Order Number 8803034

Qualitative kinematics in mechanisms

Faltings, Boi Volkert, Ph.D.

University of Illinois at Urbana-Champaign, 1987

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

PLEASE NOTE:

In all cases this material has been filmed in the best possible way from the available copy. Problems encountered with this document have been identified here with a check mark __✓__.

1. Glossy photographs or pages _____
2. Colored illustrations, paper or print _____
3. Photographs with dark background _____
4. Illustrations are poor copy _____
5. Pages with black marks, not original copy _____
6. Print shows through as there is text on both sides of page _____
7. Indistinct, broken or small print on several pages ✓
8. Print exceeds margin requirements _____
9. Tightly bound copy with print lost in spine _____
10. Computer printout pages with indistinct print _____
11. Page(s) _____ lacking when material received, and not available from school or author.
12. Page(s) _____ seem to be missing in numbering only as text follows.
13. Two pages numbered _____. Text follows.
14. Curling and wrinkled pages _____
15. Dissertation contains pages with print at a slant, filmed as received ✓
16. Other _____

_____

_____

UMI

QUALITATIVE KINEMATICS IN MECHANISMS

BY

BOI VOLKERT FALTINGS

Diploma, Swiss Federal Institute of Technology, 1983

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1987

Urbana Illinois

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

THE GRADUATE COLLEGE

JULY 1987

WE HEREBY RECOMMEND THAT THE THESIS BY

BOI VOLKERT FALTINGS

ENTITLED QUALITATIVE KINEMATICS IN MECHANISMS

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF DOCTOR OF PHILOSOPHY

Director of Thesis Research

Head of Department

Committee on Final Examination†

Chairperson

† Required for doctor's degree but not for master's.

O-517

Qualitative Kinematics in Mechanisms

Boi Volkert Faltings, Ph. D.
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign, 1987
Kenneth D. Forbus, Advisor

Qualitative Kinematics is the qualitative analysis of the possible geometric interactions of physical objects. This thesis investigates the problem of reasoning about the kinematic interactions between parts of a 2-dimensional mechanism as a first step towards a general theory of qualitative kinematics. We introduce the concept of *place vocabularies* as a generative symbolic description of the possible motion of the parts of a mechanism. Place vocabularies are particularly useful to describe higher kinematic pairs such as ratchets and escapements.

We examine the requirements for the representation and introduce a definition of place vocabularies that satisfies them. We show how this representation can be computed from metric data about the shapes of the parts of a mechanism and used as a basis for qualitative envisionments of its behavior. In particular, we give implemented algorithms for computing place vocabularies for 2-dimensional higher kinematic pairs. The objects involved in the pair can have arbitrary shapes, but their boundaries must consist of straight lines and arcs, and each object must have one degree of freedom only. Complete mechanisms are analyzed as compositions of kinematic pairs.

The algorithms for computing place vocabularies fall within the qualitative reasoning paradigm. They are based on splitting the computation into 2 parts: a purely symbolic reasoning part and a *metric diagram.* The metric diagram is an abstract device that gives access to information about quantities by evaluating predicates on them. We propose that this division is a plausible model of

human reasoning. When quantitative information is incomplete, the metric diagram defines a set of *landmark values* for unknown metric parameters. The resulting place vocabulary changes only at these landmark values, and an exhaustive list of all place vocabularies that can be achieved by varying the parameters is found by computation at representative points in each interval. We show how the mechanism design problem of picking suitable values for parameters can be solved by searching the list, and give an implemented example. The exhaustive list forms an ambiguous but complete prediction of the possible behavior. This proves that contrary to common belief, it is in principle possible to reason about kinematics with limited or no knowledge of the metric dimensions of the objects.

Contents

Chapter 1

INTRODUCTION

The question of how to describe the physical world in a discrete, symbolic manner is fundamental for Artificial Intelligence. Intelligent beings must solve problems in the physical world. Artificial Intelligence has developed a large body of computational theory at the *symbolic* level, but the important question of how precisely this symbolic level relates to the physical world has not yet been answered satisfactorily. This problem is important, as the physical world is *continuous*, while the symbols used for reasoning are *discrete*.

This thesis is about a computational theory of how symbolic reasoning can be based on physical data. We investigate the problem of *qualitative kinematics* ([FNF87,FALT86,FALT87]), meaning reasoning in a symbolic manner about the kinematic interactions of rigid objects. We restrict the investigation to the domain of rigid-body mechanisms. Our goal is to investigate methods for constructing a description of the qualitative behavior of a mechanism which is useful for understanding its function within the framework of naive physics ([FOR84,FOR81,DKL79,HAY79,DKL75]). Our solution for the special case of mechanisms points out a possibility that could be generalized to other cases. At the same time, the limitations we find also apply to the general case.

Besides its fundamental value for AI, this work also has practical applications. Reasoning about the function of mechanical devices allows us to automatically verify and design mechanisms. Kinematic verification becomes possible because we can automatically identify the function of a mechanism and compare it to the desired one. We will show later how the theory we develop allows us to automatically

solve simple mechanical design problems.

In current technology, there are two commonly used ways to manipulate information about the continuous physical world: (i) numerical solutions and (ii) reasoning with a finite symbolic representation. Numerical solutions exploit the mapping of all parameters of the system into the same domain (the number system) to achieve greater efficiency, but this fact also limits their power. The result of the computation is given as a set of numbers, with no further information about *why* the particular result was obtained. But this latter information is often very important for problem solving: it tells us where to look for the solution. With numerical techniques, all we can do is simulate the effect of changing the system in various ways and see if the result happens to be what we desire. The numerical techniques also have another drawback. Because they are based on evaluating the description of the system at *points* in parameter space, the technique is necessarily incomplete: the point where undesired behavior occurs may not be among those evaluated. There has to be an explicit choice of parameter values for the computation, which makes it very difficult to reason with incomplete or uncertain information. Much work has gone into this problem, the most successful results being Fuzzy Logic ([ZADEH79]) and the Shafer-Dempster theory ([SHA76]), but the solutions are still unsatisfactory. An application of numerical methods to geometric problems has been studied in the ACRONYM system ([BRO81]).

In certain restricted domains, methods have been developed to reason symbolically about the interactions of objects. For example, a CAD system might have a finite set of symbols to describe the standardized types and sizes of nuts and bolts and rules that state how these can fit together. An example of this type of approach in AI applications is ([STA83]). In the domain of mechanisms, Andrew Gelsey ([GELSEY87]) has recently developed a system that determines the function of a mechanism whose parts are taken from some predetermined vocabulary, including joints, gears and simple cams. While such techniques are often very useful and efficient, they can not deal with cases that fall outside the finite symbol set. This limitation is hard to overcome because the geometric properties of general objects depend on the metric dimensions in ways that can not easily be captured by a simple set of rules. What we must look for is a *generative* system powerful enough to analyse objects it has never seen before.

What are the requirements for such a generative system? Generativity requires that the reasoning

can be based on a general geometric description of the objects, capable of representing arbitrary shapes. Unfortunately, finding such a representation is itself an unsolved problem. The difficulty lies in the fact that any arbitrary small feature of a physical object may become important in its interaction with others. A symbolic representation groups objects into equivalence classes, losing information about the features that vary among objects in the same class. But these features may be the ones that determine their interaction! For example, representing the diameter of pegs and holes by a set of intervals makes it impossible to determine whether a peg will fit into a hole whose diameter falls within the same interval. A key idea of this thesis is that symbolic representations should be constructed for *pairs* of objects, not single objects themselves. This determines the important features of the objects and allows us to take just those features into account in the symbolic representation. The idea itself is not new: In 1875 Franz Reuleaux ([REU75,REU76]) proposed that the elementary parts of mechanisms are not the objects themselves, but the interactions between pairs of them. A possible drawback of this approach is that it requires a separate representation for each pair of objects, a much larger number than the number of objects itself.

The representation we propose for representing the possible interactions of pairs of objects is called a *Place Vocabulary.* The idea is that the space of parameters describing the positions and orientations of the objects can be broken up into a finite number of regions, called *places.* All points of the parameter space within a place are considered equivalent in the understanding of the mechanism. The places thus define a symbolic *vocabulary* in which the configurations of the mechanism can be expressed. The possible motions of the mechanism are represented as changes in its configuration. In the place vocabulary, they are expressed as transitions between places, given by the adjacencies between the regions that define them. As the place vocabulary covers the space of all legal configurations, it provides a complete representation of all possible kinematic behaviors of the mechanism. The place vocabulary idea itself was first proposed by Ken Forbus in ([FOR81,FOR80]). This thesis generalizes the original concept, which was defined for motion of points, to the general case of geometric objects.

In the rest of the introduction, we first define the mechanism domain and outline its particular implications for our theory. We then give a brief overview of qualitative reasoning and show what the requirements for place vocabularies are. We then go on to define place vocabularies, and finally discuss some pertinent issues important to their computation from information about the shapes of

the objects.

The rest of the thesis is organized as follows: Chapter 2 analyzes the mathematics of mechanism kinematics necessary for the understanding of the place vocabulary theory. Chapter 3 discusses the theory of how place vocabularies can be computed, and Chapter 4 gives details of the actual implemented algorithms. Chapter 5 discusses the implications of the research for qualitative kinematics, and Chapter 6 gives examples of place vocabularies. Finally, Chapter 7 reviews other related work, outlines further research issues and gives the conclusions.

1.1 Mechanisms

As the general problem of qualitative kinematics is too unconstrained to be analyzed in general, we have restricted the domain of our theory to *mechanisms* in 2 dimensions. There are several reasons for this choice of domain:

- Mechanisms exhibit interesting and complex kinematics.
- Mechanisms have been studied since ancient times, so it is a well-understood domain.
- There is practical interrest in reasoning about mechanisms.
- Mechanisms are man-made devices. Their kinematics are such that people are good at analyzing them. Techniques which are useful for mechanisms might thus lead to the "correct" theory of human spatial competence.

Following the classic work of Reuleaux ([REU75,REU76]), we define a mechanism as an implementation of a *kinematic chain.* In simple terms, a kinematic chain is a device that transmits the motion of and force acting on an input element through the chain to an output element. An obvious example is a gearbox in which the motion and forces are transmitted from an input gear to an output gear.

A kinematic chain is made up of *kinematic pairs.* A kinematic pair consists of 2 objects which can touch and thus influence each other. Reuleaux makes the distinction between *higher pairs* and *lower pairs.* If the objects can touch at pairs of surfaces only, the pair is a *lower pair*; if contacts between a vertex and a surface are also possible, the pair is a *higher pair.* The lower pairs represent what are commonly called joints: One object is free to move with respect to the other object in one or more directions. There exist only 6 possible lower pairs, namely revolute, prismatic, helical, cylindric,

spheric and planar joints between objects ([REU75,REU76]). Their symbolic representation is thus straightforward, and there exists an exact mathematical analysis ([DEHA55]) that can be used as a basis for reasoning about them.

Most of the interresting functions in mechanisms are performed by higher pairs, such as gearwheels, escapements, ratchets and so on. An important feature of higher pairs is that motion of the objects can change the contact configuration. There are infinitely many possible higher pairs, and their symbolic representation thus requires a generative language. In this thesis, we propose place vocabularies as such a representation language. As the analysis of lower pairs is a very different problem from the analysis of higher pairs, and furthermore has already been solved by Reuleaux, we restrict our theory to the analysis of higher pairs only.

An important feature of higher pairs is that both of the objects involved are somehow attached by joints such that they can only move in a single degree of freedom. Motion in one degree of freedom can be either rotation around some fixed point in space or translation along a fixed straight line. For example, in gears, both parts can only rotate, and in cams, one part can rotate while the other can only translate in one particular direction. This property is important because it means that the configuration of the kinematic pair can be specified by 2 parameters.

While the kinematic pairs make up the elementary elements of a mechanism, its function is accomplished by the kinematic chains. A kinematic chain is formed by rigid connection between parts involved in successive kinematic pairs. The configurations of the pairs are related by the position parameter of the shared object. This makes it possible to analyze the chain by composing analyses for the constituent kinematic pairs. Our theory is split into 2 parts: the analysis of the kinematic pairs, which is the main body of the theory, and the composition of the place vocabularies derived for kinematic pairs. The composition turns out to be a simple problem of 1-dimensional interval intersections. It can be handled by established techniques of qualitative reasoning.

A mechanical device may consist of more than a single kinematic chain, and the individual chains may influence each other via common elements. The possible chains can be identified in the mechanism by constructing a graph whose nodes are the parts of the mechanism and the edges represent all possible kinematic pairs between them. Each path in this graph is a kinematic chain. Importantly, because each chain determines motion and force of each of its elements, most practical mechanisms

have no cycles in this graph. In a cycle, the motion of each element would be determined by at least 2 kinematic chains simultaneously. If each object has only one degree of freedom, both chains determine the same single motion parameter. Besides being redundant, the influences from the 2 chains will have to agree exactly, which is true only in singular cases. While there are mechanisms that have such cycles, such as differentials, they have parts with more than a single degree of freedom and are not covered by our current theory. Thus, for any pair of parts, there exists at most one kinematic chain relating them. The influence that the 2 parts have on each other can be analyzed by analysis of that single kinematic chain.

Finally, in order to make the problem more tractable for research, we consider mechanisms in 2 dimensions and restrict the boundary curves of the parts considered to straight lines and arcs only. The main reason for the restriction to 2 dimensions is that it allows inspection of the problem. Human intuitions for 3 dimensions are very poor, and this makes it difficult for the researcher to analyze the problem correctly. Furthermore, because people are not capable of analyzing 3-dimensional problems very well, we also believe that the correct cognitive theory is more likely to be based on 2-dimensional analyses of projections of the 3-dimensional objects rather than a straightforward analysis in 3 dimensions.

The boundary curves are restricted to simplify the implementation. The place vocabulary theory is valid for arbitrary boundary curves, but implementing this requires general algebraic manipulation routines which would detract from the main effort of the thesis.

To summarize, the domain of the theory to be described is kinematic chains in 2-dimensional mechanisms, consisting of higher pairs in which both objects have one degree of freedom each. This covers a large range of practical mechanisms, including mechanical clocks. Our theory could be profitably combined with work such as Andrew Gelseys ([GELSEY87]) to build a system capable of analyzing almost all practical mechanisms.

1.2 Qualitative Reasoning and Kinematics

The theory in this thesis falls within the framework of qualitative reasoning and naive physics ([FOR84,DKL84,FOR81,DKL79,HAY79,DKL75]). The goal of naive physics is to develop a for-

mal theory of people's commonsense knowledge of the physical world. While there are well-developed mathematical theories about physical phenomena, they presume a large amount of *qualitative* knowledge needed to properly apply them to physical situations. As commonsense knowledge does not make use of numbers, theories of naive physics employ *qualitative* reasoning. Quantities are manipulated not as numbers, but as qualitative expressions. Most theories of qualitative reasoning represent quantities by their sign only, i.e., by one of +, 0 and -. While this representation seems extremely weak, it actually turns out to be sufficient to accurately predict the behavior of many physical systems. A refinement of the simple +, 0 and - representation is the notion of a *quantity space* ([FOR84]), which is a graph whose edges represent known relative magnitudes of the quantities making up the nodes.

Theories of naive physics represent the possible dynamic behaviors of physical systems as *envisionments*. An envisionment is a directed transition graph whose nodes are *qualitative states* and whose edges indicate possible transitions between them. A qualitative state of the system is defined by a choice of qualitative value for each of the quantities in the system, and possibly a set of symbolic switches indicating discrete changes in the physical system. The envisionment represents all behaviors of the system that are perceived possible, thus, each path through the graph represents a possible actual behavior. In general, there is more than a single possible transition from each qualitative state, so that the envisionment is ambiguous. It represents the physical system as a non-deterministic automaton which accepts the set of possible behaviors of the device.

In order to apply qualitative reasoning to problems of kinematics, we have to find a useful definition of a qualitative kinematic state, or *place*. Note that kinematic interactions between objects occur only when they are in contact, and that the form of the interaction depends on where this contact occurs. Reasoning about kinematics must therefore be based on configurations of contact, and changes in it must lead to a change of place. This gives us the first criterion for the definition of places: each qualitatively different contact configuration defines a place as the set of all configurations that exhibit this particular configuration of contact.

We must also take into account that we want to reason qualitatively about the transmission of forces and motion through the mechanism. This is helped greatly by requiring that the qualitative relations between these parameters be constant within each place. For example, if a positive moment on the input element of a kinematic pair causes a positive moment on the output element in one

configuration and a negative moment in another, the 2 configurations should lie in distinct places, even if the contact configuration is otherwise identical.

Finally, to predict the dynamics of mechanisms, we represent the adjacencies between the places so as to restrict the possible transitions between them. This information is given by the *place graph*, a graph whose nodes are places and whose edges are the adjacencies between them. The place graph serves as the spatial substrate for an envisionment of the mechanisms behavior.

In the next section, we give a formal definition of places, using the abstraction of *configuration space*, and give an example of its application.

1.3 Place Vocabularies

The concept of a place vocabulary originated in FROB ([FOR81,FOR80]). This work investigated the qualitative analysis of the motion of point masses in polygonal, two-dimensional regions under the influence of gravity. The qualitative representation of the physical space was derived by dividing it into regions, called *places*, and arranging them in a connectivity graph.

Because FROB considered the motion of *points*, and the spaces considered were plane polygonal regions, the places could be obtained by a tesselation of the physical space using only horizontal and vertical lines. In this paper, we generalize the notion of a place to moving objects of finite dimensions by making an abstraction from physical space to *configuration space*. This transforms the constraints on the motion of the physical objects into constraints on the motion of a point in configuration space. Similar to FROB, we divide up this space into regions corresponding to places. However, our domain requires us to take into account many additional constraints, making this division a much more complicated process.

1.3.1 Configuration space

The position of a physical object can be described by a small set of parameters. In the case of unrestricted motion in 3 dimensions, there are 3 Euclidian position parameters and 3 orientation parameters which completely determine the placement of the object in physical space. In mechanisms, the motions of the parts are restricted by joints. By our definition, a single parameter suffices to

describe the position of a part. We call the space spanned by the parameters characterizing the positions of all the objects of a mechanism its *configuration space* ([LPW79,BLP83,DON84]). A particular choice of position of all the parts of the mechanism is called a *configuration* and is specified by a point in this space. As the parts of a mechanism are not allowed to overlap, the configuration space consists of regions corresponding to legal and illegal configurations. We call the union of all legal regions the *free space* and its complement the *blocked space.* The condition for a configuration to fall on the boundary between free and blocked space is that there is both a point in blocked space and a point in free space that can be reached by an arbitrarily small motion. This is the case if and only if the 2 objects are in contact.

There are 2 possible ways that plane objects O_a and O_b can touch:

- A vertex of O_a touches a boundary segment of O_b, or vice versa. Note that this subsumes the case where a pair of vertices touch each other.
- A boundary segment of O_a touches a boundary segment of O_b.

Each possible instance of these cases defines a *configuration space constraint* in the configuration space for O_a and O_b. We call the first *vertex constraints* and the second *boundary constraints.* A configuration space constraint is an algebraic curve containing all the configurations where the defining condition is satisfied. As each point of touch rules out motion in the direction of the surface normal at the point of touch, it reduces the degrees of freedom of the objects by 1. The dimensionality of the constraints is thus exactly 1 less than that of the configuration space itself. The algebraic curve can have infinite length, but only a finite part of it actually corresponds to configurations satisfying the condition. Its applicability is therefore restricted by *applicability constraints* ([DON84]). The actual boundaries between regions of free and blocked space are defined by the envelope of the applicable segments of the constraints. Constraints are *subsumed* wherever they fall inside this envelope.

The idea of configuration space was developed for the problem of motion planning, where it is reasonable to approximate objects by polyhedra. Earlier formulations of configuration space restricted themselves to vertex constraints only, as the touch of a pair of straight lines implies a touch of one of the vertices of the ends, which is already described by a vertex constraint. When we consider kinematics, however, an approximation of a curve by line segments results in spurious discontinuities in behavior. A wheel that is approximated by a polygon will not run smoothly, contrary to its actual

behavior. We can not make this approximation, and have therefore extended the theory to boundary constraints.

The configuration space is defined by the motions we consider. In general, a mechanism has an underlying configuration space of a finite, but possibly enormous, number of dimensions, describing the simultaneous locations of each of its parts. In the analysis, one considers interactions between only small numbers of parts, ideally kinematic pairs. The dimensionality of the configuration sub-spaces for these analyses is then much smaller (usually 2 for kinematic pairs), so that the problem remains tractable no matter how large the complete mechanism is.

1.3.2 Place vocabularies

The place vocabulary is defined as a set of suitable regions of configuration space. In this section, we show how the regions can be defined so that the resulting place vocabulary satisfies the criteria we have developed earlier.

The first and most important requirement is that the place vocabulary must distinguish different configurations of contact. Note that the configuration space constraints themselves are defined by different points of contact. Thus, the configuration of contact in some particular arrangement of the objects is given by the constraints that are satisfied. When there are no such constraints, the objects are not in contact and therefore can not interact.

Note that if the objects are in contact in more than one point, their configuration satisfies several constraints at once and lies on the intersection of these constraints. Each point of contact reduces the freedom of the objects by one. The region of configuration space containing all configurations with k contacts thus has dimension k less than the dimension of the full configuration space. When the full configuration space has dimension n, we find configurations with 0 up to n contact points, and the dimensionality of the places thus ranges from n down to 0.

The configuration of the mechanism can leave a place either by establishing a new point of contact or by breaking one. It thus moves to a place whose dimensionality is either one greater or one less than that of the current one. Note that in general, a region of dimension d in space is bounded by surfaces of dimension $d-1$, and can itself bound regions of dimension $d+1$. This is exactly the arrangement of the places in configuration space; it is called a *cell complex.*

So far, we have referred only to the topology of the places. This is a very weak notion, as it allows the places to have arbitrarily complicated shapes. We need a more precise definition of the regions corresponding to places. The description we propose for this is that of *quasi-convex* cells. By this term, we mean cells defined by a set of bounding curves (surfaces) $C_i(x) = 0$ such that all of the points in the cell satisfy a certain conjunction of sign conditions on the C_i. These conditions are either inequalities or equalities, where each equality indicates the existence of certain points of contact between the objects. The C_i are given by the equations of the constraints that bound the place in question. Ideally, we would like the places to be convex, but as their boundaries are algebraic curves, this is not always possible. The quasi-convexity criterion approximates the idea of convexity as closely as possible. If the bounding curves are plane, it is in fact just the convexity condition.

Each connected place that does not already satisfy this condition is broken up into cells that each satisfy the quasi-convexity condition. This requires some additional curves to distinguish between the regions. It turns out that the applicability constraints define a set of such curves that guarantees a quasi-convex tesselation, so that its computation is straightforward.

We have thus obtained a uniform mathematical notation in which places corresponding to any number of contact points can be described. The one condition we have not considered yet is that of having constant qualitative relations between the motion parameters of the objects. In the case of motion parameters, the relation between 2 motion parameters is given by the derivative of the place in a coordinate system made up of the 2 parameters. This derivative exists only if a relation between the 2 parameters is actually enforced by some constraint, and is then given as the derivative of the constraint in that coordinate system. It is qualitatively constant whenever the constraint is monotone in the coordinate system. This results in a monotonicity condition on the places; we show later how it is satisfied.

It has been shown ([MAS79,ERDMANN84]) that forces and moments in physical space are equivalent to forces on the corresponding point in configuration space and can thus be analyzed in configuration space. It turns out that the monotonicity condition is sufficient to guarantee a qualitatively constant relation between forces as well. The dynamical analysis of mechanisms based on the place vocabulary is the topic of current research by Paul Nielsen ([NIE88]), and the reader is referred to his work for a more detailed analysis.

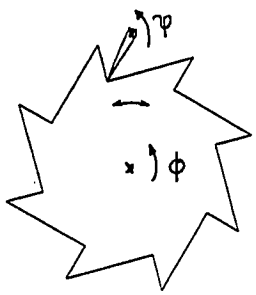

Figure 1.1: Ratchet example.

1.3.3 Example of a place vocabulary

We have now defined the notion of a place vocabulary and are ready to look at a simple example. Consider the ratchet mechanism shown in Figure 1.1. This example is also described in more detail in the example chapter. Both the lever and the wheel are hinged and can rotate around the points indicated by the crosses. In normal use of this mechanism, the lever is pulled down onto the wheel by gravity or a spring. When the 2 objects reach the configuration where the tip of the lever touches the bottom of a tooth of the wheel (configuration a in Figure 1.2), the wheel can not turn to the right. The function of the ratchet is to allow the wheel to turn to the left, but not to the right. This function is achieved when the lever touches the wheel in configurations where it points to the left (configurations a, b, c, d). The lever can also be turned over to point to the right, such as in configurations e and f. In these configurations, the wheel can turn in either direction and the function of the ratchet is not achieved.

The configuration space for this example is shown in Figure 1.3. The 2 parameters spanning the configuration space are the orientation ϕ of the wheel in the horizontal and the orientation ψ of the lever in the vertical. The shaded region in the configuration space is the blocked space and all configurations within this region are illegal. The boundary between free and blocked space is made up by segments of constraints corresponding to different configurations of touch between the objects. The constraint segments are labeled to indicate the configurations they correspond to. The free space is tesselated by the applicability constraints as indicated by the dashed lines.

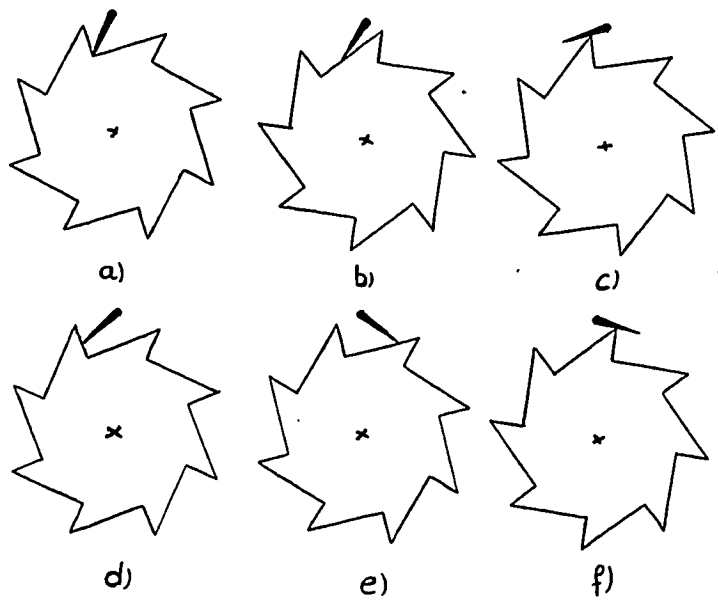

Figure 1.2: Examples of possible configurations for the ratchet.

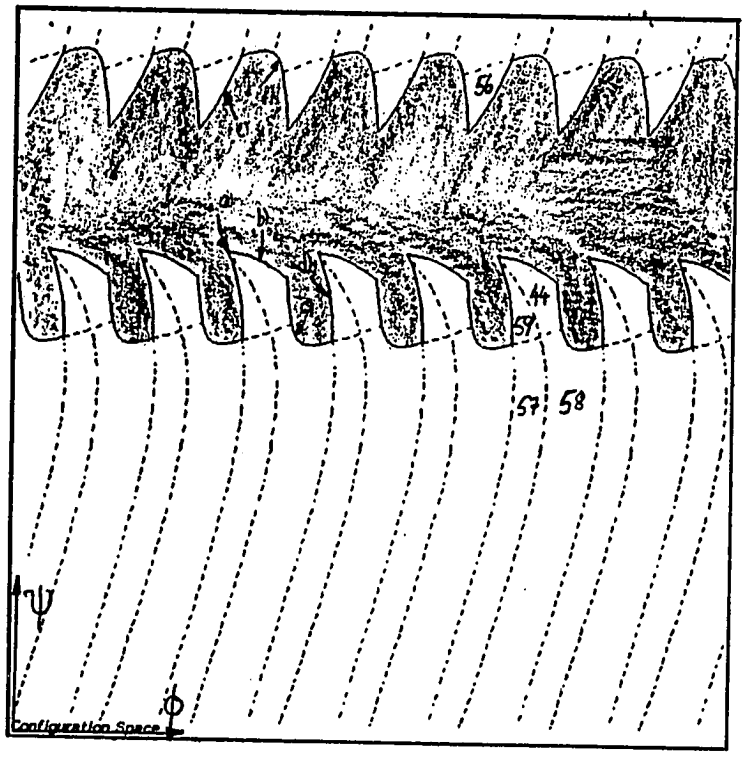

Figure 1.3: The configuration space for the ratchet example. The horizontal parameter is the orientation ϕ of the wheel, the vertical parameter the orientation ψ of the lever.

To illustrate the meaning of the configuration space, let us envision the normal operation of the ratchet. Consider configuration a). The point in configuration space corresponding to this configuration has no adjacent free-space to its left, thus there is no possible motion in the negative ϕ direction. The wheel cannot turn to the right, as this would decrease its orientation angle ϕ. Consider the wheel turning left from this point, and let the lever be pushed onto the wheel by gravity. We now reach configuration b). Further turning the wheel to the left, at some configuration the boundaries of the wheel and the lever become parallel, and the configuration of touch changes into one as shown in configuration c). Turning further left, we reach the configuration where the tip of the lever touches the tip of the tooth on the wheel. Following this, the lever "drops down" to reach a configuration like b) again. The cycle then repeats.

If we were to turn the wheel to the right from configuration c), we reach configuration b) and then configuration a). From a), we can not turn the wheel any further, and the ratchet is blocked. Note that configuration d) is not reachable given our assumptions; gravity exerts a moment on the lever in such a way that such a contact is not stable. However, this configuration is ruled out only by the external forces applied to the kinematic pair, not by the shapes of the parts themselves. If the external forces were different, these configurations could well become reachable. Therefore, they must be included in the place vocabulary.

As the configuration space in this example is 2-dimensional, the place vocabulary contains places of dimensionality 0, 1 and 2. The 2-dimensional places correspond to the regions of free space formed by the tesselation. There are 5 such places for each tooth of the wheel. The 1-dimensional places are the constraint segments that make up the boundary between free space and blocked space. The 1-dimensional places correspond to the configurations of touch b) through f) in Figure 1.2. Some of the constraints have extreme points where their derivative becomes zero. They are broken up at these points to satisfy the monotonicity criterion. The 0-dimensional places are intersection points between constraint segments. An example of a 0-dimensional place is configuration a) in Figure 1.2.

The adjacencies of the places in configuration space define the place graph. It is described in more detail in the example chapter. The place graph forms the spatial substrate for an envisionment of the ratchet's behavior. In the case of a negative external moment on the wheel (making it turn to the right), and a non-negative external moment on the lever, the places corresponding to the locked

position of the ratchet are characterized by the fact that there is no transition going out of them. The fact that the envisionment contains such a sink when the external moment is negative, but does not contain any when the moment is positive can be taken as a "definition" of a ratchet mechanism in qualitative terms.

1.4 Computing Place Vocabularies

While the place vocabulary representation itself is symbolic, the metric dimensions of the physical objects are required to compute it. As in FROB, we propose the model of a *metric diagram* as a plausible way of using this information in the computation. It turns out that this model also has other important advantages, for example, it allows us to solve mechanism design problems.

A plausible theory of human spatial competence can not assume that the eye furnishes information about the objects in terms of algebraic curves and numerical coefficients. The work of David Marr ([MARR82]) indicates that low-level processing in the eye produces a *primal sketch.* The elements of this primal sketch are symbols standing for lines and vertices (discontinuities) found in the image, and symbolic links between them. In the case of a person looking at the parts of a mechanism, or a design drawing of it, the primal sketch at some level contains a description of the object boundaries as a sequence of boundary segments and vertices between them. [1] Note that this is exactly the information that is needed to define a the set of all configurations of touch that could be possible between the 2 objects.

The actual arrangement of the constraints in configuration space is determined by the metric dimensions of the objects. This information can not be represented in the primal sketch. We assume that it is contained in an abstract device called the *metric diagram.* The information in the metric diagram is accessible via an *evaluator.* This is an abstract function that can evaluate *predicates* on the metric dimensions of the objects. These predicates are of the form $exp = 0$, where exp is some algebraic expression involving distances between the vertices and boundary segments defined in the primal sketch. The metric diagram is an abstract device, and could be implemented in various ways. For example, some tests might be implemented by mental imagery, while others might be performed

[1] We are obviously ignoring here the problems of noise in the image, which frustrates vision researchers trying to reproduce this process.

by taking measurements on the actual objects.

The algorithms we describe for the computation of place vocabularies are structured to fit the metric diagram model. It turns out that it is possible to determine the place vocabulary from knowledge of the set of possible constraints using only evaluation of predicates; the algorithms we describe can all be implemented in this way. However, in our current implementation, the metric diagram is implemented by algebra. For efficiency reasons, the division between it and the symbolic computation is not explicitly enforced.

Besides its plausibility as a commonsense model, the metric diagram also gives us an interesting way to solve design problems. Consider the problem of choosing a suitable value for a metric parameter in a mechanism. In the ratchet example, we might want to choose the distance between the centers of rotation of the lever and the wheel. In the computation of the place vocabulary, some or all of the predicates evaluated by the metric diagram make use of the value of this open parameter. The expressions whose sign these predicates test can thus be understood as polynomials in this parameter. For each predicate, the roots of this polynomial divide the range of the parameter into intervals within which the sign of the polynomial, and thus the value of the predicate, does not change. We call these roots *landmark values.*

Now consider the set of all such landmark values for all predicates that are used in the place vocabulary computation. The result of the place vocabulary computation can not change except at a landmark value, as the parameter influences it only via the predicates. This means that the complete set of landmark values divides the range of the parameter into a finite set of intervals such that the place vocabulary is the same for all choices of parameter in each interval. By computing the place vocabulary for a representative value within each interval between landmark values, we can thus make a *complete* list of all place vocabularies that can be achieved by variation of the parameter. Our implementation can be used to solve some such design problems, and an example of this is given in Chapter 6.

Finally, we will outline how our algorithms could be plausibly implemented using the ideas of computer vision, especially the theories of the curvature primal sketch ([ASBR86]) and visual routines ([ULLMAN84]). We find that some plausible modifications to these theories are required for qualitative kinematics. We hope that the place vocabulary theory can be developed into a theory of

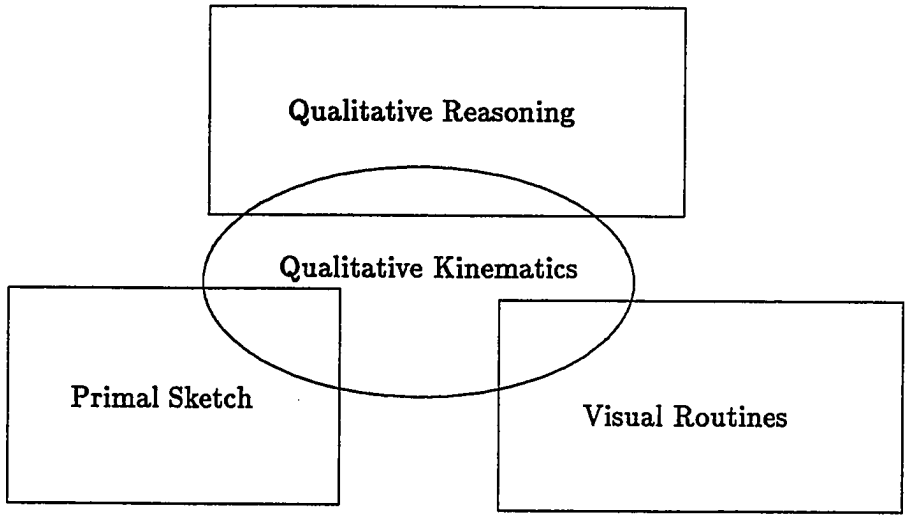

Figure 1.4: Qualitative kinematics as a link between vision and reasoning.

human spatial competence. This would link the computer vision theories at the bottom to theories of reasoning at the top, as shown in Figure 1.4. Such a link will greatly strengthen all the theories involved through the mutual constraints it imposes. It would also put experimental verification of the theories within reach. Note that because the case of mechanisms is a special case of the general kinematics problem, the constraints it imposes on the theories generalize.

Chapter 2

CONFIGURATION SPACE CONSTRAINTS

Kinematics is based on the fact that no two physical objects can occupy the same space at the same time. This condition defines *constraints* on the relative positions of the objects. These form the borders between the continuous spaces of legal and illegal configurations of the objects. We use the term *configuration* to refer to a specific choice of values for the motion parameters of the objects, and *configuration space* to refer to the space of configurations. It consists of a set of legal configurations, called *free space*, and a set of illegal ones, called *blocked space*. The dimension of configuration space is equal to the number of degrees of freedom that the objects have relative to each other. The topology of the configuration space need not be Euclidian. In the case where objects can rotate relative to each other, it includes special order groups, such as SO(3) (homotopic to a sphere with opposite points identified) in the case of unrestricted 3-dimensional rotation. In our case, where we restrict our analysis to 2 dimensions, it can contain S(1), homotopic to a circle.

Configurations that exactly satisfy a constraint are characterized by the fact that both legal and illegal configurations can be reached from them by an arbitrarily small motion. This condition is satisfied only if the 2 objects touch each other: the constraints are thus given as a subset of those configurations where the objects are in contact.

More precisely, we assume that the boundaries of the objects consist of discontinuities, which we call *vertices*, and smooth *boundary segments* between them. The condition that 2 objects are in contact is then satisfied whenever a vertex or boundary segment of one object touches a boundary segment of the other object. [1] We call the constraints that are formed by a vertex touching a boundary segment *vertex constraints*, and those formed by 2 convex boundary segments touching each other *boundary constraints*. When the objects are free to move relative to each other, the configurations that satisfy this condition form a *constraint curve* in configuration space. As the object boundaries in general cover only a part of the algebraic curve that describes them, the constraint curves also consist of a *valid segment* and an *invalid segment*. The valid segments can be delimited using *applicability constraints*.

Not all configurations where the objects touch each other are actually on the border between free space and blocked space, as the configuration may itself be illegal due to other overlaps at other parts of the objects. In such a case, the configuration violates other constraints, which *subsume* the constraint that the configuration lies on. The border between free space and blocked space is formed by the *envelope* of the constraints. The envelope consists of segments of constraints. Each segment is delimited by its intersections with others. These intersections occur either where the constraint ceases to be valid, or where it is intersected by a subsuming constraint. We call these latter types of intersections *subsumption intersections*. In principle, any pair of constraints can form such intersections, which makes the analysis of configuration space constraints very expensive.

Before a detailed discussion of the form of the constraints, we have to define the notation we are going to use to describe configurations and our input representation. We then derive the algebraic form of the constraints for the cases of 2-dimensional objects whose boundaries are made up of arcs and straight lines. Finally, we further refine the analysis for the case of 2 objects with one degree of freedom each, which is the common case in mechanism kinematics.

[1] Note that this includes the case of 2 vertices touching each other, as a vertex is part of 2 boundary segments

2.1 The Configuration Space and the Metric Diagram

In this section, we first describe the notation we use to describe the configuration space for the interaction of a pair of objects. We assume a global Euclidian coordinate system for physical space. For each object, we assume a local coordinate system centered on a *reference point.* In 2 dimensions, the position of each object can be described by 3 parameters: 2 coordinates for its reference point, and an angle giving the orientation of the local coordinate system with respect to the global one. However, because the transformation from local to global coordinates involves transcendental functions (sine and cosine) of the angle, this formulation will result in non-algebraic constraints. It is therefore advantageous to describe the orientation angle in terms of its sine and cosine, giving a total of 4 parameters for each object. Angles specified in this form can be added and subtracted using

$$\sin(\alpha \pm \beta) = \sin\alpha\cos\beta \pm \cos\alpha\sin\beta \tag{2.1}$$

$$\cos(\alpha \pm \beta) = \cos\alpha\cos\beta \mp \sin\alpha\sin\beta \tag{2.2}$$

The configuration space for the interaction of 2 objects therefore has 6 dimensions and 8 parameters. However, because the interactions between the objects are dependent only on their *relative* positions, the constraints are invariant under common translation of the reference points, and under rotation of the arrangement of the 2 local coordinate systems. This would allow us to reduce the configuration space to 3 dimensions and 4 parameters. However, in such a formulation the configuration space for objects with constrained freedom of motion is not a simple projection of the full space anymore. It is thus more convenient for our purposes to use the full specification.

The position and orientation of each object determine the relation of its local coordinate system to a global one and thus appear in the equations describing its boundaries in the global system. We adopt the following notation: Objects have names of the type O_a, where a is some letter, and the configuration space parameters of object O_a are x_a, y_a , c_a $(= \cos\phi_a)$, and s_a $(= \sin\phi_a)$. Points in the local coordinate system of O_a are denoted by (x_i^a, y_i^a), and in the global coordinate system we simply omit the superscript, i.e., (x_i, y_i). Note that we then have the transformations

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{bmatrix} c_a & -s_a \\ s_a & c_a \end{bmatrix} \begin{pmatrix} x_i^a \\ y_i^a \end{pmatrix} + \begin{pmatrix} x_a \\ y_a \end{pmatrix} \tag{2.3}$$

and

$$\begin{pmatrix} x_i^a \\ y_i^a \end{pmatrix} = \begin{bmatrix} c_a & s_a \\ -s_a & c_a \end{bmatrix} \left\{ \begin{pmatrix} x_i \\ y_i \end{pmatrix} - \begin{pmatrix} x_a \\ y_a \end{pmatrix} \right\} \tag{2.4}$$

to transform between the global and local coordinate systems.

