

A Representation Language for Mechanical Behavior

Leo Joskowicz

IBM T.J. Watson Research Center
Yorktown Heights, NY 10598, USA

Dorothy Neville

Computer Science and Engineering
University of Washington
Seattle, WA 98195, USA

Abstract

Mechanism design automation support requires developing a representation language for describing mechanism behavior. The language is necessary to specify design requirements, to describe existing mechanisms, and to catalog them for design reuse. This paper presents a simple and expressive language for describing the behavior of *fixed-axes* mechanisms. The language symbolically captures the important aspects of the kinematics and simple dynamics. It uses predicates and algebraic relations to describe the configurations and motions of each part of the mechanism and the relationships between them. It allows both accurate and complete descriptions and partial, abstract, and underspecified descriptions. We show that the language is computationally viable by describing how to automatically derive behavioral descriptions stated in the language from the mechanism structure. To test its usefulness, we describe a design validation program that determines if a given mechanism structure can produce desired behaviors stated in the language.

Key words: mechanisms behavior, representation language, mechanism classification and design, model-based reasoning, kinematics, dynamics.

In *Artificial Intelligence in Engineering* **10** (1996) 109-116.

Send all correspondence to: Leo Joskowicz, Institute of Computer Science The Hebrew University, Givat Ram, Jerusalem 91904, Israel. E-mail: josko@cs.huji.ac.il

1 Introduction

Current research in computer-aided mechanical design aims at producing software tools that will assist engineers in all the steps of the design process. These tools will improve the designers' productivity by allowing them to analyze, prototype, evaluate, modify, document, and store designs electronically. They will make mechanism design and evaluation faster, easier, more systematic and more intuitive. Supporting mechanism design automation requires developing formal representations for capturing the important properties of mechanisms and developing algorithms that use these representations to support the desired tasks.

One important aspect of mechanism design is reasoning about the shape and motion of its parts. Mechanisms perform their function by transforming motions via part interactions. The input motions, the part shapes, and the part contacts determine the output motions. The behavior of a mechanical assembly is a description of how its parts move and interact to achieve a desired function. For example, a gearbox achieves its function by changing the transmission ratio between the input and output shafts. Behavioral descriptions, determined by the kinematics and dynamics of the mechanism, are pervasively used in engineering practice in all steps of the design cycle.

Developing a language to represent mechanism behavior is necessary to specify design requirements, to describe known mechanisms, and to catalog them for design reuse. To support the wide variety of tasks associated with design, including design validation, comparison, cataloging, and case-based retrieval, the language should be expressive and should naturally capture the behavior of a significant class of mechanisms. It should allow both detailed, complete descriptions and partial, abstract, and underspecified descriptions. It should be computationally viable, so that tasks that generate and manipulate statements expressed in the language can be efficiently implemented.

In this paper, we present a simple and expressive language for describing the behavior of an important class of mechanisms: *fixed-axes mechanisms*. Parts in a fixed-axes mechanism rotate and translate along axes that are fixed in space and interact via intermittent or permanent contacts. This class includes many common mechanisms such as door locks, staplers, and transmissions. The language symbolically captures the important aspects of the kinematics and simple dynamics of the mechanism. It symbolically captures the important aspects of the kinematics and simple dynamics of the mechanism. It uses predicate and algebraic relations to describe the configurations (positions and orientations) and motions of each part of the mechanism and the relationships between them. It allows partial, multilevel abstract descriptions. It complements existing languages for describing linkage mechanism behaviors, the other main class of mechanisms.

To justify the proposed language, we first identify the key desired features of an effective representation language. We then evaluate the expressiveness of the language by showing that it appropriately describes the behavior of many mechanisms, based on a survey of 2500 mechanisms from an engineering encyclopedia. We show that the language is computationally viable by describing how to automatically derive behavioral descriptions stated in the language from the structure of the mechanism. To test its usefulness, we de-

scribe a design validation algorithm that determines if a given mechanism structure can produce desired behaviors stated in the language.

