From Kinematics to Shape: An Approach to Innovative Design

Leo Joskowicz
Department of Computer Science
Courant Institute of Mathematical Sciences
New York University
251 Mercer Street, New York, NY 10012

Sanjaya Addanki
IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598
USA

1 Introduction

This paper addresses the problem of designing elementary components for mechanical devices. The automatic design of mechanisms presents a number of interesting issues, not encountered in other domains [Dixon, 1986]. One of the key issues in mechanism design is the ability to reason explicitly about the relationship between the geometry of objects and their function in a mechanism. The motions of each object and the relationships between these motions (i.e., the mechanism's *kinematic behavior*) are directly determined by the shapes of the objects and the nature of the contacts between them. Unlike other domains, the basic building blocks of a mechanism are *pairs* of objects, rather than individual objects [Reuleaux, 1876]. Examples of elementary components (called *kinematic pairs*) are a screw and bolt, a pair of meshed gears, and prismatic joints. Complex mechanisms are designed by assembling kinematic pairs to achieve the desired behavior.

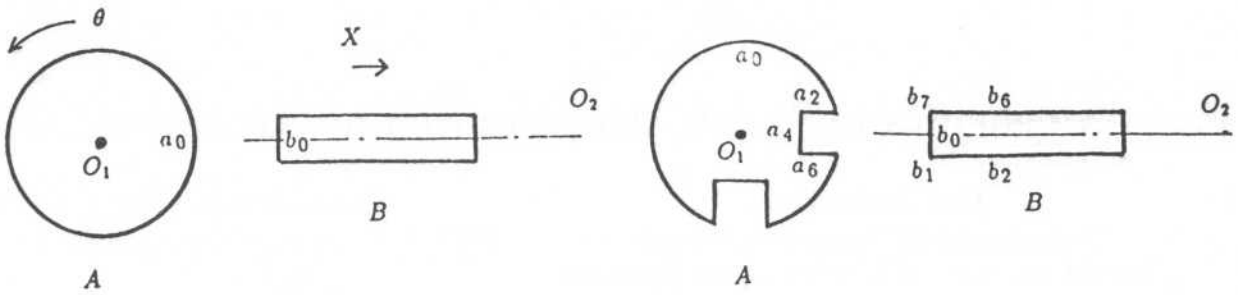
It is a common observation that in order to comply with a set of design requirements, new or modified shapes of objects in kinematic pairs need to be considered. In most existing Computer-Aided Design (CAD) systems, the decision on the creation or modification of an object's shape is the task of the human designer; the CAD system is responsible for handling and verifying the consistency of the design decision. Other systems are capable of modifying the object's shape by varying the values of predefined parameters, such as the diameter, thickness, etc. (routine design) [Brown and Chandrasekaran, 1986], [Mittal et al., 1986], [Mitchell et al., 1985]. These systems configure their designs from a library of existing elementary components that have been parameterized to reflect the important aspects of the design problem. When the design specifications require the consideration of an additional parameter, or the introduction (or modification) of a new elementary component, the design process fails. In order to modify or introduce a new component, the system must be capable of reasoning about the structure and the function of the component. A first approach to this problem is presented in [Murthy and Addanki, 1987] for the domain of structural beam design.

This paper presents a new method for designing shapes of objects that is capable of handling both incomplete and qualitative functional specifications of the desired behavior. Our method is based on previous work showing that configuration spaces are an appropriate representation for relating kinematic behavior and object geometry for mechanism analysis [Faltings, 1986; 1987], [Forbus et al, 1987], [Joskowicz, 1987a; 1987b; 1988].

2 Presentation of the Problem

Consider the following design scenario: we are given a rotating disc A and a translating rectangle B (Figure 1a). Our design goal is to modify the shapes of the objects so that for two specific orientations of A, 0 and $\pi/2$, B prevents the rotation of A. For all other orientations, the motions of A and B must remain independent. A possible solution is to modify the shape of A by introducing two slots that allow B to create new contacts that prevent the rotation of A (Figure 1b).