If the objects O_a and O_b are constrained to one degree of freedom only, the configuration space becomes a two-dimensional surface. This does not always mean, however, that it can be described by 2 parameters only; rotation angles must be described by the sine and cosine in order to allow algebraic formulation of the constraints.

The objects considered and their freedom of motion are described in a *metric diagram.* It consists of a symbolic and a metric part. As a symbolic description of the parts, we assume the following boundary-based representation:

- an *object* consists of a set of *boundaries* and a local coordinate system centered at the *reference point.*
- a *boundary* is an alternating sequence of *vertices* and *boundary segments* such that each boundary segment connects the 2 vertices adjacent to it, and the first and last vertex of the list are identical. The sequence of the boundary segments in the boundary is assumed to be such that the object is to the *left* of the boundary when it is traversed in order.
- a *vertex* corresponds to a discontinuity in the direction of the boundary, or a change in curvature or algebraic form.
- *Attachments* describe an object's freedom of motion and consist of a symbol describing a type of joint, e.g., :ROT or :TRANS, and possibly a unit vector giving the direction of translation.

In our implementation, we allow the algebraic type of the boundary curves to be either straight lines or arcs. Allowing general algebraic curves would require a general algebraic engine to solve problems like determining intersections and extrema; while the algorithms for such systems exist, it is not the focus of our research to implement them. Note that the symbolic representation we assume could be readily obtained by rearranging the output representation of a vision system such as the *Curvature Primal Sketch* ([ASBR86]).

The symbolic part of the metric diagram defines a set of numeric parameters whose values are given in the metric part. These are the parameters of the algebraic equations of the boundary segments, the

coordinates of the vertices, and the directions of the freedom given by the attachments. Conceptually, we do not explicitly spell out this metric information (as numbers could only be approximated anyway), but assume that it can be accessed via the evaluation of predicates, such as comparisons of algebraic expressions.

This allows treatment of kinematics in a qualitative way: uncertainty about the value of numeric parameters may lead to ambiguities in the results, but will not invalidate them. In fact, it is possible to compute (highly ambiguous) place vocabularies without any numeric information at all. It also means that our algorithms can plausibly be implemented using theories of computer vision, namely Brady's Curvature Primal Sketch ([ASBR86]) and S. Ullman's theory of visual routines ([ULLMAN84]). This is discussed in detail in a later chapter.

2.2 Constraints in the General Case

The idea of configuration space constraints is due to T. Lozano-Perez ([LPW79,BLP83]), with further extensions by B. Donald ([DON84]), and was developed to formulate the constraints on motion planning problems. The derivation of vertex constraints we give here is identical to his formulation. Because we also allow arcs as boundary segments, we then extend the theory to boundary constraints.

2.2.1 Vertex constraints

In this section, we derive the exact algebraic form of the configuration space vertex constraints for the cases of straight lines and arcs. We formulate the constraints on the relative motion of the objects in the 8-dimensional space formed by the C-space parameters of the 2 objects and consider the 2 objects O_a and O_b and their associated C-space parameters.

Consider a vertex of object O_a (x_1^a,y_1^a) touching the straight edge from vertex (x_2^b,y_2^b) to vertex (x_3^b,y_3^b) of object O_b (Fig. 2.1). We transform the coordinates of the vertex into the coordinates of O_b using (2.3) and (2.4)

$$\begin{pmatrix} x_1^b \\ y_1^b \end{pmatrix} = \begin{bmatrix} c_b & s_b \\ -s_b & c_b \end{bmatrix} \left\{ \begin{bmatrix} c_a & -s_a \\ s_a & c_a \end{bmatrix} \begin{pmatrix} x_1^a \\ y_1^a \end{pmatrix} + \begin{pmatrix} x_a - x_b \\ y_a - y_b \end{pmatrix} \right\} \tag{2.5}$$

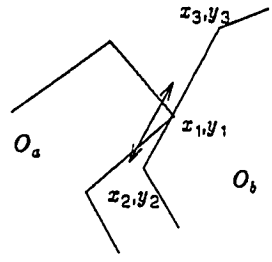

Figure 2.1: Situation analyzed for straight line Vertex Constraints.

which can be used in the line equation defining the constraint:

$$((y_3 - y_2)\,(x_2 - x_3)) \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} - ((y_3 - y_2)\,(x_2 - x_3)) \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \leq 0 \tag{2.6}$$

Note that we have omitted the superscripts since this equation holds in any coordinate system. In order to expose the influence of the configuration space parameters, Equations (2.6) and (2.5) can be combined and rewritten as

$$(c_b\ s_b)\,\mathbf{N} \left\{ \mathbf{X} \begin{pmatrix} c_a \\ s_a \end{pmatrix} + \mathbf{\Delta} \right\} - c_0 \leq 0 \tag{2.7}$$

where

$$\mathbf{N} = \begin{bmatrix} y_3^b - y_2^b & x_2^b - x_3^b \\ x_3^b - x_2^b & y_3^b - y_2^b \end{bmatrix},\ \mathbf{X} = \begin{bmatrix} x_1^a & -y_1^a \\ y_1^a & x_1^a \end{bmatrix},\ \mathbf{\Delta} = \begin{pmatrix} x_a - x_b \\ y_a - y_b \end{pmatrix},\ c_0 = y_3^b x_2^b - x_3^b y_2^b \tag{2.8}$$

The same approach can be generalized to the case of general algebraic boundary curves; however, their form may be considerably more complicated. Below, we show the derivation for the case of a vertex traveling on an arc (Fig. 2.2). The curve of the vertex constraint generated by the situation shown in Fig. 2.2 above is characterized by the condition:

$$(x_1 - x_4)^2 + (y_1 - y_4)^2 \geq R^2 \tag{2.9}$$

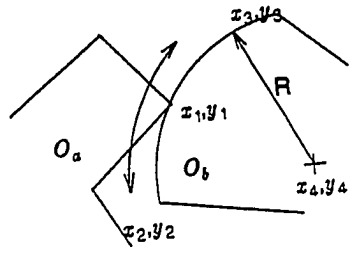

Figure 2.2: Situation analyzed for Arc-Vertex Constraints.

Using the coordinate transformation (2.3), this can be written using local coordinates:

$$\left(c_a x_1^a - s_a y_1^a + x_a - c_b x_4^b + s_b y_4^b - x_b\right)^2 + \left(s_a x_1^a + c_a y_1^a + y_a - s_b x_4^b - c_b y_4^b - y_b\right)^2 \geq R^2 \qquad (2.10)$$

After some manipulation, this equation can be rewritten to expose the influence of the configuration space parameters

$$(c_a\ s_a)\mathbf{X_a}\mathbf{X_b^T}\begin{pmatrix} c_b \\ s_b \end{pmatrix} - (c_b\ s_b)\mathbf{X_b}\Delta - (c_a\ s_a)\mathbf{X_a}\Delta - c_0/2 \leq 0 \qquad (2.11)$$

where

$$\mathbf{X_a} = \begin{bmatrix} x_1^a & -y_1^a \\ y_1^a & x_1^a \end{bmatrix},\ \mathbf{X_b} = \begin{bmatrix} x_4^b & -y_4^b \\ y_4^b & x_4^b \end{bmatrix},\ \Delta = \begin{pmatrix} x_a - x_b \\ y_a - y_b \end{pmatrix},\ and$$

$$c_0 = x_1^{a2} + y_1^{a2} + x_4^{b2} + y_4^{b2} + |\Delta|^2 - R^2$$

Note that the situation shown in Fig. 2.2 shows a convex arc participating in the interaction. In the case of a concave arc, the inequality in Equation 2.9 must be reversed, i.e., we obtain the same constraint equation but the legal side is $\geq R^2$.

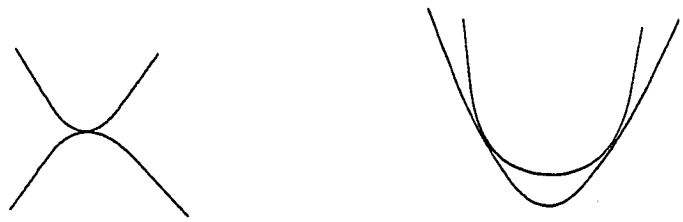

Figure 2.3: Touch of convex (left) and of convex and concave (right) boundaries.

2.2.2 Boundary constraints

The boundary constraints are formed by a convex boundary segment on object O_a touching a boundary segment on object O_b. [2] We have to distinguish between the case where the boundary segment on O_b is convex and the case where it is concave. As shown in Figure 2.3, in the convex case there can exist only a single point of touch between the segments, whereas in the concave case there can be any number of them. [3] In the convex-concave case, each possible point of touch gives rise to a separate constraint surface. These points of touch vary following the *root locus* curves of the minimum distance between the 2 curves. As the number of these is determined by the algebraic degree of the curve, the constraints could not be formulated at all without knowledge of the algebraic form of the curves. The case of multiple points of touch requires a rather complicated algebraic analysis; as it does not seem to occur in mechanisms, we do not discuss it any further in this paper. In the following, we limit our discussion to boundary constraints with only one possible point of touch.

A boundary constraint is thus defined by 2 boundary segments B_a on object O_a and B_b on object O_b touching each other. We assume some parameterization of the boundary segments and use P_a and P_b to refer to the single point of touch in the local parameterization.

A point on the boundary constraint is then characterized by the following conditions:

(i) P_a and P_b must be in the same location.

[2] Two concave boundary segments can not touch one another.

[3] Where the number is bounded by the maximum algebraic degree of the 2 curves.

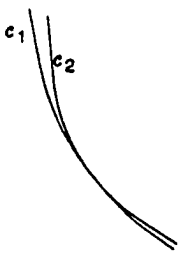

Figure 2.4: C_1 and C_2 fulfill the 2 conditions for touching, but intersect nonetheless.

(ii) The derivatives of both curves must be parallel at that point.

For any choice of points P_a and P_b, these 2 conditions define a fixed relative configuration of the 2 curves. For a pair of convex curves, there exists a legal configuration for any choice of points that satisfies the conditions. For a convex curve and a concave curve, this is not always the case (see Figure 2.4), as the constraint generated by this point might be subsumed by that generated by another possible point of touch. In the general case, the constraint can be obtained as the intersection of the manifolds defined by the 2 conditions. It is possible that this intersection manifold does not exist in closed algebraic form, in which case the constraint must be defined as a conjunction of the 2 manifolds.

In the case where one of the boundary segments involved is a convex arc, however, the boundary constraint is actually identical to an analogous vertex constraint. This is because an arc of radius R touches any curve exactly when its center is at a distance R away from the curve, as shown in Figure 2.5. In our case, where we restrict ourselves to arcs and straight lines, one of the boundary segments involved in each boundary constraints must be a convex arc. A boundary constraint is then identical to a vertex constraint formed by the center of the arc, the *imaginary vertex*, traveling on a curve a distance R from the actual boundary, the *expanded boundary*.

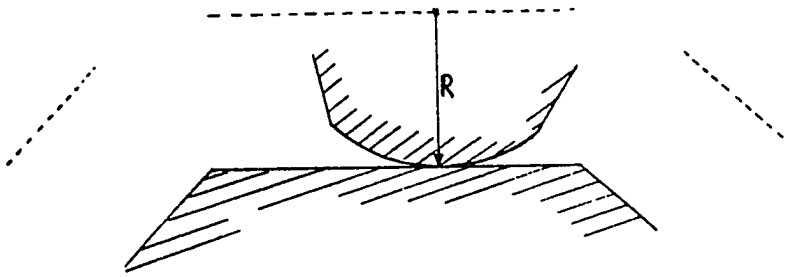

Figure 2.5: An arc touches a curve exactly when its center travels on a curve at distance R.

2.2.3 Determining the valid segments

In the general case, the endpoints of the range of validity of a constraint generated by a vertex and a boundary segment are given by the configurations where the vertex involved in the constraint touches the vertices at each end of the involved boundary segment; we call these configurations *touchpoints*. A touchpoint bounds the validity of up to 4 constraints, namely the ones generated by each of the vertices and the boundary segments that adjoin them (Fig. 2.6). Because a concave vertex of an object boundary can not touch the other object, a pair of a convex and a concave vertex defines touchpoints with only 2 incident constraints, and one of two concave vertices defines no touchpoint at all. Constraints that are generated by the touch of two boundary segments end in a somewhat different manner. Their range of validity is always limited by their intersections with the constraints generated by the vertices at the ends of one of the segments touching the other segment. In the case of boundary constraints generated by arcs using the analogy to vertex constraints, these points can also be computed precisely. Also, the range of validity of such boundary constraints can be bounded by "touchpoints", configurations where the center of the participating arc lies on the endpoint of the expanded boundary segment.

Because the boundary defines a sequence of the segments, there is a natural way to define half of the constraints incident at a touchpoint as *outgoing constraint* and half as *incoming*, depending on whether the boundary segment that participates in the constraint comes before or after the vertex participating in the touchpoint in the ordering of the boundary. This distinction is important because

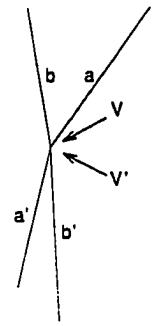

Figure 2.6: The configurations where V and V' touch are endpoints for the 4 constraints arising from V touching a',b' and V' touching a and b.

it defines on which side the active segment of the constraint lies.

The touchpoints define the ranges of validity of the constraint itself. As the constraint function is defined over the whole of configuration space, it is necessary to generalize this restriction to define the domain of configuration space where the constraint is applicable. The device we use for this purpose is *applicability constraints.*

Consider a constraint generated by a vertex V on object O_v and a boundary segment B on O_b. The algebraic curve of the boundary segment divides the space of possible relative locations O_v and O_b into a half-space where V is on the legal side of B and one where V is on the illegal side of B. But the illegal half-space is too large and makes many possible relative positions illegal, as indicated by the dashed outlines in Figure 2.7. The domain of applicability of the constraint has to be restricted to rule out such cases. We do this by defining applicability constraints $A_i > 0$ whose conjunction delimits a convex region of locations of the vertex V relative to the object O_b (as shown in Figure 2.7). This region maps into the quasi-convex region of configuration space defined by the conjunction of the applicability constraints; this is exactly the valid region of the corresponding constraint C.

More precisely, we distinguish the different cases of straight lines and arcs as boundary curves. In the case of a straight line, shown in Figure 2.7, we make the constraint applicable in a thin strip along the boundary curve and bound its applicability by 3 different applicability constraints: A_1 running

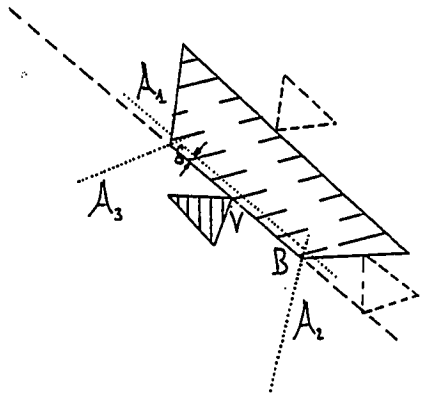

Figure 2.7: Many legal configurations (shown by dashed outlines) violate the constraint generated by B and V. Its domain thus has to be restricted by applicability constraints as shown.

parallel to the boundary curve a small distance δ into the object, and A_2 and A_3 which run through the vertices at the ends of the boundary segment. Note here that we have some freedom in choosing the directions of A_2 and A_3. We use the following rule:

- If the applicability constraint runs through a concave object vertex, it is generated by the vertex traveling on the bisector of that vertex.
- If the vertex is convex, we simply use the adjoining constraint, to save introducing an additional curve.
- In the case of an adjoining arc, we use its applicability constraint.

For arc boundary segments, we have 2 applicability constraints: A_1 running parallel to the boundary a small distance δ into the object, and A_2 as a straight line which distinguishes the valid part of the circle from the invalid one (see Figure 2.8). Note that A_2 can also serve as applicability constraint for an adjoining straight line boundary segment at the vertex at either end of the arc. The applicability constraint A_1 requires a choice of the distance δ. There are 2 competing requirements for this choice: on the one hand, δ must be small enough so that A_1 lies completely within the object in the range determined by the other applicability constraints. If this were not the case, there would be some legal

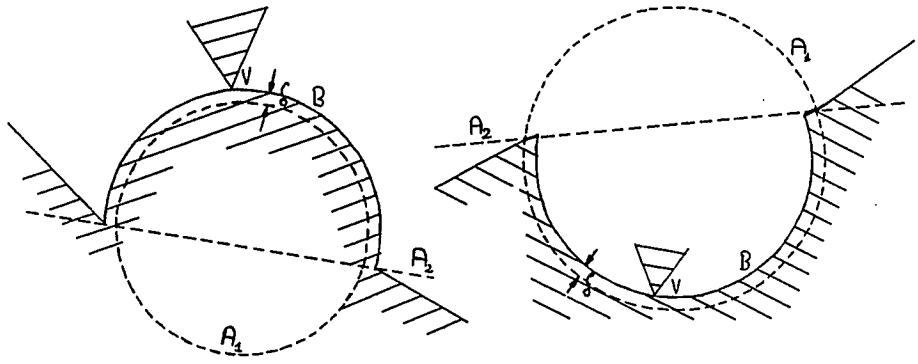

Figure 2.8: The applicability constraints for a convex (left) and a concave (right) boundary segment.

position of the vertex V that nonetheless violates C. On the other hand, if calculations are done using floating point arithmetic with limited precision, δ must be large enough to allow for roundoff errors.

For boundary constraints generated by arcs using the analogy shown earlier, the same applicability constraints apply to the analogous vertex constraint.

Note now that the way we have defined the applicability of the constraints may leave large regions within the object where no constraint applies to declare the corresponding configurations illegal. We have to be careful not to declare such regions a part of free space as well. But notice that they are necessarily disjoint from the real free space: to reach the inside of an object, a vertex has to cross one of the regions declared illegal by the constraints. Furthermore, they are not bounded by proper constraints, but by applicability constraints. These regions can thus be avoided by considering only regions that are bounded by proper constraints to be part of free space.

As a final point, note the following alternative way to define applicability constraints (used by T. Lozano-Perez). For each object, we construct a representation as a set of overlapping convex regions. The condition that a point lies within one of these convex regions is given as a conjunction of equations, and the condition that a point lies within an object as a disjunction of such conjunctions. Blocked space can then be characterized by a disjunction of all these disjunctions, and free space is just the logical complement of it.

In this formulation, the tesselations made on the object to define the convex regions form the applicability constraints. The reason why this approach does not seem to work as well for kinematics is that it is non-local: the applicability constraints depend on the whole object, and can not be found by local analysis of the boundary segment and its environment. This seems to violate our principle of basing the theory on local analyses as much as possible, and that is the reason for choosing the formulation we have given. However, if one wants a theory that is based on constructive solid geometry, an alternative formulation based on descriptions of blocked space might be more appropriate. The other parts of our theory do not make any particular assumption about the choice of applicability constraints; alternative formulations can thus be substituted if necessary.

2.3 Constraints for Kinematic Pairs

In this section, we investigate the case of kinematic pairs of 2 objects, each attached so that it has only a single degree of freedom. The kinds of *attachments* we allow to give the one degree of freedom are either a purely translational or a purely rotational one. This ignores cases where the object can move in some more complicated fashion; they could be incorporated into the theory if a precise definition could be found. Fortunately, such cases are very rare in mechanisms (we have not found a single example), and we do not treat them further here.

With this restriction, the configuration space is two-dimensional, the constraints are plane curves and their intersections are points. The computation of the place vocabulary thus becomes a problem of plane geometry, much simpler than in the unconstrained general case. In the restricted case that we are considering here, the constraints can be very precisely characterized, allowing computation of place vocabularies by other than brute-force methods. In the following sections, we show some of the features of constraints for kinematic pairs.

2.3.1 Topological features

Depending on the different combinations of attachments of the objects, the configuration space can have one of the following different topologies:

Euclidian in the case where both objects have translational attachment.

A cylinder in the case where one object has rotational and one has translational freedom.

A torus in the case where both objects have rotational freedom.

If the configuration space topology is Euclidian, the constraint curves are unambiguous plane curves. In the case where one or both objects have rotational attachment, there are always 2 configurations in which the objects can touch at a certain point. The constraint curve in configuration space is formed as the intersection of the cylinder or torus with the algebraic manifold defining the constraint. This manifold can either

- not intersect the configuration space at all, in which case the constraint is *inactive*.
- touch the configuration space, a singular case.
- intersect the configuration space. In this case, the manifold will form 2 intersection curves. We call one of these the *original*, the other the *mirror image*. This ambiguity is the same as that which may be observed for the configurations of touch.

The convention for distinguishing originals and mirror images will be given at the end of the next section.

2.3.2 Parameterization

The constraint curve can be parameterized by the location of the point of touch between the objects. The most convenient parameter to choose is:

- in the case of rotational attachment: the radius of the point of touch from the center of rotation.
- in the case of translational attachment: the signed distance of the point of touch in the direction of the right-handed normal to the direction of translation.

For simplicity, we shall refer to both of these quantities as *radius*, $\mathbf{r}$. The 2 different cases are illustrated in Figure 2.9.

There are several advantages to this choice of parameter. As we will see later, it simplifies the calculations of extremal endpoints and the breakup into monotone constraint segments. In certain cases, there can be ambiguities as a boundary segment may have several points with the same value for the parameter. In this case, the segment has to be split to make the parameterization unique,

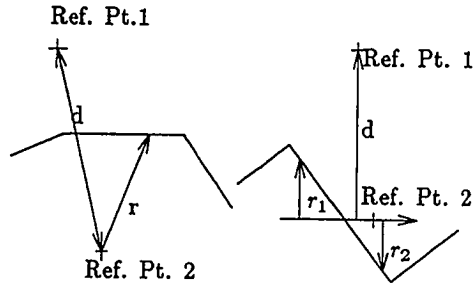

Figure 2.9: The definition of the radius as parameter. Note that r_2 is negative.

as shown in Fig. 2.10. We shall call the segment *decreasing* if the parameter decreases when the segment is traversed from lower to upper end, and *increasing* otherwise. The restriction that each segment has a constant sign curvature is insufficient to bound the number of segments that might have to be made (see Figure 2.10). In the case of arcs and straight lines, it is possible to limit the number of subsegments to at most 3. Along with the radius, we also define the *distance* d between the attachments of the objects, also illustrated in Figure 2.9. We distinguish the following cases:

- When both objects are rotationally attached, d is just the Euclidian distance between the reference points.
- When one object O_r is rotationally attached, and the other object O_t is translationally attached, it is the signed distance of the reference point of O_r in the direction of the right-handed normal to the direction of translation of O_t.
- When both objects are translationally attached, d is undefined.

We compose the parametric description of the constraint segments from 2 parts: the absolute location of the point of touch as a function of the radius and the offset of the point of touch in the local coordinate systems of the objects. The 2 sets of configuration space parameters can be added algebraically.

We describe the particular attachment by a pair (x,y), where x, y stand for the attachments of the 2 objects O_a and O_b and can be one of translation or rotation. We then have the 3 cases shown in Fig. 2.11 below: Shown in Figure 2.11 are the radius vectors of the points of touch between the objects. We now derive the conditions that characterize where these points may lie. Note that the angles/distances shown in the figure are *not* the configuration space parameters, which are obtained

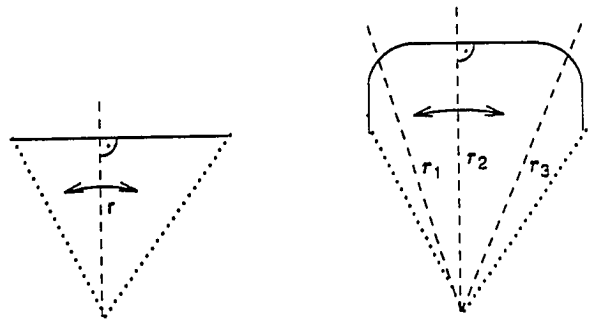

Figure 2.10: The boundary segments must be split at the points indicated by the dashed lines. Note that the boundary on the right has positive curvature throughout and is thus a single segment.

by adding the positions of the points of touch in the local coordinate systems of the objects. We distinguish the 3 cases:

(r-r): We have the 2 equations

$$R_a \cos\phi_a + R_b \cos\phi_b = \mathrm{d} \text{ and } R_a \sin\phi_a = R_b \sin\phi_b$$

from which we find

$$\cos\phi_a = \frac{R_a^2 - R_b^2 + d^2}{2dR_a}\ ,\ \sin\phi_a = \pm\frac{\sqrt{2R_a^2R_b^2 + 2R_a^2d^2 + 2R_b^2d^2 - R_a^4 - R_b^4 - d^4}}{2dR_a}$$

Expressions for $\cos\phi_b$ and $\sin\phi_b$ can be found by symmetrical relations. Note that the roots in the expressions for $\sin\phi_a$ and $\sin\phi_b$ are identical. Note that in order for the angles to refer to the same global coordinate system, the sign of $\cos\phi_b$ must be inverted.

The sign ambiguity in the expression for $\sin\phi_a$ refers to the distinction between constraint *original* and *mirror image*. We choose the positive sign for the original and the negative sign for the mirror image.

(r-t): We have the equations

$$R_a \cos\phi_a + R_b = \mathrm{d} \text{ and } R_a \sin\phi_a = x_b$$

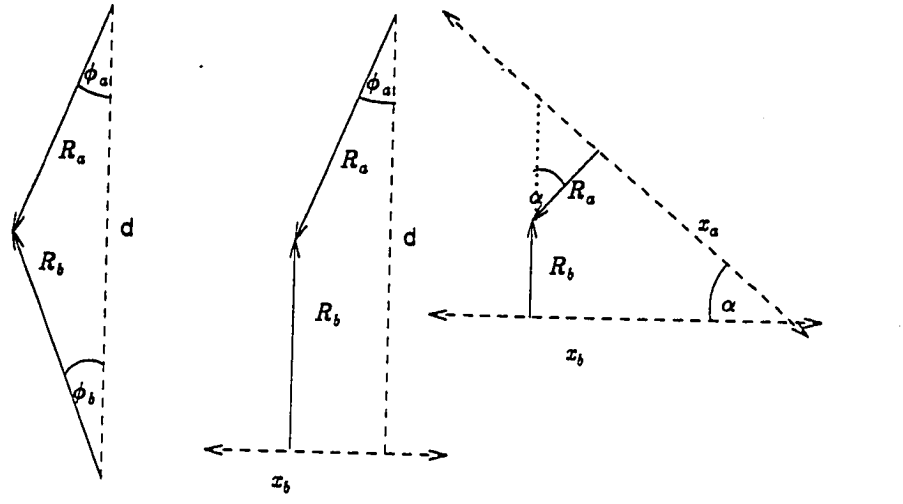

Figure 2.11: The 3 different possible attachment combinations: (r-r), (r-t) and (t-t). The arrows meet at the point of touch of the objects.

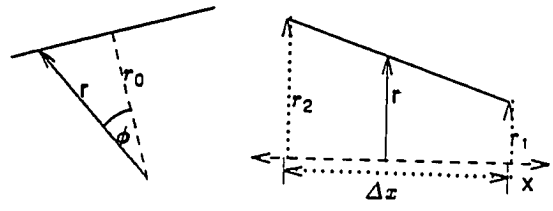

Figure 2.12: Construction of local offsets for straight boundary segments.

from which we find

$$\cos\phi_a = \frac{d - R_b}{R_a}\ ,\ \sin\phi_a = \pm\sqrt{1-\cos^2\phi_a} \text{ and } x_b = \sin\phi_a R_a$$

The sign ambiguity in the expression for $\sin\phi_a$ again refers to the distinction between *original* and *mirror image* of the constraint, where the positive sign is chosen for the original and the negative one for the mirror image.

(t-t): We have the equations

$$\frac{R_a}{\cos\alpha} + R_b = x_b tan\alpha \text{ and } \frac{R_b}{\cos\alpha} + R_a = x_a tan\alpha$$

which can be rewritten to give

$$x_a = \frac{R_a}{tan\alpha} + \frac{R_b}{\sin\alpha} \text{ and } x_b = \frac{R_a}{\sin\alpha} + \frac{R_b}{tan\alpha}$$

To use these equations to derive a parameterization of the constraints, we have to find the offset in ϕ or x of the point on the boundary segment with the particular radius. This offset depends on the form of attachement of the particular object.

In the case of straight line boundary segments shown below in Figure 2.12, we have for the offset angle ψ from the point where r is at its minimum r_0

$$\cos\psi = r_0/r,\ and \sin\psi = \sqrt{1-\cos^2\psi}$$

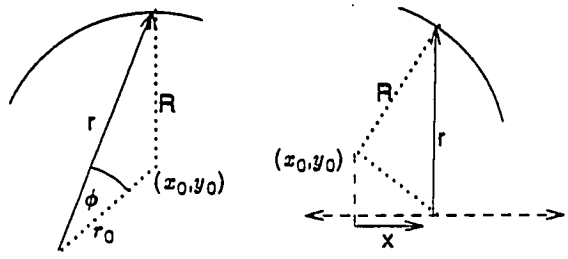

Figure 2.13: Construction of the local offsets for arc boundary segments.

and for the offset x,

$$x = \frac{r - r_1}{r_2 - r_1}\Delta x$$

For the case of arcs, shown in Figure 2.13, we have by the law of cosines

$$R_2 = r_2 + r_0^2 - 2rr_0\cos\phi$$

and thus

$$\cos\phi = \frac{r^2 + r_0^2 - R^2}{2rr_0}$$

The expression for $\sin\phi$ can be found from $\cos\phi$. In the case of translational attachment, we obtain

$$x = \pm\sqrt{R^2 - (r - y_0)^2} + x_0$$

While we can not give a general solution here for boundary constraints, in the case of a boundary constraint involving an arc, we can use the analogy to vertex constraints to arrive at a parameterization. In this case, the boundary constraint is *not* parameterized by the radius of the actual point of touch, but rather by the radius of the point where the imaginary vertex touches the expanded boundary. However, the 2 parameters are related by a one-to-one mapping, as will be shown in a later section.

Note that the constraints can not all be algebraically parameterized in a single parameter. Algebraic parameterization is achieved by introducing new parameters, which are bound to the values of the required transcendental expressions. This results in a parameterization by several algebraically related parameters.

2.3.3 Further subdivisions for monotonicity

Configurations that satisfy constraints are of great importance for kinematic analysis, as they are the ones where the objects are in contact and can thus influence each other. For a correct qualitative analysis of the kinematics, it is necessary that the constraint segments be *monotone* so that the relation they enforce between the configuration space parameters of the objects can be stated as a qualitative equation.

For the purposes of discussion, we consider a constraint formed by a vertex on object O_v touching a boundary segment on object O_b, and refer to the position/orientation parameters of the objects as p_v and p_b. As the constraints are well-behaved algebraic curves with a continuous derivative, the monotonicity requirement will be satisfied whenever a constraint has no zero crossings of the derivatives dp_v/dp_b and dp_b/dp_v except possibly at the endpoints. It turns out that additional subdivisions of constraint segments are needed to satisfy this requirement. In order to find these, we examine where such zero-crossings of the derivatives may occur.

Zero crossings of dp_v/dp_b

A zero crossing of dp_v/dp_b occurs at an extremum of $p_v(p_b)$. If and only if p_{b0} is an extremum of this function, there are 2 points p_{b1} and p_{b2} in an infinitesimal neighborhood of p_{b0} such that $p_{b1} < p_{b0} < p_{b2}$ and $p_v(p_{b1}) = p_v(p_{b2})$. This means that there exist 2 configurations on the constraint where the O_v is in exactly the same position, but O_b is in an infinitesimally different one.

If O_b is rotationally attached, there may indeed exist such pairs, namely whenever p_{b1} and p_{b2} result in configurations where the point of touch lies at the identical radius r with respect to the attachment (see Figure 2.14). Such pairs of points with identical radii occur around extrema of the radius on the boundary, thus, zero-crossings of dp_v/dp_b will occur at these points.

But note that we have already broken up the boundary segments at the extrema of the radius in

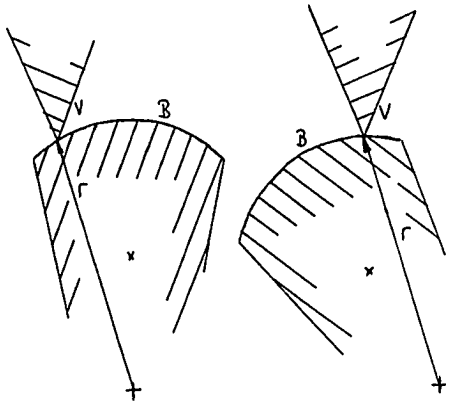

Figure 2.14: Pairs of configurations where the vertex stays in the same position occur at identical radii. Extrema of radii are thus extrema of the constraint.

order to achieve a unique parameterization. No further divisions are thus necessary to handle this case; this was in fact one of the reasons for the choice of parameterization.

If O_b is translationally attached, the sets of physical points occupied by the boundary segment are disjoint for any pair of distinct positions, thus no such pair is possible. An exception to this is the case where the boundary segment is parallel to the direction of translation: if there is a vertex touching this segment, the derivative of the constraint is identically zero.

Zero crossings of dp_b/dp_v - dead points

Similar to the preceding section, we again examine the conditions for pairs of configurations (p_{v1}, p_{b0}) and $(p_{v2}), p_{b0})$. The possible motion of O_v defines a *locus curve* for the vertex, which may be parameterized by p_v. The points on this curve corresponding to p_{b1} and p_{b2} must both touch the boundary on O_b. As the 2 points are only an infinitesimal distance apart, they lie just at those points where the locus curve touches the boundary. These points of touch correspond to the zero crossings of dp_b/dp_v, and we call them *dead points.*

If O_v is translationally attached, the locus is a straight line. If it touches a straight boundary segment along its length, the derivative of the constraint is zero throughout, and monotonicity is thus

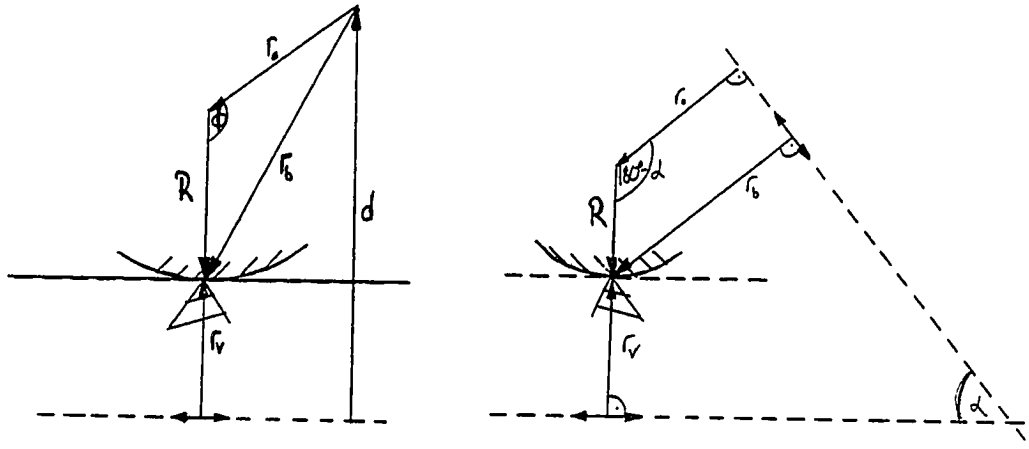

Figure 2.15: Situation analyzed for the zero crossings of dp_b/dp_v, O_v translating and O_b rotating (left) or translating (right).

satisfied. If the boundary segment is a curve, the vertex may touch it in a point as shown in Figure 2.15. This imposes a particular condition on the radius of the point of touch where the zero crossing occurs. Note that in Figure 2.15, the arc is convex. In the case of a concave arc, we have to reverse the appropriate signs and shall indicate the ambiguity in the computation. In the case where O_b is free to rotate, we have that

$$\begin{aligned} d &= r_v \pm (R - r_0 \cos\phi) sign(d) \\ \Rightarrow r_0 \cos\phi &= R \mp \frac{d - r_v}{sign(d)} \end{aligned} \tag{2.12}$$

and, by the law of cosines,

$$r_b^2 = R^2 + r_0^2 - 2Rr_0 \cos\phi \tag{2.13}$$

Using (2.12) in (2.13), we obtain

$$r_b = \sqrt{r_0^2 - R^2 \pm 2R(d - r_v) sign(d)} \tag{2.14}$$

Note that the cases of a convex and a concave arc are related by simple reversal of the sign of R. When O_b translates, we have from Figure 2.15 that

$$r_b = r_0 \pm R\cos\alpha \tag{2.15}$$

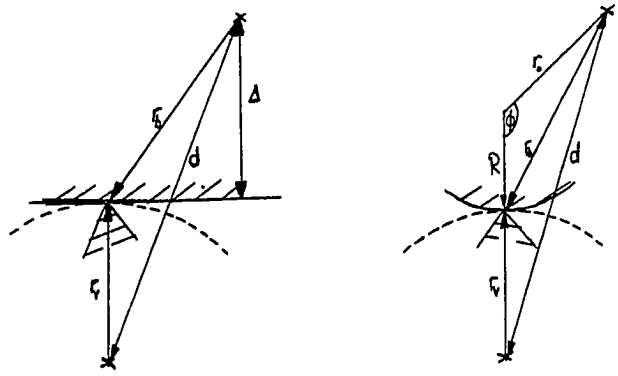

Figure 2.16: Situation analyzed for the zero crossings of dp_b/dp_v, O_v and O_b rotating.

If O_v is rotationally attached, the locus is a circle and can touch both straight boundary segments as well as arcs as shown in Figure 2.16 for a rotating O_b. We have to distinguish the cases where O_b rotates and where O_b translates.

In the case of rotation, we have that

$$d\cos\phi = r_v \pm \Delta \tag{2.16}$$

where the sign is ambiguous because the center of rotation could also be on the same side of the boundary as the other object. Also, by the law of cosines we have

$$r_b^2 = r_v^2 + d^2 - 2r_v d\cos\phi \tag{2.17}$$

Combining (2.16) and (2.17) gives

$$r_b = \sqrt{d^2 - r_v^2 \mp 2r_v\Delta} \tag{2.18}$$

for the desired radius value. The sign is most easily disambiguated by testing on the actual constraint. For the case of a convex arc, we have by the law of cosines that

$$r_b^2 = R^2 + r_0^2 - 2Rr_0\cos\phi \tag{2.19}$$

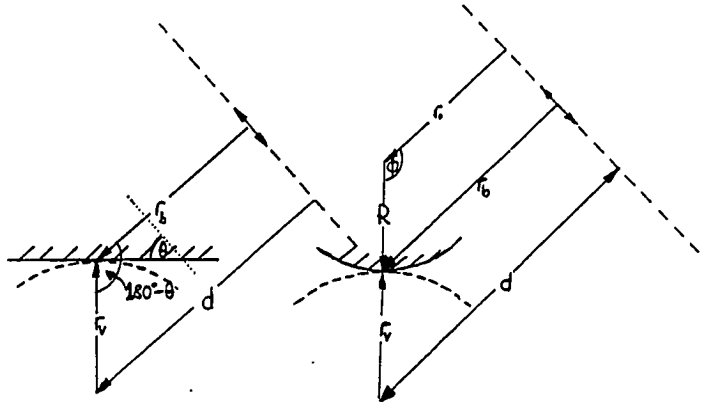

Figure 2.17: Situation analyzed for the zero crossings of dp_b/dp_v, O_v rotating, O_b translating.

and

$$d^2 = (R + r_v)^2 + r_0^2 - 2(R + r_v)r_0 \cos\phi \tag{2.20}$$

We can combine (2.19) and (2.20) and after some manipulation, we obtain

$$r_b = \sqrt{r_0^2 + R\left(\frac{d^2 - r_0^2}{r_v + R} - r_v\right)} \tag{2.21}$$

In the case of a concave arc, we can apply the same analysis, reversing the appropriate signs, to arrive at the result:

$$r_b = \sqrt{r_0^2 - R\left(\frac{d^2 - r_0^2}{r_v - R} - r_v\right)} \tag{2.22}$$

Note that (2.22) is just (2.21) with the sign of R reversed.

The situation when O_b translates is shown in Figure 2.17. For the case of a straight line boundary, we now have that

$$\begin{aligned} d &= r_b - r_v \cos(180 - \theta) \\ \Rightarrow r_b &= d - r_v \cos\theta \end{aligned} \tag{2.23}$$

where θ can be found from the object data in a straightforward manner. In the case of a convex arc

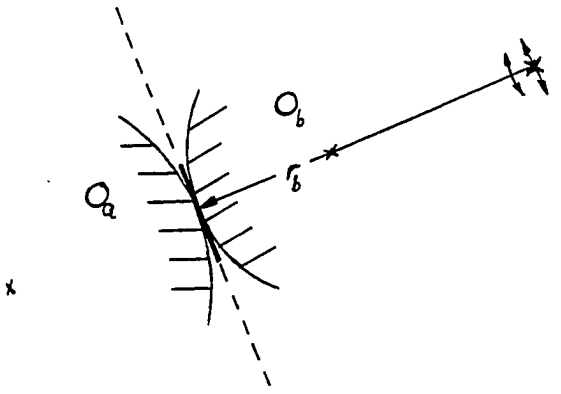

Figure 2.18: The extremum of p_a is reached where the point of touch on O_b is at an extremum of the radius.

shown on the right in Figure 2.17, we have by linearity

$$r_b = r_0 + \frac{R(d - r_0)}{R + r_v} \tag{2.24}$$

where the corresponding result for a concave arc is again obtained by reversing the sign of R.