2 Example: an indexing mechanism

We motivate the requirements of a representation language with a simple but illustrative example. Figure 1 shows an indexing mechanism used to position and lock a translating rack. The mechanism consists of a horizontal rack sliding on the table, and a plunger, a cam, a lever, and a spring mounted on a fixed frame. When the lever is in its rightmost position (shown in the Figure), the plunger is raised and the rack is free to translate on the table. When the lever is in its leftmost position, the plunger is lowered and engages one of the rack's five slots, preventing it from translating. The plunger is spring-loaded and is activated by rotating an off-center cam attached to the lever. The axes O_1 , O_2 , and O_3 are all fixed and perpendicular to each other. The distance between the rack axis O_1 and the cam axis O_2 is 20cm. The translation interval of the rack is 15cm¹.

We observe that the above description refers both to structural and 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 translation of the rack, the cam pushing the plunger, and the rack positioning are behavioral statements: they refer to part configurations, motions, and their relationships. Motion relationships can be causal, indicating how the motion of one part affects the motions and configurations of other parts (e.g., the plunger is engaged by rotating the cam). The description contains both behaviors that can happen (the cam pushing the plunger) and behaviors that cannot happen (the plunger preventing the rack from translating).

We also note that the description is only a partial description of how the indexer works. It describes certain behaviors, but not others. For example, it indicates 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. However, the description is sufficient to understand how the device works. When used for design specification, it allows for a range of possible solutions.

3 Requirements of a representation language

Adequate behavioral descriptions depend on the information available and the task at hand. To be useful and expressive, an effective behavior representation language should:

¹Text and figure adapted from example 403, "Indexing device for a rack" [1]

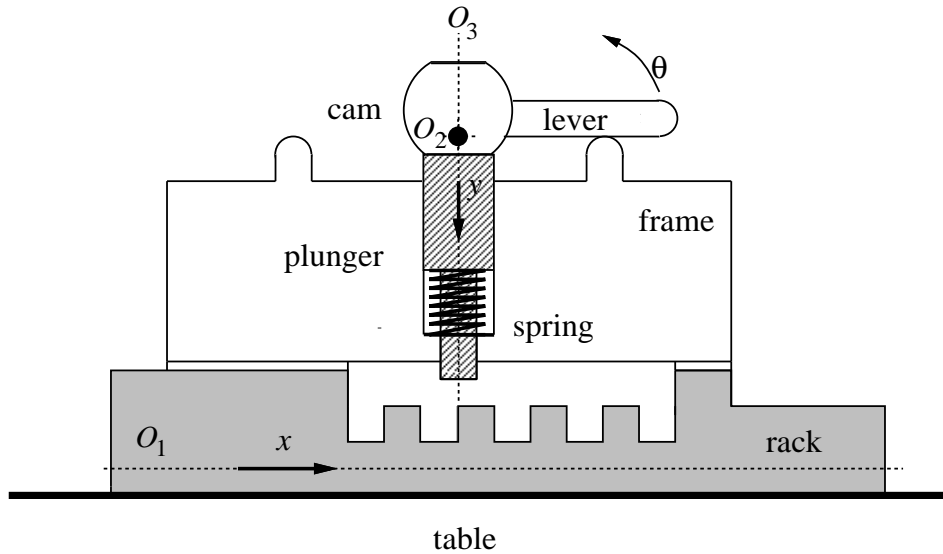


Figure 1: The indexing mechanism.

- distinguish between structural and behavioral information. Lumping them together complicates the design and retrieval process and may unnecessarily overconstrain the resulting design.
- be based on a mechanism ontology that relates behavioral statements to established engineering concepts. This makes the design specification and query formulation simple and intuitive.
- allow the causal description of both feasible and infeasible behaviors. Such descriptions are pervasive, naturally capture design intent, and improve retrieval efficacy.
- allow behavioral descriptions of a subset of parts. Desired behaviors and mechanism functions are frequently described behavior as an input/output relation between the configurations and motions of parts.
- allow partial descriptions of a subset of all possible behaviors. Design specifications almost never exhaustively describe all the possible behaviors under all possible conditions.
- allow behavioral abstraction, approximation, and simplification. Design specifications, especially in the early stages of design, are often qualitative, approximate, and underspecified.
- have a good computational basis. Statements in the language are intended for automatic processing.
- be useful for a variety of tasks, including design specification, explanation, comparison, and cataloging.