In the following, we assume that objects are two-dimensional, that their contours are formed by line segments and circular arcs, and that each object has at most one degree of freedom (either rotation or translation) along



(a) Initial Shapes (b) Modified Shapes

Figure 1: A Design Example

an axis fixed in the plane. We distinguish between five *design spaces*, corresponding to the degrees of freedom of each object in the pair: fixed-rotation, fixed-translation, translation-translation, rotation-translation and rotation-rotation.

3 Functional Specification of Kinematic Behavior

The kinematic behavior of a mechanism can be described in terms of *possible motions* or in causal terms [Joskowicz, 1987a]. Both descriptions are functional since they specify motion relationships between objects without referring to their actual geometry.

A possible motions description specifies all the possible motions that each object (represented by a reference point) can have, together with the relationships between these motions. Every degree of freedom is associated with a motion parameter. The relationships between motions are specified by a function relating motion parameters. Functions can be real-valued or qualitative, indicating whether the motion parameters' ratio is increasing, decreasing or constant. Each motion parameter is bounded by intervals that define its legal range. Since we assumed that objects are two dimensional and move on fixed axes, an object A can only have one of the following three types of motions: no motion ($fixed(A,p)$), rotation ($p\_rotation(A,O,\theta)$), or translation ($p\_translation(A,O,X)$).

Kinematic behavior can be described as the union of several possible motion *regions*. For example, all the reachable behaviors of the pair in Figure 1b are described as the union of three regions:

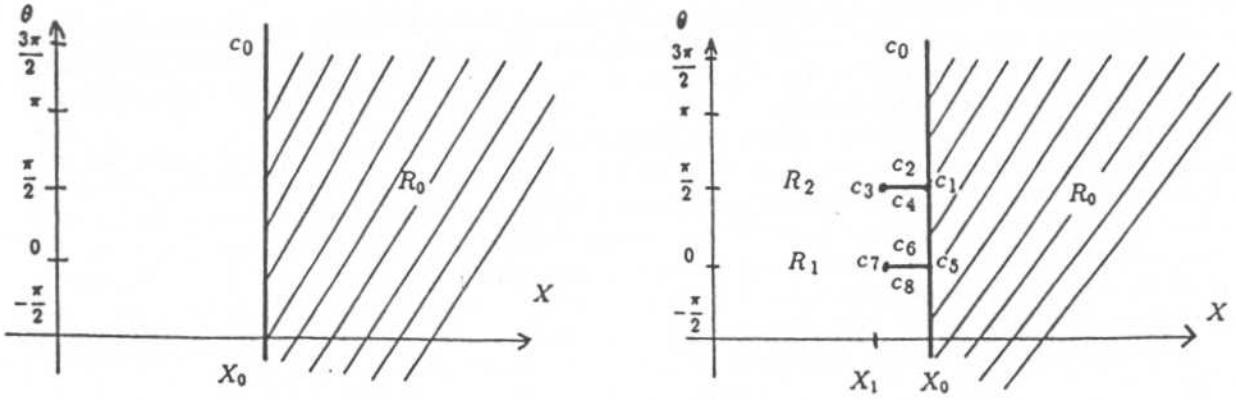
R_0: $p\_rotation(A,O_1,\theta)$, $p\_translation(B,O_2,X)$, for $\theta \in [0,2\pi]_{mod2\pi}$ and $X \in [X_0,\infty)$

R_1: $fixed(A,\theta)$, $p\_translation(B,O_2,X)$, for $\theta = 0$ and $X \in [X_1,X_0)$

R_2: $fixed(A,\theta)$, $p\_translation(B,O_2,X)$, for $\theta = \pi/2$ and $X \in [X_1,X_0)$

In a previous paper, we showed that there is a direct, one-to-one correspondence between possible motion descriptions and configuration spaces[1] [Joskowicz, 1987a]. Since each object has at most one degree of freedom, a two-dimensional configuration space fully describes the kinematic behavior of a pair of objects. Figure 2 shows the configuration space of the pair (A,B) before and after the modification. Note the direct correspondence between the above description and the regions of free object placements, indicated by hatched areas.

