A Representation Language for Conceptual Mechanism Design

Dorothy Neville
Computer Science and Engineering
University of Washington
Seattle, WA 98195

Leo Joskowicz
IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

June 23, 1992

Abstract

Automating conceptual mechanism design requires developing a representation language to support common design tasks such as analysis and validation. This paper describes a simple and expressive language for describing the structure and behavior of *fixed axes* mechanisms. The language uses a mixture of predicates and algebraic relations to describe the mechanism's parts, their positions, their motions, and the relationships between them. It allows both abstract, incomplete, and underspecified behavioral descriptions, and accurate and complete descriptions. With an example, we illustrate how the language naturally describes the kind of design specifications found in the conceptual and configuration stages of the design process. We show that the language captures the mechanism descriptions of a significant number of mechanisms from an engineering encyclopedia. We describe a partially implemented design verification algorithm that determines if a mechanism structure matches desired structural and behavioral specifications stated in the language.

1 Introduction

Automating conceptual mechanism design requires developing a behavior and structure description language to support common design tasks. The language should be descriptive enough to capture naturally the behavior of a significant class of mechanisms. It should be flexible enough to allow both precise and complete descriptions and abstracted, underspecified, and incomplete design specifications. It should have a computational basis, so that specific tasks such as analysis and validation can be automated.

Existing representation methods for design focus on relatively narrow and specialized mechanism classes: linkages, cams, and gearbox configurations, to name a few. General representation methods, such as bond graphs, only capture certain behavioral aspects and abstract away geometry altogether. Constraint languages are expressive enough but require complete specifications and are computationally intractable. Recent research has begun to address these issues, and some progress has been made [2, 3, 7, 9, 10]. However, we found that none of these approaches is entirely appropriate for representing a large class of everyday mechanisms, such door locks, staplers, and brakes.

In this paper, we develop a simple and expressive structure and behavior language for *fixed-axes* mechanisms. We first identify the requirements of such a language with an example. We then propose a concise representation language which uses a mixture of predicates and algebraic relations to describe parts' positions, motions, and their relationships. With the example, we illustrate how the language naturally describes the kind of design specifications found in the conceptual and configuration stages of the design process. We show that the language captures the mechanism descriptions of a significant number of mechanisms from an engineering encyclopedia. We then describe a design verification algorithm that determines if a mechanism structure matches desired structural and behavioral specifications stated in the language. We conclude by describing our current implementation efforts, a review of literature, and a discussion of future work.

1.1 Example

We motivate the requirements of a representation language with a simple example. Figure 1 shows an indexing mechanism used to position and lock a horizontal rack. It consists of a rack, a table, a plunger, a cam, a lever, a spring, and a frame. The rack is mounted on the table and is free to translate horizontally in either direction when the plunger is raised. When the plunger is lowered, it engages one of the rack's teeth, thereby preventing any further translation of the rack. The rack can be positioned at ten horizontally aligned, evenly spaced locations. The spring-loaded plunger is engaged and disengaged via a cam. The cam is mounted off-center on a fixed axis and has two stable positions, a disengaged position (shown in the figure) and an engaged position. The cam is activated by rotating the lever, which is permanently

