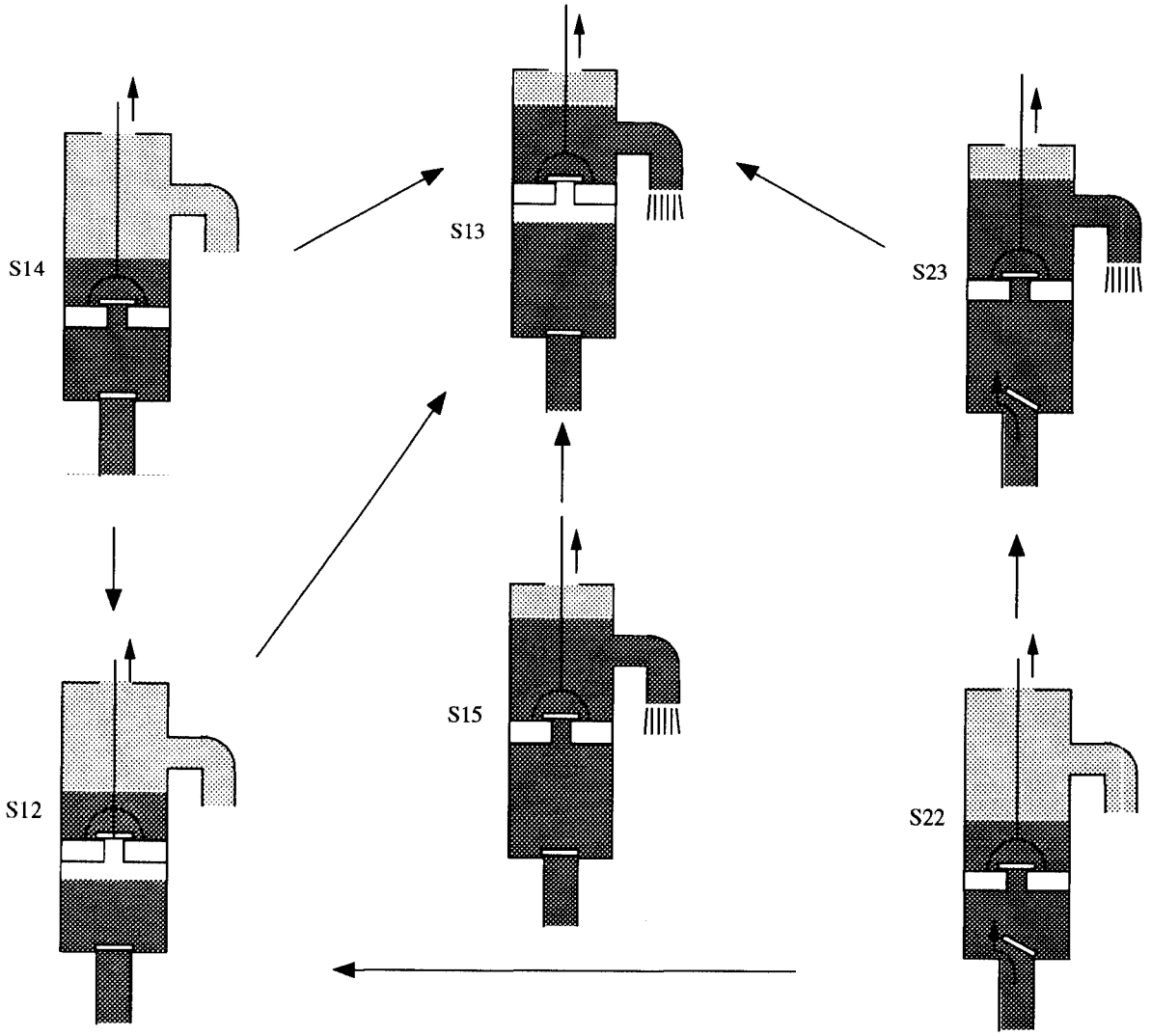


Figure 8.13: Part of the envisionment corresponding to step 3.

Quantity	s14	s13	s23	s12	s15	s22
Ds[Amount-of-in(water,liquid,M)]	0	0	1	0	0	1
Ds[Flow-rate(VI8)]	?	1	1	?	1	?
Ds[Flow-rate(PI7)]	?	?	-1	?	?	-1
Ds[Fluid-level(M)]	0	0	1	0	0	1
Ds[Fluid-level(T)]	1	1	1	1	1	1
Ds[Position(piston)]	1	1	1	1	1	1
Ds[Pressure(t-s(M,B-S(water,liquid,M)))]	0	0	?	0	0	?
Ds[Pressure(t-s(M,B-S(water,liquid,M-R)))]	?	?	1	?	?	1
Ds[Pressure(t-s(T,B-S(water,liquid,T)))]	1	1	1	1	1	1
Ds[volume(M)]	1	1	1	1	1	1
A[Pressure(t-s(M,B-S(water,liquid,M)))] A[Pressure(t-s(R,B-S(water,liquid,R)))]	>=	>=	?	>=	>=	?
A[Pressure(t-s(M,B-S(water,liquid,M-R)))] A[Pressure(t-s(R,B-S(water,liquid,M-R)))]	?	?	<	?	?	<
Active(VI8)	F	T	T	F	T	F
Active(PI2)	T(up)	T(up)	T(up)	T(up)	T(up)	T(up)
Active(PI3)	T(up)	T(up)	T(up)	T(up)	T(up)	T(up)
Active(PI7)	F	F	T	F	F	T

Processes and Views:

```
PI2: Motion(piston)
PI3: Acceleration(piston,handle)
PI7: Liquid-surface-motion(M,R,liquid,M-R)
VI8: Pouring(water,P1,T)
```

Bounded-liquids:

```
s14: Bounded-Liquid(M-T),Bounded-Liquid(R)
s13: Bounded-Liquid(M),Bounded-Liquid(R),Bounded-Liquid(T)
s23: Bounded-Liquid(M-R),Bounded-Liquid(T)
s12: Bounded-Liquid(M),Bounded-Liquid(R),Bounded-Liquid(T)
s14: Bounded-Liquid(M),Bounded-Liquid(R),Bounded-Liquid(T)
s13: Bounded-Liquid(M-R),Bounded-Liquid(T)
```

Figure 8.14: Partial state descriptions for d the states of Figure 8.13.

8.3 Internal Combustion Engine Example

One motivation of this thesis was to capture the qualitative behaviors of an internal combustion engine. Partial analyses have already been shown in previous chapters, here we present the overall analysis of this example.

A scenario description for this example appears in Figures 8.15 through 8.17. The description about the surfaces of the piston and cylinder, and how they are related is shown in Figure 8.16 and Figure 8.17.

Surfaces are defined with the `defentity` form:

```
(defentity (Surface (surf ?obj ?surf)) )
```

In the figures, `KIN` proceeds a kinematic description. `INIT` represents the initial configuration of a given system. It does not necessarily imply that our envisionment will start from the configuration, but it filters some configurations. For example, the envisionment will not include the behavior of the piston outside of the cylinder.

We assume a spark may occur as soon as the crankshaft reaches `TDC` and that the spark lasts for an instant (Figure 8.18a). If we allowed sparking during the expansion period, the crankshaft will be accelerated more. In the compression period, the parts might reverse their motion if the pressure built up is big enough.

Combustion is understood as an abrupt pressure increase (Chapter 7) and `combustion-pres` represents this change. This pressure can be reached only by the combustion process at `TDC`.

Figure 8.19 shows all possible configurations of the internal combustion engine. The domain model of linkages determines all possible configurations and motions for the slider-crank mechanism of the engine. Each state generated by `QSA` is distinguished from others by the mechanical configurations, the pressure in the cylinder, and the motions of each part. Each transition represents changes in dynamics and geometry, including angular changes and discontinuous changes due to combustion. The power cycle is distinguished from expansion cycle without combustion at `TDC` by the global filtering module.

Figure 8.20 shows a partial description of all possible states at `TDC`. All states except `s1` have transitions from the compression period. `s1` represents the state after combustion. `s4` represents the system at rest. The system can stop in every possible configuration of the slider crank mechanism. The global filtering module splits the expansion period, including `TDC` and `BDC`, by introducing two different views for the period (see Figure 7.11). `s1` has a transition to `CLTDC-EXP` while `s3` has a transition to ~`CLTDC-EXP`. There are two discontinuous changes: one for spark ignition and the other for the combustion process from `s2` to `s1`.

```
;;; linkage description
(assertq (link-pins-connection crank1 p1 p2))
(assertq (link-pins-connection connecting-rod p2 p3))
(assertq (grounded l3 p1 p3 (:ZERO :PLUS)))
(assertq (length-greater (p2 p3) (p1 p2)))

;;; Parts
(assertq (crank crank1))
(assertq (piston piston1))
(assertq (mobile piston1))
(assertq (attached p3 piston1))
(assertq (part-of cylinder1 piston1))
(assertq (slider-crank piston1 crank1))

;;; assume crank1 does not rotate ccw direction
(assertq (not (greater-than (A (angular-velocity crank1)) ZERO) ))

;;; Gas inside cylinder1
(assertq (assertq (state gas)))
(assertq (substance fuel-mix))
(assertq (container cylinder1))
(assertq (Can-Contain-Substance cylinder1 fuel-mix gas))
(assertq (greater-than (a (amount-of-in fuel-mix gas cylinder1)) ZERO))
(assertq (consider-combustion-at cylinder1))
```

Figure 8.15: Partial scenario description of a simple internal combustion engine.

```
;;; Surface Entities
(assertq (surface (surf piston1 r)))
(assertq (surface (surf piston1 l)))
(assertq (surface (surf piston1 t)))
(assertq (surface (surf piston1 b)))

;;; The surface normals of each surface
(assertq (KIN (surf-norm (surf piston1 r) (:PLUS :ZERO))))
(assertq (KIN (surf-norm (surf piston1 l) (:MINUS :ZERO))))
(assertq (KIN (surf-norm (surf piston1 b) (:ZERO :MINUS))))
(assertq (KIN (surf-norm (surf piston1 t) (:ZERO :PLUS))))

;;; The end-points of each surface
(assertq (KIN (end-point (surf piston1 l) (p1 p2))))
(assertq (KIN (end-point (surf piston1 t) (p2 p3))))
(assertq (KIN (end-point (surf piston1 r) (p3 p4))))
(assertq (KIN (end-point (surf piston1 b) (p1 p4))))

;;; The relative positions of one end-point to
;;; the other point in each surface
(assertq (KIN (rel-pos  piston1 (p1 p2) (:ZERO :MINUS))))
(assertq (KIN (rel-pos  piston1 (p2 p3) (:MINUS :ZERO))))
(assertq (KIN (rel-pos  piston1 (p3 p4) (:ZERO :PLUS))))
(assertq (KIN (rel-pos  piston1 (p4 p1) (:PLUS :ZERO))))
```

Figure 8.16: Surface description of `piston1`.

```
;;; ---- Surface description for cylinder1 ----

;;; Surface Entities
(assertq (surface (surf cylinder1 l)))
(assertq (surface (surf cylinder1 r)))
(assertq (surface (surf cylinder1 t)))
(assertq (surface (surf cylinder1 b)))

;;; The surface normals of each surface
(assertq (KIN (surf-norm (surf cylinder1 l) (:PLUS :ZERO))))
(assertq (KIN (surf-norm (surf cylinder1 r) (:MINUS :ZERO))))
(assertq (KIN (surf-norm (surf cylinder1 t) (:ZERO :MINUS))))
(assertq (KIN (surf-norm (surf cylinder1 b) (:ZERO :PLUS))))

;;; The end-points of each surface
(assertq (KIN (end-point (surf cylinder1 l) (p1 p2))))
(assertq (KIN (end-point (surf cylinder1 t) (p2 p3))))
(assertq (KIN (end-point (surf cylinder1 r) (p3 p4))))
(assertq (KIN (end-point (surf cylinder1 b) (p1 p4))))
;;; The relative positions of one end-point to
;;; the other point in each surface
(assertq (KIN (rel-pos  cylinder1 (p1 p2) (:ZERO :MINUS))))
(assertq (KIN (rel-pos  cylinder1 (p2 p3) (:MINUS :ZERO))))
(assertq (KIN (rel-pos  cylinder1 (p3 p4) (:ZERO :PLUS))))

;;; Initial surface contacts
(assertq (INIT (surface-contact (surf piston1 r) (surf cylinder1 r))))
(assertq (INIT (surface-contact (surf piston1 l) (surf cylinder1 l))))

;;; ---- about the shape ----
(assertq (KIN (= (distance (surf piston1 r) (surf piston1 l))
         (distance (surf cylinder1 r) (surf cylinder1 l)))))
(assertq (KIN (< (distance (surf piston1 b) (surf piston1 t))
 (distance (surf cylinder1 b) (surf cylinder1 t))))  )

;;; piston1 does not touch the top and the bottom of cylinder1
(assertq (greater-than
            (A (distance-between y (surf piston1 t) (surf cylinder1 t)))
            ZERO) )
(assertq (greater-than
            (A (distance-between y (surf piston1 b) (surf cylinder1 b)))
            ZERO) )
```

Figure 8.17: Surface description of `piston1` and `cylinder1` (cont.).

```
(adb:rule :intern (((slider-crank ?pst ?crk) . :TRUE)
                   ((piston ?pst) . :TRUE)
                   ((part-of ?cyl ?pst) . :TRUE)
                   ((crank-at-tdc ?crk) . :FALSE) :var ?not-tdc
              ((spark ?sub ?cyl) . :TRUE) :var ?spark)
  (adb:rnogood (?spark ?not-tdc)) )
                                   (a)

(adb:rule :intern
          (((compression ?pst ?sub ?cyl ?c-g) . :TRUE) :var ?comp
           ((greater-than (A (Combustion-pres (C-S ?sub gas ?cyl)))
             (A (pressure (C-S ?sub gas ?cyl)))) . :TRUE) :var ?p)
  (adb:rnogood (?comp ?p)) )
                                   (b)
```

Figure 8.18: Assumptions made in the simple internal combustion engine scenario.