[1] The configuration space of a mechanism defines the set of *free placements* (position and orientations) of objects in a mechanism so that no two objects overlap [Lozano-Pérez, 1983], [Schwartz and Sharir, 1983].



Initial Space, $CO(A, B)$. Space After Modification, $R(A, B)$.

Figure 2: The Configuration Space Before and After the Modification.

An alternative description of kinematic behavior is a *causal* description. This description states the effects that the motion of one object has upon the others (e.g., if A rotates clockwise then B rotates counterclokwise). The kinematic behavior of a mechanism can then be described by the motions of its objects resulting from a sequence of input motions. Section 6 shows that causal descriptions can also be mapped into an equivalent configuration space specifying the desired behavior.

4 Shape Design from Configuration Space

We use configuration spaces as the basis of the design procedure. In this section, we assume that the desired pairwise behavior is given as a two-dimensional configuration space with exact boundaries.

Initially, we are given two objects, A, B (possibly empty), and a desired configuration space $R(A, B)$, corresponding to the desired kinematic behavior. The actual kinematic behavior of the objects corresponds to their actual configuration space, $CO(A, B)$. Comparing both the actual and desired behaviors amounts to comparing the two configuration spaces, $CO(A, B)$ and $R(A, B)$. The differences between them indicate where and how these behaviors differ. For example, in the previous design problem, the desired configuration space $R(A, B)$ contains two regions, R_1 and R_2, not present in $CO(A, B)$ (Figure 2).

The behavior of a kinematic pair can be modified by changing the boundaries of $CO(A, B)$ so that they match with the boundaries of $R(A, B)$. Boundaries of the configuration space are formed by the contact of two object features (a vertex, an edge, or an arc). Therefore, configuration space boundaries can be modified by removing contacts or introducing new ones. This in turn implies that the shape of the objects must be changed by adding and deleting edges and arcs to their contours. In the previous example, there are six configuration space boundaries, $c_2, c_3, c_4, c_6, c_7, c_8$, that must be added to $CO(A, B)$, and two that must be deleted (c_1 and c_5) to allow transitions from R_0 to R_1 and R_2 [2]. The design problem consists in finding a sequence of feature additions and deletions to the objects' contours so that the actual and the desired configuration space boundaries match and the set of non-kinematic design constraints are satisfied.

4.1 Configuration Space Boundaries

The form of the configuration space boundaries is determined by the design space and by the features that come in contact to create it. For example, arc-vertex or arc-edge contacts produce (when the center of the arc coincides with the center of rotation) a configuration space boundary that is a line, such as the boundary c_0 in

[2] Regions R_1 and R_2 are rectangles of width zero, and thus have four sides, two of which of zero length.

Figure 2a produced by the contact (a_0, b_0). We have classified the different types of boundaries that arise from the nine possible pairwise contacts in each of the five design spaces. The result is a table of elementary contacts that specifies, for each type of contact and design space, the type configuration space boundary produced, together with the set of equations that define it.

Given a desired configuration space boundary, the design task consists in finding a pair of object features that, when in contact, will create this boundary. Note that not every contact between features can produce a desired configuration space boundary. For example, in the rotation-translation space, a vertex-edge contact can never be used to produce a line boundary in $CO(A, B)$, since its boundary equation is not a line. In this case, only a vertex-arc or an edge-arc contact can produce the desired boundary. This means that arc a_0 cannot be substituted by a vertex and still produce the boundary c_0 when in contact with b_0. Thus, the type of the configuration space boundary can be used to determine which pair of features can, in principle, produce the boundary. Having determined the type of contact, we then find the precise coordinates of the features that create the boundary.

4.2 A Backtracking Algorithm for Shape Design

The design procedure starts by comparing the actual and the desired configuration spaces. The goal is to delete the configuration space boundaries of $CO(A, B)$ that do not match boundaries of $R(A, B)$ and to add to $CO(A, B)$ the boundaries that appear in $R(A, B)$ but not in $CO(A, B)$. Two boundaries match iff their form is identical and the free object placements lie on the same region.