Zero crossings in the case of boundary constraints

We consider a boundary constraint formed by 2 boundaries on O_a and O_b. Because of the symmetry of the situation, it is sufficient to examine the conditions for zero crossings of dp_a/dp_b. Consider the situation shown in Figure 2.18. Let p_{b0} be a configuration on the constraint such that $dp_a/dp_b = 0$. In order for the 2 boundaries to touch, they must be parallel at the point of touch; in fact, the 2 curves are identical in an infinitesimal neighborhood of the point of touch.

In order to allow small movements of O_b while keeping O_a in place, the boundary on O_a must touch but may not intersect the envelope of O_b formed by these movements. In the infinitesimal neighborhood of the point of touch, this envelope is given by the curve traced by that particular point under motion, i.e., an arc if O_b is rotationally attached and a straight line if the attachment is translational. The boundary on O_a must be parallel to this envelope at the point of touch, and

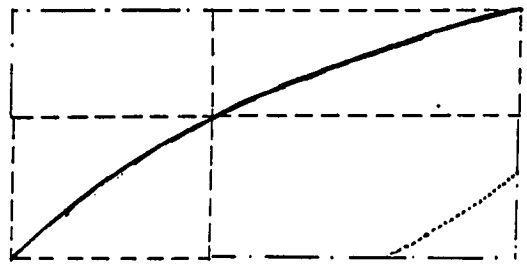

Figure 2.19: The constraint can be enclosed in the rectangle through the endpoints. Several rectangles might have to be made if the invalid part of the constraint curve passes through the original rectangle.

likewise for the boundary on O_b as the 2 boundaries touch. But this means that the boundary on O_b touches the envelope, and $dr/dp_b = 0$.

The extremal points of touch are thus extrema of the boundary curve, and the boundaries are already broken up at these points to give the unique parameterization. Thus, boundary constraint segments always fulfill the monotonicity requirement.

2.3.4 Bounding rectangles

We have already seen that the derivative of the constraint segments are qualitatively constant in the coordinate system defined by the c-space parameters of the 2 objects. This makes it possible to construct a *bounding rectangle* whose sides are parallel to the coordinate axes of configuration space such that the constraint segment must lie completely inside it, as shown in Figure 2.19. Note that this is the closest bound we can place on the shape of the constraint without further assumptions. The purpose of bounding rectangles is to delimit a region of configuration space such that the only part of the constraint curve that passes through it is the valid part. When the bounding rectangle is chosen simply as the rectangle through the endpoints, it is still possible that the invalid part of the constraint curve passes through it. This case can be detected by testing the opposite corners of the

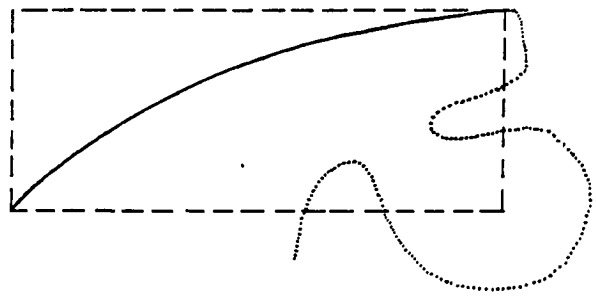

Figure 2.20: The constraint curve may intersect a bounding rectangle in complicated ways. Such cases are ruled out in our implementation by the fact that 2nd order curves can have only 2 intersections with a straight line.

enclosing rectangle for whether they lie on opposite sides of the constraint curve. We then replace the bounding rectangle by a pair of them, each enclosing one half of the valid constraint segment, until all such bounding rectangles contain only valid segments of the constraint curve.

The method above would fail in cases such as that shown in Figure 2.20, where the constraint curve intersects a side of the bounding rectangle three times. Note that in our case, the constraint curves have degree at most 2; therefore, they can have at most 2 intersections with the straight line bounding the rectangle. The invalid part of the constraint curve can thus intersect any side of any bounding rectangle at most once, and any such intersection will cause the 2 opposite corners of the bounding rectangle to lie on the same side of the constraint curve.

2.3.5 Endpoints of constraint segments

If the freedom of the objects is completely unconstrained, the configurations defined by the touchpoints are always achievable and the touch "point" are not points, but form a single connected region in configuration space. When the freedom of the objects is constrained, however, this is not always the case. Constraints or parts of them are nonexistent if the corresponding points of touch are not achievable.

For an object with one degree of freedom, the location of a given point on the object is constrained

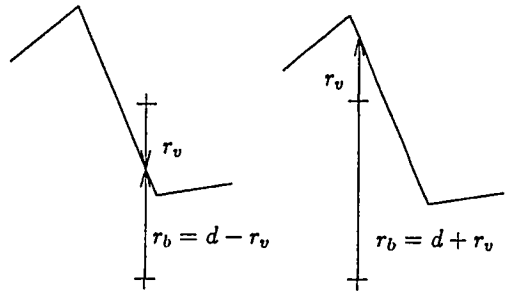

Figure 2.21: Extreme configurations and the radius values that result from them.

to its *locus curve*. If the freedom is rotational, this curve is a circle; if it is translational, a straight line. A given pair of points on the 2 objects can touch if and only if their locus curves intersect, and the location of the point of intersection defines the configuration where this touch occurs. Note that in the case of at least one object having rotational attachment, there exist two such points of intersection, defining the original and the mirror image of the touchpoint.

When both objects are free to translate, either they can move in parallel (a singular case) or each pair of vertices forms exactly one touchpoint, as 2 straight lines are either parallel or have exactly one point of intersection. When one or both of them are constrained rotationally, there are non-singular cases where no intersection and thus no touchpoint of the pair of vertices exists.

A constraint C exists whenever at least one configuration on it exists. In the case where there is such a point but both endpoints of the constraint are inactive, the actual endpoints of the active part of the constraint lie at extremal points of touch. In the following, consider a vertex constraint generated by a vertex V with radius r_v touching a boundary segment B at radius r_b.

If the vertex is attached rotationally, there always exists a minimum and a maximum of the radius of the point of touch, corresponding to the 2 configurations where the radius from the center of rotation to the vertex is parallel to the direction between the attachments (see Figure 2.21). These radii are thus given as $r_{b1} = d + r_v$ and $r_{b2} = d - r_v$. If the boundary is rotationally attached, the sign is of no importance and we have that $r_{b2} = |d - r_v|$.

If the vertex has translational and the boundary rotational attachment, there is only one possible extremum of the point of touch. This is reached when $r_b = |d - r_v|$, corresponding to the configuration on the left in Figure 2.21.

If the radius of a vertex at the end of the boundary segment B falls outside of the interval defined by these extremal values, the particular endpoint does not exist. In the case where the interval of radii between the 2 endpoints does *not* contain the interval of the extremal values, the constraint does not exist at all. Otherwise, the actual endpoints can be *inferred* by replacing the values at those endpoints that fall outside of the interval with the corresponding bound.

Note that at such inferred endpoints, the constraint does not intersect the one generated by the boundary adjacent to B at the vertex in question. Also, the extremal configurations are identical for the constraint and its mirror image. The constraint thus joins smoothly with its mirror image at inferred endpoints. No applicability constraints need apply in connection with this smooth join.

The same reasoning as in the case of vertex constraints also applies to boundary constraints. In the case of boundary constraints generated by the touch of an arc, the constraints are directly analogous to vertex constraints and the analysis thus can be carried over directly. In other cases, a more complicated analysis may be necessary.

2.3.6 Endpoints of boundary constraints

Boundary constraints in general do not end at touchpoints, but connect smoothly with a vertex constraint. For each boundary constraint, there are 4 possible candidates for these constraints as shown in Figure 2.22 below. For each of these candidates, we further have to distinguish whether the constraint has to be linked to the original or the mirror image version, and to which segment if the constraint is broken up.

First, let us consider the different conditions for an endpoint of a boundary constraint. An *obvious endpoint* of a boundary constraint is reached when the center of the participating arc reaches an endpoint of the imaginary boundary it travels on. In this case, the constraint connects with the vertex constraints a) and b) in Figure 2.22. However, it is possible that in this configuration, the point of touch would not fall within the valid segment of the arc. In this case, the endpoint of the constraint is determined by the endpoint of the arc, and connects with the vertex constraint c) or d) in Figure 2.22. We call this case an *inferred endpoint.*

While we otherwise parameterize arc boundary segments by their radius, in this case it is more convenient to use the angle, ψ. In order to determine the location of the point of touch on the arc, and

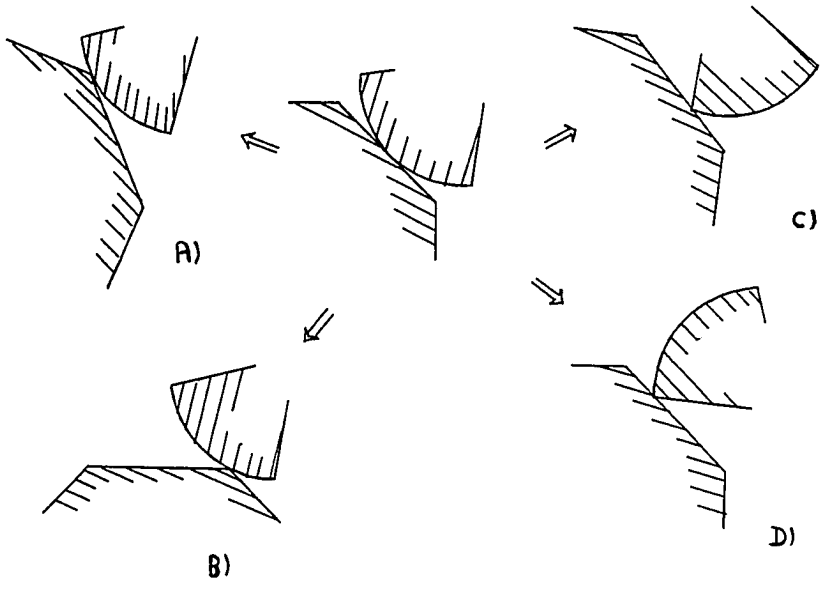

Figure 2.22: The 4 possible vertex constraints to end a boundary constraint.

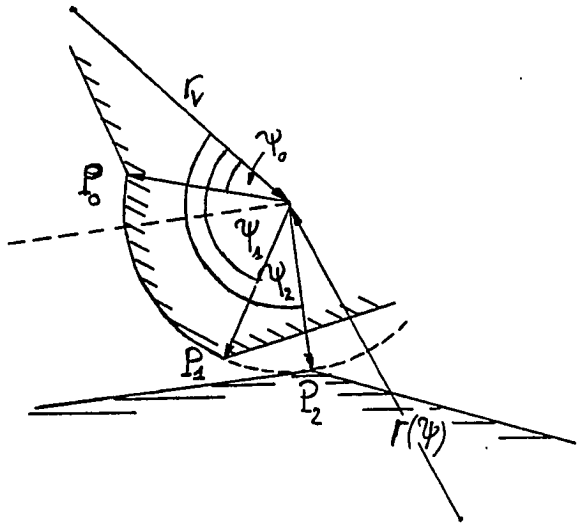

Figure 2.23: Determining the endpoints of boundary constraints.

the parameter value to be used for the endpoint when the constraint is bounded by an endpoint of the arc, we need to examine the relationship between the constraint parameter r and the angle ψ of the point of touch on the arc. Given the relationships $r(\psi)$ and $\psi(r)$, we can test the obvious endpoints for legality by comparing ψ against the endpoints of the arc. If they are found to be illegal, the parameter value of the endpoint is given by $r(\psi)$ and ψ of the endpoints of the arc. This is illustrated in Figure 2.23.

At its endpoints, the boundary constraint connects smoothly with the adjoining vertex constraint. The parameter value of the vertex constraint corresponding to this endpoint is determined by the parameter value of the boundary constraint. Therefore, we also have to find the functions relating the parameters of the 2 constraints.

The final problem is to determine whether to use the original or the mirror image version of the vertex constraint, and which segment should be used if the constraint is broken up. Fortunately, as this is a problem of deciding among a small number of alternatives, we can avoid lengthy computation by using a generate-and-test method. For each alternative, we find the configuration corresponding to the parameter value we computed for the endpoint and compare this configuration to the actual one of the endpoint of the boundary constraint. As these configurations are necessarily distinct for each alternative, the test always determines a unique adjoining constraint.

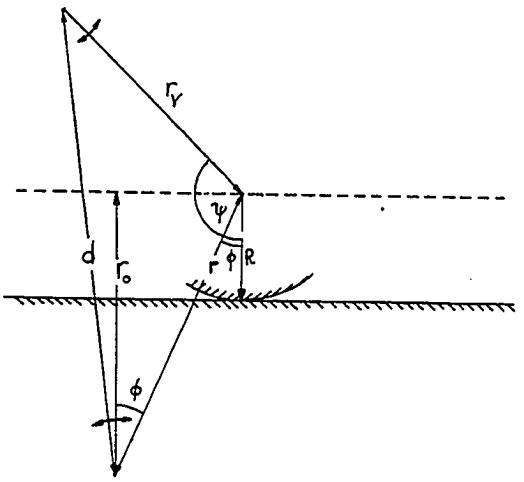

Figure 2.24: Line boundary constraint with both objects rotationally attached.

The algebraic analyses referred to above are given in the following sections.

2.3.7 Determining $r(\psi)$ and $\psi(r)$ - case of line

The relationships between r and ψ depend on the attachments of the objects involved. We therefore analyze each combination of attachments separately.

Both objects rotationally attached

In Figure 2.24, we have by the law of cosines

$$d^2 = r^2 + r_v^2 - 2rr_v \cos(\phi + \psi)$$

Note further that

$$\cos\phi = r_0/r, \text{ and } \sin\phi = \pm\sqrt{1 - r_0^2/r^2}$$

where $\sin\phi$ is negative for a descending segment, positive for an ascending segment. Using the addition formulae for sine and cosine, and the formula for the solution of a quadratic equation, we then obtain

$$r_{12}^2 = d^2 - r_v^2 + 2r_0 r_v \cos\psi + 2r_v^2 \sin^2\psi \pm 2r_v \sin\psi\sqrt{d^2 - (r_0 - r_v\cos\psi)^2}$$

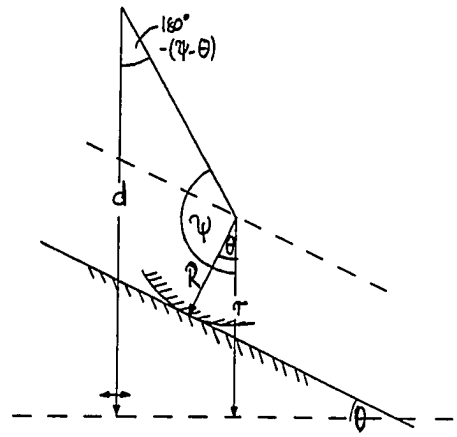

Figure 2.25: Line boundary constraint with boundary on translating object, vertex on rotating object.

where the sign is positive for a descending segment, negative for an ascending segment.

When r is given, we have that

$$\cos(\psi - \phi) = \frac{r^2 + r_v^2 - d^2}{2rr_v}, \text{ and } \sin(\psi - \phi) = \pm\sqrt{1 - \cos^2(\phi + \psi)}$$

where the sign of the sine is positive if the constraint is a mirror image. The angle ψ can be found from this by application of the addition formulae.

Boundary on translating object, vertex on rotating object

In Figure 2.25, we have for original constraints that

$$r = d - r_v \cos(\psi - \theta - 180)$$

and for mirror images that

$$r = d - r_v \cos(180 - (\psi - \theta))$$

By the symmetries of the cosine function, these both reduce to

$$r = d + r_v \cos(\theta - \psi)$$

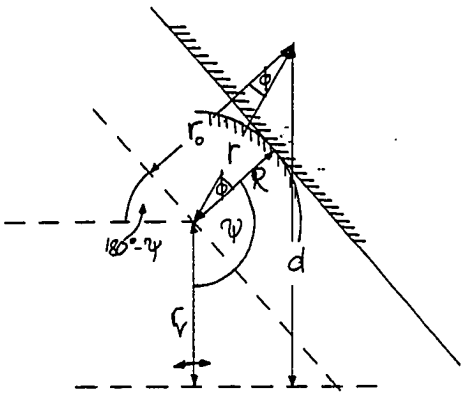

Figure 2.26: Line boundary constraint with boundary on rotating object, vertex on translating object.

As θ is fixed for the boundary segment, r is found from this by application of the summation formulae. Note that the equations also hold if the center of rotational attachment is on the negative side of the translational attachment, i.e., d is negative. When we invert the equation to find ψ, however, we must invert the signs when r becomes negative, thus obtaining that

$$\cos(\theta - \psi) = sign(r)\frac{r-d}{r_v}, \text{ and } \sin(\theta - \psi) = \pm\sqrt{1 - \cos^2(\theta + \psi)}$$

where the sine is negative for an original, and positive for a mirror image. We can again find ψ from this using the summation formulae.

Boundary on rotating object, vertex on translating object

Referring to Figure 2.26, we have that

$$r^2 = r_0^2 + \Delta^2$$

and

$$\Delta = \frac{(d - r_v)sign(r_v) + r_0 \cos\psi}{\sin(-\psi)}$$

We have to take into account the additional condition that

$$\sin(\phi - \psi) = \begin{cases} < 0 & \text{for mirror images} \\ > 0 & \text{for originals} \end{cases}$$

If this condition is violated, the boundaries do not properly touch each other in the corresponding configuration, and the constraint does not exist. The condition can be tested for using

$$\tan\phi = |\Delta|/r_0,\ \cos\phi = \sqrt{\frac{1}{1+\tan^2\phi}},\ \sin\phi = \pm\sqrt{1-\cos^2\phi}$$

where the sign of the sine is positive for an ascending segment, negative for a descending segment.

To determine ψ as a function of r, note that

$$\cos(\phi - \psi) = sign(r_v)\frac{r_v - d}{r},\ \sin(\phi - \psi) = \pm\sqrt{1-\cos^2(\phi-\psi)}$$

where the sine is positive for a constraint original, negative for a mirror image. The angle ϕ can be found in the usual manner using $\cos\phi = r_0/r$, and ψ thus computed using the summation formulae.

Both objects translationally attached

In this case, the relative angle between the 2 objects is constant, and thus ψ does not vary along the constraint. As shown in Figure 2.27, ψ is given as

$$\psi = (\alpha - \theta) \tag{2.25}$$

Note that the test for whether ψ falls within the boundary segment now is a test for the *existence* of the boundary constraint.

2.3.8 Determining $r(\psi)$ and $\psi(r)$ - case of arc

Again, as the relationships depend very heavily on the attachments of the objects, we make a case split and analyze each case separately.

Both objects rotationally attached

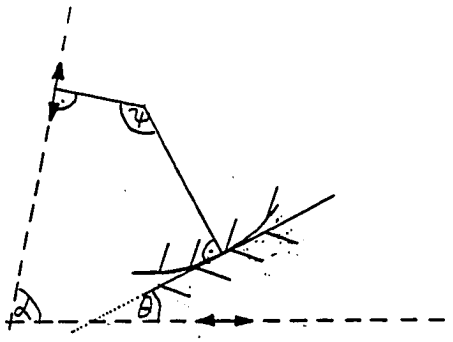

Figure 2.27: Line boundary constraint - both objects translating.

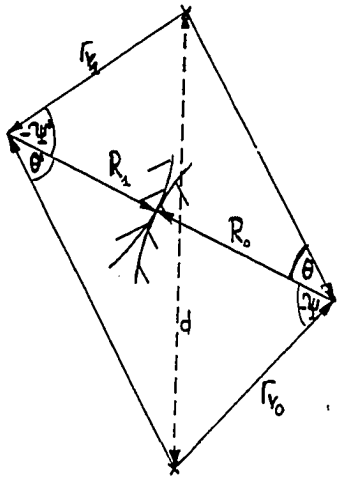

Figure 2.28: Arc boundary constraint, both objects rotating.

The situation corresponding to this case is shown in Figure 2.28, where the imaginary vertex is assumed to lie on object 1, and the boundary on object 0. By the law of cosines, we have the following relations:

$$r^2 = r_{v_1}^2 + (R_0 + R_1)^2 - 2r_{v_1}(R_0 + R_1)\cos\psi' \tag{2.26}$$

$$d^2 = r_0^2 + r_{v_1}^2 - 2r_0 r_{v_1}\cos(\theta' - \psi')$$

$$\Rightarrow \cos(\theta' - \psi') = \frac{r_0^2 + r_{v_1}^2 - d^2}{2r_0 r_{v_1}} \tag{2.27}$$

$$r_{v_0}^2 = r_0^2 + (R_0 + R_1)^2 - 2r_0(R_0 + R_1)\cos\theta'$$

$$\Rightarrow \cos\theta' = \frac{r_0^2 + (R_0 + R_1)^2 - r_{v_0}}{2r_0(R_0 + R_1)} \tag{2.28}$$

By the law of sines, we also have

$$\frac{r_{v_0}}{\sin\theta'} = \frac{r_0}{\sin(-\psi)} \Rightarrow \sin\theta' = \frac{r_{v_0}}{r_0}\sin\psi \tag{2.29}$$

Further observing that

$$\sin(\theta' - \psi') = \pm\sqrt{1 - \cos^2(\theta' - \psi')}$$

where the sign is positive for originals, negative for mirror images, we can find $\cos\psi'$ from (2.27), (2.28) and (2.29) using the summation formulae. Using this value in (2.26) gives the result

$$r^2 = r_{v_1}^2 + r_{v_0}(R_0 + R_1)\cos\psi - \frac{R_0 + R_1}{r_0^2}\left[(r_{v_1}^2 - d^2)(R_0 + R_1 - r_{v_0}\cos\psi)\right.$$
$$\left.\pm r_{v_0}\sin\psi\sqrt{2r_0^2 r_{v_1}^2 + 2r_0^2 d^2 + 2r_{v_1}^2 d^2 - r_0^4 - r_{v_1}^2 - d^4}\right] \tag{2.30}$$

where the sign can not be predicted, but must be found by testing consistency.

We can find ψ from the formula for ψ' by symmetry of the figure as

$$\cos\psi = \frac{(r^2 + r_{v_0}^2 - d^2)(R_0 + R_1 - r_{v_1}\cos\psi') \pm r_{v_1}\sin\psi'\sqrt{2r^2 r_{v_0}^2 + 2r^2 d^2 + 2r_{v_1}^2 d^2 - r_0^4 - r_{v_1}^4 - d^4}}{2r^2 r_{v_0}} \tag{2.31}$$

where the sign is positive for mirror images, negative for originals, and $\sin\psi = \pm\sqrt{1 - \cos^2\psi}$ has a negative sign for originals and a positive sign for mirror images.

One object translating, one object rotating

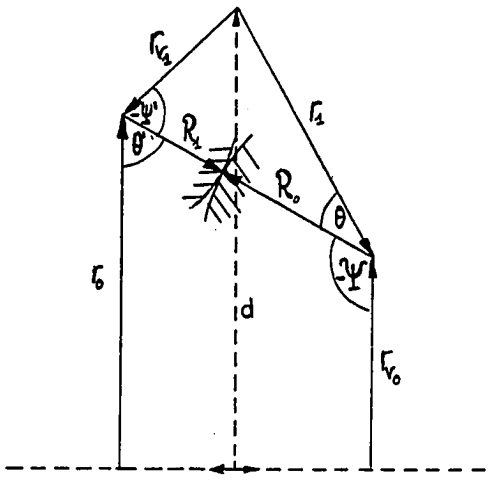

Figure 2.29: Arc boundary constraint. Object 0 translates, object 1 rotates.

Because of the symmetry of the situation, the cases where the imaginary vertex is on the rotating object and the one where it is on the translating object may be analyzed together. Consider the situation shown in Figure 2.29. First, observe that

$$\theta' = 180 + \psi \tag{2.32}$$

and then that

$$\begin{aligned} r_0 &= d + sign(d) r_{v_1} \cos(\theta' - \psi') = d + sign(d) r_{v_1} \cos(\psi + 180 - \psi') \\ r_0 &= r_{v_0} + (R_0 + R_1)\cos\theta' = r_{v_0} - (R_0 + R_1)\cos\psi \qquad (2.33) \\ &\Rightarrow \cos(\psi' - \psi) = \frac{r_{v_0} - (R_0 + R_1) sign(r_{v_0}) \cos\psi - d}{sign(d) r_{v_1}} \qquad (2.34) \end{aligned}$$

Note that since $\theta' = 180 + \psi$, the sign of $\sin\psi'$ is positive for the mirror image and negative for the original; $\sin\psi'$ can thus be found from $\sin(\psi' - \psi) = \pm\sqrt{1 - \cos^2(\psi' - \psi)}$. This allows us to express ψ' in terms of ψ by using the summation formulae. Note that $\psi(r_0)$ can be found from Equation 2.33, noting that the sign of $\sin\psi$ is positive for mirror images, negative for originals. For r_1, we have by the law of cosines

$$r_1 = \sqrt{r_{v_1}^2 + (R_0 + R_1)^2 - 2 r_{v_1} \cos\psi'} \tag{2.35}$$

By using the previous result in this equation, we can obtain $r_1(\psi)$. We also have for r_0:

$$\begin{aligned} r_0 &= d - r_{v_1}\cos(180 - \psi' - \theta')sign(d) \\ &= d + r_{v_1}\cos(\psi' + \theta')sign(d) \qquad (2.36) \\ r_0 &= r_{v_0} + (R_0R_1)\cos\theta' \Rightarrow \cos\theta' = \frac{r_0 - r_{v_0}}{R_0 + R_1} \qquad (2.37) \end{aligned}$$

The term $\cos(\psi' + \theta')$ in Equation (2.37) can be found from Equation (2.37) (noting that $\sin\theta'$ is positive for originals, negative for mirror images) and ψ' by application of the summation formulae. This gives a quadratic equation which can be solved for r_0 to give

$$\begin{aligned} r_{0_{12}} = & \frac{r_{v_0} + d(R_0 + R_1)^2/r_{v_1}^2 - \cos\psi'(d + r_{v_0})sign(d)(R_0 + R_1)/r_{v_1}}{N} \\ & \pm(R_0 + R_1)\sin\psi'\sqrt{1/N - \frac{(d - r_{v_0})^2}{r_{v_1}^2N^2}} \qquad (2.38) \end{aligned}$$

where

$$N = 1 + (R_0 + R_1)^2/r_{v_1}^2 - 2sign(d)\cos\psi'(R_0 + R_1)/r_{v_1}$$

and the sign can not be predicted, but must be found by consistency check. We have already seen how to compute ψ' from ψ. It remains to show how to derive ψ from ψ'. Note that we have that

$$\begin{aligned} d &= r_{v_0} + r_1\cos(180 - \theta - \psi) \\ &= r_{v_0} - r_1\cos(\theta + \psi) \\ \Rightarrow \cos(\theta + \psi) &= \frac{r_{v_0} - d}{r_1} \qquad (2.39) \end{aligned}$$

and that $\sin(\theta + \psi)$ is positive for mirror images, negative for originals. We also have by the law of cosines that

$$\begin{aligned} r_{v_1}^2 &= r_1^2 + (R_0 + R_1)^2 - 2r_1(R_0 + R_1)\cos\theta \\ \Rightarrow \cos\theta &= \frac{r_1^2 + (R_0 + R_1)^2 - r_{v_1}^2}{2r_1(R_0 + R_1)} \qquad (2.40) \end{aligned}$$

where $\sin\theta$ is positive for mirror images, negative for originals. We can then obtain ψ from (2.39) and (2.40) using the summation formulae.

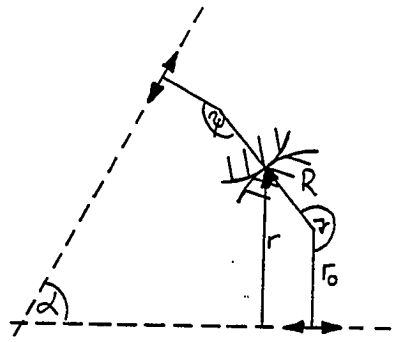

Figure 2.30: Arc boundary constraint - both objects translating.

Both objects translating

The situation for this case is shown in Figure 2.30. Note that we have for ψ

$$\psi = \alpha - \gamma \tag{2.41}$$

and for r,

$$r = r_0 + R\cos(\gamma) = r_0 + R\cos(\alpha - \psi) \tag{2.42}$$

The angle γ can be determined directly from the boundary segment.

2.3.9 Relations between constraint parameterizations

At the endpoints of the boundary constraints, the value of the radius value for the vertex constraint corresponding to that configuration has to be found from the radius value of the boundary constraint. We have to distinguish the following 2 cases:

- The vertex of the vertex constraint is on the other object
- The vertex of the vertex constraint is on the same object as the arc of the boundary constraint

In the first case, the radius values refer to boundaries on different objects. The parameter value on the vertex constraint is in this case best computed using the relations to the angle ψ found earlier which can serve to index to point of touch. Making the appropriate distinctions with respect to the

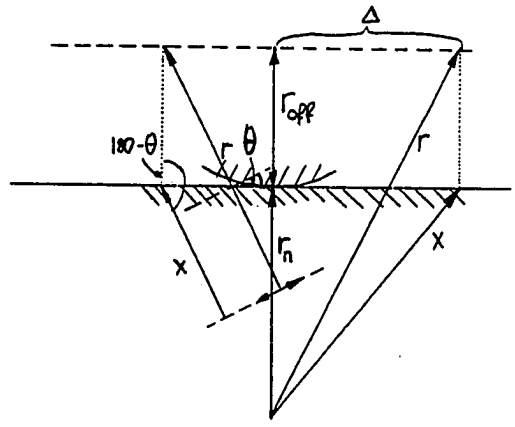

Figure 2.31: An straight line boundary segment "expanded" to derive a boundary constraint.

attachments, we have here that

$$r = \begin{cases} \sqrt{r_0^2 + R^2 - 2r_0 R\cos\psi} & \text{for rotational attachment} \\ r_0 - Rsign(r_0)\cos\psi & \text{for translational attachment} \end{cases}$$

In the second case, the radius value of the vertex constraint and that of the boundary constraint refer to the same object boundary. They can thus be computed without any further reference to the other object. In the case of a boundary constraint generated by an arc and a straight line, we have the situation shown in Figure 2.31. In the case of rotational attachment, indicated on the right half of Figure 2.31, we have the 2 equations

$$x^2 = r_n^2 + \Delta^2 \tag{2.43}$$

$$r^2 = (r_n + r_{off})^2 + \Delta^2 \tag{2.44}$$

Combining these 2 equations, we have that

$$x^2 = r_n^2 + r^2 - (r_n + r_{off})^2 \tag{2.45}$$

For translational attachment, shown in the left half of Figure 2.31, we have by linearity that

$$x = r - r_{off}\cos\theta \tag{2.46}$$

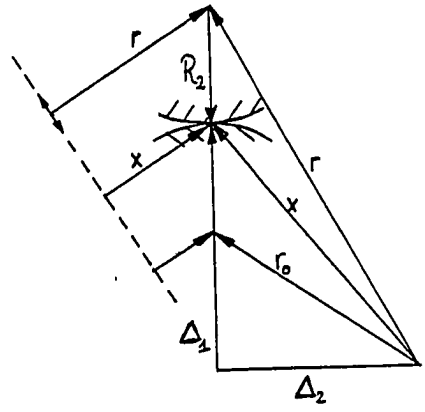

Figure 2.32: An arc boundary segment "expanded" to derive a boundary constraint.

where θ is found in a straightforward manner from the metric diagram.

In the case of 2 rotating arcs, shown on the right of Figure 2.32, we have that

$$r_0^2 = \Delta_1^2 + \Delta_2^2 \tag{2.47}$$

$$x^2 = (\Delta_1 + R_1)^2 + \Delta_2^2 \tag{2.48}$$

$$r^2 = (\Delta_1 + R_1 \pm R_2)^2 + \Delta_2^2 \tag{2.49}$$

where the ambiguity of the sign in Equation (2.49) expresses the different cases of convex/concave arcs. These equations can be combined to give

$$x = \sqrt{r_0^2 + R_1^2 + \frac{R_1}{R_1 \pm R_2}(r^2 - R_1^2 - R_2^2 - r_0^2 \mp 2R_1R_2)} \tag{2.50}$$

In the case of translational attachment, shown on the left half in Figure 2.32, we have by linearity that

$$x = r_0 + \frac{R_1}{R_1 \pm R_2}(r - r_0) \tag{2.51}$$

where the ambiguous sign again expresses the distinction of convex/concave arcs.

2.3.10 Singular cases

There are 2 kinds of singularities that we need to distinguish in this context. The first are singular cases of object boundaries which can not be parameterized in the parameterization we are using: these are straight lines parallel to a translational attachment, and arcs whose center coincides with the center of a rotational attachment. For these curves, the radius is constant and thus can not be used as a parameter. Instead, we use the distance from the lowest endpoint. We have also seen previously that these cases result in constraints whose derivative with respect to the configuration space parameter of the boundary segment is identically zero. Besides this feature and the different parameterization, these constraints share the same properties as the other constraints.

When 2 singular boundary segments of this kind are combined, the resulting constraint may be singular in a different way. There exist in particular the following cases:

1. Two parallel straight line boundary segments.
2. A straight line, translating parallel to itself, and an arc, with its center of curvature in the center of rotation of the object.
3. Two circular boundary segments, rotationally attached, both having their center of curvature in the center of rotation of the respective objects.
4. A straight line boundary segment and a vertex, both with translational freedom parallel to the boundary segment.

Case (1) is the only case where 2 straight lines can touch. In the case of attached objects, it in general results in a one-dimensional constraint. However, there is the singular case where both objects have translational freedom parallel to the boundary segments. In this case, the constraint is actually 2-dimensional, as both objects can move while maintaining the constraint. The same situation is given in case (4).

Similarly, in cases (2) and (3), both objects can move while maintaining the constraint, and it is thus a 2-dimensional region of configuration space. Fortunately, the boundaries of this region are rather simple: it is just a rectangle in configuration space through the touchpoints formed by the vertices at the ends of the boundary segments in question.

The 2-dimensional constraints do not serve any kinematic function in mechanisms, as the contact

is trivially maintained no matter how the parts move, and they do not constrain this motion at all. They are therefore ignored for the construction of the place vocabulary; if a particular application is sensitive to such points of contact, e.g., electrical machinery, the corresponding regions can be specified and intersections with the places of the place vocabulary could be formed in a postprocessing phase.

Chapter 3

PLACES

In this chapter, we describe how the constraint segments can be combined to form the place vocabulary. The computation of place vocabularies proceeds in 2 stages: First, place vocabularies are found for the individual kinematic pairs, and then the place vocabulary for the complete kinematic chain is found by composition of these pairwise place vocabularies. We discuss first the computation of the place vocabularies for kinematic pairs.

The place vocabulary is a representation of free space as a *cell complex*, i.e., a set of cells of different dimensions. In configurations in the regions of free space where no contact between the objects exists, the objects do not constrain each other; these are the *full-dimensional* places. In general, every contact between the objects reduces the degrees of freedom by one. The places of dimension d-1 (with d being the dimension of the configuration space) are thus made up by the configurations that exactly satisfy one constraint, the constraint curves themselves. The places of dimension d-n comprise the configurations that satisfy n constraints simultaneously, i.e., the intersection manifolds of n constraints. Except in singular cases, n points of contact exist between the objects in these places.

Only regions which lie within or on the boundary of free space can form places. The first step in finding the place vocabulary thus consists of determining the connected components of free space. As we already noted, the boundary between free and blocked space is made up of the envelope of the constraints. In the case of a 2-dimensional configuration space, these boundaries are 1-dimensional *chains* of constraint segments. We define a *chain* as a maximal sequence of constraint segments such

that

- successive segments intersect each other, and have the free space on the same side.
- no segment intersects anything else between the intersections with its 2 neighbors in the sequence
- no segment is subsumed by another intersecting constraint within the interval where it forms part of the chain. This still leaves the possibility of chains subsuming each other.

The chains can be found by tracing the structure formed by the constraint segments and their intersections. In the general case of a configuration space with more than 2 dimensions, this envelope will itself be a cell complex and can not be characterized in this simple manner. Our subsequent discussion is therefore restricted to the case of 2-dimensional configuration spaces.

The constraint segments that occur as part of some legal chain make up the 1-dimensional places in the place vocabulary, and their intersections are the 0-dimensional places. No other 0- or 1-dimensional places can exist, as they would form additional boundaries of free space.

Given the chains, it is possible to find the connected components of free space and associate the chains into sets bounding the same component. Note that there may be an arbitrary number of such chains associated with the same region. First, if one or both of the attachments are rotational, the topology of the configuration space is that of a cylinder or a torus, where 2 disjoint boundaries are required to bound a single connected region. Second, the components need not all be simply connected. Within any component of free space, there may exist an arbitrary number of *obstacles*, each of which has a chain as boundary.

Finally, each connected component must be described by a set of quasi-convex cells, i.e., a conjunction of predicates ($\mathcal{F}_i > / < 0$), where the $\mathcal{F}_i$ are equations of constraints or applicability constraints. To find this description, we must take into account not only the set of constraint segments in the chains, but also their applicability constraints. The applicability constraints require place divisions wherever they run through free space. As free space is defined by a logical formula on the signs of the constraints and the applicability constraints, and the signs of these do not change within the regions thus defined, they constitute the desired breakup of the connected component into full-dimensional, quasi-convex places.

In the following sections, we discuss the steps involved in computing the place vocabulary from the constraints. First, we discuss the problem of determining the envelopes of the constraints, then

we show how they can be associated with the different connected components, and finally how these components are broken up by the applicability constraints. We then discuss what further subdivisions might be appropriate to improve the results of the qualitative analysis, and how to compose the place vocabularies for analyses of kinematic chains. Finally, we show how algebraic decision methods could be applied as an alternative way to compute place vocabularies.

3.1 Determining Chains of Constraints

The first step in finding the chains of constraint segments consists of detecting all intersections between constraints. Some intersections are already given by the endpoints of the constraints; we call these *endpoint intersections*. Besides these, any pair of constraints can form intersections distinct from their endpoints. We call these intersections *subsumption intersections*, as at these points one constraint is subsumed by another.

The number of possible intersections between a pair of constraints is limited by the algebraic degree of the constraints involved. In our particular case of boundary curves restricted to arcs and straight lines, we have seen in the previous chapter that the algebraic degree of the constraint curves is never greater than 2, thus limiting the number of possible intersections to 2 also.

The only simple way to test for intersections between constraint segments is to test the relative locations of their endpoints. However, this test will only succeed when the number of intersections is odd. An even number of intersections between constraints reflects a local subsumption between constraints. It has no global influence on the place vocabulary, and such cases can thus be viewed in isolation from the rest of the place vocabulary computation. There is no simple way to detect such multiple intersections. However, in the section on algebraic decision methods, we show how the number of intersections between algebraic curves can be reliably computed.

The intersections between constraints define a graph $\mathcal{G}$. The set of vertices of $\mathcal{G}$ is the set of intersections of constraints, and the edges are the valid constraint segments between the intersections. The ordered set of intersection points on some constraint C defines a set of valid constraint segments in the intervals on C that do not violate any of the intersecting constraints. Note that each intersection point is linked to at most 2 such valid segments. Note now that each loop in $\mathcal{G}$ is a legal chain, and

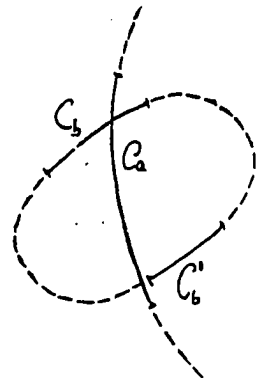

Figure 3.1: The endpoints of constraint C_a lie on the same side of C_b, but there is only one intersection where C_b is valid.

that any legal chain must form a loop in $\mathcal{G}$. The chains can thus be found by tracing $\mathcal{G}$ to find all the loops. Chains found in this way could still lie completely within blocked space. Because the way we have defined constraints does not give a full description of blocked space, we can not test the chain directly for inclusion in blocked space. However, if the chain is a part of blocked space, there must exist a subsuming chain. We describe later how we can test all chains to see if they subsume another.

3.1.1 Finding single intersections

The set of candidates for pairs of intersecting constraints can be limited by considering only pairs for which there is a pair of intersecting bounding rectangles. Usually, a pair of constraints that is tested for intersection does not cover exactly the same range of parameters. It is possible that a constraint segment intersects the algebraic curve of another constraint segment 2 times, but that only one of the 2 intersections falls within the area where the other constraint is actually valid (see Figure 3.1). To avoid this case, we construct the intersection of the bounding rectangles of the 2 constraints, then find the endpoints of those segments of the constraints that fall within the intersection(s), and test these for intersection. As neither bounding rectangle contains any invalid part of the constraint curves, no intersections with them can occur within the intersecting region.