- allow descriptions of simple dynamical behaviors to account for the action of gravity, friction, and springs.
- cover a broad and well-defined class of common mechanisms.

Developing a language that meets all these requirements for a broad class of mechanisms is clearly a very difficult task. Based on a careful examination of mechanism encyclopedias and a survey of literature, we decided to focus on fixed-axes mechanisms. Parts in a fixed-axes mechanism rotate and translate along axes that are fixed in space and interact via contacts. The contacts can be permanent or intermittent, as in varying topology mechanisms. They can form lower (surface-surface) or higher (surface-line, surface-point, line-line, line-point) contact pairs. Parts can be standard (gears, bearings, etc.) or non-standard and have multiple degrees of freedom. Mechanisms can have multiple operating states and operating ranges, as in a clutch or a gearbox.

4 The language

We distinguish between three types of behaviors: possible behaviors, intended behaviors, and actual behaviors. Possible behaviors describe all the motions that are physically possible for all inputs. Intended behaviors describe motions applied to parts. Actual behaviors describe the motions resulting from applying specific input motions to parts. Since possible behavior descriptions characterize all the possible outcomes of intended motions, they constitute an envisioning of the mechanism's behavior [2]. They are defined by a set of constraints on part configurations, part motions, and their relationships. These constraints are described with *region diagrams*, an annotated partition of a mechanism's configuration space into regions characterizing its operating modes and the transitions between them (Section 5). Intended and actual behaviors describe a single behavior of the mechanism under specific conditions, like in a simulation. They capture the causal relations between the input and output motions. There is currently no language that fully covers all fixed-axes mechanisms (Section 6).

We develop a new language for describing intended and actual behaviors resulting from them. The language represents a mechanism as a set of parts, a behavioral description, and a set of structural predicates. The set of parts is a partial or complete list of the parts comprising the mechanism. The behavioral description is a set of statements about the configurations and motions of each part in the mechanism and the relationships between them. The structural predicates specify the parts' structure, contacts, and axes configurations and spatial relations. Statements are formulated using logical predicates and algebraic relations, as in constraint logic programming [7].

4.1 Part and axis descriptions

Parts are uniquely described by their name and have associated to them a motion type, an axis, motion parameters and their extents. Motion types indicate the motion of the

part along the axis (e.g. rotation, translation). Axes have a name and are defined by a three-dimensional vector and an origin defined with respect to a common global coordinate frame. They indicate the axis along which the part moves. Motion parameters indicate the configuration of the part along the axis with respect to its origin. The motion extent indicates the range of the motion parameter. For example, the description:

$$(\mathbf{Rack}, \mathbf{Translation}, O, x, 0 \leq x \leq 4)$$

indicates that the rack has a translational degree of freedom along axis O with motion parameter x ranging from 0 to 4.

4.2 Behavior descriptions

Table 1 shows the formal BNF (Backus-Naur Form) specification of the language. We informally describe it next, starting from the derivation at the top. A behavior description is a pair of causally related sequences of one or more motion sequences. The left sequence describes the input part motions and their starting configurations; the right sequence describes the resulting output part motions and their configurations. The arrow in between indicates the causal relation between them: the motions sequence on the left causes the motions sequence on the right. A motion sequence is composed of sequential and simultaneous motions of single parts. Sequential motions occur one after the other in the order indicated by the sequence. Simultaneous motions occur in parallel. The relationships between them are described by motion relations (see below).

The single motion clause contains the motion information associated with an individual part. It consists of a unique part name, motion type, axis, initial part configuration along the axis, and extent of the motion. The motion type can be a continuous motion along an axis: **Translation**, **Rotation**, **Helical** (coupled translation and rotation), simultaneous uncoupled **Translation-and-Rotation**, or no motion. We distinguish between two types of “no motion” to account for the effects of forces. **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 by another part. **Hold** indicates that the part is externally prevented from moving: it maintains its configuration regardless of the forces acting on it.

Repetitive motion patterns are expressed with a motion modifier. The most common are alternation and dwell. **Alternate** indicates a constant change in the direction of motion, such as the motion of windshield wipers. **With-Dwell** indicates a rest period in a motion with constant direction, such as stop-and-go motions. **Alternate-With-Dwell** indicates an alternating motion with a dwell period in-between.