For each boundary difference, a pair of object features to either delete or add the required boundary is selected. For a deletion, at least one of the features that contributed to the boundary creation must be deleted. For an addition, one or two new features must be created to produce the boundary. The type of features that produce the boundary in question is determined from the table of elementary contacts. For example, in order to delete c_1, either a_0 or b_0 must be deleted. In order to add c_2, it is sufficient to add the edge a_2 (but not an arc) since its contact with edge b_6 creates c_2.

In both cases of addition and deletion, there might be more than one candidate feature pair and thus a (nondeterministic) choice must be made. For example, c_3 can be created with the existing edge b_0 and a new edge a_4, or with a new arc b_9 and a new edge a_4. In this case, the first choice is preferred since it introduces fewer new features. After every object contour change, the configuration space $CO(A, B)$ is updated. If the new features violate a design constraint (except closed contour), the pair is rejected and a new candidate pair is selected. This guarantees that a bad choice is rejected as soon as a violation occurs, instead of waiting until the whole design process is completed. Note that the final designed objects might not be consistent, i.e., their contour might not be closed. For example, if we remove the edge b_0 from B, and take A as shown in Figure 1b, we still have that $CO(A, B) = R(A, B)$, although B does not have a closed contour An attempt to "fill in" the missing contours is made, without altering $CO(A, B)$. If this attempt fails, the algorithm backtracks over its previous choice. The design process is successful when all the differences between $CO(A, B)$ and $R(A, B)$ have been eliminated, and both objects are consistent with the design constraints. Figure 3 shows a backtracking algorithm that is design-space independent.

The analysis of feature contacts reveals that the equations relating a configuration space boundary c_i to the features that created it are underconstrained when only c_i is given. Thus, there is, in principle, an infinite number of coordinate values for features to create a new configuration space boundary, leading to an infinite number of feature choices. Nevertheless, for most of the interesting design cases, the number of choices is finite. When one of the objects (B) is not allowed to change, the number of possible choices of features of B that can participate in the creation of a new boundary is bounded by the number of features in B. Also, if only one new object feature is introduced at a time (to either A or B, but not both), the number of choices is bounded by the number of features of A and B. In both cases, the overall complexity of the algorithm is exponential in the number of choices. The actual running time of the algorithm is improved by incorporating two heuristics for choosing candidate features based on the adjacency properties of local object convexity.

For many special design cases, we were able to develop efficient design algorithms. For example, if we assume that both objects must be convex, the number of choices in each step is reduced to four, and the correct choice

Procedure DESIGN($A, B, R(A, B)$, Design-Constraints)

1. Compute $CO(A, B)$.
2. DELETE := boundaries in $CO(A, B)$ that do not match boundaries in $R(A, B)$.
   ADD := boundaries in $R(A, B)$ that do not match boundaries in $CO(A, B)$.
3. While $CO(A, B) \neq R(A, B)$ do
   - **3.1** For a boundary c_i in ADD, do
     - **a.** Using the table of elementary interactions, determine the type of features that can produce the type of boundary of c_i.
     - **b.** Choose a pair of features (a, b) of the appropriate type that produce c_i. Prefer pairs in which one of the features is already existing and is connected to the object boundary.
     - **c.** Check whether the new feature(s) comply with the design constraints.
   - **3.2** Update $CO(A, B)$, ADD and DELETE.
   - **3.2** For a boundary c_i in DELETE, choose a feature from the pair that created it and delete it from the corresponding object. Do not delete new features.
   - **3.3** Update $CO(A, B)$, ADD and DELETE.
4. End
5. Complete the object contours without modifying $CO(A, B)$. If this is not possible, return "FAIL".

Figure 3: Algorithm for Shape Design.

---

can be made in constant time. The result is a deterministic algorithm whose time complexity is linear in the size of $R(A, B)$. For the translation-translation space, all the design algorithms, including those dealing with non-convex objects, have polynomial time complexity [Joskowicz and Addanki, 1988].

5 Qualitative Shape Design

Up to now, we assumed that we either have, or can produce, an exact description of the desired configuration space. In some cases, such a precise description is not available, or not required.