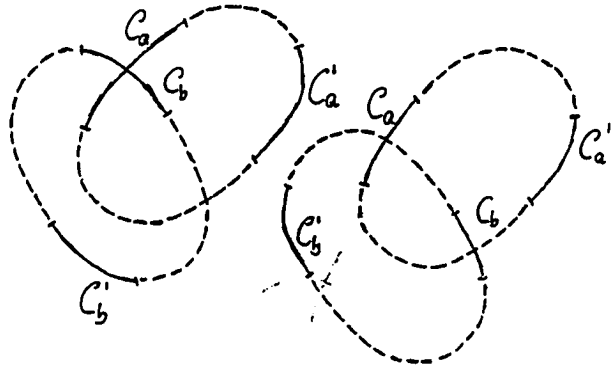

Figure 3.2: Proper (left) and improper (right) intersection of C_a and C_b.

The condition of intersection of enclosing rectangles also avoids another undesirable case. Consider 2 constraint segments C_a and C_b which are to be tested for intersection. Each segment forms part of a closed constraint curve, and we call the rest of this closed curve the *complement* of the constraint segment. If, by evaluating the equation of the constraint, the endpoints of C_a have been found to lie on opposite sides of C_b, and the same holds for C_b, then we have one of the 2 cases (shown in Fig. 3.2):

- C_a and C_b intersect, and so do their complements
- C_a intersects the complement of C_b, and C_b the complement of C_a.

Intersections of the first kind we call *proper* intersections, intersections of the second kind *improper*. Only proper intersections are of any interest for the place vocabulary. As the bounding rectangles do not enclose any invalid part of the constraint curve, the condition that they intersect rules out improper intersections.

For constraint segments which share a common endpoint, the endpoint tests are not possible, as the common endpoint lies *on* both constraints and thus can not be on the opposite side of any other endpoint. For a pair of constraints C_a and C_b with a common endpoint, where C_a subsumes C_b at the common endpoint, a subsumption intersection exists whenever the other endpoint of C_a is subsumed by C_b. The condition of intersection of bounding rectangles applies to this case as well.

3.1.2 Ordering the intersections

For each constraint, the intersections with other constraints must be ordered so that the valid segments can be found. While this ordering amounts to tesselating a surface in the case of constraint curves with more than 2 parameters, in the special case of a 2-dimensional configuration space and parameterized constraints, the intersection points can be ordered on the constraint curve in the direction of its tangent. In order to construct this ordering, we need a test to compare the parameter values of the intersection points.

Such a test can readily be devised for the case where the 2 intersecting constraints themselves do not form any subsumption intersections. If they do not intersect at all, one completely subsumes the other and their ordering can be found by testing which side both endpoints lie on. If they share a common touchpoint, then their ordering is determined by the local analysis at the touchpoint.

In the case where 2 constraints C_a and C_b intersecting a third C_c have a free intersection, it must be determined on which side of C_c that intersection lies in order to determine the ordering of the constraints. But as the coordinates of the intersection point are not known at this point, this can not be tested directly. In this case, we determine the order by finding a point on C_c *between* the intersection points of C_a and C_b with C_c and determining on which side of C_a and C_b this point lies.

3.1.3 Detecting multiple intersections

The case of an even number of intersections between a pair of constraint segments can not be detected by the endpoint test. Multiple intersections must be found by algebraic methods based on Sturm's theorem, these are described in a later section. This leaves the algorithm incomplete for the case where the algebraic form of the boundary segments is not known, but note that in this case the possibility of error is indicated by the intersection of the bounding rectangles. We have not implemented these algorithms as we have not come across a case where multiple intersections actually occurred.

3.2 Association of Boundary Sequences

Once the chains have been determined, it is necessary to find the connected regions of configuration space that they enclose. First, we have to consider the topology of configuration space. Depending

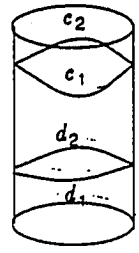

Figure 3.3: The chains $c_1 - c_2$ and $d_1 - d_2$ together bound a single region of space.

on the combination of attachments, we have the following cases:

translational-translational: The configuration space is the Euclidian plane.

translational-rotational: The configuration space is a cylinder

rotational-rotational: The configuration space is a torus

A region may have several disjoint boundaries, depending on whether it is simply or multiply connected. In the case where the configuration space is a cylinder or a torus, there are boundaries which do not delimit a region by themselves, but require a complementary boundary to form one. This is the case whenever chains are "slung around" one or both rotational dimensions, as shown in Figure 3.3 below. We call such chains *traversing* chains, as they traverse the whole of one or more rotational parameters. In Euclidian space, we distinguish *convex* and *concave* chains. Convex chains enclose a finite region of free space, while concave chains bound finite regions of blocked space and are also referred to as *obstacles*. Both the cylinder and the torus are locally Euclidian, so that the same classification applies to them for non-traversing chains. However, as the torus itself is a finite space, the definition may not use infinity. Instead, we define a convex chain on a torus as one that encloses a region that does not contain a closed curve spanning the whole range of a rotational parameter. Examples of convex and concave chains are shown in Figure 3.4.

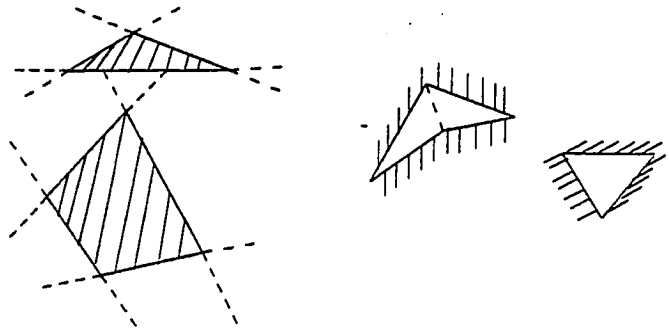

Figure 3.4: Concave (left) and convex (right) chains in configuration space.

Because traversing chains do not intersect one another, all such chains must be parallel to each other. This means that all of them cover the same absolute angles in each rotational parameter. This angle can be described by its *direction index*. For the torus, this is a pair (n,m), giving the multiples of 2π covered in each rotational dimension. For the cylinder, we have only a single such number. Note that there are 2 complementary kind of chains, some that have index (n,m) and some with (-n,-m). Because of the monotonicity of the constraint segments, the index can be found by a march along the chain, adding up the angles covered by each segment.

Each chain of index (n,m) must be paired up with the "closest" chain of index (-n,-m) to bound a doubly connected region of free space. The chains must thus be ordered along a curve that traverses them, i.e., has a different direction index. In general, any of the 2 coordinate axes of configuration space can be chosen for this purpose. The exception to this rule is the case where the chains are parallel to one of these axes: in the case of (0,m), we must use the first axis; in the case of (n,0) the second axis.

Chains which are not traversing must be classified into convex and concave ones. Again, because of the monotonicity, this can be done by a march along the vertices of the chain, adding up the directional changes in the constraint segments. The directional changes at the vertices are ambiguous, as it is not clear whether the turn is to the left or to the right. For example, if 2 successive segments

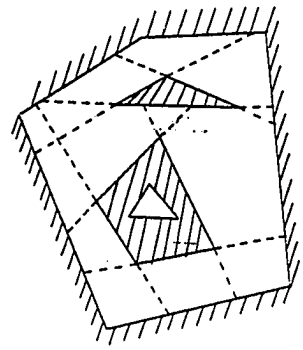

Figure 3.5: Convex chain contained in a concave chain ("obstacle") enclosed in a convex chain.

have qualitative directions (+, +) and (+, -) the turn is -90 degrees for a turn to the right, but +270 degrees for a turn to the left. However, recall that the free space is always to the *left* of the constraint when it is traversed from bottom to top. Thus, the turn is leftward if and only if the vertex is a concave touchpoint. If the sum of the angles is -360 degrees, the chain is convex, otherwise it is concave.

The regions of free space enclosed by pairs of traversing chains or by convex chains usually have to be tesselated further to form the quasi-convex description of the full-dimensional places. This is described in detail in the next section.

The obstacles, given by the concave chains, must be associated with the region of free space they fall into. In general, it is possible for several concave chains to be contained in the region enclosed by a convex chain, and each concave chain may in turn contain a set of convex chains in its blocked space (see Figure 3.5). For the purposes of the place vocabulary and qualitative analysis of a mechanism, it is completely unimportant whether a convex region of free space is contained in some obstacle or not. The only association that is of interest is that of chains with the region of free space that they bound.

Chains may also subsume each other. This is the case if we find that a chain falls within the free space enclosed by another chain, but that chain is not included in the region bounded by the other. After quasi-convex tesselations of the regions have been constructed, this can be tested by one-shot inclusion tests of representative points. Subsumed chains and their associated regions are then simply thrown out.

Finally, the obstacles have to be associated with the regions of free space that they are contained in, and the quasi-convex tesselation has to be recomputed to incorporate these. In the case where no convex or traversing chains exist, all concave chains are embedded in the full configuration space. This is the only case where we have observed concave chains in mechanisms.

3.3 Subdivisions by Applicability Constraints

The applicability constraints for some constraint C do not always run through blocked space, but may also intersect free space. Whenever an applicability constraint is crossed, the set of applicable constraints changes. As all points in the same place are characterized by the fact that the same set of constraints applies to them, applicability constraints require place distinctions wherever they occur in free space.

Recall that applicability constraints are defined by the condition that the vertex (or imaginary vertex) generating the constraint is located on certain lines defined by the boundary involved in the constraint. A necessary condition for an applicability constraint to run through free space is that this defining line itself runs through empty space. Configurations where the vertex is located within the other object are illegal and never part of free space.

This criterion restricts the set of applicability constraints that have to be considered. The cases where applicability constraints may run through free space are the following:

- The constraints meeting at touchpoints formed by concave object vertices.
- The straight lines bounding the applicability of arc boundary segments.

Given some connected region of free space, we first find the set of applicability constraints of the above types associated with the constraints bounding the regions. Each member of this set is tested for intersection with the chains bounding the region. This will define the segments, if any, where the

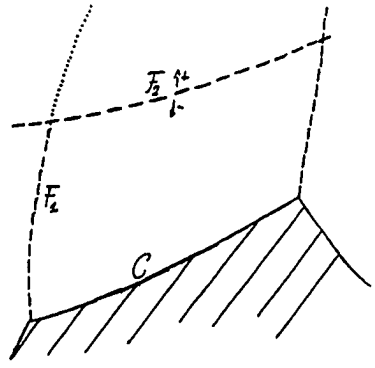

Figure 3.6: The constraint C is not applicable anywhere on the positive side of $\mathcal{F}_2$. $\mathcal{F}_1$ is thus not needed there.

applicability constraint runs through free space, and thus the tesselation imposed by the applicability constraints. We call the individual tesselations *free space divisions.*

The intersections can be found by endpoint tests of the valid constraint segments with the monotone segments of the free space divisions. This again leaves the case of multiple intersections to be handled by algebraic techniques. The newly introduced free space divisions may intersect each other, and such intersections must be taken into account in determining the set of regions (and thus places) formed by the tesselation. Note that in many cases, we can terminate a free space division $\mathcal{F}_1$ at its first intersection with another, $\mathcal{F}_2$. This is the case whenever $\mathcal{F}_2$ does not intersect the constraint C that required $\mathcal{F}_1$, because in this case C is not applicable anywhere on the other side of $\mathcal{F}_2$ (see Figure 3.6). The set of quasi-convex regions formed by the tesselation is given as the set of faces of the graph formed by the constraints and free space divisions, and can be found by a tracing process described below. Each region is a full-dimensional place with the bordering constraints and free space divisions as its boundaries.

For each obstacle (concave chain), the applicability constraints associated with the bounding constraints impose a tesselation of the free space around it. For a set of obstacles in some region of free space, the tesselations for each of the obstacles and that of the region itself must be superimposed.

Thus, we determine the set of intersections of the free space divisions and constraints associated with the obstacle and the divisions and constraints that already define the region and its tesselation.

A special case occurs when this graph consists of several disconnected components. This case is described in the next section.

3.4 Composing Tesselations

In this section, we consider the case where the graph formed by constraints and free space divisions for some connected component consists of several disjoint parts. This case occurs when there exist regions formed by the tesselation which have more than a single disjoint boundary.

We consider first the case where the graph of constraint segments and free space divisions consists of a pair of components $\mathcal{X}_1$ and $\mathcal{X}_2$. Each of the components defines a set of quasi-convex regions, and their intersection forms the quasi-convex tesselation that defines the places. We define the *bounding region* $\mathcal{B}$ of a graph component $\mathcal{X}$ as the finite region bounded by the envelope of the set of constraint segments and free space divisions. Note that because there are no intersections between constraints and free space divisions of $\mathcal{X}_1$ and $\mathcal{X}_2$, the bounding regions of each must be contained in one of the regions defined by the other. This inclusion can be tested using a representative point, such as a point of intersection between constraint segments, or a sample point on a constraint.

Let $\mathcal{R}_1$ be the region defined within $\mathcal{X}_1$ containing $\mathcal{B}_2$, and $\mathcal{R}_2$ the region defined by $\mathcal{X}_2$ that contains $\mathcal{B}_1$. There then exists

- a region $\mathcal{R}_0$ bounded by the union of the boundaries of both regions, i.e., the intersection of the 2 regions: $\mathcal{R}_0 = \mathcal{R}_1 \cap \mathcal{R}_2$.
- all other regions of $\mathcal{X}_1$ and $\mathcal{X}_2$ besides $\mathcal{B}_1$ and $\mathcal{B}_2$.

This is illustrated in the example shown in Figure 3.7. The region $\mathcal{R}_0$ forms the connection between the 2 disjoint components of the graph, as it is bounded by the 2 boundaries from $\mathcal{X}_1$ and $\mathcal{X}_2$. Note that $\mathcal{R}_0$ satisfies the quasi-convexity requirement without any further modification.

In the case where there is a set of components $\mathcal{X}_i$, $i = 1 \ldots m$, the solution must be generalized. Let $\{\mathcal{R}_i^k\}$ be the set of regions defined by $\mathcal{X}_i$. First, we determine the indices of mutual inclusions $k_{i,j}$ such that $\mathcal{R}_i^{k_{i,j}}$ contains $\mathcal{X}_j$. Each set of components $\{\mathcal{X}_i,\ i = 1 \ldots n\}$ such that $k_{i,j} = k_{i,l},\ j,l = 1 \ldots n$

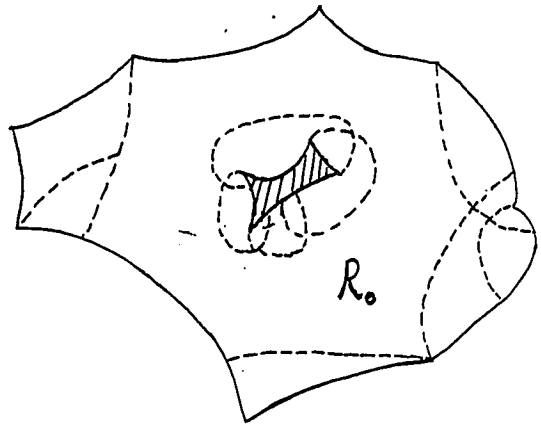

Figure 3.7: The 2 disjoint tesselations form a region with 2 disjoint boundaries $\mathcal{R}_0$, leaving all other regions unaffected.

for all i defines a region $\mathcal{R}_0$ with the n boundaries $\mathcal{R}_i^{k_{i,l}}$, similar to the case of 2 boundaries.

Note that we have to take into account the transitivity of the inclusion relation. If $\mathcal{X}_i$ contains $\mathcal{X}_j$, $\mathcal{X}_j$ will inherit the inclusion relations with respect to the other components from $\mathcal{X}_i$. In a region formed by a set of components including $\mathcal{X}_i$, $\mathcal{X}_j$ is excluded because the inclusion relations between $\mathcal{X}_i$ and $\mathcal{X}_j$ are necessarily different from the others.

The following algorithm will determine the regions connecting the different components. First, for each sign class $\mathcal{R}_i^k$ defined by the component $\mathcal{X}_i$, we determine the set $\mathcal{C}(R_i^k)$ of $\mathcal{X}_j$ included in it. We now form an *inclusion graph* $\mathcal{G}$ in the following way. The nodes of $\mathcal{G}$ are defined by the union of all sign classes of all components that contain some other component, i.e., the $\mathcal{R}_i^k$. The edges are defined by the mutual inclusion relations: an edge exists between $\mathcal{R}_i^k$ and $\mathcal{R}_j^l$ if and only if $\mathcal{R}_i^k$ contains $\mathcal{X}_j$, and $\mathcal{R}_j^l$ contains $\mathcal{X}_i$. Note now that regions bounded by multiple boundaries correspond to maximal cliques in $\mathcal{G}$. The set of regions with multiple boundaries is found by determining them.

Note that each node of $\mathcal{G}$ participates in at most one place and at most one of the resulting cliques. This clique is the maximal one in the set of cliques that the node participates in. This maximal clique can be found by elimination: we start with the set of nodes that the node is connected to and search breadth-first by eliminating one node at a time from this set. We conduct this search for all nodes

in $\mathcal{G}$. We then find the set of maximum cliques by repeatedly picking from the set of cliques the maximum one and eliminating from consideration all cliques associated with its nodes.

Note that the runtime of this algorithm is exponential in the size of $\mathcal{G}$. This is not surprising, as the problem of finding a maximum clique in a graph is NP-hard. However, in well-behaved practical mechanisms inclusions of regions in one another rarely exist , and thus almost all links in graph $\mathcal{G}$ will participate in one of the resulting cliques. In practice, the algorithm should thus run very fast.

3.5 Subdivisions for Mechanical Analysis

The motion of the parts in a mechanism is governed by the forces they exert on each other. An envisionment of the possible behavior of a mechanism is found from the place vocabulary by a qualitative analysis of these forces and the possible place transitions they may cause. For this analysis, we need the following information:

(a) The relationships expressing the sum of the forces and moments acting on the objects (static analysis).

(b) The possible place transitions that may be caused by movement in a certain direction (dynamic analysis).

Note that the qualitative relation between the direction of movement and the forces is determined by Newtonian mechanics; this is the subject of ongoing research (see [NIELSEN]).

Part (a) applies only in the case where a contact between the objects exists, as otherwise no forces or moments can be transmitted. The configuration thus satisfies a constraint, and the relationship of the forces is determined entirely by the derivative of the constraint. For the case of attached objects, we have already seen that the derivatives of the constraint segments are qualitatively constant; the condition is therefore automatically satisfied. In the general case, further tesselations might be necessary to satisfy this requirement.

The set of possible place transitions out of a place given some particular direction of motion is a subset of the set of neighboring places. For 0- or 1-dimensional places, which correspond to constraint segments or intersections of them, there are never more than 3 possibilities, and this set can not be narrowed down any further except by variation of the qualitative coordinate system used. This is

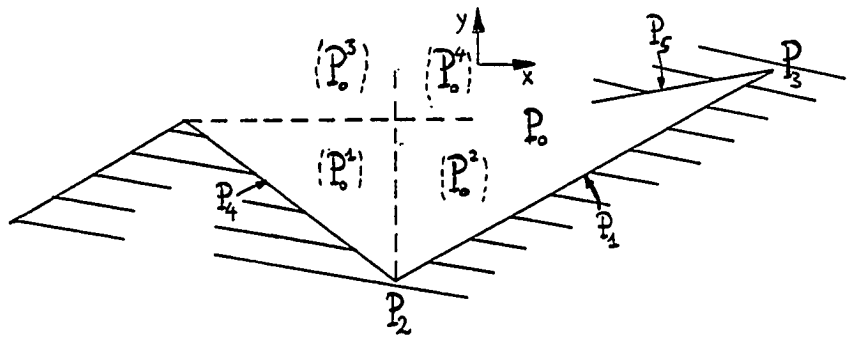

Figure 3.8: Example of places in configuration space. Qualitative directions are specified as pairs of (x,y) in the coordinate system shown.

illustrated in Figure 3.8. From the 1-dimensional place P_1, motions in the direction (+,+) or (-,-) will end up either in the full dimensional place P_0, or in the 0-dimensional places P_2 or P_3. Note that motion in the direction (-,+) will uniquely lead to place P_0, and the motion (+,-) is ruled out. In place P_3, we have the maximum ambiguity: Motion in direction (-,+) might lead either to P_0, P_1 or P_5. In the case of 2-dimensional places, there can exist considerably more ambiguity. In the case of P_0, motion with a negative y-component can lead either to P_1, P_2 or P_4, and this number could become arbitrarily large as there is no bound on the number of possible places at the "lower" boundary of a P_0. It is possible for this number to become unacceptably large. In this case, it can be reduced by breaking up P_0 using additional tesselations as indicated by the dashed lines.

As the research on qualitative mechanical analyses is still in progress, we do not know yet whether such tesselations are necessary or not, and thus have not implemented any of them.

3.6 Place Composition

So far, we have only discussed the computation of place vocabularies for kinematic pairs, in particular for pairs of boundaries on 2 objects forming a kinematic pair. For an analysis of a complete kinematic

chain, we must *compose* these place vocabularies to capture the dependencies that arise because of the common configuration space parameters. There are 2 different cases in which such place composition is necessary: to compose place vocabularies generated by different boundaries of the same objects, and for composing pairwise interactions into descriptions of interactions between many objects. We refer to the first case as *co-dimensional* composition, and to the second case as *chain* composition.

3.6.1 Co-dimensional place composition

To compose places in the same configuration space, we have to find the regions formed by intersections of the regions forming the places. In the general case, this is not an easy problem (see [PS85]). In the case of 2-dimensional configuration space, the boundaries of the places are one-dimensional and have points as intersections. We can find the intersections between the regions by finding all intersections and tracing out the faces of the resulting graphs. Note that because the intersections of quasi-convex regions are again quasi-convex, the resulting regions will automatically be quasi-convex places.

Note that this process is just the same as the one we used for combining the tesselations of a region imposed by different chains of constraints. Thus, we can either use the same procedure over again, or combine the chains defined by the interactions of different boundaries before the association to regions and places.

3.6.2 Chain composition

In the problem of chain composition, we compose 2 place vocabularies whose configuration spaces have one or more parameters in common. Such chain compositions are needed to find a place vocabulary of a kinematic chain from those of the individual kinematic pairs.

To form the composition, the regions corresponding to the places must be "backprojected" into the product space of the 2 C-spaces. The composed place vocabulary is defined by the regions formed by the intersections of these backprojections. Like the co-dimensional problem, in the general case, this is also a very difficult problem.

In the case of 2-dimensional configuration spaces, however, the place vocabularies share at most one single parameter. The surfaces formed by the backprojections of the constraint segments bounding the places (see Figure 3.9) intersect if and only if the intervals they cover in the common parameter

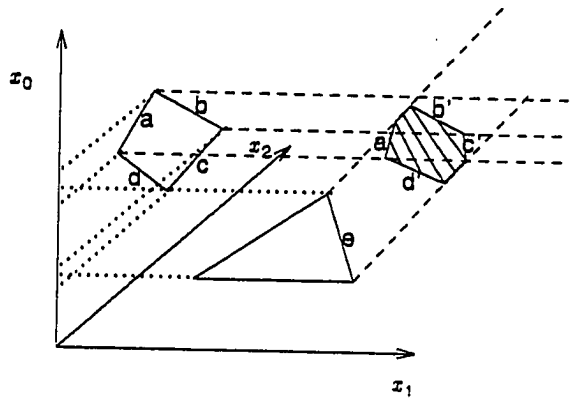

Figure 3.9: The intersections of the backprojections of a, b, c and d define a face bounding the intersection volume on the backprojection of e.

intersect. Because of the monotonicity of the places, there can be only a single continuous interval of the common parameter where the surfaces intersect.

This makes the intersection problem very simple, as it is reduced to interval intersection tests in the common parameter. Furthermore, none of these intersections can be subsumed by other backprojections, so that all the intersections actually occur as boundaries of the intersection. The composition process is illustrated in Fig. 3.9.

3.6.3 Complexity of place vocabularies for kinematic chains

The most straightforward way to qualitatively analyze a kinematic chain is to construct a common place vocabulary that contains *all* dependencies of all kinematic pairs in the chain. In practice, this is a bad idea, because the chain composition process leads to a combinatorial explosion in the number of places. Consider a kinematic chain that consists of n pairs, and let the place vocabularies of each have a size that is O(k). In a chain composition, each place intersects all places whose common configuration space parameter is about the same. If we assume that the places are evenly distributed in the 2-dimensional configuration spaces, the number of these intersecting places is about $O(\sqrt{k})$.

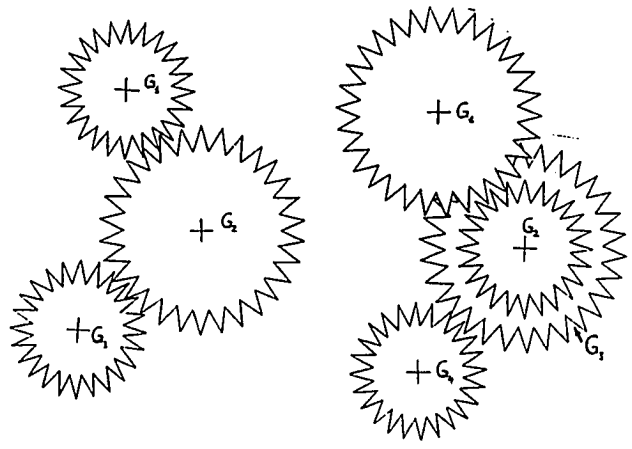

Figure 3.10: In the transmission on the left, the effect of $\mathcal{G}_2$ can be eliminated from the composed place vocabulary. In the transmission on the right, this is not completely possible.

Therefore, the size of the output of the chain composition is $O(k^{3/2})$. For successive compositions, we thus expect the size of the place vocabulary to grow exponentially with k.

In considering a kinematic chain, we are usually interested only in the relation between the first (input) and last (output) elements of the chain. In certain cases, we can deal with the combinatorial explosion by eliminating all information about intermediate elements of the chain from the composed place vocabulary and projecting it into a configuration space for the input and output elements.

As an example, consider a transmission made up of 3 gearwheels G_1, G_2 and G_3 with n_1, n_2 and n_3 teeth, respectively, such that G_2 forms kinematic pairs with G_1 and G_3. This is shown on the left in Figure 3.10. Let P_1, P_2 and P_3 denote the configuration space parameters of the 3 objects. Both place vocabularies consist of a single connected region which is broken up into places corresponding to the interactions between pairs of teeth. For the purposes of this discussion, we assume that one place is formed for each interaction of teeth. These places are periodic repetitions of each other, and there are $n_1 n_2$ and $n_2 n_3$ of them, respectively. These places are arranged in periodic sequences of

length $LCM(n_1, n_2)$ and $LCM(n_2, n_3)$, respectively. Note that if there is no distinction between the teeth of the gearwheels, the sequences are all equivalent and may be represented as a single one. Now consider a place X_0 in the place vocabulary for G_1 and G_2, and let X_1 be the adjacent place offset by $2\pi/n_1$ in P_1 and $-2\pi/n_2$ in P_2. Note that X_1 is the place reached from X_0 when the wheels are turned by one tooth. Let Y_0 and Y_1 be similar places in the place vocabulary for G_2 and G_3, i.e., Y_1 is offset from Y_0 by $-2\pi/n_2$ in P_2 and $2\pi/n_3$ in P_3. The intersections of the backprojections of X_0 with Y_0 and X_1 with Y_1 are also adjacent, and offset from each other by $2\pi/n_1$ in P_1, $-2\pi/n_2$ in P_2, and $2\pi/n_3$ in P_3. The length of this sequence will be $LCM(n_1, n_2, n_3)$, and the total number of places in the intersection $n_1 n_2 n_3$.

However, the number of different offset combinations in the P_1 and P_3 parameters is only n_1 and n_3, respectively. This means that when we consider the projections of the places into the P_1/P_3 plane, there can only exist $n_1 n_3$ periodic copies of any place. This in turn implies that each such projection is shared by n_2 places with different offsets in the P_2 direction. If the P_2 coordinate is of no interest, each such set of n_2 copies can be replaced by a single place, and the resulting combined place vocabulary for G_1 and G_3 has only $n_1 n_3$ places.

In the case where n_1, n_2 and n_3 have common divisors, some of the sequences in the combined place vocabulary may again be represented as a single one.

Note that such collapsing can be done only in the case of periodic repetitions with the same periods. In the above example of a transmission, this is the case, because the same boundary of G_2 is used in both kinematic pairs. In most transmissions, however, successive kinematic pairs use entirely different sets of gears, joined by common shafts. This case is illustrated on the right in Figure 3.10. In this example, the gear G_1 drives a gear G_2, which shares a common shaft with G_3, which in turn drives G_4. We compose the place vocabularies for the interaction of G_1/G_2 with that for the interaction of G_3/G_4. Contrary to the earlier example, the offsets in the common parameter are now different for the 2 place vocabularies. But this means that the places resulting from the composition intersect with different relative offsets in that parameter, and can not be collapsed into a single one. The only possible simplification is the reduction to a single periodic sequence of states when n_1, n_2, n_3 and n_4 have common divisors.

3.7 Computation by Algebraic Decision Methods

The subproblem of determining intersections between constraints and their ordering can be solved reliably only by algebraic *decision methods.* Furthermore, based on such decision methods there are algorithms for algebraic cell decomposition that give a (terribly inefficient) way to compute place vocabularies for the general case. We shall first discuss the mathematical basis of algebraic decision methods and then show how they apply to the problems of intersection detection and place vocabulary computation.

3.7.1 Decision methods

There exists a theorem in algebra which allows exact calculation of the number of roots of a polynomial in a given interval. This theorem is

Sturm's Theorem: The number of simple roots of a polynomial P(x) with only simple roots in some interval [a..b] is given by $|n(b) - n(a)|$, where n(x) is the number of sign changes in the sign-inverted remainder sequence of P(x), P'(x). By a sign-inverted remainder sequence we mean a remainder sequence where each element is the *negative* of the remainder of the pre-preceeding element divided by the preceeding one.

Note that this theorem determines only the existence of the roots, but not their location. It has been generalized by Alfred Tarski and used to develop a decision method for algebra ([TAR48]). In this work, he develops an algorithm to derive, for a quantified logical expression made up of predicates on the signs of polynomials, an equivalent expression containing no quantifiers. This latter expression can then be evaluated to determine the truth of the original expression.

The method is based on Tarski's generalization of Sturm's theorem:

> Given 2 simple and relatively prime polynomials p(x) and q(x), the difference in the number of roots of p on [a..b] such that q(x)>0 and the number of roots such that q(x)<0 is equal to n(b)-n(a), where n(x) is the number of sign changes in the sign-inverted remainder sequence of P(x), P'(x)Q(x).

Recently, Collins ([COLL75]) and later Ben-Or, Kozen and Reif ([BKR85]) developed simplified decision methods based on the idea of *algebraic cell decomposition.* Using Tarski's generalization

together with Sturm's theorem itself, we can compute the number of roots of P for Q>0 and the number of roots where Q<0 in a given interval. These allow us to determine the *sign classes* of P and Q, i.e., the set of legal simultaneous sign assignments to P and Q. This divides up the domain of the polynomials into a set of *cells* because each sign assignment will be valid for a certain range of the parameter. The boundaries of the cells are found as the roots of the polynomials that appear in the remainder sequences of Sturms theorem.

We can extend this method and use it to determine whether a set of multivariate polynomials has any simultaneous roots (intersections) in the following way. We recursively isolate one variable in all the polynomials considered and compute the required remainder sequences. The next step in the recursion applies to the set of polynomials that appear in the remainder sequences until only a single parameter remains. Note that at this point, the signs of each polynomial in the set will influence the result. Therefore, there exists an ambiguity in the sign classes, and therefore a common root, exactly when one of the polynomials in this set has a root within the range of the parameters. Note that any application of Sturm's theorem or its generalization must be preceeded by eliminating all multiple roots of the polynomial in question. This can be done by algebraic techniques using computations of greatest common divisors as described in ([KNUTH81]).

3.7.2 Application to constraint intersection detection

In the case of a 2-dimensional configuration space, the constraints can be parameterized by one or more parameters. If we can parameterize the constraint by a single parameter r, then we can find the number of intersections of 2 constraints C_a and C_b as just the roots of $g_a(f_b(r))$, where f is the function from the radius to a sample point on the constraint and g is the equation of the constraint itself. The number of such roots in the interval corresponding to the constraint segment can be computed readily using Sturm's theorem.

In the case where the constraint is parameterized by several parameters r_0, r_1, ... , r_n which are related by algebraic conditions $Q_0(r_0, r_1, \ldots) = 0$, ..., $Q_{n-1}(r_0, r_1, \ldots) = 0$, we can determine whether an intersection exists by applying the multivariate method outlined above to the set $\{g_a(f_b(r_0, r_1, \ldots))\}$. This test, however, can become quite expensive if the number of variables is high. We have seen in the earlier chapter that for straight lines and arcs, all constraints can be parameter-

ized algebraically with at most 4 parameters. In the general case, we can always limit the number of parameters that are necessary by the number of configuration space parameters by applying the above technique to the set $\{g_a, g_b\}$ augmented by any algebraic relations that may be given between the configuration space parameters (e.g., sine and cosine in the angle specifications).

3.7.3 Constructing place vocabularies by algebraic methods

Decision methods have been applied to motion planning problems by observing that the condition that objects do not overlap can be expressed as a quantified formula. One can construct an algebraic cell decomposition of the expressions that occur in this formula and then, for each cell, determine whether the points in it are legal configurations or not. This gives a description of free space as a set of cells, which can be arranged in their adjacency graph. Paths for a desired motion can then be found by graph searching. The first algorithm proposed for this is that of Schwarz & Sharir ([SCHS83b]), and recent work on the topic has resulted in more efficient algorithms ([KY85]). The latter paper also shows that the problem is in the class NC and can therefore be efficiently implemented in parallel.

While the adjacency graph constructed for the cell decomposition appears similar to our place graph, the two are in fact quite different. The cells constructed for the solution of the decision problem contain no information whatsoever about *applicability* of constraints. Thus, cell distinctions are forced everywhere along a constraint curve, not just along the valid segment. In our case, this leads to meaningless place distinctions, as a constraint would have influence on the place distinction in completely unrelated parts of the mechanism.

Based on algebraic decision methods, there exists the following method to construct a place vocabulary for kinematic pairs. First, we construct the cell decomposition of the C-space with respect to all constraints and applicability constraints using, for example, the method of Kozen & Yap ([KY85]). This gives us a set of quasi-convex, connected cells within which there are no sign changes of constraints or applicability constraints. We then merge the cells according to the following criteria:

> Two cells can be merged if they differ only in a single boundary $\mathcal{F}_i$ such that one has the condition $\mathcal{F}_i < 0$ or $\mathcal{F}_i > 0$ and the other has the complementary condition or $\mathcal{F}_i = 0$, and $\mathcal{F}_i$ is either (i) an applicability constraint whose associated constraint is not among the boundaries of either cell, or (ii) a constraint which is not applicable in both cells.

Some computation can be saved in the decomposition step by not computing separate sign assignments for cases where the cells would be merged anyway, for example, applicability constraints need only be included together with their associated constraint.

Not that the problem of merging cells corresponding to the different sign classes is formally the same as that of designing a switching function from a truth table. We can use algorithms such as that of Quine-McKluskey to find the prime implicands. The regions corresponding to these will in general overlap and then have to be divided to make disjoint regions. Logic design has developed algorithms for this problem as well.

The place graph is found by eliminating from the resulting set of regions those that lie in blocked space, as defined by the sign assignments.

Chapter 4

ALGORITHMS

In this chapter, we describe the actual implemented algorithms. for computing place vocabularies for kinematic pairs. The computation is divided into 2 stages:

- an *instantiation* stage, where the possible constraints defined by the symbolic part of the metric diagram are collected, and
- an *evaluation* stage, where the numeric information is used to generate an actual place vocabulary.

The instantiation stage allows unknown numeric parameters. Leaving all parameters open results in a large number of ambiguities in the instantiation and makes the algorithms very slow. In our implementation, we have therefore chosen to allow symbolic specification of parameters only for the reference points in the global coordinate system (that is, the relative position of the objects may be unknown). This restriction is made only for the sake of efficiency of the code, and there are no algorithmic problems with eliminating it.

In the evaluation stage, tests on the metric parameters are performed to determine the set of active constraints, their intersections and finally the place vocabulary. These tests can be understood as a set of predicates on the metric dimensions of the objects. Each of them thus defines a *landmark value* for whatever parameter might be unknown. As many of the necessary tests are already known during the instantiation stage, an initial set of landmark values is generated there. These remaining

landmark values are not independent of the choice of parameters; they are found by an iterative procedure described later.

In the evaluation phase, all the metric parameters must be fixed. There is no mechanism for dealing with ambiguities; ambiguous cases must be generated by running the evaluation for different choices of metric parameters. This is discussed in detail in the next chapter.

In the next sections, we describe the algorithms used in the 2 stages. First we discuss the instantiation and the data structures used to represent the objects. Next, we describe how the chains are found and finally how the actual place vocabulary is computed. Finally, we discuss how periodicity of the parts can be exploited, and how the computation of single place vocabularies is embedded in the qualitative analysis of complete mechanisms.

4.1 The Instantiation Phase

In the instantiation phase, we compile the set of possible constraints from the description of the object boundaries and their attachments. Recall that constraints arise from interactions of the following pairs of elements:

- a convex vertex of one object boundary and a segment of the other (vertex constraints)
- a convex arc segment of one boundary and a segment of the other (boundary constraints)

The instantiation thus proceeds in a loop through all possible pairs of boundary segments and vertices, instantiating the possible constraints for each pair. The input representation allows specification of a periodic repetition of a sequence of boundary segments. This is exploited by allowing the constraints for the repeated segments to be instantiated as copies of the original constraints, offset in configuration space by the appropriate amount.

Vertex constraints are joined together in the configurations where a pair of vertices touch each other. We call these configurations *touchpoints*. Each constraint we instantiate has to be linked to the touchpoints at its ends, and thus the first step is the instantiation of the data structures for the touchpoints. For each pair of object boundaries, these are held in a 2-dimensional array called the *basis*, indexed by the vertices of the object boundaries they correspond to. The basis also allows indexing of the constraints via the touchpoints at their ends.

For each pair of boundary segments, up to 3 possible constraints are generated: the 2 vertex constraints corresponding to the vertex at the beginning of one segment sliding on the other segment, and the boundary constraint corresponding to the touch of the 2 segments. Each of these is then possibly further subdivided to satisfy the requirements of monotonicity. In the case of rotational attachments, we generate the mirror images. At this stage, we also instantiate the touchpoints between the 2 vertices at the beginning of the segments.

In the following sections, we first describe the data structures we use for the constraints and touchpoints. We then describe how the instantiation of each of the elements takes place.

4.1.1 Data structures

In this section, we describe the data structures used to represent the configuration space constraints. The main elements here are *constraints* and *touchpoints*.

Constraints

For a constraint, we specify the following information:

1. its *equation*, a function of the 2 configuration space parameters that is zero on the constraint curve itself, and >0 in free space.
2. its *tangents*, a pair of functions giving a tangent vector to the constraint curve in configuration space. The tangent points in the direction from the lower to the upper endpoint of the constraint.
3. its *applicability constraints*, split up into 2 parts: a set which can and a set which can not run through free-space. For the first set, we specify only a list of functions of configuration space coordinates which is >0 wherever the constraint is applicable. The second set is specified as a list of applicability constraint structures.
4. its *endpoints*: we distinguish between a lower and an upper endpoint, which correspond exactly to the lower and upper end of the boundary segment involved. For mirror images, the direction of the tangents is reversed. In order to preserve the convention that the tangent points to the upper end of the constraint, the upper and lower endpoints are exchanged.

5. a *find-point-form*, a function from parameter (radius) values to the configuration on the constraint corresponding to the parameter value. Due to the existence of transcendental parameters, this is split up into 2 parts: the parameter definition and the actual algebraic forms to be evaluated.
6. lower and upper bounds on the parameter r. Given by the radius values of the endpoints of the boundary segment. Note that lower and upper here refer to the ends of the constraint, not the magnitude of the values.
7. lower and upper r_Δ, the possible extremal values of the parameter. In this case, lower and upper refer to the magnitude of the values.
8. some duplicated fields to save the values of evaluations in order to make the code more efficient.
9. bookkeeping information about the type (:BOUNDARY or :VERTEX), indices of boundary and vertex involved, etc.

The structure for applicability constraints has the following elements:

1. an *equation* and *tangent*, similar to a constraint.
2. one or several *find-point-form*(s), because the applicability constraint may have a non-unique parameterization.
3. a set of intervals of r_Δ to allow tracing out the complete curve of the applicability constraint.
4. a set of *endpoints* of the intervals covered by the intervals of r_Δ.

Note in particular that the lines that make up the applicability constraints are usually not broken up into segments that ensure unique parameterizations. The applicability constraint might be parameterized in several segments to account for the different segments. The endpoints of each constraint are linked to the *touchpoints*.

Touchpoints

We associate the following information with touchpoints:

1. its coordinates in configuration space

2. an array of 4 endpoints of the constraints that may end there. Indices 0 and 1 hold lower endpoints, and 2 and 3 hold upper endpoints.

3. up to 2 predicates to indicate if the radius values corresponding to the touchpoint are too large or too small for the touchpoint to exist. The violation of these indicates which r_Δ has to be used to infer endpoints of the constraints.

4. other bookkeeping information about the type, indices, etc.