Motion relations indicate the coupling between part motions. The coupling is achieved with equational constraints on the motion parameters associated with the parts. For example the rotation of the cam is related to the translation of the plunger by the equation $x = r.\sin(\theta)$, where x is the motion parameter of the plunger, θ is the motion parameter of the cam, and r is the radius of the cam. Equations can be algebraic, linear, or qualitative equalities and inequalities. The absence of constraints indicates that there is no relation between the part motions.

$\text{;BEHAVIOR-DESCRIPTION}_i$	$::=$	$\text{;MOTION-SEQUENCE}_i^+ \rightarrow \text{;MOTION-SEQUENCE}_i^+$
$\text{;MOTION-SEQUENCE}_i$	$::=$	$\text{;SINGLE-MOTION}_i \mid$ $\text{;SEQUENTIAL-MOTIONS}_i \text{ ;MOTION-RELATIONS}_i \mid$ $\text{;SIMULTANEOUS-MOTIONS}_i \text{ ;MOTION-RELATIONS}_i$
$\text{;SEQUENTIAL-MOTIONS}_i$	$::=$	$\text{;SINGLE-MOTION}_i \text{ then ;MOTION-SEQUENCE}_i$
$\text{;SIMULTANEOUS-MOTIONS}_i$	$::=$	$\text{;SINGLE-MOTION}_i \text{ and ;MOTION-SEQUENCE}_i$
;SINGLE-MOTION_i	$::=$	$(\text{;PART}_i, \text{;MOTION-TYPE}_i, \text{;AXIS}_i,$ $\text{;INITIAL-CONFIGURATION}_i, \text{;EXTENT}_i)$
$\text{;MOTION-RELATIONS}_i$	$::=$	$\text{INEQUALITY} \mid \text{EQUALITY}$
;PART_i	$::=$	PART-NAME
;MOTION-TYPE_i	$::=$	$\text{;MOTION}_i \mid \text{;MOTION}_i \text{ ;MOTION-MODIFIER}_i$
;MOTION_i	$::=$	Translation Rotation Helical Translation-and-Rotation Stationary Hold
$\text{;MOTION-MODIFIER}_i$	$::=$	Alternate With-Dwell Alternate-With-Dwell
;AXIS_i	$::=$	AXIS-NAME
$\text{;INITIAL-CONFIGURATION}_i$	$::=$	$\text{;MOTION-PARAMETER}_i = \text{;AMOUNT}_i \mid$ $(\text{;MOTION-PARAMETER}_i = \text{;AMOUNT}_i,$ $\text{;MOTION-PARAMETER}_i = \text{;AMOUNT}_i)$
;EXTENT_i	$::=$	$\text{;MOTION-PARAMETER}_i \text{ by ;AMOUNT}_i$
;MOTION-PARAMETER	$::=$	$\text{MOTION-PARAMETER-NAME}$
;AMOUNT_i	$::=$	$\text{REAL-VALUE} \mid \text{CONSTANT} \mid \text{VARIABLE} \mid \infty$

Table 1: BNF definition of a language for representing the behavior of fixed axes mechanisms. Symbols enclosed by brackets, e.g. ;MOTION_i are non-terminals. Bold symbols, e.g., **Translation** are terminals. Other symbols, i.e. **INEQUALITIES** stand for classes of terminals defined in a separate dictionary. $^+$ is an abbreviation for one or more symbols.

Initial part configurations specify the part configuration in terms of the value of the associated motion parameter. The amount can be a real value, a symbolic constant, a variable, or infinity. The extent of the motion is specified by the amount that the motion parameter changes. Positive or negative values indicate motion in the same or opposite direction as the vector defining the axis.

4.3 Structural predicates

To make the mechanism descriptions more complete, we complement the behavioral statements with structural predicates. Unlike the behavioral language, which is intended to be complete, the structural language is by nature 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 outside the scope of our research. Instead, we identified the most common structural predicates used in describing fixed-axes mechanisms: predicates relating axes (parallel, perpendicular, skewed, intersect, coplanar), predicates relating parts (keyed-to-shaft, free-on-shaft, in-contact-when), and predicates describing the size, shapes, and other characteristics of the parts (lever, gear, length).