Consider the following example: we are given a disk A that can rotate around axis O_1 and a rectangle B that can translate along axis O_2. Let θ and X be their rotation and translation parameters, respectively. Suppose we want, for a full rotation of A, B to slide up, then down, and then stay stationary. The precise relationship between X and θ is not important. We only require X to increase when θ increases for the intervals $X \in [0, X_0]$ and $\theta \in [0, \pi/2]$, and X to decrease when θ increases for $X \in [X_0, 0]$ and $\theta \in [\pi/2, \pi]$. For $\theta \in (\pi, 2\pi)$, X is to remain constant, $X = 0$. This description is not sufficient to produce an exact configuration space since the type of configuration space boundary in the first two regions is unknown. Indeed, *any* boundary is satisfactory as long as the qualitative relations between the parameters hold continuously in each region. Figure 4 shows a solution that meets these requirements. The given boundary points are matched exactly, but also new boundary points are introduced.

To design shapes from qualitative descriptions, we no longer require an exact boundary match between $CO(A, B)$ and $R(A, B)$. The matching requirement for qualitative boundaries is relaxed as follows: let S be a set of boundary segments of $CO(A, B)$. S matches a qualitative boundary defined by two given points P_1 and P_2 of $R(A, B)$ iff:

1. The boundary segments of S form a connected, piecewise differentiable boundary whose endpoints are P_1 and P_2.

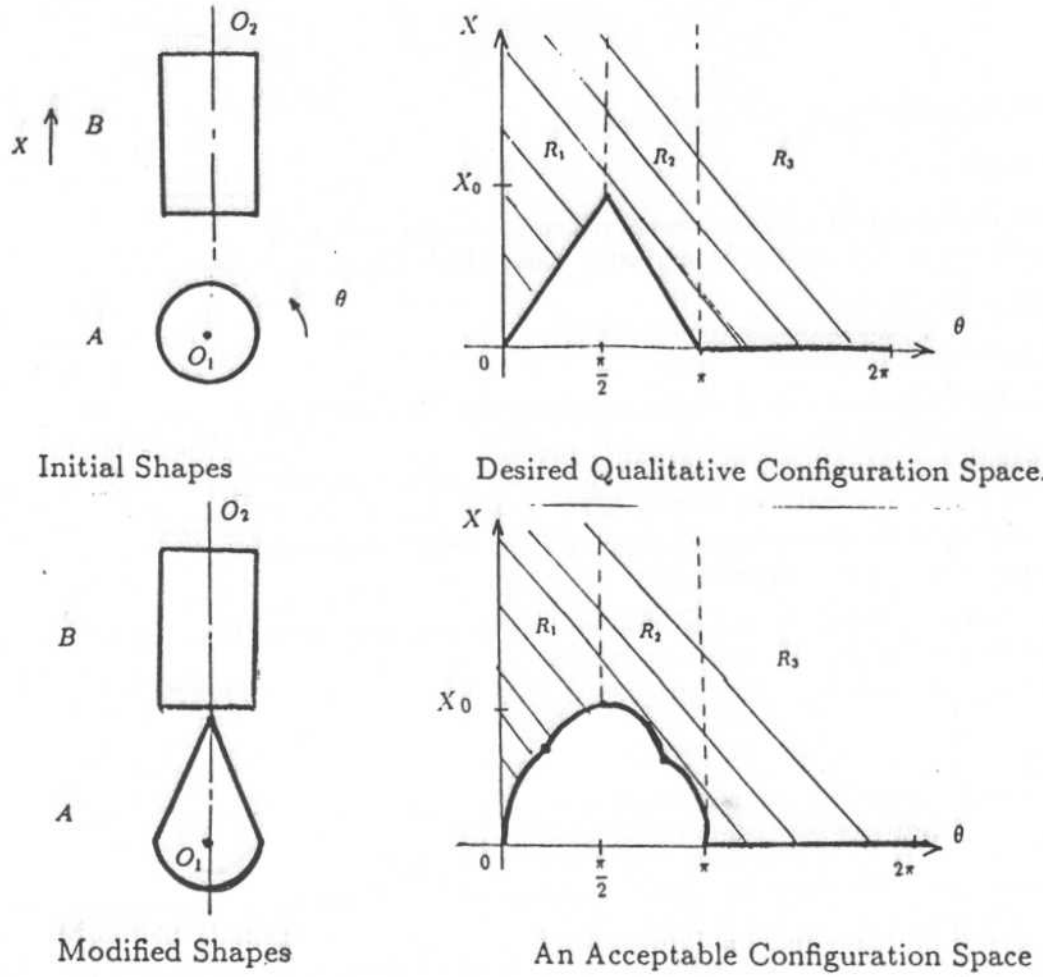


Figure 4: An Example of Qualitative Boundary Match.