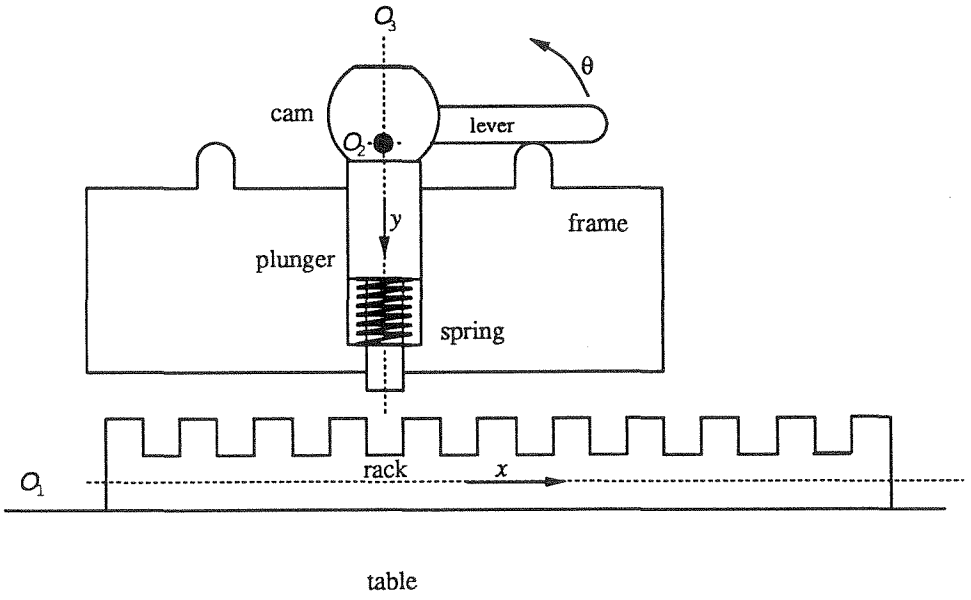

Figure 1: The indexing mechanism.

attached to the cam. The distance between rack's translation axis O_1 and the cam's rotation axis O_2 is 10cm.

We first observe that this description refers to both the structural and the behavioral characteristics of the mechanism. The distance between the axes, their relative positions, and the contacts between parts refer to structural properties and relations. The rack's translation, the cam pushing the plunger, and the rack positioning are behavioral statements. Behavioral statements further distinguish between part motions (the rack's translation), part motion relationships (the cam pushing the plunger), and part positions (the ten positions of the rack). Motion relationships are causal, indicating the effect of a part's motion on another part: the plunger is engaged by rotating the cam. The description contains both feasible and infeasible behaviors: the cam pushes the plunger; once the plunger engages the rack, the rack cannot translate.

We also note that the description is only a partial and simplified description of the mechanism. It describes certain behaviors, but not others: the description states what happens to the plunger when the cam is rotated, but not vice-versa. It only describes the behavior of a subset of parts: the cam, the plunger, and the rack, but not the spring. It does not specify the exact relation between the rotation of the cam and the translation of the plunger: it only states that the plunger goes down as the cam is rotated counterclockwise. It ignores altogether the transient behavior of the spring and the effects of friction.

The indexer is a prototypical example of the kind of mechanisms we want to cover: it has non-standard parts (the plunger), ' as multiple degrees of freedom (the cam

and the rack can move independently), has varying contacts (the plunger engages and disengages the rack's teeth), has multiple operating states (the rack can be locked and unlocked), and utilizes dynamical elements, such as the spring. Many practical mechanisms have these properties.

We contend that a conceptual design representation language for mechanisms should at least have the above characteristics to adequately capture design specifications. To summarize, the mechanism representation language should:

- distinguish between structural and behavioral specifications. Most design specifications describe the desired mechanism in terms of desired behaviors subject to small number of structural constraints. Lumping them together complicates the design process, precludes function sharing, and may unnecessarily overconstrain the resulting design.
- allow causal descriptions of both feasible and infeasible behaviors. Such descriptions are pervasive and naturally capture design intent.
- allow descriptions of behavior of a subset of parts. Design specifications almost always describe the desired behavior as a relation between input parts and output parts. The goal is to find the structure and behavior of the intermediate parts that achieve the desired relationship.
- allow descriptions of only a subset of all possible behaviors. Design specifications almost never exhaustively describe all the possible behaviors of the desired mechanism under all possible conditions. Rather, they describe the desired behaviors under the desired conditions.
- allow behavioral abstraction and simplification. Design specifications, especially in the early stages of conceptual design, are often underspecified or qualitative. They tend to group together sets of behaviors that will be examined in more detail and further differentiated later.
- allow descriptions of simple dynamical behaviors. Sophisticated dynamical models are seldom necessary for the conceptual design of common mechanisms. However, a simple account of dynamics is necessary to capture the action of gravity, friction, and springs.
- cover a broad and well-defined class of common mechanisms with non-standard parts, multiple operating states, multiple degrees of freedom, varying contacts and topology, and springs.