The touchpoints are arranged in an array called *basis*. The 2 coordinates of this array are the indices of the vertices involved in each touchpoint. Whenever at least one of the objects has rotational attachment, the first coordinate is doubled so that the even coordinates refer to touchpoint originals, the odd ones to their mirror images.

4.1.2 Instantiation of constraints and touchpoints

The first step in the instantiation is to break up the boundary segments on each boundary into segments that have a unique parameterization in terms of the radius. In our current implementation, this is actually done *during* the instantiation of the constraint, as this allows some savings of computation. For the description of the algorithm, however, we will consider this a separate step. In the following, we thus assume that each boundary segment has such a unique parameterization.

Consider then the pair of boundary segments B_{ai}, the i-th boundary segment on boundary a, and B_{bk}, the k-th boundary segment on boundary b. Let the initial vertices of the 2 segments be V_{ai} and V_{bk}. The following elements have to be instantiated for this pair:

1. the *touchpoint* between V_{ai} and possibly its mirror image.

2. if V_{ai} is convex: the *vertex constraint* between V_{ai} and B_{bk} and possibly its mirror image.

3. if V_{bk} is convex: the *vertex constraint* between V_{bk} and B_{ai} and possibly its mirror image.

4. if either of the 2 boundary segments is a convex arc, and the other is anything but a concave arc of smaller radius than the first, the *boundary constraint* between B_{ai} and B_{bk} and possibly its mirror image.

The touchpoints are instantiated as touchpoints indexed $(2i,\ k)$ and $(2i+1,\ k)$ in the basis array. Recall that the first dimension of the basis array is doubled to account for the mirror images. The

lower endpoints of both original vertex constraints are linked to the touchpoint just instantiated as constraints indexed 0 and 1, and the upper endpoints of the mirror image vertex constraints are linked to the mirror image touchpoint as constraints indexed 2 and 3. The upper end of the original vertex constraint between V_{ai} and B_{bk} is linked to the touchpoint indexed $(2i, k \oplus 1)$ as endpoint with index 2, where the symbol $\oplus$ indicates summation modulo the size of the array. The upper end of the other original vertex constraint is linked to the touchpoint with index $(2i \oplus 2, k)$ as endpoint with index 3. The lower ends of the mirror images of the vertex constraints are linked the mirror images of these touchpoints as endpoints with index 0 and 1.

Note that by this instantiation, each touchpoint that has an incoming constraint with index 0 must have an outgoing constraint with index 2, and similarly for indices 1 and 3, as these pairs are generated by the same vertex.

Note also that the upper and lower endpoints of mirror images are exchanged. This is because the tangents of the constraints reverse their direction on the mirror image. Thus, traversal of the mirror images in the direction of the tangents means traversal in the opposite direction to the originals.

The boundary constraints are not linked into an array at all. The only a priori link that exists between them is between boundary constraints generated by segments that were broken up to give a unique parameterization. The actual link to the vertex constraints at the ends can be determined only in the evaluation stage.

4.1.3 Algebraic simplification

The instantiation of the various equations, predicates and functions requires symbolic manipulation of algebraic expressions. The algebraic simplification system we use is a modified version of a system originally developed by G. Sussman of MIT. The major addition that had to be made were methods for the manipulation of square roots. Fortunately, this is the only transcendental function that appears in the expressions, and there exists the following algorithm for taking the square root of a polynomial (if it exists).

Let $P(x)$ be the monic polynomial of degree n whose square root we would like to obtain, and let

P have the k roots $r_i, i = 1 \ldots k$:

$$P = \prod_{i=1}^{k} (x - r_i)^{j_i}$$

The derivative P' is then

$$\begin{aligned} P' &= \sum_{i=1}^{k} j_i (x - r_i)^{j_i - 1} \prod_{l=1, \neq i}^{k} (x - r_l)^{j_l} \\ &= \left(\prod_{l=1}^{k} (x - r_l)^{j_l - 1} \right) \left(\sum_{i=1}^{k} j_i \prod_{l=1, \neq i}^{k} (x - r_l) \right) \end{aligned}$$

Note that the second factor in the expression for P' is not divisible by any $(x - r_l)$ for l in $\{1 \ldots k\}$, and thus does not occur in the greatest common divisor of P and P':

$$GCD(P, P') = \prod_{l=1}^{k} (x - r_l)^{j_l - 1}$$

In particular, we can now find the product of all the roots of P as

$$R = \frac{P}{GCD(P, P')} = \prod_{l=1}^{k} (x - r_l)$$

Now, define a sequence Q_i in the following way:

$$\begin{aligned} Q_1 &= GCD(P, P') \\ Q_{i+1} &= GCD(Q_i, Q_i') = \prod_{l=1}^{k} (x - r_l)^{max(j_l - i, 0)} \end{aligned}$$

Now note that

$$\begin{aligned} GCD(P, R) &= \text{product of all roots of P (= R)} \\ GCD(Q_2, R) &= \text{product of all roots of P with multiplicity} > 2 \\ GCD(Q_i, R) &= \text{product of all roots of P with multiplicity} > \text{i} \end{aligned}$$

Hence, if $P = S^2$, we have that

$$\begin{aligned} S &= \left(\prod_{l=1}^{k} (x - r_l) \right) \left(\prod_{l, j_l > 2} (x - r_l) \right) \cdots \\ &= R \, GCD(Q_2, R) GCD(Q_4, R) \cdots \end{aligned}$$

The GCD computations are implemented using the subresultant method of Collins, Brown and Traub described in ([KNUTH81]). Whether the candidate S is actually a root of P is verified by comparison of S^2 with P. For non-monic P, we split off the leading coefficient at the beginning of the algorithm and have to multiply the result S with the root of this coefficient.

4.2 Computation of Chains

In this section, we describe how we find the chains of constraint segments that bound regions of free space from the instantiation of possible constraints and touchpoints. This computation proceeds in the following 3 stages:

- first, the instantiation is completed for the actual parameter values. The subset of touchpoints and constraints which are actually active is determined, endpoints are inferred if necessary, constraints are ordered around touchpoints, boundary constraints are linked to the vertex constraints where they end, and other bookkeeping information is instantiated.
- the subsumption intersections between active constraints are determined and ordered.
- the resulting graph of constraint segments is traced out to find the chains.

In the following sections, we describe the algorithms used in each of these steps.

4.2.1 Determining the set of active constraints and touchpoints

For each touchpoint, we have instantiated predicates to determine if it is active. The first step is to evaluate these predicates to determine the active subset of touchpoints. If a touchpoint is inactive, the particular violations of the predicates indicate at what parameter value the endpoints of the constraints might be inferred, as described earlier.

A constraint is active whenever at least one of the configurations on it exists. This means that there must be an intersection between the interval of the radius values of the possible points of touch and the interval of the radius that the boundary segment covers. This intersection is determined by the predicates for the existence of the touchpoints at the ends of the constraint. If the intersection interval exists, the parameter values of the endpoints of the constraint are given by the ends of this

interval. The second step is to determine the active subset of constraints and instantiate the inferred endpoints, if necessary. If an endpoint of a constraint is inferred, that endpoint links the constraint to its mirror image, and we mark an intersection of the constraint with its mirror image.

For a boundary constraint, there are 4 possible vertex constraints (any of the 4 endpoints of the 2 boundary segments touching the other) and their mirror images that either end could connect with. The set is narrowed down to 2 candidates (a constraint and its mirror image) by the criteria described earlier. Which of the 2 remaining possibilities is actually applicable is determined by testing for each of them whether the candidate actually touches the boundary constraint at its end. All of these tests may generate additional landmark values.

For each active constraint segment, it must be determined whether there exists a *dead point.* Recall that a dead point is a point where the derivative of the constraint becomes zero, requiring the constraint segment to be broken up. The algorithm determines the radius value at which such a dead point could exist and then tests whether the corresponding configuration actually is a dead point, as there exists ambiguity in the radii of the possible dead points. [1] We then determine a set of bounding rectangles for the constraint. Note that at least 2 of them are required in the case where a dead point exists, and further breakups may be necessary as described in the previous chapter.

At each active touchpoint, only 2 of the 4 constraints that meet are actually applicable. The applicable pair is determined by the ordering of the constraint directions around the touchpoint. This ordering can be reduced to a set of predicates on the relative directions of the boundary segments when the objects are in the touchpoint configuration, which may generate additional landmark values. For the 2 constraints that make up the active pair at the touchpoint, we mark an intersection at their endpoints. For the endpoints of the other 2 constraints at the touchpoint, no intersecting constraint is marked.

Finally, we mark the direction of each constraint segment by evaluating the tangents at a sample point in the center of the parameter range. For each endpoint of an active constraint, we store a copy in a separate field. This copy has fully evaluated numeric coordinates and the value of the radius parameter that was actually used for the endpoint. We also instantiate a simplified version of the find-point-form where all parameters are eliminated to speed up later computation.

[1] This ambiguity could be eliminated in the instantiation phase, but the solution we use here is simpler.

4.2.2 Determining intersections

Each constraint may form intersections with others at its endpoints; these are found and marked when the endpoints of the constraints are determined. Besides these, *subsumption intersections* can exist between any pair of constraints whose bounding rectangles intersect. Constraints C_1 and C_2 form a subsumption intersection whenever one of the endpoints of C_1 violates C_2 and the other does not, and likewise with C_1 and C_2 exchanged. Based on the earlier discussion, these intersections are found in the following way. Once a pair of intersecting bounding rectangles on C_1 and C_2 has been found, the ranges of the constraint parameters where they actually fall within the intersecting area of the bounding rectangles is determined. If these intervals are non-empty, we test the constraint points at the ends of these intervals by the endpoint test and thus determine if an intersection exists. This test is reliable: the intersection of the bounding rectangles does not contain any inapplicable segments of the constraint curves.

Note that it is possible that the constraint segments intersect each other in an even number of points. Such intersections will not be found by endpoint tests, and the algebraic methods described in the previous chapter are required to reliably detect them. Note that multiple intersections will cause local errors in the place vocabulary, which could be corrected in a later cleanup stage. As we have not observed any instance of such a case in practical mechanisms, we have not implemented any algorithms for this yet.

If a constraint has subsumption intersections with several others, they must be ordered. This requires in particular a way to determine the ordering relation of the intersection points on a constraint segment. The current implementation does this by finding a point between the intersections using a binary search, and then testing the relative locations at that point. Note that this does not require determining the actual configurations where the subsumptions occur. In certain cases, it might be possible to use simpler methods to determine the ordering relations. However, this requires making assumptions about well-behavedness of the constraint curves, and all our attempts at this have been thwarted by actual examples of constraint curves that violated the assumptions.

There exist landmark values where subsumption intersections appear, disappear or their ordering changes. These are best found by considerations different from the algorithms described above. The landmark values where subsumption intersections between constraints C_1 and C_2 appear are charac-

terized by the fact that an endpoint of C_1 falls on C_2 or vice versa. The predicates for the landmark values are thus defined by the roots of the instantiation of the constraint equation of C_1 with the endpoint coordinates of C_2, and vice versa.

The ordering of the intersections of 2 constraints C_1 and C_2 with C_0 changes when all three constraints intersect each other in a single point. The corresponding landmark values for this can be found using algebraic techniques discussed earlier, and there does not seem to be any simpler way to do this. We have not implemented this particular part of the landmark value computation. We detect whether the ordering of the intersections changes within an interval of a parameter by testing the orderings when the parameter is at the endpoints. If we detect that the ordering changes, the resulting place vocabulary is ambiguous. This test might fail in the case where the ordering changes more than once, and this case is beyond the capabilities of the current implementation.

Intersections are marked as lists of special data structures. For each constraint, there exists a separate list for each bounding rectangle, as the other constraint may intersect within any of these subsegments. The ordering of the intersections is thus partially given by the ordering of the bounding rectangles.

4.2.3 Tracing out the chains

The tests outlined so far uniquely determine the boundaries between free and blocked space as pieces of configuration space constraints. The boundaries between regions of free space and blocked space are *chains* of these constraints. They are found by tracing out the faces of the graph formed by the constraints and their intersections, picking only those where the inside of the face lies on the free space side of all the bounding constraints.

In particular, on each constraint we mark the valid segments between the intersections. The first and last intersections are defined by the constraints that the endpoint of the constraint links to. An interval between 2 intersecting constraints is a valid *constraint segment* if and only if the 2 intersecting constraints together bound a region of free space, as shown in Figure 4.1. Each such constraint segment is linked to the intersection points at its ends. This defines the graph of constraint segments and intersections. Note that each intersection point is associated with at most 2 constraint segments, but many will have only one or even none at all. This is the case when the intersection

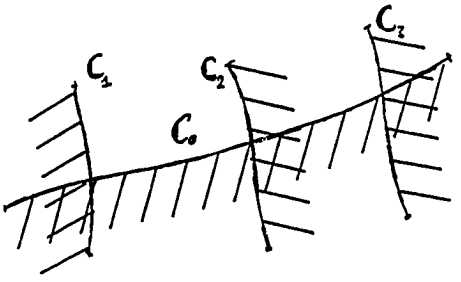

Figure 4.1: On C_0, the segment between intersections with C_1 and C_2 is legal, but the one between C_2 and C_3 is not.

point is subsumed by other constraints on one or both of the constraints whose intersection it marks.

The graph is searched to find all closed chains of constraint segments. This is done by a march along the vertices, starting at an arbitrary unexamined vertex, until either the beginning of the march is reached, or a vertex with less than 2 constraint segments is reached. In the first case, a chain has been found, whereas in the second case, none of the vertices examined in the march can be a member of any chain. Applying this tracing procedure until no more unexamined vertices exist will find all chains in the graph.

4.3 Determining the Places

In this section, we describe how the chains are associated with the connected regions of free space that they bound and how the place vocabulary is computed from them.

For each chain, we have to determine its *direction index* as discussed in the last chapter. In the case of rotational freedom, the topology of the configuration space is not Euclidian, and there may be chains that do not bound a region of configuration space by themselves. This occurs in the case of traversing chains. The direction index is determined by a march along the chain which counts the number of zero crossings in each rotational dimension. All chains with a common direction index are then ordered along a transversing coordinate axis. This ordering is determined by the value at the

zero crossing points of the coordinate with the least zero crossings. This particular rule ensures that the chains are never parallel to the chosen coordinate. Regions of free space are bounded by pairs of chains adjacent in this ordering such that one chain has index (n,m) and the other (-n,-m). This determines the association between traversing chains and regions of configuration space. The rest of the chains are classified into convex and concave ones by a march along the vertices as described earlier.

The next step is to determine the tesselations that the applicability constraints impose on the regions bounded by convex chains and pairs of complex type chains. For each region of configuration space, we determine the set of applicability constraints associated with the constraints bounding the region. The subset of these which potentially run through free space is tested for intersections with the constraint segments bounding the region. These intersections are found by endpoint tests like those of the constraints themselves. There are 2 ways in which an applicability constraint could enter free space: either the touchpoint it emanates from is part of the region boundary, or the applicability constraint intersects one of the constraints bounding the region. If such points are found to exist, *free space divisions* are introduced between them. If the applicability constraint intersects a constraint segment, it is broken into 2 parts at the intersection. A free space division is represented by a special data structure. Similar to a constraint, it has an equation, a find-point-form, and a parameterization.[2] However, each free space division has 4 endpoints (2 each for the constraint segments it is linked to at each end), and free space exists on both sides of it. In specifying it as a region boundary, we thus have to distinguish its positive and negative sides.

The resulting structure is traced out once again to find the regions formed by this tesselation. It is possible that the graph formed for some region consists of several disjoint parts. In this case, there exist places with multiple disjoint boundaries. These are found by testing mutual inclusions using the algorithms described in the previous chapter.

Finally, the concave chains (obstacles) are tested for inclusion in the regions bounded by the convex and traversing chains. Each obstacle by itself defines a tesselation of configuration space through its constraints and free space divisions. These tesselations must be superimposed on the tesselation that already exists for the region. This requires computing the additional intersections between pairs of

[2]Just like in applicability constraints, this parameterization may consist of several parts.

applicability constraints associated with the obstacle boundaries and constraints associated with the region boundary, and vice versa. The resulting graph is then traced out one more time as described earlier. The resulting regions, the full-dimensional places, are represented in structures of type *full-face.*

Note that the kinematic behavior of a mechanism is completely determined by the possible chains and their association with connected regions of free space. The quasi-convex tesselation does not reflect a kinematic interaction, but is a device to keep track of constraints that might potentially be reached by intermittent motion. The connectivity of free space does not change unless some of the chains change. There are thus no new landmark values generated in this stage of the evaluation.

4.4 Periodicity

In mechanisms, many parts have boundaries which consist of periodic repetitions of small sequences of segments. For example, the teeth in the boundary of a gearwheel are periodic repetitions of each other. We refer to each instance of such periodic repetitions as *period.*

The input representation allows specification of periodicity in the direction of attachments. Specifically, we have the following 2 cases:

- in the case of translational attachments: a sequence of boundary segments can be repeated n times along the axis of possible translation.
- in the case of rotational attachments: a sequence of boundary segments can be repeated n times around the center of rotation *to form a closed boundary.*

Periodicity is used only in this restricted form because this is the only way it can be exploited profitably. The periodic repetitions of the boundary segments generate *identical* constraints with different configuration space offsets exactly when the periodic repetitions are offset by a variation in the configuration space parameter. We distinguish between the *period original* and the *period copies* of each constraint. The reason for requiring the periodic segments to form a closed boundary in the case of rotational attachment is that this is the normal case, and very difficult to specify otherwise because of roundoff errors. The implementation exploits the restriction and automatically generates the proper offsets.

The periodicity is used to improve the efficiency of the code in several places. First, in the instantiation each periodic copy of a constraint or touchpoint is instantiated as a copy of the original by adding the proper offsets where necessary. Each combination of periods from the 2 objects defines a set of offsets for the constraints generated by the parts of the boundaries that belong to the periods. In the evaluation, only the originals are tested for intersections. For each constraint that intersects the original, the analogous intersecting constraint is found for each periodic copy by picking the copy with the proper offset. The intersections of the constraint originals are ordered before the copies are made, and this ordering is preserved in the copies. Note that we must test the originals against *all* constraints, because a constraint may intersect any of the periodic copies. This is the reason why the algorithm actually instantiates *all* the periodic copies. Also, intersections can be copied only from original to periodic copy if all constraints involved are part of the *same* periods, as the configuration space periods of different periods will in general not agree. All pairs of constraints belonging to different periods must be tested for intersection individually.

In the case where the boundaries of both objects consist of a single period, all constraints belong to the same periods, and the resulting place vocabulary is thus periodic. As it is useful for the qualitative analysis to capture this periodicity, the output representation is marked to indicate it. Specifically, we construct a *period descriptor* for the pair of periods, which describes the number and step size of the different offset combinations. The output place vocabulary then contains only a set of original places, and all periodic copies of it are defined by the offsets in the period descriptor.. The places are specified in the place graph by indicating the place original together with the appropriate offsets. This representation is constructed from the output place vocabulary using the links between constraints and their periodic copies.

4.5 Mechanism Analysis

A complete mechanism consists of a set of kinematic chains, each of which is in turn made up of a set of kinematic pairs. In the previous sections, we have described the algorithms to compute place vocabularies for kinematic pairs. In this section, we outline the framework within which these can be used for the analysis of complete mechanisms.

By applying the composition algorithm described in the previous chapter to the place vocabularies for each kinematic pair, a combined place vocabulary for the chain could be constructed. However, this may lead to a very large number of places. Only in certain cases of periodic repetitions can the combinatorial explosion be avoided.

Due to the definition of the places, the composition process itself consists only of a set of interval intersection tests. It can thus very easily be carried out on demand during the mechanical analysis. Instead of computing an explicit place vocabulary for the kinematic chain, our implementation provides the necessary functions for composing the places on the fly.

An ongoing research project is investigating the complete qualitative kinematic analysis of mechanisms based on the place vocabularies for kinematic pairs we construct in this work. We refer the reader to ([NIELSEN]) for more details of this research.

The input to the program is provided by drawing the objects using a simple graphic editor, and the same routines can also be used to display sample configurations in the places. The place vocabularies computed by the program are written to a file for use by the qualitative analysis programs. There also exist facilities to inspect the configuration space and the place graph directly.

Beyond the qualitative analysis of a given mechanism, our algorithms also allow us to determine the complete list of place vocabularies that can be achieved by varying a certain parameter P. This is discussed in detail in the next chapter; we show here how it is actually implemented. The computation involves 2 stages: first, the complete set of landmark values is found, and then the place vocabularies for each case are determined separately. The set of landmark values contains 2 parts:

- an initial set, defined as the values of P where the predicates for the existence of touchpoints change.
- a set of landmark values for the subsumptions of the constraints, which is defined only for a given set of active constraints and touchpoints.

Our implementation allows us to instantiate constraints and touchpoints with symbolically specified parameters. During this instantiation, the initial set of landmark values is found from the predicates for the existence of touchpoints. All possible landmark values for subsumption are tested for inclusion in the intervals where the elements that define them actually exist. All values for which inclusion is found are added to the list of landmark values. For each interval between these landmark values, we pick a

representative point and evaluate the place vocabulary for it. Each of the resulting place vocabularies is then representative for a certain interval of the parameter. However, as our implementation does not determine landmark values of the ordering of intersections, we must test each result for whether this ordering changes between the limits of the intervals. If such a change is found, the ambiguity is reported. An example of the application of this procedure is given later in this report.

Chapter 5

QUALITATIVE KINEMATICS

In this chapter, we discuss the requirements for a theory of qualitative kinematics and show how our place vocabulary theory satisfies them. First, we place the work in the context of qualitative physics, and then we go on to examine its relationship with computer vision theories.

5.1 Kinematics and Qualitative Reasoning

The place vocabulary representation itself is designed to serve as the spatial substrate for a qualitative kinematic analysis of mechanisms. It describes kinematic interactions in a manner compatible with the paradigm of qualitative reasoning, and its use for qualitative analysis of mechanisms is shown elsewhere ([NIELSEN]). Beyond this, the algorithms to compute place vocabularies themselves are based on the qualitative reasoning paradigm. In particular, we emphasize the distinction, common to all work in qualitative reasoning, of the input information into 2 parts:

- a *symbolic* part, in this case specifying the parts of the mechanism, the shape (algebraic form) of their boundary curves, the way in which they are arranged, etc. The symbolic information defines a parameter space, the *metric space*, of all the continuous parameters associated with the physical situation described.
- a *metric* part, which specifies the actual values of the parameters of the metric space.

In a *pure* qualitative physics, no information about the metric part is assumed. All solutions that are possible given proper choice of these parameters are produced. We define a *partial qualitative physics* as one where the some of the metric parameters are known or confined to some interval, and can thus be tested to resolve ambiguous cases.

As we have seen, the symbolic part alone completely defines a finite set of possible configuration space constraints for each kinematic pair. The subset of these that is actually valid and the arrangement of these into place boundaries is determined by the metric part. As the number of possible constraints is finite, there are only a finite number of possible active subsets of them. For each set of active constraints, there are only a finite number of ways in which they can intersect and form place vocabularies. This implies the important result that *the symbolic part alone defines a finite set of possible place vocabularies.*

Because the set of possible place vocabularies is finite, there exists a set of predicates on the metric information, defined by the symbolic information, which is sufficient to decide which is the correct one. Conceptually, the computation of the place vocabulary is thus split into 2 processes: a symbolic computation which works from the symbolic part of the input information, and the evaluation of certain predicates on the metric information.

The set of curves where the values of these predicates change, which we call *landmark curves*, divides the parameter space into regions within which the place vocabularies do not change. An initial set of these curves is defined by the symbolic information alone. There remain certain tests that do not become defined until a certain choice of metric parameters is assumed. The complete set of landmark curves can be found by an iterative procedure described below.

This implies first of all that a purely symbolic kinematics is indeed possible, at least for the restricted case we have analyzed here. However, the number of ambiguities that arise is so enormous that such a theory could hardly be called practical or psychologically plausible. It is feasible, however, to implement a partial qualitative kinematics, where one or several of the metric parameters are unknown. In the following, we examine this possibility and show how it can be applied to practical problems.

5.1.1 Determining the set of landmark curves

As the metric information is used only through the evaluation of predicates, changes in the metric information influence the resulting place vocabulary only if they change the outcome of a predicate evaluation. If we consider changing a metric parameter, for each predicate there exists some threshold for the parameter where the value of the predicate changes. We call these thresholds *landmark values.* It is useful to make a distinction between the cases of considering only a single parameter and of considering several. In the case of a single unknown parameter, there exist landmark *values* which define intervals of possible values. In the case of several such parameters, we are dealing with landmark *curves* which divide the parameter space into *landmark regions.* It is a nontrivial problem to construct these regions given the curves, but it can be solved in polynomial time using the algebraic cell decomposition techniques described earlier. In our implementation, we have restricted ourselves to the case of a single open parameter, so that only interval tests need to be considered. In the following discussion, we shall refer to both cases as landmark curves.

The set of predicates determining the existence of touchpoints is defined during the instantiation and gives an initial set of landmark curves. By choosing a *representative point* in each landmark region, we can generate an initial set of possible place vocabularies. There remain the tests for ordering of constraints at touchpoints, for subsumption intersections, and for the ordering of intersections, which are defined only when an actual set of active constraints is given. The landmark curves defined by these are valid only within the region where the touchpoints and constraints that define them are actually active; they thus yield new landmark values only if they intersect that region. The second set of landmark curves is thus found by intersecting the candidate curves generated by all possible remaining tests with the intervals where the elements involved in the tests are active.

Recall that our implementation does not construct closed form predicates for the ordering of intersections. Instead of computing landmark values for these, we test only whether the ordering changes between the limits of each interval to detect the presence of further landmark values. Should such values exist, the result is further ambiguous.

It remains to be shown that the set of landmark curves is finite, so that the procedures will terminate. For any given touchpoint, as the set of constraints that meet at the touchpoint is finite, the set of predicates expressing the ordering of the constraints around the touchpoint is also finite. For

a given set of active constraints, the set of landmark curves for the subsumption intersections, defined by the condition that the endpoint of one constraint falls on another, contains 2 tests for each pair of constraints and is thus finite. The ordering of intersecting constraints changes whenever 3 constraints intersect in one common point. The possible set of landmark curves defined (using algebraic decision methods) for this is also finite. Thus, for a given set of active constraints and touchpoints, there can be only a finite number of landmark curves. Hence the procedure will eventually converge. Note, however, the numbers involved: if the number of active constraints/touchpoints is $O(n)$, then the number of predicates for the ordering at touchpoints is $O(n)$, the number of predicates for the subsumption intersections is $O(n^2)$, and the number of predicates for the ordering is $O(n^3)$. The number of landmark regions thus grows very quickly as the number of constraints increases.

5.1.2 Applications of a partial qualitative kinematics

There are many potential applications of a theory such as we have developed. The same advantages that make qualitative physics useful also apply to qualitative kinematics. These are

- No explicit representation of metric information is needed, and thus no explicit representation of uncertainty is required either.
- The reasoning is *complete* within the model: every possible behavior admitted by the mathematical model is found.
- The representation is symbolic: AI techniques of symbolic reasoning can be applied to solve kinematic problems.

The last point in particular is of importance: so far, the representational problems involved in spatial and geometric reasoning had made it intractable for AI. While our implementation of qualitative kinematics only covers a tiny subproblem of general spatial reasoning, the place vocabulary theory is perfectly general. It defines a symbolic representation of kinematics that is rich enough to allow us to apply AI techniques to this domain.

But the other points have important merits as well. The completeness of the reasoning makes it possible to use a qualitative analysis for verification of designs. There may exist possible kinematic behaviors that a designer misses. For example, it is possible to arrange a pair of gears such that they

lock up when turned in a certain direction, but not when turned in the other. This fact had been unknown to the author until it actually was pointed out by the place vocabulary computation.

Because we do not need to represent the metric information explicitly, the effects of uncertainties about these values can be cleanly captured. If the value of some predicate evaluation becomes uncertain, it simply results in an ambiguity in the resulting place vocabulary. There are no certainty factors or other ill-defined notions that hide the direct influence of uncertainty on the result.

The division of parameter space into landmark regions allows us to make a complete list of all possible place vocabularies that can be achieved by varying the parameters. We can use this to solve problems of design: the list of all possible place vocabularies defines a set of possible place vocabularies from which we can pick one that satisfies the design specifications. If no such place vocabulary exists, the design problem has no solution. Note that in general, the number of possible place vocabularies will be too large to make a complete list of them. Instead, the search must proceed incrementally, possibly using heuristics to pick a parameter to vary past the next landmark value. But such a theory is beyond the scope of the current research.

In the next chapter, we show an example of the application of the ideas we have given above. Using our implemented program, we have generated all the possible place vocabularies that can be achieved by varying the distance between the wheel and the lever of a ratchet.

5.2 Qualitative kinematics and computer vision

In this section, we investigate the implications that our research has for theories of computer vision, as well as the constraints that computer vision might impose on qualitative kinematics. We put the algorithms we have developed earlier in this paper in the context of 2 vision theories: the curvature primal sketch idea developed by Brady and Asada ([BRADY86]), and the theory of visual routines of S. Ullman ([ULL84]).

5.2.1 Instantiation from the curvature primal sketch

The representation of the objects we assumed in the metric diagram is very close to that of the curvature primal sketch. This makes it plausible to obtain the symbolic part of the input representation

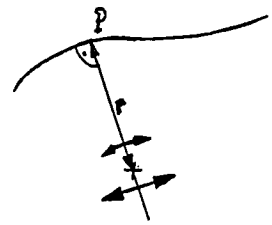

Figure 5.1: Any point P can be made extremum of the radius parameter by proper choice of attachment.

from visual input. However, the instantiation also requires further breaking up the object boundaries to allow parameterization of boundaries and constraint segments. Because the points where the divisions have to be made are defined for single objects alone, it is the responsibility of the symbolic object representation to indicate where they should be placed. [1] It will be sufficient if it can be indicated how many such breakups are possibly required for a boundary segment. This will leave some ambiguity, but that could be resolved in the evaluation phase.

Recall that the additional breakups of the boundary segments must be placed at the extrema of the radius with respect to the attachment of the object. This requires knowledge of the attachment, or at least of possible attachments, of the object. Note that for any point on a smooth boundary curve, there exists an attachment of either type that will require a breakup of the boundary at that point, as shown in Figure 5.1. [2] For rotational attachment, the center of rotation must lie on the normal to the boundary segment at that point. For translational attachment, the direction of translation must be parallel to the boundary at that point. Note that if the boundary segment is a straight line, there will never be another point on the segment that also becomes an extremum for this choice. Thus, for straight line boundaries, at most one extremum is possible.

In the curvature primal sketch, each boundary segment is guaranteed to have a curvature of

[1] We thus leave dead points to be found in the evaluation.

[2] Ignoring singular cases for the moment.

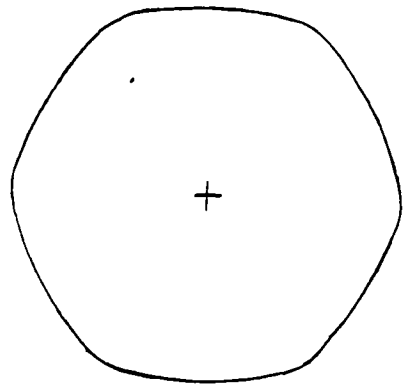

Figure 5.2: This curve has a constant sign curvature, but might have to be broken up several times. Note that it also seems to be conceptually divided by the human visual system.

constant sign. Unfortunately, this does not limit the number of points on the boundary segment where extrema of the radius might occur, as it allows boundary segments such as the one shown in Figure 5.2. If the curve is attached rotationally in the center, we find 12 extrema of the radius parameter. Yet, the curvature is negative throughout. We can construct curves of this type with an arbitrary number of "corners", thus requiring an arbitrary number of divisions. But note that the curve does not appear as one homogeneous segment, but seems conceptually to be broken up at each "corner". It seems plausible that these divisions apply to the primal sketch as well, and therefore the theory of curvature primal sketch should be modified.

The modification of the curvature primal sketch should give all boundary segments the property that the number of extrema of the radius is bounded. Note that it can be limited to a single one only in the case of straight line boundaries. In all other cases, the normals to the boundary segment are not parallel, and any point of intersection of 2 such normals can be chosen as a center of rotation, making the points where they originate extrema of the radius. It would be desirable to impose some further local condition on the curve that would limit the number of extrema to these 2.

The first possibility that comes to mind is to require a constant sign of the *derivative* of the

curvature as well. This would properly split up the curve shown in Figure 5.2, as the curvature in that example has extrema at the "corners". However, there are still examples for which such criteria will fail. A logarithmic spiral has a monotone decreasing curvature, yet it has an unbounded number of extrema with respect to any translational attachment.

The number of possible extrema of a curve, like the number of possible intersections of curves, is determined by the *algebraic* form of the curve. Curvature can not be used to bound the algebraic degree, except when it is identically 0. Therefore, a low-level description based on curvature alone seems insufficient for accurate reasoning.

The problem here could be solved by using assumptions about the possible attachments of objects in the low-level image analysis. Under the assumption of a likely center of rotation or direction of translation, the extrema of the radius and thus the points where the boundary should be broken are straightforward to compute. If this hypothesis is accurate, the accuracy of human kinematic analysis must show strong variation when the attachments of the objects are varied.

There is another hypothesis that can be made. This is that we use the curvature primal sketch as given, but approximate each segment of constant sign curvature by an arc whose curvature is an average of the curvature of the segment. If the arc fits the curve very closely, the reasoning errors that result from this approximation will usually be minor. Given the evidence for the curvature primal sketch, this seems the more likely hypothesis.

While there are certain problems with using the curvature primal sketch as a basis for the symbolic part of the place vocabulary computation, it still matches the requirements very closely. The question of how the deficiencies can be overcome is an interesting new research question.

5.2.2 Evaluation by visual routines

The evaluation phase of the algorithms we have described is based on 2 major operations:

- *indexing*: to pick out a certain configuration of touch of the 2 objects
- *predicate evaluation*: to evaluate a predicate on the objects when they are in an indexed configuration.

This distinction is exactly the same as that made in the theory of visual routines.

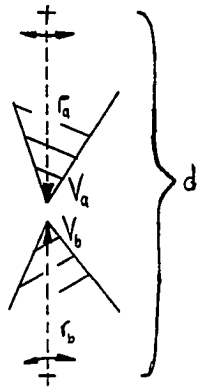

Figure 5.3: The predicates for the existence of a touchpoint can be evaluated by distance comparison with the objects placed in the configuration shown.

To examine what kind of indexing and evaluation operations are required, let us assume for the sake of simplicity that both objects are rotationally attached. To determine if a touchpoint exists, we then first perform an indexing operation, placing the objects in a configuration where the radii to each vertex point in the direction of the other object's center of rotation, as shown in Figure 5.3. With the objects placed in this indexed configuration, the predicates for the existence of the touchpoint can be evaluated by linear distance comparison, an operation that does not involve any algebra. We have seen earlier that these predicates are sufficient to determine the set of active touchpoints and constraints. For the ordering of the constraints around a touchpoint, a similar set of operations applies: first, the touchpoint configuration is indexed, and the ordering of the boundary curves around the point of touch then gives the ordering of the constraints. The existence of dead points can also be tested by indexing the configuration where such a dead point would occur and then testing whether it is a legal point of touch. These evaluations are local: they require local information around a certain point.

Intersections between constraints could be found by pairwise tests as we have described in our algorithms. Note, however, that it is possible to find all the intersecting constraints using only 2 indexing operations: we index the configurations at each end of the constraint segment and find the sets S_1 and S_2 of constraints that are violated in each configuration. Now the set of intersecting

constraints is just the set of constraints violated at one end, but not at the other, i.e., $S_1 \cup S_2 - S_1 \cap S_2$. Note that if there are constraints that are violated at both ends, i.e., $S_1 \cap S_2$ is nonempty, the constraint is subsumed altogether and need not be considered any further. The number of indexing operations we require is then proportional to the number of constraints, which in turn is a good measure of the "complexity" of the objects. The ordering of the constraint intersections could be achieved by the binary search method our program uses. However, this requires a number of tests which grows as the square of the number of constraints. It might be simpler to develop some probabilistic method that indexes random configurations until enough information has been gathered. The predicate evaluations required are now parallel evaluations of point inclusions, local tests as before will not suffice.

Note that this procedure ignores the condition of intersecting bounding rectangles that was discussed earlier and will therefore fail in certain cases. This could provide examples with which the theory can be validated as a theory of human competence: if the errors that the algorithm makes are the same as those that human subjects make, we have a good indication that the theory actually models human performance. However, in practice this case is more subtle because people have experienced common cases of these errors many times and may have developed special heuristics to avoid them.

In order to be plausibly implemented as visual routines, the algorithms for computing place vocabularies must be weakened and specialized to fit the different type of computation allowed by the visual system. This is a promising direction for further research.

Chapter 6

EXAMPLES

In this chapter, we give examples of place vocabularies that have been generated using our implementation of the algorithms. The description of each example will consist of:

- The physical objects and their arrangement
- The configuration space constraints and the breakup of free space
- The place graph describing free space

Whenever the objects involved in the interaction are periodic, we will discuss the periodic version of the place graph computed by the program. This results in a much simpler display than displaying the full graph, but is somewhat harder to understand because of the additional periodic links appearing in the graph.

Besides the fully worked examples, we have also computed place vocabularies for an actually existing clock, the QRG clock. As these place vocabularies are considerably more complicated than the others, we give only abbreviated descriptions of them.

6.1 A Ratchet

We consider the example of the simple ratchet mechanism shown in Figure 6.1. This example has been presented in the introduction and we now discuss it in more detail. For easier reference, we again

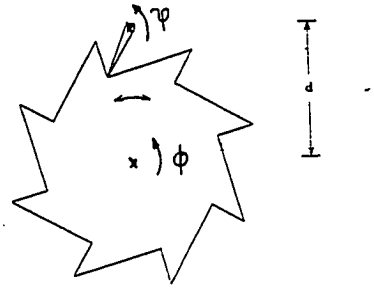

Figure 6.1: Ratchet example.

show examples of possible configurations and the configuration space in Figures 6.2 and 6.3. The resulting place graph for this example is shown in Figure 6.4. First, we shall explain the notation used in the graph. Nodes are labeled as either IP-n, CSEG-n, FD-n or FULL-FACE-n, where n is some integer. IP-n stands for **I**ntersection **P**oints between constraint segments, i.e., the 0-dimensional places or points in configuration space. CSEG-n designates **C**onstraint **Seg**ments, i.e., 1-dimensional places. FD-n stands for **F**ree Space **D**ivision, i.e., the tesselations of free space defined by applicability constraints. Finally, FULL-FACE-n designates regions of configuration space, i.e., the 2-dimensional places.

Adjacencies between the places are indicated by the edges of the graph. The edges are marked to indicate the type of relationship the elements have. The mark "->" means that the place on the left end of the edge bounds the place on the right end of the edge. Note that because constraint segments and intersection points mutually bound each other, they have this mark going both ways. When the adjacent place is a periodically offset copy, this is indicated by marking the link. We use a pair of numbers indicating the number of *vertices* on the boundaries of each of the 2 objects by which the place is offset.

The graph and its relation to the configuration space is best understood by focussing on the 2 dimensional places, of which there are 5. The regions that these places correspond to are indicated in Figure 6.3. After identifying the 5 FULL-FACEs, note that their bounding places are arranged around each of them in a circle. The association of constraint segments to the sample configurations

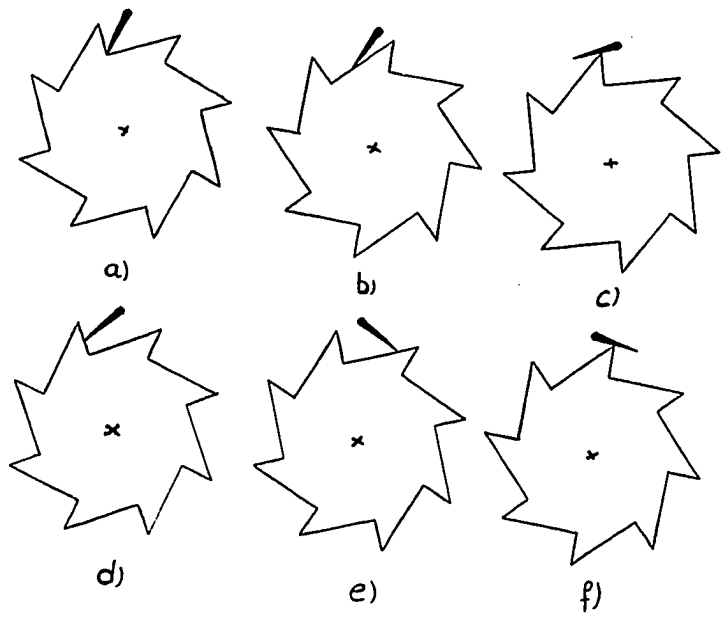

Figure 6.2: Examples of possible configurations for the ratchet.

