AD-A096 455 MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTE--ETC F/G 12/2
A STUDY OF QUALITATIVE AND GEOMETRIC KNOWLEDGE IN REASONING ABO--ETC(U)
FEB 81 K D FORBUS N00014-80-C-0505
UNCLASSIFIED AI-TR-615-REV NL

1 OF 2

AD A
096455

AD A096455

AI•TR•615

(12)

BS

LEVEL II

A STUDY OF QUALITATIVE AND GEOMETRIC KNOWLEDGE IN REASONING ABOUT MOTION

KENNETH D. FORBUS

FEBRUARY 1981

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

ARTIFICIAL INTELLIGENCE LABORATORY

81 3 17 094

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

14

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER
AI-TR-615-REV

2. GOVT ACCESSION NO.
AD-A096455

3. RECIPIENT'S CATALOG NUMBER
9

4. TITLE (and Subtitle)
6
A Study of Qualitative and Geometric Knowledge in Reasoning about Motion. Revision.

5. TYPE OF REPORT & PERIOD COVERED
Technical Report,
16

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
10
Kenneth ~~R.~~ Forbus Dale

8. CONTRACT OR GRANT NUMBER(s)
15
N00014-80-C-0505

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Artificial Intelligence Laboratory
545 Technology Square
Cambridge, Massachusetts 02139

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Advanced Research Projects Agency
1400 Wilson Blvd
Arlington, Virginia 22209

11

12. REPORT DATE
Feb[redacted]81

13. NUMBER OF PAGES
123

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)
Office of Naval Research
Information Systems
Arlington, Virginia 22217

12 125

15. SECURITY CLASS. (of this report)
UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

DTIC
MAR 1 7 1981
E

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Spatial Reasoning
Geometric Representation
Problem Solving
Qualitative Knowledge
Constraints
Qualitative Simulation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Reasoning about motion is an important part of our commonsense knowledge, involving fluent spatial reasoning. This work studies the qualitative and geometric knowledge required to reason in a world that consists of balls moving through space constrained by collisions with surfaces, including dissipative forces and multiple moving objects. An analog geometry representation serves the program as a diagram, allowing many spatial questions to be answered by numeric calculation. It also provides the

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED 407483

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. foundation for the construction and use of a place vocabulary, the symbolic descriptions of space required to do qualitative reasoning about motion in the domain.

The actual motion of a ball is described as a network consisting of descriptions of qualitatively distinct types of motion. Implementing the elements of these networks in a constraint language allows the same elements to be used for both analysis and simulation of motion. A qualitative description of the actual motion is also used to check the consistency of assumptions about motion.

A process of qualitative simulation is used to describe the kinds of motion possible from some state. The ambiguity inherent in such a description can be reduced by assumptions about physical properties of the ball or assumptions about its motion. Each assumption directly rules out some kinds of motion, but other knowledge is required to determine the indirect consequences of making these assumptions. Some of this knowledge is domain dependent and relies heavily on spatial descriptions.

B

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the Laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-80-C-0505

A Study of Qualitative and Geometric Knowledge in Reasoning about Motion

by

Kenneth Dale Forbus

Massachusetts Institute of Technology

February 1981

Revised version of a thesis submitted to the Department of Electrical Engineering and Computer Science on February 6, 1980 in partial fulfillment of the requirements for the degree of Master of Science.

Abstract

Reasoning about motion is an important part of our commonsense knowledge, involving fluent spatial reasoning. This work studies the qualitative and geometric knowledge required to reason in a world that consists of balls moving through space constrained by collisions with surfaces, including dissipative forces and multiple moving objects.

An analog geometry representation serves the program as a diagram, allowing many spatial questions to be answered by numeric calculation. It also provides the foundation for the construction and use of a place vocabulary, the symbolic descriptions of space required to do qualitative reasoning about motion in the domain.

The actual motion of a ball is described as a network consisting of descriptions of qualitatively distinct types of motion. Implementing the elements of these networks in a constraint language allows the same elements to be used for both analysis and simulation of motion. A qualitative description of the actual motion is also used to check the consistency of assumptions about motion.

A process of qualitative simulation is used to describe the kinds of motion possible from some state. The ambiguity inherent in such a description can be reduced by assumptions about physical properties of the ball or assumptions about its motion. Each assumption directly rules out some kinds of motion, but other knowledge is required to determine the indirect consequences of making these assumptions. Some of this knowledge is domain dependent and relies heavily on spatial descriptions.

Thesis Supervisor: Professor Gerald Sussman

Title: Associate Professor of Electrical Engineering and Computer Science

Acknowledgements

I would like to thank Gerald Sussman, my thesis advisor, for showing me what it means to do "real AI".

Patrick Winston provided much needed encouragement in the darkest hours.

Johan deKleer has influenced my ideas over the years.

Marilyn Matz was a signifigent factor in changing my writing from gibberish to passable english.

Tomas Lozano-Perez was always willing to discuss computational geometry.

Al Stevens provided encouragement and fresh perspective.

Peter Andreae, Anni Bruss, and Luc Steels, my long-suffering officemates, put up with my incoherent ramblings.

Tom Knight, Jack Holloway, and Richard Greenblatt masterminded the creation and maintence of the computational resources at the lab, especially the Lisp Machine. Without such resources serious experimentation would be impossible.

Cathy Medich, David Quinto, Richard B. Bernstein, and Marjorie Roberts kept me from getting too cut off from the rest of the world.

CONTENTS

FIGURES

1. Introduction

1.1 The problem under study

We are very good at reasoning about motion through space. For example, we know that if two balls are thrown into a well they can collide, but if one ball is always in the well and the other always outside they cannot. The knowledge we use in this situation appears simpler than formal mechanics, and is based on our experience with the physical world. This qualitative kind of understanding requires fluent spatial reasoning. Capturing the knowledge needed to reason about motion brings us closer to an explicit understanding of commonsense knowledge, important both for understanding how people work and for making our machines smarter.

In this work the qualitative and geometric knowledge needed to understand the motion of balls through space under the influence of gravity is examined. The computational aspects of these problems were investigated by writing a computer program called FROB, which reasons about these situations.

1.2 Main Ideas

This work studies the role of geometric and qualitative knowledge in reasoning about motion through space. The central ideas are summarized here.

Spatial Reasoning-

1. We have not yet discovered quite why people are good at reasoning about space. Certain techniques we know how to program (such as algebraic manipulation and proving theorems) do not seem to capture this ability. My theory is that an extra edge people have in dealing with space is their visual apparatus. In particular, spatial questions can often be decided by interpreting the results of perception. This method can also be applied to pencil and paper diagrams, where the marks on paper reflect the spatial arrangements of the things they represent. The geometry representation used in this work, the Metric Diagram, requires all numerical parameters of its elements be specified. As a result, most spatial questions can be decided by calculation.

2. There are several different classes of problems that we consider to involve spatial reasoning, including navigation, knot tying, and motion problems. I claim that the most important factor these problems share is a notion of PLACE. By PLACE, I mean a piece of space (point, line, surface, region, etc.) such that all parts of it share some common property. Qualitative reasoning about space involves the use of a vocabulary of PLACES, whose interconnections and relationships are specified symbolically. This vocabulary is especially useful if it is based on a more analog representation of space (such as the Metric Diagram used here). My work explicates the PLACE vocabulary for a particular class of motion problems.

Qualitative knowledge

1. The motion of an object can usually be described as a sequence of qualitatively distinct actions. In computational terms this corresponds to a network consisting of descriptions of each action, linked by descriptions of the state of the object before and after each action. Forward deduction within this network allows proposed motions to be analyzed, including checking for inconsistencies. If each type of action can be described by equations of motion, such a network can be built by a process of simulation. The advantage of this form of simulation over the traditional incremental time simulations is that the computation time and the size of the resulting description are proportional to the qualitative complexity of the motion rather than the size of whatever time increment is used.

2. A description of the kinds of motion possible from some state can be generated by a qualitative process of simulation called envisioning. One way to resolve the ambiguities inherent in such a description is to simulate the motion. Physical assumptions (such as the elasticity and energy of a ball) and simple qualitative constraints on motion (such as assuming that a ball cannot be in a particular place, for example) can also reduce the number of possible motions. While each type of assumption directly removes some of the possiblities, qualitative knowledge about motion must be used to determine the consequences of their removal. This qualitative knowledge is domain dependent and makes heavy use of spatial descriptions.

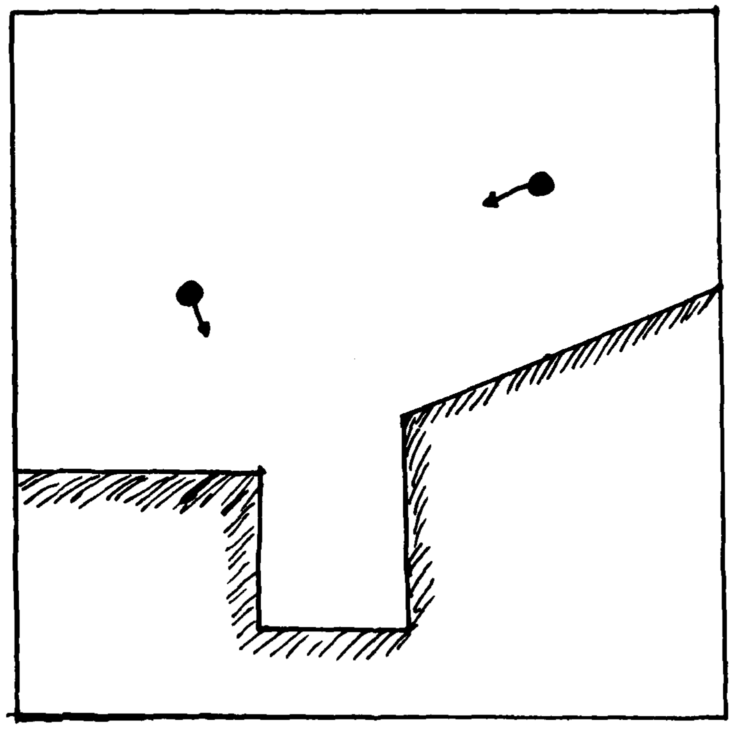
1.3 The Domain and Some Questions

The Bouncing Ball world is the domain I have created for studying motion. A situation in this world consists of a scene and one or more balls. The scene is a specification of surfaces in the form of a two dimensional diagram, where the surfaces are restricted to being line segments. The balls are considered to be point masses, and need not be perfectly elastic. Figure 1 contains a drawing of a Bouncing Ball situation.

By restricting myself to this domain, I am ignoring

1. The third dimension
-a two dimensional diagram is all that is required
2. All forces other than gravity
-we do not allow the balls to be magnetic, charged, or made of water
3. Sliding
-friction is not an issue, nor the shape of a surface except at a point of contact
4. Complex shape
-we don't have to worry about how something lands, how it bounces differently if it lands on a corner as opposed to a face, nor about the way a falling pendulum might flail about
5. Spin
6. Air Resistance

Fig. 1. The Bouncing Ball world
To study motion through space, the Bouncing Ball world was created. A situation in the bouncing ball world consists of a scene and one or more balls. The scene consists of surfaces modelled as line segments specified in a diagram. Balls are modelled as point masses.



Motion after two balls collide will not be described. In part this is due to a lack of analytic models for the actual exchange of energy such collisions involve. More fundamentally, I wish to avoid the problem of retracting the previously computed descriptions.

These restrictions simplify the domain greatly. They seem reasonable because people often do not reason well about situations where air resistance, spin, and shape come into play.[1][2][3]

At this point the reader may despair of there being any questions of interest left. This is not the case. The four basic question this work focuses on are

1. What can a ball do next?
2. Where can it go next?
3. Where can it end up?
4. Can these two balls collide?

As we shall see, answering these questions requires good descriptions of motion. Since motion pervades our dealings with the physical world, knowing about it must be an important part of our common sense knowledge. Understanding the Bouncing Ball world is important because it corresponds to an important class of motion problems. I would identify three classes of simple motion as

SLIDE (motion constrained by constant contact with a surface)
FLY (motion constrained by occasional contact and gravity)
SWING (motion constrained by rotational connection)

The relation of these classes of motion to common sense knowledge is illustrated by figure 2. This simplfied ontology does not include shape (where SLIDE could become ROLL), nor does it include objects that are more interesting because of their material composition, such as strings and springs. A theory for the Bouncing Ball world is a (two dimensional) theory for the FLY class of motion. Having theories for each class of motion should allow the description of more complex motions by decomposition into regions where these theories are applicable. Figure 3 shows how the operation of a sling can be described in this manner. Further research is necessary to determine how much physical reasoning can be done by representing space in two dimensional slices.

---

1. In his book Newtonian Mechanics A. P. French mentions an "ill-advised" bet from the folklore of physics. Imagine two balls, one which is dropped while the other is shot horizontally from a cannon at the same height. The bet concerns which one will strike the ground first. If both balls are moving slowly, they will of course hit at the same time. But air resistance will actually cause the second ball to take longer to hit the ground if the muzzle velocity is high.
2. It is certainly true that people can be experts at tennis, ping pong, and billards, all of which require understanding spin. This understanding is almost certainly qualitative and formalizable in the framework presented here. Our difficulty at explaining why a dropped book will spin in a stable fashion about only two of the three possible axes of rotation argues against our common sense theory of spin being very deep.
3. Craps would be less a sucker's game if we could predict how dice would roll. The problem we have is not where they will hit (which for most of the trajectory is independent of exact shape), but where they will bounce.

Fig. 2. A view of the role of dynamics in Commonsense Knowledge
The theory of dynamics for the Bouncing Ball world is a 2D theory for FLY.

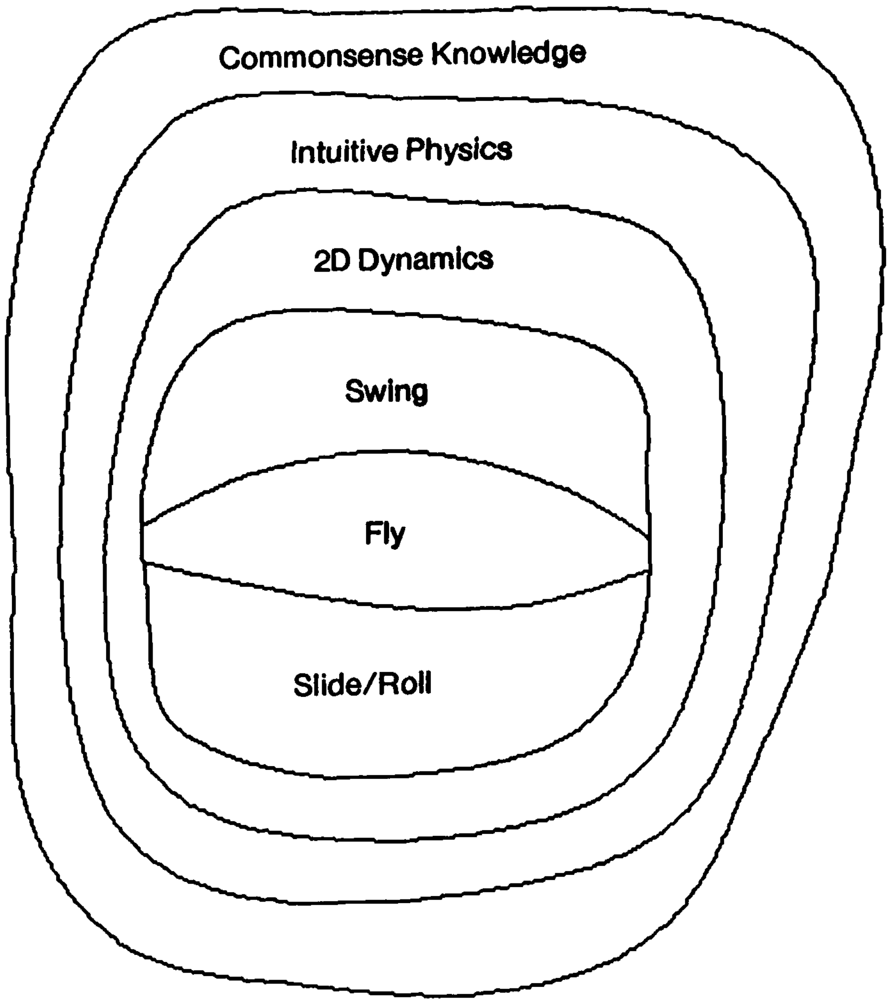
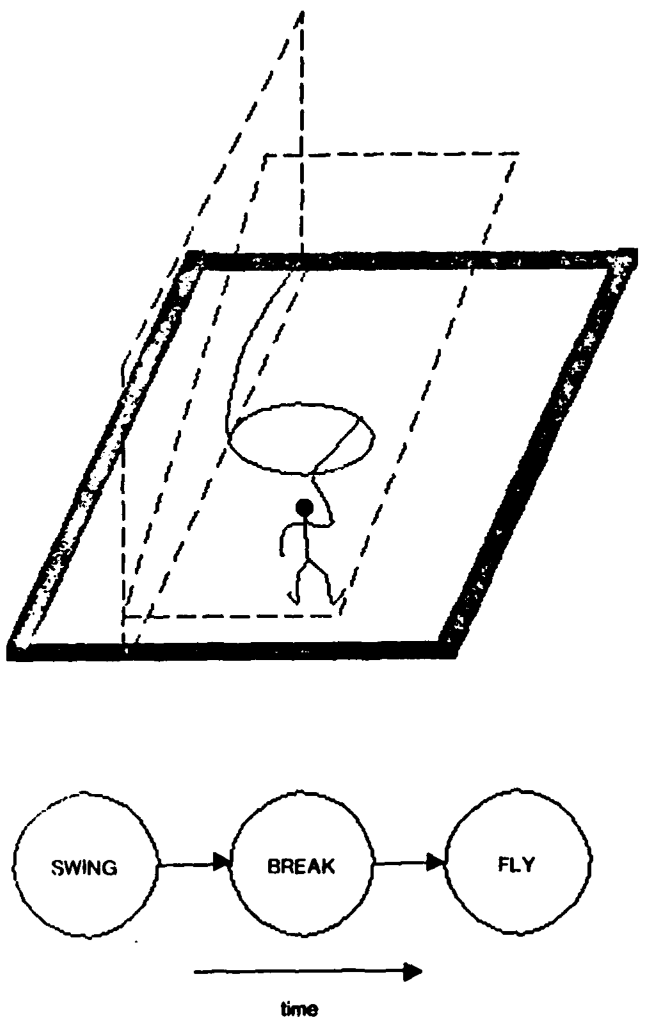


Fig. 3. Describing complex motion
Theories of simple motion can be composed to form a description of a complex motion.



1.4 FROB

A program called FROB was implemented to explore the issues of reasoning about the Bouncing Ball world. This section provides an overview of the descriptions and processes involved in this program and how they reflect the main ideas mentioned previously.

The interrelationships between the types of descriptions in FROB and the processes that create and manipulate them are illustrated in Figure 4. Figure 5 is a schematized example of the descriptions produced for a simple Bouncing Ball world situation.

The Metric Diagram is the principle geometric representation in FROB. Initially it contains a specification of the surface geometry for the scene. A set of geometric analysis routines breaks up the space into solid regions and qualitatively distinct PLACEs. All spatial aspects of the various descriptions are represented by annotating this diagram with new elements as processing proceeds.

The basic qualitative description of free space is the Space Graph. The nodes of this graph are regions of free space and the boundaries between them (including surfaces and borders). The edges express adjacency relations according to the label attached to them. The nodes of this graph and collections of these nodes form the place vocabulary for a particular scene. Since each node is specified within the Metric Diagram, mapping positions and trajectories into the qualitative space representation is easy. The fact that nodes in the Space Graph do not overlap enables new places to be easily created by composing nodes. The graph structure also serves as a framework for several kinds of processing involving qualitative descriptions of space. The place vocabulary is the same for all balls because each has the same forces and none have shape or size.

The Sequence Graph is a description of the possible motions a ball can undergo from some state. It is couched in qualitative terms, and produced by a qualitative process of simulation known as envisioning [deKleer, 1975]. Intuitively, envisioning corresponds to "imagining what can happen". It can be used as a way to summarize motion and as a device for assimilating certain assumptions about physical properties of a ball and constraints on its motion. This assimlation process allows the use of the Sequence Graph to check the correspondence of the actual motion of a ball with the assumptions made about it.

The actual motion of a ball is described by the Action Sequence. The Action Sequence consists of instances of qualitatively distinct types of motion, linked by descriptions of the state of the ball between these motions. It can either be created by the user for purposes of analysis, or by a process of quantitative simulation. In FROB the descriptions of balls, states of a ball, and types of motion are expressed as constraints, which allows the same data structures to be used for simulation and analysis.

Descriptions of balls and their motion can be added in an incremental fashion. Each new piece of information can cause FROB to build new descriptions of motion and update old ones, including checking for inconsistencies. Simulation is performed only upon request. These descriptions constitute the program's understanding of the situation. The user may ask questions by calling procedures that interrogate these data structures to produce an answer. Giving FROB more information can result in a better answer. For example, it takes more information to determine where two balls actually collide than it does to determine only that they might collide.

Fig. 4. Flow of information in FROB

A situation consists of surfaces specified as a Metric Diagram. Geometric analysis determines what parts of space are considered solid and computes a qualitative description of free space. Information about balls, their parameters, and their motion can be added interactively.

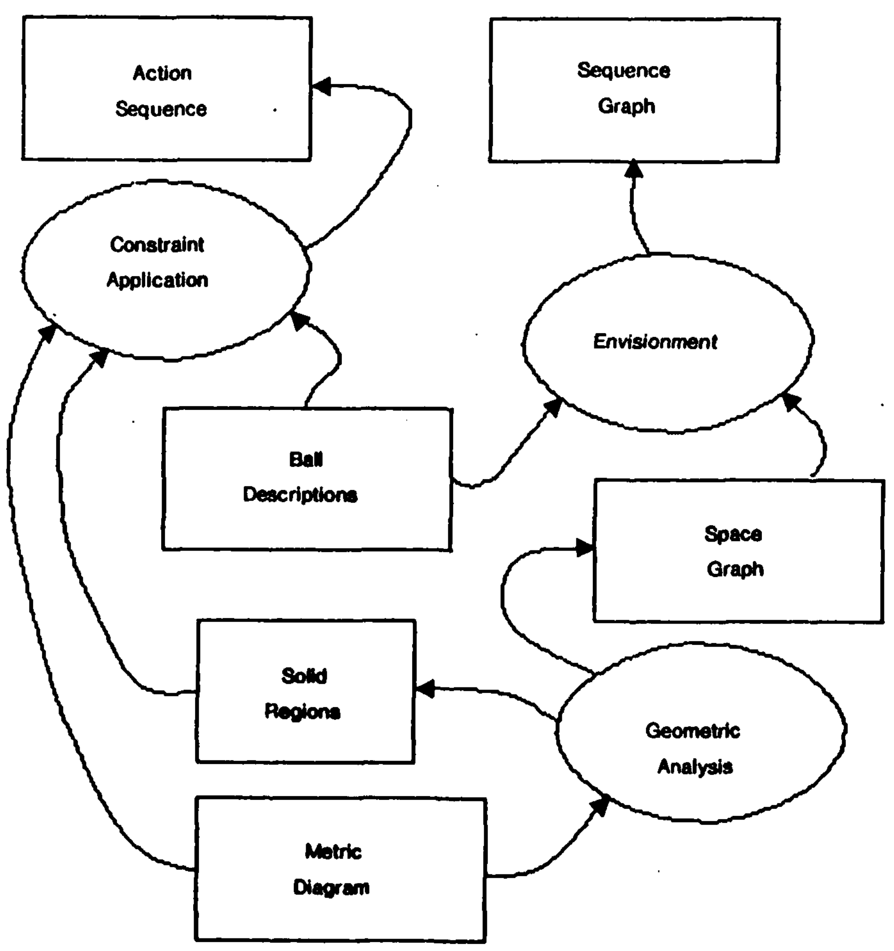
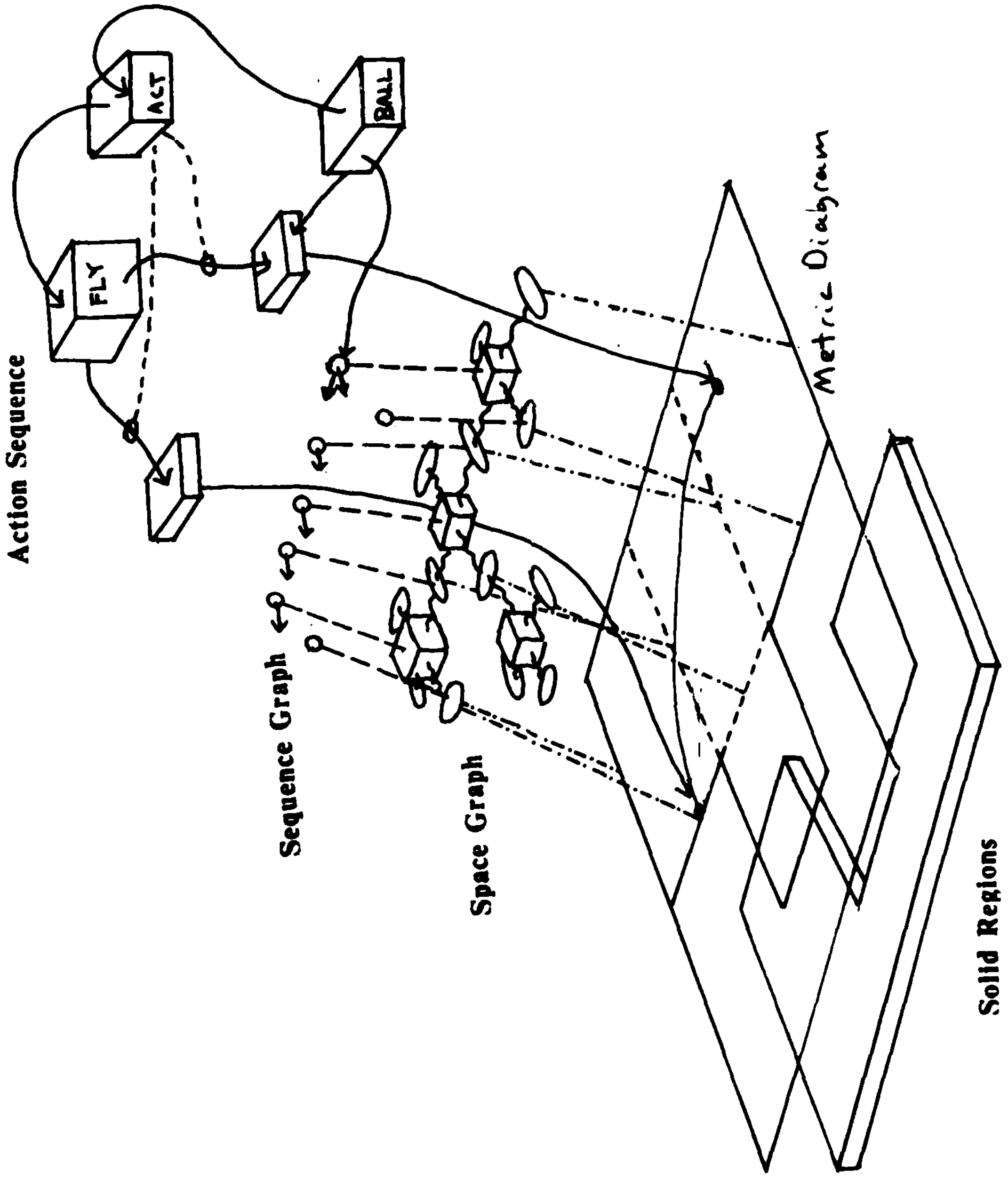


Fig. 5. Descriptions used in FROB
Metric Diagram - basic geometry representation
Space Graph - Qualitative description of space
Sequence Graph - Possible motions of a ball from a state
Action Sequence - Actual motion of a ball



FROB has been implemented both in MACLISP on the PDP 10 and in the dialect of Lisp used by CADR, the MIT Lisp Machine. Many illustrations in this report use hard copy from the program.

1.5 Relationship to other work

The view of common sense knowledge used here is very similar to the Naive Physics approach of Hayes[Hayes a]. Believing that computational issues may cloud epistemological problems, Hayes identifies axioms of common sense physics independent of implementation. The only Naive Physics theory developed at present concerns reasoning about liquids. Despite a shared interest in capturing the knowledge needed to deal with simple physical situations, my work differs in two major ways. My work explores computational issues, particularly the use of a diagram. Secondly, the impact of simpler kinds of knowledge on a quantitative description is also considered.

A very different approach is taken in [Bundy],[McDermott and Larkin], and [Novak], which study the knowledge required to solve freshman physics problems. The reasoning these programs perform is very much like that explicitly reflected in the protocols of human subjects solving the same problem.[1] These programs usually includes some linguistic skill (the initial description is a word problem), and concentrate on manipulating algebraic expressions to get "the answer". Only one program of this type has been extended to deal with motion[2] , and none of them can solve problems involving motion through space. [Novak] and [McDermott and Larkin] contain geometry representations they refer to as diagrams, but these express connectivity rather than free space.

These programs address many issues in addition to common sense reasoning, such as understanding natural language input and minimizing the effort required to solve a specific problem. These issues are ignored in this work in order to better concentrate on describing motion.

The role of qualitative knowledge in describing motion was first explored in [deKleer 75], which described a program called NEWTON that solved problems about point masses sliding without friction on surfaces. This "Roller Coaster" world is a subset of the SLIDE case of motion in the ontology presented earlier.

deKleer introduced the notion of envisioning, which is the process of "imagining what will happen". By using local rules on a qualitative description of a scene NEWTON builds a data structure that captures the possible motions an object can undergo. To determine what actually occurs requires more information, because of the qualitative nature of the description. In NEWTON the qualitative ambiguities are classified, and special quantitative knowledge is attached to each class of ambiguity so that it can be resolved by algebraic manipulation if relevant to the problem being solved.

1 Matching a description of a program's behaviour to the protocol of a subject is a seductive way of evaluating a program. As deKleer has pointed out, an intelligent problem solver should be able to answer stupid questions as well as hard ones. The representations developed by trying to match program performance to verbal protocols have not as a rule had this property.

2 In response to deKleer's NEWTON, the MECHO group augmented their program to handle the Roller Coaster world. The modified MECHO generated only the part of the envisionment necessary to answer a particular question.

There are three areas in which FROB is more powerful than NEWTON. First, the geometric representation in FROB is more sophisticated than that of NEWTON. In the Roller Coaster world all problems are essentially one dimensional. The only metric properties that can be specified in NEWTON's geometry are heights of named points. The Bouncing Ball world is inherently two *dimensional, requiring a more robust geometric representation.*

Secondly, the envisionment in FROB includes the effects of dissipative forces and deducing the consequences of qualitative assumptions about an object's motion. It is also used to detect the possibility of a collision between two moving objects. None of these issues were addressed in NEWTON. The envisioner in NEWTON was used mainly as a planning device in developing an answer to a posed question by either providing a direct answer or by determining what quantitative information should be used. The envisionment FROB produces could certainly be used in the same way, but this work does not do so.

Lastly, the role of quantitative knowledge in the two programs is quite different. NEWTON used equations to derive numeric and symbolic values for sought after quantities. FROB uses equations as a means of simulation. Despite these differences they can be compared. NEWTON's representation can be thought of as dynamically building the subset of the constraint network representation used in FROB for a situation that will provide a value relevant to some goal. This makes redundant forms of an equation troublesome. In NEWTON, having the polar form of a circle and the cartesian form would lead to substitutions deriving $0=0$. On the other hand, redundancy in FROB's constraints allows as much as logically possible to be computed from information given. Redundant computations which yield different results indicate an inconsistency in the assumptions underlying the anaysis. Constraint networks like those used in FROB could be used to compute symbolic values and drive algebraic manipulation systems (see [Sussman and Stallman], [Steele and Sussman]), if care is taken with the information gleaned from the diagram. This issue is discussed in section 3.6.

The notion of constraint networks and dependencies used in FROB originated with Sussman's Engineering Problem Solving group at the MIT Artificial Intelligence Laboratory. The constraint language used in much of FROB was designed by Steele and Sussman[Steele and Sussman] to explain in a more general fashion the ideas developed by the group's work in electronics. A circuit is described by replacing its components with constraint bodies that use cells that hold values for properties of the part, and rules to enforce the electrical relationships between these cells. When some value is known, the rules fire to compute other values. This process continues until either all cells are filled or no more can be deduced. There are several advantages to this reasoning technique. The number of possible deductions is bounded by the size of the network, and having a vocabulary for the domain enables the class of problems which may be solved to be stated more clearly. Also, it is easy to keep track of what rules set which cells, which makes finding out what a value depends upon simple. FROB is an example of how these ideas can be useful outside of electronics[1]

The recent interest in making programs keep track of the assumptions underlying their deductions [Doyle][McAllester] led to the consideration of how qualitative assumptions about motion should be assimilated. This added discipline has proven quite fruitful in the development of the qualitative representation.

1.6 Overview of this report

For the reader who wishes to see the program in action, an annotated session with FROB is presented in Appendix 1. All dialog in this session was generated by the program without patching or other assistance. Many of the illustrations in this report are direct output from versions of the program.

Chapter two discusses geometric issues. The Metric Diagram is defined and compared with other geometric representations. The interpretation of diagram elements as objects in the Bouncing Ball world is discussed, as well as the geometric analysis which breaks up the space represented by the diagram.

The application of qualitative and geometric knowledge to quantitative descriptions of motion is examined in Chapter three. A constraint representation of the objects in the Bouncing Ball world is described, along with using networks constructed from them for simulation and analysis of motion.

A qualitative description of motion is the subject of Chapter four. The considerations involved in defining the Space Graph, the basic qualitative description of space for the Bouncing Ball world, are discussed. A notion of a qualitative state for a ball is introduced, along with the rules of qualitative simulation necessary for envisionment. Using assumptions about motion to prune the results of envisionment is also discussed.

Chapter five describes how these representations are used to answer questions about motion. Describing some actual motion as a path of qualitative states is shown to be useful in checking the consistency of assumptions about motion. The processes of summarizing the motion of a ball and the detection of collisions are explained, which completes the basic set of questions about the domain.

Chapter six contains a discussion of the psychological implications of this work and some suggestions for future work.

A brief exposition of the constraint language used in this program appears as Appendix 2. The modifications made for this program and a critique of its usefulness is included.

---

1 The original metaphor of Propagation via Constraints was developed out of experience with a program called EL [Sussman & Stallman] This program contained a number of seminal ideas, including a "fact garbage collector" that led to the development of various dependency systems [Doyle][London][McAllester] Mason [Mason, S.B. thesis] used a version of EL to model a hog farm. Hillis [personal communication] used the algebraic manipulation capabilities of EL in developing quantitative models for spring systems Shrobe is currently using a constraint language of his own for integrated circuit design, and the MIT-AI VLSI effort (and particularly Steele) are pushing forward with new ideas about the use of constraint languages in design.

2. Geometric Considerations

2.1 The Role of Diagrams

When dealing with motion problems, people usually draw a diagram. Since many physical constraints have a clear expression in geometric terms, a diagram serves as an organizing tool that makes their spatial arrangements explicit. For example, suppose we wished to know whether the ball in Figure 6 will ever be to the right of both walls. It cannot if the dotted line represents the maximum height the ball can reach, for it will not get past the fir t wall. Using this diagram we can "see" the effects of the relevant relationships. Compare the ease of it use to the process of coming to the same conclusion using the assertions in Figure 7, which contains the information pertinent to this question.