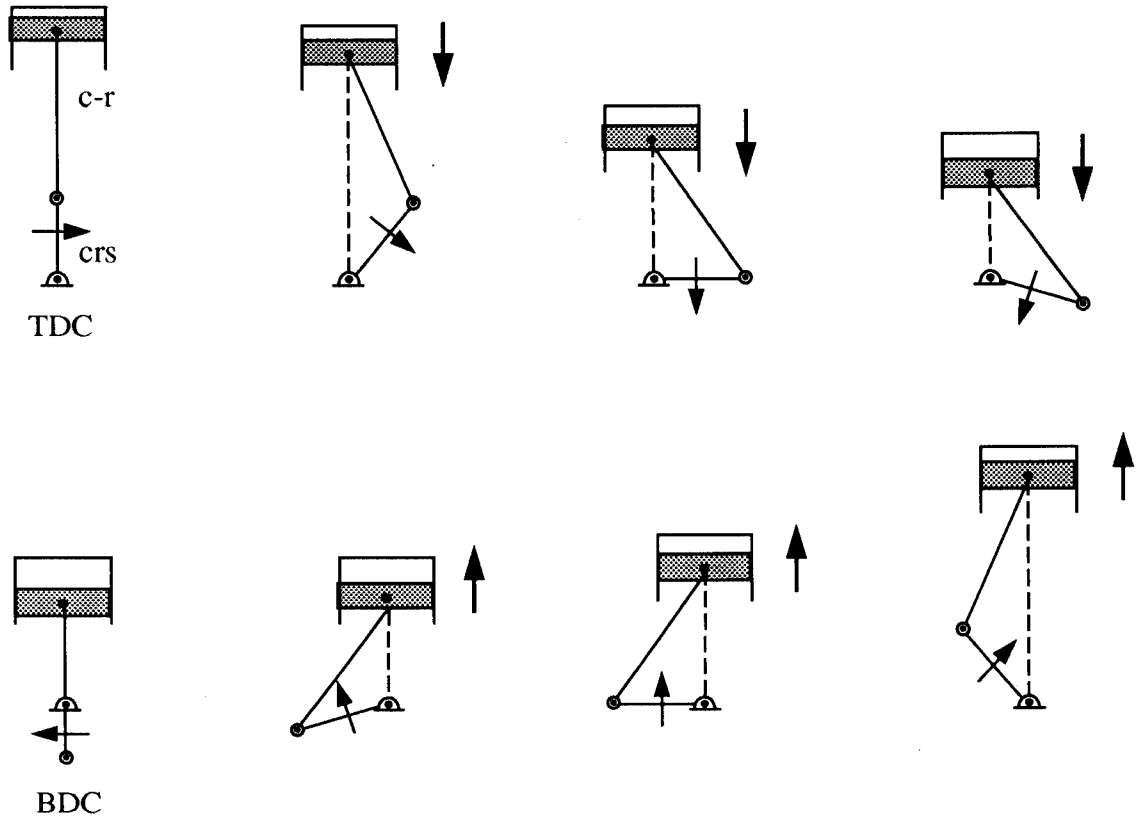


Figure 8.19: Possible configurations for the connecting-rod and crankshaft. Each configuration is shown with its possible motion. Two different states are possible for each configuration: one with the motions shown here and the other without motions.

Quantity	s1	s2	s3	s4
Ds[force(crank1,piston1)]	0	0	0	0
Ds[velocity(piston1)]	-1	-1	-1	0
Ds[position(piston1)]	0	0	0	0
Ds[angular-velocity(crank1)]	1	1	1	0
Ds[pressure(c-g)]	0	0	0	0
A[force(crank1,piston1)]	<0	<0	<0	=0
A[velocity(piston1)]	=0	=0	=0	=0
A[angular-velocity(crank1)]	<0	<0	<0	=0
A[pressure(c-g)] A[combustion-pres]	>	<	<	<
A[relative-position x (p2 p1)]	=0	=0	=0	=0
A[relative-position y (p2 p1)]	>0	>0	>0	>0
A[relative-position x (p3 p2)]	=0	=0	=0	=0
A[relative-position y (p3 p2)]	>0	>0	>0	>0
A[relative-motion (p2 p1)]	<0	<0	<0	=0
A[relative-motion (p2 p1)]	>0	>0	<0	=0
(spark piston1 cylinder1)	true	true	false	false

Figure 8.20: Partial state descriptions for the states at TDC.

Chapter 9

Discussion

This chapter concludes the thesis. We begin by summarizing our results in Section 9.1. Section 9.2, compares our work with related research. Finally, Section 9.3 discusses directions for future research.

9.1 Summary

The primary contributions of this work are:

- We developed a qualitative vector representation which extends Nielsen's version by including sense and inclination. This extension provides the resolution needed to accurately describe the geometric configurations of linkages.
- We developed a representation for geometric constraints and reasoning techniques which use these constraints, plus simple dynamical representation to predict the possible behaviors of two dimensional linkages from qualitative information. Linkages form an important component in many mechanical systems, so this ability to analyze moving-axis mechanisms is important.
- We developed the *bounded-stuff* ontology, a version of the contained-stuff ontology which individuates a piece of liquid by the surfaces it is in contact with. This representation allows the complex fluid geometries often found in mechanisms (like the lift pump) to be described.
- We developed a technique for reasoning about the geometry of laminar fluid flow, which partitions a flowing fluid incrementally into regions where the overall flow direction is constant. This representation captures the intuition used in streamline diagram for understanding fluid systems, including a causal explanation of how an air plane wing produces lift.
- We developed a representation for discontinuous influences, and showed how limit analysis could be extended using this representation to reason about discontinuous changes. Such

changes are commonly used as approximations in describing complex mechanisms, e.g., modeling combustion as a discrete process.

- We developed a representation for events which allows envisionments to more accurately represent global properties of behavior. This representation allows the behaviors of complex mechanisms to be described more accurately.

9.2 Related Work

9.2.1 Forbus' FROB

Forbus (1981) investigated the qualitative and geometric knowledge necessary to reason about motion in space. The problem that FROB solves is, given a description of space and the positions of balls in a state, find possible motions of the balls from the state. Space is described as a set of polygonal, two dimensional regions under the influence of gravity, and a ball is represented as a point mass in space.

An important contribution of FROB was the introduction of the concept of a place vocabulary. A place is defined as "a piece of space (point, line, surface, region, etc) such that all parts of it share some common property". The basic idea is to divide a given space into qualitatively distinct regions that can be symbolically reasoned about. This can be viewed as a quantization of space. Since the motion envisioned is of a point mass, the space is divided using horizontal and vertical lines, without considering the shape of balls.

The work in this thesis extends the concept of place vocabulary in several ways. In Chapter 4, a bounded fluid is individuated by the geometry of the surface contacts between fluids and their boundaries. In Chapter 5, fluid is partitioned so each part has the same qualitative fluid motion. While the original place vocabulary is computed from metric representation, fluids are quantized based on purely qualitative information.

9.2.2 CLOCK

CLOCK (Faltings, 1986; Nielsen, 1988) is a system which derives the behavior of mechanical systems (e.g., gears, cams, and clocks) by extending the framework of FROB.

Faltings developed a general theory of place vocabulary based on configuration space (Lozano-Perez, 1983). The key idea is to find the geometric interaction of a system through the analyses of pairs of parts before integrating them. Given a metric description of a mechanical system, Faltings' program computes the configuration space, i.e, the space of all possible configurations, for each pair of parts. A place vocabulary is computed from the configuration space representation of a system by analyzing the shapes of the surfaces of the objects in contact. The possible transitions between places are also computed.

Nielsen developed a qualitative theory of rigid body statics. His program begins with the place vocabulary of a system computed by Faltings' program and produces an envisionment. Possible directions of motion in each place are computed by the analysis of surface contact

between parts and the forces acting on the system. To represent directions, such as the directions of surface contacts, forces, and velocities, a qualitative vector representation was developed.

While this system successfully generates the envisionments of several mechanical systems, it has several limitations. First, the configuration space approach requires a huge amount of computation and the results are not necessarily intuitive. Second, it focuses on fixed-axis rigid body systems, ignoring the analysis of movable axis mechanisms. Third, it does not deal with fluids, which are parts of many important systems. Lastly, it focuses on systems which require only very simple dynamic analysis and thus ignores many interesting dynamical changes. The present work is partly motivated by these limitations and overcomes these limitations.

9.2.3 Joskowicz and Sacks' Kinematic Simulation

Joskowicz' (1987, 1989a, 1989b) presented a qualitative kinematic analysis of mechanical mechanisms. The basic idea of his approach is very similar to the work of Faltings: computing the configuration space of a given mechanism, partitioning it into spaces of uniform motion, and determining the potential transitions between spaces. His system suffers the same limitations as Faltings'.

Based on Joskowicz' kinematic analysis, Joskowicz and Sacks (1991, 1992, 1993) developed a qualitative simulation program for rigid part mechanisms. Their program derives the behavior of mechanisms by analyzing part contacts and driving motions along with a limited dynamics of gravity, springs, and friction. Due to the intractability of the configuration space representation, they restricted the shapes, motions, and interactions of parts. Their program handles mechanisms which include fixed axes subassemblies and linkages. The program partitions a given mechanism, analyzes the fixed axes subassemblies and the linkages individually, and combines the analyses into an overall behavioral description. Linkages are analyzed with a linkage package. The program derives the behavior of a mechanism from a specific initial configuration under a specific driving motion. They do not represent fluids or discontinuous changes, and therefore they cannot handle the examples `QSA` can.

9.2.4 Randell and Cohn's Formalism

Randell and Cohn (1989) have developed a formalism for the representation of spatial topology. Their representation is based on spatial regions, using a vocabulary of binary relationships which capture intuitive topological relationships. The ordering structure between the relations is represented by a lattice. For instance, *P(x,y)*, i.e., x is part of y, implies *C(x,y)*, i.e., x connects with y, but not vice versa. This ordering structure provides the continuity constraints on transition of configurations over time. Besides the basic relations, *inside*, *outside*, and *partially inside* between two regions are defined by a function *conv(x)*, i.e., the convex-hull of x.

Since the positions of physical objects change as the objects moves, the configuration between objects is described by the spaces occupied by objects at each state rather than the physical objects themselves. A function *space(x,t)* is used to map a physical object x into the

space occupied by x at the time t. By this function, the relations defined for spaces are used to describe the configurations of given physical systems.

The descriptive power of this formalism has been illustrated by a model of a force pump. They have shown how the behavior of the pump can be derived from the representation of the pump and the axioms. Compared to our liquid pump model, their model derives the behavior from functional description rather than physics. Their model violates "no function-in-structure" in several ways. For instance, directionality of liquid flow through a valves is fixed: *In-valve* allows only inflow while *Out-valve* allows only outflow.

9.2.5 Fluid Ontologies

AI research on fluid ontology was started by Hayes (1985). He identified two distinct ontologies: *contained-liquid* and *pieces-of-stuff* ontologies. Forbus (1984) generalized the contained-liquid into the *contained-stuff* ontology to cover liquids and gases. The contained-stuff ontology has been successfully used in most qualitative physics research. On the other hand, Collins and Forbus (1987) proposed the *molecular-collection* (MC) ontology, which can be viewed as a specialization of Hayes' pieces-of-stuff. An MC is large enough to have macroscopic properties such as temperature and pressure. While the contained-stuff ontology is useful for deriving the overall behavior of a system, the MC ontology is useful for tracing the behavior of fluid through a system. For instance, MC can be used to recognize and classify thermodynamic cycles (e.g., recognizing a system as a heat pump). Recent work on the *Lagrangian plug flow* (PF) ontology (Skorstad, 1992) has been designed to capture continuously changing thermodynamic fluid flow. This ontology is also a specialization of the pieces-of-stuff ontology and also describes macroscopic behavior of fluid. While the contained-stuff ontology is based on the *Eulerian* viewpoint, the others use *Lagrangian* viewpoint.