1.2 Our solution

The rest of this paper describes our proposed language for *fixed-axes* mechanisms, an important class of mechanisms mostly not covered by existing representation languages and design techniques. Parts in a fixed-axes mechanism can only translate,

rotate, or both, along fixed line axes. The indexer in Figure 1 is an example of a fixed-axes mechanism.

In our language, we distinguish between possible behaviors and actual behaviors. Possible behaviors describe all the behaviors that are physically possible for all inputs. Actual behaviors describe the behaviors that result from applying specific input motions to parts. Possible behaviors descriptions constitute an envisioning of the mechanism's behavior, while actual behaviors best describe the simulation of a mechanism under specific conditions. We represent possible behaviors with *region diagrams* [8], an annotated partition of a mechanism's configuration space into regions characterizing its operating modes. Because region diagrams are a concise and complete representation of mechanism behavior, they are appropriate for describing and analyzing possible mechanism behavior [6, 7, 8].

We develop a new language for describing actual behaviors resulting from input motions. The language is a mixture of predicate and algebraic relations and has separate components for describing structure and behavior. Structural statements specify the locations and spatial relations between axes, contacts between parts, etc. Behavioral statements specify motions of parts, relations between motions, and relations between motions and positions. They identify the different operating states of the mechanism at varying degrees of abstraction. Partial descriptions are captured by bounded or uninstantiated relations on the parts' motion parameters. The language is complete in the sense that it can accurately describe all the behaviors produced by fixed-axes mechanisms.

The proposed language has a sound computational basis. In section 3, we describe a design validation algorithm. It takes as input a specification of the mechanism's desired structure and behavior and an actual mechanism. It then determines if the mechanism satisfies the specifications. The algorithm validates the design specifications by matching them with the mechanism region diagram to determine if the actual motions are indeed possible.

2 A language for behavior and structure

Our language represents a mechanism as a set of parts, a set of behavioral descriptions, and a set of structural predicates. The set of parts is a partial or complete list of the parts comprising the device. The behavioral description is a set of statements about the parts' positions, motions, and their relationships. The structural description is a set of predicates about the parts' structure, contacts, and axes positions.

2.1 Parts and axes descriptions

Parts and axes are uniquely described by their name. Parts have associated with them motion axes, motion types, motion parameters and parameter bounds. Since

BEHAVIOR-DESCRIPTION ::= MOTIONSEQUENCE {,MOTIONSEQUENCE} →
MOTIONSEQUENCE {,MOTIONSEQUENCE}

MOTIONSEQUENCE ::= MOTION | SEQUENTIALMOTIONS | PARALLELMOTIONS

SEQUENTIALMOTIONS ::= MOTION {,MOTION}

PARALLELMOTIONS ::= [MOTION, MOTION {,MOTION}]

MOTION ::= SIMPLEMOTION | COMPLEXMOTION

SIMPLEMOTION ::= <OBJECT, SM-TYPE, AXIS, INITIALPOSITION, EXTENT, RELATIONS>

SM-TYPE ::= **Translate** | **Rotate** | **Screw** | **Translate-and-Rotate** | **Stationary** | **Hold**

EXTENT ::= AXISPARAMETER **by** AMOUNT

AMOUNT ::= REALVALUE | CONSTANT | VARIABLE | ***infinity***

COMPLEXMOTION ::= <OBJECT, SM-TYPE, CM-TYPE, AXIS,
For IDENTIFIER = VALUE **to** VALUE>
Begin MOTIONSEQUENCE **End**

CM-TYPE ::= **Alternates** | **WithDwell** | **AlternatesWithDwell**

Figure 2: Language for Behavior Specifications

we consider only fixed-axes part motions, this description is complete. For example,

<RACK, TRANSLATE, O_1, X, $[0, 10]$>
<LEVER, ROTATE, O_2, θ, $[0, \pi]$>

states that the rack can translate along axis O_1 with parameter X ranging from 0 to 10 and the lever can rotate around axis O_2 with parameter θ ranging from 0 to π.