4.4 Indexer descriptions

A succinct behavioral specification for the indexer is described by two behaviors, one for each of the lever's stable positions. Let x be the motion parameter of the rack and θ the motion parameter of the lever. First, we state that the rack cannot move when it is in any one of its five locked positions:

$$\begin{aligned}
 & \text{(Lever, Stationary, } O_2, \theta = 0, \theta \text{ by } 0) \\
 & \quad \text{and} \\
 & \text{(Rack, Translation, } O_1, x = 0 \dots 4, x \text{ by } c) \\
 & \quad \rightarrow \\
 & \text{(Lever, Stationary, } O_2, \theta = 0, \theta \text{ by } 0) \\
 & \quad \text{and} \\
 & \text{(Rack, Stationary, } O_1, x = 0 \dots 4, x \text{ by } 0)
 \end{aligned}$$

for any constant c , The initial configuration indicates that rack in one of the five locking configurations and the lever at its leftmost position $\theta = 0$, 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:

$$\begin{aligned}
 & \text{(Lever, Stationary, } O_2, \theta = \pi, \theta \text{ by } 0) \\
 & \quad \text{and} \\
 & \text{(Rack, Translation, } O_1, x = c_1, x \text{ by } c_2) \\
 & \quad \rightarrow
 \end{aligned}$$

(Lever, Stationary, $O_2, \theta = 0, \theta$ by 0)
and
(Rack, Translation, $O_1, x = c_1, x$ by $\max(c_2, 4 - c_1)$)

for any constant $0 \leq c_1 \leq 4$, This description matches the informal description of the rack's behavior in Section 2.

We capture dynamic behavior by specifying its effects in terms of motions. For example, the 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 the attempted motion is **Stationary**:

(Plunger, Stationary, $O_3, y = 1, y$ by 0)
and
(Lever, Stationary, $O_2, \theta = 0, \theta$ by 0)
 \rightarrow
(Plunger, Translation, $O_3, y = 1, y$ by 2)
and
(Lever, Stationary, $O_2, \theta = 0, \theta$ by 0)

In this description, the plunger initially is engaged in the rack ($y = 1$) and the lever is in its unlocked position ($\theta = 0$). The input motion **Stationary** indicates the plunger's motion is unobstructed. The actual motion shows that the plunger will translate upwards until it contacts the cam.

We account for simultaneous motions with motion sequences that contain one or more motions sequentially or in parallel. For example, we can state that rotating the lever causes the cam and plunger to move:

(Lever, Rotation, $O_2, \theta = \pi, \theta$ by $-\pi$)
and
(Cam, Rotation, $O_2, \psi = \pi, \psi$ by $-\pi$)
and
(Plunger, Stationary, $O_3, y = 1, y$ by 0)
 \rightarrow
(Lever, Rotation, $O_2, \theta = \pi, \theta$ by $-\pi$)
and
(Cam, Rotation, $O_2, \psi = \pi, \psi$ by $-\pi$)
and
(Plunger, Translation, $O_3, y = 1, y$ by 2)

Note that this description omits the actual relationship between the positions of the lever and plunger: it simply states that they change simultaneously by the specified extent during the same time interval.

5 Coverage and expressive power

To empirically determine the expressiveness and coverage of the language, we surveyed 2500 mechanisms and their descriptions in Artobolevsky's encyclopedia of mechanisms [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. These mechanisms constitute the functional components of larger, specialized mechanisms, such as printing presses, mills, motion-picture cameras, and cars. The encyclopedia includes a drawing of the mechanism and a short explanatory text. A representative example from the encyclopedia is shown in Figure 2.

Our survey determined that about 35% of mechanisms are linkages, 22% are fixed-axes, and 9% are fixed-axes subassemblies connected by linkages. In addition, the I/O behavior of some linkages and many complex mechanisms is described as a fixed-axes behavior. We also found that 21% of mechanisms have at least one spring, that 30% have more than one degree of freedom, and that 18% have intermittent contacts and 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 clearly shows that fixed-axes mechanisms constitute an important category and quantifies our claims on coverage. For a details on the survey, see [9, 17].