There are some situations which require microscopic analyses of fluids (e.g., diffusion and osmosis). Compared to macroscopic viewpoint, microscopic analyses can provide deeper understanding in terms of more fundamental concepts. Rajamoney and Koo (Rajamoney & Koo, 1990) and Amador and Weld (Amador & Weld, 1990) have developed theories of fluid at microscopic level.

These current ontologies have a common limitation: the lack of geometric information. They do not include information about the geometry of the surfaces which interact with fluids. Thus none of these are adequate for the reasoning tasks tackled in our work. On the other hand, our representations ignore the thermodynamic properties of fluids. Since we are interested in how the contact configuration of fluid changes and how fluid and its neighbors interact (e.g., force and motion transfer), we use the macroscopic level. Our bounded-stuff ontology can be viewed a generalization of the contained-stuff ontology with the information about the shapes of the surfaces fluids interact with.

9.2.6 Gelsey and McDermott

Gelsey and McDermott (1987, 1988, 1989) describe a method for spatial reasoning about mechanisms. They incorporate a kinematic analyzer into a behavior simulator. Given a structural

description of a mechanism using a constructive solid geometry representation (augmented with other quantitative information), their kinematic analyzer recognizes kinematic pairs and the relationships between their possible motions. The motion relationship is determined by analyzing the geometric constraint imposed by the structure of the mechanism and is expressed by mathematical functions between the positions of the pairs. Their approach can derive the possible motions of piston and crankshaft mechanisms, as well as differentials. The behavior simulator calculates forces acting on a state by analyzing collisions and friction.

Their linkage analyzer, which is part of the kinematic analyzer, finds the possible motions of linkages like our LINKAGE. They differ in several ways. First, while their linkage analyzer works in a purely quantitative way, LINKAGE analyzes linkages in a qualitative manner. Their linkage analyzer requires a precise three dimensional description based on predefined primitives and produces numerical functions describing the motion relations between parts. For example, it outputs two functions for a piston and crankshaft mechanism: one mapping the crankshaft position into the piston position, and another mapping the piston position into the crankshaft position. On the other hand, LINKAGE takes relative lengths of links and the connections between links in two dimensional space and produce a set of possible configurations and transitions between them. Instead of recognizing each pair from the initial description, each part is assumed to be a link, whose motion is constrained to rotate. Possible motions of each link is determined by the connections and its relative lengths with its neighbors. Since in many cases all parts of a linkage move in parallel planes, even though not all parts of the linkages lie in the same plane, 2D analysis by LINKAGE suffices for many linkages. Second, their analyzer is very limited. If the number of links is grater than three, it is difficult to represent the motion relationships between links simply by functions. This is because the relations vary in each configuration, as shown in 4-bar analysis in chapter 3. Our analysis can be generalized to more complicated linkages by propagating constraints across pairs of adjacent links.

9.3 Future Work

9.3.1 Integrating with Quantitative Analysis

Our approach to spatial reasoning is purely qualitative: reasoning processes start with purely qualitative information and describe physical phenomena in qualitative terms. While they have been successfully tested on our problems, they should be extended with quantitative representation to be applied for more sophisticated reasoning. As claimed by Forbus et al. (1991), we believe purely qualitative kinematics has limitations for capturing a broad range of phenomena. Thus to integrate qualitative kinematics with quantitative representation will be one of the important next steps toward a complete theory of qualitative kinematics. This will be necessary particularly for engineering problem solving, which we are interested in.

One way is to derive qualitative inferences from quantitative representations: this approach starts with a quantitative description of a system and then transforms it into a qualitative description before drawing inferences. The configuration space approach (Faltings, 1987; Nielsen, 1988; Joskowicz, 1988), which is commonly used in current qualitative kinematics, is an exam-

ple. While the configuration space method has been successfully tested on simple problems, this approach in general has limitations for practical problems: First, it is hard to have appropriately quantize precise information about shape and space into qualitative information. This quantization is harder than that of numbers since it requires a multidimensional analysis. We will also need different quantizations to support various detail of reasoning. Then a theory to choose the appropriate level of quantization for given reasoning task will be required. Second, intensive computation time currently required must be improved. Since every possible configuration of a system is calculated using precise information before qualitative reasoning, it currently suffers that limitation even for two dimensional systems.

Another possible way is to have two different levels for qualitative and quantitative reasoning with communication between the two. While the qualitative level is used to guide underlying quantitative analysis by using an abstract model, the quantitative level is used to provide precise analysis by using already well developed methods. In general, complex problem solving almost certainly requires multiple levels of understanding of a given system. Especially, engineering problem solving (e.g., prediction, design, and diagnosis) requires integrating overall understanding at various level of abstraction (Roddis and Martin, 1991). Consider an intelligent computer-aided design tool. Qualitative analysis can provide rough solutions at the early stages with little information. The qualitative analysis then guides the more precise analysis involving more detailed levels. The precise results are then interpreted at the qualitative level. Through this communication, design revision continues until getting a final solution. How to link the two levels–how to guide quantitative analysis and how to abstract the result of quantitative analysis– will be one of the important problem to be solved.

9.3.2 Modeling Fluids

Our theories of fluids have captured some aspects of fluids, but not everything. In fact, it seems impossible for a single ontology to cover every aspects of fluids. The choice of an ontology depends critically on the nature of reasoning task.

There are many unsolved problems involving fluids which must be solved in order to explain many important systems. We would like to extend our theories to cover phenomena such as mixing of air and gas in a carburetor, knocking during combustion inside of a cylinder, lubrication, and more. Tackling these phenomena will require further research in spatial issues and ontological shifts.

9.3.3 Adding Richer Topological Description

To describe space, as Hayes (1985) notes, we need a way to represent "piece of space" and a way to represent the relationship between the pieces of space. A system is described by representing each part and how it is related to others. Since a mechanical force is transmitted and modified by the geometry of the contact between objects, we describe a object in terms of surfaces. But this representation alone is sometimes insufficient to capture topological descriptions between objects. For instance, Figure 9.1a and b have the same surface contact relationships between `obj1` and `obj2` even though `obj1` is inside of `obj2` in a and `obj1` is outside of `obj2` in b.

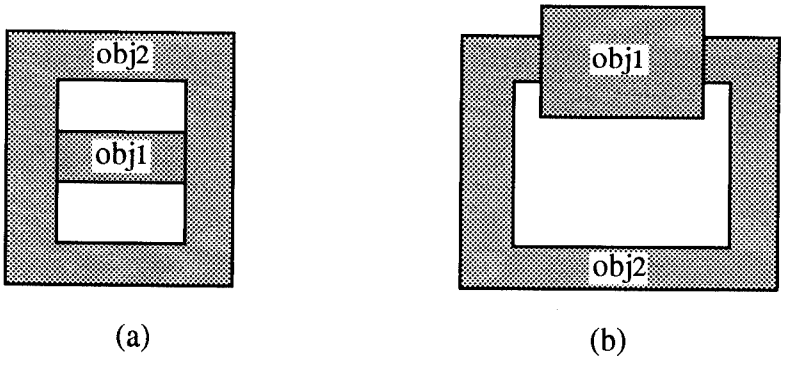


Figure 9.1: Two different configurations with same surface contacts.

We are looking for a way to combine our surface representation with the formalism of Randell and Cohn (1989). Their formalism seems to be adequate to describe topological relationships between objects and seems compatible enough to be integrated with our representation. As space is captured by richer representations, the spatial reasoning process may require more complicated and sophisticated techniques. One of the problems is how to map the extended structural description into one dimensional quantity spaces. Once this is done, limit analysis can capture spatial changes in the same manner of other quantities.

9.3.4 Increasing Scale

As the size of system being modeled gets larger, generating all possible states of the system would be impossible and undesirable; imagine the problem of constructing a total envisionment for a whole automobile in terms of its components. For building up large-scale models, avoiding combinatorial explosion is crucial. Several techniques have been suggested for this (Forbus, 1988). One idea is decomposition of a complex system into a set of subsystems. The whole system is described by combining the descriptions of the subsystems. Another alternative is to have layered models with various levels of detail for the system (Falkenhainer and Forbus, 1988). The appropriate model is selected depending on the nature of a given task, although this selection problem is very difficult.

While these approaches seem promising, more research is needed. We are especially interested in spatial reasoning aspects of modeling complex systems. Geometry is not involved in any of existing scale-up techniques, and many unsolved geometric problems are left. For a decomposition approach, how should a system be divided into kinematically independent pieces and how should they be combined? How should space decomposed into different level of approximation to build a model with hierarchy? What is the relationship between the levels, and how should the different layers be integrated to describe the system? To save computation time, what parts of the system can be precomputed independently of remaining parts?

9.3.5 Exploring Structural Abstraction

Almost all qualitative models, including ours, assume the input descriptions in terms of abstract language used in their representation. For instance, our scenarios for fluid flow are built based on containers, fluid paths, and valves since our domain model is constructed in terms of them. These objects are conceptual counterparts of real world objects and are captured by ourselves for analyses. This assumption, that the vocabulary of input description is consistent with the vocabulary of domain model, simplifies our effort to develop domain models and thus is used in most qualitative research.

Eventually, we want our reasoning system to interact with the real world. For this, advances in structural abstraction will be crucial; given a real world description through a visual system or CAD program, extracting the essentials relevant to our analysis will be required. The analysis will critically depend on this abstraction; consider an analysis with unnecessary detail or and an analysis with important missing detail. This mapping from a real world description to an abstract description appropriate for a domain model will also be crucial for layered models. Spatial issues will play a key role for structural abstraction and we would especially like to explore these issues.

References

[1] Franz Amador and Daniel Weld. Multi-level modeling of populations. In *Proceedings of the Fourth International Qualitative Physics Workshop*, 1990.

[2] Danny Bobrow, editor. *Qualitative Reasoning about Physical Systems.* MIT Press, Cambridge, MA, 1985.

[3] Eugene Charniak, Christopher K. Riesbeck, and Drew V. McDermott. *Artificial Intelligence Programming.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.

[4] John Collins and Kenneth Forbus. Reasoning about fluids via molecular collections. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, 1987.

[5] A. Cowie. *Kinematics and Design of Mechanisms.* International Textbook Co., Scranton, PA., 1961.

[6] Ernest Davis. A logical framework for solid object physics. Technical Report 245, New York University, Computer Science Department, 1986.

[7] Johan de Kleer. Qualitative and quantitative knowledge in classical mechanics. Technical Report 352, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA., 1975.

[8] Johan de Kleer. An assumption-based TMS. *Artificial Intelligence*, 28(2), March 1986.

[9] Johan de Kleer. Problem solving with the ATMS. *Artificial Intelligence*, 28(2), March 1986.

[10] Johan de Kleer and John Seely Brown. A qualitative physics based on confluences. *Artificial Intelligence*, 24:7–83, 1984.

[11] Thomas Dean and Greg Siegle. An approach to reasoning about continuous change for applications in planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 132–137, 1990.

[12] J. Dixon. Will mechanical engineers survive artificial intelligence? *Mechanical Engineering*, 108(2):8–10, 1986.

[13] Brian Falkenhainer. *Learning from Physical Analogies: A Study in Analogy and the Explanation Process.* PhD thesis, University of Illinois at Urbana-Champaign, Urbana, Illinois, December 1988.

[14] Brian Falkenhainer and Kenneth Forbus. Compositional modeling: Finding the right model for the job. *Artificial Intelligence*, 51, 1991.

[15] Brian Falkenhainer and Kenneth D. Forbus. Setting up large-scale qualitative models. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 301–306, St. Paul, MN, August 1988.

[16] Boi Faltings. A theory of qualitative kinematics in mechanisms. Technical Report UIUCDCS-R-86-1274, University of Illinois at Urbana-Champaign, May 1986.

[17] Boi Faltings. Qualitative place vocabularies for mechanisms in configuration space. Technical Report UIUCDCS-R-87-1360, University of Illinois at Urbana-Champaign, July 1987.

[18] Boi Faltings. Qualitative kinematics in mechanisms. *Artificial Intelligence*, 44, 1990.

[19] Colin Ferguson. *Internal Combustion Engine.* John Wiley & Sons, New York, 1976.

[20] Ken Forbus. A study of qualitative and geometric knowledge in reasoning about motion. Technical Report TR-615, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA., 1981.

[21] Ken Forbus, Paul Nielsen, and Boi Faltings. The inferential structure of qualitative kinematics. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, August 1987.

[22] Ken Forbus, Paul Nielsen, and Boi Faltings. Qualitative mechanics: A framework. *Artificial Intelligence*, 51, 1991.

[23] Kenneth Forbus. QPE: A study in assumption-based truth maintenances. *International Journal of AI in Engineering*, 1988.