2.2 Behavior descriptions

The motions of parts in a mechanism are described by one or more MOTIONSEQUENCES, which describe the motion of some (or all) parts in an operating region. Motions in motion sequences can be either simultaneous (denoted by square brackets) or sequential. In a fixed-axes mechanism, part motions can either be stationary, rotate, translate, or do both along a fixed axis in space. The motions can alternate, or have a rest period between alternations. Figure 2 shows the complete BNF specification of the language.

We distinguish between simple and complex motion types. Simple types of motion include TRANSLATE, ROTATE, SCREW, which indicates helical motion resulting from combined rotation and translation, ROTATE-AND-TRANSLATE which indicates independent rotation and translation, and two special types of "no motion" which take into account forces: STATIONARY and HOLD. STATIONARY indicates that the part does not move by itself either because it is not subject to any force or because its motion is blocked. HOLD indicates that the part is externally prevented from moving and is is used to maintain a part's position regardless of the forces acting on it.

Initial part positions are expressed as equations on motion parameters. They can specify single spatial locations ($\theta = \pi$) or sets of locations ($0 \leq X \leq 10$). The extent of the motion is specified by the amount that motion parameter changes. This amount can be a constant number, a constant symbolic number, an unknown amount (variable), or unbound. Relations between motions are expressed as a set of equations between motion parameters. Two motions are related iff their corresponding motion parameters are functionally dependent on each other. For example, the cam's rotation is related to the plunger's translation by the equation $x = f(c)$ (where f is a sinusoidal function). When disengaged, the plunger's translation is independent of the rack's translation. Equations can be linear or qualitative equalities and inequalities. We approximate nonlinear relations (such as the cam/plunger relation) by piecewise linear functions.

Complex motions capture the most common repetitive motions: alternation and dwell. We use macro-like forms allowing the expression of repeated motions in specific patterns. ALTERNATES indicates a constant change in the direction of motion, such as the motion of windshield wipers. WITHDWELL indicates a rest period in a constant direction motion such as stop-and-go motions. ALTERNATESWITHDWELL indicates an alternating motion with a dwell period in-between. The motion is repeated a number of times, determined by the FOR - TO BNF clause.

A *behavior* consists of a pair of motion sequences connected by an arrow, describing the attempted *input motions* and the resulting *actual motions*. This represents "what we try to move" versus "what actually moves and by how much". This distinction is a key property of the language and allows us to express many types of behaviors in a simple and clear manner, as shown with indexer example.

2.2.1 Desired behaviors of the indexer

The minimal behavior specification for the indexer is described by two behaviors, one for each of lever's the stable positions. First, we state that the rack cannot move when it is in any one of its ten locked positions:

$$\begin{aligned}&\text{<RACK, TRANSLATE, } O_1, \{x = i,\ i = 0, 1, \ldots, 9,\ \theta = \pi\},\ x \text{ BY } c, \{\} > \rightarrow \\ &\quad \text{<RACK, STATIONARY, } O_1, \{x = i,\ i = 0, 1, \ldots, 9,\ \theta = \pi\}, \{\}>\end{aligned}$$

The initial positions field shows that rack in one of the ten locked positions and the lever at its leftmost position ($\theta = \pi$), which corresponds to the locked configuration. Trying to move the rack by any amount c results in the rack remaining stationary. Translating the rack while the lever is in the unlocked position yields:

$$\begin{aligned}&\text{<RACK, TRANSLATE, } O_1, \{X = c_1,\ \theta = 0\},\ X \text{ BY } c_2, \{\}> \rightarrow \\ &\quad \text{<RACK, TRANSLATE, } O_1, \{X = c_1,\ \theta = 0\},\ X \text{ BY } c_2, \{\ 0 \leq c_1 + c_2 \leq 10\ \}>\end{aligned}$$

The rack moves by the extent given, provided that it remains within its range. These two behavioral descriptions succinctly describe the desired behavior of the indexer,

matching the informal description in Section 1.1.