People find diagrams useful b cause they allow some class of spatial questions to be decided by interpreting information gleaned by perception. A problem that can be mapped into a spatial representation might be solved with the aid of visual processing, instead of more linguistic methods of inference. This is advantageous because formal reasoning seems to require conscious effort, where perception does not.

My program incorporates a geometry representation that provides some of the computational advantages that people gain by using a diagram. It is not a slavish imitation of the details of human perception and performance. It is designed to make answering the kinds of questions we seem to use a diagram for easy.

The geometry representation is a distinct module since a class of purely geometric questions can be isolated from questions that involve the physical interpretation of a geometric element. For example, whether a line segment represents a trajectory or a surface is irrelevant to the problem of finding its intersection with another line segment. The geometry module answers three types of questions: identity, parity, and intersection.

Identity questions concern the identification of an element in the geometric representation with the physical entities they model. This correspondence must be clearly marked if the geometry module is to be used by other programs. Aside from communication requirements, the physical interpretations can be used to speed up searches involving elements in the diagram. For example, detecting points where a collision between two balls occurs is much faster if the geometry module is used to look for intersection points between the trajectories, rather than the intersection of one trajectory with all other elements in the diagram.

The most common questions about spatial relationships between elements are parity questions. A geometric entity implicitly divides space into distinct parts, which we shall call its sides. The geometry module can determine on what side of an element some point is, and to tell if some element is on a particular side of some other element. Detecting the inappropriate placement of a moving object inside a solid, for example, requires the ability to determine if the point representing the object is inside the geometric region that represents the solid.

Although they could be considered subsumed by parity questions, intersection questions merit

Fig. 6. Can the ball ever be to the right of the far wall?
If the dotted line represents the maximum height it can reach, the answer is no. The ball cannot get to the right of the first wall, let alone the second. People have no trouble coming to this conclusion given this diagram.

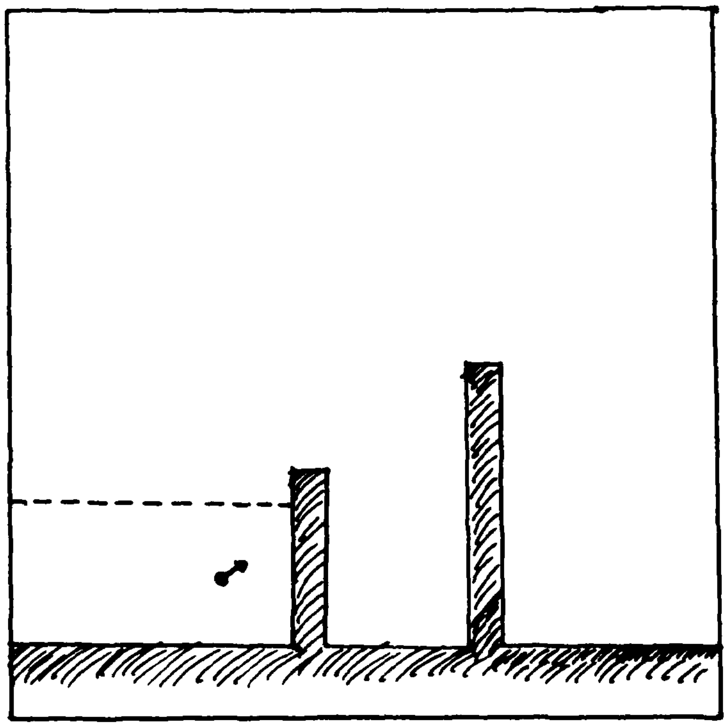


Fig. 7. Can the ball ever be to the right of wall2
These assertions contain enough information to deduce the correct answer, but are much harder for people to use.

```
(POSITION WALL1 -1.0)
(POSITION WALL2 4.0)
(POSITION BALL (-3.0 2.0))
(MAX-HEIGHT BALL 4.0)
(HEIGHT WALL1 5.0)
(HEIGHT WALL2 8.0)
```

seperate consideration. They are important because interacting physical constrai its are usually reflected in the diagram by things that touch.

A drawback to using diagrams is that to create one all metric properties of the things in it must be specified. Arbitrary choices for properties not specified by the problem can be misleading. A classic example is the argument that all triangles are isosceles, which uses the equality of line segments constructed in a triangle that has two equal sides. The problem of generating a diagram from a symbolic description of a scene will not be addressed here. The artful choice of unspecified parameters can be quite difficult [Boberg 72], and until a target representation is known and demonstrated to be useful such an enterprise is risky.

2.2 The Metric Diagram

The marks that represent the geometric aspects of a problem in a diagram have a fixed location and size. Their arrangement on paper models the spatial relations between the things they represent. This property allows our visual apparatus to interpret these relationships as we would those of the real objects. We do not yet understand the complexities of human vision, but there are other ways to encode the spatial structure of a diagram for use by a program. One way is to model the elements of a diagram by analytic geometry. We define a <u>Metric Diagram</u> as a geometry representation comprised of a vocabulary of symbolic elements drawn from analytic geometry, whose parameters are numerical and embedded in a bounded global coordinate system.

By restricting parameters to numerical values, the programs of the geometry module need not perform algebraic manipulation or construct proofs. Calculation suffices to answer geometric questions. Having a global coordinate system insures that all geometric elements are comparable. The coordinate system is bounded to avoid the need for explicit inequalities to divide space into regions.

We will now examine how the Metric Diagram representation can be used to answer the three questions raised previously. This will include describing the properties of the Metric Diagram implemented in FROB for the Bouncing Ball world.

<u>Identity</u>

An element in a Metric Diagram is a symbolic object drawn from a vocabulary of types. The programs in the geometry module must be able to process elements of these types, and the programs that use the module must know the parameters needed to specify them. The types of elements implemented for the Bouncing Ball world are points, line segments, regions bounded by line segments, and vertically oriented parabolas. The properties associated with each type are illustrated in Figure 8. Redundant information is stored to speed computation. The symbolic name can be used by an external representation as a reference, and it can be annotated to mark the domain object it corresponds to.

Parity

A geometric element provides a way to distinguish one part of space from another. In the simplest case, a point, another element can either be at the same coordinates or not. The divisions of space imposed by an element in the Metric Diagram will be called its sides. The sides defined for elements in the Metric Diagram are illustrated in Figure 9. The labels correspond to the answer produced by the geometry module for a point in that region of space. For a point only ON and OFF are defined. Four sides are defined for segments and parabolas to express their limited spatial extent. A region is called closed if a point on its boundary is considered to be inside the region, and open otherwise (these terms are used in the same sense as open and closed in Real Analysis).

The two kinds of parity questions the Metric Diagram answers are

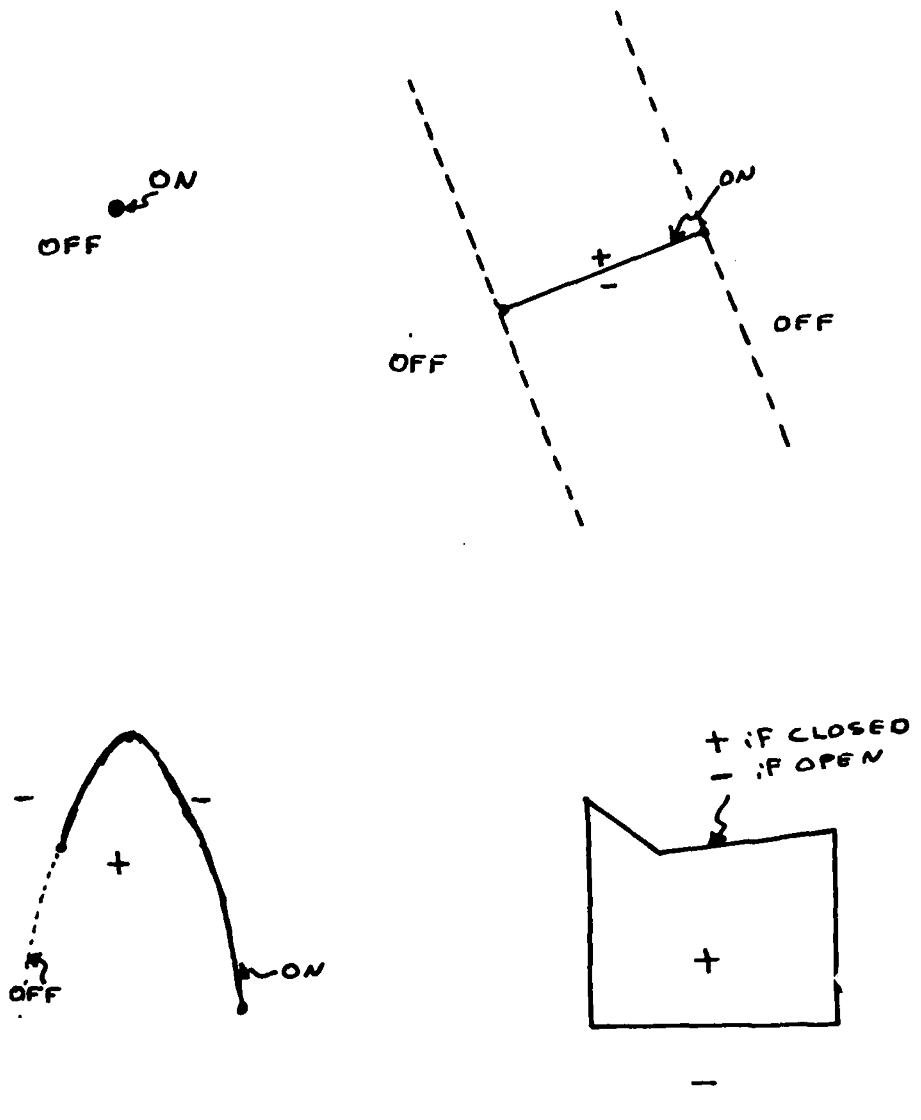
P1: On which side of element A is point B?

P2: What elements are on the S side of element A?

Fig. 8. Properties associated with Metric Diagram elements

```
        POINT                    SEGMENT
        TYPE: POINT              TYPE: SEGMENT
        X: <number>              END1: <point>
        Y: <number>              END2: <point>
                                 EQUATION: (<sin(th)> <cos(th)> <rho>)
                                 UNIT-VECTOR: (<number> <number>)
                                 UNIT-NORMAL: (<number> <number>)
                                 PERPENDICULAR-EQUATIONS: (<equations>)


        PARABOLA                 REGION
        TYPE: PARABOLA           TYPE: REGION
        VERTEX: <point>         BOUNDARY: (<segment> <segment>...)
        END1: <point>            CORNERS: (<point> <point> ...)
        END2: <point>
        EQUATION:
          (<Vertex x> <Vertex y> <p>)

Other Pointers are

PART-OF: ((<prop> . <element>) ...)
        means that the element is the <prop> of <element>

INTERPRETATION: <constraint>
        means that the element is the value of <constraint>
```

Fig. 9. Sides defined by Metric Diagram elements
The four labels ON, OFF, +, and - denote the divisions imposed on space by a particular type of element.



Questions of type P2 require search through an index of elements. This search can be very fast if elements are indexed by their interpretation.

Intersections

Two elements intersect when there is at least one point such that the answer to P1 for this point and each element is ON. The intersection procedure for two elements uses the equations associated with them to calculate possible points of intersection. These candidates are filtered to take the finite spatial extent of the elements into account.

2.3 Interpretations and Geometric Analysis

This section describes how the entities of the physical model of the Bouncing Ball world are related to the entities of the geometric model of the Bouncing Ball world, and how the Metric Diagram is used to enforce the geometric constraints imposed by the physical constraints.

The mapping between domain objects and Metric Diagram elements is simple:

Ball -> Point
Surface -> Segment
Trajectory -> Segment, Parabola

Surfaces are immobile. Either the + or - side of the geometric element representing the surface is designated as its SOLID side. All points not ON a surface are considered to be either in a SOLID region or part of FREE space. The portion of the space represented by the diagram that is FREE is assumed to be connected. Surfaces that do not form a connected boundary around some portion of the diagram are continued along part of the diagram's border to do so. This defines the set of SOLID regions implied by the surfaces. The SOLID regions for typical diagrams are illustrated in Figure 10. To be in FREE space, a point must be inside the diagram and not inside any solid region. The SOLID regions are considered open, in the sense defined above, to make being ON a surface distinct from being inside the region.

The qualitatively distinct parts of FREE space must also be determined, to serve as a set of PLACEs for a qualitative description of motion. Because all balls have the same spatial properties (i.e., zero extent) the only source of geometric constraint on places are the configuration of the surfaces. This means the same set of places is relevant for each ball and can be computed as soon as the surfaces are known.

The basic set of places for a scene is represented in a data structure called the Space Graph. The nodes in this graph are chunks of free space and the segments that bound them, and the arcs are labelled with directions and express the adjacency relationships of the nodes. Figure 11 illustrates the geometric elements that comprise the graph for a typical scene and a schematic of the pointer structure. The considerations that define the elements in this graph are detailed in Section 4.2.

The computation of the Space Graph using the Metric Diagram is simple. It requires slicing free space with vertical and horizontal lines from the endpoints of surfaces and collecting the regions that

Fig. 10. Solid regions of typical diagrams

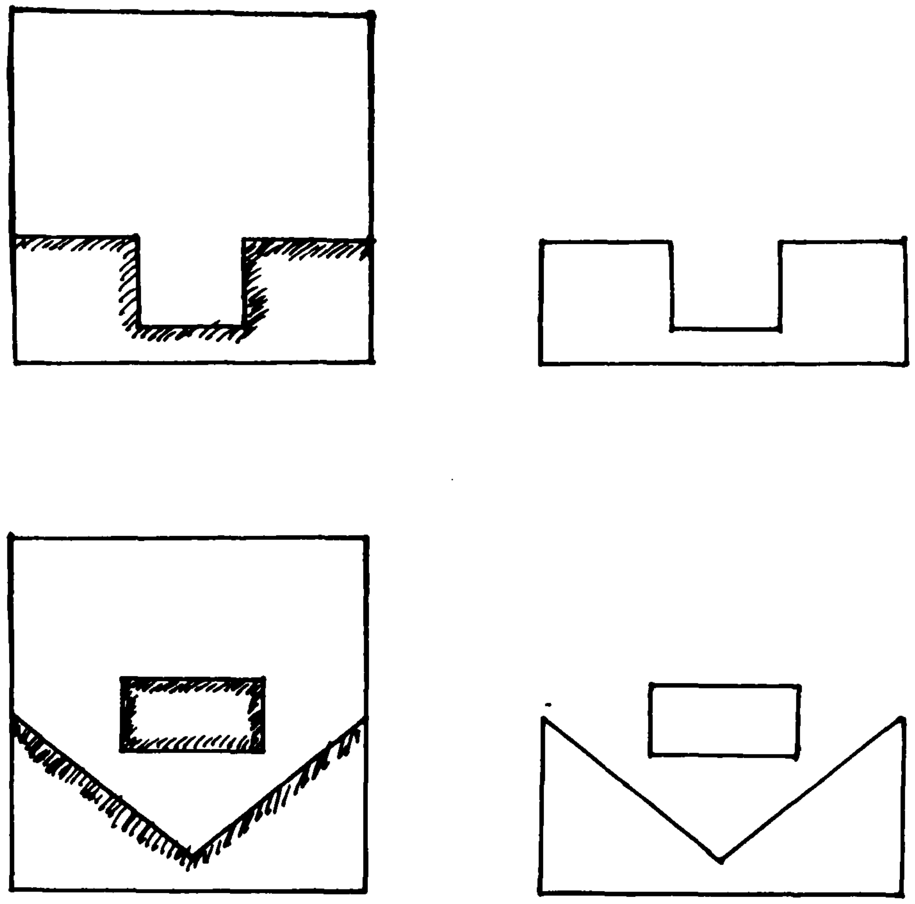
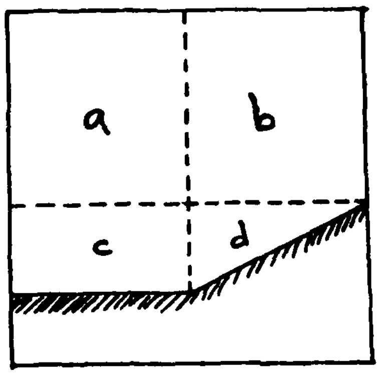
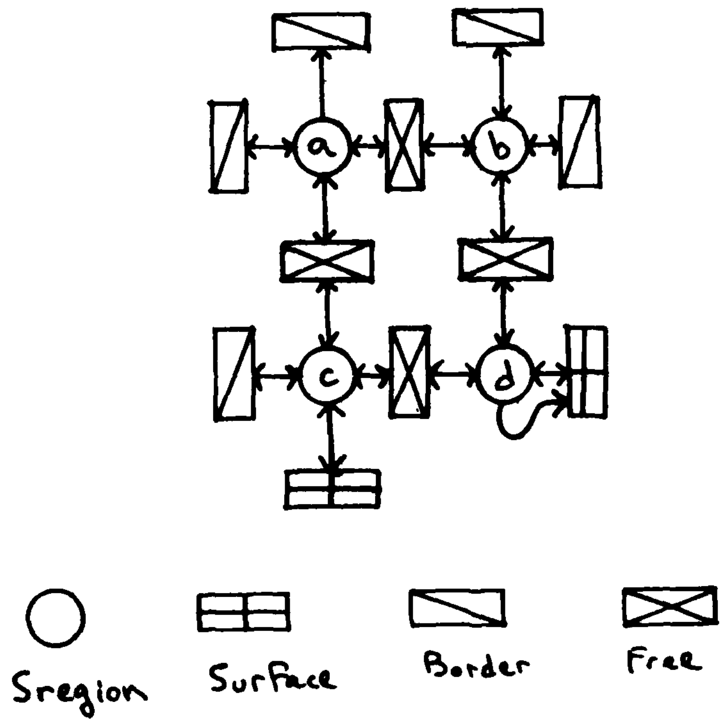


Fig. 11. Schematic of Space Graph for a typical diagram





result. Since the Space Graph elements are specified in terms of Metric Diagram elements (the nodes of the graph are either regions or segments), the Metric Diagram serves as a bridge between the qualitative and quantitative descriptions of motion.

2.4 Other Geometry Representations

To gain perspective on the Metric Diagram, this section compares it with other representations of geometry that have been used in Artificial Intelligence.

The most popular class of geometry representations in AI are the relatio ial representations. An object or place is given a symbolic name, and its shape, location, and extent are specified by predicates on them (such as IS-CIRCLE) and relations between these elements, such as LEFT-OF, ABOVE, and INSIDE. Problem solvers in the Blocks world often use relational representations [Winston 70].

Reasoning about space in a purely relational system can be difficult. Transitive axioms, such as

```
(implies (and (left-of X Y) (left-of Z X)) (left-of Z Y)))
```

are often needed answer parity questions. As [Waltz and Bogess] point out, reasoning with such axioms can lead to combinatorial explosions. Lacking enough information to prove or disprove a relationship can lead to creating subgoals involving every known element in the diagram. It is also difficult to infer any intersections of elements that are not explicitly given.

A more interesting structure for a relational system was used in TOPLE [McDermott 74]. Space is represented by a tree of "places", each of which is the space filled by an object or a part of an object. These place definitions are adequate for some discussion of physical objects but too limited to deal with motion.

Pure relational systems are very weak models of space. [Hayes a] notes that the axioms for the geometry of blocks in many problem solvers can be satisfied by modelling a block as an ordered pair of integers, one component for the number of blocks below it, and one number for discrete locations on the table. This is far from the intuitive notions of space they are intended to capture. The weakness of these systems is also the source of their generality. For a fixed vocabulary of predicates and relations there is only one full relational description (all possible relations and predicates are asserted) up to isomorphism between object names for any Metric Diagram, but for a relational description there can be infinitely many Metric Diagrams. Since the relational descriptions given with a problem are often incomplete, generating a Metric Diagram from a relational description would involve filling out the relational description and then finding numeric parameters that satisfy this description. The fact that people are willing to go through this trouble in generating pencil and paper diagrams seems to indicate that our fluency in dealing with space does not come from a set of very clever axioms for reasoning with a relational description.

Another class of spatial representations uses a regular array of cells as an analog of physical space. An object's location and extent is represented by the set of cells in the array containing the symbol that corresponds to that object. WHISPER [Funt 76] and a program by Kosslyn and Shwartz [Kosslyn and Shwartz] which simulates phenomena associated with mental imagery both use arrays.

Both systems use a simple local process, motivated by early theories about the role of the retina in perception, to compute with the array. The arg[illegible]nt is that spatial reasoning is like perception, so a representation of space should reflect the struc[illegible] the perceptual system, including a "retina" that "fixates" on parts of the array to denote attenti[illegible]ontrary to the assumptions in these papers, the available evidence about retinal function does <u>not</u> suggest that its purpose is to scale, translate, and rotate the projections of objects on the visual field. Such a process would only make sense after processing has been done to seperate portions of the visual field. This would imply some sort of processing before the retina that segmented the visual field.

An array based scheme supported by parallel hardware might have some useful properties. One such scheme would model space as an array of cells, each of which is a processor capable of storing a small number of marker bits as in NETL [Fahlman 79]. The contents of a cell are defined to be the bits that are turned on, and instantiating an object in the array corresponds to selecting a marker bit to represent it, and turning on that bit in the cells in the array where it is supposed to be. Finding out what objects are in a particular region of space could be done in constant time by asking the cells in that region of space for their contents. Intersection of two objects could also be performed in constant time by asking if any cells contain a marker for both objects. The regions of space needed to answer parity questions could be determined by propagating marker bits through the array.

If the array is not composed of parallel processors the situation is very different. Using a "retina" as the processing element means that most operations become searches. The time for a search would depend on the size of the array, rather than the number of elements in the diagram as it does when a search involves the Metric Diagram.

Whether parallel or not, an array system needs an external representation to write elements into it and to recover from the degrading effects of rotation on a discrete grid. Placing an object into an array involves setting up parameters for its location, scale, and rotation and then turning on the correct cells of the array. The instantiation of a Metric Diagram element involves only the first part of this process. If the same questions can be answered by each, the array seems superfluous. [Hinton 79] argues against the use of array based representations in mental imagery on the same grounds.

Array based representations using parallel hardware certainly deserve some study. But on the whole the case for array based representations is not yet convincing.

Analytic geometry was used to model a diagram in a program by Gelernter [Gelernter 63] that proved geometry theorems. The diagram functioned as a source of counterexamples. An assertion made as a subgoal was checked to see if it was true in the diagram. If the assertion was false the subgoal was abandoned since to be true it must hold in all diagrams that satisfy the premises of the theorem. Although the diagram in FROB is used very differently (FROB has no goals, so the notion of a subgoal filter is meaningless), Gelernter's program helped motivate the use of a quantitative geometry representation.

Robotics programs make heavy use of analytic geometry. Approximating physical objects with polyhedra has proven useful for systems that plan mechanical assemblies [Lozano-Perez 76]. [Fahlman 73] planned construction tasks using a numerical model of blocks, including a model of sliding friction.

The spatial modelling done in robotics programs has been directed towards detailed shape description and planning paths rather than reasoning about motion. However, the growing literature of computational geometry inspired by robotics applications has been a source of algorithms for the Metric Diagram implementation.

Shape description often involves numerical parameters attached to symbolic structures. [Hollerbach 75] examined the use of generalized cycinders both to recognize Greek vases and to interpret scenes of blocks. Generalized cylinders are also the primitives of the Spasar theory of 3D shape recognition [Marr and Nisihara]. In this theory a shape is recognized by finding a set of numbers for the *lengths and relative orientations in space of axes that are derived from an image* in order to match the object with a model from a catalog of shapes. Both works suggest that the ideas underlying the Metric Diagram may play an important role in more complex forms of spatial reasoning.

3. Quantitative Representation of Motion

The motion of an object can be described by specifying its coordinates in a frame of reference for each instant in time. In some simple cases the evolution of its state parameters with time can be described by equations. Using equations to compute a description of motion involves both qualitative and geometric knowledge. Qualitative knowledge is needed to identify what set of equations are applicable. For example, knowing that a ball is in free space is necessary to apply the equations for free fall under gravity, as opposed to equations for sliding on a surface. Geometric knowledge is required to identify boundary conditions. The equations of motion may be able to specify how fast a falling object will hit the ground, but they do not a priori determine just where the ground is.

In this chapter we explore the role of qualitative and geometric knowledge in quantitative descriptions of motion. First we present the equations relevant to the Bouncing Ball world. We discuss the decomposition of motion using a catalog of qualitatively distinct types of motion, the Action Sequence. We describe the implementation of such a description as a network of constraints after a brief overview of the implementation language. The application of this representation to the simulation and analysis of motion is demonstrated, and some potentials and pitfalls are discussed.

3.1 Equations of motion

The equations of motion which pertain to the bouncing ball world are those of projectile motion in a vacuum. They are listed in figure 12.

The amount of energy retained by a ball in a collision is represented by its coefficient of

---

Fig. 12. Equations of motion for the Bouncing Ball world
The equations of motion for the Bouncing Ball world are those of projectile motion in a vacuum.

For FLY,

$$X_f = X_0 + V_x * \Delta t$$
$$t_f = t_0 + \Delta t$$
$$Y_f = Y_0 + V_{y_0} \Delta t + \frac{g*(\Delta t)^2}{2}$$
$$V_{y_f} = V_{y_0} + g*\Delta t$$

For COLLIDE,

$$V_{\parallel_{in}} = V_{\parallel_{out}}$$
$$V_{\perp_{out}} = COR*V_{\perp_{in}}$$

restitution, abbreviated COR. When the COR is 1 the ball is perfectly elastic, and when it is 0 the ball is completely inelastic. This model of dissipation is not very precise since in real situations the amount of energy lost often depends on the speed of the ball and the type of surface it collides with. Surfaces in the Bouncing Ball world are considered uniformly impenetrable, so that only the COR of a ball determines the amount of energy it loses. Collisions are assumed to happen instantaneously.

I have not found a general model that describes how energy is lost when two moving balls collide. The physics textbooks I have examined discuss what happens when at least one ball is completely inelastic or when both are perfectly elastic, but are silent about all other cases. This is not surprising due to the gross nature of the approximation involved in letting the COR be a constant. One model that fits the extreme conditions is to assign the product of the individual CORs to both balls, but this is ad hoc. Motion past a collision between two balls is not described by this program.

3.2 Breaking up motion

Let us consider using equations to describe the motion of a ball. The state parameters of a ball are its position and velocity. From its initial state the relevant set of equations can be identified by qualitative knowledge, along with the method of determining the boundary conditions. Since the equations are continuous, the state parameters of the ball will vary smoothly until the boundary conditions are met. The value of the state parameters at the boundary can be computed using the equations. The set of equations and boundary detection method associated with this new state can be chosen by again identifying the type of motion, and so forth. Each application of the equations represents some portion of time for the ball (even if only an instant, as in a collision), and these applications are linked by descriptions of the state of the ball. Each such application will be called an act.

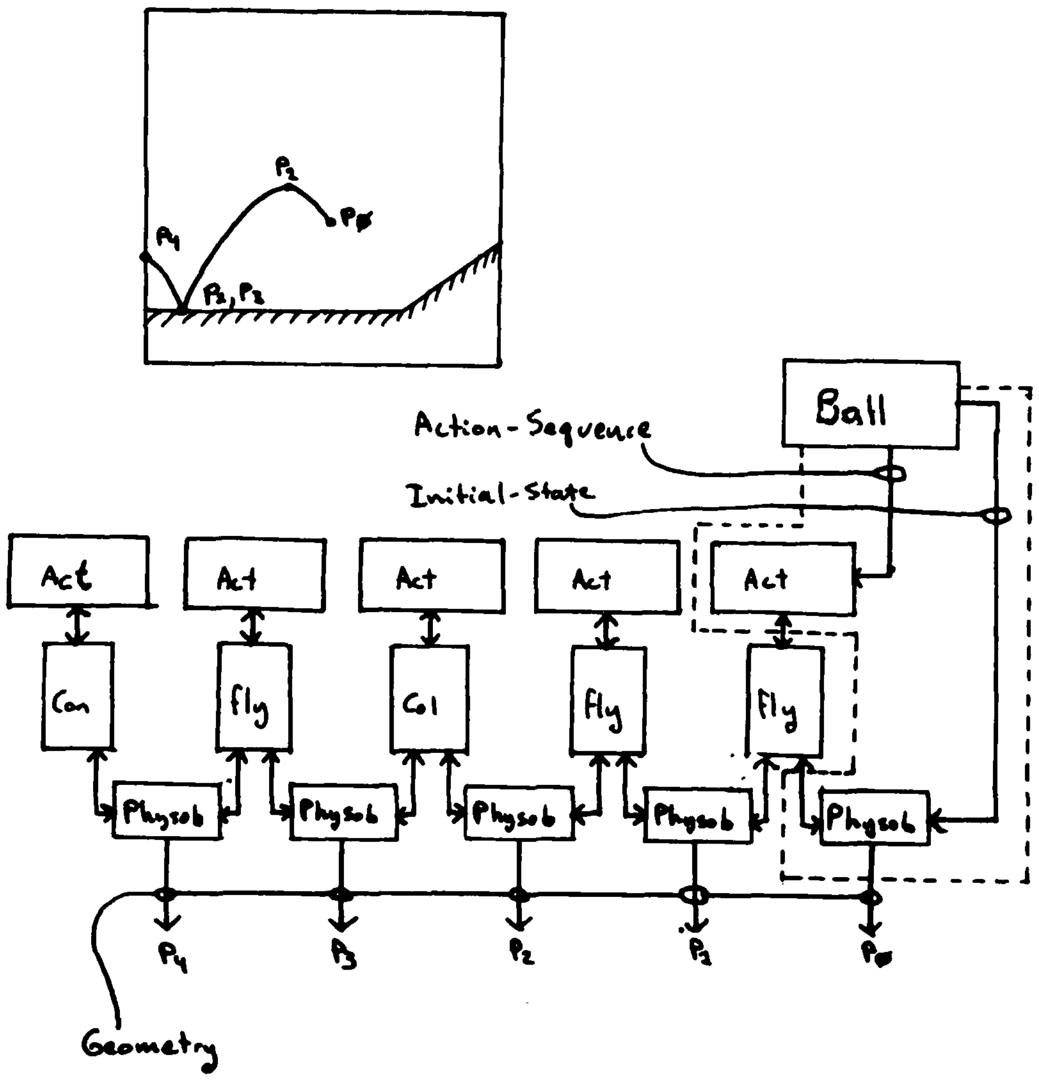
The two types of motion defined by equations in the Bouncing Ball world are FLY and COLLIDE. The determination of boundary conditions is different for flying up as opposed to flying down. In both cases collision with a surface provides a boundary, but a ball that is rising can also begin to fall because of gravity. Therefore flying up and flying down are considered as distinct acts.

Qualitative knowledge must also identify those situations where the equations of motion are no longer applicable. For the purpose of description these situations can be considered as special types of motion. Boundary conditions cannot be computed once a ball leaves the diagram. Leaving a diagram will be called CONTINUE. STOP denotes the cessation of all motion. SLIDE/STOP and SLIDE/STOP/FALL will indicate that the motion is along some kind of surface, and thus is outside the competence of the program.

By linking representations of these motion types with descriptions of the ball's state in between them, any motion in the Bouncing Ball domain can be described. We will call this form of description an Action Sequence. Figure 13 shows the Action Sequence description for a typical motion in schematic form.

The Action Sequence description can be used for the analysis of motion. Just as an electrical circuit can be described by building a network of computational objects that describe its components and

Fig. 13. Decomposition of a typical motion into an Action Sequence
The trajectory shown in the Metric Diagram is the geometric aspect of the Action Sequence whose schema is exhibited below.



the connections between them, the motion of a ball can be described by building a network of objects that describe the kinds of motion it undergoes and its state at various times in that motion.

Given numerical values for some initial state and the ability to use a diagram, the Action Sequence can also be used to build a description by simulation. This process and the description it produces is more perspicuous than the description produced by the traditional incremental time evolution of the differential equations of motion. Incremental time simulations use the equations of motion to determine how far a ball will move during interval of time, and iterate this process to produce a list consisting of the state parameters of the ball at discrete instants of time. The size of the incremental time description (and the time to compute it) depend on the interval size used. By contrast, the size of the Action Sequence description is proportional to the qualitative complexity of the motion. Using a larger interval size in the incremental time simulation decreases the size of the description, but increases the error because the boundary conditions are less likely to occur near the end of an interval. The Action Sequence representation avoids this problem by explicitly computing the boundary conditions instead of searching for them.

In FROB the Action Sequence is implemented by describing the vocabulary of motion in a constraint language. The next section provides a brief introduction to constraint languages as a prelude the discussion of the actual representations used.

3.3 A Constraint Language Note

Balls, the types of motion, and other parts of the Action Sequence are represented in FROB as constraint objects. The constraint language used is a variant of CONLAN [Steele and Sussman]. Specifying a constraint is similar to declaring the relationships between the properties of what that constraint represents, as opposed to writing a procedure that computes a fixed set of output values given a fixed set of inputs.

A constraint object can have parts, which are either cells or other constraint objects. The role of a cell is to hold values that correspond to the properties of what the constraint object represents, such as the X coordinate of a ball at some particular instant of time. Some properties may best be expressed by other constraint bodies. An example is the velocity of a ball at some instant, which is represented in the constraint language by a vector constraint object that is a part of the constraint object describing the state of a ball.

Computations must be specified to enforce the relationships between the parts of a constraint. Rules are attached to a constraint object to perform this function. A rule has a fixed set of cells that it can use to compute a value, and usually has a fixed cell that its result is to be stored in. When all the cells used by a rule are known, the rule is queued and eventually run. A rule will dismiss itself if it decides it cannot yield a value, and signal a complaint if it detects an inconsistency. Rules that never set a cell perform monitoring or bookkeeping tasks. One such monitor rule enforces the requirement that a ball cannot be inside a solid part of space.

The source of a value is always marked on a cell. This can be either a rule that sets the cell, or an assumption made by the person or program using the contraint interpreter. If conflicting values are discovered for some cell, as determined by a very simple matching process, a contradiction is signalled and the assumptions underlying it are offered to the user for correction.

When a rule sets a cell, it may enable other rules to fire. This process can continue until either all cells are filled, the queue of rules to run is empty, or a contradiction is signalled. The consequences of making an assumption are reflected by the newly set values in the constraint network. The reasons for each value can be examined, making this process a powerful tool for analysis.

The major differences in the version of CONLAN used in FROB stem from the necessity of using external representations (the Metric Diagram and the qualitative descriptions to be described later) and allowing rules in a constraint network to create additions to the network. Allowing the constraint network to grow makes simulation easy. This process is not without peril. If a ball never stops moving, the network describing its motion would grow arbitrarily large, thus mitigating the advantages of deduction in a network. For this reason simulation is done only by user request. The details of these changes are relegated to Appendix 2.

A schematic notation similar to that of logic circuits is used to illustrate constraints in this report. Cells are denoted by rectangles, and rule are drawn as circles. The direction of arrows that connect rules and cells indicates the flow of information between them. Rules that are executed for effect are denoted by attaching their output to a MONITOR label. Parts of a constraint that are themselves constraint objects are represented by various shapes. These constraint diagrams are only intended to provide an idea of the structure and complexity of the system's knowledge, rather than to reveal the exact details of the flow of computation through them.