[24] Kenneth D. Forbus. Spatial and qualitative aspects of reasoning about motion. In *Proceedings of the First National Conference on Artificial Intelligence*, August 1980.

[25] Kenneth D. Forbus. Qualitative reasoning about physical processes. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, August 1981.

[26] Kenneth D. Forbus. Qualitative process theory. *Artificial Intelligence*, 24, 1984.

[27] Kenneth D. Forbus. Intelligent computer-aided engineering. *AI Magazine*, 1988.

[28] Kenneth D. Forbus. Qualitative physics: Past, present, and future. In *Exploring Artificial Intelligence.* Morgan Kaufmann, 1988.

[29] Kenneth D. Forbus. Introducing actions into qualitative simulation. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 1989.

[30] Kenneth D. Forbus and Dedre Gentner. Learning physical domains: Towards a theoretical framework. In *Proceedings of the Second International Workshop on Machine Learning*, Monticello, IL, June 1983. A revised version appears in *Machine Learning: An Artificial Approach Vol. II*, R.S. Michalski, J.G. Carbonell, and T.M. Mitchell (Eds.), Morgan Kaufmann, 1986.

[31] Andrew Gelsey. Automated reasoning about machine geometry and kinematics. In *Third IEEE Conference on Artificial Intelligence Applications*, 1987.

[32] Andrew Gelsey. Automated physical modeling. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 1989.

[33] Andrew Gelsey and Drew McDermott. Spatial reasoning about mechanisms. Technical report, Yale University, 1988.

[34] M.R. Genesereth and N.J. Nilsson. *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1987.

[35] David Halliday and Robert Resnick. *Fundamentals of Physics.* John Wiley & Sons, New York, 1974.

[36] Patrick J. Hayes. The naive physics manifesto. In Donald Michie, editor, *Expert Systems in the Micro-Electronic Age.* Edinburgh University Press, 1979.

[37] Patrick J. Hayes. The second naive physics manifesto. In J. Hobbs and R. Moore, editors, *Formal Theories of the Commonsense World.* Ablex, 1985.

[38] John Hopcroft and Jeffrey Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley publishing company, Massachusetts, 1979.

[39] Franklin D. Jones, editor. *Ingenious Mechanisms for Designers and Inventors.* The Industrial Press, New York, 1930.

[40] Leo Joskowicz. A framework for the kinematic analysis of mechanical devices. Technical Report TR No. 313, New York University, 1987.

[41] Leo Joskowicz. Shape and function in mechanical devices. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, July 1987.

[42] Leo Joskowicz and Elisha Sacks. Unifying kinematics and dynamics for the automatic analysis of machines. 1989.

[43] Leo Joskowicz and Elisha Sacks. Computational kinematics. *Artificial Intelligence*, 51, 1991.

[44] Hyun-Kyung Kim. Qualitative reasoning about the geometry of fluid flow. In *Proceedings of the Twelfth Meeting of the Cognitive Science Society*, pages 117–124, 1990.

[45] Hyun-Kyung Kim. Augmenting qualitative simulation with global filtering. In *Proceedings of the Fourteenth Meeting of the Cognitive Science Society*, 1992.

[46] Hyun-Kyung Kim. Extending the contained-stuff ontology with geometry. In *Proceedings of the Sixth International Qualitative Physics Workshop*, 1992.

[47] Hyun-Kyung Kim. Qualitative kinematics of linkages. In *Recent Advances in Qualitative Physics.* MIT Press, 1992.

[48] Hyun-Kyung Kim. Qualitative reasoning about discontinuous changes by limit analysis. In *Pacific Rim International Conference on Artificial Intelligence*, 1992.

[49] Ben Kuipers. Qualitative simulation. *Artificial Intelligence*, 29:289–338, 1986.

[50] Benjamin Kuipers. Abstraction by time-scale in qualitative simulation. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 621–625, 1987.

[51] Benjamin Kuipers and Daniel Berleant. Using incomplete quantitative knowledge in qualitative reasoning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 324–329, 1988.

[52] S. Laughton. Explanation of mechanical systems through qualitative simulation. Technical Report AITR85-19, University of Texas at Austin, December 1985.

[53] Wood Lee and Benjamin Kuipers. Non-intersection of trajectories in qualitative phase space: A global constraint for qualitative simulation. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, 1988.

[54] Zheng-Yang Liu and Arthur Farley. Shifting ontological perspectives in reasoning about physical systems. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, 1990.

[55] T. Lozano-Perez and M. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22:560–570, 1979.

[56] Paul Nielsen. A qualitative approach to mechanical devices. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, 1988.

[57] Paul Nielsen. *A Qualitative Approach to Rigid Body Mechanics.* PhD thesis, University of Illinois at Urbana-Champaign, Urbana, Illinois, November 1988.

[58] Toyoaki Nishida and Shuji Doshita. Reasoning about discontinuous change. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 278–283, 1987.

[59] Yung-Choa Pan. *Qualitative Reasonings With Deep-Level Mechanism Models for Diagnoses of Dependent Failures.* PhD thesis, University of Illinois at Urbana-Champaign, 1984.

[60] Pearl Pu. Simulating both dynamic and kinematic behaviors of mechanisms. In *Proceedings of the Third International Qualitative Physics Workshop*, 1989.

[61] Shankar Rajamoney and Sang Hoe Koo. Qualitative reasoning with microscopic theories. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 401–406, 1990.

[62] D. Randell and A. Cohn. Exploring naive topology: Modelling the force pump. In *Proceedings of the Third International Qualitative Physics Workshop*, 1989.

[63] Manny Rayner. On the applicability of nonmonotonic logic to formal reasoning in continuous time. *Artificial Intelligence*, 49:345–360, 1991.

[64] Franz Reuleaux. *The Kinematics of Machinery, Outlines of a Theory of Machines.* Macmillan and Co., London, 1876. Translated and Edited by Alex B. W. Kennedy, C.E.

[65] W. Kim Roddis and Jeffrey Martin. Qualitative reasoning about steel birdge fatigue and fracture. In *Proceedings of the Fifth International Qualitative Physics Workshop*, 1991.

[66] Elisha Sacks and Leo Joskowicz. Model-based kinematic simulation. In *Proceedings of the ASME Winter Annual Meeting*, 1992.

[67] Elisha Sacks and Leo Joskowicz. Automated modeling and kinematic simulation of mechanisms. *CAD Journal*, 1993.

[68] Erik Sandewall. Combining logic and differential equations for describing real-world systems. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, 1989.

[69] Erik Sandewall. Filter preferential entailment for the logic of action in almost continuous worlds. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 1989.

[70] Yoav Shoham. Naive kinematics: One aspect of shape. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 436–442, 1985.

[71] Gordon Skorstatd. Towards a qualitative lagrangina theory of fluid flow. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, 1992.

[72] Craig Stanfill. *Form and Function: The Representation and Machine.* PhD thesis, University of Maryland, 1983.

[73] Peter Struss. Global filters fro qualitative behaviors. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 275–279, 1988.

[74] Milton Van Dyke. *An album of Fluid Motion.* The Parabolic Press, California, 1982.

[75] Gordon Van Wylen and Richard Sonntag. *Fundamentals of Classical Thermodynamics.* John Wiley & Sons, New York, 1978.

[76] Edward Warner. *Aerodynamics.* McGraw-Hill Book Co., New York, 1927.

[77] Dan Weld. The use of aggregation in causal simulation. *Artificial Intelligence*, 1986.

[78] Dan Weld. *Theories of Comparative Analysis.* PhD thesis, Massachusetts Institute of Technology, May 1988.

[79] Frank White. *Fluid Mechanics.* McGraw-Hill Book Co., New York, 1979.

[80] Brian Williams. The use of continuity in a qualitative physics. In *Proceedings of the Fourth National Conference on Artificial Intelligence*, August 1984.

[81] Brian C. Williams. Doing time: Putting qualitative reasoning on firmer ground. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 105–112, Philadelphia, PA, August 1986.

[82] Patrick Henry Winston. *Artificial Intelligence, Second Edition.* Addison Wesley, Reading, MA, 1984.

Appendix A

Total Envisionments of Linkages

Each qualitative state consists of a kinematic state representing the orientation of each link, and a dynamic state representing the motions. The orientation of links are represented by qualitative vectors and inequalities of inclination. The qualitative vector corresponds to the signs of the numerical values in a right-handed Cartesian coordinate system. The motion is also represented by the signs of numerical values in a right-handed Cartesian coordinate system. A counter-clockwise rotation will be + and clockwise rotation will be −. The following table shows the qualitative states and their transitions. Diagrams of the kinematic states are included to the right.

A.1 Slider-Crank Envisionment

K.S. No	Kinematic State Senses P2/P1	P3/P2	Incl L1/L2	Dynamic State L1/L2/L3	Next K.S. No	Next Dynamic state
1	+0	−+		+0+	2	000
				+0+	2	+ − +
				−0−	8	− − −
				−0−	8	000
				000	1	−0−
				000	1	+0+
2	++	−+		+ − +	3	000
				+ − +	3	+ − 0
				+ − +	2	000
				− + −	1	−0−
				− + −	1	000
				− + −	2	000
				000	2	+ − +
				000	2	− + −
3	0+	0+		+ − 0	4	000
				+ − 0	4	+ − −
				− + 0	2	− + −
				− + 0	2	000
				000	3	− + 0
				000	3	+ − 0

K.S. No	Kinematic State Senses P2/P1	P3/P2	Incl L1/L2	Dynamic State L1/L2/L3	Next K.S. No	Dynamic state
4	−+	++		+−−	5	000
				+−−	5	+0−
				+−−	4	000
				−++	4	000
				−++	3	−+0
				−++	3	000
				000	4	−++
				000	4	+−−
5	−0	++		+0−	6	000
				+0−	6	++−
				−0+	4	−++
				−0+	4	000
				000	5	−0+
				000	5	+0−
6	−−	++	<	++−	7	000
				++−	7	++0
				++−	6	000
				−−+	6	000
				−−+	5	−0+
				−−+	5	000
				000	6	−−+
				000	6	++−
7	0−	0+		++0	8	000
				++0	8	+++
				−−0	6	−−+
				−−0	6	000
				000	7	−−0
				000	7	++0
8	+−	−+	>	+++	1	000
				+++	1	+0+
				+++	8	000
				−−−	8	000
				−−−	7	−−0
				−−−	7	000
				−−−	8	−−−
				000	8	+++

A.2 Drag-Link Envisionment

K.S. No	Kinematic State: Senses P2/P1	P3/P2	P4/P3	Incl L1/L2	L2/L3	L3/L1	Dynamic State L1/L2/L3	Next K.S. No	Next Dynamic State
1	+0	−+	−−				+ + +	7	+ + +
							− − −	80	− − −
2	+0	−+	0−				+ + +	9	+ + +
							− − −	80	− − −
3	+0	−+	+−		>		+ + +	9	+ + +
							− − −	82	− − −
4	+0	−−	++		<		+ + +	13	+ + +
							− − −	86	− − −
5	+0	−−	0+				+ + +	15	+ + +
							− − −	86	− − −
6	+0	−−	−+				+ + +	15	+ + +
							− − −	88	− − −
7	++	−+	−−			>	+ + +	8	+ + +
							+ + +	10	+ + +
							− − −	1	− − −
							− − −	9	− − −
8	++	−+	0−				+ + +	9	+ + +
							− − −	7	− − −
9	++	−+	+−		>		+ + +	12	+ + +
							− − −	3	− − −
							− − −	2	− − −
							− − −	8	− − −
10	++	−0	−−				+ + +	17	+ + +
							− − −	7	− − −
11	++	−0	0−				+ + +	20	+ + +
							− − −	7	− − −
12	++	−0	+−				+ + +	20	+ + +
							− − −	9	− − −
13	++	−−	++	<	<	>	+ + +	14	+ + +
							− − −	4	− − −
14	++	−−	0+	<			+ + +	15	+ + +
							− − −	13	− − −
15	++	−−	−+	<			+ + +	16	+ + +
							+ + +	22	+ + +
							+ + +	21	+ + +
							− − −	6	− − −
							− − −	5	− − −
							− − −	14	− − −
16	++	−−	−0	<			+ + +	18	+ + +
							− − −	15	− − −
17	++	−−	−−	>	<	>	+ + +	19	+ + +
							− − −	10	− − −
18	++	−−	−−	<	>	<	+ + +	23	+ + +
							− − −	16	− − −
19	++	−−	0−	>			+ + +	20	+ + +
							− − −	17	− − −
20	++	−−	+−	>			+ + +	27	+ + +
							− − −	12	− − −
							− − −	11	− − −
							− − −	19	− − −