2.2.2 Simple dynamics

The previous indexer description did not require dynamics, as it omitted the plunger. In general, more complete descriptions do require dynamics. We capture dynamics by specifying its effects in terms of motions. For example, to describe the effect of the spring on the plunger, we first create a part description for it:

<PLUNGER, TRANSLATE, O_3, Y, $[1,3]$>

In the device, a spring keeps the plunger in contact with the cam. We implicitly model the spring's actions through a behavior description in which initially the plunger is not in contact with the cam and and the attempted motion is STATIONARY:

<PLUNGER, STATIONARY, O_3, $\{Y = 1, \theta = 0\}$, {}> →
  <PLUNGER, TRANSLATE, O_3, $\{Y = 1, \theta = 0\}$, {Y BY 2}, {}>

In this description, the plunger initially is engaged in the rack (since Y has its minimum value of 1), while the lever is in its unlocked position ($\theta = 0$). The input motion STATIONARY indicates what happens when we leave the plunger in this initial position. The actual motion shows that the plunger will translate upwards until it contacts the cam.

2.2.3 Simultaneous motions

The examples so far have considered one motion at a time: either the rack will translate or it won't depending on the position of the lever. More generally, mechanisms exhibit simultaneous motions given a single input motion. We account for simultaneous motions with motion sequences that contain one or more motions sequentially or in parallel. For the indexer example, we can state that rotating the lever causes the plunger to move by the following behavior description:

<LEVER, ROTATE, O_2, $\{\theta = \pi, Y = 1\}$, {θ BY $-\pi$}, {}> →
  [<LEVER, ROTATE, O_2, $\{\theta = \pi, Y = 1\}$, {θ BY $-\pi$}, {}>
   <PLUNGER, TRANSLATE, O_3, $\{Y = 1\}$, {Y BY 2}, {}>]

Note that in this example we do indicate the actual relationship between the positions of the lever and plunger as they are changing, just that they change by the given extent during the same time interval.

Simultaneous actions are also necessary to express the input motions of many devices. In the example, we do not represent what would happen if motions were attempted on both the lever and the rack simultaneously. However some mechanisms rely on simultaneous input: opening a door with a doorknob requires rotating the knob around its axis and simultaneously holding it in position while opening the door. This action can be represented by a ROTATE knob motion followed by simultaneous

PREDICATES RELATING AXES:
(PARALLEL AXIS1 AXIS2)
(PERPENDICULAR AXIS1 AXIS2)
(INTERSECT AXIS1 AXIS2 {AT PT})
(PLANE AXIS1 AXIS2)
(COPLANAR PLANE1 PLANE2)
(SKEWED AXIS1 AXIS2)
(DISTANCE AXIS1 AXIS2 NUM)

PREDICATES RELATING PARTS:
(KEYED-TO-SHAFT PART SHAFT)
(FREE-ON-SHAFT PART SHAFT)
(IN-CONTACT PART1 PART2)
(IN-CONTACT-WHEN PART1 PART2 {POSITION RELATIONS})

GENERIC PART PREDICATES:
(GEAR PART RADIUS NUM-TEETH AXIS)
(LEVER PART LENGTH)
(LENGTH PART)
(WIDTH PART)

Figure 3: Structure Specifications

motions of HOLD knob and ROTATE door for both the input and actual motion sequences.

2.3 Structure

While the behavioral language is intended to be complete, the structural language is intentionally incomplete and open-ended. Parts can have virtually any shape and have any spatial relation with other parts. A catalog of geometric shapes, part features, and part spatial relations is clearly outside the scope of our research. Instead, we identified the most common structural predicates used in describing fixed-axes mechanisms: predicates relating the axes, predicates relating parts, and predicates constraining the sizes, shapes, and other characteristics of parts. Figure 3 lists a representative sample of predicates. We envisage the user to add component part types and structural predicates as necessary.