3.4 Constraint Representations for the Bouncing Ball World

The Action Sequence description is built out of a vocabulary of computational objects that correspond to types of motion and states of balls. This section describes the actual constraint objects used in FROB that implement this vocabulary. Readers who do not care for implementation details should skip this section. Those who thirst for more should turn to Appendix 1, which illustrates how these constraints are used.

A SCENE constraint object performs the necessary geometric analysis once the surfaces are specified in the Metric Diagram. This includes finding solid regions, computing the Space Graph, and defining other places using the Space Graph. Figure 14 contains a block diagram of the SCENE constraint. The only information associated with a surface in the Bouncing Ball world are its representation in the diagram and which side of that diagram element should be considered SOLID. The SURFACE constraint has a cell for each of these properties and little else.

The state of a ball for a particular instant of time is represented by the PHYSOB constraint body. The block diagram for PHYSOB is contained in Figure 15. If both the X and Y coordinates of a ball are given, a point with those coordinates is placed in the diagram and its name stored in the GEOMETRY cell. Given the GEOMETRY, the diagram is used to determine what surfaces or borders

Fig. 14. The SCENE constraint
Rules attached to cells of the SCENE constraint perform the geometric analysis of the diagram. The "monitor" denotes a rule that is run for effect, in this case to add geometric elements to an index by their interpretations.

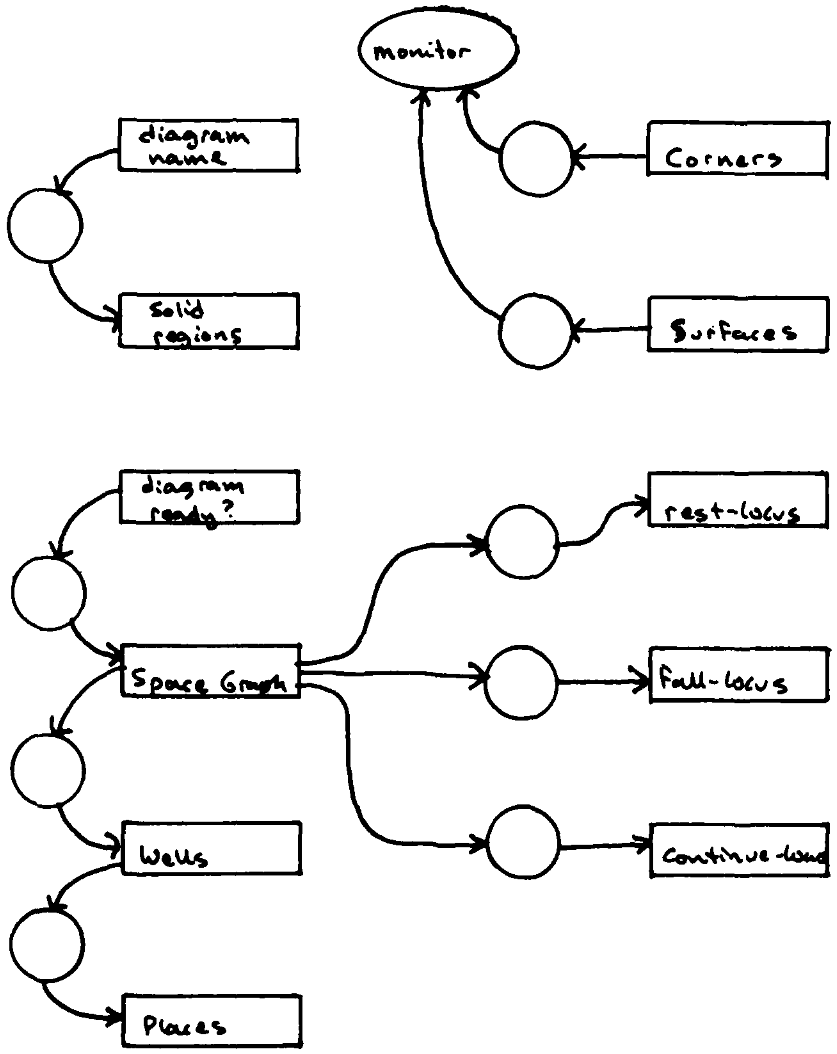
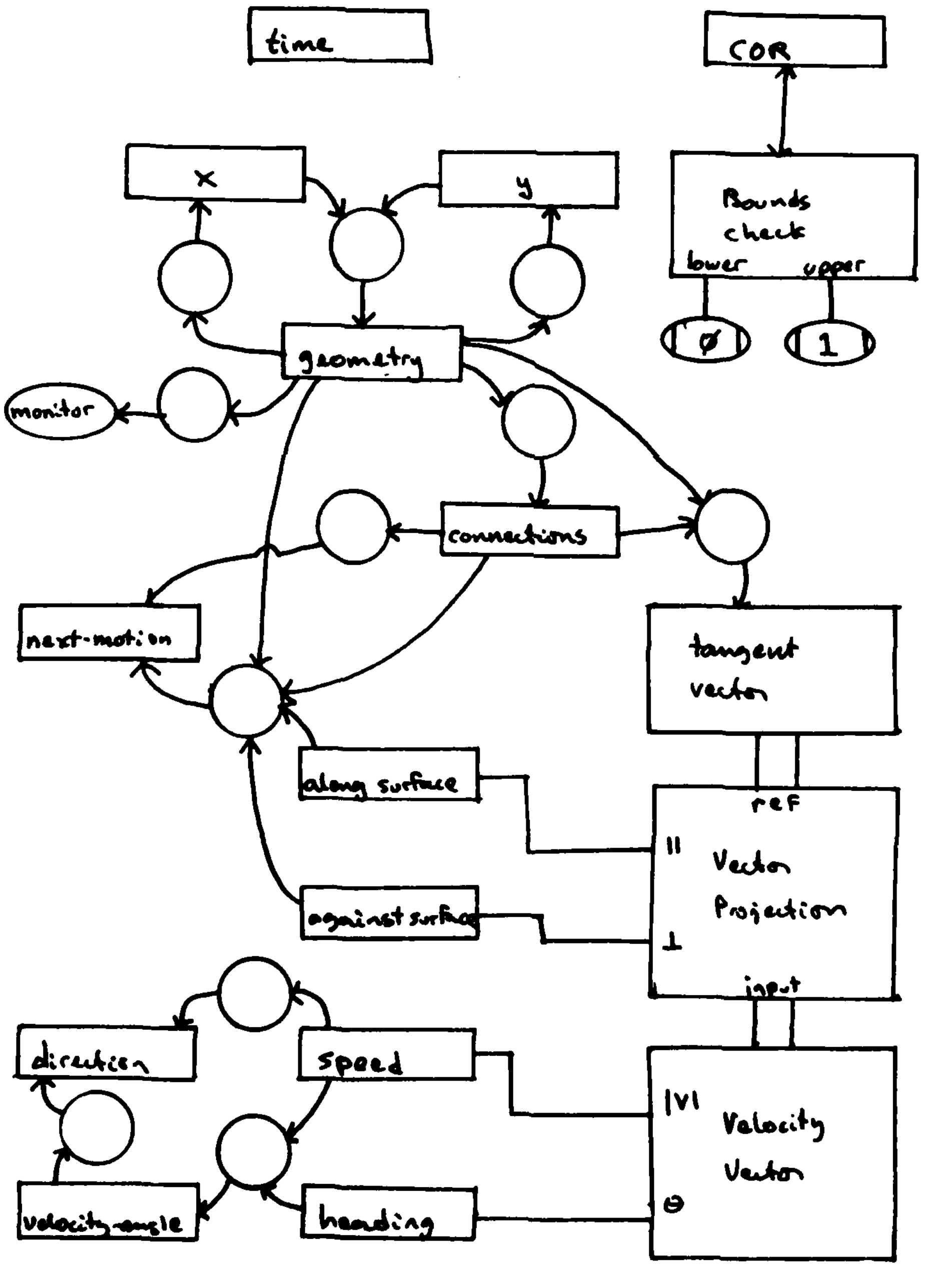


Fig. 15. The PHYSOB constraint
The state of a ball at a particular time is represented by the PHYSOB constraint.
The MAX-HEIGHT cell and associated rule is not shown.



the ball touches and stored as the value of the CONNECTION cell. The GEOMETRY cell is also monitored to insure that the ball is inside free space at the instant of time the constraint represents. DIRECTION is a qualitative description of the heading consisting of a list of directions, such as (RIGHT UP). Having a CONNECTION causes the velocity of the ball to be decomposed into components parallel and perpendicular to the geometric element (either surface or border) it is touching. The MAX-HEIGHT cell computes how high the ball would get if all of its speed was disspated against gravity, to provide a geometric interpretation of an energy constraint.

The type of motion that will occur from a state is stored in the NEXT-MOTION cell. The rule is

```
If no connections, FLY
If connected to a BORDER
      and the direction of velocity is outside, CONTINUE
      else FLY.
Otherwise connection is a surface.
      If the velocity of the ball is into the surface, COLLIDE.
      If there is a significant velocity away from the surface, FLY.
      If surface does not provide support,
              then if there is a significant velocity along the
                      surface
                      then SLIDE/STOP/FALL.
                      Else FLY.
      Otherwise
              if there is a significant velocity along the surface,
                      then SLIDE/STOP
                      Else STOP
```

A qualitative summary of the current state is also computed for the value of the STATE cell. The qualitative state description used will be discussed in Chapter four.

The BALL constraint body serves as an index for the different descriptions of its motion. These include a descripton of the motions possible from the initial state and a qualitative description of the Action Sequence, as well as pointers to the Action Sequence itself. These other descriptions will be discussed in Chapters four and five. The initial state of a ball is represented by a copy of the PHYSOB constraint. Figure 16 illustrates the BALL constraint.

The ACT constraint represents a portion of the ball's motion. PHYSOBs that describe the state of the ball before and after the act are contained in the OBJECT and AFTER cells, and the MOTION cell holds the constraint that describes the motion occuring during this ACT. Simulation is controlled by a cell of the ACT. If that cell is non-zero, the COMPUTE-NEXT-MOTION cell is turned on. Rules attached to the COMPUTE-NEXT-MOTION cell instantiate a PHYSOB constraint to describe the state of the ball after the motion and a constraint describing the appropriate type of motion. Figure 17 shows this part of the ACT constraint. What is not shown are the rules that wire the OBJECT and AFTER to the MOTION when each is known, as well as the bookkeeping associated with decrementing the count that corresponds to the length of simulation. Let it suffice to say that these rules are defined and do the

Fig. 16. The BALL constraint
The BALL constraint provides an index for diverse representations of motion. The INITIAL-STATE is a PHYSOB constraint. The representations corresponding to the other constraints will be discussed later.

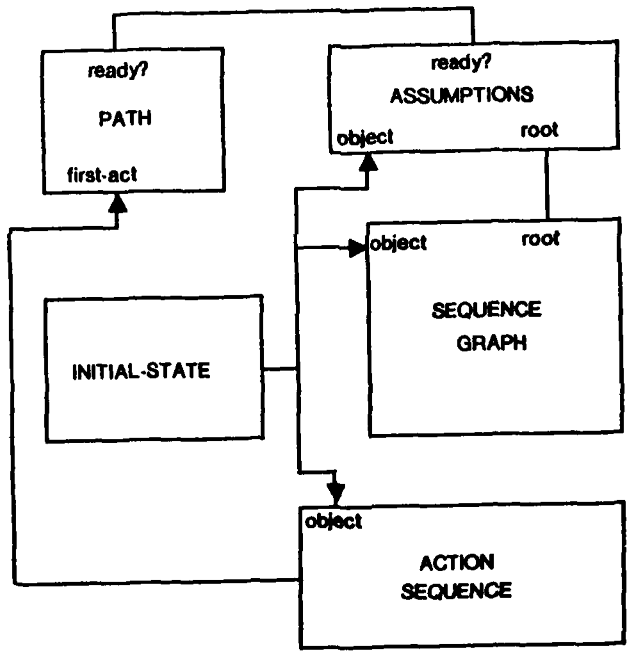
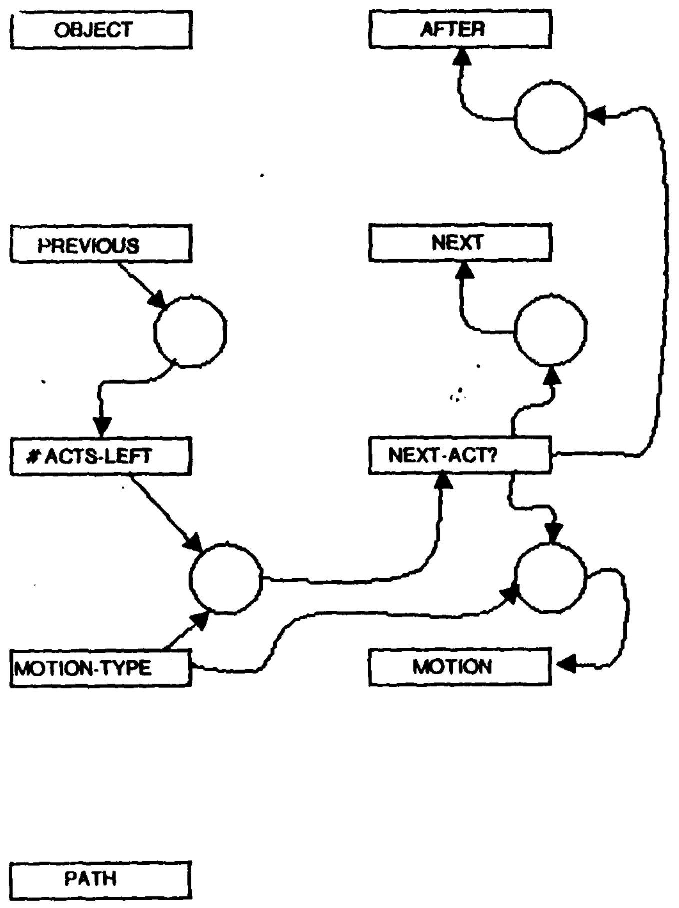


Fig. 17. The ACT constraint
The ACT constraint does much of the bookkeeping associated with the Action Sequence. The wiring rules that actually make the connections between parts as they become known are not shown.



right things.

FLY is the most complex motion in this domain. The block diagram illustrating the FLY constraint appears as Figure 18, with its parts shown in Figures 19, 20, and 21. The ON-PATH constraints are used to express the fact that both endpoints must be on the trajectory followed by the ball. Any two points on the trajectory must satisfy the equations described by FLY-ENERGY, so three copies of the constraint are required. FLY-GEOMETRY provides an interface with the diagram. It computes the Metric Diagram elements associated with the start, vertex, and end coordinates of the motion (or contrawise asserts the coordinates if the Metric Diagram elements are provided) as well as the trajectory. Finding points of intersection of the trajectory with surfaces and borders is done by rules attached to this constraint. In case such intersection points exist, the point closest to the start point is used as a collision point and the COLLISION? cell is turned on. When part of the FLY constraint, the COLLISION? cell set to NIL which denotes being off. This will cause a complaint to occur if a described motion requires that a ball penetrates a surface or flies outside the diagram.

The FLY-SIMULATE constraint contains additional knowledge about discovering boundary conditions. It uses the FLY constraint as a part (although there is a special integrated version for efficiency). The maximum height a ball flying upward can reach is determined by its energy. It will not reach this height if it leaves the diagram or collides with a surface. These collisions are detected by a seperate copy of FLY-GEOMETRY, whose start point is linked to the initial point of the FLY and whose end point is linked to the vertex. If a collision occurs, the point of the collision is used as the end point in the FLY. A ball that is falling will continue to do so until it hits something or leaves the diagram. In this case the end point for the simulation copy of FLY-GEOMETRY is a point on the trajectory which intersects the border of the diagram, and the collision point becomes the end point for the FLY.

A block diagram for the COLLIDE constraint is shown in Figure 22. This constraint merely flips the velocity vector of the ball about the surface normal at the point of contact, suitably modified by its COR.

This completes the basic constraint vocabulary for the Bouncing Ball world. It should be noted at this point that the first two questions for this domain,

1. What can a ball do next?
2. Where can it go next?

can be answered if enough numeric information is provided about the state of a ball.

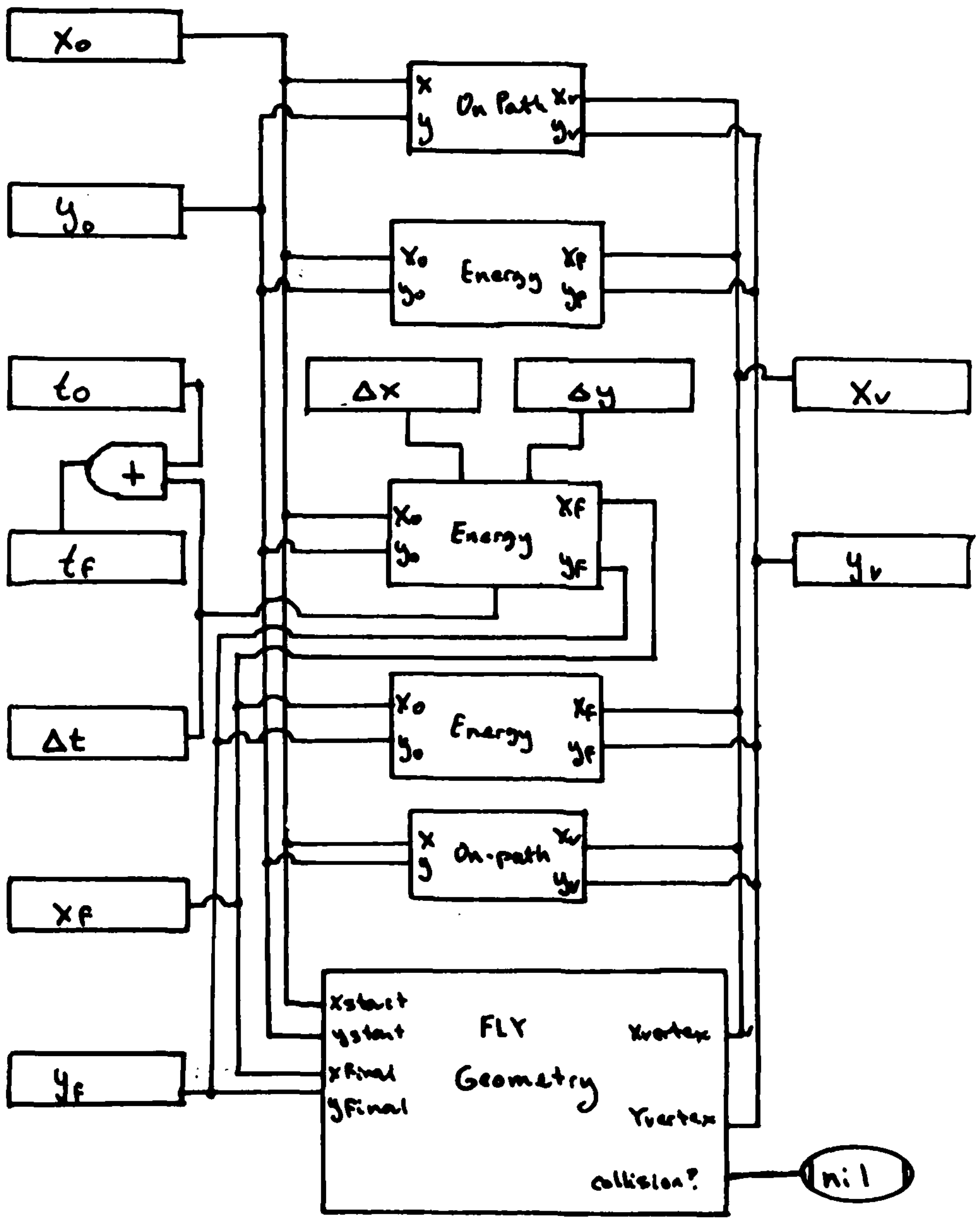
3.5 Examples of Analysis and Simulation

The constraint vocabulary of the previous section can be used to build networks that describe any motion in the Bouncing Ball world. This section demonstrates the application of the Action Sequence representation to analysis and simulation of motion.

Our first example is drawn from Newtonian Mechanics, by A.P. French.

"A perfectly elastic ball is thrown against a house and bounces back over the head of the thrower, as shown in the figure. When it leaves the thrower's hand, the ball is 2 meters above the ground

Fig. 18. The FLY constraint



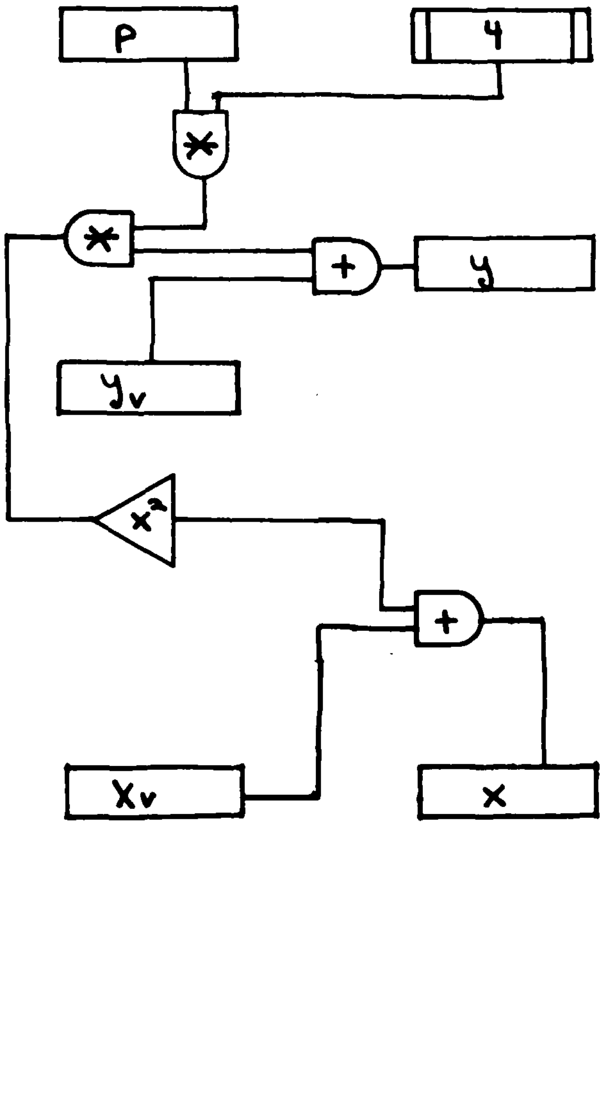


Fig. 19. The ON-PATH constraint
When P=0 the trajectory is a line segment, otherwise it is a parabola.

Fig. 20. The FLY-ENERGY constraint
The positive root is always the correct choice because the constraint is never applied "backwards".

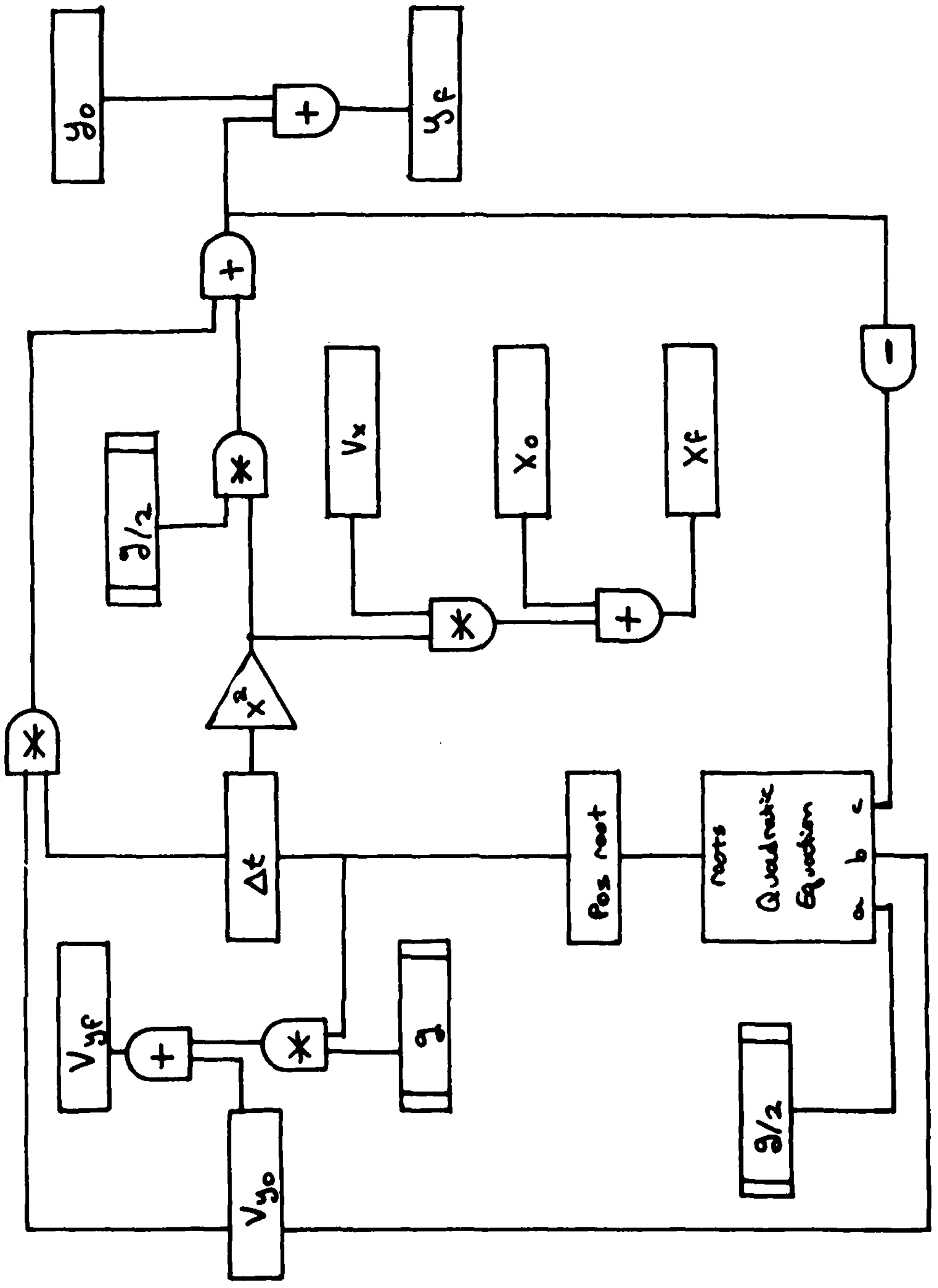


Fig. 21. The FLY-GEOMETRY constraint
Trajectory, Start-Point, End-Point, Vertex-Point, and Collision-Point cells hold Metric Diagram elements.

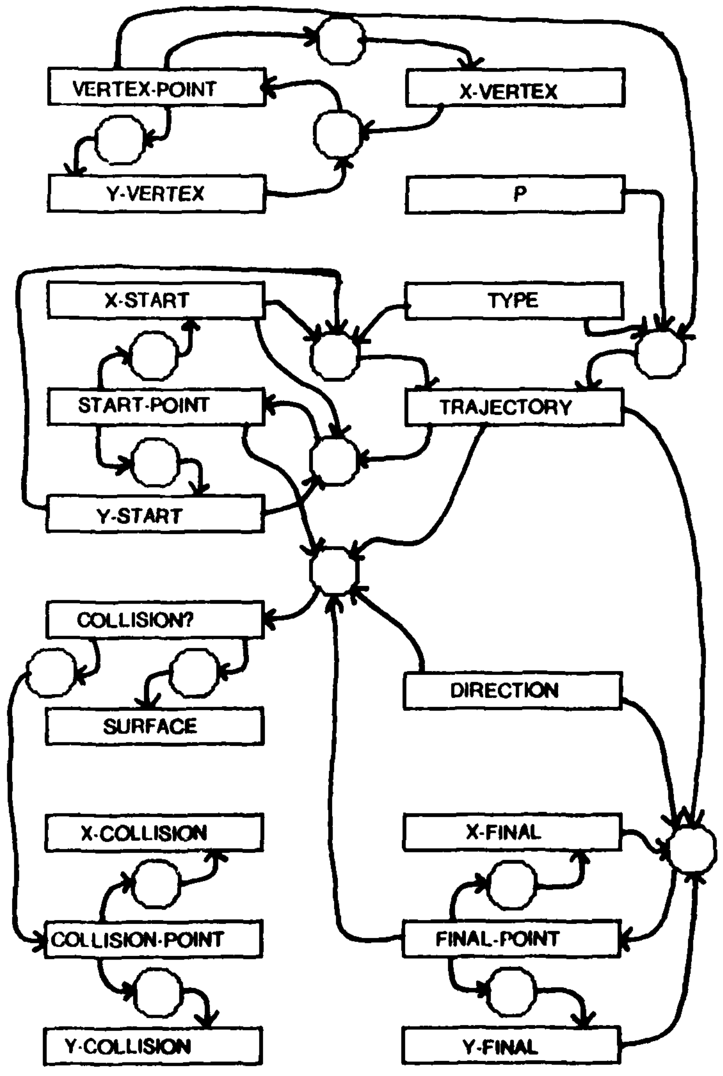
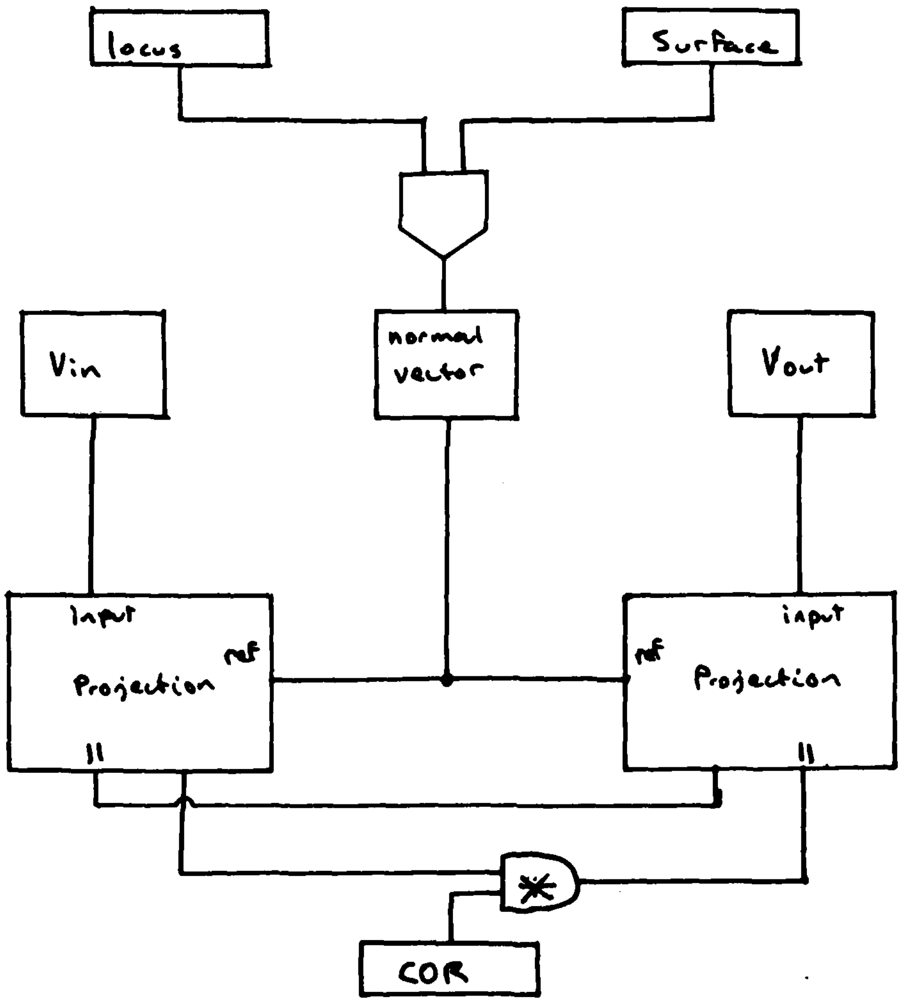


Fig. 22. The COLLIDE constraint
Collisions with a surface are assumed to be instantaneous.



and 4 meters from the wall, and has both its x and y velocities equal to 10 meters/sec. How far behind the thrower does the ball hit the ground?"

Figure 23 contains the diagram given with the problem and the statements used to create the Action Sequence. The Action Sequence for the problem consists of three FLYs and one COLLIDE, and the X coordinate of the initial state can be subtracted from the X component of the final state to yield the desired answer. Figure 24 shows the Metric Diagram and the properties of the initial and final states that yield the answer. Problems that do not require algebra are rare in physics textbooks, because algebra and calculus are the formal techniques that must be practiced to connect them with the common sense understanding the student already has.

Using the diagram enables FROB to detect certain inconsistencies in a description of motion. Figures 25 and 26 show that the program will enforce the semantics of SOLID regions and the impenetrability of surfaces.

Since the program cannot deal directly with the real world, simulation is useful to determine what will actually happen from some state of a ball (assuming the equations of motion are accurate). Some simulations performed by FROB are illustrated in Figures 27 and 28

3.6 Beyond Numbers

The constraints in FROB use the equations of motion in a "cookbook" manner. Placing numbers in the cells of a constraint representation for an equation results in other numbers being computed. This section raises the issues involved in other uses of these equations.

FROB can require far more information than people need to detect an inconsistency. Figure 29 illustrates one such case. A comparison of the heights reached by the ball before and after the collision is sufficient to tell a person that the proposed motion cannot happen, except perhaps under spin. FROB requires a number for the COR so that it can propagate information from the last FLY through the COLLIDE. A specific point must be chosen for the AFTER of the last FLY, along with adequate velocity information. These two pieces of information are required because there is no way for the constraints in the Action Sequence to use a description of the form "and goes at least this high", which is all that is necessary for the comparison to be made.

The problems raised by this example might be solved by adding an additional layer of rules onto the constraints of the domain that dealt with qualitative properties of the numbers. These properties might include simply the signs and relative magnitudes of the quantities compared. For example, the ACT constraint could have a comparison attached to the MAX-HEIGHTs of the states it connects that would signal a contradiction if the MAX-HEIGHT of the later ball was greater than the MAX-HEIGHT of the earlier one. Extra versions of the rules for a constraint could also perform an incremental analysis like that in [deKleer, PhD]. This is illustrated in Figure 30 by an incremental constraint expression of Boyle's law. Neither kind of reasoning has been well thought out for Bouncing Ball problems so these are no more than tenative suggestions.

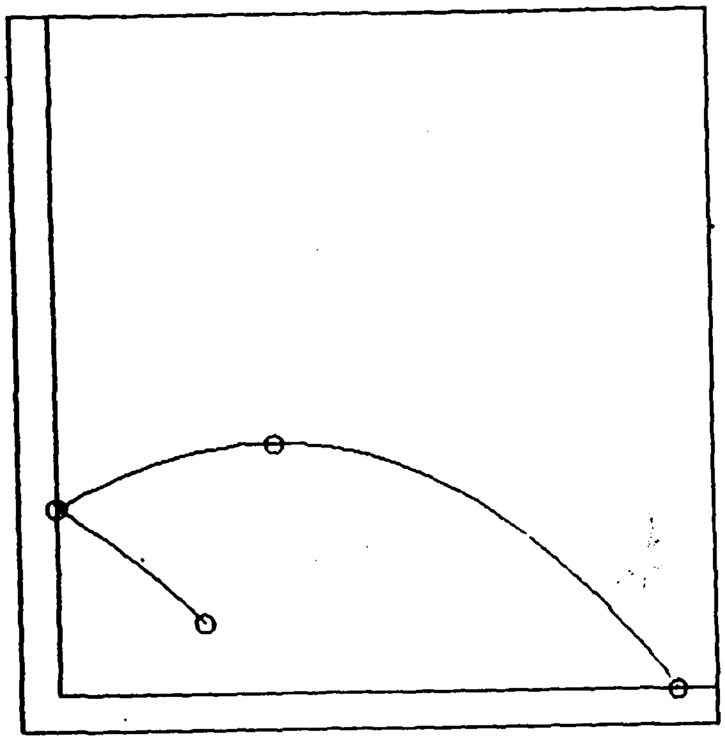
Using variables as values of parameters would make parameterized solutions possible.