K.S. No	Kinematic State: Senses P2/P1	P3/P2	P4/P3	Incl L1/L2	L2/L3	L3/L1	Dynamic State L1/L2/L3	Next K.S. No	Next Dynamic State
21	++	0−	−+				+++	24	+++
							−−−	15	−−−
22	++	0−	−0				+++	26	+++
							−−−	15	−−−
23	++	0−	−−			<	+++	26	+++
							−−−	18	−−−
24	++	+−	−+		<		+++	25	+++
							−−−	21	−−−
25	++	+−	−0				+++	26	+++
							−−−	24	−−−
26	++	+−	−−			<	+++	28	+++
							−−−	23	−−−
							−−−	22	−−−
							−−−	25	−−−
27	0+	−−	+−				+++	37	+++
							−−−	20	−−−
28	0+	+−	−−				+++	43	+++
							−−−	26	−−−
29	−+	+0	−−				+++	32	+++
							−−−	43	−−−
30	−+	+0	0−				+++	34	+++
							−−−	43	−−−
31	−+	+0	+−			<	+++	34	+++
							−−−	46	−−−
32	−+	++	−−		<		+++	33	+++
							−−−	29	−−−
33	−+	++	0−				+++	47	+++
							−−−	32	−−−
34	−+	++	+−			<	+++	47	+++
							−−−	33	−−−
							−−−	31	−−−
							−−−	30	−−−
35	−+	−−	+0				+++	36	+++
							−−−	37	−−−
36	−+	−−	++		>		+++	39	+++
							−−−	35	−−−
37	−+	−−	+−			<	+++	35	+++
							+++	38	+++
							+++	40	+++
							−−−	27	−−−
38	−+	0−	+0				+++	42	+++
							−−−	37	−−−
39	−+	0−	++				+++	42	+++
							−−−	36	−−−
40	−+	0−	+−				+++	45	+++
							−−−	37	−−−
41	−+	+−	+0	>			+++	42	+++
							−−−	45	−−−
42	−+	+−	++	<			+++	48	+++
							−−−	38	−−−
							−−−	39	−−−
							−−−	41	−−−

K.S. No	Kinematic State: Senses P2/P1	P3/P2	P4/P3	Incl L1/L2	L2/L3	L3/L1	Dynamic State L1/L2/L3	Next K.S. No	Next Dynamic State
43	−+	+−	−−	<			+++	44	+++
							+++	30	+++
							+++	29	+++
							−−−	28	−−−
44	−+	+−	0−	<			+++	46	+++
							−−−	43	−−−
45	−+	+−	+−	>	<	>	+++	41	+++
							−−−	40	−−−
46	−+	+−	+−	<	>	<	+++	31	+++
							−−−	44	−−−
47	−0	++	+−				+++	57	+++
							−−−	34	−−−
48	−0	++	++				+++	64	+++
							−−−	42	−−−
49	−−	+0	++			>	+++	55	+++
							+++	53	+++
							−−−	64	−−−
							−−−	48	−−−
50	−−	+0	0+				+++	56	+++
							−−−	64	−−−
51	−−	+0	−+				+++	56	+++
							−−−	66	−−−
52	−−	++	+0	<			+++	54	+++
							−−−	57	−−−
53	−−	++	++	>	<	<	+++	55	+++
							−−−	49	−−−
54	−−	++	++	<	>	<	+++	59	+++
							−−−	52	−−−
55	−−	++	0+	>			+++	56	+++
							−−−	53	−−−
56	−−	++	−+	>			+++	67	+++
							−−−	55	−−−
							−−−	51	−−−
							−−−	50	−−−
57	−−	++	+−	<			+++	52	+++
							+++	58	+++
							+++	60	+++
							−−−	47	−−−
58	−−	0+	+0				+++	62	+++
							−−−	57	−−−
59	−−	0+	++			<	+++	62	+++
							−−−	54	−−−
60	−−	0+	+−				+++	63	+++
							−−−	57	−−−
61	−−	−+	+0				+++	62	+++
							−−−	63	−−−
62	−−	−+	++			<	+++	68	+++
							−−−	58	−−−
							−−−	59	−−−
							−−−	61	−−−
63	−−	−+	+−		<		+++	61	+++
							−−−	60	−−−

K.S. No	Kinematic State: Senses P2/P1	P3/P2	P4/P3	Incl L1/L2	L2/L3	L3/L1	Dynamic State L1/L2/L3	Next K.S. No	Next Dynamic State
64	−−	+−	++		>		+++	65	+++
							+++	50	+++
							+++	49	+++
							−−−	48	−−−
65	−−	+−	0+				+++	66	+++
							−−−	64	−−−
66	−−	+−	−+		>		+++	51	+++
							−−−	65	−−−
67	0−	++	−+				+++	69	+++
							−−−	56	−−−
68	0−	−+	++				+++	75	+++
							−−−	62	−−−
69	+−	++	−+			>	+++	70	+++
							+++	73	+++
							+++	72	+++
							−−−	67	−−−
70	+−	++	−0				+++	71	+++
							−−−	69	−−−
71	+−	++	−−		>		+++	74	+++
							−−−	70	−−−
72	+−	0+	−+			>	+++	77	+++
							−−−	69	−−−
73	+−	0+	−0				+++	80	+++
							−−−	69	−−−
74	+−	0+	−−				+++	80	+++
							−−−	71	−−−
75	+−	−+	++	<			+++	76	+++
							+++	84	+++
							+++	83	+++
							−−−	68	−−−
76	+−	−+	0+	<			+++	78	+++
							−−−	75	−−−
77	+−	−+	−+	>	<	>	+++	79	+++
							−−−	72	−−−
78	+−	−+	−+	>	>	<	+++	85	+++
							−−−	76	−−−
79	+−	−+	−0	>			+++	80	+++
							−−−	77	−−−

A.3 Crank-Rocker Envisionment

K.S. No	Kinematic State: Senses P2/P1	P3/P2	P4/P3	Incl L1/L2	L2/L3	L3/L1	Dynamic State L1/L2/L3	Next K.S. No	Next Dynamic State
1	+0	++	−−		<		+−−	11	+−−
							−++	85	−++
2	+0	++	0−				+−−	11	+−−
							−++	87	−++
							−++	88	−0+
3	+0	++	+−				+−−	17	+−−
							−++	87	−++
4	+0	+−	++				+−−	28	+−−
							−++	98	−++
5	+0	+−	0+				+−−	28	+−−
							−++	104	−++
6	+0	+−	−+		>		+−−	30	+−−
							−++	104	−++
7	++	++	−−	>	<	<	+++	32	+0+
							+++	31	+++
							+++	8	+0+
							−−−	8	−0−
8	++	++	−−	>	<	=	+0+	7	+++
							−0−	9	−+−
9	++	++	−−	>	<	>	+−+	32	+0+
							+−+	12	+−+
							+−+	8	+0+
							−+−	10	−+0
							−+−	8	−0−
10	++	++	−−	=	<	>	+−0	9	+−+
							−+0	11	−++
11	++	++	−−	<	<	>	+−−	10	+−0
							−++	2	−++
							−++	1	−++
							−++	14	−++
12	++	++	0−	>			+−+	15	+−+
							−+−	9	−+−
13	++	++	0−	=			+−0	32	+0+
							+−0	12	+−+
							−+0	14	−++
14	++	++	0−	<			+−−	11	+−−
							−++	17	−++
15	++	++	+−	>			+−+	33	+−+
							−+−	16	−+0
							−+−	12	−+−
							−+−	13	−+0
16	++	++	+−	=			+−0	15	+−+
							−+0	17	−++
17	++	++	+−	<			+−−	16	+−0
							+−−	14	+−−
							+−−	13	+−0
							−++	3	−++
18	++	−−	++	>	>	<	+++	34	+++
							−−−	19	−−0

K.S. No	Kinematic State Senses P2/P1	P3/P2	P4/P3	L1/L2	Incl L2/L3	L3/L1	Dynamic State L1/L2/L3	Next K.S. No	Next Dynamic State
19	++	−−	++	=	>	<	++0	18	+++
							−−0	20	−−+
20	++	−−	++	<	>	<	++−	35	++0
							++−	19	++0
							++−	21	+0−
							++−	23	++−
							++−	24	+0−
							−−+	21	−0+
21	++	−−	++	<	>	=	+0−	20	++−
							−0+	22	−++
22	++	−−	++	<	>	>	+−−	21	+0−
							−++	25	−++
23	++	0−	++			<	++−	26	++−
							−−+	20	−−+
24	++	0−	++			=	+0−	23	++−
							+0−	25	+−−
25	++	0−	++			>	+−−	22	+−−
							−++	28	−++
26	++	+−	++			<	++−	36	++−
							−−+	24	−0+
							−−+	23	−−+
							−−+	27	−0+
27	++	+−	++			=	+0−	26	++−
							−0+	28	−++
28	++	+−	++			>	+−−	25	+−−
							−++	5	−++
							−++	4	−++
							−++	29	−++
							−++	27	−0+
29	++	+−	0+				+−−	28	+0−
							−++	30	−++
30	++	+−	−+		>		+−−	37	+0−
							+−−	29	+−−
							−++	6	−++
31	0+	++	−−		<		+++	38	+++
							−−−	7	−−−
32	0+	++	0−				+0+	40	+++
							−0−	9	−+−
33	0+	++	+−				+−+	42	+−+
							−+−	15	−+−
34	0+	−−	++		>		+++	43	+++
							−−−	18	−−−
35	0+	0−	++				++0	45	+++
							−−0	20	−−+
36	0+	+−	++				++−	47	++−
							−−+	26	−−+
37	0+	+−	0+				+0−	47	++−
							−0+	30	−−+
38	−+	++	−−		<		+++	56	+++
							+++	55	+++
							+++	39	+++
							−−−	31	−−−

K.S. No	Kinematic State: Senses P2/P1	P3/P2	P4/P3	Incl L1/L2	L2/L3	L3/L1	Dynamic State L1/L2/L3	Next K.S. No	Next Dynamic State
39	−+	++	0−				+++	40	+++
							−−−	38	−−−
40	−+	++	+−			<	+++	57	+++
							+++	41	+++
							−−−	41	−0−
							−−−	39	−−−
							−−−	32	−0−
41	−+	++	+−			=	+0+	40	+++
							+0+	42	+−+
							−0−	42	−+−
42	−+	++	+−			>	+−+	41	+0+
							−+−	41	−0−
							−+−	33	−+−
							−+−	32	−0−
43	−+	−−	++		>		+++	44	+++
							−−−	34	−−−
44	−+	0−	++				+++	45	+++
							−−−	43	−−−
45	−+	+−	++	>			+++	59	+++
							+++	58	+++
							+++	48	+++
							+++	45	+++
							−−−	44	−−−
							−−−	46	−−−
							−−−	35	−−−
46	−+	+−	++	=			++0	45	+++
							−−0	47	−−+
47	−+	+−	++	<			++−	46	++0
							−−+	37	−0+
48	−+	+−	0+	>			+++	50	+++
							−−−	45	−−−
49	−+	+−	0+	=			++0	48	+++
							−−0	37	−0+
50	−+	+−	−+	>	>	<	+++	60	+++
							−−−	51	−−0
							−−−	48	−−−
51	−+	+−	−+	=	>	<	++0	50	++−
							−−0	52	−−+
52	−+	+−	−+	<	>	<	++−	51	++0
							++−	49	++0
							−−+	53	−0+
53	−+	+−	−+	<	>	=	+0−	52	++−
							−0+	54	−++
54	−+	+−	−+	<	>	>	+−−	53	+0−
55	−0	++	−−		<		+++	65	+++
							−−−	38	−−−
56	−0	++	0−				+++	70	+++
							−−−	38	−−−
57	−0	++	+−				+++	70	+++
							−−−	40	−−−

K.S. No	Kinematic State Senses P2/P1	P3/P2	P4/P3	L1/L2	Incl L2/L3	L3/L1	Dynamic State L1/L2/L3	Next K.S. No	Next Dynamic State
58	−0	+−	++				+ + +	75	+ + +
							− − −	45	− − −
59	−0	+−	0+				+ + +	77	+ + +
							− − −	45	− − −
60	−0	+−	−+		>		+ + +	77	+ + +
							− − −	50	− − −
61	−−	++	−−	>	<	<	− + +	62	−0+
62	−−	++	−−	>	<	=	+0−	61	+ − −
							−0+	63	− − +
63	−−	++	−−	>	<	>	+ + −	62	+0−
							− − +	66	− − 0
							− − +	64	− − 0
64	−−	++	−−	=	<	>	+ + 0	63	+ + −
							− − 0	65	− − −
65	−−	++	−−	<	<	>	+ + +	66	+ + 0
							+ + +	67	+ + +
							+ + +	64	+ + 0
							− − −	55	− − −
66	−−	++	0−	=			+ + 0	78	+0−
							− − 0	67	− − −
67	−−	++	0−	<			+ + +	70	+ + +
							− − −	65	− − −
68	−−	++	+−	>			+ + −	79	+ + −
							+ + −	78	+0−
							+ + −	80	+ + 0
							+ + −	69	+ + 0
							+ + −	66	+ + 0
							− − +	69	− − 0
69	−−	++	+−	=			+ + 0	68	+ + −
							− − 0	70	− − −
70	−−	++	+−	<			+ + +	80	+ + 0
							+ + +	69	+ + 0
							+ + +	71	+ + +
							− − −	67	− − −
							− − −	57	− − −
							− − −	56	− − −
71	−−	0+	+−				+ + +	72	+ + +
							− − −	70	− − −
72	−−	−+	+−		<		+ + +	81	+ + +
							− − −	71	− − −
73	−−	−+	++			<	+ − +	83	+0+
							+ − +	82	+ − +
							+ − +	74	+0+
							− + −	74	−0−
74	−−	−+	++			=	+0+	73	+ − +
							−0−	75	− − −
75	−−	−+	++			>	+ + +	83	+0+
							− − −	74	−0−
							− − −	58	− − −
76	−−	−+	0+				+ + +	77	+ + +
							− − −	75	− − −