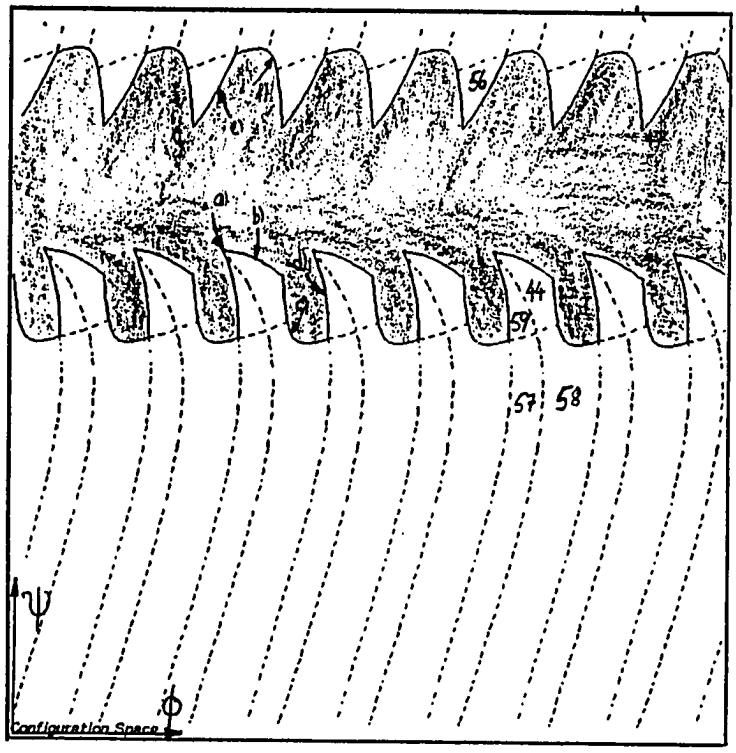

Figure 6.3: The configuration space for the ratchet example. The horizontal parameter is the orientation ϕ of the wheel, the vertical parameter the orientation ψ of the lever.

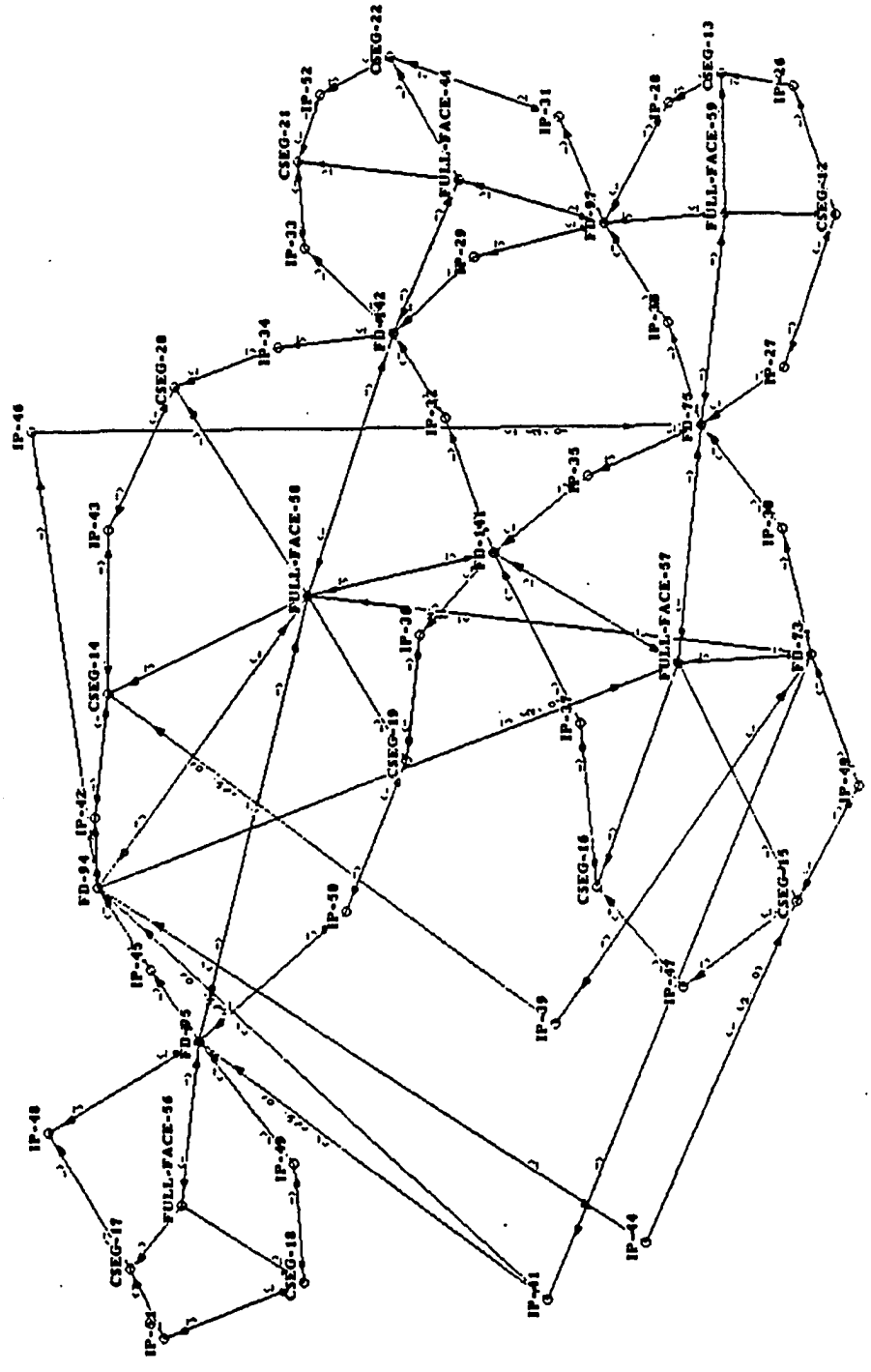

Figure 6.4: The place graph for the ratchet example. Note that an explicit representation is provided for one of the 8 periodic regions only, all others are represented as periodic copies of these.

Place	Touch Configuration
CSEG-12	d)
CSEG-13	d)
CSEG-14	c)
CSEG-15	f)
CSEG-16	f)
CSEG-17	f)
CSEG-18	e)
CSEG-19	f)
CSEG-20	c)
CSEG-21	c)
CSEG-22	b)
IP-28	a)
IP-31	a)

Figure 6.5: The association between constraint segments and configurations of touch.

of touch is shown in the table in Figure 6.5. Note that there are often several constraint segments corresponding to the same configuration of touch. This is for one of the following 3 reasons:

- The constraint changes qualitative directions and must be broken up to satisfy monotonicity.
- The constraint is broken up to satisfy the parameterization requirements. Note that this does not necessarily entail a change in qualitative direction.
- The constraint is broken up because of an intersecting free-space division.

The adjacencies between 2-dimensional places are defined by free-space divisions. In the graph, the FULL-FACEs are thus connected by shared free-space divisions. Note that the intersection of free-space divisions FD-75, FD-97, FD-141 and FD-142 forms a "cycle" by itself. FULL-FACE-57 and FULL-FACE-58 are mutually adjacent to periodic copies of each other. These adjacencies occur at the free-space divisions FD-94 and FD-73, which are periodic copies of each other, offset by 2 vertices.

Thus, all places connected to FD-94 are also connected to FD-73 with a periodic offset, and vice versa. Each of these connections is marked either by an offset (2,0) or (14,0), as these are the mutual offsets of the 2 free-space divisions. Which of the 2 is actually chosen is determined by the direction of the link: if the direction is from FD-94 to FD-73, the offset is (14,0), otherwise it is (2,0). Note that because the links occur via intersection points, there is some ambiguity as to what period this intersection point belongs to. In Figure 6.4, the 4 intersection points at the ends of FD-73/FD-94 are duplicated as IP-39, IP-41, IP-44 and IP-46. Out of these, IP-39 and IP-41 are in the same period with FD-73, and IP-44 and IP-46 are in the same period as FD-94. Therefore, the indices of the periodic repetitions are (14,0) for the first 2 and (2,0) for the last 2. Finally, note that FD-94 is also linked to FULL-FACE-57 by a periodic link, similarly for FD-73 and FULL-FACE-58.

This place graph forms the spatial substrate for an envisionment of the behavior of the ratchet. The links in the graph define all the geometrically possible transitions, many of which are impossible for a given configuration of external forces. In the actual envisionment, many of them can be ruled out based on analysis of the forces.

6.2 Parameter Variation for the Ratchet

In this section, we describe the changes that the place vocabulary undergoes under variation of a paramter. The parameter we vary is the distance between the center of rotation of the wheel and the center of rotation of the lever, and we shall refer to it as d. In the analysis of the previous section, the distance has a value of d=30 units.

6.2.1 Generation of landmark values

The values of the parameter at which the place vocabulary undergoes possible changes are called *Landmark Values*. In the example of the ratchet, there are 8 of them: at 21.05, 23.06, 27.29, 28.3, 29.27, 32, 35.5 and 40.28. When d is smaller than 21.05, there are no legal configurations. When d is greater than 40.28, the objects are too far apart to touch, thus there is no interaction.

As discussed earlier, landmark values arise from the following conditions:

- For each touchpoint: the 2 conditions for its existence, and the predicates determining the

relative ordering of the constraints around it.

- For each pair of constraints: the condition that an endpoint or midpoint (point where the derivative is zero) falls onto the other constraint (*subsumption* landmark values).

The actual landmark values are defined by an equation $\mathcal{F}(d) = 0$, where $\mathcal{F}(d)$ defines the condition as an equation, written in terms of the open parameter d. The equation is obtained in the following ways:

- For predicates determining existence of touchpoints, the predicate itself is defined as an inequality which is readily converted into an equation.
- For the relative ordering of 2 constraints around a touchpoint, the predicate is defined by the condition that the 2 constraints have the same tangent direction. This is also readily expressed as an equation.
- For subsumption landmark values: substituting the coordinates of the endpoint, written in terms of the open parameter, in the equation of the other constraint gives the equation for the landmark value.

For many of these landmark values, it is possible to restrict the range of validity. For example, the landmark values defined by the conditions involving the endpoints of constraints are valid only in the interval where these endpoints actually exist. Even within this range of validity, in most cases a possible landmark value results in a change in the structure of the constraints in blocked space only. It does not affect the boundaries of free space and therefore is not an actual landmark value. In the current implementation, this fact is detected only in the final result of the place vocabulary computation. It would be conceivable to construct an explicit dependency structure of landmark values on other predicates to predict irrelevant landmark values more directly, but this has not been implemented.

In the example given here, the numerical values of the landmark values were found by binary search using the defining equation. The initial interval was defined either by other landmark values governing the existence of touchpoints or endpoints or by an a priori interval which was set to [1..50]. Binary search is a very crude method for finding roots of such equations, and it turns out that not all landmark values are found by this procedure. The procedure failed to produce the value at d=23.06,

which is not a minor omission. For a proper automatic analysis, reliable methods for finding roots (such as using Sturm sequences) will be required. In this simple example, missing landmark values were found by hand-calculation and verification using the corresponding defining equations.

6.2.2 Representative place vocabularies between the landmark values

To illustrate the major changes that the place vocabulary undergoes when r is varied, we show place vocabularies for 3 different cases: d=22, d=25 and d=34. We have already investigated the place vocabulary for d=30 in the previous section. This leaves the intervals from 27.29 to 28.3 to 29.27 and from 35.5 to 40.28 uncovered, but the variations in the place vocabulary that occur at these landmark values are only further breakups of constraint segments corresponding to the same configuration of touch. These are introduced to satisfy requirements of changes in qualitative direction or unique parameterization. We do not examine these variations separately here.

When d=22, the configuration space contains eight periodic regions of free space as shown in Figure 6.6. Each region is bounded by eight different constraint segments, as indicated by the numbers in Figure 6.6. To get some idea of the configurations that the constraints correspond to, consider the sample configurations in Figure 6.7. Note that again, certain groups of constraint segments share the same configuration of touch. The place graph for this case corresponds directly to the configuration space arrangement and is shown in Figure 6.8.

When d becomes greater than the landmark value at d=23.06, it becomes possible for the tip of the lever to touch the tip of the teeth of the wheel. This results in 2 simultaneous changes in the place vocabulary: The appearance of a strip extending upwards from the tip of each region, and a vertex in the bottom boundary of each region. These vertices require the addition of free-space divisions. The resulting configuration space for d=25 and its tesselation is shown in Figure 6.9. Again, the numbers in Figure 6.9 indicate the correspondences to the place graph shown in Figure 6.10. To aid in understanding, sample configurations corresponding to the new constraint segments are shown in Figure 6.11.

When d is increased further, direction changes appear in the 2 vertical boundaries of FULL-FACE-85 (in Figure 6.9) at d=27.29 and d=28.3. These direction changes are very small and do not show within the resolution of the graphic display, so the configuration space for these intervals looks

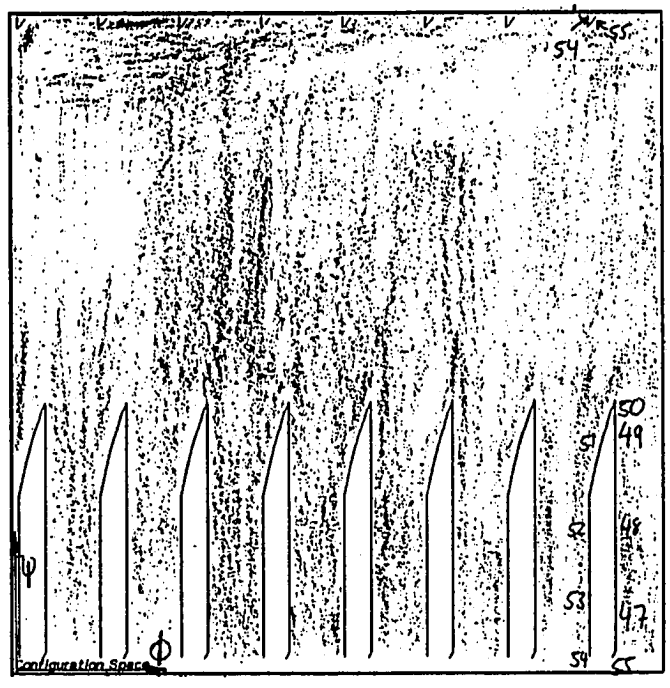

Figure 6.6: The configuration space for the ratchet example when d=22.

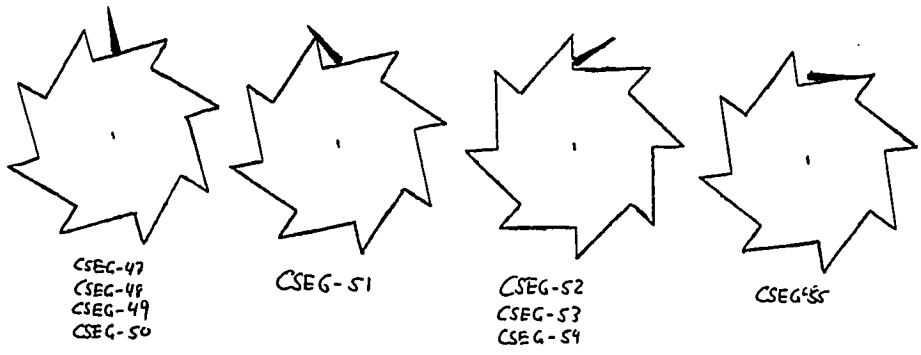

Figure 6.7: Examples of configurations satisfying the various constraint segments.

qualitatively the same as that for d=25. When the landmark value at d=29.27 is crossed, the 8 disjoint regions become connected and the wheel is free to turn. We have already examined a representative place vocabulary for this interval in the previous section.

When d increases beyond d=32, the touchpoint between the tip of the lever and the vertex at the "bottom" of the teeth disappears. The lever is now free to rotate a full rotation, and blocked space consists of 8 disjoint "obstacles." The configuration space and its tesselation for the representative value of d=34 is shown in Figure 6.12. The numbers in Figure 6.12 again indicate the correspondence to the place graph, which is shown in Figure 6.13. Note that FULL-FACE-130 is connected to periodic copies of itself at the free-space divisions FD-252 and FD-249. The links are made in the same way as those in the example for d=30, given in the last section. Sample configurations for the places are shown in Figure 6.14 to aid in comprehension of the example. Note in particular the direction change from CSEG-63 to CSEG-64, both of which share the same configuration of touch.

When d is increased further beyond the landmark value at d=35.5, the direction change between CSEG-63 and CSEG-64 disappears, and we obtain the configuration space tesselation shown in Figure 6.15 for the representative case of d=37.

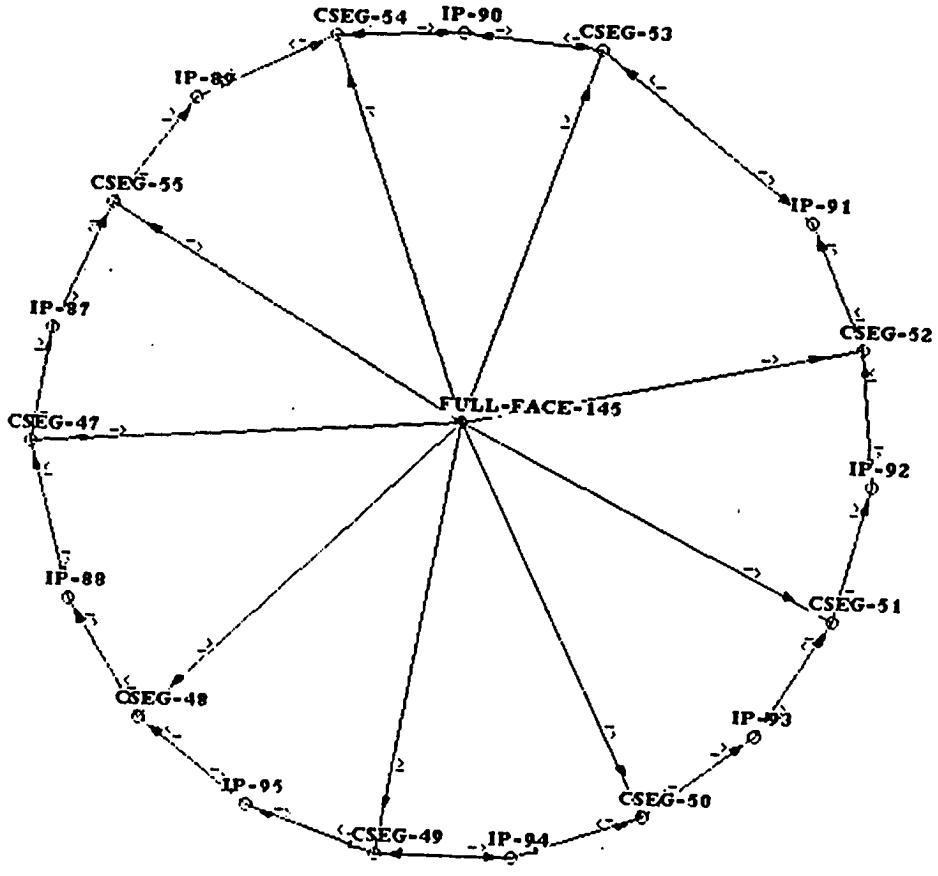

Figure 6.8: One period of the place graph for the ratchet when d=22.

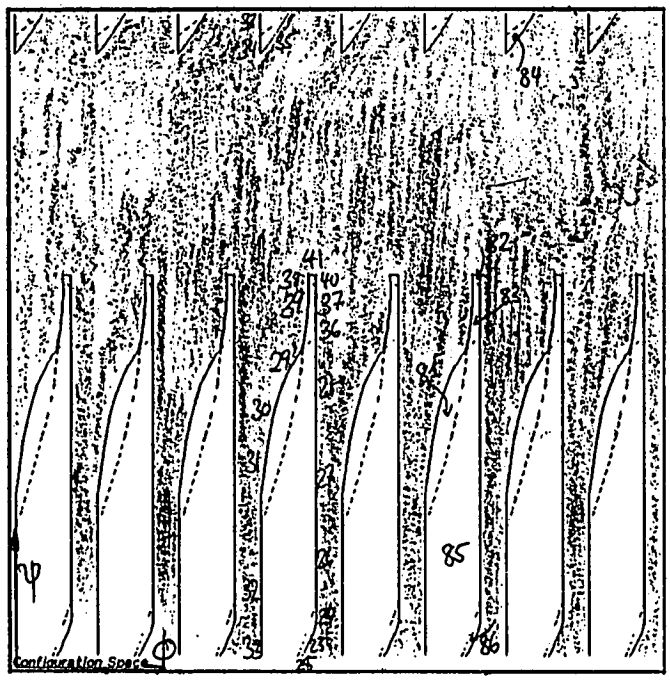

Figure 6.9: The configuration space for the ratchet example for d=25.

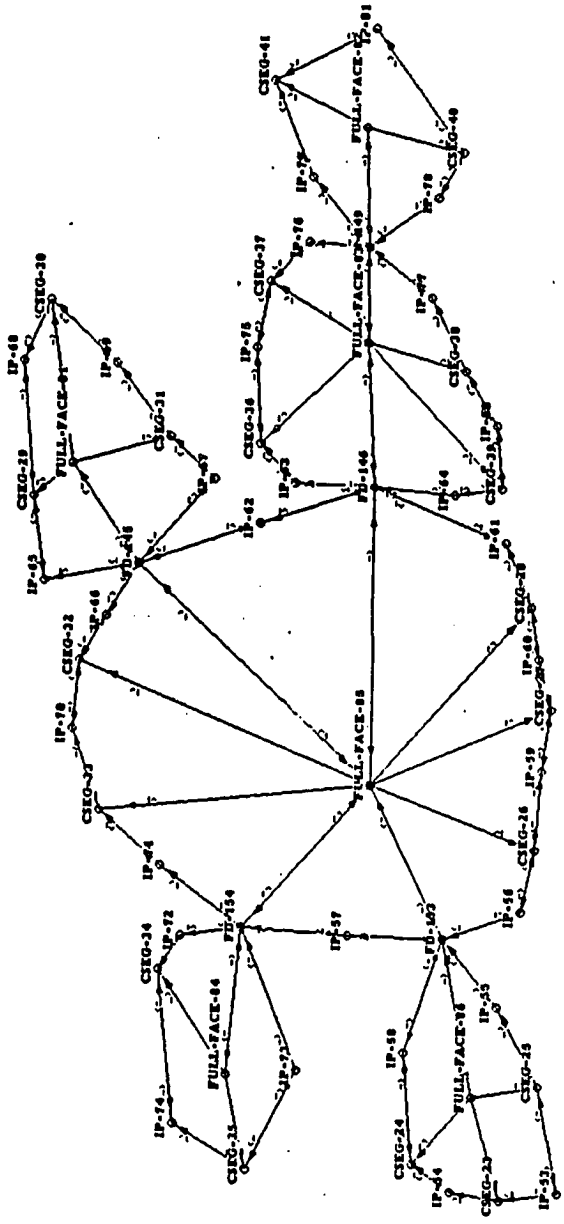

Figure 6.10: The place graph for the ratchet example when d=25.

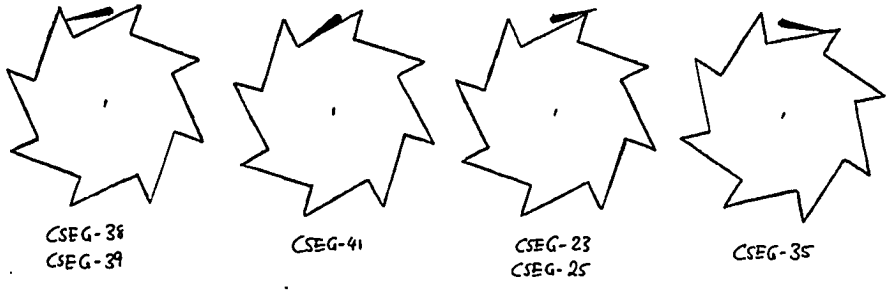

Figure 6.11: Sample configurations for some of the constraint segments in the ratchet place vocabulary for d=25.

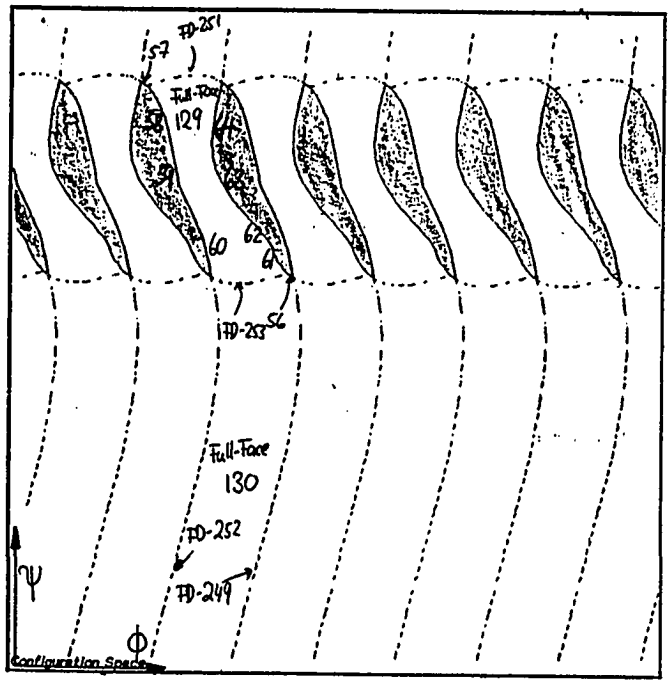

Figure 6.12: The configuration space for the ratchet example when d=34.

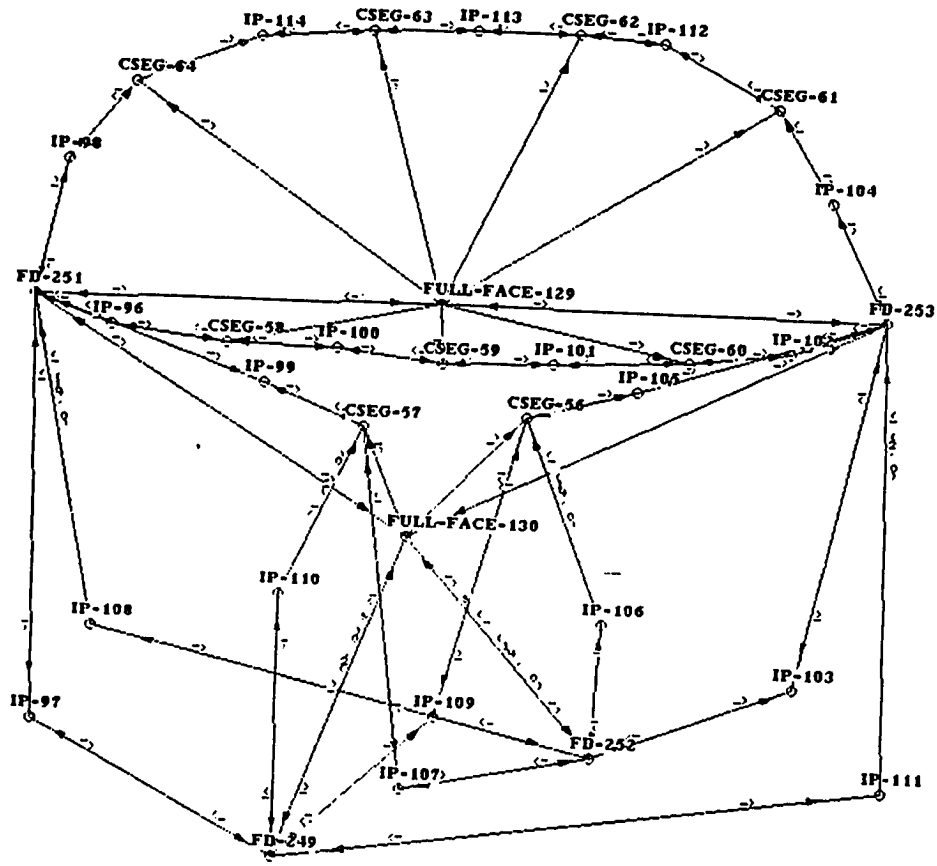

Figure 6.13: The place graph for the ratchet example when d=34.

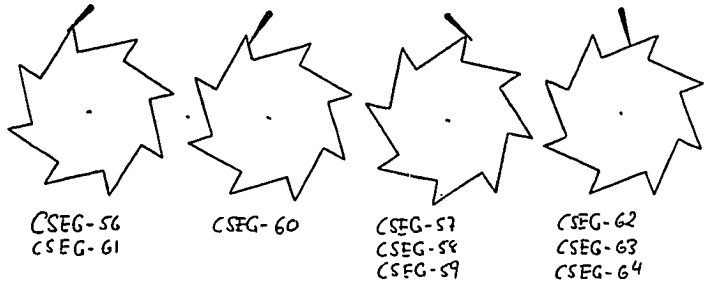

Figure 6.14: Sample configurations for some of the constraint segments in the ratchet place vocabulary for d=34.

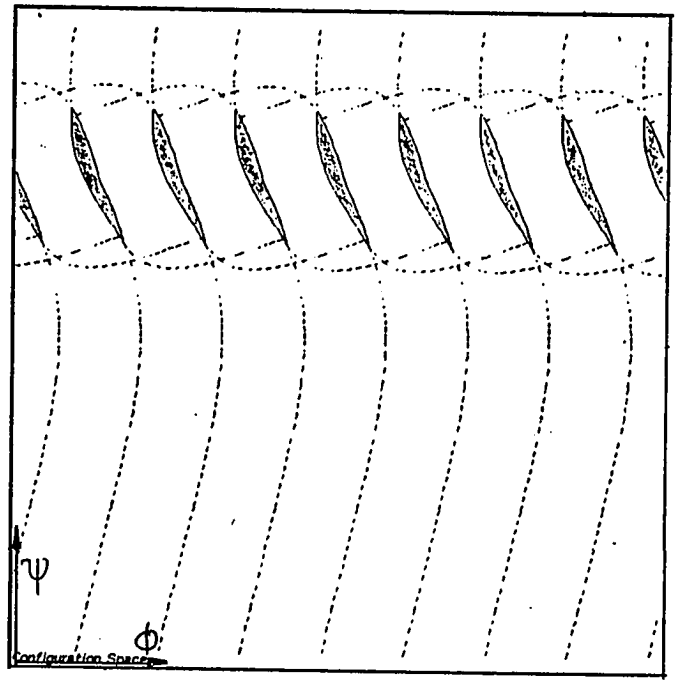

Figure 6.15: The configuration space for the ratchet example when d=37.

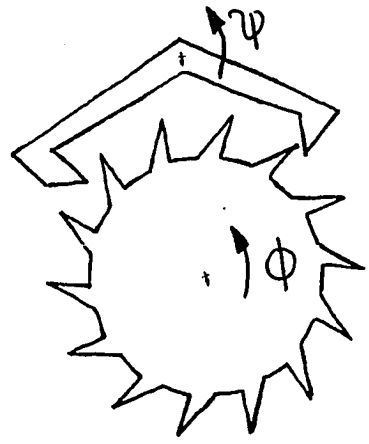

Figure 6.16: Recoil escapement example

We have thus found all possible place vocabularies that can be achieved by varying the distance between the center of rotation of the wheel and the center of rotation of the lever. Note that we do not take into account the structure of the tesselation imposed by the free-space divisions. This is because this tesselation does not influence the mechanical behavior of the mechanism itself, but is added only in order to satisfy the quasi-convexity criterion for the 2-dimensional places. While it appears possible to compute landmark values for changes in the tesselation as well, they do not reflect changes in actual behavior and we thus do not consider them necessary.

6.3 Three Clock Escapements

In this section, we discuss the place vocabularies for 3 different types of escapements: a recoil escapement, a deadbeat escapement, and a cylinder escapement.

6.3.1 Recoil escapement

Consider the recoil escapement shown in Fig. 6.16. The escapement consists of 2 parts, the *escape*

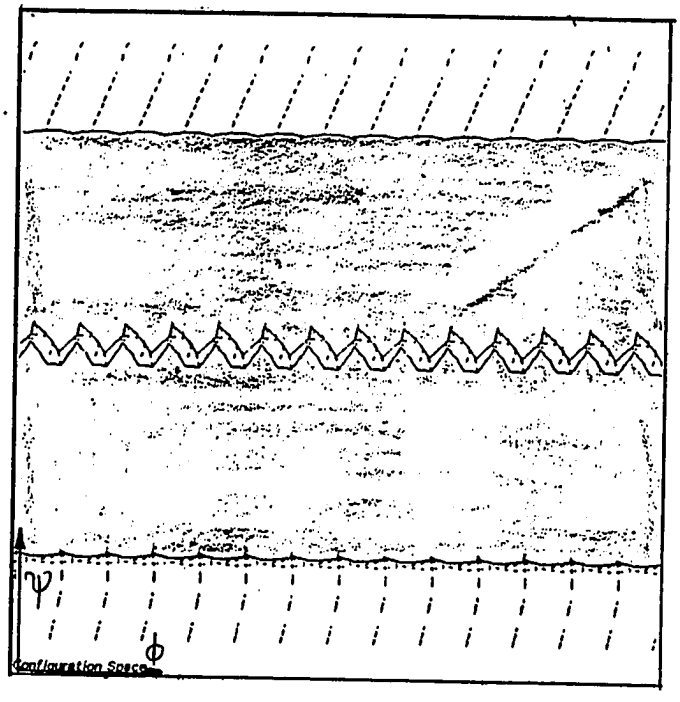

Figure 6.17: The configuration space for the recoil escapement example. Note that the space is a torus surface and thus wraps around on the sides. The horizontal direction is ϕ, the vertical ψ.

wheel and the *anchor*, both hinged with rotational freedom. It is the central element of many pendulum clocks. Attached to the anchor is a pendulum which oscillates with a constant period. There is a driving moment acting on the wheel that turns it clockwise. Each time the pendulum swings, the escapement allows the wheel to turn by one tooth. As the period of the pendulum is constant, so are the intervals between successive advances of the wheel.

The configuration space for this example is shown in Figure 6.17, where the horizontal direction is the orientation angle ϕ of the escape wheel, and the vertical is the orientation angle ψ of the anchor. Note that there are 2 disjoint regions of free space: one through the center, corresponding to the normal operation of the ratchet, and one that wraps around from the top to the bottom, which corresponds to the anchor being inverted. We discuss the region corresponding to the normal operation of the

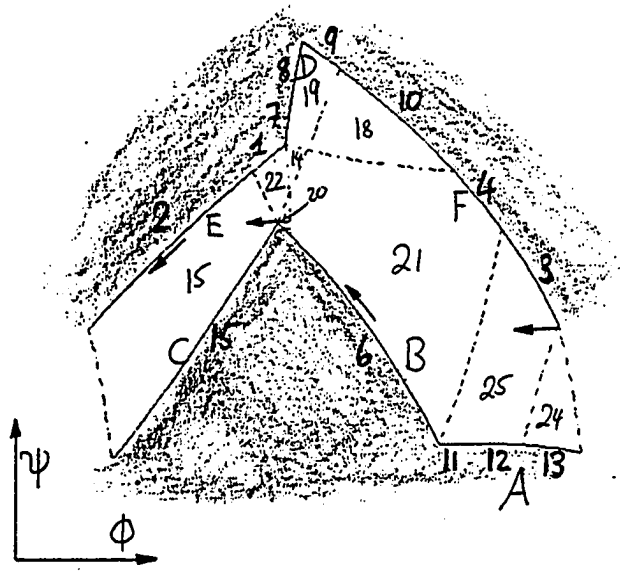

Figure 6.18: An enlarged section of the configuration space. The letters refer to the configurations shown in the respective figure, the arrows indicate motion of escapement during normal operation.

escapement. Figure 6.18 shows an enlarged section of this part of the configuration space. The region is bounded by sequences of constraint segments corresponding to the right side and the left side of the anchor touching the wheel. These form the 1-dimensional places: the set of configurations where there exists a point of contact between a certain vertex and a certain boundary segment of the object. For each tooth, the program finds 1-dimensional places corresponding to the 6 different configurations of touch shown in Figure 6.19. The constraint segments corresponding to these are indicated by letters in Figure 6.18. In cases A), C) and D), there exists a point where the qualitative relation between the configuration space parameters changes; they are thus further subdivided by the program. Note that in the actual place graph, these places are further broken up by intersecting free-space divisions. The places shown in each row of Figure 6.19 are connected in a sequence as indicated by the arrows. There are 14 teeth on the wheel, so there are 14 periodic repetitions of the places shown. In the actual operation of the escapement, the wheel is moving clockwise and the anchor alternatively touches the wheel with its right and its left end. This motion is indicated by the arrows in Figure 6.18. The sequence of places it passes through is then A) → B) → intermittent motion → E) → intermittent

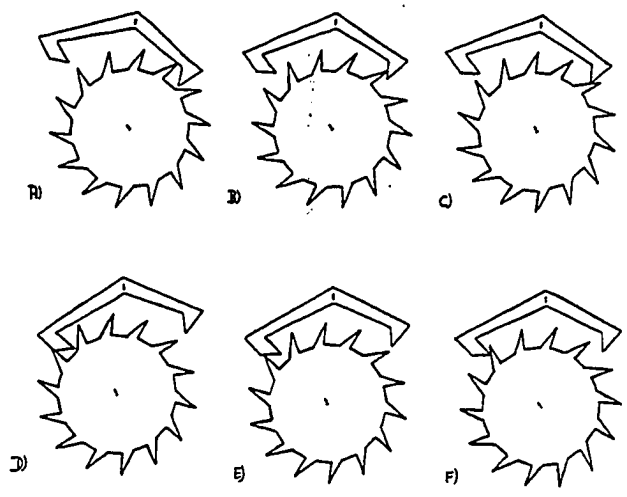

Figure 6.19: Sample configurations for the 1-dimensional places in the escapement example.

motion → A)' ..., where A)' refers to the next periodic repetition of A). The intermittent motions, where no contact between the objects exists, are represented by 2-dimensional places. .

The place graph for the escapement example is shown in Figure 6.20. Similar to the case of the ratchet discussed earlier, the place graph contains two 2-dimensional places, FULL-FACE-15 and FULL-FACE-24, which are adjacent to periodic copies of each other, and this is again indicated by links marked for periodic offset.

We outline how an envisionment of the escapement's operation can be obtained using the place graph. For each place, the set of possible transitions is given as the set of adjacent places. We describe the velocities and forces or moments of the objects by their signs. Note that the 2 quantities are related by a simple qualitative differential equation. Each of the one-dimensional places enforces a monotone relation between the configuration space parameters. This results in a constant relation between the qualitative parameters for the motion and forces within these places. The 0-dimensional places are contained in 1-dimensional places and thus share these characteristics. For the 2-dimensional places, there is no contact between the objects and they therefore do not influence each other. Details of the actual qualitative simulation are described in ([FNF87]).

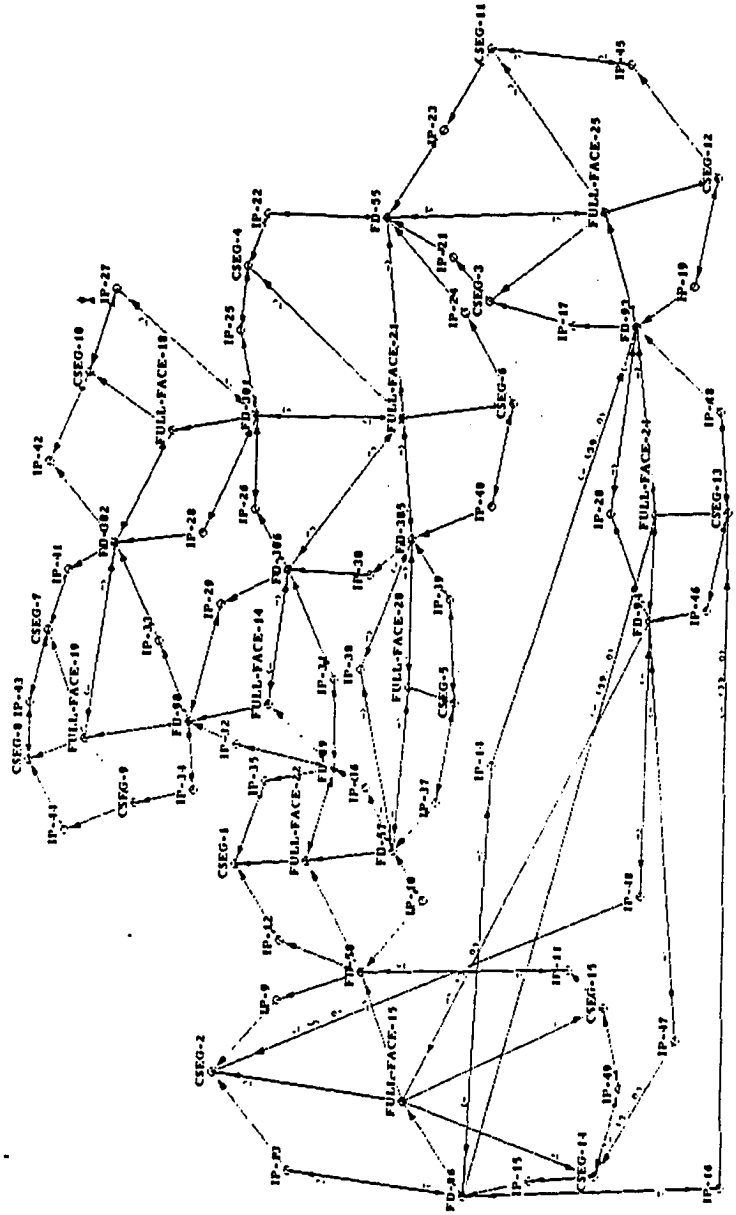

Figure 6.20: The place graph for the recoil escapement.

Figure 6.21: A deadbeat escapement.

As a final point, note that we have not described the other region of free space. This region is also broken up into subregions by free-space divisions and forms a disjoint component of the place graph.