Fig. 23. A sample physics problem

This file sets up the situation of the problem.

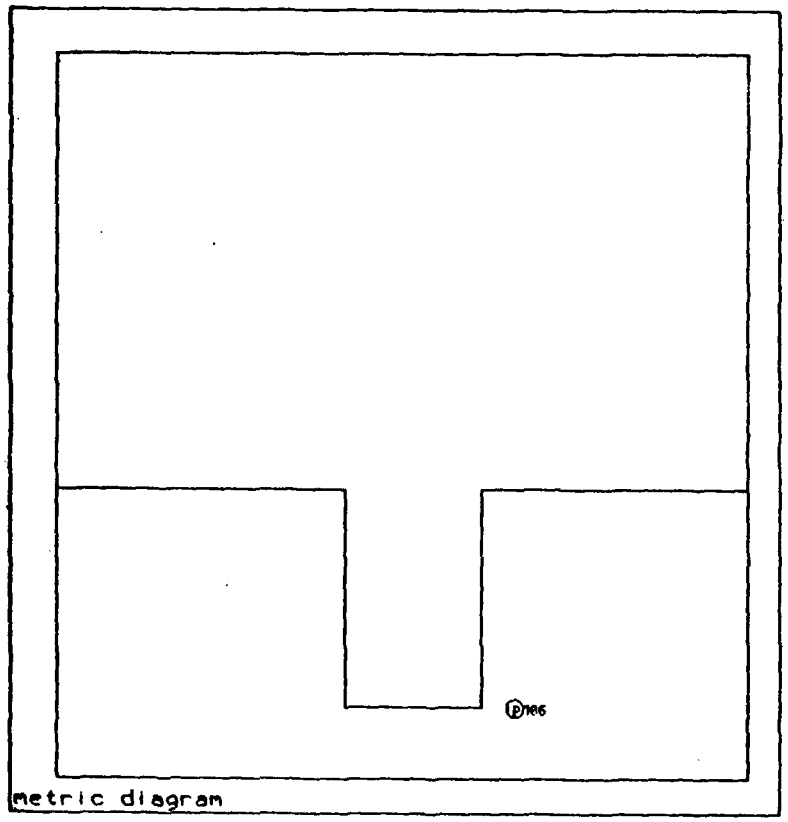
```
;-*-LISP-*-
;Situation for Problem 3-12, Newtonian Mechanics, by A.P. French

(NEW DIAGRAM (QUOTE FRENCH DIAGRAM))
(CREATE S0 (QUOTE SURFACE))
(CREATE S1 (QUOTE SURFACE))
(CREATE C1 (QUOTE CORNER))
(CREATE C2 (QUOTE CORNER))
(CREATE C0 (QUOTE CORNER))

;connect up the surfaces
(SET-PARAMETER (>> FIRST-CORNER S0) C0)
(SET-PARAMETER (>> SECOND-CORNER S0) C1)
(SET-PARAMETER (>> FIRST-CORNER S1) C1)
(SET-PARAMETER (>> SECOND-CORNER S1) C2)
(SET-PARAMETER (>> FIRST-CONNECTION C1) S0)
(SET-PARAMETER (>> SECOND-CONNECTION C1) S1)
(SET-PARAMETER (>> FIRST-CONNECTION C2) S1)
(SET-PARAMETER (>> SECOND-CONNECTION C2) NIL)
(SET-PARAMETER (>> FIRST-CONNECTION C0) NIL)
(SET-PARAMETER (>> SECOND-CONNECTION C0) S0)

;specify surface geometry
(SET-PARAMETER (>> GEOMETRY C1) (CREATE-POINT -9.0 -9.0))
(SET-PARAMETER (>> GEOMETRY C2) (CREATE-POINT 10.0 -9.0))
(SET-PARAMETER (>> GEOMETRY C0) (CREATE-POINT -9.0 10.0))
(SET-PARAMETER (>> GEOMETRY S0)
               (CREATE-SEGMENT (VALUE? (>> GEOMETRY C0))
                               (VALUE? (>> GEOMETRY C1))))
(SET-PARAMETER (>> SOLID-SIDE S0) (QUOTE +))
(SET-PARAMETER (>> GEOMETRY S1)
               (CREATE-SEGMENT (VALUE? (>> GEOMETRY C1))
                               (VALUE? (>> GEOMETRY C2))))
(SET-PARAMETER (>> SOLID-SIDE S1) (QUOTE +))
(MAPC (QUOTE (LAMBDA (GEOMS)
                     (PUTPROP (VALUE? (>> GEOMETRY GEOMS))
                              GEOMS
                              (QUOTE INTERPRETATION))))
      (LIST S0 S1 C1 C2 C0))
(CREATE THE-SCENE (QUOTE SCENE))
(SET-PARAMETER (>> DIAGRAM-NAME THE-SCENE) CURRENT-DIAGRAM)

;perform geometric analysis of scene
(SET-PARAMETER (>> DIAGRAM-READY? THE-SCENE) T)

;set up the ball
(CREATE PHOB 'BALL)
(SET-PARAMETER (>> X INITIAL-STATE PHOB) '(-5.0 M))
(SET-PARAMETER (>> Y INITIAL-STATE PHOB) '(-7.0 M))
(SET-PARAMETER (>> X-COMPONENT VELOCITY INITIAL-STATE PHOB)
              '(-10.0 M//SEC))
(SET-PARAMETER (>> Y-COMPONENT VELOCITY INITIAL-STATE PHOB)
              '(10.0 M//SEC))
(SET-PARAMETER (>> C-O-R INITIAL-STATE PHOB) 1.0)

;from the problem we know four acts will give us the answer
(SIMULATE PHOB 4.)
```

Fig. 24. Solution to the problem
The descriptions FROB computes contains the information necessary to answer the question. It should be noted that FROB itself performs no reasoning about questions and how to most efficiently answer them.



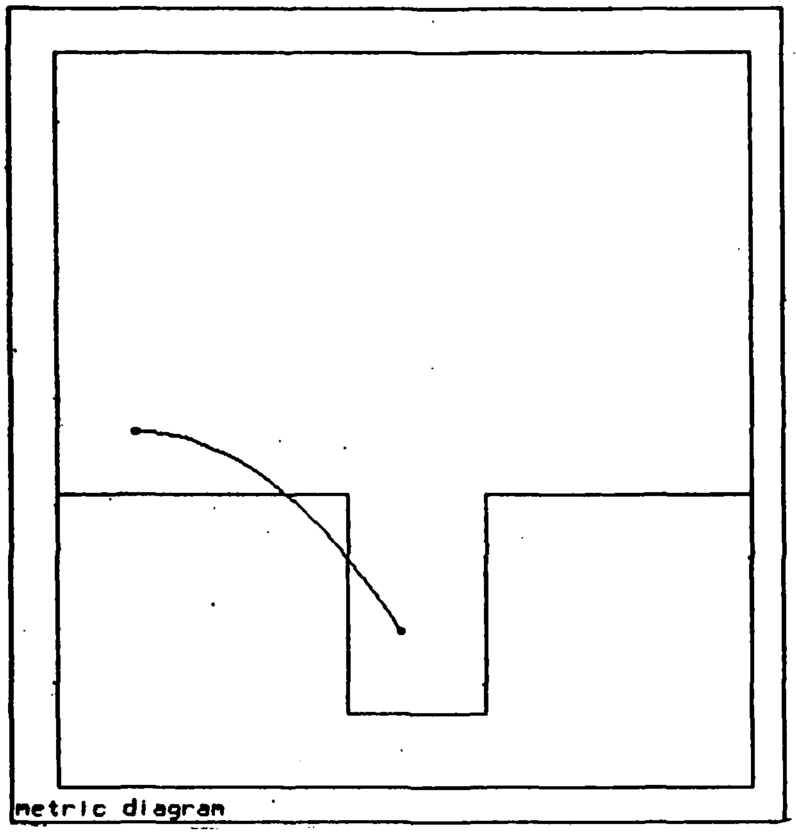
```
NIL
(draw-scene)
(PARABOLA2 PARABOLA1 PARABOLA0)
(- (value? (>> x physob3))
   (value? (>> x initial-state phob)))
13.87252352
```

Fig. 25. A ball must never be in a SOLID region of space



```
->>(create b0 'ball)
G0263
->>(set-parameter (>> x initial-state b0) 3.0)
3.0
->>(set-parameter (>> y initial-state b0) -8.0)
-8.0
 POINT106 IMPOSSIBLE LOCATION FOR MOVING OBJECT!
 CONTRADICTION DISCOVERED BY (RULE-9 . G0265)
 IT DEPENDS ON
1 (>> X INITIAL-STATE B0)=3.0 from USER
2 (>> Y INITIAL-STATE B0)=-8.0 from USER
 CHOOSE ONE TO RETRACT BY CALLING ANSWER WITH ITS NUMBER
;BKPT CONTRADICTION-HANDLER
```

Fig. 26. Surfaces are assumed impenetrable



```
 FLY DUE TO GRAVITY AND UNSUPPORTED PHYSOB
 LOOKING FOR COLLISION GOING ALONG (LEFT UP) ON PARABOLA0 FROM POINT114 TO POINT113
 ON THE PATH ARE (((-3.768646965 -2.0) SEGMENT4) ((-2.0 -3.752982302) SEGMENT5))
 SURFACES ARE (((-2.0 -3.752982302) SEGMENT5)) BORDERS ARE NIL
 CONTRADICTON DISCOVERED CONCERNING (>> COLLISION? GEOMETRY THE-FLY)
 WHOSE VALUE NIL DEPENDS ON
 THE NEW VALUE ((-2.0 -3.752982302) SEGMENT5) COMPUTED BY (RULE-6 . G2984) DEPENDS ON
1 (>> GEOMETRY NEXT-B0)=POINT107 from USER
2 (>> GEOMETRY INITIAL-STATE B0)=POINT106 from USER
3 (>> Y-COMPONENT VELOCITY NEXT-B0)=0.0 from USER
 CHOOSE ONE TO RETRACT BY CALLING ANSWER WITH ITS NUMBER
;BKPT CONTRADICTION-HANDLER

(draw-scene)
(PARABOLA0)
█
```

Fig. 27. A FROB simulation

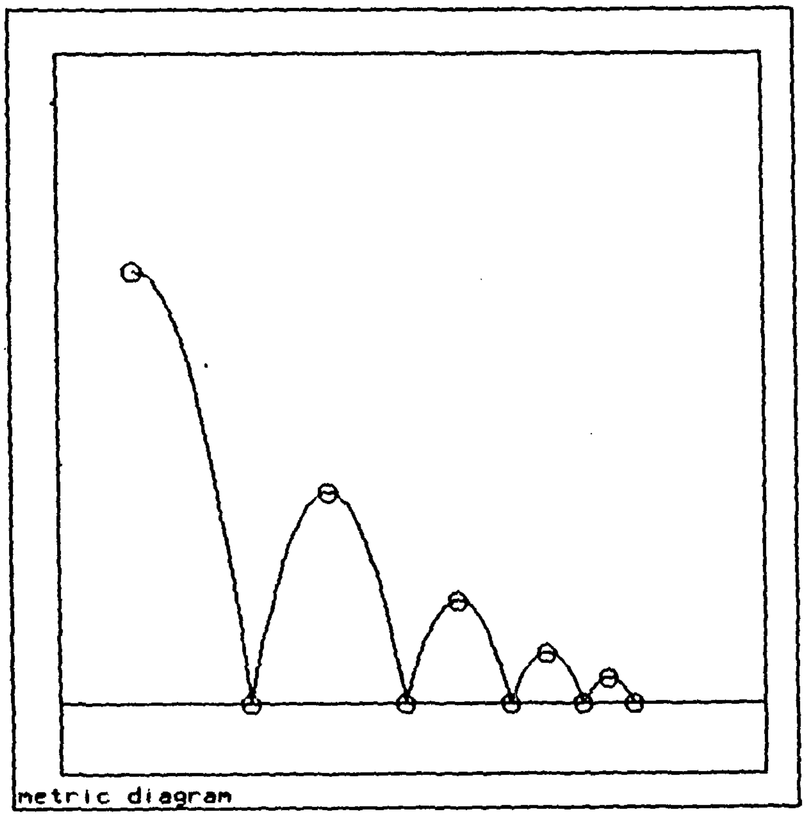


Fig. 28. Another FROB simulation

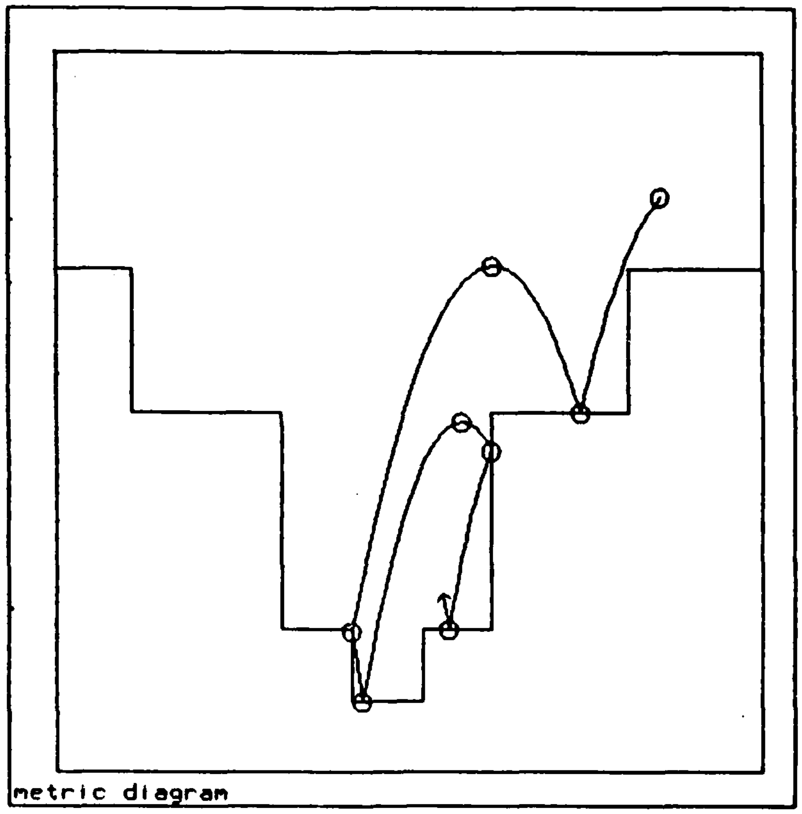
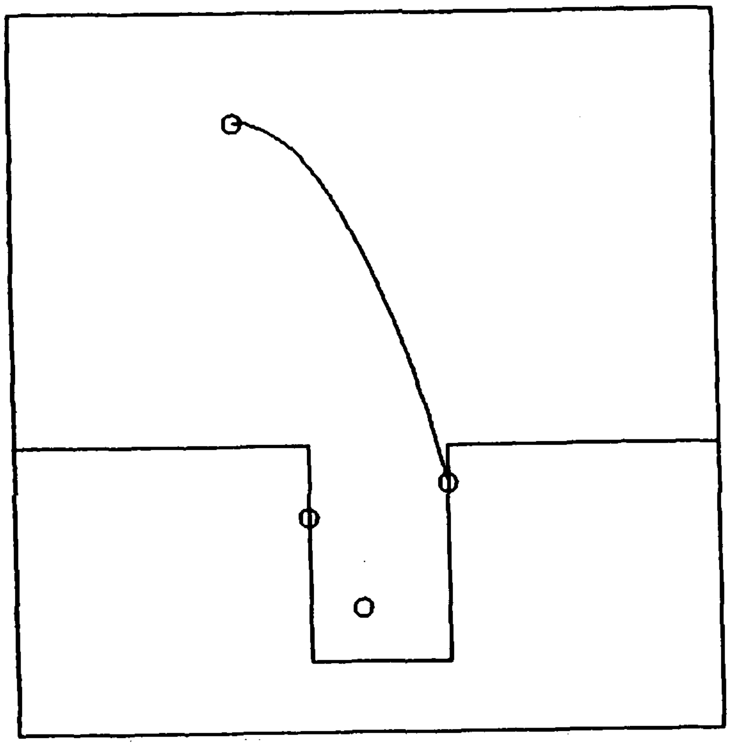


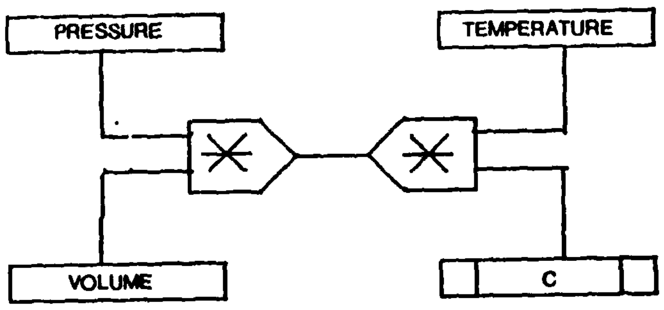
Fig. 29. A detected inconsistency
FROB often needs more information than people do to discover that a description is inconsistent.



```
 CONTRADICTON DISCOVERED CONCERNING (>> (A2 YSUM1 ENERGY F1)
 WHOSE VALUE 8.888888896 DEPENDS ON
 THE NEW VALUE 2.111111112 COMPUTED BY (RULE-3 . G0892) DEPENDS ON
1 (>> Y S1)=-4.0 from USER
 BOTH VALUES SHARE THESE ASSUMPTIONS -
2 (>> (CHECKED-VALUE COR-CHECK S3) (C-O-R S3))=0.5 from USER
3 (>> Y-COMPONENT VELOCITY S5)=0.0 from USER
4 (>> Y S5)=7.0 from USER
5 (>> X S5)=-4.0 from USER
6 (>> Y S3)=-3.0 from USER
7 (>> X S1)=-2.0 from USER
8 (>> X S3)=2.0 from USER
 CHOOSE ONE TO RETRACT BY CALLING ANSWER WITH ITS NUMBER
;BKPT CONTRADICTION-HANDLER
```

Fig. 30. Some qualitative reasoning could fit into the constraints
This constraint embeds Boyle's law describing an ideal gas. U means the quantity is increasing, D means decreasing, and C means steady. Such reasoning has not been applied to the Bouncing Ball world in FROB.

```
(create boyle 'gas-law)
G0024
>>(set-parameter (>> volume boyle) 'c)
C
>>(set-parameter (>> pressure boyle) 'u)
U
>>(what-is (>> temperature boyle))
(>> (M1 P2 BOYLE) (TEMPERATURE BOYLE)) = U
NIL
>>(why (>> temperature boyle))
I used rule (RULE-2 >> P2 BOYLE) on the following inputs:
   (>> (PRODUCT P2 BOYLE) (PRODUCT P1 BOYLE))
   (>> M2 P2 BOYLE)
(G0029 G0031)
>>(premises (>> temperature boyle))
(>> (M2 P1 BOYLE) (VOLUME BOYLE)) = C
(>> (M1 P1 BOYLE) (PRESSURE BOYLE)) = U
T
>>
```



Constraint networks have been used for algebraic manipulation before [Sussman and Stallman], but the diagram used by FROB provides an extra complication. An equation generated from an Action Sequence would have to include a region of applicablity attached to it due to the finite extent of surfaces in the diagram. The process of computing these bounds deserves some study.

Another potential use of the Action Sequence is to provide a target representation for checking the consistency of quantitative data. Even when equations of motion are not availible for some type of motion, general properties such as smoothness of trajectory usually hold. A list of state parameters for motion could be parsed into an Action Sequence description by looking for the data points where these general properties change and using them as the transitions between act descriptions. To perform this type of reasoning, the qualitative constraints associated with each type of act need to be explicated, as well as the details of the parsing process.

4. Qualitative Representation of Motion

A description of a motion is qualitative if it uses terms that reflect its essential features. Specific values of parameters, such as position or elasticity, are mapped into classes with distinct properties. For example, the position of a ball might be specified only as being inside a region of space instead of at a point. Since there are a number of parameters associated with motion, each of which could be quantized in many ways, there can be more than one qualitative description of a situation.

A good basis for a qualitative description is to abstract the quantitative concept of state. If the corresponding qualitative state is defined properly, a set of rules can be defined that describe what qualitative states can occur after some other state. These rules can be used to generate a descripton of all the motions possible from some initial state by a process of simulation. The results of this process are called the envisionment for that state. Envisioning can be thought of as "imagining what can happen" in some situation. The envisionment can be used to plan solutions to problems (as in deKleer's NEWTON), to compute a summary of motion, or to evaluate collision possibilities.

Many assumptions about motion are properly expressed in qualitative terms. One might, for example, assume that a ball never collides with a certain surface and must be in a particular region of space at some time during its flight. The description of possible motions provides a natural place to incorporate these assumptions.

In this chapter a concept of qualitative state for the Bouncing Ball world is defined. Discussed next are the considerations that define the Space Graph, the basic qualitative representation of space in this domain. The rules for qualitative simulation are then described, along with the process of using them to compute the Sequence Graph, which is the description of possible motion for the domain. In the last section I discuss the process of using qualitative assumptions about motion to constrain the Sequence Graph.

4.1 Qualitative properties of Motion

The quantitative state of a ball consists of a set of numbers that describe its position and velocity relative to some coordinate frame. Implicit in this state is information about what the ball is or is not touching and the kind of motion it is undergoing. A qualitative state must make the latter information explicit, as well as generalize the position and velocity.

The actual position of a ball at any particular time is a point in space. If we wish to be less precise about where a ball is, we must use the concept of a PLACE. A PLACE is a connected part of a diagram where conditions are in some sense uniform. An instance of a PLACE in a diagram might be the free space above a particular surface.

The type of motion occuring is an obvious component of a qualitative state. The vocabulary we shall use is almost the same as developed in Chapter 3. To reiterate, the motion types were FLY, COLLIDE, CONTINUE, STOP, SLIDE/STOP, and SLIDE/STOP/FALL. One addition will be made later to describe a transition between qualitatively distinct regions of space, called PASS.

The gravity vertical and the independence of directions imposes a natural quantization of heading. The standard names of UP, DOWN, LEFT, and RIGHT will be used for the four principal directions. Motion can have both UP-DOWN and LEFT-RIGHT components, so a heading can also be specified by a two element list, such as (RIGHT UP). A heading of NIL corresponds to zero velocity.

These three properties - type of motion, position, and heading - define the qualitative state of a ball. Defining the position component of the qualitative state will be the concern of the next section. Before that we will examine some other properties of the Bouncing Ball world to see if they have natural quantizations that are useful for reasoning about motion.

A physical property that causes distinctly different behaviour is elasticity. There are three cases corresponding to the COR being 0, 1 and somewhere in between. If the COR is 1 ,he ball must recoil and cannot stop, because the possiblity of grazing contact with a surface is ignored. If the COR is 0 the ball cannot recoil and must either stop or fall. If it is in between it can do either.

Aside from the degenerate case of not moving, there does not appear to be a natural quantization of speed. When speed is mentioned in qualitative terms it is always defined relative to something else, such as "moving fast enough to escape the well".

Still another property of motion that could be abstracted is the sequence of motion types. People describing motion in the Bouncing Ball world often distinguish two patterns of motion that could be used to summarize a larger sequence of motion types. The first are occurences of the FLY-COLLIDE-FLY pattern on a particular surface, and the second is all the FLYs and COLLIDEs that occur within a particular place. Examples of these are the informal statements "bouncing on the floor" or "bouncing around inside the well". While these summaries allow a very concise description of motion, a more detailed analysis would require expansion into individual motion types. By creating only the most detailed qualitative description, FROB avoids both the problems involved in choosing the proper level of description for a question and in expanding descriptions.

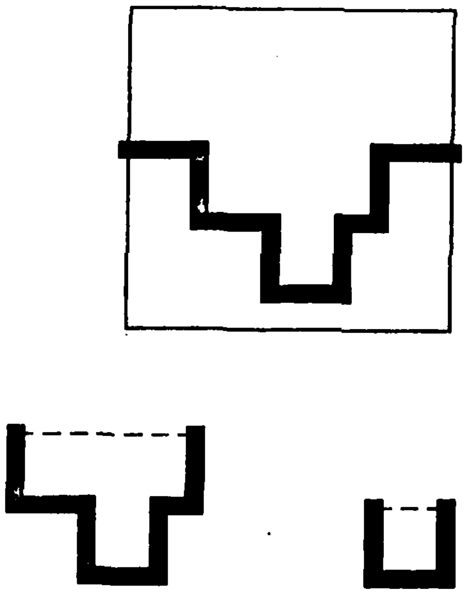
4.2 Breaking up Space

Qualitative spatial reasoning requires the notion of a place. A place is a connected subset of space in which some distinguished property holds true. The properties that define places depend on the type of reasoning being done, as well as the geometric properties of the specific situation. This section examines how space should be broken up to define places that are useful for reasoning in the Bouncing Ball world.

The assumptions of the Bouncing Ball world simplify the description of places. Since all balls are modelled as point masses, the only geometric factor in the definition of places is the configuration of surfaces in the diagram. The set of places relevant to one ball must be relevant to all because every ball is subject to the same forces. In a more general domain each object could require a seperate description of space because of differences in shape, size, and interactions.

The set of places that are required for reasoning may overlap in spatial extent. This is illustrated in Figure 31 , where a well is contained within another well. Rather than using the Metric Diagram to

Fig. 31. PLACEs may overlap
A particular region of space may belong to more than one qualitatively distinct PLACE. Being inside the small well implies being inside the large one, but not vice versa.



compute new place descriptions as they are needed, we will break space up into a set of non-overlapping parts that will form a basic vocabulary of places. Any other place that needs to be defined can be composed of places in this vocabulary. The advantages of a decomposition of space into non-overlapping parts are threefold. First, the vocabulary is suitable for the position component of the qualitative state because every point in free space maps in to some unique place. Secondly, the problem of dynamically expanding a description is avoided. Lastly, such a decomposition allows space to be represented as a graph. The graph structure can be used as a spatial index and provides a framework for propagation algorithms.

There are three considerations in defining the basic place vocabulary for the Bouncing Ball world. First, we would like to avoid making unnecessary distinctions. This will keep the size of the motion descriptions small. Secondly, the descriptions we compute will be simpler if for any particular direction, only a single type of qualitative event can occur. This reduces the branching factor of motion descriptions. These two factors point to using changes in the surface geometry to indicate changes in regions. Lastly, the existence of a gravity vertical means that bouncing just up and down is different from bouncing up and down with a left or right component, and so we should cut space with vertical and horizontal lines. This makes the description of the simplest motion in the domain, bouncing up and down on a horizontal surface, simple. These points are illustrated in Figure 32.

Space can be quantized to satisfy these considerations by slicing free space with vertical and horizontal lines from all corners to the intersection with another surface or border. These regions of space and the edges that bound them are transformed into the nodes of a graph, connected by pointers labelled with directions that express the adjacency of the elements. This graph is called the <u>Space Graph</u>.

The elements of a Space Graph fall into four distinct classes. The chunks of free space are called <u>Sregions</u> (<u>Space regions</u>). <u>Surfaces</u> and <u>Borders</u> are subsets of the surfaces and border originally specified by the Metric Diagram for the scene. The edges used to cut free space are called <u>Free</u> edges.

Each element has four adjacency pointers, which correspond to the name of the element reached by travelling in the direction specified by the label (see figure 33). Space outside the diagram is denoted by the label SPATIUM-INCOGNITO. The Sregions are considered to be open since their boundary is a distinct part of the representation. This is consistent with the representation of surfaces and solid regions.

In the present implementation corners are not given independent existence. This can be a source of inconsistency in the mapping from points in the diagram to places in the space graph because a corner belongs to more than one place. It also means that falling straight down along a free edge cannot be defined, although it is consistent with the rest of the semantics of the Bouncing Ball world.

The elements of the Space Graph are the largest set of places that can be distinguished using the principal directions and without introducing new points of reference. The network structure imposed by the adjacency pointers makes it useful as a framework for the envisionment process. However, more complex places also need to be defined.

A <u>well</u> is a set of connected Sregions such that the elements on the bottom and sides are surfaces. Wells are interesting because a ball can be trapped in them. Figure 34 shows the wells computed for two simple diagrams.

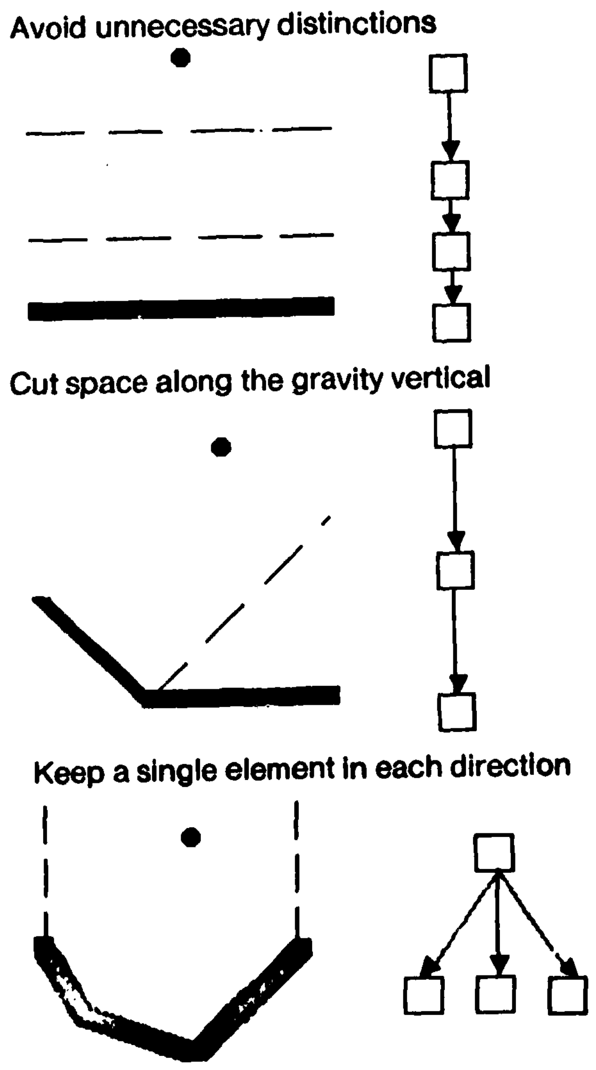

Avoid unnecessary distinctions
Cut space along the gravity vertical
Keep a single element in each direction

Fig. 32. Constraints on the qualitative description of space
The physical constraints are gravity and the configuration of surface. The desire to keep the description of motion simple provides a computational constraint.

Fig. 33. Space Graph datastructures
The datastructures for the Space Graph are annotated Metric Diagram elements. The Metric Diagram properties, as well as index information stored on them by the envisioning process, are not shown.

```
SREGION0                          SEGMENT1
left: SEGMENT2                    up: SREGION0
right: SEGMENT10                  connecting-region: SREGION0
up: SEGMENT7                      class: SURFACE
down: SEGMENT1
class: SREGION

SEGMENT2                          SEGMENT10
right: SREGION0                   left: SREGION0
left: SPATIUM-INCOGNITO           right: SREGION3
connecting-region: SREGION0       class: FREE
class: BORDER
```

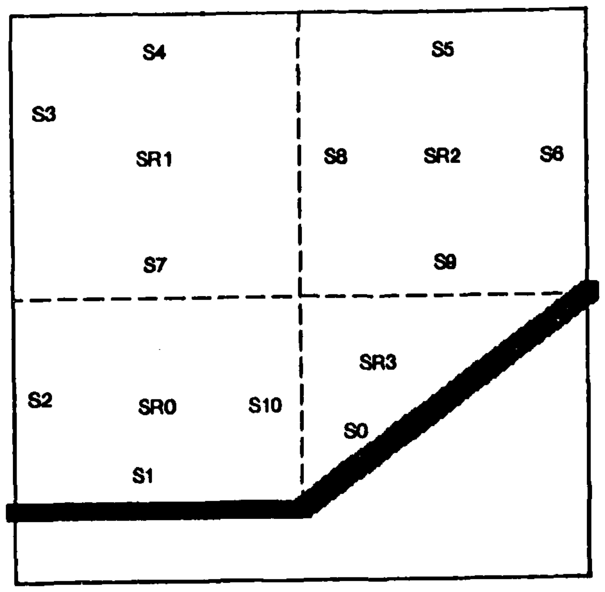
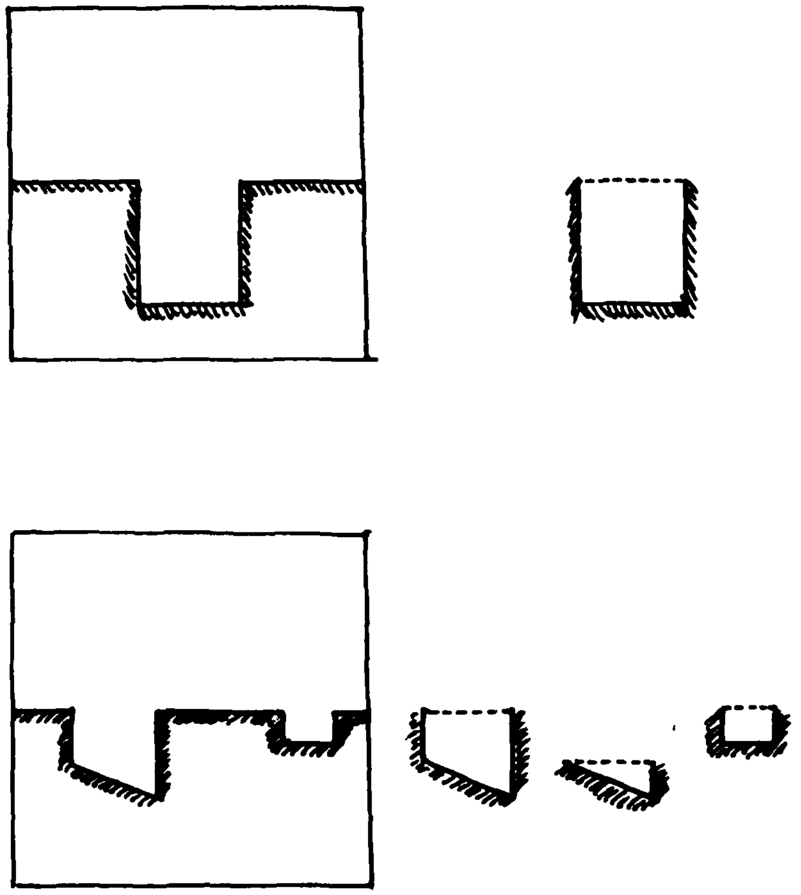


Fig. 34. Wells
A well is a connected region of FREE space whose sides and bottoms are surfaces and whose top is FREE.



The free space above a surface with horizontal extent and the free space between two surfaces with vertical extent form places that are used in pruning the Sequence Graph. These places are shown for the diagram of a well in Figure 35.