K.S. No	Kinematic State: Senses P2/P1	P3/P2	P4/P3	Incl L1/L2	L2/L3	L3/L1	Dynamic State L1/L2/L3	Next K.S. No	Next Dynamic State
77	−−	−+	−+		>		+++	84	+++
							−−−	76	−−−
							−−−	60	−−−
							−−−	59	−−−
78	0−	++	0−				+0−	85	+−−
							−0+	68	−−+
79	0−	++	+−				++−	89	++−
							−−+	68	−−+
80	0−	0+	+−				++0	95	++−
							−−0	70	−−−
81	0−	−+	+−		<		+++	95	++0
							+++	97	+++
							−−−	72	−−−
82	0−	−+	++				+−+	100	+−+
							−+−	73	−+−
83	0−	−+	0+				+0+	106	+−+
							−0−	75	−−−
84	0−	−+	−+		>		+++	108	+++
							−−−	77	−−−
85	+−	++	−−		<		+−−	1	+−−
							−++	86	−++
							−++	78	−0+
86	+−	++	0−				+−−	1	+−−
							+−−	85	+−−
							−++	90	−++
87	+−	++	+−			<	+−−	3	+−−
							+−−	2	+−−
							+−−	88	+0−
							+−−	86	+−−
							−++	88	−0+
							−++	90	−++
							−++	91	−0+
88	+−	++	+−			=	+0−	87	+−−
							−0+	89	−−+
89	+−	++	+−			>	++−	88	+0−
							++−	92	++−
							++−	91	+0−
							−−+	88	−0+
							−−+	79	−−+
90	+−	0+	+−			<	+−−	87	+−−
							−++	93	−++
91	+−	0+	+−			=	+0−	90	+−−
							−0+	92	−−+
							−0+	80	−−0
92	+−	0+	+−			>	++−	95	++−
							−−+	89	−−+
93	+−	−+	+−	>	<	<	+−−	90	+−−
							−++	94	−0+
94	+−	−+	+−	>	<	=	+0−	93	+−−
							−0+	95	−−+

K.S. No	Kinematic State: Senses P2/P1	P3/P2	P4/P3	Incl L1/L2	L2/L3	L3/L1	Dynamic State L1/L2/L3	Next K.S. No	Next Dynamic State
95	+ −	− +	+ −	>	<	>	+ + −	94	+ 0 −
							− − +	91	− 0 +
							− − +	92	− − +
							− − +	94	− 0 +
							− − +	96	− − 0
							− − +	80	− − 0
96	+ −	− +	+ −	=	<	>	+ + 0	95	+ + −
							− − 0	97	− − −
97	+ −	− +	+ −	<	<	>	+ + +	96	+ + 0
							− − −	80	− − 0
							− − −	81	− − −
98	+ −	+ −	+ +	>			+ − −	4	+ − −
							− + +	102	− + 0
							− + +	101	− + +
							− + +	99	− + 0
99	+ −	+ −	+ +	=			+ − 0	98	+ − −
							− + 0	100	− + −
100	+ −	+ −	+ +	<			+ − +	102	+ − 0
							+ − +	103	+ − +
							+ − +	99	+ − 0
							− + −	82	− + −
101	+ −	+ −	0 +	>			+ − −	98	+ − −
							− + +	104	− + +
102	+ −	+ −	0 +	=			+ − 0	101	+ − −
							− + 0	103	− + −
103	+ −	+ −	0 +	<			+ − +	106	+ − +
							− + −	100	− + −
104	+ −	+ −	− +	>	>	<	+ − −	6	+ − −
							+ − −	5	+ − −
							+ − −	101	+ − −
							− + +	105	− + 0
105	+ −	+ −	− +	=	>	<	+ − 0	104	+ − −
							− + 0	106	− + −
106	+ −	+ −	− +	<	>	<	+ − +	107	+ 0 +
							− + −	105	− + 0
							− + −	107	− 0 −
							− + −	103	− + −
							− + −	83	− 0 −
107	+ −	+ −	− +	<	>	=	+ 0 +	106	+ − +
							− + 0	108	− − −
108	+ −	+ −	− +	<	>	>	+ + +	107	+ 0 +
							− − −	84	− − −
							− − −	83	− 0 −

A.4 Double-Rocker Envisionment

K.S. No	Kinematic State: Senses P2/P1	P3/P2	P4/P3	L1/L2	Incl L2/L3	L3/L1	Dynamic State L1/L2/L3	Next K.S. No	Next Dynamic State
1	+0	+0	+−				+−+	25	+−+
							−+−	9	−+−
2	+0	++	−−	>	<	>	+−+	6	+−+
							−+−	3	−+0
3	+0	++	−−	=	<	>	+−0	2	+−+
							−+0	4	−++
4	+0	++	−−	<	<	>	+−−	3	+−0
							−++	8	−++
5	+0	++	−−	<	>	>	+++	12	+++
6	+0	++	0−	>			+−+	9	+−+
							−+−	2	−+−
7	+0	++	0−	=			+−0	6	+−+
							−+0	8	−++
8	+0	++	0−	<			+−−	4	+−−
							−++	11	−++
9	+0	++	+−	>			+−+	27	+−+
							+−+	26	+−+
							+−+	1	+−+
							−+−	10	−+0
							−+−	6	−+−
							−+−	7	−+0
10	+0	++	+−	=			+−0	9	+−+
							−+0	11	−++
11	+0	++	+−	<			+−−	10	+−0
							+−−	8	+−−
							+−−	7	+−0
							−++	13	−++
12	+0	0+	−−			>	+++	14	+++
							−−−	5	−−−
13	+0	0+	+−				+−−	11	+−−
							−++	15	−++
14	++	−+	−−			>	−−−	12	−−−
15	++	−+	+−		<		+−−	28	+−0
							+−−	13	+−−
16	++	−+	+−		=		0−−	15	+−−
							0++	17	+++
17	++	−+	+−		>		+++	30	+++
							+++	31	+++
							+++	18	+++
							−−−	16	0−−
18	++	−0	+−				+++	19	+++
							−−−	17	−−−
19	++	−−	+−	>			+++	32	+++
							+++	33	++0
							+++	20	++0
							−−−	18	−−−
20	++	−−	+−	=			++0	21	++−
							−−0	19	−−−
21	++	−−	+−	<			++−	22	++−
							−−+	20	−−0

K.S. No	Kinematic State: Senses P2/P1	P3/P2	P4/P3	Incl L1/L2	L2/L3	L3/L1	Dynamic State L1/L2/L3	Next K.S. No	Next Dynamic State
22	++	0−	+−				++−	23	++−
							−−+	21	−−+
23	++	+−	+−		<		++−	35	0+−
							++−	34	++−
							++−	24	0+−
							−−+	22	−−+
24	++	+−	+−		=		0−+	23	−−+
							0+−	25	−+−
25	++	+−	+−		>		+−+	35	0−+
							+−+	36	+−+
							+−+	24	0−+
							−+−	1	−+−
26	0+	+0	+−				+−+	51	+−+
							−+−	9	−+−
27	0+	++	+−				+−+	38	+−+
							−+−	9	−+−
28	0+	0+	+−				+−0	38	+−+
							−+0	15	−++
29	0+	−+	+−		=		0++	30	+++
30	0+	−+	+−		>		+++	43	+++
							−−−	17	−−−
31	0+	−0	+−				+++	45	+++
							−−−	17	−−−
32	0+	−−	+−				+++	45	+++
							−−−	19	−−−
33	0+	0−	+−				++0	49	++−
							−−0	19	−−−
34	0+	+−	+−		<		++−	49	++−
							−−+	23	−−+
35	0+	+−	+−		=		0−+	34	−−+
							0+−	36	−+−
36	0+	+−	+−		>		+−+	51	+−+
							−+−	25	−+−
37	−+	+0	+−			>	+−+	51	+−+
							−+−	38	−+−
38	−+	++	+−			>	+−+	37	+−+
							−+−	27	−+−
							−+−	28	−+0
39	−+	−+	+−	>	<	>	−+−	40	−+0
40	−+	−+	+−	=	<	>	+−0	39	+−+
							−+0	41	−++
41	−+	−+	+−	<	<	>	+−−	40	+−0
							−++	42	0++
							−++	29	0++
42	−+	−+	+−	<	=	>	0−−	43	−−−
							0−−	41	+−−
							0++	43	+++
43	−+	−+	+−	<	>	>	+++	44	+++
							−−−	42	0−−
							−−−	30	−−−
							−−−	29	0−−
44	−+	−0	+−			>	+++	45	+++
							−−−	43	−−−

K.S. No	Kinematic State: Senses P2/P1	P3/P2	P4/P3	Incl L1/L2	L2/L3	L3/L1	Dynamic State L1/L2/L3	Next K.S. No	Next Dynamic State
45	−+	−−	+−			>	+++	46	+++
							−−−	44	−−−
							−−−	31	−−−
							−−−	32	−−−
46	−+	0−	+−			>	+++	47	+++
							−−−	45	−−−
47	−+	+−	+−	>	<	>	+++	48	++0
							−−−	46	−−−
48	−+	+−	+−	=	<	>	++0	49	++−
							−−0	46	−−−
							−−0	47	−−−
49	−+	+−	+−	<	<	>	++−	50	0+−
							−−+	48	−−0
							−−+	33	−−0
							−−+	34	−−+
50	−+	+−	+−	<	=	>	0−+	49	−−+
							0+−	51	−+−
51	−+	+−	+−	<	>	>	+−+	50	0−+
							−+−	37	−+−
							−+−	36	−+−
							−+−	26	−+−
52	−−	+0	++			<	+−+	66	+−+
							−+−	53	−+−
53	−−	++	++	>	<	<	+−+	68	+−+
							+−+	67	+−+
							+−+	52	+−+
							−+−	54	−+−
54	−−	++	++	>	=	<	0−+	53	+−+
							0+−	55	++−
55	−−	++	++	>	>	<	++−	70	++−
							++−	71	++0
							++−	56	++0
							−−+	54	0−+
56	−−	++	++	=	>	<	++0	70	++−
							++0	71	++0
							++0	57	+++
							−−0	55	−−+
57	−−	++	++	<	>	<	+++	58	+++
							−−−	56	−−0
58	−−	0+	++			<	+++	59	+++
							−−−	57	−−−
59	−−	−+	++			<	+++	72	+++
							+++	73	+++
							+++	60	+++
							−−−	58	−−−
60	−−	−0	++			<	+++	61	+++
							−−−	59	−−−
61	−−	−−	++	>	<	<	+++	75	0++
							+++	74	+++
							+++	62	0++
							−−−	60	−−−
62	−−	−−	++	>	=	<	0−−	61	−−−
							0++	63	−++

K.S. No	Kinematic State: Senses P2/P1	P3/P2	P4/P3	Incl L1/L2	L2/L3	L3/L1	Dynamic State L1/L2/L3	Next K.S. No	Next Dynamic State
63	−−	−−	++	>	>	<	+−−	75	0−−
							+−−	62	0−−
							−++	64	−+0
64	−−	−−	++	=	>	<	+−0	63	+−−
							−+0	65	−+−
65	−−	−−	++	<	>	<	+−+	64	+−0
66	−−	+−	++			<	+−+	76	+−0
							+−+	77	+−+
							−+−	52	−+−
67	0−	+0	++				+−+	95	+−+
							−+−	53	−+−
68	0−	++	++		<		+−+	79	+−+
							−+−	53	−+−
69	0−	++	++		=		0−+	68	+−+
							0+−	70	++−
70	0−	++	++		>		++−	81	++−
							−−+	55	−−+
71	0−	0+	++				++0	83	++−
							++0	85	+++
							−−0	55	−−+
72	0−	−+	++				+++	85	+++
							−−−	59	−−−
73	0−	−0	++				+++	87	+++
							−−−	59	−−−
74	0−	−−	++		<		+++	87	+++
							−−−	61	−−−
75	0−	−−	++		=		0−−	74	−−−
							0++	74	+++
76	0−	0−	++				+−0	89	+−−
							−+0	66	−+−
77	0−	+−	++				+−+	95	+−+
							−+−	66	−+−
78	+−	+0	++				+−+	95	+−+
							−+−	79	−+−
79	+−	++	++		<		+−+	78	+−+
							−+−	80	0+−
							−+−	68	−+−
							−+−	69	0+−
80	+−	++	++		=		0−+	79	+−+
							0+−	81	++−
81	+−	++	++		>		++−	82	++−
							−−+	80	0−+
							−−+	70	−−+
							−−+	69	0−+
82	+−	0+	++				++−	83	++−
							−−+	81	−−+
83	+−	−+	++	>			++−	84	++0
							−−+	82	−−+
84	+−	−+	++	=			++0	85	+++
							−−0	83	−−+