6.3.2 Deadbeat escapement

The escapement we have discussed in the previous section has an undesirable behavior. In the normal operation of the escapement, consider the intermittent motion from configuration b) to configuration e). At the moment where the contact in configuration e) is made, the pallet is still rotating counter-clockwise. But in order to continue the motion of the mechanism along CSEG-2, this motion has to be reversed. The pendulum is thus driving the clock backwards, an undesirable behavior.

This situation is corrected in the *deadbeat* escapement shown in Figure 6.21. The boundaries where the escape wheel comes into contact with the pallet following intermittent motion are now curved with the center of curvature equal to the center of rotation of the pallet. When the escape wheel touches these surfaces, motion of the pallet will not affect it. The pallet thus no longer drives the clock backwards. The difference between the 2 escapements can also be explained using the configuration space, which is shown in full in Figure 6.22 and in an enlarged version in 6.23. Again, we have indicated the motion during normal operation of the escapement by arrows in Figure 6.23. Note now that the 2 intermittent motions in the duty cycle both end in places whose derivative with respect to the motion of the pallet is zero, namely CSEG-2 and CSEG-10. Sample configurations for some of the places are shown in Figure 6.24. As the device is very similar to the ordinary escapement

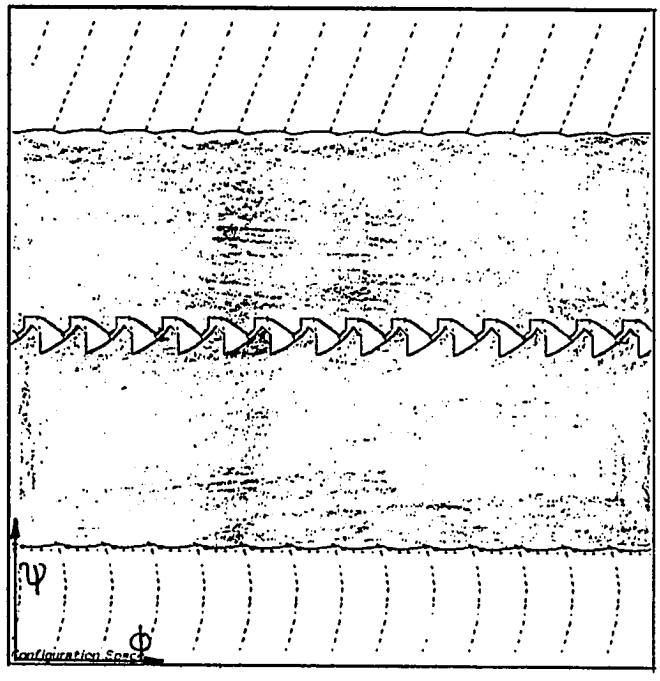

Figure 6.22: The full configuration space for the deadbeat escapement.

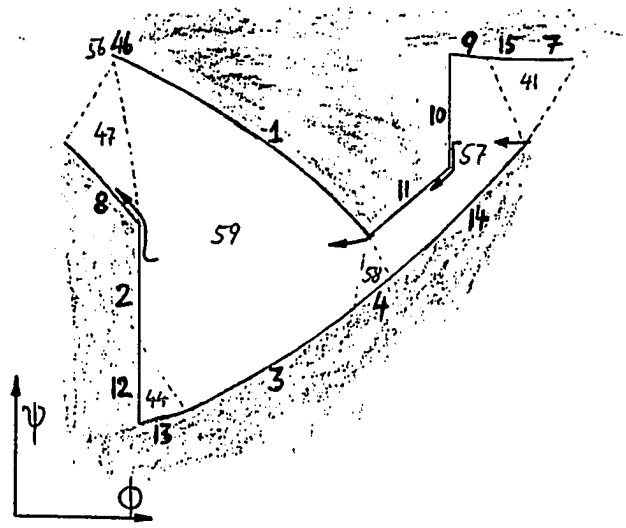

Figure 6.23: An enlarged section of the configuration space for the deadbeat escapement. The numbers refer to constraint segments in the place graph.

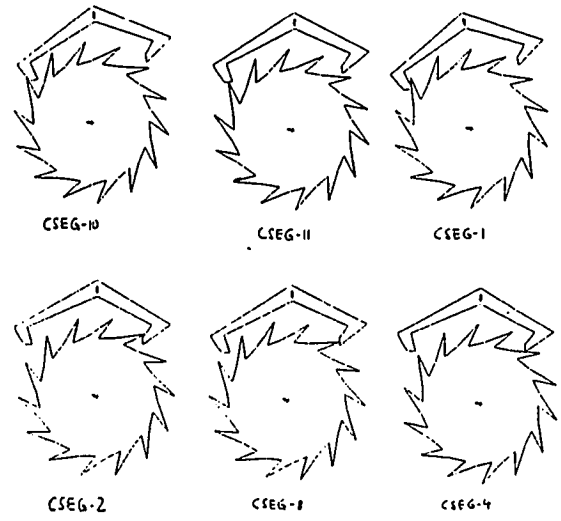

Figure 6.24: Sample configurations in some of the places for the deadbeat escapement.

shown in the previous section, so are the configuration space and the place graph, shown in Figure 6.25. The correspondences between the configuration space and the place graph are indicated by the numbers is Figure 6.23. Figure 6.23 should be compared with Figure 6.18 to better understand the difference between the 2 types of escapements.

6.3.3 Cylinder escapement

Escapements of the pallet type we have seen are rather brittle devices. The distance between the centers of rotation of the wheel and the pallet, for example, has to be very precisely adjusted for the escapement to operate properly. A cylinder escapement, shown in Figure 6.26, is a much less problematic design. It consists of 2 parts: a cut-out *cylinder*, on the left, which rotates around its center, and a spoked wheel. Note that of the first part, only the hollow cylinder is actually in the same plane as the wheel. In the plane of our analysis, its center of rotation thus lies outside the object, although in some other plane it must be fixed by a joint.

The cylinder is attached to a balance spring and rotates back and forth. The wheel is attached to a spring that drives it in the counterclockwise direction. We refer to the sample configurations shown in Figure 6.27. Starting in CSEG-18, with the cylinder turning clockwise, we arrive at CSEG-17. The spoke of the wheel is now released, and it moves intermittently in the counterclockwise direction. The next spoke now hits the other side of the cylinder, as in CSEG-15. The cylinder swings to its extremal point and reverses its direction of motion. Now rotating counterclockwise, we reach the configuration in CSEG-3, where the spoke is again released. After intermittent motion, the device reaches CSEG-18. In this place, the cylinder now swings to its extremal point, reverses its motion, and the cycle repeats.

Figure 6.27 also shows some other configurations to illustrate the display of the configuration space, shown in Figure 6.28. The duty cycle of the mechanism is also indicated by arrows in this figure.

The place graph for this example is shown in Figure 6.29. The place graph consists of 6 2-dimensional places for each spoke of the wheel. The correspondences between places and the configuration space are indicated by the numbers is Figure 6.27. Note that FULL-FACE-1 is linked periodically to FULL-FACE-6.

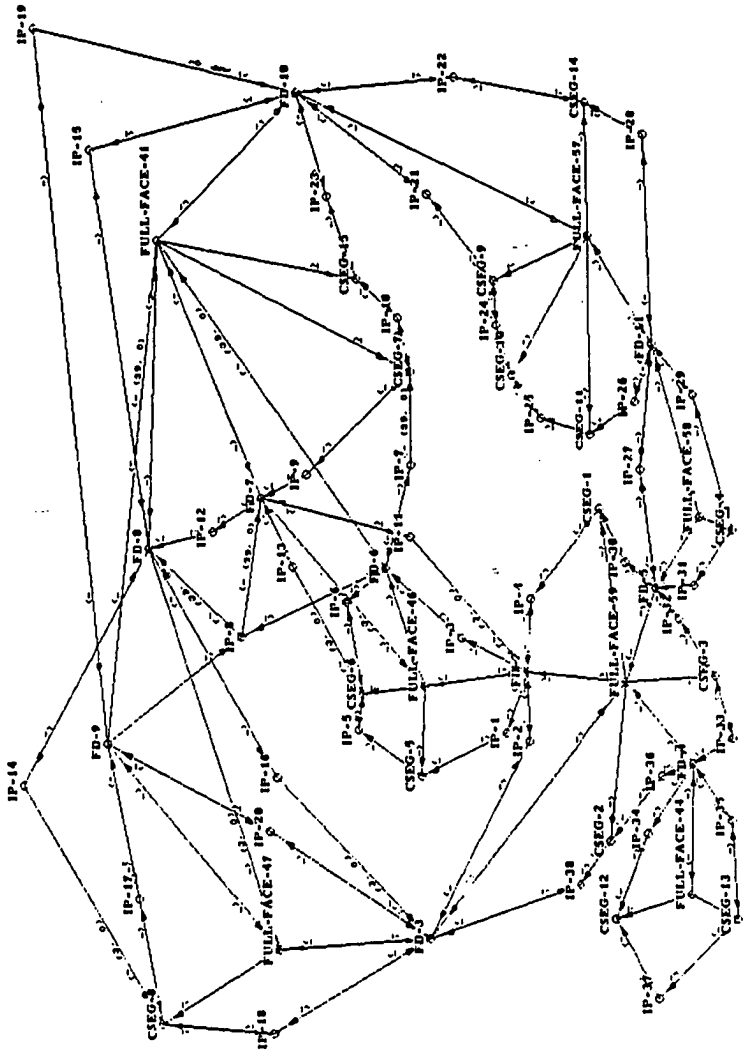

Figure 6.25: The place graph for the deadbeat escapement.

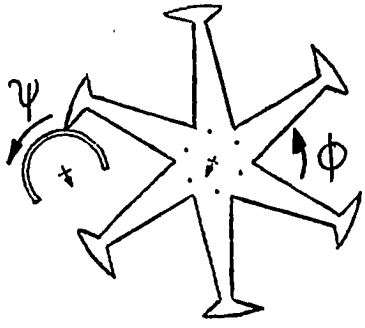

Figure 6.26: A cylinder escapement.

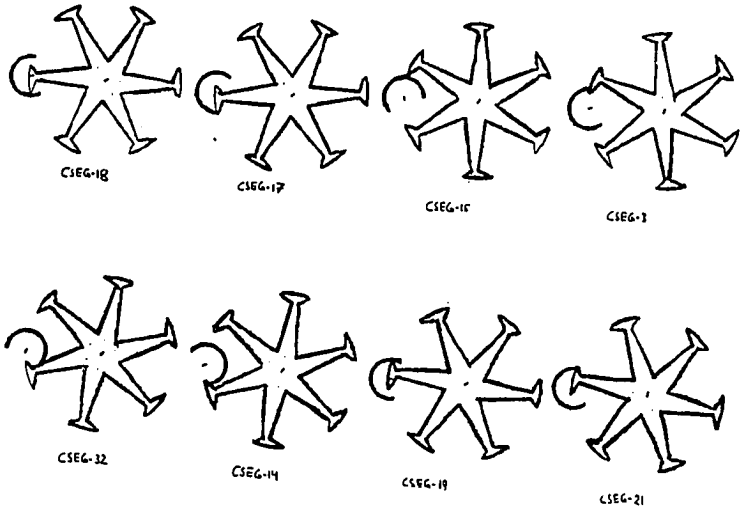

Figure 6.27: Sample configurations of the cylinder escapement.

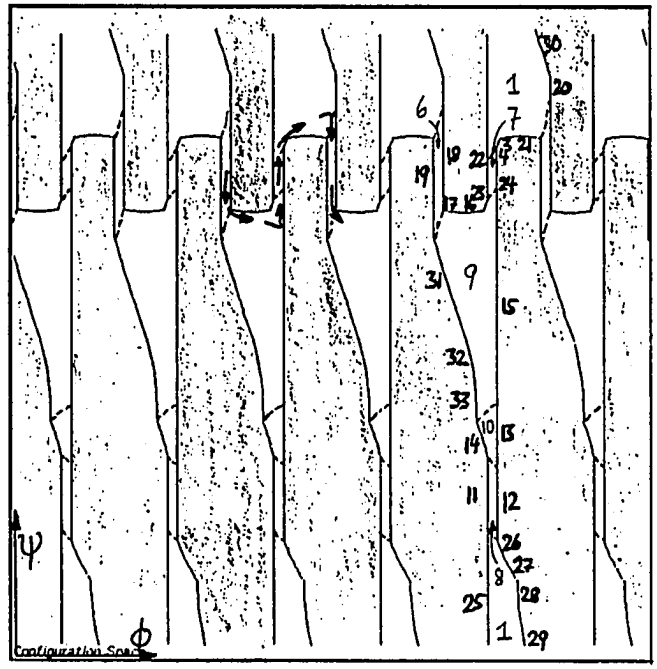

Figure 6.28: The configuration space of the cylinder escapement.

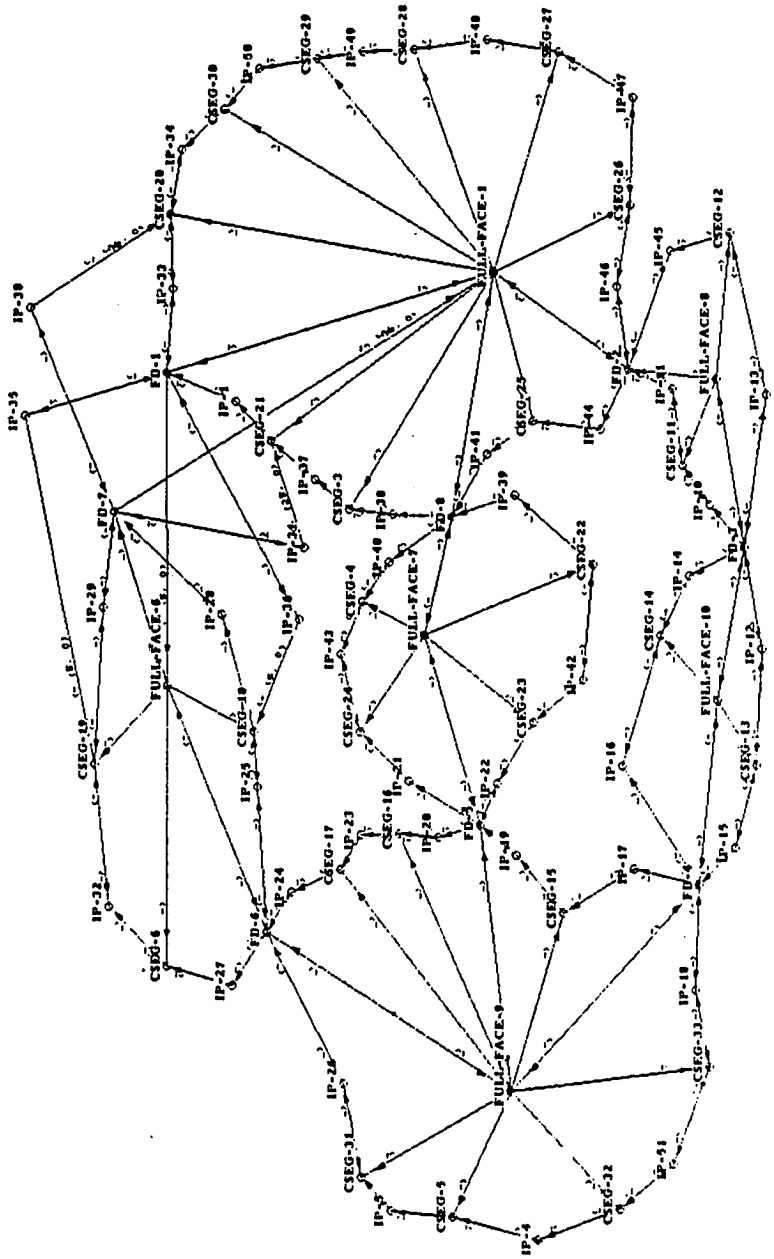

Figure 6.29: The place graph for the cylinder escapement.

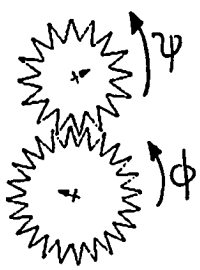

Figure 6.30: A pair of gears.

6.4 Gears

Consider the pair of gears shown in Figure 6.30. The configuration space for this example is again 2-dimensional, the 2 dimensions being made up of the orientation angle ϕ for the lower gear and the angle ψ for the upper gear. The gears in this example have 16 and 24 teeth, respectively. As the greatest common divisor of these is 8, the configuration space contains 8 identical doubly connected "channels," corresponding to different initial choices for pairs of interacting teeth. The complete configuration space is shown on the left in Figure 6.31, and a single channel alone is shown on the right. A detailed display of an enlarged section of configuration space is further presented in Figure 6.32. As the interactions between the teeth are all periodic, so is the place vocabulary. In this case, we have $24 * 16 = 384$ possible choices for a pair of teeth, so there exist 384 periodic copies of each place. The place graph, shown in Figure 6.33, contains 2 2-dimensional places, connected to their periodic copies as shown. To aid in understanding, Figure 6.34 shows sample configurations for the places in the place graph. The numbers of the places are also indicated in Figure 6.32. The place graph contains links that indicate the periodic repetition. At FD-9, the 2-dimensional place FULL-FACE-9 is linked to a periodic copy of FULL-FACE-95, and vice versa at FD-10. The remaining free-space division, FD-8, runs between the 2 places. Note that as the wheels have 16 and 24 teeth, respectively, and each tooth contains 2 vertices, the periodic offsets are either the pair (2,30) or (46,2). Adjacent periodic copies are offset in both parameters, reflecting the fact that successive teeth on one wheel

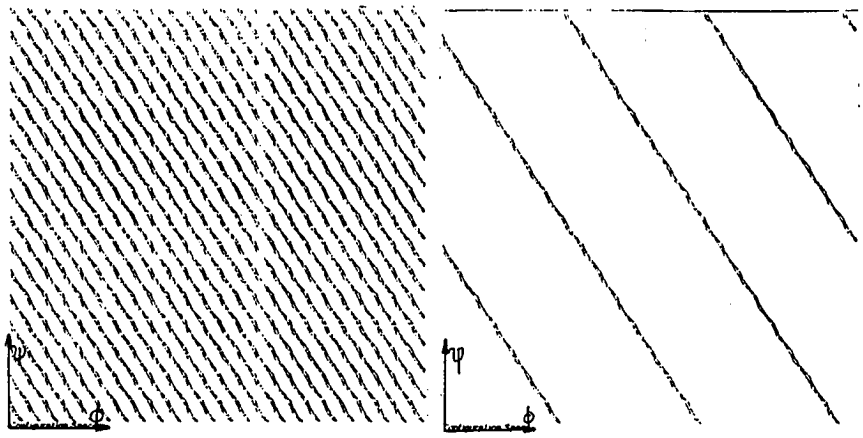

Figure 6.31: The configuration space for the gearwheel example. On the left, the complete configuration space, on the right, one of the 8 identical doubly connected components of free space.

touch successive teeth on the other.

6.5 Cams

Cams are devices used to transform rotational motion into periodic translational motion. They consist of a rotating part, the cam, which drives a translating output part, the lever. The shape of the rotating part determines how the motions are translated. By choosing the proper shapes, rather complicated relative motions can be achieved. These motions often depend on certain specific boundary curves of the cam. This feature can not be described adequately in the qualitative formalism. Just as for a human engineer, an algebraic analysis is required to determine the exact interaction. In this section, we give examples of place vocabularies for 2 types of cams: a "standard" cam and a Scotch yoke.

6.5.1 Standard cam

We consider the cam shown in Figure 6.35. The lever is free to translate in the vertical direction,

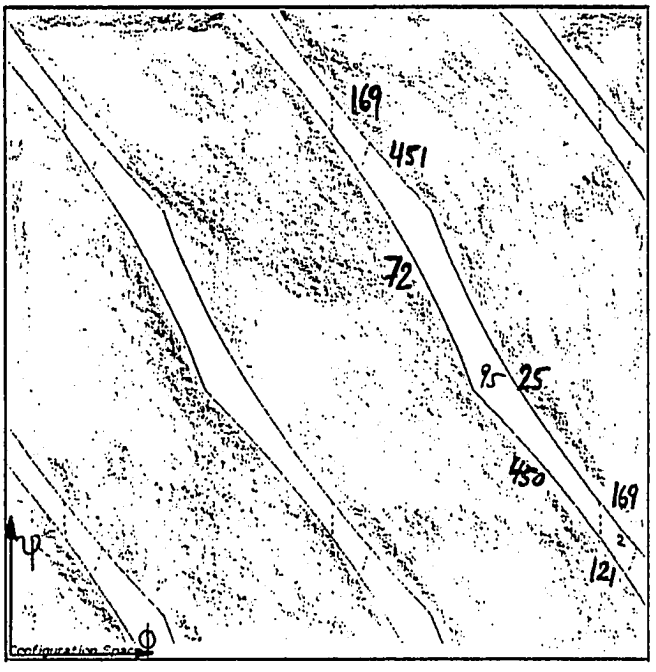

Figure 6.32: An enlarged section of the configuration space for the gearwheel example.

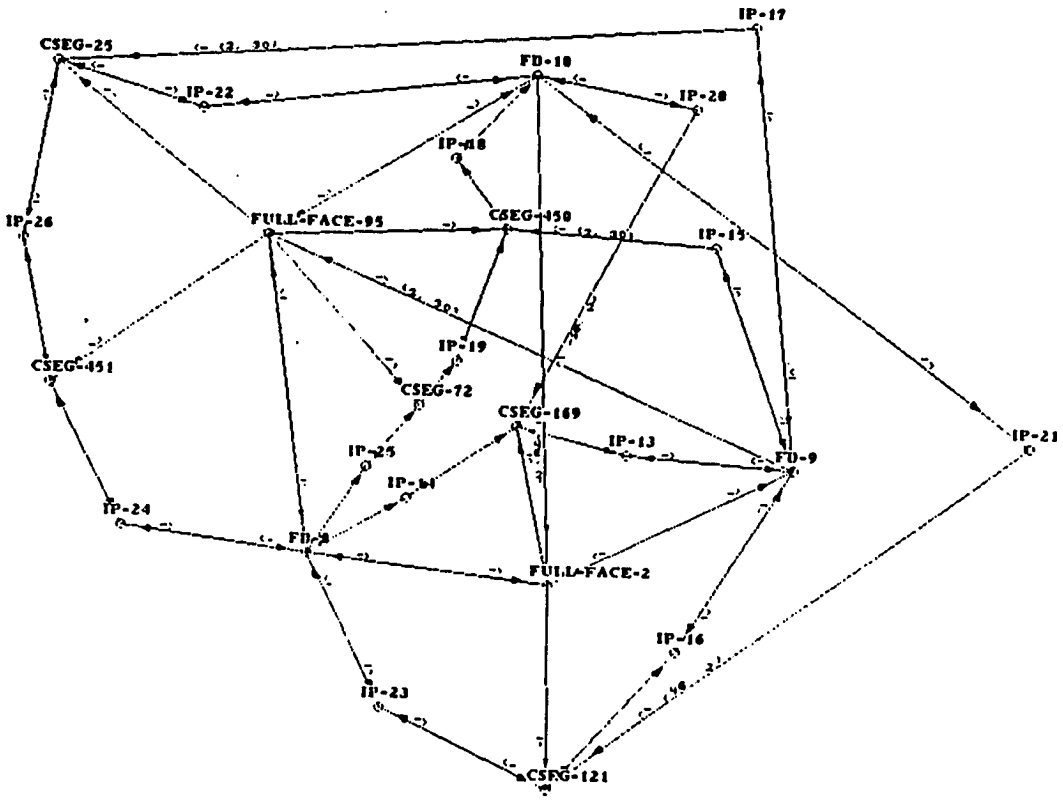

Figure 6.33: The place graph for the gearwheel example.

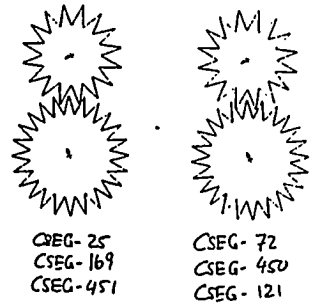

Figure 6.34: Sample configurations for some of the places for the gearwheel example.

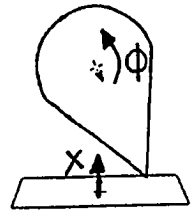

Figure 6.35: A standard cam mechanism.

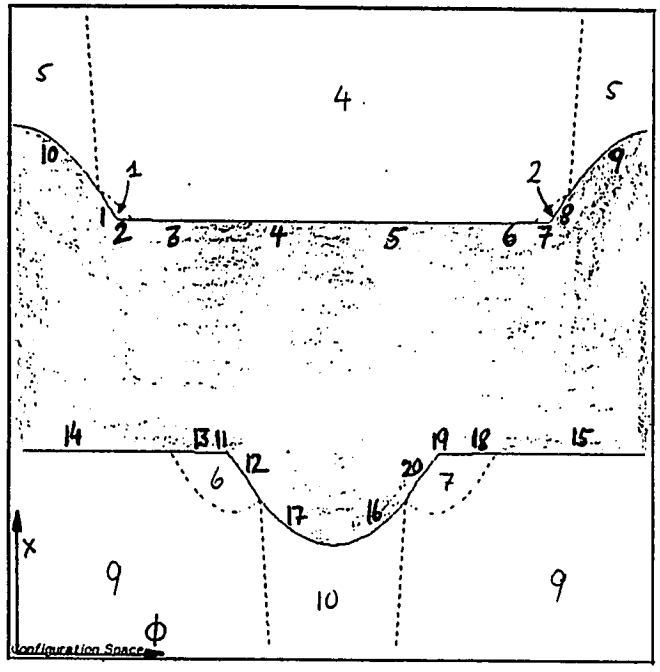

Figure 6.36: The configuration space for the cam mechanism. The horizontal parameter is the orientation angle of the cam, the vertical the translation of the lever.

and the cam rotates. The configuration space, shown in Figure 6.36, is thus a 2-dimensional cylinder surface. It contains 2 infinite regions of free space, corresponding to the lever being above and below the cam. As in the previous examples, the numbers in Figure 6.36 indicate the constraint segments that corresponds to the configurations. As the cam rotates, the asymmetries in its shape drive the lever up and down. In this example, the lever oscillates once for each rotation of the cam, but this could be varied by changing the shape of the cam. Examples of configurations occurring in this cycle are shown in Figure 6.37. They are labeled with the constraint segments that they correspond to. The place graph for this example is shown in Figure 6.38. It consists of 2 connected components, corresponding to the 2 regions of free space. In the case of translational attachment, the program assumes a division of free space that delimits the feasible region of translation. In this example, the

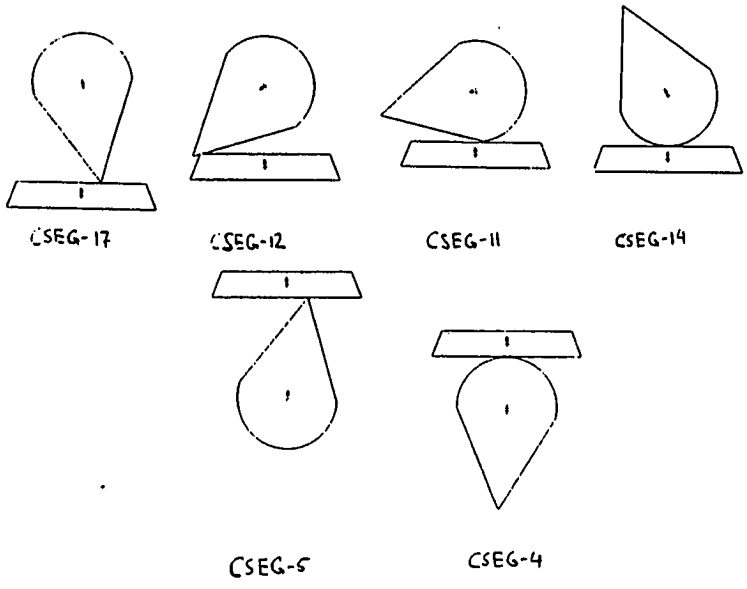

Figure 6.37: Sample configurations for the cam mechanism.

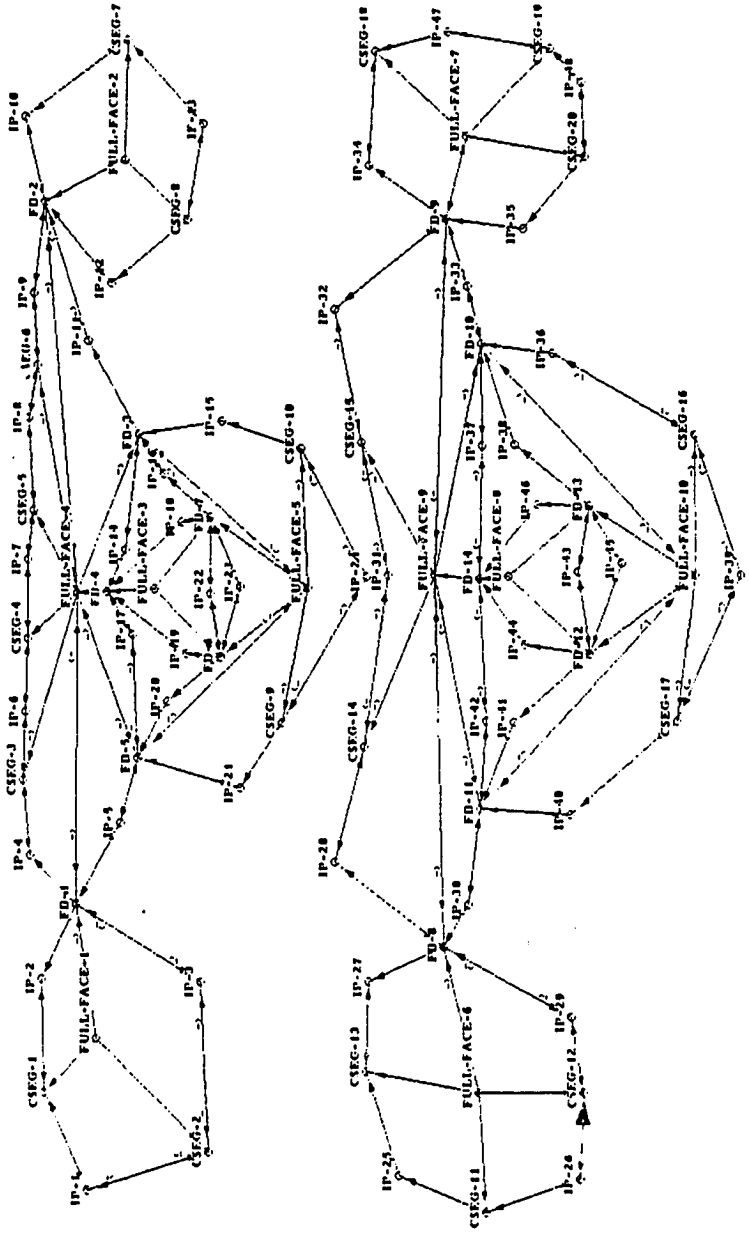

Figure 6.38: The place graph for the cam.

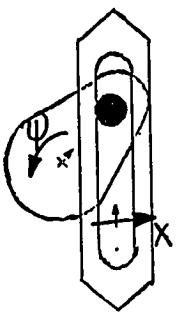

Figure 6.39: A scotch yoke.

division is made up of the free-space divisions FD-6 and FD-7 above, and FD-12 and FD-13 below. Each of the regions outside the feasible regions forms a place in itself, and these are FULL-FACE-3 and FULL-FACE-8. However, because they have no constraint segments (and thus no contacts between the objects) among their boundaries, they have no kinematic function.

6.5.2 Scotch yoke

A Scotch yoke, as shown in Figure 6.39, is a type of cam that translates rotational motion into sinusoidal translational oscillation. An important difference of this mechanism to the standard cam is that the lever can not be detached from the cam. The configuration space for this example consists of 3 regions: one where the pin of the cam is inside the slot of the lever, and 2 regions where it is outside on one side of it. Examples of these configurations are shown in Figure 6.40. Configurations in CSEG-39 and CSEG-44 lie on the 2 boundaries of the region within which the yoke normally operates. The configurations in places CSEG-52 and CSEG-47 are on the boundaries of the regions where the pin is outside of the slot and the yoke thus does not function properly. The configuration space and place graph for the yoke are shown in Figures 6.41 and 6.42. Note that no free-space divisions are needed to satisfy the quasi-convexity criterion in this example.

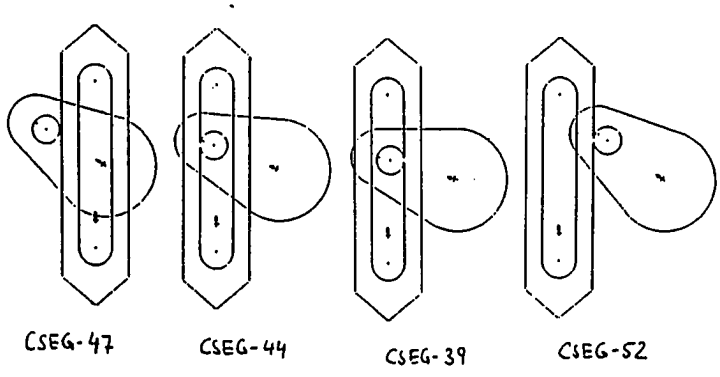

Figure 6.40: Sample configurations of the scotch yoke.

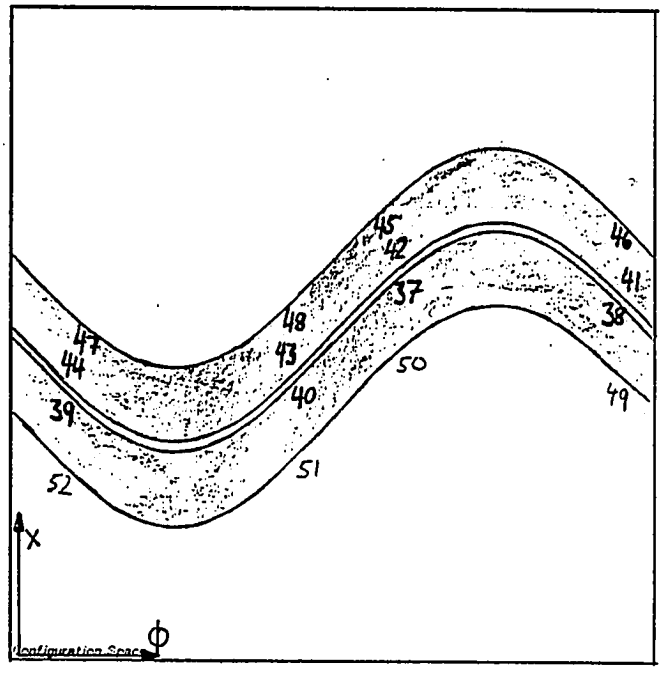

Figure 6.41: The configuration space for the scotch yoke

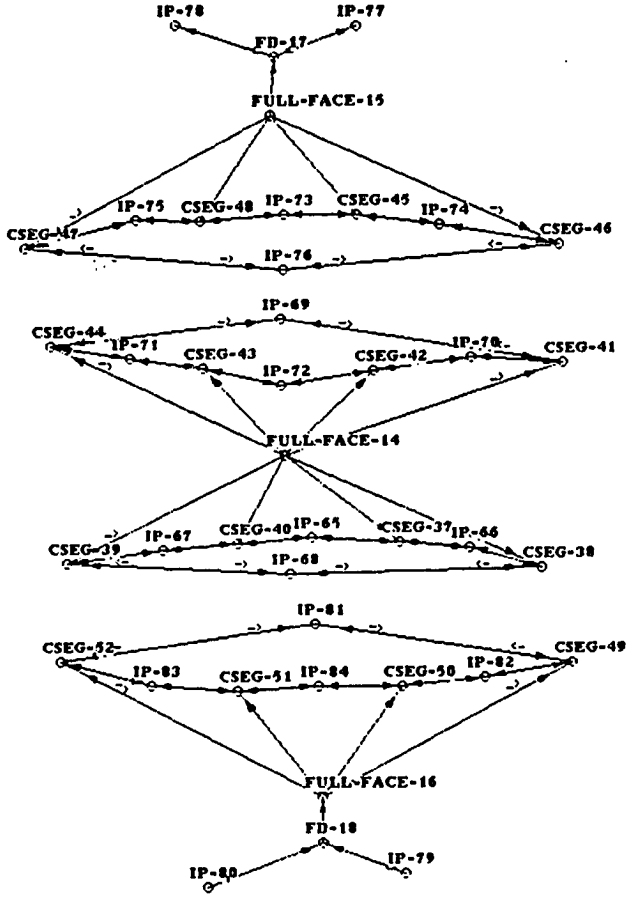

Figure 6.42: The place graph for the scotch yoke

6.6 The QRG Clock

As a test of the place vocabulary theory on a real-life mechanism, we have analyzed the QRG clock, shown in Figure 6.43. The clock contains an escapement, 6 pairs of gearwheels, and a ratchet. The arrangement of the parts and their motion during normal operation of the clock is shown in Figure 6.44. As the number of teeth of the gearwheels are not readily visible in Figure 6.44, they are listed in Figure 6.45. Gearwheel 6 is attached to a spring which drives it to the left. This spring is the source of energy for the clock. It is wound up using a handle connected to a ratchet, which prevents the handle from turning backwards and relaxing the spring. The spring and ratchet are indicated at the bottom of Figure 6.44. Two kinematic chains originate at gearwheel 6. One connects it with the escapement, thus regulating the motion of gear 6, and the other connects it with the hands of the clock.

The first kinematic chain consists of the 3 kinematic pairs between gearwheels 5/6, 3/4 and 1/2, and the escapement. It is formed by rigid connections between successive gears, i.e., 5 with 4, 3 with 2 and 1 with the escape wheel. The ratio between the number of teeth of the gearwheels in these pairs is 62/15, 56/10 and 52/11, and the number of teeth of the escape wheel is 34. Thus, during one revolution of gearwheel 6, the number of times that the pallet of the escapement oscillates is

$$34 \cdot \frac{62}{15} \cdot \frac{56}{10} \cdot \frac{52}{11} = \frac{3069248}{825} \approx 3720.3$$

The pallet is directly attached to a pendulum. When the clock is properly calibrated, Gearwheel 6 makes one revolution per hour, and the pendulum swings back and forth a little more than once per second.

The second kinematic chain, linking gearwheel 6 to the hands of the clock, consists of the 3 kinematic pairs between gearwheels 6/7, 8/9 and 10/11. It is formed by the rigid connections between gears 7 and 8 as well as 9 and 10. The first of these pairs, between gears 6 and 7, has the sole purpose of reversing the direction of motion, so that the minute hand can be driven clockwise. The following 2 pairs have ratios of 40/10 and 36/12, so that gear 11 rotates $40/10 \cdot 36/12 = 12$ times slower than gear 7. It can thus serve to drive the hour hand, making one full revolution every 12 hours.

Using our implemention, we have computed place vocabularies for each of the kinematic pairs in the clock. These place vocabularies will be used by Paul Nielsen ([NIE88]) to compute an envisionment

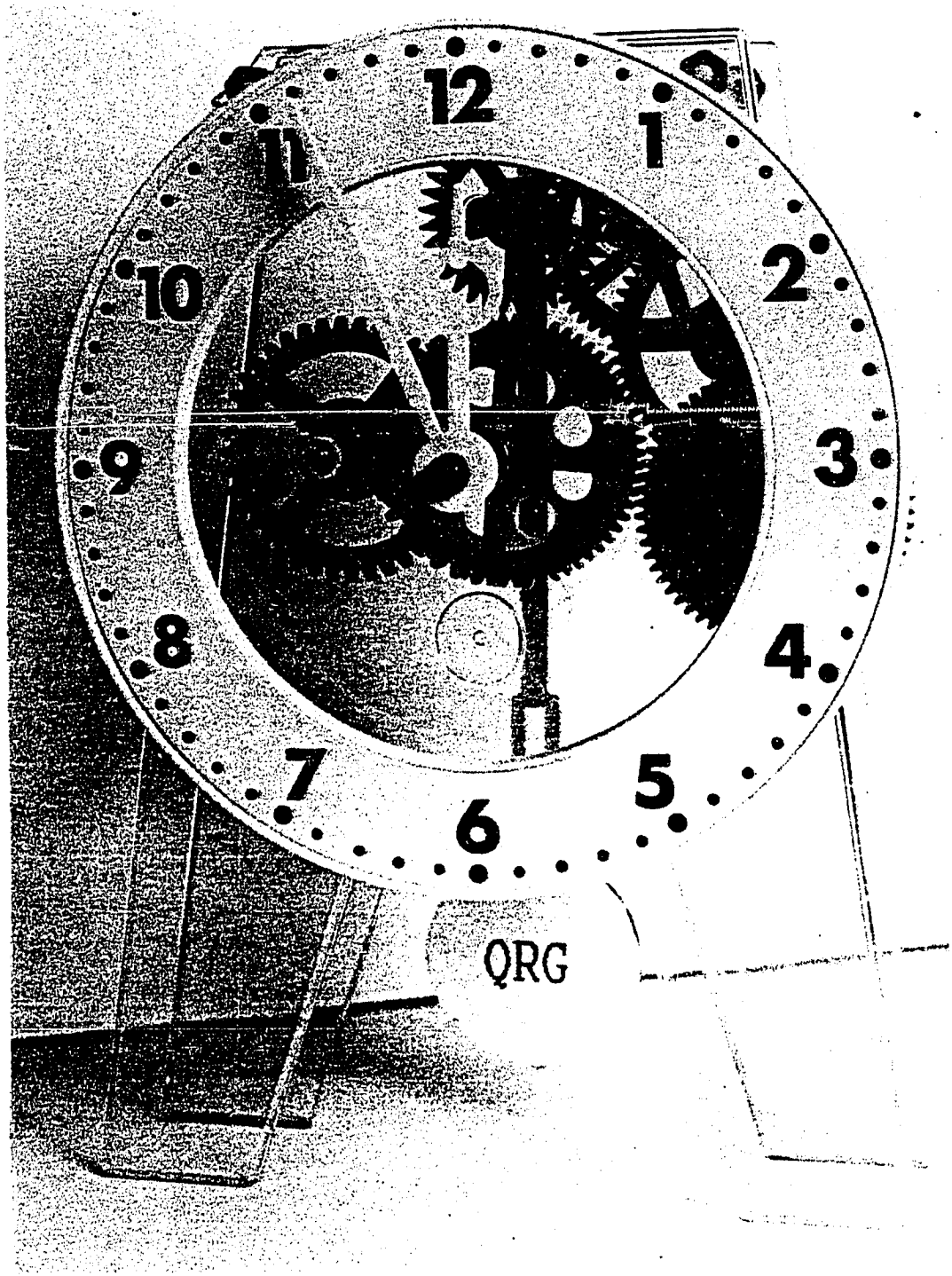

Figure 6.43: The QRG clock.