4.3 Envisioning

Given the Space Graph and the motion vocabulary of the Action Sequence, we have all the machinery we need to define a qualitative notion of state. A qualitative state of a ball, called a Qstate, is a triple of the form

(<type of motion> <place> <heading>)

where

<type of motion> = motion types from Action Sequence
<place> = Space Graph element
<heading >= basic directions or NIL

For envisioning we need rules that transform the current Qstate into the Qstates that can occur next. We will present these rules according to the class of the place associated with the current Qstate.

If the current place is an Sregion, the next place and direction depend on the current direction. Figure 36 summarizes the possiblities. Ambiguities arise from not knowing the relative magnitudes of the velocity components and whether gravity will dominate the motion. The next type of motion depends on the class of the next place. If the next place is an Sregion the ball will FLY, and if a border it will CONTINUE. A new action type is used to mark a transition between Sregions - if the place is a FREE edge, the ball is said to PASS.

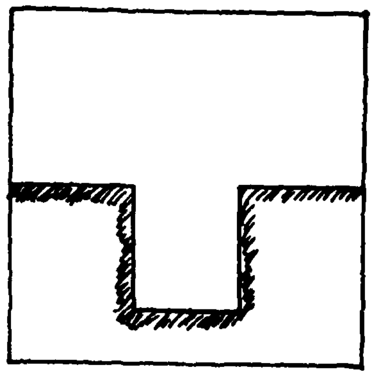
If the current place is a FREE edge, then the next place is found by getting the contents of the adjacency pointers named by the current direction. The next type will be FLY because a FREE edge always seperates two Sregions, and the direction remains the same.

For a border the ball will CONTINUE with the same direction if the current direction evaluated on the border yields SPATIUM-INCOGNITO as the next place. Otherwise the ball will FLY with the same direction and in the Sregion discovered using the adjacency pointers. Figure 37 illustrates the use of these rules.

The possibilties of motion at a surface are more complex because a ball colliding with it may dissipate enough energy that the ball could stop, slide, or fall instead of recoiling. The table in Figure 38 provides the new direction of recoil given the orientation of the surface and the direction of impact. The type of a recoil state is FLY with the place being the surface. The orientation of the surface and the incident direction determine what happens if the ball does not recoil. If the direction is DOWN and the surface orientation is RIGHT (the solid side is below the surface), STOP is the next type with direction NIL. Depending on the orientation of the surface the next type is either SLIDE/STOP or FALL. SLIDE/STOP is terminal, while FALL becomes a FLY in the region adjacent to the surface.

With our set of rules for qualitative simulation we can define the process of envisioning. It will

Fig. 35. Other PLACEs
These regions of space will prove to be useful in pruning the envisionment.



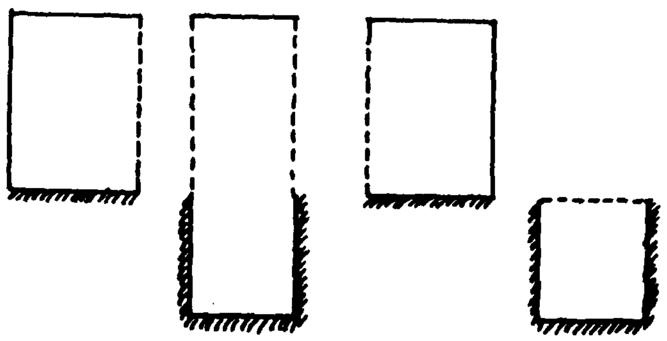


Fig. 36. Sregion transition table

Direction	Next Place	Next Direction
nil	$self	D
D	down	D
(L D)	down left	(L D) (L D)
L	$self	(L D)
(L U)	$self left left up	L L (L U) (L U)
U	$self up	D U
(R U)	$self right right up	R R (R U) (R U)
R	$self	(R D)
(R D)	right down	(R D) (R D)

Motion type of next Qstate is determined by the class of the next state

	BORDER -> CONTINUE
class =	SURFACE -> COLLIDE
	FREE -> PASS

---

prove useful to be able to compute descriptions of what motions are possible for more states of a ball than just the initial one. For example, a description of the motions possible after several bounces may reveal that the ball has lost so much energy that it is trapped in a well. Therefore the description cannot be in terms of Qstates alone.

Let a SEQ node be the representation of a Qstate in the description of motions possible from a particular state of a ball. Envisioning proceeds by creating a SEQ node representing the initial Qstate and using the rules to generate SEQ nodes corresponding to the Qstates possible from that state. The rules are then applied to each new SEQ node that is generated. SEQ nodes are linked by BEFORE and AFTER pointers which reflect their temporal ordering, and are indexed by their place in the Space Graph to

Fig. 37. Rules for BORDERs and FREE edges

```
SEGMENT2                           SEGMENT10
right: SREGIONO                    right: SREGION3
left: SPATIUM-INCOGNITO            left: SREGIONO
connecting-region: SREGIONO        class: FREE
class: BORDER


Being at SEGEMENT2 with LEFT, (LEFT UP), or (LEFT DOWN)
                -> CONTINUE in SPATIUM-INCOGNITO
                otherwise FLY in CONNECTING-REGION, SREGIONO

Being at SEGMENT10 -> PASS
        if direction contains LEFT, next place is SREGIONO
        if direction contains RIGHT, next place is SREGION3
        otherwise undefined

New direction of motion is the same as the old
```

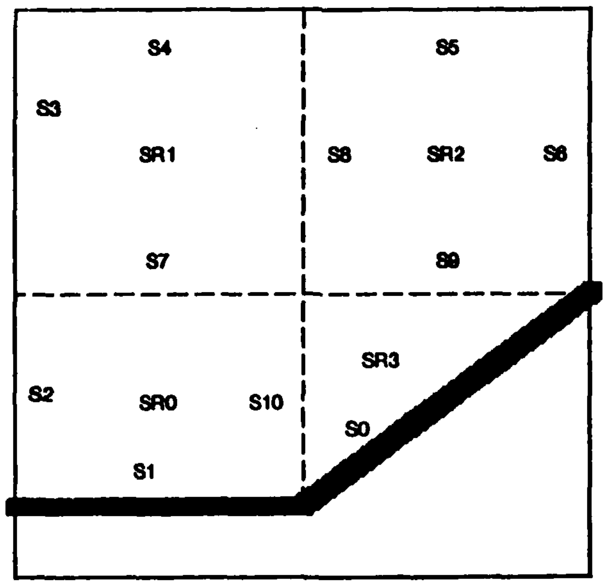


Fig. 38. Recoil table for SURFACEs
The direction of a surface is the direction travelled from one end to the other with the SOLID side on the right. A 2D projection of flat ground, for example, would be (RIGHT). "X" denotes an impossible combination.

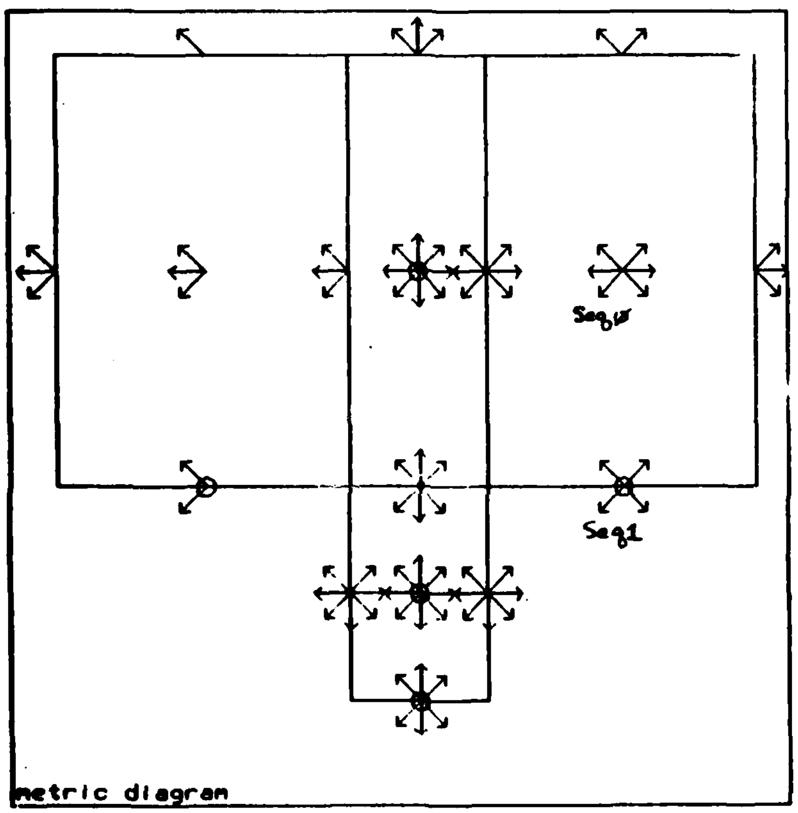
heading	Surface Orienta.ion							
	R	L	U	D	(R U)	(L U)	(R D)	(L D)
R	X	X	X	U	X	X	(R D)	(L U)
(R U)	U (R U) (L U)	X	X	L (L D) (L U)	U (R U) (L U)	X	L U (L D) (L U) (R U)	L (L U) (L D)
U	L	X	X	X	(L U)	X	(L D)	X
(L U)	D (R D) (L D)	X	L (L U) (L D)	X	L D (R D) (L D) (L U)	L (L U) (L D)	D (R D) (L D)	X
L	X	X	D	X	(R D)	(L D)	X	X
(L D)	X	D (L D) (R D)	R (R U) (R D)	X	R (R U) (R D)	R D (L D) (R D)	X	D (L D) (R D)
D	X	R	X	X	X	(R U)	X	(R D)
(R D)	X	U (L U) (R U)	X	R (R D) (R U)	X	U (L U) (R U)	R (R U) (R D)	R U (R D) (L U) (R U)

---

avoid looping. This process continues until no new SEQ nodes are generated. Termination is guaranteed by the finite number of places and the finite number of Qstates possible at each place.

The collection of SEQ nodes forms a rooted, directed graph called the Sequence Graph. Any motion from the initial Qstate can be described as a path through the Sequence Graph, and all oscillations correspond to cycles in it. SEQ nodes that correspond to leaving the FLY case of motion (CONTINUE, STOP, and SLIDE/STOP) are explicitly marked as terminals. A schematic representation of the Sequence Graph is drawn in Figure 39. Examples of the actual datastructures of the SEQ nodes may be found in Appendix one. Despite its size (about 85 nodes for a diagram containing a single well) it is quite economical to compute.

Fig. 39. Diagrammatic representation of a Sequence Graph
Each arrow or circle represents an SEQ node and is drawn over the element of the Space Graph which represents its location. The direction is indicated by the direction of the arrow, and circles mean either no direction or a non-recoil state at a surface. The root node of the graph is not distinguished, nor are the orderings of the elements.



```
->>(what-is (>> root sequence-graph phob))
(>> ROOT SEQUENCE-GRAPH PHOB) = SEQ0
NIL
->>(pseq seq0)

 THIS IS THE START NODE OF THE GRAPH FOR G2860
SEQ0
(FLY SREGION3 (LEFT DOWN))
 CAN BE REACHED BY (SEQ12)
 NEXT CAN BE (SEQ1 SEQ2)
SEQ0
->>(pseq seq1)

SEQ1
(COLLIDE SEGMENT9 (LEFT DOWN))
 CAN BE REACHED BY (SEQ0)
 NEXT CAN BE (SEQ3 SEQ4)
SEQ1
```

4.4 Using qualitative constraints on motion

Several kinds of constraints influence the possiblities for motion. Some physical parameters eliminate classes of motion. For example, a ball that is completely inelastic will not recoil when it collides with a surface. Other constraints are imposed as assumptions about the motion that reflect some desired state of affairs, such as assuming that a ball does not go into a well. We shall view physical properties as assumptions, since they are asserted by a fallible user. The effect of these assumptions is to prune Qstates from the Sequence Graph.

Not all qualitative constraints on motion can be enforced by pruning the Sequence Graph. "Bounce three times on SURFACE0 and more than five times on SURFACE1" is one example[1] The constraints that can be assimilated by pruning are those which specify that a certain Qstate is required in (or excluded from) every Qstate path, or that a particular place in the Space Graph is reached (or never reached) in every Qstate path.

Elasticity and energy are the physical properties that cause SEQ nodes to be excluded from the Sequence Graph. If the COR is 1 (the ball is perfectly elastic) all Qstates after a collision that do not correspond to recoil (SLIDE/STOP, STOP, and FALL) are excluded, and if the COR is 0 all recoil Qstates after the collision must be excluded. Energy limits the maximum height a ball can reach, and all places in the Space Graph that are completely above this height are excluded.

We will call an SEQ node alive if it corresponds to a Qstate of motion that is possible under whatever set of assumptions currently holds, and dead if it does not. Marked on each dead node is the reason for not believing that it corresponds to a possible state of motion. The Sequence Graph is made to reflect the assumptions about motion by killing the nodes required to directly satisfy each constraint in turn, and then killing off other nodes that must die as a consequence.

The direct consequences of assumptions are simple to compute. Assuming a Qstate does not occur kills its corresponding SEQ node, and assuming that a place cannot be reached kills all SEQ node at that place. To assume that certain states or places are required means that all nodes not on a path that includes these states and places cannot occur. A simple marker sweeping algorithm finds these nodes.

Once the direct effects of individual assumptions have been found the indirect consequences of these changes must be computed. Several facts of motion are involved in this process. All Qstates that are possible must be reachable via some path of live Qstates from the initial Qstate. This means that there can only be a single connected set of live nodes in any pruned Sequence Graph. Unless the ball is perfectly elastic, it must either stop or leave the diagram. This corresponds to the requirement that every Qstate (unless the COR = 1) be on some Qstate path that includes a terminal node. These restrictions on

1 The Sequence Graph can be thought of as a specification for a finite state machine that recognizes paths of Qstates corresponding to legal motions from the initial state. Constraints on motion may also be viewed as specifying a language on Qstate paths. The class of languages that can be expressed by pruning the Sequence Graph is a subset of the regular languages on this alphabet; "any path that contains exactly two instances of (FLY (SREGION0) (LEFT))" is a regular language that cannot be so expressed.

the envisionment can be enforced by another marker sweeping algorithm. All nodes that can be reached from the root are found by propagating a mark across AFTER pointers through live nodes. Marking from all live termination states across BEFORE pointers through live nodes finds the nodes which are on a terminated path. All other nodes are considered dead. The results of this pruning process on a Sequence Graph is illustrated in figure 40.

Using the requirement that motion occurs on a continuous path in space can speed up the pruning process. We define a place to be reachable in some Sequence Graph if and only if there is a live SEQ node at that place. All places that do not belong to a path of reachable places from the place of the initial Qstate are considered unreachable. The processes described above can render some places unreachable, and by using a marker sweeping algorithm the places that are unreachable as a consequence can be discovered. A mark is placed on the place containing the initial Qstate, and propagated out in all directions through the Space Graph, stoping at places that are marked as unreachable. The SEQ nodes associated with unmarked places can then be killed. The small size of the Space Graph as compared to the Sequence Graph makes this technique very attractive.

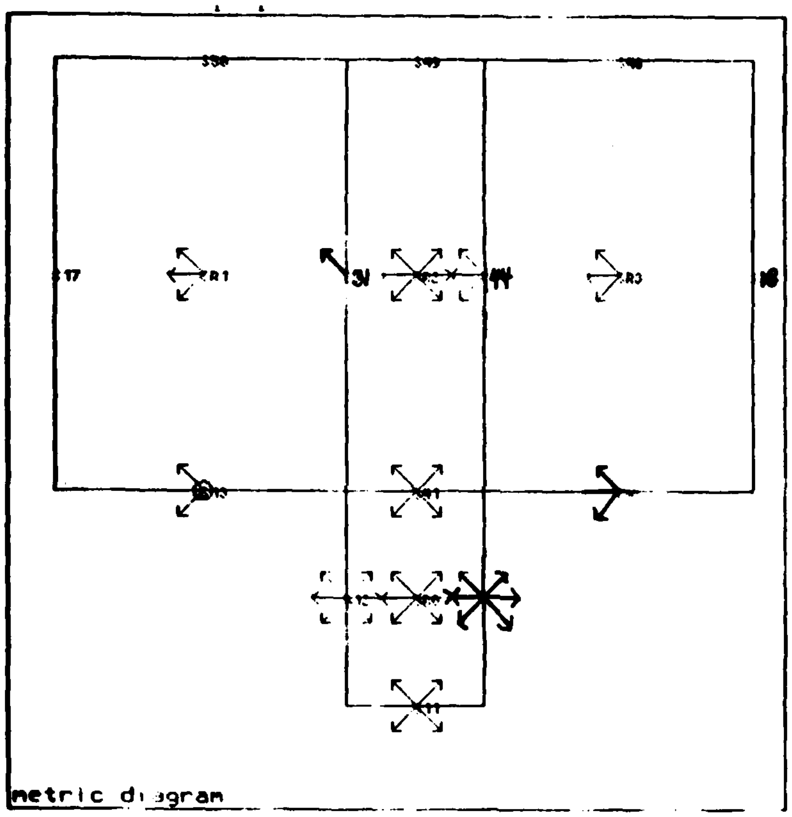
If a ball is perfectly elastic, it might bounce around inside the region of space enclosed by the diagram forever. This corresponds to a Sequence Graph with no live termination states, and thus the termination sweep described above cannot be used. A local fact that can be used is that if all the nodes that are AFTER a particular node are dead, that node must also be dead. This helps in some cases, but is stymied by cycles in the Sequence Graph. Figure 41 illustrates this. In the Sequence Graph drawn here it is considered possible for the ball to be bouncing up and down heading leftward, without ever reaching the left side of the place it is bouncing in. This could happen only if the leftward velocity is infintesimal, which is impossible in practice. We call this problem the Qualitative Zeno's Paradox, by analogy with the classical motion description problem.

The Qualitative Zeno's Paradox would be very difficult to solve within the confines of the Sequence Graph. It is akin to the problem of garbage collecting circular lists or dealing with a circularity in a dependency system, because the cycle in the graph structure makes local methods fail. In addition, the sheer number of cycles in a Sequence Graph prevents their identification and inclusion in some modification of a local method.

The difficulty disappears if we use space in our reasoning. The places where a ball can bounce are the regions of free space above some surface with horizontal extent, or between two surfaces with vertical extent. A ball can be travelling in such a place in a particular direction only if at least one of three things happens. It must either leave the place going in the same direction, it can stop inside the place after going in that direction, or it can change direction within that place. If none of these occurs the ball cannot be moving in that direction within that place. This rule is applied to each of the four principal directions in each of the places in the Space Graph as defined above.

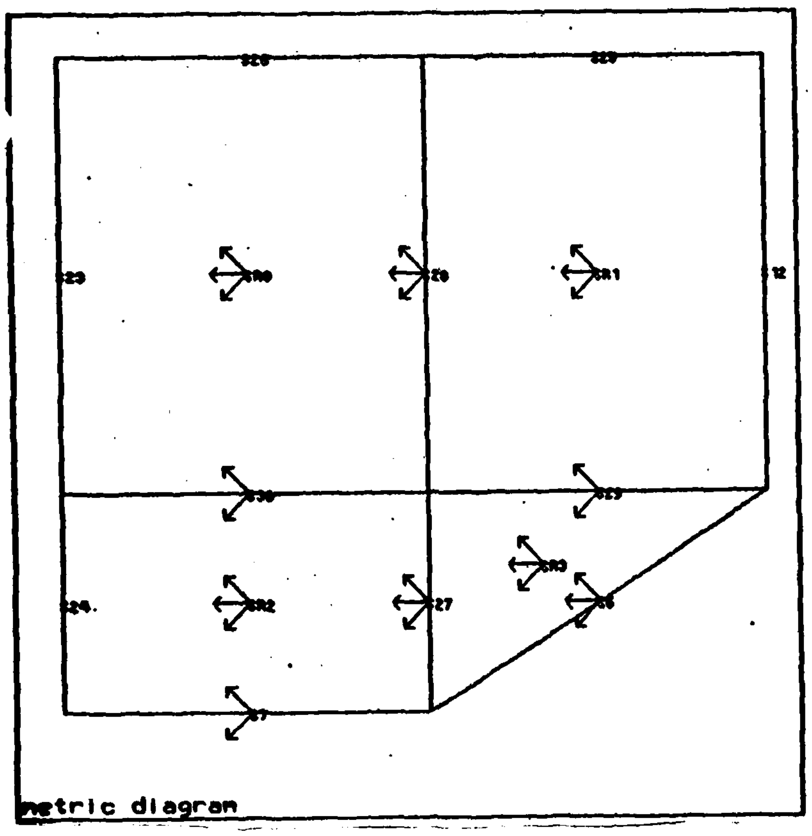
These pruning techniques are iterated over the Sequence Graph until no further nodes are killed. The algorithms are slightly more complex than described here because they annotate a node with the information about the reasons for its death. This information is useful in tracing the effects of particular assumptions.

Fig. 40. Effects of assumptions on the Sequence Graph
Ambiguity in the Sequence Graph may be reduced by making assumptions about physical parameters and the properties of motion.



```
->>(why-not seq37)
 (CONTINUE SEGMENT17 (LEFT)) IS UNATTAINABLE BECAUSE
 (CANNOT-REACH SEGMENT17)
SEQ37
->>(why-not seq35)
 (CONTINUE SEGMENT50 (LEFT UP)) IS UNATTAINABLE BECAUSE
 (ENERGY)
SEQ35
->>(why-not seq77)
 (CONTINUE SEGMENT18 (RIGHT)) IS UNATTAINABLE BECAUSE
 (REQUIRED-STATES (SEQ22))
SEQ77
->>(pseq seq22)

SEQ22
(PASS SEGMENT31 (LEFT UP))
 CAN BE REACHED BY (SEQ15)
 NEXT CAN BE (SEQ30)
SEQ22
```

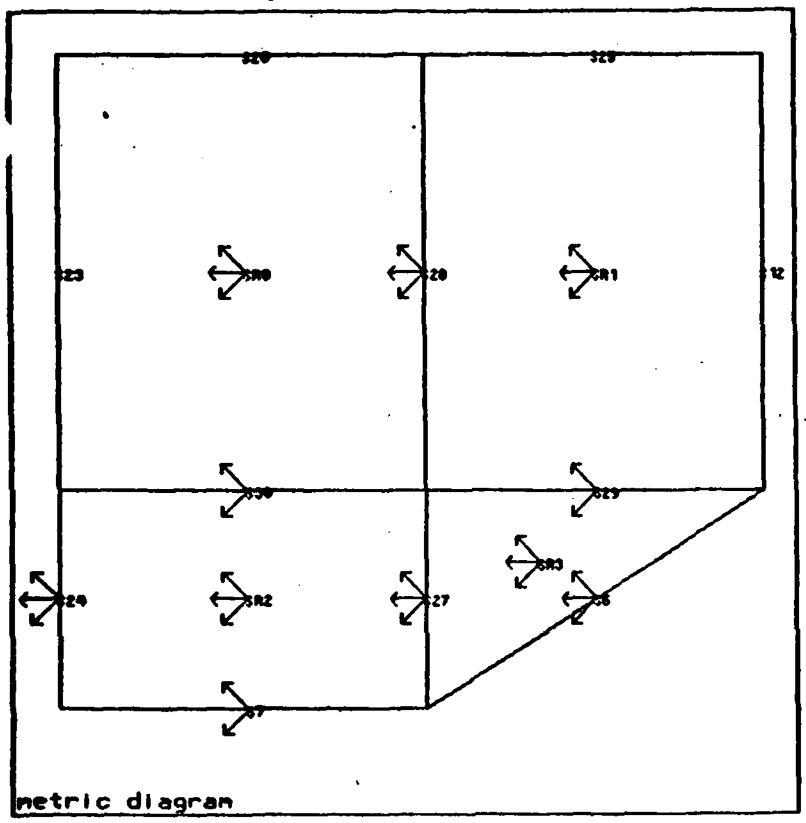
Fig. 41. Qualitative Zeno's paradox
In this description it is considered possible to bounce up and down to the left forever without stopping nor reaching the leftmost part of the region of space in which the ball is moving. This situation is impossible without velocities becoming infintesimal.



```
->>(describe-it (>> assumptions phob))

 (>> ASSUMPTIONS PHOB) = G0355
 G0355 IS THE SET OF MOTION ASSUMPTIONS FOR (>> INITIAL-STATE PHOB)
 THE ROOT OF ITS SEQUENCE GRAPH IS SEQ0
 EXCLUDED PLACES ARE (SEGMENT26 SEGMENT25 SEGMENT12 SEGMENT24 SEGMENT23)
G0355
```

There are three ways that inconsistencies can arise within the Sequence Graph. First, the assumptions may explicitly conflict. Requiring a Qstate whose place is excluded is one way for this to happen. A node or place that is required might die as a consequence of other constraints, such as an energy bound. Lastly, the root node may die, implying that the graph is overconstrained. Figure 42 shows an overconstrained Sequence Graph. In each of these cases FROB complains and offers up the offending assumptions for inspection and possible correction.

Fig. 42. An overconstrained Sequence Graph



```
->> (cannot-leave-diagram phob)
(SEGMENT23 SEGMENT24 SEGMENT12 SEGMENT25 SEGMENT26)
 UPDATING ASSUMPTIONS FOR (>> INITIAL-STATE PHOB)
 SEQUENCE GRAPH FOR (>> INITIAL-STATE PHOB) OVERCONSTRAINED
 (FLY SREGION1 (LEFT DOWN)) IS UNATTAINABLE BECAUSE
 (CANNOT-REACH SEGMENT24)
 (CANNOT-REACH SEGMENT23)
 (CANNOT-REACH SEGMENT26)
 (ELASTIC)
 (CANNOT-REACH SEGMENT25)
 WOULD YOU LIKE TO CORRECT THE MOTION ASSUMPTIONS?yes

 TYPE ♦P TO CONTINUE
;BKPT MOTION-ASSUMPTION-CHECK

(might-be-at phob segment24)
(SEGMENT26 SEGMENT25 SEGMENT12 SEGMENT23)
♦p
 UPDATING ASSUMPTIONS FOR (>> INITIAL-STATE PHOB)
 CHECKING PATH OF MOTION AGAINST ASSUMPTIONS
->>
```

5. Solving Motion Problems

A description should be judged by how appropriate it is for some purpose. In defining the Bouncing Ball world four questions were introduced to serve as a focus. They were

1. What can a ball do next?
2. Where can it go next?
3. Where can it end up?
4. Can these two balls collide?

The descriptions produced by FROB contain explicit answers to the first two questions. In particular, what a ball does next can be discovered by looking at the NEXT-MOTION cell for that state if the Action Sequence exists, or at the motion types of the SEQ nodes listed as the AFTERs of the node that corresponds to the state in question. Where a ball is at some instant can be discovered precisely by examining the Action Sequence, more vaguely by examining the path of Qstates that corresponds to the Action Sequence, and where it might be by examining a Sequence Graph computed from the state in question. The consistency of quantitative parameters and certain qualitative assumptions about motion are enforced within the individual descriptions.

Using these descriptions of motion to answer the last two questions is the subject of this chapter. We will first examine the mechanics of linking the Action Sequence and the Sequence Graph by means of a path through the SEQ nodes first. Summarizing simple global properties of motion (question 3) will be discussed next. Finally the problem of detecting collisions (question 4) will be addressed.

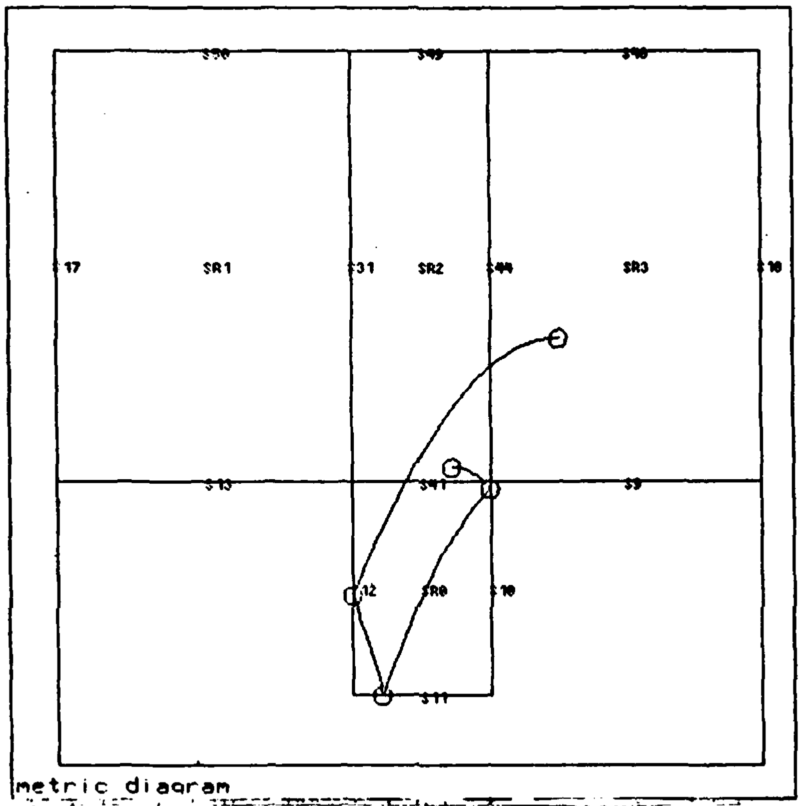
5.1 Qualitative and Quantitative Interactions

Every motion that is possible from some state can be described qualitatively by a sequence of Qstates that corresponds to a path in a Sequence Graph computed for that initial state. It is important to map the description provided by the Action Sequence into such a path to check that the actual motion reflects the assumptions made about it. Comparing the symbolic places in two Qstate paths also provides a filter for collision possiblities that is simpler than intersecting specific trajectories.

Mapping from an Action Sequence into a Qstate path is simple. Explicit in the description of each act are the type of motion and the direction which form two of the three components of the Qstate. The places in the Space Graph that correspond to a point or a trajectory can be determined by the parity and intersection capabiiities of the Metric Diagram. Rules that carry out this process are attached to each motion constraint. Figure 43 shows the Qstate path generated by a particular trajectory.

To check the assumptions about motion, the Qstate path is matched against the Sequence Graph for the initial state of the ball. Any dead SEQ nodes that correspond to a Qstate in the path signal an inconsistency in the assumptions. Assumptions about required Qstates or places are harder to check, since the Qstate or place might occur in a portion of a path that is not yet computed. A Sequence Graph for the last PHYSOB in the Action Sequence can be computed and inspected to see if the required state or place can still occur in the future. The current implementation of FROB does not create new

Fig. 43. Qstate trajectory description
Any Action Sequence can be described as a path of Qstates through the Sequence Graph.



```
 THE STATE TRAJECTORY IS
(FLY (SREGION3) (LEFT))
(FLY (SREGION3) (LEFT DOWN))
(PASS (SEGMENT44) (LEFT DOWN))
(FLY (SREGION2) (LEFT DOWN))
(PASS (SEGMENT41) (LEFT DOWN))
(FLY (SREGION0) (LEFT DOWN))
(COLLIDE (SEGMENT12) (LEFT DOWN))
(FLY (SEGMENT12) (RIGHT DOWN))
(FLY (SREGION0) (RIGHT DOWN))
(COLLIDE (SEGMENT11) (RIGHT DOWN))
(FLY (SEGMENT11) (RIGHT UP))
(FLY (SREGION0) (RIGHT UP))
(COLLIDE (SEGMENT10) (RIGHT UP))
(FLY (SEGMENT10) (LEFT UP))
(FLY (SREGION0) (LEFT UP))
(PASS (SEGMENT41) (LEFT UP))
(FLY (SREGION2) (LEFT UP))
(FLY (SREGION2) (LEFT))
```

Sequence Graphs to check required Qstate or place assumptions.

5.2 Motion Summaries

Summarization makes explicit certain global features of a description that are implicit in its original form. The Qstate path is a summarization of the Action Sequence, and it in turn could be summarized by parsing it into patterns of repetitive motion. This section deals with summarizing the Sequence Graph to describe what a ball can eventually do. Without knowing details, the only two things that can be said about the eventual state of a ball is whether or not it might stop and where it might be.

A ball will stop moving only if it loses energy. If we ignore the possibility of a surface being struck at grazing incidence to produce a SLIDE, the elasticity of a ball is the sole factor that determines whether or not a ball will move forever. The places it can stop at are simply those places in the Space Graph that have a live SEQ node whose motion type is SLIDE/STOP or STOP.

A ball will either leave the space enclosed by the diagram while it moves or stay inside it forever. Leaving the diagram is possible at the places named in the live SEQ nodes whose motion type is CONTINUE. A ball that does not leave the diagram either stops within it, bounces around forever inside some portion of it, or is trapped inside a well. Being trapped in a well can be detected by the existence of a live SEQ node within the places that comprise the well and no live SEQ nodes in places outside it.

Figures 44, 45, and 46 show examples of the summaries FROB produces.

5.3 Collisions

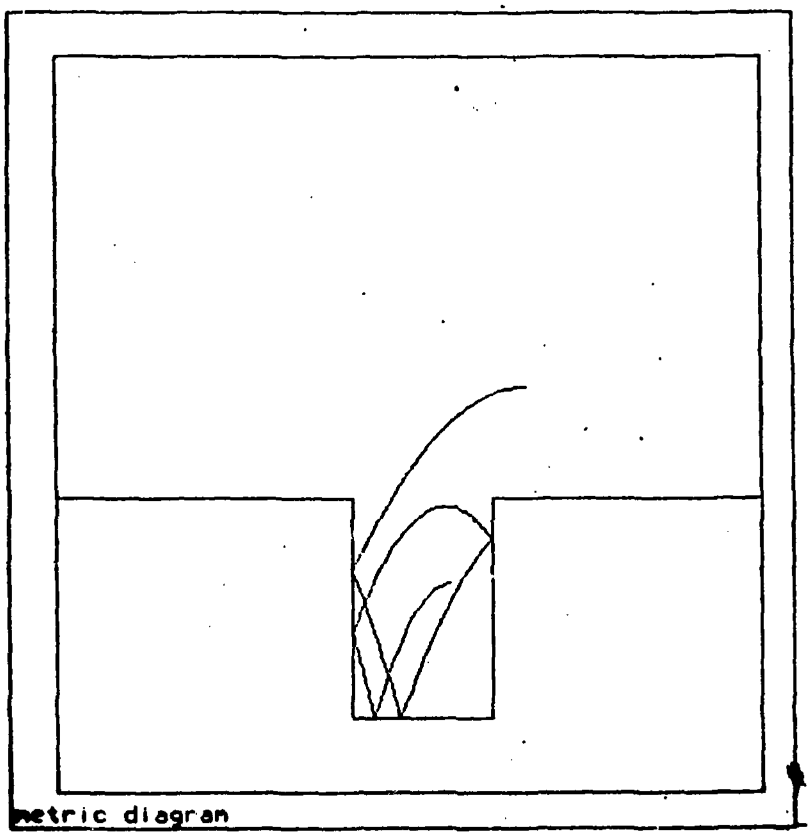
"In a world of cause and effect, all coincidences are suspect."
-Nero Wolfe

A collision occurs when two objects are in the same place at the same time. Even with incomplete information we can often decide that a collision is not possible, simply because one motion is finished before another is begun, or that the two moving objects are never in the same region of space. If there is an overlap in either space or time, simulation can determine whether or not a collision actually does occur during that period. Since there are several different resolutions for the position of a ball (the places marked in its sequence graph, the Qstate path describing its motion thus far, and the actual trajectory it takes), the program can often decide there is no place for a collision to occur long before it deals with the constraint imposed by time.