K.S. No	Kinematic State: Senses P2/P1	P3/P2	P4/P3	Incl L1/L2	L2/L3	L3/L1	Dynamic State L1/L2/L3	Next K.S. No	Next Dynamic State
85	+−	−+	++	<			+++	86	+++
							−−−	84	−−0
							−−−	71	−−0
							−−−	72	−−−
86	+−	−0	++				+++	87	+++
							−−−	85	−−−
87	+−	−−	++		<		+++	88	0++
							−−−	86	−−−
							−−−	73	−−−
							−−−	74	−−−
88	+−	−−	++		=		0−−	87	−−−
							0++	89	−++
89	+−	−−	++		>		+−−	88	0−−
							−++	91	−++
							−++	76	−+0
90	+−	−−	−+			<	+++	92	+++
91	+−	0−	++				+−−	89	+−−
							−++	93	−++
92	+−	0−	−+			<	+++	99	+++
							−−−	90	−−−
93	+−	+−	++	>			+−−	91	+−−
							−++	97	−+0
							−++	96	−++
							−++	94	−+0
94	+−	+−	++	=			+−0	93	+−−
							−+0	95	−+−
95	+−	+−	++	<			+−+	97	+−0
							+−+	98	+−+
							+−+	94	+−0
							−+−	78	−+−
							−+−	77	−+−
							−+−	67	−+−
96	+−	+−	0+	>			+−−	91	+−−
							+−−	93	+−−
							−++	100	−++
97	+−	+−	0+	=			+−0	96	+−−
							−+0	98	−+−
98	+−	+−	0+	<			+−+	102	+−+
							−+−	95	−+−
99	+−	+−	−+	>	<	<	−−−	92	−−−
100	+−	+−	−+	>	>	<	+−−	96	+−0
							−++	101	−+0
101	+−	+−	−+	=	>	<	+−0	100	+−−
							−+0	102	−+−
102	+−	+−	−+	<	>	<	+−+	100	+−0
							−+−	98	−+−

Appendix B

The Lift Pump Envisionment

Quantity	s3	s4	s6	s7	s8	s9
Ds[Amount-of-in(water,liquid,M)]	0	0	1	0	1	1
Ds[Amount-of-in(air,gas,M)]	0	0	0	0	0	0
Ds[Amount-of-in(water,liquid,T)]	0	-1	0	0	0	-1
Ds[Amount-of-in(air,gas,T)]	0	0	0	0	0	0
Ds[Flow-rate(Pouring(water,P1,T))]	-	-1	-	-	-	-1
Ds[Flow-rate(Gas-flow(air,M,T))]	-	-	-	-	-	-
Ds[Flow-rate(Surface-motion(M,R,liquid,M-R))]	-	-	-	-	-1	-1
Ds[Fluid-level(M)]	0	0	0	0	1	1
Ds[Fluid-level(T)]	-1	-1	-	-	-1	-1
Ds[Position(piston)]	-1	-1	-1	-1	-1	-1
Ds[Pressure(M)]	0	0	0	1	0	0
Ds[Pressure(t-s(M,B-S(water,liquid,M)))]	0	0	0	1	-	-
Ds[Pressure(t-s(M,B-S(water,liquid,M-R)))]	-	-	-	-	1	1
Ds[Pressure(t-s(T,B-S(water,liquid,T)))]	-1	-1	-	-	-1	-1
Ds[Pressure(t-s(T,B-S(water,liquid,M-T)))]	-	-	-	-	-	-
Ds[volume(M)]	-1	-1	-1	-1	-1	-1
A[Pressure(T)] A[Pressure(M)]	>	>	>	>=	>	>
A[Pressure(t-s(M,B-S(water,liquid,M)))] A[Pressure(t-s(R,B-S(water,liquid,R)))]	>=	>=	=	>=	-	-
A[Pressure(t-s(M,B-S(water,liquid,M-R)))] A[Pressure(t-s(R,B-S(water,liquid,M-R)))]	-	-	-	-	<	<
Active(Pouring(water,P1,T))	F	T	F	F	F	T
Active(Motion(piston))	T(dn)	T(dn)	T(dn)	T(dn)	T(dn)	T(dn)
Active(Acceleration(piston,handle))	T(dn)	T(dn)	T(dn)	T(dn)	T(dn)	T(dn)
Active(Gas-flow(air,M,T))	F	F	F	F	F	F
Active(Surface-motion(M,R,liquid,M-R))	F	F	F	F	T(up)	T(up)
Bounded-Liquid(M)	T	T	T	T	F	F
Bounded-Liquid(T)	T	T	T	F	T	T
Bounded-Liquid(M-R)	F	F	F	F	T	T
Bounded-Liquid(M-T)	F	F	F	F	F	F
Next state		s3		s28	s3	s3 s4 s8

Quantity	s10	s11	s12	s13	s14	s15
Ds[Amount-of-in(water,liquid,M)]	1	1	0	0	0	0
Ds[Amount-of-in(air,gas,M)]	0	0	0	0	0	0
Ds[Amount-of-in(water,liquid,T)]	0	0	0	-1	0	-1
Ds[Amount-of-in(air,gas,T)]	0	0	0	0	0	0
Ds[Flow-rate(Pouring(water,P1,T))]	-	-	-	1	-	1
Ds[Flow-rate(Gas-flow(air,M,T))]	-	-	-	-	-	-
Ds[Flow-rate(Surface-motion(M,R,liquid,M-R))]	-1	-1	-	-	-	-
Ds[Fluid-level(M)]	1	1	0	0	0	0
Ds[Fluid-level(T)]	0	-	1	1	0	1
Ds[Position(piston)]	-1	-1	1	1	1	1
Ds[Pressure(M)]	0	1	0	0	0	0
Ds[Pressure(t-s(M,B-S(water,liquid,M)))]	1	-	0	0	0	0
Ds[Pressure(t-s(M,B-S(water,liquid,M-R)))]	-	1	-	-	-	-
Ds[Pressure(t-s(T,B-S(water,liquid,T)))]	-	-	1	1	1	1
Ds[Pressure(t-s(T,B-S(water,liquid,M-T)))]	-	-	-	-	-	-
Ds[volume(M)]	-1	1	1	1	1	1
A[Pressure(T)] A[Pressure(M)]	>	>=	>=	>=	>=	>=
A[Pressure(t-s(M,B-S(water,liquid,M)))] A[Pressure(t-s(R,B-S(water,liquid,R)))]	-	-	>=	>=	>=	>=
A[Pressure(t-s(M,B-S(water,liquid,M-R)))] A[Pressure(t-s(R,B-S(water,liquid,M-R)))]	<	<	-	-	-	-
Active(Pouring(water,P1,T))	F	F	F	T	F	T
Active(Motion(piston))	T(dn)	T(dn)	T(up)	T(up)	T(up)	T(up)
Active(Acceleration(piston,handle))	T(dn)	T(dn)	T(up)	T(up)	T(up)	T(up)
Active(Gas-flow(air,M,T))	F	F	F	F	F	F
Active(Surface-motion(M,R,liquid,M-R))	T(up)	T(up)	F	F	F	F
Bounded-Liquid(M)	F	F	T	T	T	T
Bounded-Liquid(T)	F	F	T	T	T	T
Bounded-Liquid(M-R)	T	T	F	F	F	F
Bounded-Liquid(M-T)	F	F	F	F	F	F
Next state		s28 s29	s13		s12 s13	s13

Quantity	s16	s17	s18	s19	s20
Ds[Amount-of-in(water,liquid,M)]	0	0	0	0	1
Ds[Amount-of-in(air,gas,M)]	-1	0	0	0	0
Ds[Amount-of-in(water,liquid,T)]	0	0	0	0	0
Ds[Amount-of-in(air,gas,T)]	1	0	0	0	0
Ds[Flow-rate(Pouring(water,P1,T))]	-	-	-	-	-
Ds[Flow-rate(Gas-flow(air,M,T))]	-1	-	-	-	-
Ds[Flow-rate(Surface-motion(M,R,liquid,M-R))]	-	-	-	-	-1
Ds[Fluid-level(M)]	0	0	0	0	1
Ds[Fluid-level(T)]	-	-	-	-	1
Ds[Position(piston)]	1	1	1	1	1
Ds[Pressure(M)]	-1	0	-1	0	0
Ds[Pressure(t-s(M,B-S(water,liquid,M)))]	-1	0	-1	0	-
Ds[Pressure(t-s(M,B-S(water,liquid,M-R)))]	-	-	-	-	1
Ds[Pressure(t-s(T,B-S(water,liquid,T)))]	-	-	-	-	1
Ds[Pressure(t-s(T,B-S(water,liquid,M-T)))]	-	-	-	-	-
Ds[volume(M)]	1	1	1	1	1
A[Pressure(T)] A[Pressure(M)]	<	>=	>=	>=	>
A[Pressure(t-s(M,B-S(water,liquid,M)))] A[Pressure(t-s(R,B-S(water,liquid,R)))]	<	>=	>=	>=	-
A[Pressure(t-s(M,B-S(water,liquid,M-R)))] A[Pressure(t-s(R,B-S(water,liquid,M-R)))]	-	-	-	-	<
Active(Pouring(water,P1,T))	F	F	F	F	F
Active(Motion(piston))	T(up)	T(up)	T(up)	T(up)	T(up)
Active(Acceleration(piston,handle))	T(up)	T(up)	T(up)	T(up)	T(up)
Active(Gas-flow(air,M,T))	T	F	F	F	F
Active(Surface-motion(M,R,liquid,M-R))	F	F	F	F	T(up)
Bounded-Liquid(M)	T	T	T	T	F
Bounded-Liquid(T)	F	F	F	F	T
Bounded-Liquid(M-R)	F	F	F	F	T
Bounded-Liquid(M-T)	F	F	F	F	F
Next state	s18 s27		s27	s17	s21

Quantity	s21	s22	s23	s24	s26
Ds[Amount-of-in(water,liquid,M)]	1	1	1	1	0
Ds[Amount-of-in(air,gas,M)]	0	0	0	0	0
Ds[Amount-of-in(water,liquid,T)]	-1	0	1	0	0
Ds[Amount-of-in(air,gas,T)]	0	0	0	0	0
Ds[Flow-rate(Pouring(water,P1,T))]	1	-	1	-	-
Ds[Flow-rate(Gas-flow(air,M,T))]	-	-	-	-	-
Ds[Flow-rate(Surface-motion(M,R,liquid,M-R))]	-1	-1	-1	-1	-1
Ds[Fluid-level(M)]	1	1	1	1	1
Ds[Fluid-level(T)]	1	1	1	-	-
Ds[Position(piston)]	1	1	1	1	1
Ds[Pressure(M)]	0	0	0	0	0
Ds[Pressure(t-s(M,B-S(water,liquid,M)))]	-	-	-	-	-
Ds[Pressure(t-s(M,B-S(water,liquid,M-R)))]	1	1	1	1	1
Ds[Pressure(t-s(T,B-S(water,liquid,T)))]	1	1	1	-	-
Ds[Pressure(t-s(T,B-S(water,liquid,M-T)))]	-	-	-	-	-
Ds[volume(M)]	1	1	1	1	1
A[Pressure(T)] A[Pressure(M)]	>	>=	>=	<	>=
A[Pressure(t-s(M,B-S(water,liquid,M)))] A[Pressure(t-s(R,B-S(water,liquid,R)))]	-	-	-	-	-
A[Pressure(t-s(M,B-S(water,liquid,M-R)))] A[Pressure(t-s(R,B-S(water,liquid,M-R)))]	<	<	<	<	<
Active(Pouring(water,P1,T))	T	F	T	F	F
Active(Motion(piston))	T(up)	T(up)	T(up)	T(up)	T(up)
Active(Acceleration(piston,handle))	T(up)	T(up)	T(up)	T(up)	T(up)
Active(Gas-flow(air,M,T))	F	F	F	F	F
Active(Surface-motion(M,R,liquid,M-R))	T(up)	T(up)	T(up)	T(up)	T(up)
Bounded-Liquid(M)	F	F	F	F	F
Bounded-Liquid(T)	T	T	T	F	F
Bounded-Liquid(M-R)	T	T	T	T	T
Bounded-Liquid(M-T)	F	F	F	F	F
Next state		s13	s12 s23		s17