We selected a dozen representative examples from the encyclopedia and examined the text accompanying each mechanism. We then reproduced the English description using our language, and compared the two. In all cases, the two descriptions described the same behavior. We also successfully described the behavior of several mechanisms found in common appliances, such as the release mechanism of a 35mm camera. This provides support to our claim that the language is adequately expressive.

Figure 2: Example of a mechanism and its description in Artobolevsky's encyclopedia.

6 Computation with the language

To show that the language has a sound computational basis, we addressed two tasks: (1) the automatic derivation of behavioral descriptions stated in the language from a description of the mechanism structure, and (2) the validation of design requirements to determine if a given mechanism can produce a set of desired behaviors.

Mechanism behavior is determined by kinematics and dynamics. Kinematics enforces the constraint that no two parts can overlap in space. Dynamics determines the effects of forces on parts. The kinematics of a mechanism is fully characterized by its *configuration space*. The configuration space of a mechanism is the space of configurations (positions and orientations) of its parts. The dimension of the configuration space equals the number of degrees of freedom of the parts. For example, the configuration space of a gear pair is a two-dimensional configuration space because each gear has one rotational degree of freedom. The gear orientations provide a natural coordinate system. The configuration space partitions into free space where the parts do not touch and into blocked space where some parts overlap. The common boundary contains the configurations where some parts touch without overlap. Only free space and contact space are physically realizable. Part motions correspond to paths in the free space of the configuration space. A motion is a continuous function specifying the configuration of parts at every instant.

Configuration spaces provide a uniform geometrical model of kinematic function that is concise, complete, and explicit. They can be computed efficiently for most mechanisms consisting of fixed-axes and linkage subassemblies [9]. Dynamics, on the other hand, requires solving sets of differential equations and does not have, in general, a compact and computable representation. However, in most cases a simplified (algebraic) version of

dynamics is sufficient to describe and understand the behavior of mechanisms. It is this kinematic and simple dynamics model that the language presented in the previous section captures, and the one that is amenable to efficient computation.

6.1 Generating behavioral descriptions

Behavioral descriptions can be obtained either directly from the configuration space of the mechanism or from symbolic simulation. In previous work, Joskowicz has developed a set of operators that simplify and abstract kinematic descriptions derived from configuration spaces and from simulations [8]. Simplification operators discard irrelevant information by incorporating constraints and assumptions. Abstraction operators discard detail by defining multiple levels of resolution. The mechanism behavior representation language is based on this work.

Joskowicz and Sacks have developed an efficient kinematic analysis program for mechanisms consisting of fixed-axes and linkage mechanisms [9]. The input to the program is a geometric description of its parts (boundary representation) and their initial configuration. The program computes the configuration space of the mechanism by identifying its degrees of freedom, analyzing its linkage and higher pair subassemblies individually, and composing the results. The program outputs a region diagram, an annotated partition of a mechanism's configuration space into regions characterizing its operating modes and the transitions between them. The region diagram is a compact, symbolic representation of the possible behaviors of the mechanism.

Sacks and Joskowicz have also developed a simulation program that derives the actual behavior of a mechanism for given driving motions [17]. The program traces the configuration space path that the mechanism traverses under the driving motions, constructing the configuration space regions along the path. It simulates external forces and frictions using a simple model of dynamics that captures their steady-state effect without the conceptual and computational cost of dynamical simulation. It outputs a concise, symbolic interpretation of the simulation and a realistic, three-dimensional animation. The program runs at interactive speeds on moderate size mechanisms and have been tested on dozens of examples.

The descriptions produced by the simulation program constitute a proper subset of the representation language described in this paper. They identify part configurations, motions, and relationships between motions, and group them into regions of behavior. Missing are further aggregation and abstraction. For example, the behavior of the indexer mechanism is described with six behaviors, one for each of the rack's teeth engaging the plunger, and one for the disengaged position. A more compact description has groups the five behaviors into a single parameterized behavior, as described previously. Some of the simplification and abstraction operators developed by Joskowicz [8] can be applied to the output of the simulation to produce the high level behavioral descriptions defined by the language.