2. Each boundary segment in S reflects the same qualitative change than the change from P_1 to P_2.

Qualitative boundaries broaden the number of choices for pairwise contacts in the backtracking algorithm. The elementary contact table is augmented with additional information, indicating the value range for which the configuration space boundary is monotonically increasing, decreasing, or constant. New boundary points are introduced only when all other choices fail. The boundary endpoints P_1 and P_2 must be matched precisely.

6 Causal Descriptions

In this section, we show how to obtain the configuration space corresponding to a given causal description. A causal description can be represented as a a collection of *state diagram* ([DeKleer and Brown, 1984], [Forbus, 1984]), where each state corresponds to a qualitatively different behavior. Two kinematic behaviors are said to be qualitatively different when they specify different possible motions, when the axes of motion are different, when at least two motion parameter intervals are disjoint, or when the functions relating motion parameters are different.

While possible motion descriptions specify all the potential kinematic behaviors of a mechanism, causal descriptions might only specify a subset of these behaviors. Indeed, a causal description can be interpreted

as either being a *partial* or a *complete* description of the desired behavior. Both descriptions require the described behaviors to take place, but the partial description allows additional qualitatively different behaviors. A complete description requires that no other qualitatively different behaviors take place. In both cases, the design is considered successful when the input motion sequences applied to the objects produce exactly the original state diagrams.

Let $S = \{S_1, \ldots, S_n\}$ be a collection of state diagrams, where each state diagram S_i is a triple $[\sigma_i, \{s_{ij}\}, \{< s_{ij}, s_{ik} >\}]$. σ is the input motion sequence, $\{s_{ij}\}$ is the set of states describing the motion of each object, and $\{< s_{ij}, s_{ik} >\}$ is the set of state transitions. The function $apply(\sigma, CO(A,B))$ produces the state diagram corresponding to the input sequence σ and the configuration space $CO(A,B)$ (the procedure to compute *apply* is described in [Joskowicz, 1987]). The shapes of A and B *satisfy* a given collection S of state diagrams iff:

$$\forall S_i \in S \ \wedge \ \forall \sigma_i \in S_i, \ \ apply(\sigma_i, \ CO(A,B)) = S_i$$

i.e., the application of each input motion sequence to the actual configuration space produces the same state diagram as the one desired one. A configuration space that satisfies the above property is *acceptable*. For a given set of state diagrams, the goal is to construct an acceptable desired configuration space, $R(A,B)$.

We construct $R(A,B)$ by composing individual configuration spaces $R_i(A,B)$ resulting from each S_i. The space $R_i(A,B)$ is in turn constructed by composing configuration space regions r_{ij} resulting from each state s_{ij}. Each state s_{ij} is mapped into a region of the configuration space by using the information contained in the state about object motions and their relationships:

1. The type of motions determines the design space.
2. The intervals of the motion parameters determine the region of the configuration space in which the behavior takes place.
3. The boundary of the configuration space is determined either by an explicitly given relation ($X_A \geq f(X_B)$), or deduced from the causal description that defines the instigator of the movement and the direction of change for the motion parameters: $motion(A)$ CAUSES $motion(B)$, $direction(X_A)$, $direction(X_B)$

The configuration space boundary resulting from a causal description is a qualitative boundary, whose endpoints are determined by the intervals of X_A and X_B. The region of free placements is determined by one of the eight possible combinations of values for $direction(X_A)$, $direction(X_B)$ and $motion(A)$ CAUSES $motion(B)$, as shown in Figure 5. For example, in the first case, the qualitative configuration space boundary is defined by the endpoints (X_1^A, X_1^B) and (X_2^A, X_2^B). The set of free placements corresponds to the region $X_B \leq f(X_A)$, where f is the equation of the boundary line.