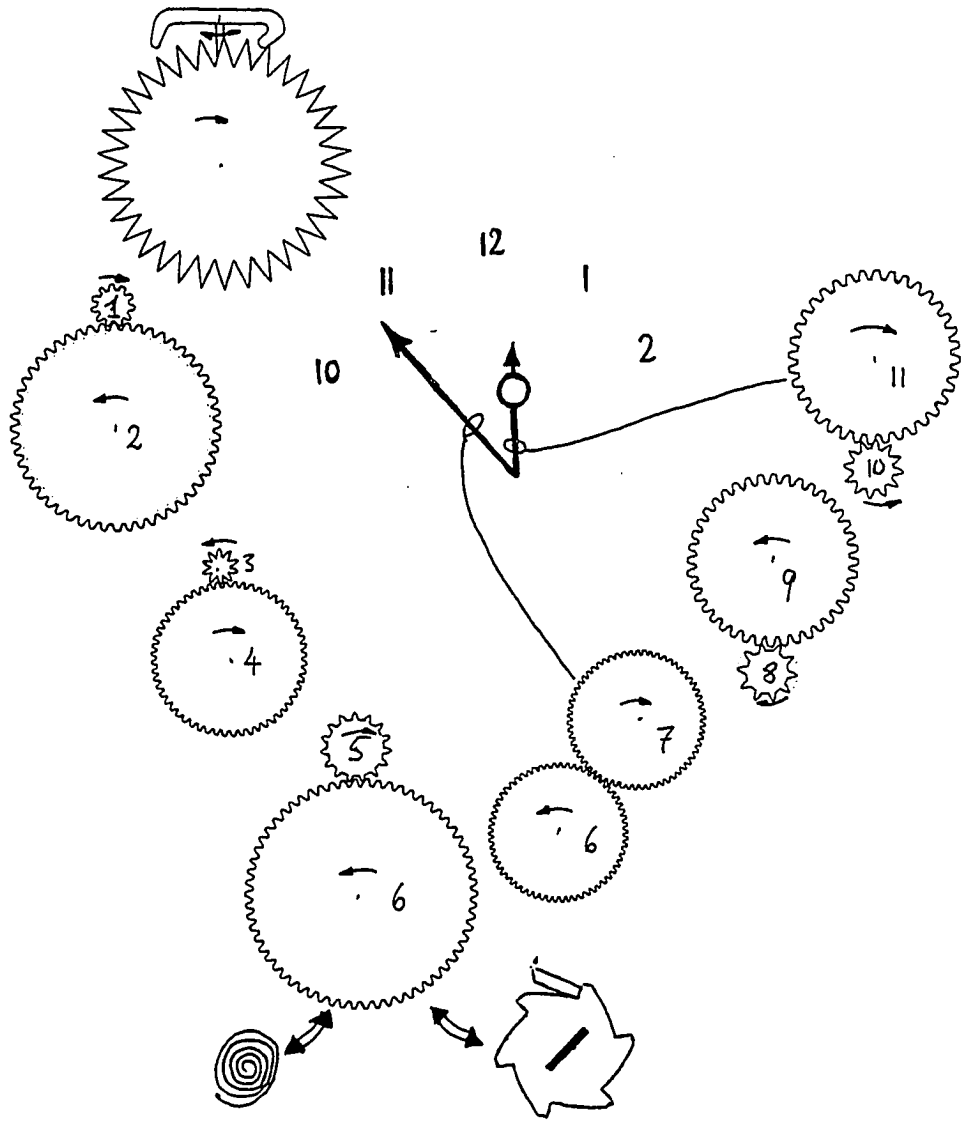

Figure 6.44: The kinematic chains of the QRG clock.

Gearwheel	Number of Teeth
Escape Wheel	34
1	11
2	52
3	10
4	56
5	15
6	62
7	62
8	10
9	40
10	12
11	36

Figure 6.45: The number of teeth for the gearwheels in the QRG clock.

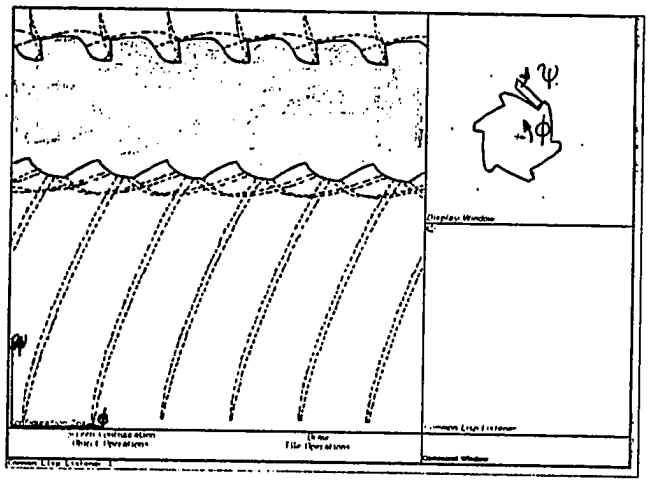

Figure 6.46: The configuration space for the ratchet.

of the clock's behavior. As his work is not completed yet, we can not report on the final result of this analysis. We thus describe the place vocabularies we have generated for each of the kinematic pairs. As the graphs get rather large and complicated, we describe the place vocabularies by the configuration space tesselation only. For the ratchet, we show the full configuration space in Figure 6.46, but for all other examples, we illustrate the tesselation using an enlarged section of the configuration space in Figures 6.47 through 6.53. The reader is referred to the similar examples discussed earlier for a full discussion of the form of the configuration space constraints.

In this example, we had to contend with several major difficulties. It was very difficult to get accurate descriptions of the geometries of the parts. While we obtained some quite good approximations from photographs of the device, these had to be corrected by trial and error using the results of the place vocabulary computation. In the case of the escapement, the distance between the escape wheel and the pallet has to be accurate to 0.1 % in order for the pair to function properly. Floating-point truncation errors pose the second major problem. The computer used for the implementation does not have a very refined arithmetic, and the program thus uses a rather large tolerance (10^{-7}) to allow for truncation error. However, it turned out (in the case of the ratchet) that the example required

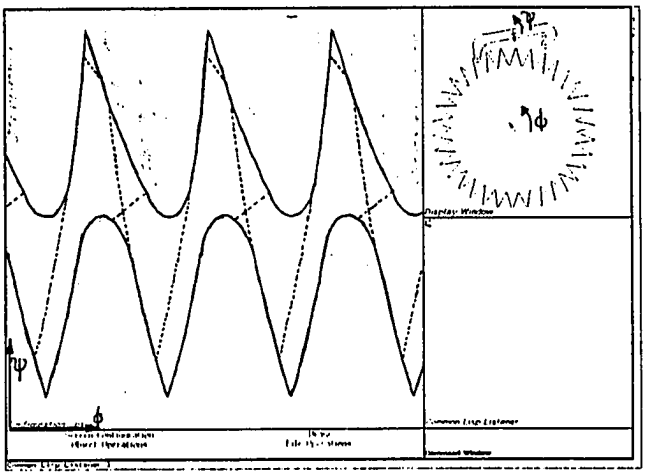

Figure 6.47: An enlarged section of the configuration space for the escapement.

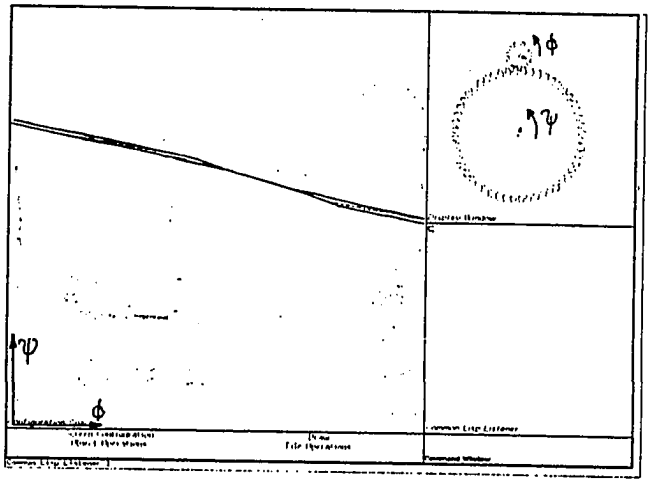

Figure 6.48: An enlarged section of the configuration space for the pair between gears 1 and 2.

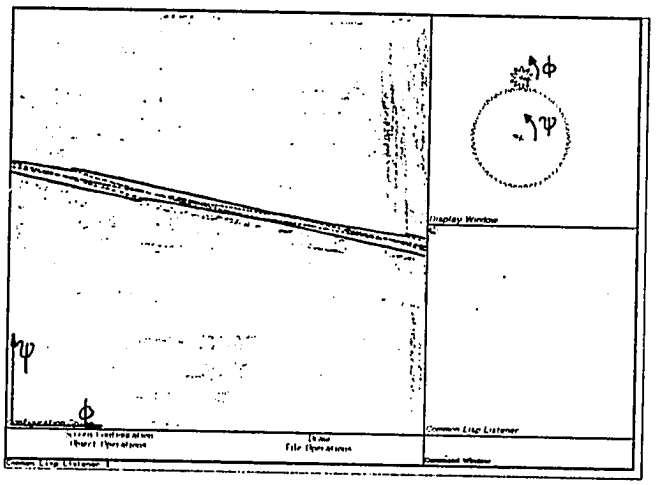

Figure 6.49: An enlarged section of the configuration space for the pair between gears 3 and 4.

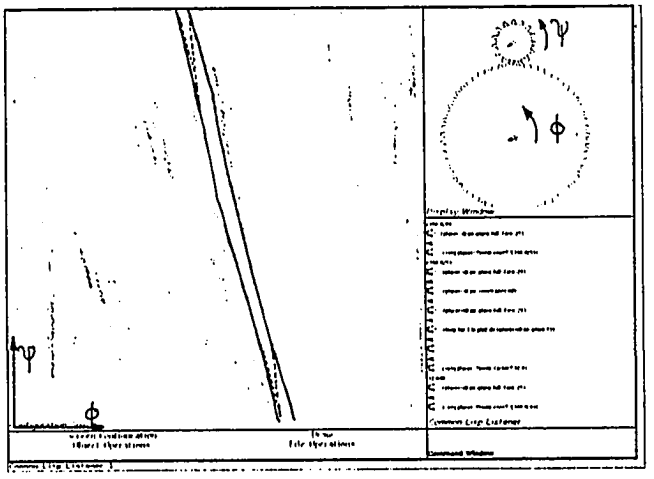

Figure 6.50: An enlarged section of the configuration space for the pair between gears 5 and 6.

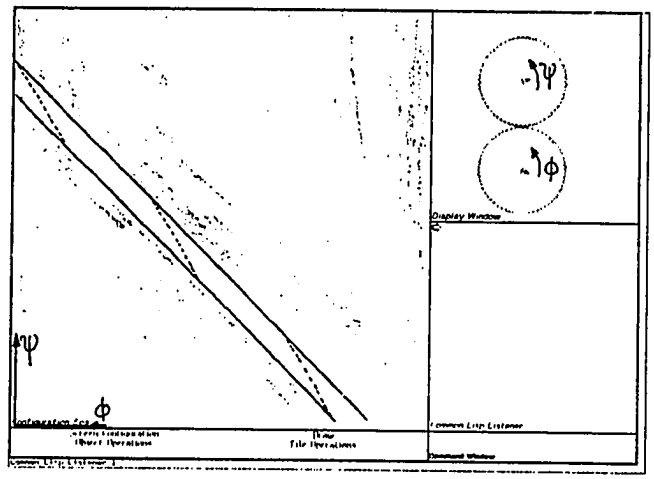

Figure 6.51: An enlarged section of the configuration space for the pair between gears 6 and 7.

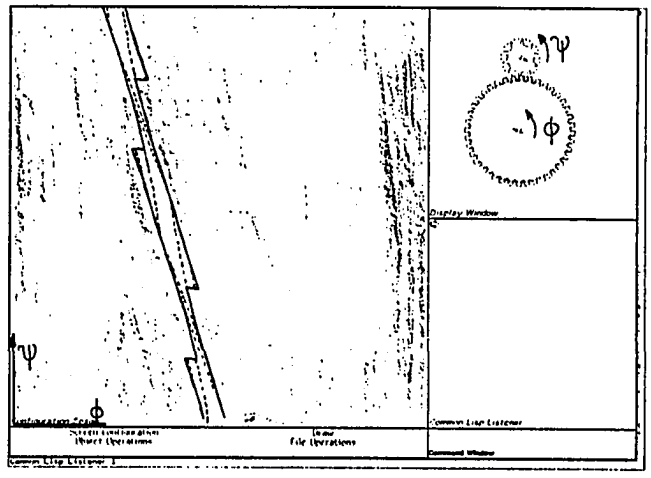

Figure 6.52: An enlarged section of the configuration space for the pair between gears 8 and 9.

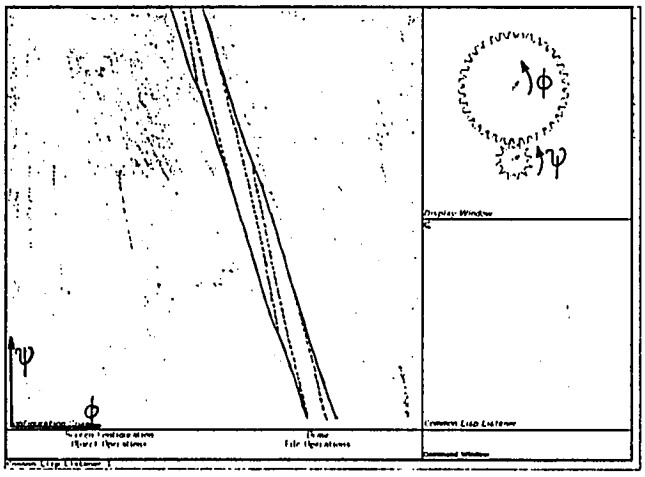

Figure 6.53: An enlarged section of the configuration space for the pair between gears 10 and 11.

finer distinctions. We avoided the problem by a very slight variation of the dimensions of the parts, but this is by no means a satisfactory solution. The third problem is that the implementation is very slow: with the exception of the ratchet, the computation for all the kinematics pairs had to be run overnight. However, we are confident that our algorithms can be implemented in a way that avoids these problems.

Chapter 7

CONCLUSIONS

We begin by briefly reviewing other work in spatial reasoning and motion planning. We then discuss several categories of future work. First, we discuss potential improvements in both the place vocabulary theory and the algorithms. We then describe possible extensions and improvements to the place vocabulary theory itself. Finally, we outline the other requirements for a theory of mechanical design, and list the work that will be needed to achieve a complete theory of human reasoning about mechanism kinematics.

7.1 Review of Related Work

We have argued in the introduction that it is a bad idea to attempt to reason about kinematic interactions using analog representations of individual objects. However, such approaches have been explored in related areas. The most important work of this kind is the ACRONYM system of R. Brooks ([BRO81]). This system dealt with recognition of objects from images and used reasoning to help the recognition process. Closer to the kinematic domain, E. Davis developed the MERCATOR system ([DAV83]), which used the idea of fuzzboxes to reason with partially specified geographic information. However, neither system was designed to reason about kinematic interaction.

There has been substantial work on purely symbolic spatial reasoning. The idea is that when objects are represented by a finite set of symbols, it is possible to express their interactions by a set

of rules. This approach is implicitly assumed in most of the work in the blocks world, but had not been formalized for complex domains. C. Stanfill [STA83]) developed a symbolic system to represent and reason about cylinders. A more ambitious project is recent work by E. Davis ([DAV86]), who is devising a logic-based system to reason about objects moving through space. As his work is still in its beginning stages, it is not clear yet how powerful his approach will be. Finally, Andrew Gelsey ([GELSEY87]) has implemented a system that analyzes mechanism kinematics. He is trying to model complete mechanisms using symbolic descriptions of the function of kinematic pairs and rules that relate between them. His system incorporates numerical information into the description, but does not use it for reasoning. Because his system is based on a finite set of possible kinematic pairs, it is not powerful enough to completely represent higher pairs. However, his work demonstrates that reasoning about the function of complex mechanisms is tractable. The place vocabulary could very well provide the necessary generative vocabulary to make his system general.

While these symbolic approaches are powerful in the domains for which they are designed, they lack the important property of *generativity*. The set of objects they can represent is limited by the symbol set they use, and therefore they are in principle unable to reason about *general* geometric shapes. To overcome this problem, we need a generative symbolic representation. This is one of the goals of modern qualitative physics ([FOR81,FOR84,DKL79,DKL84]). The place vocabulary idea originated in FROB ([FOR81]), which analyzed qualitatively the possible motions of a point mass in polygonal regions. Places were created by a tesselation of the regions into rectangles. FROB could then reason about the possible motions of the point mass within these regions. For example, it could answer what places the mass could come to rest in given that it started in some given way. FROB, however, could not yet reason about kinematics, and the objects it dealt with were just points.

The abstraction we use in this thesis to generalize the idea of place vocabularies is *configuration space*. This idea is due to Thomas Lozano-Perez ([LPW79,LMT83,BLP83]) and has been developed for the problem of robot motion planning. The development of configuration space as a suitable abstraction of the problem has resulted in a large amount of work on algorithms for robot motion planning. Of particular relevance to our work is the idea of using *algebraic cell decomposition*, first proposed by Schwartz & Sharir ([SCHS83]). In this work, a method for motion planning using a decomposition of configuration space into cells very similar to place vocabularies is proposed. How-

ever, their cell decomposition does not use applicability constraints. The unbounded domain of the constraints means that *all* constraints are applicable at *all* legal configurations. This results in a large number of unnecessary divisions of free space, which is very undesirable for a place vocabulary. However, in motion planning, this distinction is irrelevant. Unfortunately, despite some recent improvements ([BKR85,KY85]), the algorithms proposed to compute the algebraic cell decomposition are all so inefficient that even for toy examples, none of them has ever been implemented. We have not been able to use them for the computation of place vocabularies.

Finally, the idea of kinematic pairs and chains is due to Franz Reuleaux ([REU75] in German, English translation in [REU76]). Many of the ideas in the framework of this thesis are due to his seminal work. Yoav Shoham ([SHO85]) analyzes static arrangements of objects, but unfortunately the analysis breaks down as soon as the objects move, making the representation unusable for kinematics.

7.2 Improvements in the Current Place Vocabulary Theory

In this section, we discuss some desirable extensions and improvements in various parts of the current place vocabulary theory. All of the ideas proposed in this section assume the place vocabulary theory as given in this thesis.

7.2.1 Faster algorithms

Unfortunately, the current place vocabulary implementation is very slow. A faster version would facilitate further research. Recoding the current implementation could speed up the program somewhat, but the potential for such improvement is limited. It looks more promising to try to decrease the complexity of the algorithms.

In the instantiation of the constraints, the current implementation uses symbolic algebra to compute the various equations and functions. This has given us a lot of flexibility for experimenting with various definitions of place vocabularies. The instantiation could now be sped up by replacing the symbolic algebra by a set of functions for each of the coefficients of these equations.

A large part of the rest of the computation is spent on detecting subsumption intersections between constraints. There are various ways to speed up this process. First, one could better index the

constraints to narrow the set of constraints considered for intersection tests. Recall that the current implementation uses a test for intersection of bounding rectangles to narrow this set. As all the rectangles are distinct, testing their intersection for each pair of constraints is a slow process, and could probably be sped up by using a fixed global set of bounding rectangles to narrow the search.

As another possibility, it might be possible to further narrow the set of possible candidates by combining the intersection detection with the computation of regions of free space. Constraints would thus be tested for intersection only when they are considered as boundaries of free space. This might allow us to compute only those subsumption intersections that occur at legal configurations, and not waste any effort on those whose configurations are not legal anyway. To do this, we need a fast algorithm to determine whether a given configuration is legal, and recent work in robotics and computational geometry might provide such a method.

Finally, the computation could take fuller advantage of periodicities in the objects. The most computation-intensive parts of the program already do this, but some improvement can still be gained in various places. For example, periodical copies of constraints and touchpoints could share the same data structures to save memory, and pointers between them could be replaced by an indexing scheme.

7.2.2 Analysis of complete mechanisms

The research presented in this thesis is concerned mostly with the analysis of higher kinematic pairs. As one of the goals of qualitative kinematics is to analyze complex mechanisms, it would be desirable to apply this kinematic pair analysis in the context of a larger system.

We have already given an analysis of a complete clockwork, the QRG clock. Paul Nielsen ([NIE88]) is devising a system called ALEX which uses place vocabularies to analyze such clockwork mechanisms. The main goal of his research is to determine how the forces and motions of the parts of a complete mechanism can be analyzed qualitatively. The theories he is devising will allow us to compute an envisionment of the mechanism's actual behavior from the place vocabularies for the kinematic pairs.

What is missing in the system is the automatic identification of the lower kinematic pairs in the mechanism, i.e., an anlysis of the joint configurations to determine the freedom of the parts. The input information explicitly specifies the freedom of each part as either translation along a certain axis or rotation. The problem of identifying this freedom is very different from the analysis of higher

kinematic pairs, and we have thus excluded it from our work. There exists an exact mathematical analysis for this problem ([DEHA55]), and recent work by Andrew Gelsey ([GELSEY87]) shows that it is possible to solve this problem by rule based methods, a much simpler and faster procedure than the expensive place vocabulary analysis. It would be interesting to combine his techniques with the ALEX system to yield a system covering almost all practical mechanisms.

7.2.3 Extension of coverage

The current implementation is limited to 2-dimensional objects whose boundaries are made up of straight lines and arcs. While this covers a large portion of practical mechanisms, there are cases that fall outside this domain.

A higher kinematic pair in 3 dimensions differs from a 2-dimensional one only in the form of the configuration space constraints. Both objects have just one degree of freedom each, so that the configuration space is still 2-dimensional. All that has to be done to extend the implementation to 3 dimensions is to provide representations and configuration space constraints for 3-dimensional objects. This is not an easy task, but it does not require any new theories.

Extending the theory to cover more cases of boundary curves is more difficult. To handle general algebraic curves, we need an algebraic engine that finds the equations of the configuration space constraints. Furthermore, as the configuration space constraints become high-order algebraic curves, the endpoint tests we now use to detect intersections between them will become too unreliable. It will thus be necessary to test for intersections using algebraic decision methods, a very slow procedure. Further research is needed to determine how to make this extension in a reasonable manner.

7.3 Requirements for a Revised Place Vocabulary Theory

In this section, we discuss some possible modifications to improve the place vocabulary theory.

7.3.1 Three dimensions from two dimensions

One of the justifications for limiting the current theory to 2 dimensions is that people solve 3-dimensional problems not by direct 3-dimensional analysis, but by a set of related 2-dimensional

ones. So far, we have not investigated how this process might work.

Mechanical engineers represent and reason about mechanisms in suitable projections. Reasoning about 3-dimensional interactions thus seems to be split up into 2 parts: first, find a suitable set of 2-dimensional projections of the problem, and second, the analyses of these subproblems are composed to predict the behavior of the actual 3-dimensional mechanism. The place vocabulary theory provides a framework within which these phenomena can be studied.

7.3.2 Division of free space

In the current theory, free space is tesselated into different places by applicability constraints. There are many ways in which such applicability constraints could be defined, and the one we have chosen in our theory is underconstrained. Given the crucial influence of this choice on the place vocabulary, this is an undesirable situation.

Currently, the only requirement placed on applicability constraints is that they bound the algebraic domain of applicability on the corresponding constraint itself. To obtain a unique definition, we must find additional criteria for constraint applicability for configurations in free space. The most promising way to define such criteria is to make assumptions about probable motions of the objects. A constraint is then applicable if a probable motion will lead to a configuration that satisfies it. An interesting empirical question is whether there exists a reasonable set of such assumptions.

7.3.3 Recognition of mechanism function

As the place vocabulary provides a representation for mechanism function, it might be used to categorize place vocabularies into equivalence classes that express a certain mechanical function. Thus, the place vocabularies of different ratchets all have certain ratchet-defining features in common. It would be interesting to see if recognition functions for common classes of mechanisms can be based on the place vocabulary theory.

7.3.4 Compositional computation

When people analyze a mechanism, they seem to be able to predict local behavior from local information. For example, for a gearwheel, people can decompose the behavior into that of individual teeth

"pushing" each other onward, and this analysis seems to be based on information about single teeth only.

Currently, we compute place vocabularies by a global analysis of the complete objects. This does not correspond to the way people seem to analyze the problem. The theory could be improved to compute the place vocabulary in parts based on localized information. Based on this, an interesting question is whether we can find a set of primitive interactions from which place vocabularies for complex objects can be composed. This might also give us much more efficient algorithms for computing place vocabularies.

Note that it will be difficult to preserve completeness in the composition process. For example, a pendulum escapement (examples of which are found in the example chapter) fails when the distance between the parts becomes too small. This failure is manifested in subsumptions between constraints generated by interactions of either end of the pallet. In order to find these subsumptions, the composition would have to test for subsumptions between any of the places in the primitive elements. But this amounts again to a global analysis! However, predicting such subsumptions seems to be a problem in which people frequently fail, indicating that people probably solve this problem in a heuristic manner.

7.3.5 Leaving the mechanism domain

Kinematic interactions between physical objects are not restricted to mechanisms, but occur in many other, less constrained settings. In the general case, objects have 6 degrees of freedom. The place vocabulary theory as such covers this case as well, but the computation in 6-dimensional configuration spaces is not tractable with current computer technology.

However, it might be possible to find approximations using the 2-dimensional place vocabulary theory. In research on dot pattern perception (Jon Webb, personal communication), it has been shown that the human visual system tends to assume that objects move in one degree of freedom only. It is thus possible that people analyze kinematic problems under the same assumption. It would be interesting to develop a theory that decomposes the general spatial reasoning problem into subproblems with 2-dimensional configuration spaces to see if this hypothesis is indeed true.

7.4 Towards a Theory of Mechanical Design

One of the motivations for Qualitative Kinematics is the development of a theory of mechanical design. We have already indicated how the place vocabulary theory can be used for the solution of certain design problems, and have given an example of the solution of a parameter design problem. For general design problems, however, the current theory is still insufficient. In this section, we examine some of the issues that further research in this area should address.

An interesting empirical question is what the number of landmark values in typical cases of mechanism design actually is. Before we can answer this question, the implementation must be improved in various ways. As we have already indicated in the earlier discussion, a more elaborate algorithm is needed to determine the actual values of the roots of the expressions defining the landmark values. Besides this, we also need an efficient way to indicate the effect that the crossing of a landmark value has on the place vocabulary.

Our current implementation allows us to generate landmark values for the arrangement of the parts in the mechanism. Mechanical design does not concern only the relative arrangements of parts, but also the design of the parts themselves. Landmark values for variations in the objects themselves must also be analyzed.

In a design problem, there are usually many parameters that can be varied. Often, their variation has opposite effects on the result, and thus it is important to consider them simultaneously. Multiple open parameters in the place vocabulary computation mean that we have to deal with *landmark regions*, not values. Finding these regions amounts to finding the regions formed by a set of complicated curves in a high-dimensional space. Exact solutions to this problem can not be computed efficiently, so that it becomes important to devise approximation algorithms for this tesselation.

Besides varying parameters, it is also possible to vary the structure of a mechanism itself. This includes varying the number of teeth in a ratchet, varying the number of gears used in a transmission, and so forth. While such reasoning stands at a higher level than the place vocabulary analysis, it is grounded in the elementary analysis of mechanism behavior and this relation should be investigated.

7.5 Towards a Theory of Human Competence

The development of the place vocabulary theory has been heavily influenced by the idea of psychological plausibility. While we can not yet hope to give an accurate theory of human spatial reasoning, we have taken great pains to ensure that the place vocabulary does not make assumptions that contradict what is known about human cognition. In this section, we sketch what issues should be addressed to develop the current theory into one of human understanding of mechanism kinematics.

7.5.1 The primal sketch as an object description

The work of David Marr ([MARR82]) has shown that the human visual apparatus computes a symbolic representation of the image at a very low level. In particular, there exist arrangements of cells in the retina itself that detect the presence of lines and line endings. We have seen that these are just the elements that are required to predict the possible kinematic constraints in configuration space. However, we are also interested in the qualitative relations between the object parameters. This often requires subdivisions of the boundary segments. As the number of such subdivisions is not known a priori, it is not reasonable to assume that they can be found from predicate evaluations in the metric diagram. Their existence must be predicted in the primal sketch itself.

We have seen that the subdivisions require information about the possible motion of objects. However, the primal sketch is a low-level representation. Information about possible motions can not be assumed to be available in the low-level visual system. The visual system must thus make fixed *assumptions* about how physical objects move. For example, a reasonable assumption is that physical objects always rotate around the center of gravity in the image. [1] It would be interesting to make experiments that determine what assumptions are actually made.

Note that it is also possible that the visual system uses approximations of object boundaries, which would make the number of possible subdivisions known a priori. In this case, their existence could be determined by the metric diagram. The approximations, however, should result in other imperfections in people's predictions. Making experiments to find them will tell us more about how the human visual system actually represents objects.

[1] This requires the assumption that low-level vision can compute this point.

7.5.2 Implementation of the metric diagram

We distinguish two types of implementation of the metric diagram: *internal* and *external.* In an internal implementation, the predicates are evaluated by operations on some mental representation of the object or its image. In an external implementation, new measurements are taken on the actual objects to evaluate the predicate.

As the power of mental imagery is very limited, not all of the metric diagram can be assumed to be internally implemented. This is reflected in the fact that people often can not predict kinematic relations simply from seeing the objects involved. People then often resort to experiments on the actual objects themselves, and thus use an external implementation of the metric diagram.

As the internal implementation of the metric diagram is based on mental imagery alone, the question of which predicates it can and can not evaluate sheds light on the power of the mental imagery apparatus. Together with S. Ullman's theory of visual routines ([ULLMAN84]), qualitative kinematics can serve as a basis for devising very precise experiments that test this function.

7.5.3 The role of learning

The place vocabulary theory is a theory of first principles. Because it often requires additional measurements in the world, it is too slow and cumbersome to serve as a plausible model of people's everyday performance. A more realistic model would be one of learned analyses and recognition procedures that index them. For example, in the analysis of a mechanism with many gearwheels, each kinematic pair of gearwheels can be recognized and its function predicted from previously analyzed cases of gearwheels. If introspection is to be trusted, this seems to be the way that people analyze common mechanisms.

As it is unreasonable to assume that people are born with knowledge of gearwheels and ratchets, their function must be learned. But in order to learn about the functioning of a pair of gearwheels or a ratchet, we must analyze its function based on first principles. This is precisely where the place vocabulary theory is important: it defines a way how this first principles analysis could be computed for any arbitrary mechanism. Place vocabularies can provide the elementary vocabulary from which learning about mechanical phenomena can start.

7.6 Final Remarks

We have seen that the place vocabulary theory provides a useful qualitative representation of mechanism kinematics. The restriction to the mechanism domain has allowed us to formulate a very concrete theory. Contrary to many previous theories of qualitative kinematics, the power and limitations of the place vocabulary theory are well-understood. The current theory is only a first step, showing that a precise and generative theory of qualitative kinematics is indeed possible. We hope that the place vocabulary can serve as a basis for further research into understanding kinematics and spatial reasoning in general.

We have also seen that the solutions to spatial reasoning problems are not easy to achieve. Research in this area is very tedious, and even the simple implementation described in this thesis took well over 1 year of programming effort. However, we hope the reader will agree that it has led to progress towards solving the mysteries of spatial reasoning.

Bibliography

[ASBR86] **H. Asada, M. Brady**: "The Curvature Primal Sketch," IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(1986), pp. 2-14

[BKR85] **Ben-Or, D. Kozen, J. Reif**: "The Complexity of Elementary Algebra and Geometry," Journal of Computer and System Sciences **32**, 1986, pp. 251-264

[BLP83] **R. Brooks, T. Lozano-Perez**: "A Subdivision Algorithm In Configuration Space For Findpath With Rotation," Proceedings of the IJCAI 1983, pp. 799-806

[BRO81] **Rodney A. Brooks**: "Symbolic Reasoning Among 3-D Models and 2-D Images," Artificial Intelligence **17**(1981), pp. 285-348

[COLL75] **George E. Collins**: "Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition," Second GI Conference on Automata Theory and Formal Languages, Lecture Notes in Computer Science **33**, Springer Verlag, Berlin, 1975

[DAV83] **Ernest Davis**: "The Mercator Representation of Spatial Knowledge," Proceedings of the AAAI 1983, pp. 295-301

[DAV86] **Ernest Davis**: "A Logical Framework for Solid Object Physics," New York University Technical Report 245, October 1986

[DEHA55] **J. Denavit, R.S. Hartenberg**: "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices," Journal of Applied Mechanics **22**, 1955

[DKL75] **Johann DeKleer**: "Qualitative and Quantitative Knowledge in Classical Mechanics," MIT AI TR-352, 1975

[DKL79] **Johann DeKleer**: "Causal and Teleological Reasoning in Circuit Recognition," MIT AI Lab. TR-529, September 1979

[DKL84] **Johann DeKleer, John Brown**: "A Qualitative Physics Based on Confluences," Artificial Intelligence **24**, 1984

[DON84] **Bruce Donald**: "Motion Planning with Six Degrees of Freedom," MIT AI Tech. Report 791, May 1984

[ERDMANN84] **Michael A. Erdmann**: "On Motion Planning with Uncertainty," MIT AI Tech. Report 810, August 1984

[FALT86] **Boi Faltings:** "A Theory of Qualitative Kinematics in Mechanisms," University of Illinois Technical Report UIUCDCS-R-86-1274, May 1986

[FALT87] **Boi Faltings:** "Qualitative Kinematics in Mechanisms," Proceedings of IJCAI 87, Milan, Italy, August 1987

[FNF87] **Ken Forbus, Paul Nielsen, Boi Faltings:** "The Inferential Structure of Qualitative Kinematics," Proceedings of IJCAI 87, Milan, Italy, August 1987

[FOR80] **Ken Forbus:** "Spatial and Qualitative Aspects of Reasoning about Motion," Proceedings of the National Conference on AI, Stanford, August 1980

[FOR81] **Ken Forbus:** "A Study of Qualitative and Geometric Knowledge in Reasoning about Motion," MIT AI TR 615, February 1981

[FOR84] **Ken Forbus:** "Qualitative Process Theory," MIT AI TR 789, July 1984

[GELSEY87] **Andrew Gelsey:** "Automated Reasoning about Machine Geometry and Kinematics," 3rd IEEE Conference on AI Applications, Orlando, Fla., Feb. 1987

[HAY79] **B. Hayes:** "The Naive Physics Manifesto," in: Expert Systems in the Micro-Electronic Age, D. Michie (ed.), Edinburgh University Press, May 1979

[KNUTH81] **Donald Knuth:** "The Art of Computer Programming," Vol. 2, Addison-Wesley, 1981

[KY85] **D. Kozen, C. K. Yap:** "Algebraic Cell Decomposition in NC," extended abstract, Proceedings of the 26th FOCS, Oct. 1985

[LMT83] **T. Lozano-Perez, M. Mason, R. Taylor:** "Automatic Synthesis of Fine-Motion Strategies for Robots," MIT AI Memo 759, 1983

[LPW79] **T. Lozano-Perez, M. Wesley:** "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles," Comm. of the ACM, Vol. 22 (1979), pp. 560-570

[MARR82] **D. Marr:** "Vision," W.H. Freeman & Co., San Francisco, 1982

[MAS79] **Matthew T. Mason:** "Compliance and Force Control for Computer Controlled Manipulators," MIT AI Lab. Technical Report TR-515, April 1979

[NIE88] **Paul Nielsen:** "A Qualitative Approach to Rigid Body Mechanics," Ph. D. Dissertation, University of Illinois, 1988

[PS85] **Franco Preparata, Ian Shamos:** "Computational Geometry," Springer Verlag, 1985

[REU75] **Franz Reuleaux:** "Theoretische Kinematik," Vieweg & Sohn, Braunschweig, 1875

[REU76] **Franz Reuleaux:** "The Kinematics of Machinery," Macmillan & Co, London, 1876

[SCHS83] **J. Schwartz, M. Sharir:** "On the Piano Movers Problem. II. General Techniques for Computing Topological Properties of Real Algebraic Manifolds," Advances in Applied Mathematics 4 (1983), pp. 298-351

[SHA76] **G. Shafer:** "A Mathematical Theory of Uncertainty," Princeton University Press, 1976

[SHO85] **Yoav Shoham:** "Qualitative Kinematics," Proceedings of the 9th IJCAI, Los Angeles, August 1985

[STA83] **Craig Stanfill:** "The Decomposition of a large Domain: Reasoning about Machines," Proceedings of the National Conference on AI, August 1983

[TAR48] **Alfred Tarski:**"A Decision Method for Elementary Algebra and Geometry," University of California Press, 1948

[ULLMAN84] **S. Ullman:** "Visual Routines," in: Visual Cognition, S. Pinker (Ed.), MIT Press, 1984

[ZADEH79] **L.A. Zadeh:** "A Theory of Approximate Reasoning," in: Machine Intelligence 9, J. Hayes, D. Michie, L.I. Mikulich (Eds.), John Wiley and Sons, New York 1979

Curriculum Vitae

Boi Volkert Faltings

Born April 10th, 1960, in Gelsenkirchen, W. Germany

Academic Background

Aug. 1983 - Aug. 1987

Graduate Student in Electrical Engineering at the University of Illinois in Urbana/Champaign, USA. Ph. D. in Electrical Engineering, thesis title: "Qualitative Kinematics in Mechanisms".

July 1984 & 1985

Attended Summer Program in Japanese Language at the International Christian University, Tokyo, Japan (no degree).

Oct. 1978 - Jan. 1983

Student in Electrical Engineering at the Swiss Federal Institute of Technology (ETH) in Zurich, Switzerland. Received Diploma in Electrical Engineering.

Dec. 1977

Received degree as Foreign Language Correspondent from the Chamber of Commerce, Bonn, W. Germany (German/French).

Employment

Aug. 1986 - Aug. 1987

Research Assistant, Qualitative Reasoning Group, University of Illinois

Aug. 1983 - Aug. 1984

Research Assistant, Coordinated Science Laboratory, University of Illinois

July - Oct.1981

Design and implementation of a high-level command language for ADASYS AG, Zurich

Aug. - Oct. 1980

Traineeship at the Nippon Telephone and Telegraph Corp., Tokyo, Japan

Nov. 1979 - Aug. 1983

System programmer at the Computer Center of the Swiss Federal Institute of Technology (part time during semesters, full-time during vacations)

Honors and Awards

August 1984-1986

IBM Graduate Fellow

November 1983

Silver Medal of the ETH and cash prize for the Diploma thesis

August 1983

Cash prize from SAP (Swiss Automatics Pool) for the Diploma thesis at ETH

January 1983

Graduated with Distinction from ETH

May 1978

Won 3rd prize in a German nationwide high school math contest

Publications

- "Qualitative Kinematics in Mechanisms", Proceedings of the 10th IJCAI, Milan, Italy, August 1987
- "The Inferential Structure of Qualitative Kinematics" (with Ken Forbus and Paul Nielsen), Proceedings of the 10th IJCAI, Milan, Italy, August 1987
- "A Theory of Qualitative Kinematics in Mechanisms," University of Illinois Technical Report UIUCDCS-R-86-1274, May 1986
- "The Effect of Channel-Exit Protocols on the Performance of Finite Population Random-Access Systems" (with P. Mathys), invited at the ACM Sigmetrics Conference on Measurement and Modelling of Computer Systems, May 1987
- "Towards a Model of Conceptual Knowledge Acquisition through Directed Experimentation" (with G. Dejong and S. Rajamoney), Proceedings of the 9th IJCAI, Los Angeles, August 1985

- "On the Behavior of Certain Random Access Protocols with Free Access," IEEE Int'l Symposium on Information Theory, Brighton, June 1985