The most general test is to check the places occupied by the live Qstates of the Sequence Graphs computed from the initial states for each ball. If there is no place that has a (live) Qstate from each ball, the two balls cannot collide. Also, if they do overlap but one ball is always to the left of the other (determined by knowing the initial geometry and the directions of all the possible motions) they again cannot collide.

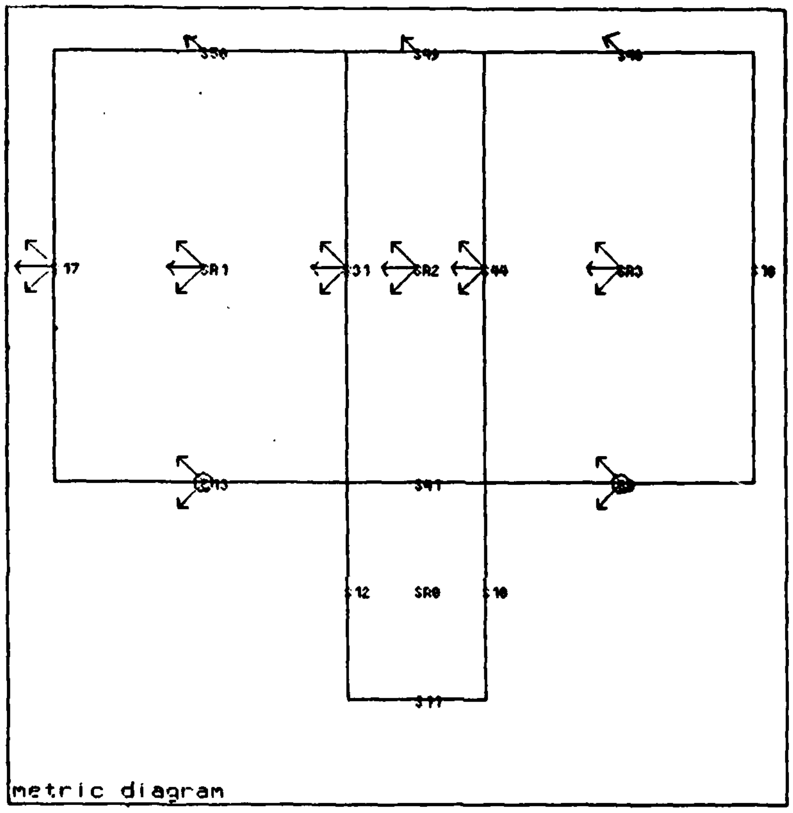
If a collision is still possible, the Action Sequences for the balls must be checked. The times of

Fig. 44. Motion summary 1
Bounds on a ball's future location can often be computed.



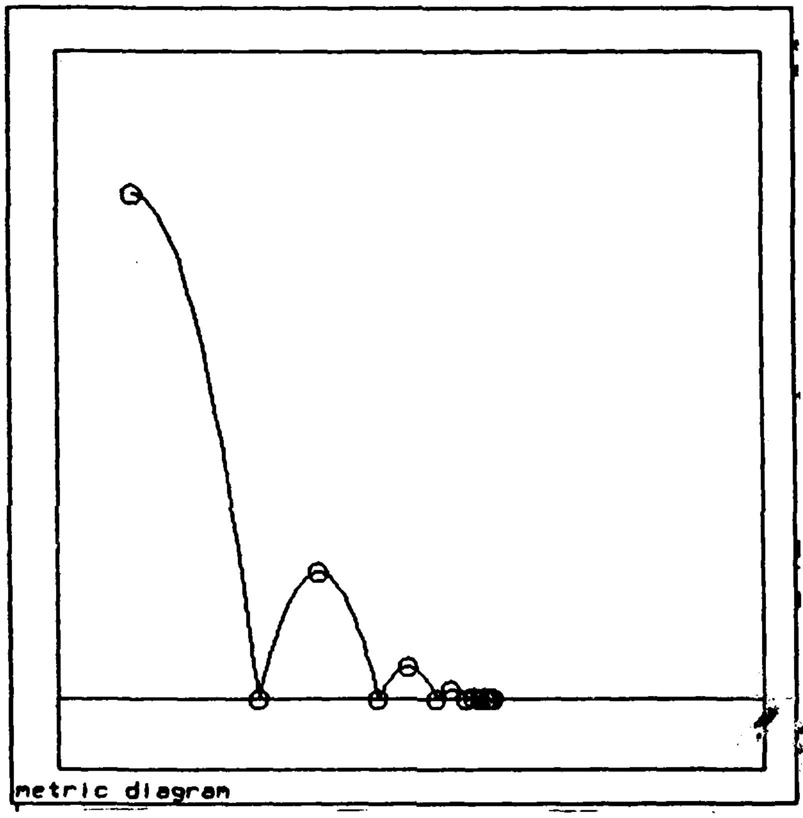
```
->>(motion-summary-for b1)

 FOR G0364
 THE BALL WILL EVENTUALLY STOP
 IT IS TRAPPED INSIDE (WELL0)
 AND WILL STOP FLYING AT ONE OF (SEGMENT11)
NIL
```

Fig. 45. Motion summary 2
Being unable to enter the well prevents the ball from moving to the right.



```
(describe-it (>> assumptions fred))

 (>> ASSUMPTIONS FRED) = G0802
 G0802 IS THE SET OF MOTION ASSUMPTIONS FOR (>> INITIAL-STATE FRED)
 THE ROOT OF ITS SEQUENCE GRAPH IS SEQ0
 EXCLUDED PLACES ARE (SEGMENT41)
G0802
->>(what-is (>> state initial-state fred))
(>> STATE INITIAL-STATE FRED) = (FLY (SREGION3) (LEFT))
NIL
->>(motion-summary-for fred)

 FOR G0725
 IT IS UNCLEAR WHETHER OR NOT IT WILL STOP
 IT MIGHT LEAVE THE DIAGRAM AT (SEGMENT48 SEGMENT17 SEGMENT49 SEGMENT50)
 OR STOP FLYING AT ONE OF (SEGMENT9 SEGMENT13)
NIL
```

Fig. 46. Motion summary 3
Sometimes the final state of the motion is known.



```
->>(motion-summary-for phob)

 FOR G0261
 THE LAST THING IT DOES IS AMBIGUITY-SLIDE/STOP
 AT POSITION (2.051562846 . -8.0) AND TIME UNKNOWN
NIL
```

each act are inspected to find out which of them have intervals that are simultaneous. Should none overlap and the Action Sequences are complete, no time coincidence can exist, so no collision can occur. Those which overlap in time are checked to see if their Qstate paths share places. If they do, the trajectories are intersected. If an intersection point is found, the motions are copied but with new PHYSOBs to describe the state afterwards. The intersection point is used as the geometry of the two new AFTERs. A collision occurs if the times on these PHYSOBs are the same. The failure of any of these steps rules out a collision at that particular point, but all other possibities must be checked.

If there are no collisions found by the comparison of the Action Sequences, it either means that no collision occurs, or that simulation hasn't been carried out yet to the point where one does. If the Action Sequences involved are incomplete, a Sequence Graph is made of the last PHYSOB in each sequence. The Sequence Graph for the earlier PHYSOB is compared against the Sequence Graph of the later one, plus the Qstate path from the time associated with the earlier PHYSOB and the later one. If there is no spatial coincidence, there can be no collision. Otherwise one is possible at the places which overlap. The process described here is illustrated in figure 47.

Figures 48, 49, and 50 show the answers the program gives for some typical problems.

Fig. 47. Collision detection can involve several descriptions

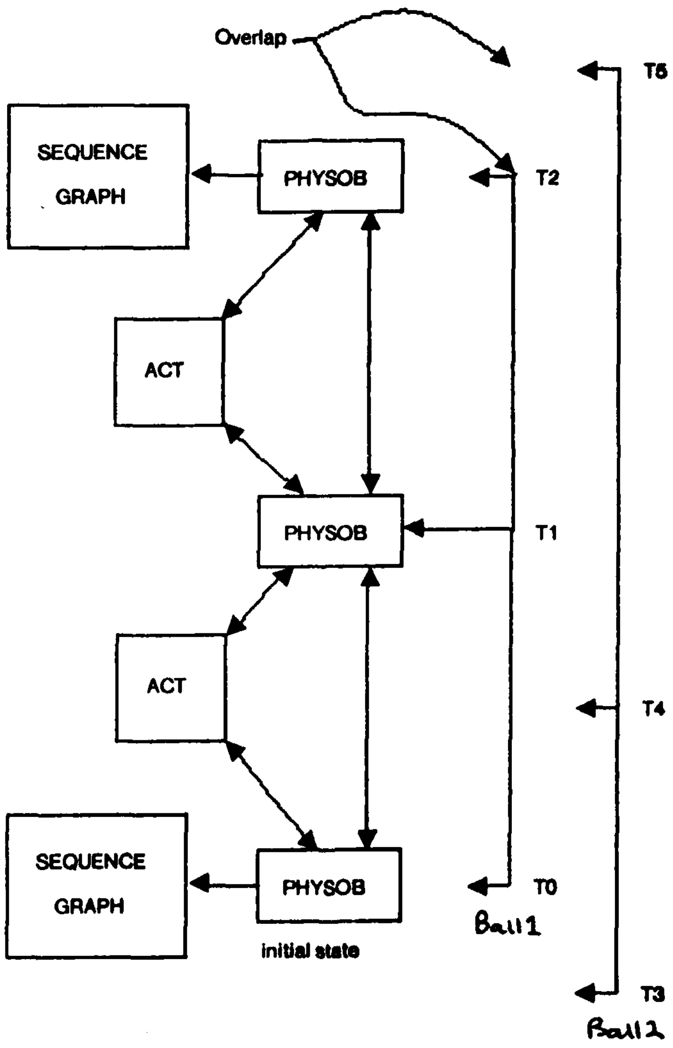
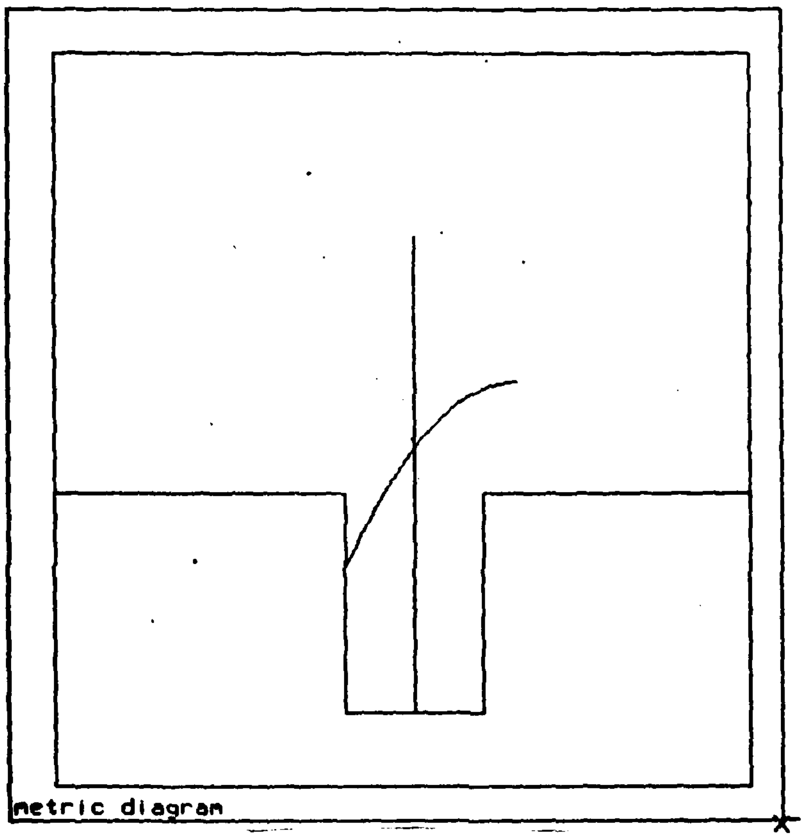
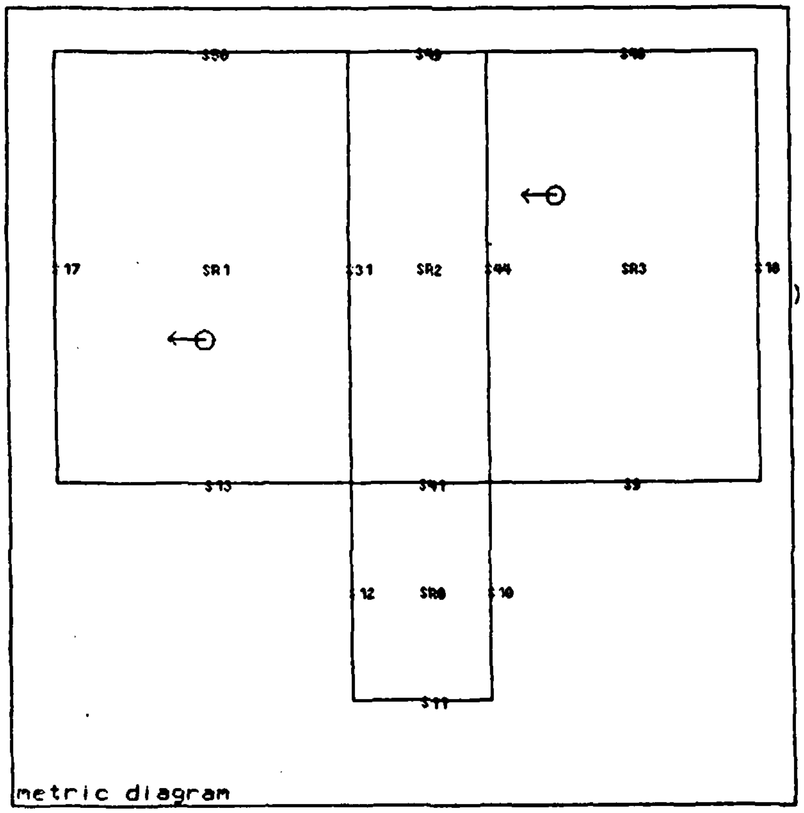


Fig. 48. Collision detection 1
If enough information is known, the exact time and position of a collision can be determined.



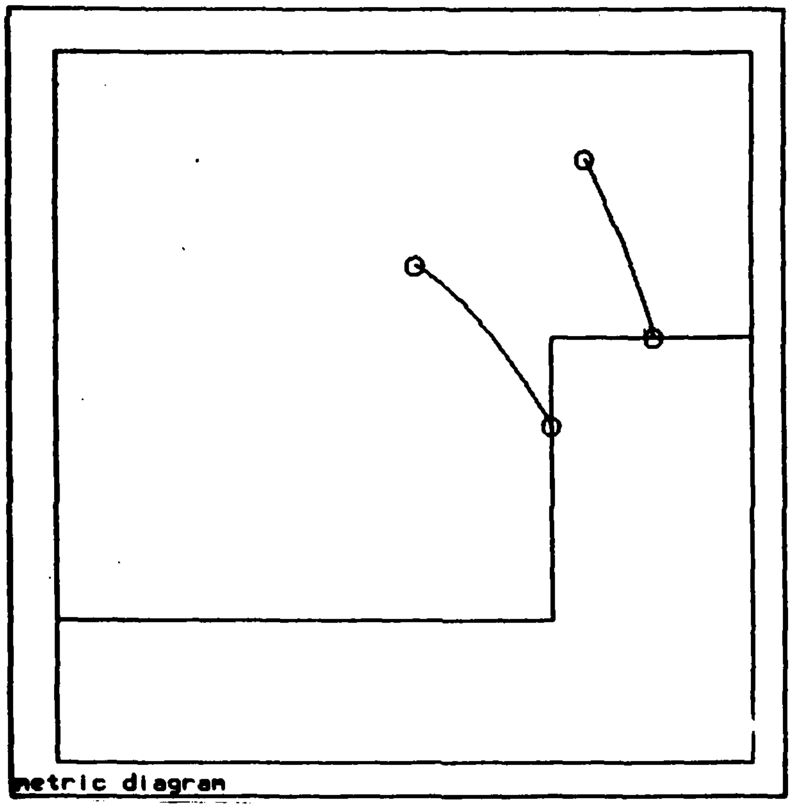
```
->>(collide? b0 b1)
 (>> ACTION-SEQUENCE B0) AND (>> ACTION-SEQUENCE B1) GO IN THE SAME PLACES
 CANNOT PLACE ENDPOINTS - PARABOLA2
 BUT THEY DO NOT COLLIDE
(POSSIBLE AT SEGMENT36 SEGMENT11 SREGION2 SREGION0)
->>(change-parameter (>> time initial-state b1) 0.0875291242)
0.0875291242
->>(collide? b0 b1)
 (>> ACTION-SEQUENCE B0) AND (>> ACTION-SEQUENCE B1) GO IN THE SAME PLACES
 CANNOT PLACE ENDPOINTS - PARABOLA3
 COLLISION AT ((1.170334457e-7 -0.7963998606)) AND 0.6875291006 SECONDS
(((1.170334457e-7 -0.7963998606)) . 0.6875291006)
->>
```

Fig. 49. Collision detection 2
If two balls can never be in the same place they cannot collide.



```
->>(collide? fred george)
(POSSIBLE AT SEGMENT50 SEGMENT17 SEGMENT13 SREGION1)
->>(cannot-be-at fred segment31)
(SEGMENT31)
 UPDATING ASSUMPTIONS FOR (>> INITIAL-STATE FRED)
 CHECKING PATH OF MOTION AGAINST ASSUMPTIONS
->>(collide? fred george)
NO
->>(what-is (>> state initial-state fred))
(>> STATE INITIAL-STATE FRED) = (FLY (SREGION3) (LEFT))
NIL
->>(what-is (>> state initial-state george))
(>> STATE INITIAL-STATE GEORGE) = (FLY (SREGION1) (LEFT))
NIL
```

Fig. 50. Collision detection 3
A small amount of simulation can often rule out a collision.



```
->>(collide? fred george)
NO
->>(motion-summary-for fred)

 FOR G2834
 THE BALL WILL EVENTUALLY STOP
 IT MIGHT LEAVE THE DIAGRAM AT (SEGMENT30 SEGMENT29)
 OR STOP FLYING AT ONE OF (SEGMENT9)
NIL
->>(motion-summary-for george)

 FOR G2948
 THE BALL WILL EVENTUALLY STOP
 IT MIGHT LEAVE THE DIAGRAM AT (SEGMENT14)
 OR STOP FLYING AT ONE OF (SEGMENT7)
NIL
```

6. Concluding Remarks

6.1 Summary

This work has studied the role of qualitative and geometric knowledge in reasoning about motion. The issues were explored by building a program, FROB, that could reason about motion in a simplified domain. This section summarizes what was learned.

The knowledge embedded in FROB constitutes a qualitative theory of the FLY case of motion. As such it is a step towards a better formal understanding of common sense reasoning about motion. However, I believe that the program illustrates ideas that are far more general.

The first and most important of these ideas concerns spatial reasoning. If metric properties for the spatial aspects of a problem are provided, simple methods can be used to decide certain spatial questions. For people this process involves drawing a diagram on a piece of paper and interpreting what they see. For FROB the process involves calculation within the Metric Diagram.

Qualitative reasoning about space requires dividing space into distinct places. Different kinds of reasoning may impose different sets of places on a diagram. In the Bouncing Ball world a physical constraint (gravity) and a computational constraint (simplify the description of possible motions) define a non-overlapping maximal place set for space whose elements can be composed to describe any other place required. Any non-overlapping place set will allow space to be described as a graph. The graph structure makes the creation of new places by composition easy and provides a framework for computing qualitative descriptions.

Different descriptions of the same situation must have a common vocabulary for communication. The constituents of the qualitative state perform this function in the Bouncing Ball world Embedding the place description in the diagram and decomposing quantitative descriptions into qualitatively distinct acts provides the common vocabulary in FROB.

The effects of assumptions about a ball and its motion can be captured by the Sequence Graph. The assumptions rule out certain states of motion so they can be pruned from the Sequence Graph. Further nodes may have to be pruned to satisfy certain constraints on motion. These constraints include the continuity of motion through space and the fact that velocities cannot be mathematically infintesimal.

Qualitative knowledge provides a structure for using the equations of motion. Distinct types of motion can be represented as computational objects and linked by descriptions of a ball's state at a particular instant of time to form networks that describe motion. Implementing these objects in a constraint language enables them to be used for simulation and analysis.

6.2 Psychological Implications

FROB captures the knowledge necessary to answer a number of questions about the Bouncing Ball domain that people find easy to answer. However, this does not imply that it uses the same knowledge or as much knowledge as humans have about the domain.

One aspect of human understanding that is missing in FROB is significance. Understanding the significance of a piece of knowledge implies the ability to relate it to other things you know. There is no other interpretation of the terms used in FROB's representations below the level of the processes that directly manipulate them, nor are they a part of a larger corpus of knowledge. This makes FROB inflexible. For example, a person would understand that the only impact on his knowledge of halving the value of the gravitational constant would be that the value used in the equations of motion must be changed accordingly. He would understand that if gravity varied in magnitude with time the equations of motion would become more complex and that if the sign varied as well his qualitative rules of motion would require revision.

The idea behind the Metric Diagram is to require specific values as parameters of elements in a geometric representation so that spatial questions can be decided by simple techniques. Using a representation of geometry with metric properties makes sense for people because they have evolved visual hardware for perceiving a world made of objects with specific properties. The very different structures available for computation in people and machines means the algorithms and implementations are probably quite different, but the principle is still the same.

The description people give of possible motions is quite different than the description computed by FROB. For envisionment people divide space into places that are much larger than the places that are elements of the Space Graph. While the resulting description is very small, it can be expanded on demand. This suggests people use a hierarchial description of places[1], with the Space Graph FROB uses corresponding to the most detailed level of the hierarchy.

Instead of pruning a description of possible motions to reflect qualitative assumptions, people appear to create a new description using the assumptions to guide and limit the generation process. This limits memory requirements at the cost of increasing the time required to figure out why something didn't happen.

The decomposition of motion represented by the Action Sequence is very natural, both for using equations and as a way of describing motion. I believe a structure like the Action Sequence is the target representation that quantitative data about motion (obtained perhaps from interpreting perception) is mapped into, so that qualitative constraints and assumptions about it can be checked. However, the actual use of equations in FROB is most unnatural. Unlike FROB, people usually do not compute every numeric parameter possible from a set of information. People can also manipulate equations of motion to generate algebraic solutions and are capable of interpreting equations in a qualitative fashion, neither of which FROB can do.

1. An example is [McDermott 74], which used a tree of places to represent space.

6.3 Future work

We are a long way from understanding the human fluency in dealing with space. If our machines are ever to exhibit common sense this situation must change. This work raises several topics whose investigation would be steps in that direction.

Reasoning about geometry needs to be smoothly integrated with other kinds of reasoning. One step in this integration is to make the processes that compute qualitative space descriptions and decide spatial questions keep dependency information. Several advantages would accrue from this. Geometric analysis could be incremental, thus facilitating the use of several sources for geometric information, perhaps someday even perception. Creating a Metric diagram from a relational description also requires the capability to identify the source of inconsistent properties in the diagram. Incremental detection of possible collisions would be simpified because each trajectory could be marked with an assumption that no other trajectories intersected it. Asserting a new trajectory could cause these assumptions to be checked.

While the Metric Diagram representation seems to be the most productive line of research, I would not suggest abandoning the investigation of other geometric representations. Improvements in relational geometric systems will be required to interface with more symbolic knowledge. The possible computational advantages of a NETL like array structure (cf section 2.4) should also be investigated.

Further studies in common sense physics are important in their own right, as well as being good domains for studying reasoning about space. There are still no qualitative theories for SWING or ROLL classes of motion. A particularly interesting subset of ROLL would be the Billard Ball world. Apart from the interesting issues of planning, the dominance of moving collisions would necessitate a better description of them.

One important use of common sense knowledge about motion is to tell us when observations of a situation conflict with the model we have of it. This requires using the qualitative vocabulary of motion to understand quantitative data from outside sources. The qualitative consistency checks people apply to quantitative data need to be explicated.

Mechanical systems also require interesting reasoning. Strings, springs, and shape must eventually be dealt with. The Metric Diagram could be extended to include hierarchial descriptions of objects as a way of dealing with this added complexity. For example, a string could be represented by a region of space at the roughest level of description and by a connected set of arcs at the finest. The literature on spatial modelling in robotics is a likely source of relevant ideas on dealing with shape.

Reasoning about motion is only one of many problems that fall under the heading of spatial reasoning. Other types of problems include navigation, knot typing, and mental imagery. The terrain is mostly uncharted, but we must press on from our small clearing.

7. Bibliography

[BKPH,Vismem]
Horn, Berthold K.P., *VISMEM: A Bag of Robotics Formulae*
in Progress in Vision and Robotics, MIT AI TR-281, P.H. Winston, editor
[BOBERG]
Boberg, Richard W., *Generating Line Drawings from Abstract Scene Descriptions*
Ph.D Thesis, MIT AI Lab, December 1972
[BUNDY]
Bundy, A., Luger, G., Stone, M. and Welham, R., *MECHO: Year one*
DAI Research Report No. 22, Edinburgh, 1976
[BUNDY, SLIDE]
Bundy, A., *Will it Reach the Top? Prediction in the Mechanics World*
Artificial Intelligence, Vol. 10 No. 2, April 1978
[DEKLEER 75]
deKleer, Johan *Qualitative and Quantitative Knowledge in Classical Mechanics*
MIT AI Lab TR-352, December 1975
[DEKLEER 79]
deKleer, Johan *Causal and Teleological Reasoning in Circuit Recognition*
MIT AI Lab, Ph.D. Thesis, January 1979
[DOYLE]
Doyle, Jon *Truth Maintenance Systems for Problem Solving*
MIT AI Lab TR-419, January 1978
[FAHLMAN 73]
Fahlman, Scott E. *A Planning System for Robot Construction Tasks*
MIT AI Lab TR-283, May 1973
[FAHLMAN 79]
Fahlman, Scott E. *NETL: A System for Representing and Using Real-World Knowledge*
MIT Press, Cambridge
[FUNT 76]
Funt, B. V. *WHISPER: A Computer Implementation Using Analogues in Reasoning*
PH.D. Thesis, University of British Columbia, 1976
[GELERNTER]
Gelernter, H. *Realization of a Geometry-Theorem Proving Machine*
in ComputersandThought McGraw-Hill Inc, 1963
[HAYES a]
Hayes, Patrick J. *The Naive Physics Manifesto*
unpublished, May 1978
[HAYES b]
Hayes, Patrick J. *Naive Physics I: Ontology for Liquids*
unpublished, August 1978
[HINTON 79]
Hitton, Geoffrey *Some Demonstrations of the Effects of Structural Descriptions in Mental Imagery*
Cognitive Science, Vol. 3, No. 3, July-September 1979
[HOLLERBACH 75]
Hollerbach, John M. *Hierarchial Shape Description of Objects by Selection and Modification of*

Prototypes
MIT AI Lab TR-346, November 1975
[KOSSLYN & SHWARTZ]
Kosslyn, Stephen M. and Schwartz, Steven P. *A Simulation of Visual Imagery*
Cognitive Science, Vol. 1, No. 3, July 1977
[LONDON]
London, Phil, *A Dependency-Based Modelling Mechanism for Problem Solving*
Computer Science Department TR-589, University of Maryland, November 1977
[MARR & NISHIHARA]
Marr, David and Nishihara, H. Keith *Representation and Recognition of the Spatial Orgainization of three-dimensional Shapes*
Proc. R. Soc. Lond., B. 200, 269-294 (1978)
[MASON]
Mason, Matthew T., *Qualitative Simulation of Swine Production*
MIT S.B. Thesis, May 1976
[McALLESTER]
McAllester, David A., *A Three Valued Truth Maintence System*
MIT AI Lab Memo No. 473
[MCDERMOTT & LARKIN]
McDermott, John and Larkin, Jill H., *Re-representing Textbook Physics Problems*
in Proceedings of the Second National Conference of the Canadian Society for Computational Studies of Intelligence
Toronto, 1978
[NOVAK]
Novak, G.S. *Computer Understanding of Physics Problems Stated in Natural Language*
Technical Report NL-30
Computer Science Department, The University of Texas at Austin, 1976
[STEELE & SUSSMAN]
Steele, Guy Lewis and Sussman, Gerald Jay *Constraints*
MIT AI Lab Memo No. 502, November 1978
[SUSSMAN & STALLMAN]
Stallman, Richard M. and Sussman, Gerald Jay *Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis*
MIT AI Lab Memo No. 380, September 1976
[TLP 76]
Lozano-Perez, Tomas *The Design of a Mechanical Assembly System*
MIT AI Lab TR-397, December 1976
[WALTZ & BOGGESS]
Waltz, D.L. and Boggess, L. *Visual Analog Representations for Natural Language Understanding*
Proceedings of the Sixth International Joint Conference on Artificial Intelligence, Tokyo 1979
[WINSTON 70]
Winston, Patrick H. *Learning Structural Descriptions From Examples*
MIT AI Lab, TR-231 September 1970

8. Appendix 1 - FROB Scenario

This section contains the trace of an interaction with FROB. It is included to give the reader some idea of what the actual implementation is like without resorting to the inclusion of a listing of the code. The capabilities shown here are a subset of what the program is actually capable of.

FROB is written in MACLISP, a dialect of LISP. It was originally used on the MIT AI lab's KA-10, and now runs on the CADR Lisp Machines designed at the laboratory. Using a CADR is important because the descriptions created by the program can be quite large, well over the 256K addressing limits of a PDP-10.

For this trace, input to the program is in lower case, while the program's responses are usually in upper case. Comments on the proceedings are in a different script.

First we start the program.

```
(frob)
->>(start-new-diagram)
```

the "->>" is the top level prompt. Creating, modifying, and interrogating the descriptions in FROB is accomplished by calling functions written in LISP. The function just called prepares the program for information about a new scene.

```
->>file-read
 FILE NAME?((dsk kdf) scene2 >)
```

This file contains the specifications for the surfaces in the scene. It was created with a simple graphics system and dumped out as a file for future use. Its contents are contained in figure 51.

```
SCENE-2
```

A number of Metric Diagram elements and constraints are created to represent the scene. What is printed during this process are the internal names of the symbols created to represent these entities. These internal names are uninteresting and have been excised from the transcript.

```
->>(draw-scene)
 |No fly-simulates to draw on|
```

The Metric Diagram specified thus far is drawn in figure 52. This drawing does not show the interpretations of the elements, nor does it show which side of the surface is considered to be free space and which solid. From now on, the commands invoked to display the diagram will not be shown

```
->>(diagram-done)
SCENE-2
 COMPUTING SPACE GRAPH FOR SCENE
```

Once all the surfaces have been specified, the areas of the diagram that are interpreted as solid are isolated and a qualitative description of free space is computed. There is of course only one solid region in this scene, shown in figure 53.