The individual regions r_{ij} are combined by taking the union of their forbidden placements. Conceptually, composing two regions amounts to requiring two behaviors to take place in the interval common to the two regions, and preserving the behaviors in the disjoint subregions. The configuration spaces $R_i(A,B)$ resulting from each S_i are composed exactly as the individual regions r_{ij}. This method produces an acceptable configuration space $R(A,B)$ that has the least constraints on free placements.

Once $R(A,B)$ is found, we apply the design procedure described previously. If the causal description is assumed to be complete, we require a qualitative match between the all boundaries of $R(A,B)$ and $CO(A,B)$. Otherwise, we allow additional regions in $CO(A,B)$ not appearing in $R(A,B)$. Then, $R(A,B)$ matches $CO(A,B)$ iff there exist a set of regions $r_1, \ldots, r_n \subset CO(A,B)$ such that $R(A,B)$ matches $r_1 \cup \ldots \cup r_n$.

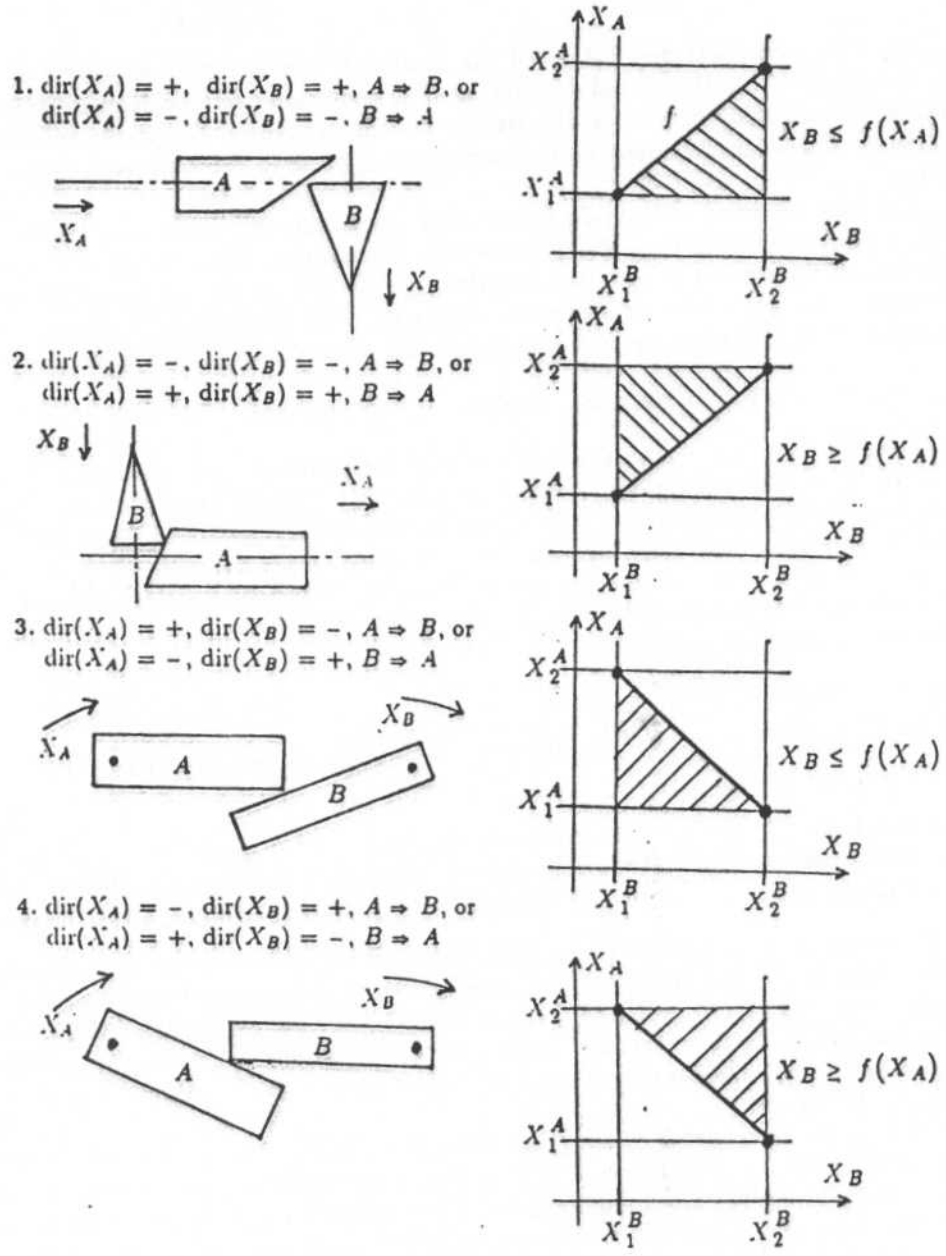


Figure 5: Causal Descriptions and their Corresponding Qualitative Configuration Space Regions.

7 References

Brown and Chandrasekaran, 1986 D. Brown and B. Chandrasekaran, "Knowledge and Control for a Mechanical Design Expert System", *Computer*, July 1986.

DeKleer and Brown, 1984 J. DeKleer and J. S. Brown, "A Qualitative Physics based on Confluences" *Artificial Intelligence* **24**, 1984.

Dixon, 1986 J. Dixon, "Artificial Intelligence and Design: A Mechanical Engineering View", *Proceedings of AAAI-86*

Faltings, 1986 B. Faltings, "A Theory of Qualitative Kinematics in Mechanisms", Report UIUCDCS-R-86-1274, University of Illinois, May 1986.

Faltings, 1987a B. Faltings, "Qualitative Place Vocabularies for Mechanisms in Configuration Space", Phd. Diss., Rep. UIUCDCS-R-87-1360, U. of Illinois, July 1987.