6.2 Design validation

The goal of design validation is to determine if a given mechanism structure can produce a set of desired behaviors. The inputs are a description of the mechanism structure and a specification, possibly partial, of its desired behaviors. The algorithm determines if the structure can produce the desired behaviors.

To verify that a mechanism exhibits a specified behavior, we analyze the kinematic pairs of the mechanism to obtain the possible behaviors of each pair, described by the region diagram describing its possible behaviors. The region diagrams are obtained by partitioning the configuration space of the kinematic pairs. We then simulate the intended input motion through the region diagram, obtaining the actual motions. Finally, we compare these motions with the actual motion sequence specified in the desired behavior.

We have implemented the validation algorithm in CLP(R) [7], a Prolog-like language that handles logic propositions (Horn clauses) and linear constraints. The language is ideally suited for the task because it supports predicate unification and linear constraint satisfaction. The validation program matches the behavioral specifications written in the representation language and the region diagram of a given mechanism. Given hand-coded or automatically derived region diagrams and intended input motions statements, the program derives the resulting actual motions for the mechanism stated in the language. The program currently validates kinematic pairs and mechanisms with one or two degrees of freedom.

7 Related work

Two general-purpose representation languages are constraint languages and bond graph representations. Constraint languages can be used to describe a mechanism with a set of algebraic and differential constraints on its behavioral and structural parameters. While expressive, they require the user to identify key parameters and define the proper approximations and abstractions. They are in general computationally intractable. Bond graph theory provides a systems dynamics description of mechanisms in terms of elementary behavioral building blocks [13]. It supports behavior simulation, abstraction, and approximation, but focuses only on dynamics and abstracts away geometry and kinematics altogether. Recent work describes a bond graph based representation language for gearboxes and define a behavior-preserving associated transformation scheme for design [4, 6].

Freudenstein and Maki present a classification scheme for linkage function based on kinematic structure [5]. Linkage structure is described as a graph whose nodes are joints and links are parts connecting the joints. Linkage kinematic structure descriptions are useful for enumerating the linkage design space and determining the linkage degrees of freedom, but provide a single level of structural abstraction and do not address most of the requirements identified earlier in the paper. Shrobe developed a program that generates explanations of how linkages work by simulating them and identifying key features of their behavior [20]. The description language explains linkage behavior based on qualitative features of the curve shapes derived by driving linkages. The language provides for mul-

tilevel, causal, abstract, descriptions and could serve as the basis for indexing and design specification.

Kota and Chiou describe a symbolic, qualitative mechanism behavior description language based on mechanism building blocks and their associated qualitative constraint matrices [14]. They developed a symbolic matrix algebra to compose behaviors, and devised a mechanism design strategy based on it. Kannapan and Marshek describe an algebraic and predicate logic language to describe mechanism structure and behavior [12]. Both languages are restricted to single mode, single degree of freedom, permanent contact mechanisms. Our language shares many common features with these languages, but is more comprehensive.

Configuration spaces provide a useful basis for developing mechanism behavior description languages. Regions of the configuration space correspond to behavioral modes, and region adjacencies correspond to transitions between modes. Faltings' place vocabularies [3] and Joskowicz' region diagrams [8] implement this idea. Their main drawback is that qualitative representations derived from configuration spaces are typically very detailed. Murakami and Gossard describe a higher pair retrieval method based on qualitative configuration space descriptions [16].

8 Conclusion

Our long term goal is to develop computer-aided design tools to support mechanism design. In this paper, we present a language for representing the behavior of fixed-axes mechanisms. The language is necessary to specify design requirements, to describe existing mechanisms, and to catalog them for design reuse. The language is simple and expressive. It uses predicates and algebraic relations to describe configurations and motions of each part of the mechanism and the relationships between them. It allows both accurate and complete descriptions and partial, abstract, and underspecified descriptions. We show that the language is computationally viable by describing how to automatically derive behavioral descriptions stated in the language from the mechanism structure. We are currently compiling examples and building a mechanism catalog with descriptions stated in the language. We are also investigating the role of the language in conceptual mechanism design [15].