Quantity	s27	s28	s29	s30	s31
Ds[Amount-of-in(water,liquid,M)]	1	0	-1	-1	-1
Ds[Amount-of-in(air,gas,M)]	0	-1	0	0	0
Ds[Amount-of-in(water,liquid,T)]	0	0	1	1	1
Ds[Amount-of-in(air,gas,T)]	0	1	0	0	0
Ds[Flow-rate(Pouring(water,P1,T))]	-	-	-1	-	-
Ds[Flow-rate(Gas-flow(air,M,T))]	-	1	-	-	-
Ds[Flow-rate(Surface-motion(M,R,liquid,M-R))]	-1	-	-	-	-
Ds[Fluid-level(M)]	1	0	-	-	-1
Ds[Fluid-level(T)]	-	-	-1	0	0
Ds[Position(piston)]	1	-1	-1	-1	-1
Ds[Pressure(M)]	1	1	0	0	0
Ds[Pressure(t-s(M,B-S(water,liquid,M)))]	-	1	-	-	-1
Ds[Pressure(t-s(M,B-S(water,liquid,M-R)))]	1	-	-	-	-
Ds[Pressure(t-s(T,B-S(water,liquid,T)))]	-	-	-	-	-
Ds[Pressure(t-s(T,B-S(water,liquid,M-T)))]	-	-	-1	0	-1
Ds[volume(M)]	1	-1	-1	-1	-1
A[Pressure(T)] A[Pressure(M)]	>=	<	=	=	=
A[Pressure(t-s(M,B-S(water,liquid,M)))] A[Pressure(t-s(R,B-S(water,liquid,R)))]	-	>=	-	-	>=
A[Pressure(t-s(M,B-S(water,liquid,M-R)))] A[Pressure(t-s(R,B-S(water,liquid,M-R)))]	<	-	-	-	-
A[Pressure(t-s(T,B-S(water,liquid,M-T)))] A[Pressure(t-s(R,B-S(water,liquid,R)))]	-	-	>=	>=	>=
Active(Pouring(water,P1,T))	F	F	T	F	F
Active(Motion(piston))	T(dn)	T(dn)	T(dn)	T(dn)	T(dn)
Active(Acceleration(piston,handle))	T(dn)	T(dn)	T(dn)	T(dn)	T(dn)
Active(Gas-flow(air,M,T))	F	T	F	F	F
Active(Surface-motion(M,R,liquid,M-R))	T(up)	F	F	F	F
Bounded-Liquid(M)	F	T	F	F	T
Bounded-Liquid(T)	F	F	F	F	F
Bounded-Liquid(M-R)	T	T	F	F	F
Bounded-Liquid(M-T)	F	T	T	T	T
Next state			s30		s30

Appendix C

The Internal Combustion Engine Envisionment

	TDC				
Quantity	s1	s9	s10	s21	s22
Ds[angular-velocity(crank1)]	1	1	1	0	0
Ds[heat(c-g)]	0	0	0	0	0
Ds[position(piston1)]	0	0	0	0	0
Ds[pressure(c-g)]	0	0	0	0	0
Ds[temperature(c-g)]	0	0	0	0	0
Ds[velocity(piston1)]	0	0	0	0	0
Ds[volume(c-g)]	0	0	0	0	0
Ds[volume(cylinder1)]	0	0	0	0	0
A[velocity(piston1)]	=0	=0	=0	=0	=0
A[angular-velocity(crank1)]	<0	<0	<0	=0	=0
A[CLTDC-tag]	>0	<=0	<=0	<=0	<=0
A[inclination(c-r,crank1)]	-	-	-	-	-
A[pressure(c-g)] A[nonstop-pres]	>	-	-	<=	<=
A[pressure(c-g)] A[combustion-pres]	>	<=	<=	<	<
(relative-position p2 p1)	0+	0+	0+	0+	0+
(relative-position p3 p2)	0+	0+	0+	0+	0+
(relative-motion p2 p1)	CW	CW	CW	CW	CW
(relative-motion p3 p2)	CCW	CCW	CCW	CCW	CCW
(spark piston1 cylinder1)	T	T	F	T	F
kinetic-friction(piston1)	F	F	F	F	F
resisted-CW-motion(crank1)	T(+)	T(+)	T(+)	F	F
motion(piston1)	F	F	F	F	F
comp-exp(c-g)	F	F	F	F	F
next-states	s2	s1			

	CLTDC-EXP						
Quantity	s2	s2-I	s4	s4-I	s7	s7-I	s6
Ds[angular-velocity(crank1)]	1	1	1	1	1	1	1
Ds[heat(c-g)]	-1	-1	-1	-1	-1	-1	0
Ds[position(piston1)]	-1	-1	-1	-1	-1	-1	0
Ds[pressure(c-g)]	-1	-1	-1	-1	-1	-1	0
Ds[temperature(c-g)]	-1	-1	-1	-1	-1	-1	0
Ds[velocity(piston1)]	1	1	1	1	1	1	0
Ds[volume(c-g)]	1	1	1	1	1	1	0
Ds[volume(cylinder1)]	1	1	1	1	1	1	0
A[velocity(piston1)]	<0	<0	<0	<0	<0	<0	=0
A[angular-velocity(crank1)]	<0	<0	<0	<0	<0	<0	<0
A[CLTDC-tag]	>0	>0	>0	>0	>0	>0	>0
A[inclination(c-r,crank1)]	-	-	-	-	<0	<0	-
A[pressure(c-g)] A[nonstop-pres]	>	>	>	>	>	>	>
A[pressure(c-g)] A[combustion-pres]	≠	=	≠	=	≠	=	<
(relative-position p2 p1)	++	++	+0	+0	+−	+−	0−
(relative-position p3 p2)	−+	−+	−+	−+	−+	−+	0+
(relative-motion p2 p1)	CW	CW	CW	CW	CW	CW	CW
(relative-motion p3 p2)	CCW	CCW	no	no	CW	CW	CW
kinetic-friction(piston1)	T(+)	T(+)	T(+)	T(+)	T(+)	T(+)	F
resisted-CW-motion(crank1)	T	T	T	T	T	T	T
motion(piston1)	T(dn)	T(dn)	T(dn)	T(dn)	T(dn)	T(dn)	F
comp-exp(c-g)	T	T	T	T	T	T	F
next-states	s2-I s4 s4-I	s2 s4	s4-I s7 s7-I	s4 s7	s7-I s6	s7	s5

	not-CLTDC-EXP					
Quantity	s11	s11-I	s12	s14	s14-I	s15
Ds[angular-velocity(crank1)]	1	1	0	1	1	0
Ds[heat(c-g)]	-1	-1	0	-1	-1	0
Ds[position(piston1)]	-1	-1	0	-1	-1	0
Ds[pressure(c-g)]	-1	-1	0	-1	-1	0
Ds[temperature(c-g)]	-1	-1	0	-1	-1	0
Ds[velocity(piston1)]	1	1	0	1	1	0
Ds[volume(c-g)]	1	1	0	1	1	0
Ds[volume(cylinder1)]	1	1	0	1	1	0
A[velocity(piston1)]	<0	<0	=0	<0	<0	=0
A[angular-velocity(crank1)]	<0	<0	=0	<0	<0	=0
A[CLTDC-tag]	<=0	<=0	<=0	<=0	<=0	<=0
A[inclination(c-r,crank1)]	-	-	-	-	-	-
A[pressure(c-g)] A[nonstop-pres]	≠	>=	<=	≠	>=	<=
A[pressure(c-g)] A[combustion-pres]	≠	<=	<	≠	<=	<
(relative-position p2 p1)	++	++	++	+0	+0	+0
(relative-position p3 p2)	−+	−+	−+	−+	−+	−+
(relative-motion p2 p1)	CW	CW	no	CW	CW	no
(relative-motion p3 p2)	CCW	CCW	no	no	no	no
kinetic-friction(piston1)	T(+)	T(+)	F	T(+)	T(+)	F
resisted-CW-motion(crank1)	T	T	F	T	T	F
motion(piston1)	T(dn)	T(dn)	F	T(dn)	T(dn)	F
comp-exp(c-g)	T	T	F	T	T	F
next-states	s11-I s12 s14 s14-I	s11 s14		s14-I s15 s19 s19-I	s14 s19	

Quantity	not-CLTDC-EXP s19	s19-I	s20	s17	s18
Ds[angular-velocity(crank1)]	1	1	0	1	0
Ds[heat(c-g)]	-1	-1	0	0	0
Ds[position(piston1)]	-1	-1	0	0	0
Ds[pressure(c-g)]	-1	-1	0	0	0
Ds[temperature(c-g)]	-1	-1	0	0	0
Ds[velocity(piston1)]	1	1	0	0	0
Ds[volume(c-g)]	1	1	0	0	0
Ds[volume(cylinder1)]	1	1	0	0	0
A[velocity(piston1)]	<0	<0	=0	=0	=0
A[angular-velocity(crank1)]	<0	<0	=0	=0	=0
A[CLTDC-tag]	<=0	<=0	<=0	<=0	<=0
A[inclination(c-r,crank1)]	<	<	<	-	-
A[pressure(c-g)] A[nonstop-pres]	≠	>=	<=	-	<=
A[pressure(c-g)] A[combustion-pres]	≠	<=	<		<
(relative-position p2 p1)	+−	+−	+−	0−	0−
(relative-position p3 p2)	−+	−+	−+	0+	0+
(relative-motion p2 p1)	CW	CW	no	CW	CW
(relative-motion p3 p2)	CCW	CW	no	CW	CW
kinetic-friction(piston1)	T(+)	T(+)	F	F	F
resisted-CW-motion(crank1)	T	T	F	T	F
motion(piston1)	T(dn)	T(dn)	F	F	F
comp-exp(c-g)	T	T	F	F	F
next-states	s19-I s20 s17	s19 s17		s18 s5 s19-I	

Quantity	Compression				
	s5	s5-I	s16	s3	s3-I
Ds[angular-velocity(crank1)]	1	1	0	1	1
Ds[heat(c-g)]	1	1	0	1	1
Ds[position(piston1)]	1	1	0	1	1
Ds[pressure(c-g)]	1	1	0	1	1
Ds[temperature(c-g)]	1	1	0	1	1
Ds[velocity(piston1)]	-1	-1	0	-1	-1
Ds[volume(c-g)]	-1	-1	0	-1	-1
Ds[volume(cylinder1)]	-1	-1	0	-1	-1
A[velocity(piston1)]	>0	>0	=0	>0	>0
A[angular-velocity(crank1)]	<0	<0	=0	<0	<0
A[CLTDC-tag]	-	-	-	-	-
A[inclination(c-r,crank1)]	>0	>0	>0	-	-
A[pressure(c-g)] A[nonstop-pres]	$\neq$	=	<=	$\neq$	=
A[pressure(c-g)] A[combustion-pres]	<	<	<	<	<
(relative-position p2 p1)	−−	−−	−−	−0	−0
(relative-position p3 p2)	++	++	++	++	++
(relative-motion p2 p1)	CW	CW	no	no	no
(relative-motion p3 p2)	CW	CW	no	no	no
kinetic-friction(piston1)	T(-)	T(-)	F	T(-)	T(-)
resisted-CW-motion(crank1)	T	T	F	T	T
motion(piston1)	T(up)	T(up)	F	T(up)	T(up)
comp-exp(c-g)	T	T	F	T	T
next-states	s5-I s16 s3 s3-I	s5 s3		s3-I s13 s0 s0-I	s3 s0

	Compression			
Quantity	s13	s0	s0-I	s8
Ds[angular-velocity(crank1)]	0	1	1	0
Ds[heat(c-g)]	0	1	1	0
Ds[position(piston1)]	0	1	1	0
Ds[pressure(c-g)]	0	1	1	0
Ds[temperature(c-g)]	0	1	1	0
Ds[velocity(piston1)]	0	-1	-1	0
Ds[volume(c-g)]	0	-1	-1	0
Ds[volume(cylinder1)]	0	-1	-1	0
A[velocity(piston1)]	=0	>0	>0	=0
A[angular-velocity(crank1)]	=0	<0	<0	=0
A[CLTDC-tag]	-	-	-	-
A[inclination(c-r,crank1)]	-	-	-	-
A[pressure(c-g)] A[nonstop-pres]	<=	≠	=	<=
A[pressure(c-g)] A[combustion-pres]	<	<	<	<
(relative-position p2 p1)	−0	−+	−+	−+
(relative-position p3 p2)	++	++	++	++
(relative-motion p2 p1)	no	CW	CW	no
(relative-motion p3 p2)	no	CCW	CCW	no
kinetic-friction(piston1)	F	T(-)	T(-)	F
resisted-CW-motion(crank1)	F	T	T	F
motion(piston1)	F	T(up)	T(up)	F
comp-exp(c-g)	F	T	T	F
next-states		s0-I s8 s9 s10 s21 s22	s0	