Faltings, 1987b B. Faltings, "Qualitative Kinematics in Mechanisms" *Proceedings of IJCAI-87*, Milano, Italy, 1987.

Forbus, 1984 K. Forbus, "Qualitative Process Theory" *Artificial Intelligence* **24**, 1984.

Forbus et al., 1987 K. Forbus, P. Nielsen and B. Faltings, "The Inferential Structure of Qualitative Kinematics", *Proceedings of IJCAI-87*, Milano, Italy, 1987.

Joskowicz, 1987a L. Joskowicz, "A Framework for the Kinematic Analysis of Mechanical Devices", Tech. Report 313, Courant Institute, New York University, August 1987.

Joskowicz. 1987b "Shape and Function in Mechanical Devices", L. Joskowicz, *Proceedings of AAAI-87.*

Joskowicz, 1988 L. Joskowicz, "Reasoning about the Kinematics of Mechanical Devices", to appear in *Artificial Intelligence in Engineering*, 1988.

Joskowicz and Addanki, 1988 L. Joskowicz and S. Addanki, "Innovative Shape Design for Kinematic Pairs" Technical Report 399, Courant Institute, New York University, March 1988.

Lozano-Pérez, 1983 T. Lozano-Pérez, "Spatial Planning: A Configuration Space Approach", *IEEE Transactions on Computers*, Vol C-32, No. 2, 1983.

Mitchell et al., 1985 T. Mitchell T., L. Stenberg and J. Shulman, "A Knowledge-Based Approach to Design" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1985.

Mittal et al., 1986 S. Mittal, C. Dym and M. Morjaria, "PRIDE: An Expert System for the Design of Paper Handling Systems" *Computer*, July 1986

Murthy and Addanki, 1987 S. Murthy and S. Addanki S, "PROMPT: An Innovative Design Tool", *Proceedings of AAAI-86.*

Reuleaux, 1876 F. Reuleaux, *The Kinematics of Machinery: Outline of a Theory of Machines*, 1876 (Reprinted by Dover Publications Inc., 1963).

Schwartz and Sharir, 1983 J.T. Schwartz and M. Sharir, "On the Piano Movers II. General Techniques for Computing Topological Properties on Real Algebraic Manifolds", *Advances in Applied Math. 4*, 1983.