PRECEDING PAGE BLANK-NOT FILMED

Fig. 51. Contents of SCENE2

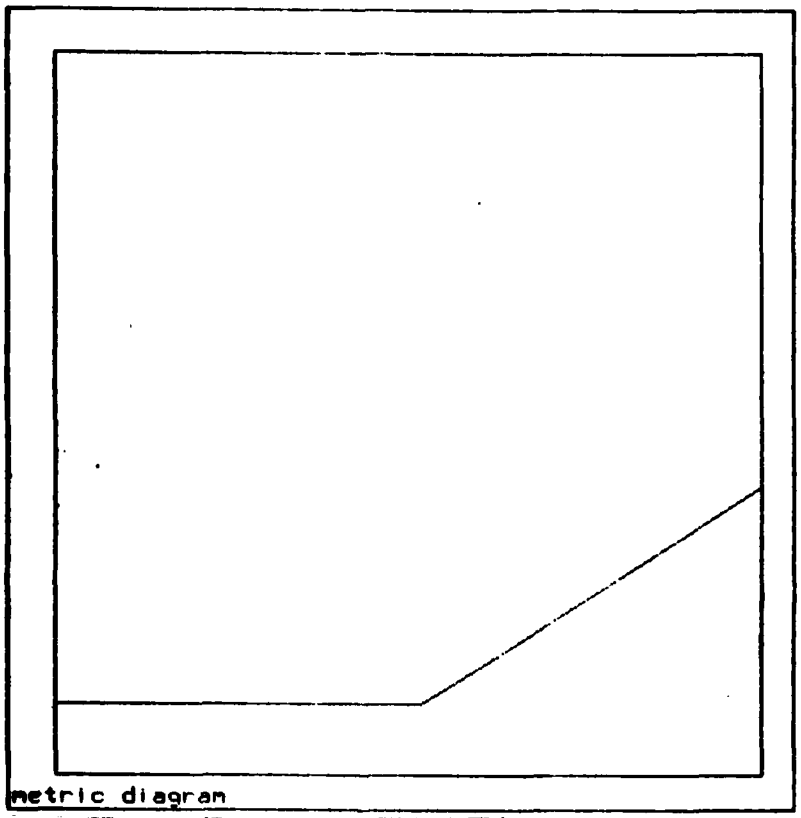
```
;-*-lisp-*-

(new-diagram 'scene-2)
(create floor 'surface)
(create incline 'surface)
(create corner0 'corner)
(create corner1 'corner)
(create corner2 'corner)

(set-parameter (>> first-corner floor) corner0)
(set-parameter (>> second-corner floor) corner1)
(set-parameter (>> first-corner incline) corner1)
(set-parameter (>> second-corner incline) corner2)
(set-parameter (>> first-connection corner0) nil)
(set-parameter (>> second-connection corner0) floor)
(set-parameter (>> first-connection corner1) floor)
(set-parameter (>> second-connection corner1) incline)
(set-parameter (>> first-connection corner2) incline)
(set-parameter (>> second-connection corner2) nil)
(set-parameter (>> geometry corner0) (create-point -10.0 -8.0))
(set-parameter (>> geometry corner1) (create-point 0.0 -8.0))
(set-parameter (>> geometry corner2) (create-point 10.0 -2.0))
(set-parameter (>> geometry floor)
               (create-segment (value? (>> geometry corner0))
                               (value? (>> geometry corner1))))
(set-parameter (>> geometry incline)
               (create-segment (value? (>> geometry corner1))
                               (value? (>> geometry corner2))))
(set-parameter (>> solid-side floor) '+)
(set-parameter (>> solid-side incline) '+)

;add the interpretations
(mapc '(lambda (geoms)
               (putprop (value? (>> geometry geoms))
                        geoms
                        'interpretation))
      (list corner0 corner1 corner2 floor incline))

(create the-scene 'scene)
(set-parameter (>> diagram-name the-scene) current-diagram)
```

The qualitative description of space is called the Space Graph. Each node in it is a Metric Diagram element that has been annotated with adjacency information as well as a specific interpretation. Figure 54 contains a labelled drawing of the Space Graph elements for the scene

"SR" means SREGION and "S" means "SEGMENT". The class of an element is its interpretation. Here are some constituents of the current Space Graph -

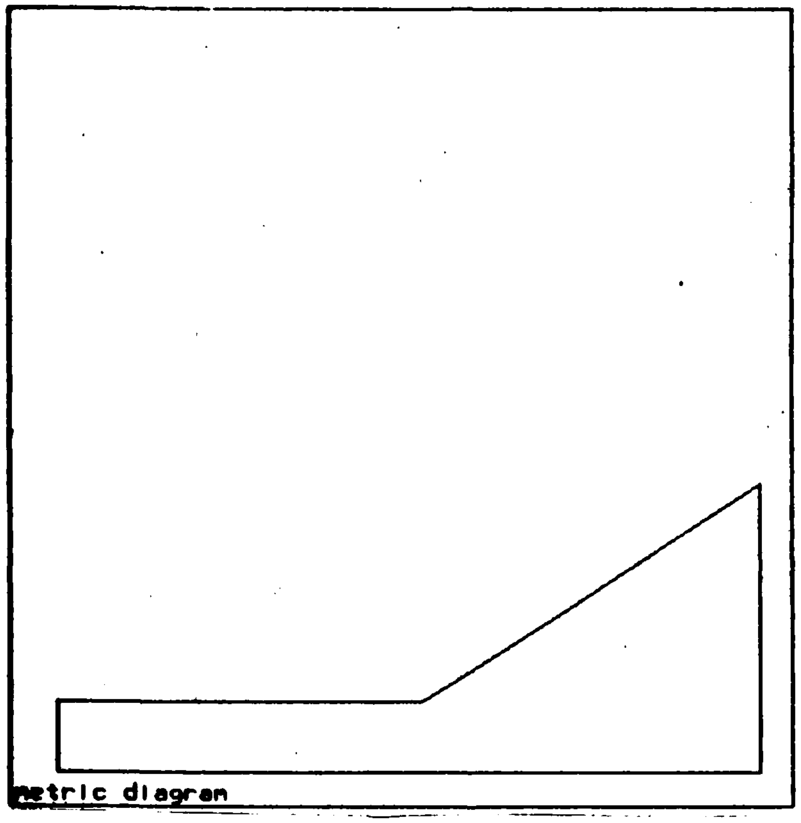
```
SREGION3
POSSIBLE-WELL? NIL
CONNECTED-SET (SREGION2 SREGION3)
MEMBER-OF *S-REGIONS*
CORNERS (POINT34 POINT17 POINT11)
BOUNDARY (SEGMENT27 SEGMENT29 SEGMENT6)
DOWN SEGMENT6
RIGHT SEGMENT6
UP SEGMENT29
```

Fig. 52. The Scene



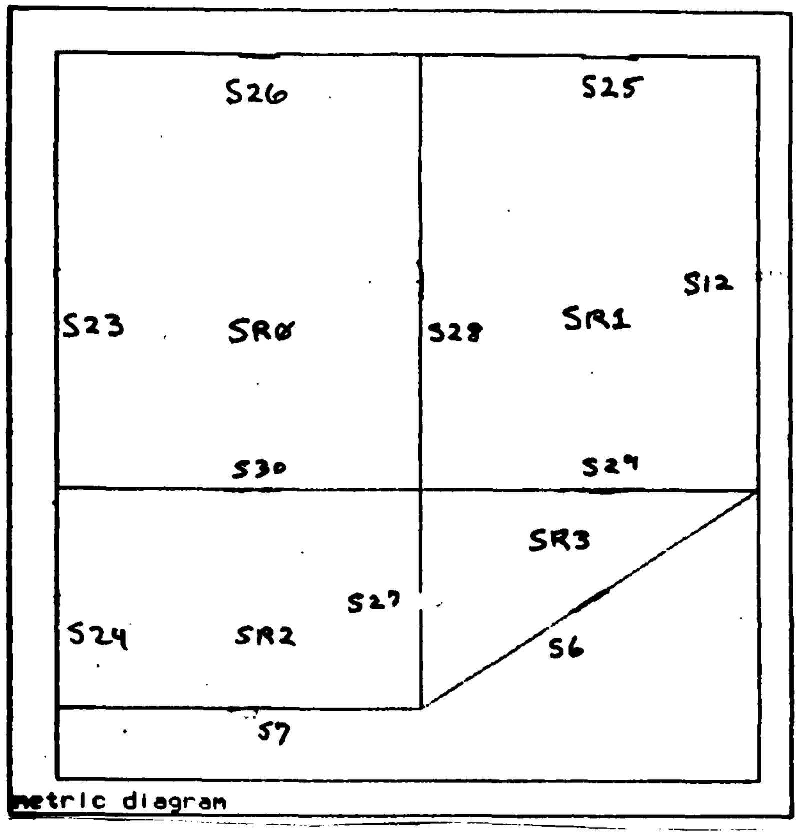
```
     LEFT SEGMENT27
     TYPE REGION
     CLASS SREGION



     SEGMENT6
     CONNECTING-REGION SREGION3
     MEMBER-OF *SPACE-GRAPH*
     PART-OF
((BOUNDARY . SREGION3) (BOUNDARY . REGION0) (BOUNDARY . REGION0))
     UP SREGION3
     LEFT SREGION3
     CLASS SURFACE
     END2 POINT17
     END1 POINT11
     COMES-FROM SEGMENT6
     PERPENDICULAR-EQUATIONS
((-0.8574929 0.5144957 -4.1159659) (-0.8574929 0.5144957 7.5459377))
     UNIT-NORMAL (0.5144957555 -0.8574929256)
     UNIT-VECTOR (0.8574929256 0.5144957555)
```

Fig. 53. Solid Region



```
TYPE SEGMENT
EQUATION (0.5144957555 0.8574929256 -6.859943405)


SEGMENT27
MEMBER-OF *SPACE-GRAPH*
PART-OF ((BOUNDARY . SREGION3) (BOUNDARY . SREGION2))
RIGHT SREGION3
LEFT SREGION2
CLASS FREE
COMES-FROM SEGMENT19
INTERPRETATION NIL
PERPENDICULAR-EQUATIONS ((0.0 1.0 -8.0) (0.0 1.0 -2.0))
UNIT-NORMAL (1.0 0.0)
UNIT-VECTOR (0.0 1.0)
TYPE SEGMENT
EQUATION (1.0 0.0 0.0)
END2 POINT34
END1 POINT11
```

Fig. 54. Space Graph



```
->>(create phob 'ball)
G0277
```

A constraint representation of a ball has now been created and given the name "phob" This constraint holds assumptions about the ball's motion, pointers to a description of its actual motion, and a representation of its initial state.

```
->>(set-parameter (>> time initial-state phob) 0.0)
0.0
```

We will be arbitrary about the time, and ignore units.

```
->>(set-parameter (>> position initial-state phob) '(sregion1))
(SREGION1)
```

The Space Graph provides a vocabulary for qualitative descriptions of position

```
->>(set-parameter (>> direction initial-state phob) '(left down))
(LEFT DOWN)
->>(set-parameter (>> next-motion initial-state phob) 'fly)
FLY
```

The qualitative description of the initial state is complete, and so the kinds of motion possible can be envisioned.

```
 COMPUTING SEQUENCE GRAPH FOR (>> INITIAL-STATE PHOB)
 UPDATING ASSUMPTIONS FOR (>> INITIAL-STATE PHOB)
 38 ALIVE OUT OF 38 FOR (>> INITIAL-STATE PHOB)
```

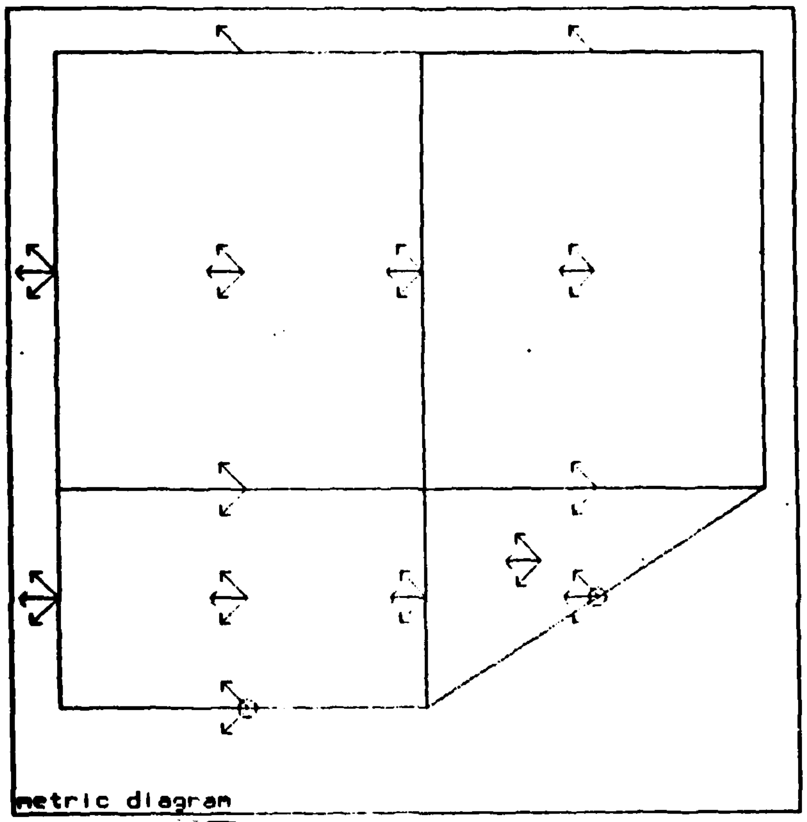
The status of the nodes in the Sequence Graph is checked whenever assumptions are changed as well as when created. Here is a printed representation of some of the SEQ nodes from the Sequence Graph. G0279 is the internal name for the initial state of phob.

```
 THIS IS THE START NODE OF THE GRAPH FOR G0279
SEQ0
(FLY SREGION1 (LEFT DOWN))
 CAN BE REACHED BY (SEQ29)
 NEXT CAN BE (SEQ1 SEQ2)

SEQ1
(PASS SEGMENT29 (LEFT DOWN))
 CAN BE REACHED BY (SEQ0)
 NEXT CAN BE (SEQ3)

SEQ2
(PASS SEGMENT28 (LEFT DOWN))
 CAN BE REACHED BY (SEQ0)
 NEXT CAN BE (SEQ4)

SEQ3
(FLY SREGION3 (LEFT DOWN))
 CAN BE REACHED BY (SEQ14 SEQ12 SEQ1)
 NEXT CAN BE (SEQ5 SEQ6)

SEQ4
(FLY SREGION0 (LEFT DOWN))
 CAN BE REACHED BY (SEQ34 SEQ2)
 NEXT CAN BE (SEQ7 SEQ8)

SEQ5
(COLLIDE SEGMENT6 (LEFT DOWN))
 CAN BE REACHED BY (SEQ3)
 NEXT CAN BE (SEQ9 SEQ10 SEQ11 SEQ12)

SEQ9
(SLIDE//STOP SEGMENT6 (LEFT DOWN))
 CAN BE REACHED BY (SEQ5)
 THIS IS A TERMINAL NODE
```

A graphical representation is far more convenient. SEQ nodes will be represented by superimposing them on the Space Graph. Nodes with a specific direction will be represented by an arrow drawn in that direction, while SLIDE/STOP, STOP, and nodes without directions are drawn as circles. Figure 55 is a drawing of the Sequence Graph for the initial state.

The Sequence Graph can be concisely summarized-

Fig. 55. Sequence Graph



```
->>(motion-summary-for phob)

 FOR G0277
 IT IS UNCLEAR WHETHER OR NOT IT WILL STOP
 IT MIGHT LEAVE THE DIAGRAM AT (SEGMENT23 SEGMENT24 SEGMENT25 SEGMENT26)
 OR STOP FLYING AT ONE OF (SEGMENT6 SEGMENT7)
NIL
->>(get sregion1 (>> initial-state phob))
(((FLY (LEFT)) . SEQ29) ((FLY (LEFT UP)) . SEQ23) ((FLY (LEFT DOWN)) . SEQ0))
```

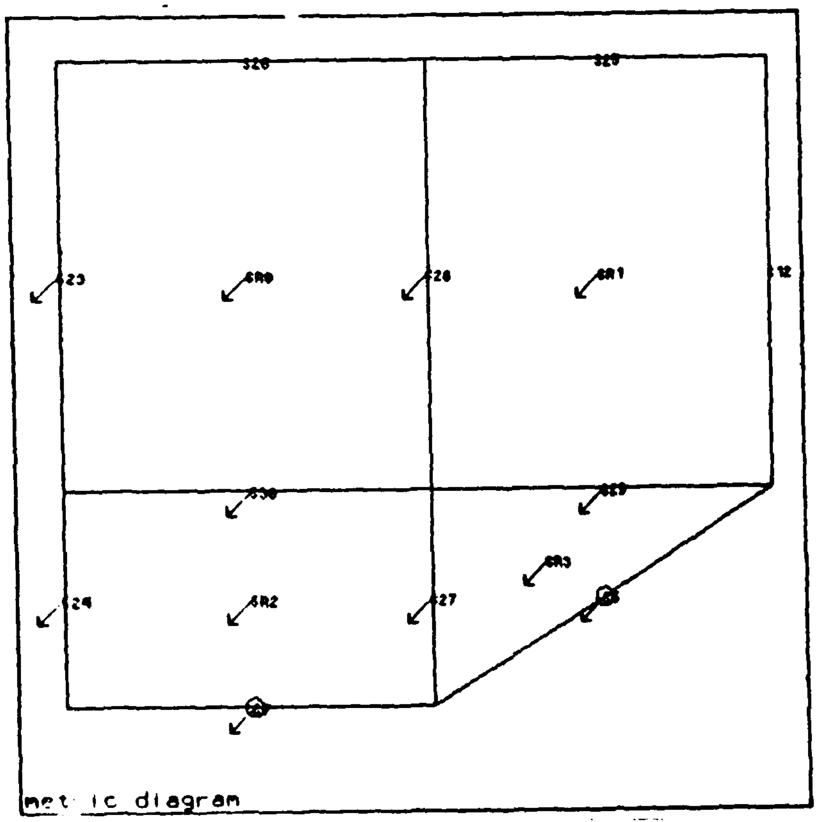
SEQ nodes are indexed in the Space Graph under the name of the place in them and the state of a ball they refer to. Adding information can only reduce the possible kinds of motion. Assume for a moment that the ball is completely inelastic.

```
->>(set-parameter (>> c-o-r initial-state phob) 0.0)
0.0
 CHECKING MOTION OF (>> PHOB)
 UPDATING ASSUMPTIONS FOR (>> INITIAL-STATE PHOB)
 14 ALIVE OUT OF 38 FOR (>> INITIAL-STATE PHOB)
 CHECKING PATH OF MOTION AGAINST ASSUMPTIONS
->>(motion-summary-for phob)

 FOR G0277
 THE BALL WILL EVENTUALLY STOP
 IT MIGHT LEAVE THE DIAGRAM AT (SEGMENT23 SEGMENT24)
 OR STOP FLYING AT ONE OF (SEGMENT6 SEGMENT7)
NIL
```

Figure 56 is a drawing of the results. All of the states of motion made possible by recoil from a surface are ruled out. Suppose the ball were perfectly elastic -

```
->>(set-parameter (>> c-o-r initial-state phob) 1.0)
1.0
 CHECKING MOTION OF (>> PHOB)
 UPDATING ASSUMPTIONS FOR (>> INITIAL-STATE PHOB)
 36 ALIVE OUT OF 38 FOR (>> INITIAL-STATE PHOB)
 CHECKING PATH OF MOTION AGAINST ASSUMPTIONS
```

Fig. 56. Sequence Graph for COR = 0.0



```
->>(motion-summary-for phob)

 FOR G0277
 THE BALL WILL CONTINUE TO MOVE
 IT WILL LEAVE THE DIAGRAM AT (SEGMENT23 SEGMENT24 SEGMENT25 SEGMENT26)
NIL
```

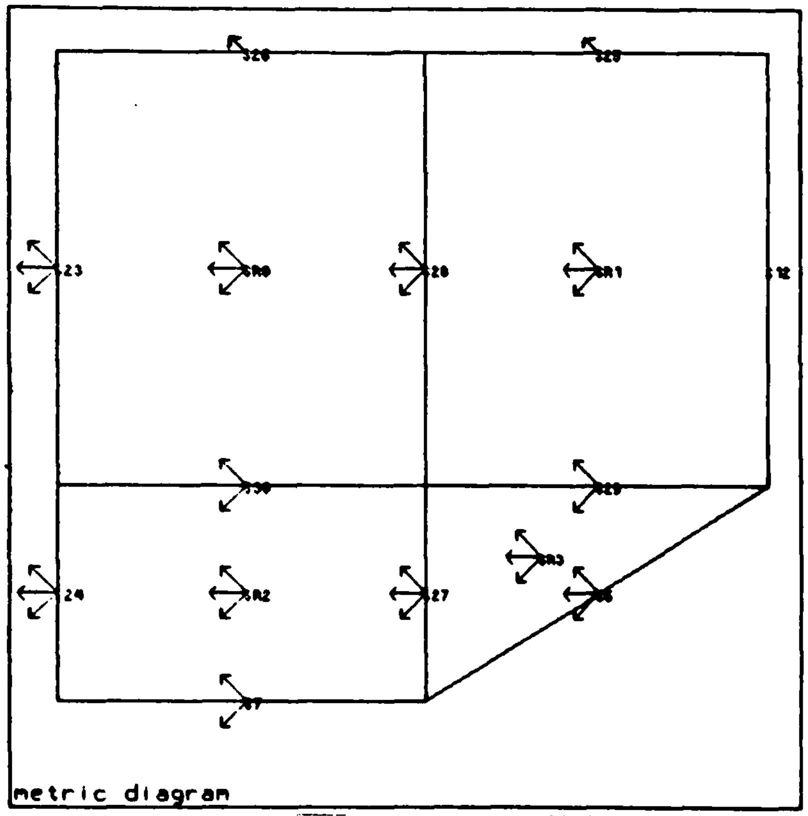
The new state of the Sequence Graph is shown in figure 57. The reason for not believing a node is maintained·

```
->>(why-not seq9)
 (SLIDE/STOP SEGMENT6 (LEFT DOWN)) IS UNATTAINABLE BECAUSE
 (ELASTIC)
SEQ9
```

We can make other assumptions about the motion of the ball.

```
->>(cannot-be-at phob segment30)
(SEGMENT30)
```

Fig. 57. Sequence Graph for COR = 1.0

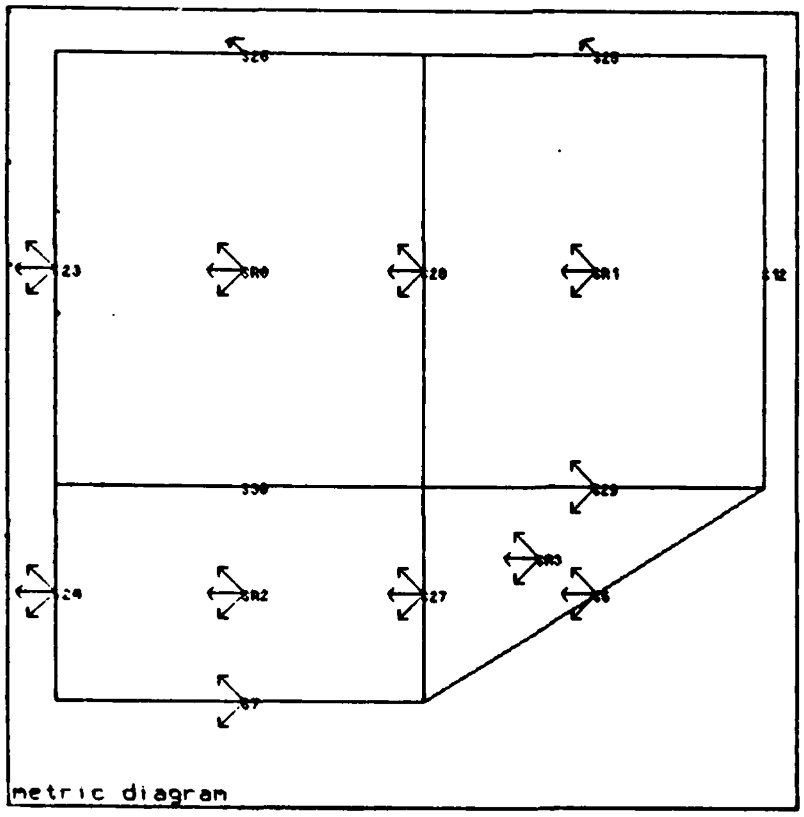


```
 UPDATING ASSUMPTIONS FOR (>> INITIAL-STATE PHOB)
 34 ALIVE OUT OF 38 FOR (>> INITIAL-STATE PHOB)
 CHECKING PATH OF MOTION AGAINST ASSUMPTIONS
->>(motion-summary-for phob)

 FOR G0277
 THE BALL WILL CONTINUE TO MOVE
 IT WILL LEAVE THE DIAGRAM AT (SEGMENT23 SEGMENT24 SEGMENT25 SEGMENT26)
NIL
```

Figure 58 shows the results of making that assumption. At first glance this assumption seems to have had little effect. Let us examine the situation more closely.

```
->>(get sregion0 (>> initial-state phob))
(((FLY (LEFT)) . SEQ34) ((FLY (LEF  UP)) . SEQ33) ((FLY (LEFT DOWN)) . SEQ4))
->>(pseq seq4)

SEQ4
(FLY SREGION0 (LEFT DOWN))
 CAN BE REACHED BY (SEQ34 SEQ2)
```

Fig. 58. Sequence Graph, assuming Segment30 unreachable



```
 NEXT CAN BE (SEQ7 SEQ8)
SEQ4
->>(why-not seq7)
 (PASS SEGMENT30 (LEFT DOWN)) IS UNATTAINABLE BECAUSE
 (CANNOT-REACH SEGMENT30)
SEQ7
->>(why-not seq8)
 (CONTINUE SEGMENT23 (LEFT DOWN)) IS ATTAINABLE
NIL
```

While the Sequence Graph doesn't look very different, there is now a set of mutually exclusive possibilities for the ball's motion. We will choose one by making another assumpt'on..

```
->>(must-be-at phob segment7)
(SEGMENT7)
 UPDATING ASSUMPTIONS FOR (>> INiTIAL-STATE PHOB)
 23 ALIVE OUT OF 38 FOR (>> INITIAL-STATE PHOB)
 CHECKING PATH OF MOTION AGAINST ASSUMPTIONS
```

Figure 59 shows the results.

```
->>(get sregion0 (>> initial-state phob))
(((FLY (LEFT)) . SEQ34) ((FLY (LEFT UP)) . SEQ33) ((FLY (LEFT DOWN)) . SEQ4))
->>(why-not seq34)
 (FLY SREGION0 (LEFT)) IS UNATTAINABLE BECAUSE
 (REQUIRED-AT (SEGMENT7))
SEQ34
->>(get sregion1 'up)
SEGMENT25
->>(get segment25 (>> initial-state phob))
(((CONTINUE (LEFT UP)) . SEQ26))
->>(why-not seq26)
 (CONTINUE SEGMENT25 (LEFT UP)) IS UNATTAINABLE BECAUSE
 (REQUIRED-AT (SEGMENT7))
SEQ26
```

The ambiguity in the description has been greatly reduced.

```
->>(motion-summary-for phob)

 FOR G0277
 THE BALL WILL CONTINUE TO MOVE
 IT WILL LEAVE THE DIAGRAM AT (SEGMENT24)
NIL
```

Let us summarize the assumptions about motion made so far...

```
->>(describe-it (>> assumptions phob))

 (>> ASSUMPTIONS PHOB) = G0354
 G0354 IS THE SET OF MOTION ASSUMPTIONS FOR (>> INITIAL-STATE PHOB)
 THE ROOT OF ITS SEQUENCE GRAPH IS SEQ0
 REQUIRED PLACES ARE (SEGMENT27)
 EXCLUDED PLACES ARE (SEGMENT30)
G0354
->>(get segment23 (>> initial-state phob))
(((CONTINUE (LEFT)) . SEQ37)
 ((CONTINUE (LEFT UP)) . SEQ36)
 ((CONTINUE (LEFT DOWN)) . SEQ8))
```

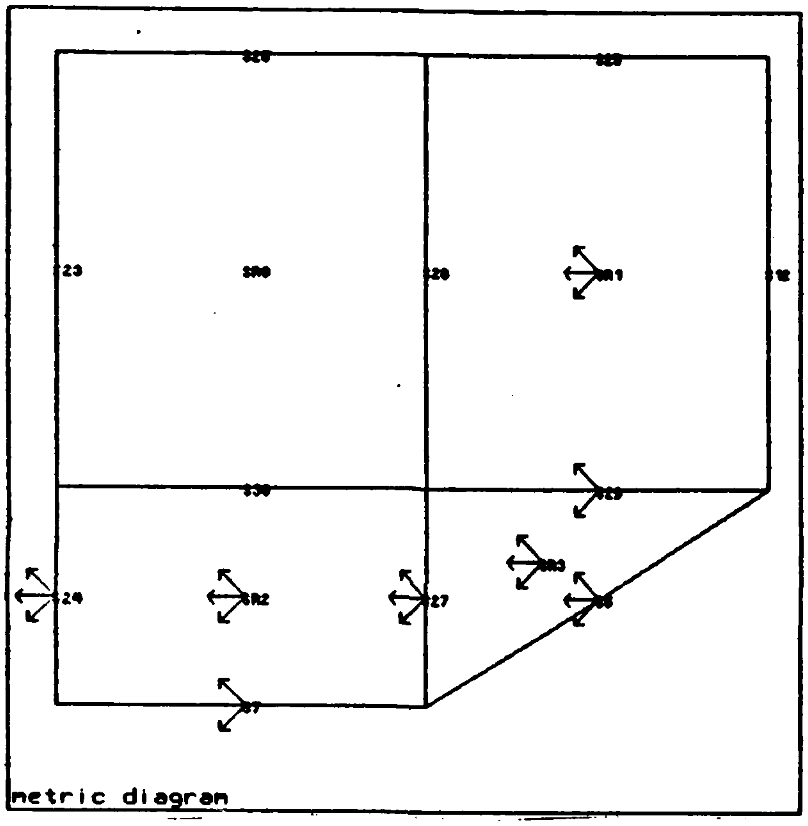


Fig. 59. Assuming Segment30 cannot be reached and Segment7 must be

```
->>(why-not seq37)
 (CONTINUE SEGMENT23 (LEFT)) IS UNATTAINABLE BECAUSE
 (REQUIRED-AT (SEGMENT27))
SEQ37
```

For simulation more exact information is required.

```
->>(set-parameter (>> x initial-state phob) -2.0)
-2.0
->>(set-parameter (>> y initial-state phob) 0.0)
0.0
 CONTRADICTON DISCOVERED CONCERNING (>> POSITION INITIAL-STATE PHOB)
 WHOSE VALUE (SREGION1) DEPENDS ON
1 (>> POSITION INITIAL-STATE PHOB)=(SREGION1) from USER
 THE NEW VALUE (SREGION0) COMPUTED BY (RULE-1 . G0279) DEPENDS ON
2 (>> X INITIAL-STATE PHOB)=-2.0 from USER
3 (>> Y INITIAL-STATE PHOB)=0.0 from USER
 CHOOSE ONE TO RETRACT BY CALLING ANSWER WITH ITS NUMBER
;BKPT CONTRADICTION-HANDLER
```

Of course, this information must be consistent with the qualitative descriptions already given. We will choose to retract the X coordinate.

```
(answer 2)
$p
->>(set-parameter (>> x initial-state phob) 2.0)
2.0
 FLY DUE TO GRAVITY AND UNSUPPORTED PHYSOB
```

Knowing the X and Y coordinates caused a Metric Diagram element representing the position to be created.

```
->>(what-is (>> geometry initial-state phob))
(>> GEOMETRY INITIAL-STATE PHOB) = POINT36
NIL
->>(why (>> geometry initial-state phob))
I used rule (RULE-8 >> INITIAL-STATE PHOB) on the following inputs:
   (>> X INITIAL-STATE PHOB)
   (>> Y INITIAL-STATE PHOB)
(G0285 G0286)
```

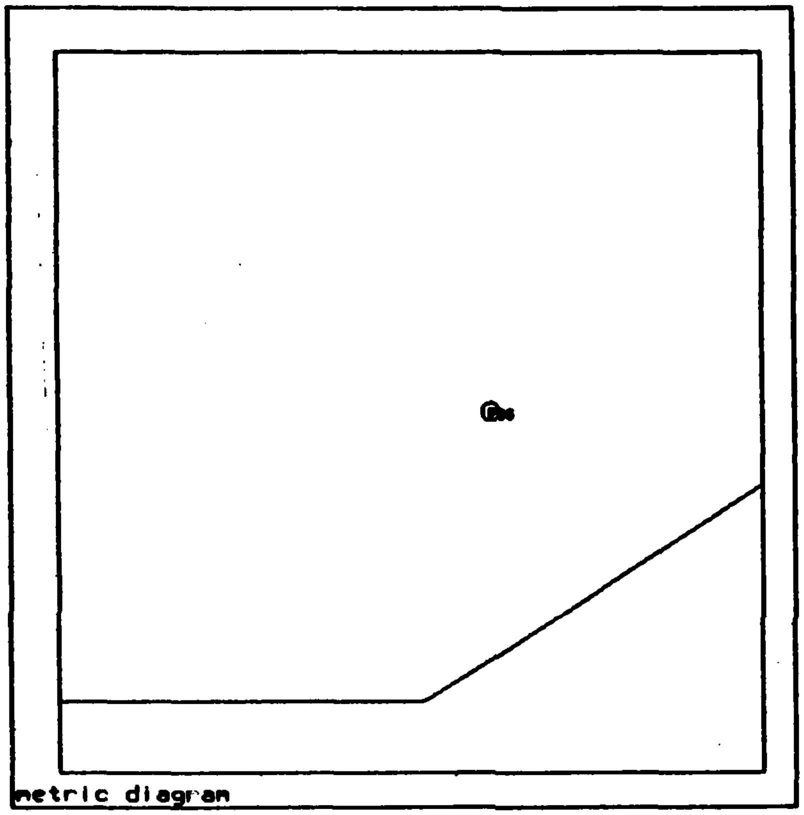
The initial position of the ball is shown in figure 60. An intermediate value of elasticity is more likely to satisfy our assumptions.

```
->>(change-parameter (>> c-o-r initial-state phob) 0.6)
0.6
 CHECKING MOTION OF (>> PHOB)
 UPDATING ASSUMPTIONS FOR (>> INITIAL-STATE PHOB)
 24 ALIVE OUT OF 38 FOR (>> INITIAL-STATE PHOB)
 CHECKING PATH OF MOTION AGAINST ASSUMPTIONS
->>(set-parameter (>> speed initial-state phob) 5.0)
5.0
->>(set-parameter (>> heading initial-state phob) 220.0)
220.0
```

Now we have enough information to simulate the motion of the ball. We should try a single bounce first -

```
->>(simulate phob 3.)
NIL
 THE CURRENT MOTION WILL BE FLY
 FINDING WHERE PARABOLA LANDS GOING (LEFT DOWN) STARTING FROM POINT54
 PARABOLA2 LIMIT IS POINT52
 LOOKING FOR COLLISION GOING ALONG (LEFT DOWN) ON PARABOLA2
        FROM POINT54 TO POINT55
 ON THE PATH ARE (((-1.770668353 -8.0) SEGMENT4))
 COLLISION IS ((-1.770668353 -8.0) SEGMENT4)
 THE CURRENT MOTION WILL BE COLLIDE
 FLY AWAY FROM G0226 AT HEADING 116.0861008
 THE CURRENT MOTION WILL BE FLY
 -4.773155354 , -4.933698222 IS (LEFT UP) LIMIT ALONG PARABOLA3
 LOOKING FOR COLLISION GOING ALONG (LEFT UP) ON PARABOLA3
        FROM POINT63 TO POINT62
 ON THE PATH ARE NIL
 NO COLLISION FOUND
 FLY DUE TO GRAVITY AND UNSUPPORTED PHYSOB
 THE CURRENT MOTION WILL BE FLY
```

Fig. 60. Initial position of the ball

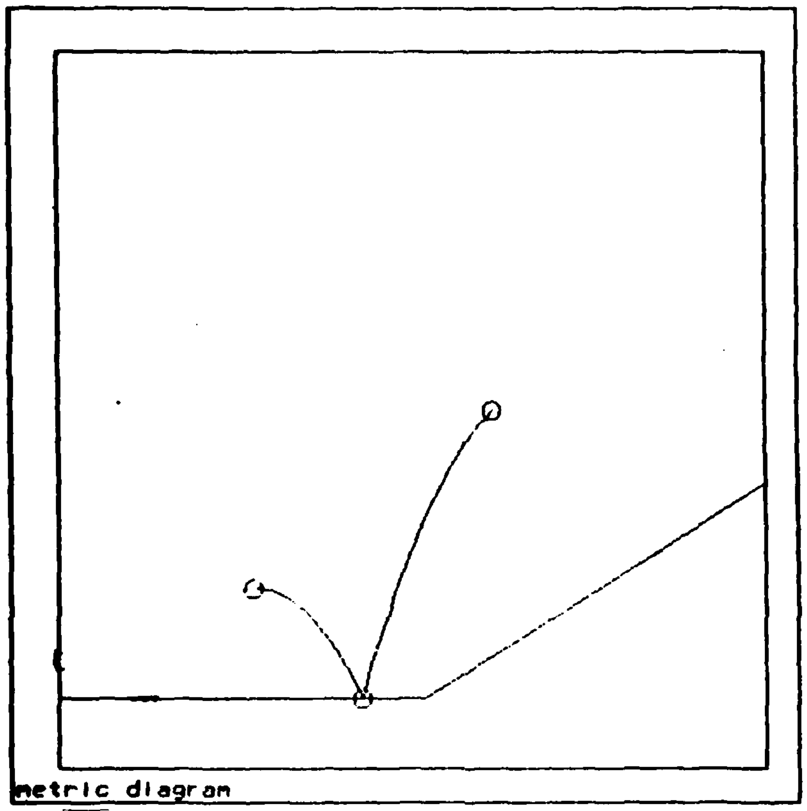


```
 BUT ENOUGH HAS BEEN DONE
 CHECKING MOTION OF (>> PHOB)
 UPDATING ASSUMPTIONS FOR (>> INITIAL-STATE PHOB)
 24 ALIVE OUT OF 38 FOR (>> INITIAL-STATE PHOB)
 CHECKING PATH OF MOTION AGAINST ASSUMPTIONS
```

The Metric Diagram is drawn in Figure 61. The path through the Sequence Graph forms a qualitative description of the actual motion

```
->>(describe-it (>> path phob))

 (>> PATH PHOB) = G0345
 G0345 IS THE STATE PATH FROM (>> INITIAL-STATE PHOB)
 THE STATE TRAJECTORY IS
(FLY (SREGION1) (LEFT DOWN))
(PASS (SEGMENT29) (LEFT DOWN))
(FLY (SREGION3) (LEFT DOWN))
(PASS SEGMENT27 (LEFT DOWN))
(FLY (SREGION2) (LEFT DOWN))
```

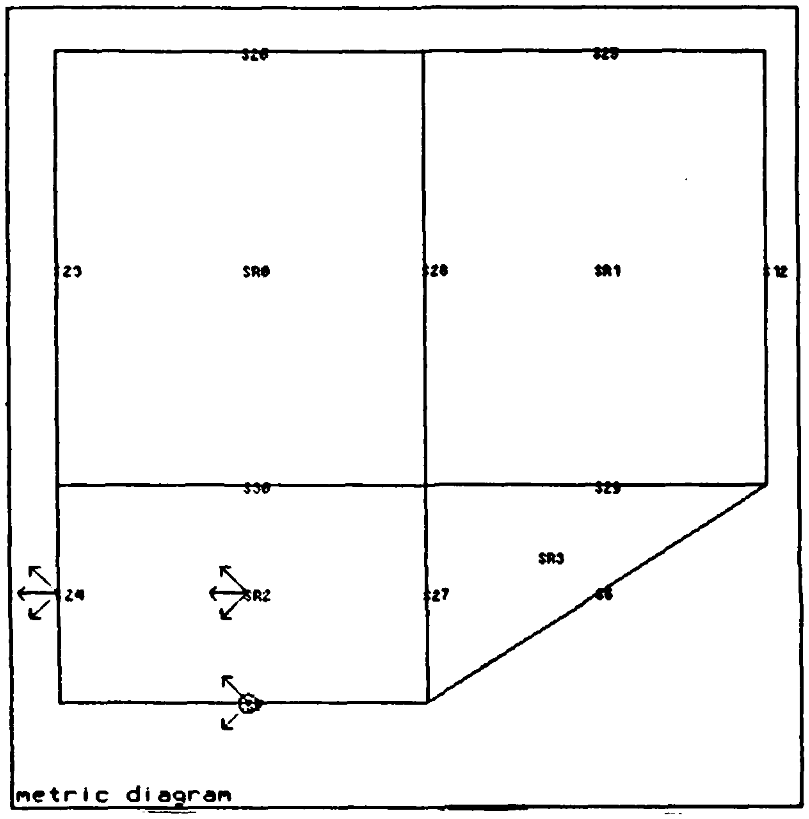
Fig. 61. The first three acts in Phob's motion



```
(COLLIDE (SEGMENT7) (LEFT DOWN))
(FLY (SEGMENT7) (LEFT UP))
(FLY (SREGION2) (LEFT UP))
(FLY (SREGION2) (LEFT))
G0345
->>(motion-summary-for phob)
 NO QUALITATIVE DESCRIPTION FOR (>> PHYSOB2)
 ONE IS BEING CREATED
 COMPUTING SEQUENCE GRAPH FOR (>> PHYSOB2)
 UPDATING ASSUMPTIONS FOR (>> PHYSOB2)
  9 ALIVE OUT OF 18 FOR (>> PHYSOB2)

 FOR G0277
 THE BALL WILL EVENTUALLY STOP
 IT MIGHT LEAVE THE DIAGRAM AT (SEGMENT24)
 OR STOP FLYING AT ONE OF (SEGMENT7)
NIL
```

The Sequence Graph for Physob2 is shown in Figure 62. We can simulate some more, until either the ball stops or it leaves the diagram...