The mechanism behavior representation language is part of a broader effort to support a variety of tasks that require reasoning about shape and motion in mechanisms. These include automated modeling, mechanism analysis, and mechanism simulation [9, 17], parametric mechanism design [10], and tolerancing [11]. We have implemented an interactive problem solving environment, called HIPAIR [19] that integrates these capabilities. The core of HIPAIR is a module that automates the kinematic analysis of mechanisms composed of linkages and higher pairs. This module provides the computational engine for all tasks, including kinematic simulation, analysis, tolerancing, and parametric design. It is comprehensive, robust, and fast. Currently, HIPAIR handles planar linkage and fixed-axes mechanisms formed of higher pairs with two degrees of freedom. We plan to incorporate

a module that will automatically generate statements in the language describe here in the future.

8.1 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 Dorothy Neville was a summer intern at IBM T.J. Watson Research Center. She 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] Artobolevsky, I., *Mechanisms in Modern Engineering Design*, volume 1–4. MIR Publishers (English translation), Moscow, 1979.
- [2] Bobrow, D., *Qualitative reasoning about physical systems*. MIT Press, Cambridge, MA. 1985.
- [3] Faltings, B., Qualitative kinematics in mechanisms. *Artificial Intelligence*, 1990, **44**(1–2).
- [4] Finger, S. and Dixon, J., A review of research in mechanical engineering design. part I: Descriptive, prescriptive, and computer-based models of design processes. *Research in Engineering Design*, 1989, **1**(1).
- [5] Freudenstein, F. and Maki, E. R., The creation of mechanisms according to kinematic structure and function. *Environment and Planning B*, 1979, **6**.
- [6] Hoover, S. and Rinderle, J., A synthesis strategy for mechanical devices. *Research in Engineering Design*, 1989, **1**(2).
- [7] Jaffar, J. et al., The CLP(R) language and system. *ACM Transactions on Programming Languages and Systems*, 1992.
- [8] Joskowicz, L., Mechanism comparison and classification for design. *Research in Engineering Design*, 1990, **1**(4).
- [9] Joskowicz, L. and Sacks, E., Computational kinematics. *Artificial Intelligence*, 1991, **51**.
- [10] Joskowicz, L. and Sacks, E., Configuration space computation for mechanism design. *Proc. of the IEEE International Conference on Robotics and Automation*. IEEE Computer Society Press, 1994.

- [11] Joskowicz, L., Sacks, E., and Srinivasan, V., Kinematic Tolerance Analysis. *Proc. of 3rd ACM Symposium on Solid Modeling and Applications*, Utah, ACM Press, 1995.
- [12] Kannapan, S. and Marshek, K., An algebraic and predicate logic approach to representation and reasoning in machine design. *Mechanism and Machine Theory* 1990, **25**(3).
- [13] Karnopp, D., Margolis, D., and Rosenberg R., *System Dynamics*. John Wiley and Sons, Inc., 1990.
- [14] Kota, S. and Chiou, S., Design representation and computational synthesis of mechanical motions. *Proc. of the 4th International Conference on Design Theory and Methodology*. ASME Press, 1992.
- [15] Neville, D. and Weld, D., Innovative design as systematic search. In *Proc. of the 11th National Conference on Artificial Intelligence*, Washington DC, 1993.
- [16] Murakami, T. and Gossard, D., Mechanism concept retrieval by behavioral specification using configuration space. In *Proc. of the 4th International Conference on Design Theory and Methodology*. ASME Press, 1992.
- [17] Sacks, E. and Joskowicz, L., Automated modeling and kinematic simulation of mechanisms. *Computer-Aided Design*, 1993. **25**(2).
- [18] Sacks E. and Joskowicz, L., Computational kinematic analysis of higher pairs with multiple contacts. *ASME Journal of Mechanical Design*, to appear, 1995.
- [19] Sacks E. and Joskowicz, L., Mechanism Design and Analysis Using Configuration Spaces. *Proc. of the 9th World Congress on the Theory of Machines and Mechanisms*, Milano, Italy, 1995.
- [20] Shrobe, H., Understanding linkages. *Proc. of the 11th National Conference on Artificial Intelligence*, Washington DC, 1993.