For example, when designing a transmission, the structural information needed is the relationship between the axes of rotation of the input and output shafts, whether they are parallel or perpendicular, and, if they do not intersect, how far apart they are. For the indexer, the minimal set of structural predicates relates the axes of motion of the rack and lever, the components that form the "external interface" of the device. We know that they are perpendicular and 10cm apart. This is represented by the predicates (PERPENDICULAR O_1 O_2) and (DISTANCE O_1 O_2 10)

2.4 Coverage

We empirically determined the appropriateness of the language by surveying about 2500 mechanisms from Artobolevsky's four-volume *Mechanisms in Modern Engineering Design* [1]. We chose the encyclopedia because of its size, uniform format, and comprehensiveness. It contains general-purpose, single-function mechanisms, such as couplers, indexers, and dwells which constitute the functional components of larger, specialized mechanisms, such as printing presses, mills, motion-picture cameras, and cars.

Our survey determined that about 35% of mechanisms are linkages, 22% are fixed-axes, and 9% are fixed-axes mechanisms connected by linkages. In addition, the I/O behavior of some linkages and many complex mechanisms is a fixed-axes behavior. This clearly shows that fixed-axes mechanisms constitute an important category. In addition, we found that 21% of mechanisms have at least one spring, that 30% have more than one degree of freedom, and that 18% have varying topology. More than half of the fixed-axes mechanisms have more than one operating state. Virtually all mechanisms have at least one non-standard part. This quantifies our claims on coverage. For a details on the survey, see [8, 11].

We selected a dozen examples from across the four volumes and examined the text accompanying each mechanism. We then reproduced the English description using our language, and compared the two. In all cases, we successfully managed to described the stated behaviors within our language.

3 Validation

To test the computational validity of the proposed language, we automated an important component of the design process: design validation. The design process starts with a structural and behavioral specification of the desired mechanism and produces a mechanism that exhibits the correct behaviors, abides by the structural constraints, and does not exhibit unwanted behaviors. Typically, the design process is incremental, whereby possible solutions are generated and need to be validated: the proposed mechanism has to be analyzed to determine its complete set of behaviors, and validated to verify that these true behaviors match the specifications. If they don't, modifications are necessary. Validation also determines if the device contains unexpected or undesirable behaviors, and submits tighter specifications if necessary.

Validation tests the behaviors and structure of the mechanism against the desired behavior and structural constraints. Structural verification is done by directly inferring the sizes, shapes and relative positions of objects and axes from the mechanism description and comparing them with the structural predicates. Validating the behavioral specifications requires matching the desired behaviors with the possible behaviors. To verify that a mechanism exhibits a specified behavior, we first analyze the mechanism to obtain its region diagram describing its possible behaviors. We then simulate the intended input motion through the region diagram, obtaining the

true motions. Finally, we compare the true motions with the actual motion sequence specified in the desired behavior.

In previous work, we describe a program that computes mechanisms' region diagrams [8] and a program that computes actual mechanism motions by simulation [11]. We are currently implementing a program in CLP(R) that automatically tests the behavioral specifications written in our representation against the region diagram of a completed design. We are using CLP(R) because it allows us to express the behaviors easily in predicate form and handles linear constraints. Given hand-coded or automatically derived region diagrams and intended input motions statements, it derives the resulting actual motions for the mechanism. The program currently validates kinematic pairs with one or two degrees of freedom, given a single input motion and a single initial position. We are working on incorporating dynamics and multiple pairs of objects.

4 Related work

Much recent work has focused on methods and representations for conceptual mechanism design. A classic work in linkage design is [4]. Freudenstein and Maki enumerate the kinematic structure of linkages in order to represent their functional properties abstractly. Conceptual design is viewed as the process of matching functional properties of the desired mechanism to potential structures. The representation and procedures are specific to linkages obeying specific mobility requirements.