Fig. 62. Sequence Graph after a bounce

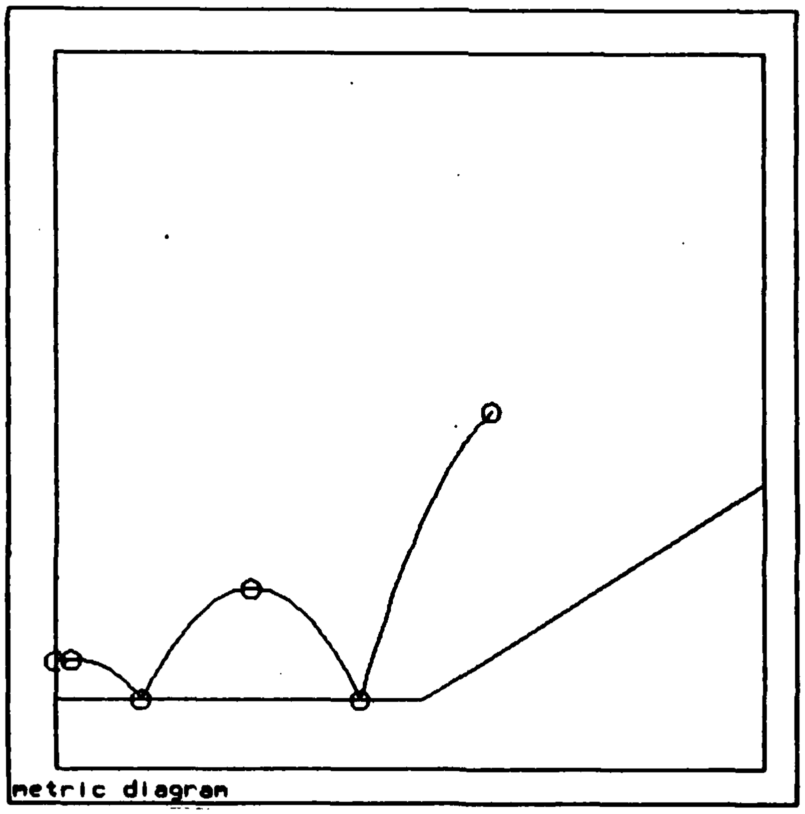


```
->>(simulate phob 5)
NIL
 THE CURRENT MOTION WILL BE FLY
 FINDING WHERE PARABOLA LANDS GOING (LEFT DOWN) STARTING FROM POINT68
 PARABOLA4 LIMIT IS POINT67
 LOOKING FOR COLLISION GOING ALONG (LEFT DOWN) ON PARABOLA4
        FROM POINT68 TO POINT69
 ON THE PATH ARE (((-7.775642358 -8.0) SEGMENT4))
 COLLISION IS ((-7.775642358 -8.0) SEGMENT4)
 THE CURRENT MOTION WILL BE COLLIDE
 FLY AWAY FROM G0226 AT HEADING 129.2141136
 THE CURRENT MOTION WILL BE FLY
 -9.57713456 , -6.896131363 IS (LEFT UP) LIMIT ALONG PARABOLA5
 LOOKING FOR COLLISION GOING ALONG (LEFT UP) ON PARABOLA5
         FROM POINT77 TO POINT76
 ON THE PATH ARE NIL
 NO COLLISION FOUND
 FLY DUE TO GRAVITY AND UNSUPPORTED PHYSOB
 THE CURRENT MOTION WILL BE FLY
 FINDING WHERE PARABOLA LANDS GOING (LEFT DOWN) STARTING FROM POINT82
 PARABOLA6 LIMIT IS POINT80
 LOOKING FOR COLLISION GOING ALONG (LEFT DOWN) ON PARABOLA6
        FROM POINT82 TO POINT83
 ON THE PATH ARE NIL
 NO COLLISION FOUND
 THE CURRENT MOTION WILL BE CONTINUE
 CHECKING MOTION OF (>> PHOB)
 UPDATING ASSUMPTIONS FOR (>> INITIAL-STATE PHOB)
 24 ALIVE OUT OF 38 FOR (>> INITIAL-STATE PHOB)
 CHECKING PATH OF MOTION AGAINST ASSUMPTIONS
```

The trajectory is shown in figure 63. Since this description of motion is complete, we know that our assumptions about it were consistent. Let us examine the full path-

```
->>(describe-it (>> path phob))

 (>> PATH PHOB) = G0345
 G0345 IS THE STATE PATH FROM (>> INITIAL-STATE PHOB)
 THE STATE TRAJECTORY IS
(FLY (SREGION1) (LEFT DOWN))
```

Fig. 63. Full trajectory for Phob



```
(PASS (SEGMENT29) (LEFT DOWN))
(FLY (SREGION3) (LEFT DOWN))
(PASS (SEGMENT27) (LEFT DOWN))
(FLY (SREGION2) (LEFT DOWN))
(COLLIDE (SEGMENT7) (LEFT DOWN))
(FLY (SEGMENT7) (LEFT UP))
(FLY (SREGION2) (LEFT UP))
(FLY (SREGION2) (LEFT))
(FLY (SREGION2) (LEFT DOWN))
(COLLIDE (SEGMENT7) (LEFT DOWN))
(FLY (SEGMENT7) (LEFT UP))
(FLY (SREGION2) (LEFT UP))
(FLY (SREGION2) (LEFT))
(FLY (SREGION2) (LEFT DOWN))
(CONTINUE (SEGMENT24) (LEFT DOWN))
G0345
```

Finally, here is a description of the Action Sequence for the motion.

```
->>(describe-action-sequence phob)

 (>> INITIAL-STATE PHOB) = G0279
 G0279 IS A PHYSOB WHOSE NAME IS UNSPECIFIED
 AT TIME = 0.0
 IT IS AT 2.0 , 0.0 REPRESENTED BY POINT36
 ITS C-O-R IS 0.6
 IT IS CONNECTED TO NIL
 IT IS MOVING (LEFT DOWN) AT A SPEED OF 5.0 AND HEADING 220.0
 THE PREVIOUS-ACT WAS UNKNOWN
 THE NEXT-MOTION IS FLY
 THE NEXT-ACT IS G0328

 (>> ACTION-SEQUENCE PHOB) = G0328
 G0328 IS AN INSTANCE OF ACT
 THE STATE BEFORE IS (>> INITIAL-STATE PHOB)
 THE MOTION IT UNDERGOES IS (>> FLY-SIMULATE0)
 THE STATE AFTER IS (>> PHYSOB0)

 (>> FLY-SIMULATE0) = G0452
 G0452 IS AN INSTANCE OF FLY
 THE STATE BEFORE IS (>> INITIAL-STATE PHOB)
 THE BALL'S STATE AFTERWARD IS (>> PHYSOB0)
 DIRECTION OF FLIGHT IS (LEFT DOWN)
 FLIGHT IS FROM 2.0 , 0.0 TO -1.770668353 , -8.0


 (>> PHYSOB0) = G0406
 G0406 IS A PHYSOB WHOSE NAME IS UNSPECIFIED
 AT TIME = 0.984451612
 IT IS AT -1.770668353 , -8.0 REPRESENTED BY POINT57
 ITS C-O-R IS 0.6
 IT IS CONNECTED TO ((CONTACT SEGMENT4))
 IT IS MOVING (LEFT DOWN) AT A SPEED OF 13.58970198 AND HEADING 253.6295071
 THE PREVIOUS-ACT WAS G0328
 THE NEXT-MOTION IS COLLIDE
 THE NEXT-ACT IS G0395

 (>> ACT0) = G0395
 G0395 IS AN INSTANCE OF ACT
 THE STATE BEFORE IS (>> PHYSOB0)
 THE MOTION IT UNDERGOES IS (>> COLLIDE0)
 THE STATE AFTER IS (>> PHYSOB1)

 (>> COLLIDE0) = G0950
```

```
G0950 IS AN INSTANCE OF COLLIDE
THE STATE BEFORE IS (>> PHYSOB0)
THE BALL'S STATE AFTERWARDS IS (>> PHYSOB1)
IT COLLIDED WITH (>> FLOOR) AT POINT57
THE SURFACE NORMAL AT THIS POINT IS (0.0 1.0)
THE OBJECT STRUCK WITH SPEED 13.58970198 AND HEADING 253.6295071
IT RECOILED WITH SPEED 8.71056743 AND HEADING 116.0861008

(>> PHYSOB1) = G0904
G0904 IS A PHYSOB WHOSE NAME IS UNSPECIFIED
AT TIME = 0.984451612
IT IS AT -1.770668353 , -8.0 REPRESENTED BY POINT58
ITS C-O-R IS 0.6
IT IS CONNECTED TO ((CONTACT SEGMENT4))
IT IS MOVING (LEFT UP) AT A SPEED OF 8.71056743 AND HEADING 116.0861008
THE PREVIOUS-ACT WAS G0395
THE NEXT-MOTION IS FLY
THE NEXT-ACT IS G0893

(>> ACT1) = G0893
G0893 IS AN INSTANCE OF ACT
THE STATE BEFORE IS (>> PHYSOB1)
THE MOTION IT UNDERGOES IS (>> FLY-SIMULATE1)
THE STATE AFTER IS (>> PHYSOB2)

(>> FLY-SIMULATE1) = G1116
G1116 IS AN INSTANCE OF FLY
THE STATE BEFORE IS (>> PHYSOB1)
THE BALL'S STATE AFTERWARD IS (>> PHYSOB2)
DIRECTION OF FLIGHT IS (LEFT UP)
FLIGHT IS FROM -1.770668353 , -8.0 TO -4.773155354 , -4.933698222


(>> PHYSOB2) = G1070
G1070 IS A PHYSOB WHOSE NAME IS UNSPECIFIED
AT TIME = 1.768345312
IT IS AT -4.773155354 , -4.933698222 REPRESENTED BY POINT64
ITS C-O-R IS 0.6
IT IS CONNECTED TO NIL
IT IS MOVING (LEFT) AT A SPEED OF 3.830222134 AND HEADING 179.9999999
THE PREVIOUS-ACT WAS G0893
THE NEXT-MOTION IS FLY
THE NEXT-ACT IS G1059

(>> ACT2) = G1059
G1059 IS AN INSTANCE OF ACT
THE STATE BEFORE IS (>> PHYSOB2)
THE MOTION IT UNDERGOES IS (>> FLY-SIMULATE2)
THE STATE AFTER IS (>> PHYSOB3)

(>> FLY-SIMULATE2) = G1338
G1338 IS AN INSTANCE OF FLY
THE STATE BEFORE IS (>> PHYSOB2)
THE BALL'S STATE AFTERWARD IS (>> PHYSOB3)
DIRECTION OF FLIGHT IS (LEFT DOWN)
FLIGHT IS FROM -4.773155354 , -4.933698222 TO -7.775642358 , -8.0


(>> PHYSOB3) = G1292
G1292 IS A PHYSOB WHOSE NAME IS UNSPECIFIED
AT TIME = 2.552239014
IT IS AT -7.775642358 , -8.0 REPRESENTED BY POINT71
ITS C-O-R IS 0.6
IT IS CONNECTED TO ((CONTACT SEGMENT4))
IT IS MOVING (LEFT DOWN) AT A SPEED OF 8.71056743 AND HEADING 243.9138994
```

```
THE PREVIOUS-ACT WAS G1059
THE NEXT-MOTION IS COLLIDE
THE NEXT-ACT IS G1281

(>> ACT3) = G1281
G1281 IS AN INSTANCE OF ACT
THE STATE BEFORE IS (>> PHYSOB3)
THE MOTION IT UNDERGOES IS (>> COLLIDE1)
THE STATE AFTER IS (>> PHYSOB4)

(>> COLLIDE1) = G1505
G1505 IS AN INSTANCE OF COLLIDE
THE STATE BEFORE IS (>> PHYSOB3)
THE BALL'S STATE AFTERWARDS IS (>> PHYSOB4)
IT COLLIDED WITH (>> FLOOR) AT POINT71
THE SURFACE NORMAL AT THIS POINT IS (0.0 1.0)
THE OBJECT STRUCK WITH SPEED 8.71056743 AND HEADING 243.9138994
IT RECOILED WITH SPEED 6.058367737 AND HEADING 129.2141136

(>> PHYSOB4) = G1459
G1459 IS A PHYSOB WHOSE NAME IS UNSPECIFIED
AT TIME = 2.552239014
IT IS AT -7.775642358 , -8.0 REPRESENTED BY POINT72
ITS C-O-R IS 0.6
IT IS CONNECTED TO ((CONTACT SEGMENT4))
IT IS MOVING (LEFT UP) AT A SPEED OF 6.058367737 AND HEADING 129.2141136
THE PREVIOUS-ACT WAS G1281
THE NEXT-MOTION IS FLY
THE NEXT-ACT IS G1448

(>> ACT4) = G1448
G1448 IS AN INSTANCE OF ACT
THE STATE BEFORE IS (>> PHYSOB4)
THE MOTION IT UNDERGOES IS (>> FLY-SIMULATE3)
THE STATE AFTER IS (>> PHYSOB5)

(>> FLY-SIMULATE3) = G1671
G1671 IS AN INSTANCE OF FLY
THE STATE BEFORE IS (>> PHYSOB4)
THE BALL'S STATE AFTERWARD IS (>> PHYSOB5)
DIRECTION OF FLIGHT IS (LEFT UP)
FLIGHT IS FROM -7.775642358 , -8.0 TO -9.57713456 , -6.896131363


(>> PHYSOB5) = G1625
G1625 IS A PHYSOB WHOSE NAME IS UNSPECIFIED
AT TIME = 3.022575233
IT IS AT -9.57713456 , -6.896131363 REPRESENTED BY POINT78
ITS C-O-R IS 0.6
IT IS CONNECTED TO NIL
IT IS MOVING (LEFT) AT A SPEED OF 3.830222134 AND HEADING 179.9999999
THE PREVIOUS-ACT WAS G1448
THE NEXT-MOTION IS FLY
THE NEXT-ACT IS G1614

(>> ACT5) = G1614
G1614 IS AN INSTANCE OF ACT
THE STATE BEFORE IS (>> PHYSOB5)
THE MOTION IT UNDERGOES IS (>> FLY-SIMULATE4)
THE STATE AFTER IS (>> PHYSOB6)

(>> FLY-SIMULATE4) = G1837
G1837 IS AN INSTANCE OF FLY
THE STATE BEFORE IS (>> PHYSOB5)
THE BALL'S STATE AFTERWARD IS (>> PHYSOB6)
```

```
DIRECTION OF FLIGHT IS (LEFT DOWN)
FLIGHT IS FROM -9.57713456 , -6.896131363 TO -10.0 , -6.956952848


(>> PHYSOB6) = G1791
G1791 IS A PHYSOB WHOSE NAME IS UNSPECIFIED
AT TIME = 3.132977562
IT IS AT -10.0 , -6.956952848 REPRESENTED BY POINT84
ITS C-O-R IS 0.6
IT IS CONNECTED TO ((CONTACT SEGMENTO))
IT IS MOVING (LEFT DOWN) AT A SPEED OF 3.985548694 AND HEADING 196.0485959
THE PREVIOUS-ACT WAS G1614
THE NEXT-MOTION IS CONTINUE
THE NEXT-ACT IS G1780

(>> ACT6) = G1780
G1780 IS AN INSTANCE OF ACT
THE STATE BEFORE IS (>> PHYSOB6)
THE MOTION IT UNDERGOES IS (>> CONTINUEO)
THE STATE AFTER IS (>> PHYSOB7)

(>> CONTINUEO) = G2003
G2003 IS AN INSTANCE OF CONTINUE
THE OBJECT IS G1791
THE NUMBER OF ACTIONS LEFT IS UNKNOWN

(>> PHYSOB7) = G1957
G1957 IS A PHYSOB WHOSE NAME IS UNSPECIFIED
AT TIME = 3.132977562
IT IS AT -10.0 , -6.956952848 REPRESENTED BY POINT86
ITS C-O-R IS 0.6
IT IS CONNECTED TO ((CONTACT SEGMENTO))
THE PREVIOUS-ACT WAS G1780
THE NEXT-MOTION IS UNKNOWN
THE NEXT-ACT IS UNKNOWN

(>> ACT7) = G1946
G1946 IS AN INSTANCE OF ACT
THE STATE BEFORE IS (>> PHYSOB7)
NIL
->>(dribble-end)
```

9. Appendix 2 - A CONLAN Overview

Physical systems are often described in terms of the constraints imposed between their parts. Computations can also be organized around constraints in a declarative fashion, instead of imposing a particular flow of information and control through the constraints to yield a normal imperative program. CONLAN [Steele & Sussman] is a language organized in this manner. A variant of this language was used to encode the representations of objects in FROB. For readers who wish to understand the representations in more detail, a brief overview of the language is presented here, including changes made to increase efficiency, support the use of a diagram, and to allow constraint networks to add parts to themselves.

9.1 Basics

The basic descriptive element in CONLAN is the constraint body. A body has parts, which are either other constraint bodies or cells. Cells hold values that describe properties of the object represented by that body. A part is named by a path, such as

```
(>> x-component velocity physob3)
```

which evaluates to the cell corresponding to "the x-component of the velocity of physob3". A cell that is shared may have a compound name, such as

```
(>> (speed physob3) (magnitude velocity physob3))
```

This cell can be accessed either by the name

```
(>> speed physob3)
```

or

```
(>> magnitude velocity physob3)
```

Computation occurs by rules attached to constraint bodies. A rule computes a value for a cell given the values of some other cells. If the rule cannot return a value it can dismiss itself, and if it detects an inconsistency it can signal a contradiction. The source of a value (either a rule or set by the user) is always marked on a cell. A simple matching process compares values generated for a cell by alternate sources, and signals a contradiction if they do not match. When a contradiction is found, the assumptions involved are offered up for inspection and possible correction.

The constraint body for VECTOR is illustrated below. The definition format is

```
(defbody <name> <partslist> <other things>)
        where <name>=name of constraint,
        <partslist>=((<partname> <type>)....)
        <other things>=specifications of rules (formulae)
                interconnections (wiring)
                bookkeeping (if-removed)
```

The specification format of the rules are

```
(formulae
       ((<cell to be set>
        <cells that are used to compute the value>
        <body of code to execute>)
       .
       .
       .)
```

Wiring and if-removed will be discused later. Here is the constraint representation for a vector-

```
(defbody vector ((magnitude cell)
             (direction cell)
             (x-component cell)
             (y-component cell))
      (formulae (x-component (magnitude direction)
                           (times magnitude (cosine direction)))
               (y-component (magnitude direction)
                           (times magnitude (sine direction)))
               (magnitude (x-component y-component)
                          (square-root (plus (square x-component)
                                             (square y-component))))
               (direction (x-component y-component)
                          (cond ((and (nearly-zero? x-component)
                                      (nearly-zero? y-component))
                                 *dismiss*);can't tell
                                (t (arctan y-component x-component))))))
```

9.2 Running Constraints

The computations specified with a constraint are performed by an interpreter embedded in LISP. The basic cycle of this interpreter can be described as READ-EVAL-PRINT-RUN, where READ, EVAL, and PRINT are the same actions taken by the normal LISP top level. RUN refers to the process of servicing the queues of the constraint interpreter. The program waits for more input once the queues are empty.

Below is an interaction with the interpreter while filling in the PHYSOB constraint (the state of a ball at an instance of time). Commentary is on lines marked by ";". User input is in lower case, the program's responses are in uppercase.

```
->>(create phob 'physob)
;create a PHYSOB and call it PHOB
;creation occurs by instantiation of prototypes, to maximize
;shared structure
 (>>) IS SETTING (>> REFERENCE-FRAME PHOB) TO
                 (NEAR-EARTH SIDE-VIEW).
 (>>) IS SETTING (>> LOWER-BOUND COR-CHECK PHOB) TO 0.0
 (>>) IS SETTING (>> UPPER-BOUND COR-CHECK PHOB) TO 1.0
;some cells have values that are constants
G2147
;print result of create, start running rules
 (RULE-16 >> PHOB) IS SETTING (>> FORCES PHOB) TO (GRAVITY)
;if near the earth, gravity works
->>(set-parameter (>> x phob) 5.0)
 (>>) IS SETTING (>> X PHOB) TO 5.0
5.0
```

```
->>(set-parameter (>> y phob) 5.0)
 (>>) IS SETTING (>> Y PHOB) TO 5.0
5.0
 (RULE-8 >> PHOB) IS SETTING (>> GEOMETRY PHOB) TO POINT70
 (RULE-12 >> PHOB) IS SETTING (>> CONNECTIONS PHOB) TO NIL
;knowing x and y determines the point, and the point is checked
;against the diagram to see what it touches
 (RULE-11 >> PHOB) IS SETTING (>> Y PHOB) TO 5.0
 (RULE-10 >> PHOB) IS SETTING (>> X PHOB) TO 5.0
;these rules compute x and y given a point.  Since values are
;already known for these, they are just checked to see if they
;are consistent
 (RULE-7 >> PHOB) IS SETTING (>> GEOMETRY-TYPE PHOB) TO POINT
 (RULE-1 >> PHOB) IS SETTING (>> POSITION PHOB) TO (SREGION3)
;mapping to qualitative space representation is done locally
->>(set-parameter (>> speed phob) 5.0)
 (>>) IS SETTING
  (>> (SPEED PHOB) (MAGNITUDE VELOCITY PHOB)) TO 5.0
5.0
 (2<-1    1<=>2) IS SETTING
  (>>     TUDE INPUT-VECTOR PROJECTOR PHOB) TO 5.0
 (RUL       PHOB) IS SETTING (>> MAX-HEIGHT PHOB) TO 6.25250501
 (1<-2    1<=>2) IS SETTING
  (>> (SPEED PHOB) (MAGNITUDE VELOCITY PHOB)) TO 5.0
;MAX-HEIGHT captures the energy of the ball
->>(set-parameter (>> heading phob) '(200.0 degrees))
 (>>) IS SETTING
  (>> (HEADING PHOB) (DIRECTION VELOCITY PHOB)) TO 200.0
(200.0 DEGREES)
 (2<-1 >> 1<=>2) IS SETTING
  (>> DIRECTION INPUT-VECTOR PROJECTOR PHOB) TO 200.0
 (RULE-17 >> PHOB) IS SETTING (>> DIRECTION PHOB) TO (LEFT DOWN)
;describe the heading in qualitative terms
 (RULE-2 >> VELOCITY PHOB)
  IS SETTING (>> Y-COMPONENT VELOCITY PHOB)
         TO -1.710100803
 (RULE-1 >> VELOCITY PHOB)
  IS SETTING (>> X-COMPONENT VELOCITY PHOB)
         TO -4.698463086
 (1<-2 >> 1<=>2) IS SETTING
  (>> (HEADING PHOB) (DIRECTION VELOCITY PHOB)) TO 200.0
 (RULE-2 >> INPUT-VECTOR PROJECTOR PHOB) IS SETTING
  (>> (VY PROJECTOR PHOB) (Y-COMPONENT INPUT-VECTOR PROJECTOR PHOB))
         TO -1.710100803
 (RULE-1 >> INPUT-VECTOR PROJECTOR PHOB) IS SETTING
  (>> (VX PROJECTOR PHOB) (X-COMPONENT INPUT-VECTOR PROJECTOR PHOB))
         TO -4.698463086
 FLY DUE TO GRAVITY AND UNSUPPORTED PHYSOB
;this cryptic message is printed by RULE-19
 (RULE-19 >> PHOB) IS SETTING (>> NEXT-MOTION PHOB) TO FLY
;not supported implies flying
 (RULE-2 >> PHOB) IS SETTING
  (>> STATE PHOB) TO (FLY (SREGION3) (LEFT DOWN))
;compute the qualitative description of the state
 (2<-1 >> 1<=>2) IS SETTING
  (>> (VY PROJECTOR PHOB) (Y-COMPONENT INPUT-VECTOR PROJECTOR PHOB))
         TO -1.710100803
 (RULE-4 >> VELOCITY PHOB)
   IS SETTING (>> (HEADING PHOB) (DIRECTION VELOCITY PHOB))
         TO 200.0000027
 (RULE-3 >> VELOCITY PHOB)
   IS SETTING (>> (SPEED PHOB) (MAGNITUDE VELOCITY PHOB))
        TO 5.000000015
 (2<-1 >> 1<=>2) IS SETTING
  (>> (VX PROJECTOR PHOB) (X-COMPONENT INPUT-VECTOR PROJECTOR PHOB))
```

```
        TO -4.698463086
 (RULE-4 >> VELOCITY PHOB) IS SETTING
  (>> (HEADING PHOB) (DIRECTION VELOCITY PHOB))
        TO 200.0000027
 (RULE-3 >> VELOCITY PHOB) IS SETTING
  (>> (SPEED PHOB) (MAGNITUDE VELOCITY PHOB))
        TO 5.000000015
 (1<-2 >> 1<=>2) IS SETTING (>> Y-COMPONENT VELOCITY PHOB) TO -1.710100?03
 (RULE-4 >> INPUT-VECTOR PROJECTOR PHOB)
  IS SETTING (>> DIRECTION INPUT-VECTOR PROJECTOR PHOB)
        TO 200.0000027
 (RULE-3 >> INPUT-VECTOR PROJECTOR PHOB)
  IS SETTING (>> MAGNITUDE INPUT-VECTOR PROJECTOR PHOB)
        TO 5.000000015
 (1<-2 >> 1<=>2) IS SETTING (>> X-COMPONENT VELOCITY PHOB) TO -4.698463?86
 (RULE-4 >> INPUT-VECTOR PROJECTOR PHOB)
  IS SETTING (>> DIRECTION INPUT-VECTOR PROJECTOR PHOB)
        TO 200.0000027
 (RULE-3 >> INPUT-VECTOR PROJECTOR PHOB)
  IS SETTING (>> MAGNITUDE INPUT-VECTOR PROJECTOR PHOB)
        TO 5.000000015
;lots of checking done
 (RULE-2 >> PHOB) ALREADY RUN
 (RULE-5 >> PHOB) IS SETTING (>> NEXT-MOTION PHOB) TO FLY
 (RULE-4 >> PHOB) IS SETTING (>> DIRECTION PHOB) TO (LEFT DOWN)
 (RULE-3 >> PHOB) IS SETTING (>> POSITION PHOB) TO (SREGION3)
->>(set-parameter (>> c-o-r phob) 0.5)
 (>>) IS SETTING
  (>> (CHECKED-VALUE COR-CHECK PHOB) (C-O-R PHOB)) TO 0.5
0.5
->>(describe-it phob)
 G2147 IS A PHYSOB WHOSE NAME IS UNSPECIFIED
 AT TIME = UNKNOWN
 IT IS AT 5.0 , 5.0 REPRESENTED BY POINT70
 THE FRAME OF REFERENCE IS (NEAR-EARTH SIDE-VIEW)
 ITS C-O-R IS 0.5
 IT IS CONNECTED TO NIL
 IT IS ACTED ON BY (GRAVITY)
 IT IS MOVING (LEFT DOWN) AT A SPEED OF 5.0 AND HEADING 200.0
 THE PREVIOUS-ACT WAS UNKNOWN
 THE NEXT-MOTION IS FLY
 THE NET-ACT IS UNKNOWN
(>> PHOB)
->>(dribble-end)
;finis
```

The source of a cell's value is noted along with the value itself. This can be the name of a rule or some mark denoting the value as an assumption or constant. If two sources compute different values for the same cell, a contradiction is signalled, and the assumptions underlying the values are traced down and presented to the user for possible retraction.

9.3 Modifications

A single constraint is not very useful. We want to build descriptions of situations by linking up constraints into networks. In the original CONLAN this was accomplished by the same equality mechanism that was used to specify that two parts within a constraint body were really the same thing. That is,

```
(== speed (>> magnitude velocity))
```

placed two rules between (>> speed phob) and (>> magnitude velocity phob), such that when one was known the other would be set to the same value. In the trace above, these rules were called (>> 1<-2 1<=>2) and (>> 2<-1 1<=>), where 1<=>2 is the prototype equality constraint. Constraints could then be formed into a network by equating parts belonging to two different constraints.

While a very useful idea, the simple notion of equality is not really adequate. First, equality within a constraint really means that two things are the same, not that their cells always have the same values. If parts of a network can be retracted, this is not true of equality between parts of different constraint bodies. Secondly, == requires unnecessary duplication of structure. For example, each ACT constraint would need two copies of the PHYSOB constraint, whose sole purpose is to hold quantities that might be desired for computation within the constraint. Lastly, it would be convenient to include within the constraint itself a way of specifying how it can be hooked up to other constraints. New mechanisms have been added to CONLAN to ameliorate these problems.

Equality between parts of a constraint are expressed using R== instead of ==. During the instantiation process R== is interpreted as "create one thing, and call it by these names". The efficiency gained in a complicated constraint can be considerable.

A special cell type is defined to facilitate specification of linkages between constraints in the constraints themselves. A CONLAN-CELL has only other constraints as its value. These cells act as indirects, in that a reference path including them goes down the constraint that is its value, rather than stopping at the cell. For example,

```
(>>i path motion action-sequence phob)
```

returns the path cell for whatever motion (an instance of FLY, for example) is the value of the MOTION cell for PHOB's Action Sequence.

The connections between constraints are specified by wiring rules. A wiring rule fires when the CONLAN-CELLs it depends upon are known. The body of a wiring rule consists of calls to == and SET-PARAMETER that connect the appropriate parts of the constraints togather. When one of the CONLAN-CELLs is forgotten, a special function is run to undo the effects of the wiring rule.

Using the Metric Diagram with CONLAN was facilitated by the definition of a GEOMETRY-CELL. The value of a GEOMETRY-CELL is always a Metric Diagram element, and when the cell is forgotten this element is destroyed. Placing a value in a geometry cell causes the value to be marked with the name of the rule that created it. This simplifies debugging. If the value of a geometry cell is known, the rules that can set it are never run. This bookkeeping measure assures that the diagram is not cluttered with extra elements. Matching to detect inconsistencies is performed on cells that hold the parameters that specify the diagram element, not on the diagram elements themselves.

The idea of a function to be run when a cell is forgotten is useful outside of just CONLAN and GEOMETRY cells. For example, the Sequence Graph computation is based on the qualitative state for the ball. When this state is no longer known the Sequence Graph must be destroyed. Arbitrary programs that will be run when a cell's value is forgotten can be specified by the IF-REMOVED construct. This code is run in addition to any actions on forgetting inherited because of the cell's type.

9.4 Limitations

There are times when the local nature of references in CONLAN is too confining[1] For example, it would be desirable for the Action Sequence constraint in FROB to have a cell that contains all the Metric Diagram elements that comprise a ball's trajectory, or a cell in a constraint that was describing the transistor in a VLSI chip that holds its connections in order to compute the amount of current the transistor needs to supply. This kind of cell cannot be explicitly specified in CONLAN because all rules take a fixed set of arguments. A rule also cannot explicitly use a global parameter in its computation. One case where this would be useful is in the interpretation of the diagram. The reference-frame cell of each PHYSOB contains the information that the diagram is to be considered as the side view of a situation near the earth. A better place for this information would be the SCENE constraint, but then some other program would have to explicitly connect each PHYSOB to the SCENE. The ability to specify a general reference path, containing pattern matching variables, in the specifications of the cell set or cells used by a rule would deal with this problem nicely.

The dependency system in CONLAN is too simple. The reference-frame, to continue the example above, is specified in FROB as a constant but in a more general system should be a default. Aside from more general truth maintaince features, signalling a contradiction during a mismatch can cite too many premises as being reponsible for the problem. The system might instead run all rules when a value is added and intersect the assumptions from all contradictions that occur to get a minimal set.

The use of a single comparison function for all cells is also very confining - some types of cells may contain numeric values of more accuracy than others, while others may be a bound for which different rules should compute very different values. An example is the computation of the maximum height a ball can reach. If the ball is going horizontally at some point in time and all surfaces below it are oriented vertically or horizontally, then the current height is the maximum since the horizontal velocity cannot be transformed into vertical velocity. This rule requires knowing the exact position of the ball, but a bound on the maximum height can be computed with only the current height by assuming that all the velocity can be dissipated against gravity. Putting both of these rules into the current system could lead to a contradiction.

1. The explicit connections between cells and the rules that use them means that CONLAN does not require a pattern directed data base While pattern directed invocation is very powerful, it is also very slow According to Sussman [personal communication], the majority of the run time in a CONNIVER program was spent in data base manipulations Splitting the data base reduces the time to access items, with the purely local referencing in CONLAN being the ultimate factorization

Ignoring the role of pattern directed invocation led Fahlman [Fahlman 79] to conclude that expanding even the simplest circuit into constraint bodies would be unacceptably slow Experience with FROB shows this is not the case, for the constraint bodies used for the Bouncing Ball world are much larger than those for circuits This in no way detracts from the importance of the issues addressed by NETL, but only implies that the problems of scale are not quite as pressing as they first seemed.

DATE
ILMED
-8