Finger, Hoover, and Rinderle [3, 5] use a graph grammar based on bond graphs to represent behavior parametrically linked to geometry graph to represent structure. They propose a series of transformations to achieve the desired design. Both Kota [10] and Marshek [9] describe a representation scheme for machine behavior and a set of behavior transformation rules for design synthesis. Our language shares many common features with these languages, but is more comprehensive: all three are limited to single-state, fixed axis, fixed topology mechanisms.

Existing qualitative representations are also not fully adequate. Faltings' *place vocabulary* [2] is a qualitative kinematic representation which captures only one aspect of mechanism behavior, as Joskowicz' region diagrams

Since our primitive elements are not transformation tasks, but motions themselves, we can represent more behaviors; we can represent devices with more parts without knowing what all the kinematic pairs are; we can represent dynamics and motions being blocked due to positions.

5 Conclusions

In this paper, we present a new language for representing the behavior and structure of fixed-axes mechanisms. The language is flexible enough to describe partially or completely specified mechanisms and is expressive enough to capture salient aspects

of their kinematics and dynamics. We claim that our language captures descriptions of mechanisms at the right level of abstraction necessary for design. We describe behavior in a way that allows us to represent subsets of parts, subsets of behaviors, part positions that block other parts from moving, part motions that cause other parts to move, and the simple dynamics of a spring or gravity. By implementing a validation module as part of a design system, we show the relationship between the behavior descriptions and the region diagram representation of the behaviors of a completely specified mechanism.

Our long term goal is an automated design system for mechanisms consisting of fixed-axes and linkage subassemblies. Toward that end, we have developed a language to express design specifications and are working on an implementation of the validation component of design. Future work includes incorporating the analysis and simulation system of [11] with these results.

Acknowledgements

We thank Franz Amador and Dan Weld for many discussions and helpful comments on drafts of this paper. We also thank Elisha Sacks and Brian Williams for many fruitful discussions. Part of this work was conducted while the first author was a summer intern at IBM T.J. Watson Research Center. The first author was also funded in part by National Science Foundation Grants IRI-8902010 and IRI-8957302, Office of Naval Research Grant 90-J-1904, and a grant from the Xerox Corporation.

References

[1] I. Artobolevsky. *Mechanisms in Modern Engineering Design*, volume 1-4. MIR Publishers, Moscow. English Translation, 1979.

[2] B. Faltings. Qualitative Kinematics in Mechanisms. *Artificial Intelligence*, 44, 1990.

[3] S. Finger and J.R. Rinderle. A Transformational Approach to Mechanical Design Using a Bond Graph Grammar. In *First ASME Design Theory and Methodology Conference*, Montreal, Canada, September 1989.

[4] F. Freudenstein and E.R. Maki. The Creation of Mechanisms according to Kinematic Structure and Function. *Environment and Planning B*, 6:375–391, 1979.

[5] S. Hoover and J. Rinderle. A Synthesis Strategy for Mechanical Devices. *Research in Engineering Design*, 1(2), 1989.

[6] L. Joskowicz. Reasoning about the kinematics of mechanical devices. *International Journal of Artificial Intelligence in Engineering*, 4:22–31, 1989.

[7] L. Joskowicz. Mechanism comparison and classification for design. *Research in Engineering Design*, 1:149–166, 1990.

[8] L. Joskowicz and E. Sacks. Computational Kinematics. *Artificial Intelligence*, 51:381–416, 1991.

[9] S.M. Kannapan and K. M. Marshek. An Algebraic and Predicate Logic Approach to Representation and Reasoning in Machine Design. *Mechanism and Machine Theory*, 25(3):335–353, 1990.

[10] S. Kota. Qualitative Motion Synthesis. In *Proc. of the First Natl. Conf. on Applied Mechanisms and Robotics*, Cincinnati, 1989.

[11] E. Sacks and L. Joskowicz. Mechanism Simulation using Configuration Spaces and Simple Dynamics. Technical report, March 1992.