Incremental Configuration Space Construction for Mechanism Analysis

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

Elisha Sacks
Computer Science Department
Princeton University
Princeton, NJ 08544

Abstract

We present an incremental configuration space (CS) construction algorithm for mechanisms described as collections of subassemblies of rigid parts. The inputs are the initial subassembly configurations and the subassembly CSs partitioned into *uniform motion regions* in which part contacts are constant and motions are monotonic. The output is a partition of the mechanism CS into uniform motion regions. The algorithm optimizes CS construction by incrementally enumerating and testing only the regions reachable from the initial configuration. We implement the algorithm for subassemblies whose uniform motion regions are polyhedral or are of dimension two or lower. The program constructs the exact CS when possible and an approximate CS otherwise. The approximate CS usually is qualitatively correct and in good quantitative agreement with the true CS. The program covers most mechanisms composed of linkages and fixed-axes kinematic pairs, two subassembly types for which CS construction programs are available.

Introduction

This paper describes part of an implemented kinematic analysis algorithm for mechanisms composed of rigid parts, such as door locks, gearboxes, and transmissions. Kinematic analysis determines the constraints on the workings of a mechanism imposed by the shapes of its parts and by the contacts among them. It derives qualitative and quantitative information about the mechanism's behavior and provides the computational basis for automating many mechanical engineering tasks such as kinematic simulation, design generation and validation, and catalog construction.

Deriving the kinematics of a mechanism entails examining every potential interaction among its parts, an intractable task even for mechanisms with few parts. Engineers simplify the task by decomposing the mechanism into subassemblies with simpler part interactions, analyzing the subassemblies, and composing the results [Joskowicz, 1989a, Reuleaux, 1963]. The most common subassemblies are sets of parts linked by permanent joints, called *linkages*, and sets of parts that move along fixed spatial axes, called *fixed-axes mechanisms*. Fixed-axes mechanisms decompose further into pairs of interacting parts, called *kinematic pairs*. Previous research provides efficient analysis algorithms for linkages [Haugh, 1984, Kramer, 1990] and for planar kinematic pairs [Brost, 1989, Faltings, 1990, Lozano-Pérez, 1983], but provides only a qualitative composition algorithm for fixed-axes mechanisms with one degree of freedom per part [Nielsen, 1988]. The algorithm cannot handle many useful mechanisms, including gearshifts, differentials, and indexers. Other research extracts partial kinematic descriptions of mechanisms from numerical simulations of their dynamics [Gelsey, 1989]. Simulations are potentially expensive and cannot guarantee complete descriptions.

We have developed a kinematic analysis program for fixed-axes mechanisms [Joskowicz and Sacks, 1990]. The program employs the standard configuration space (CS) representation used in mechanical engineering. The inputs are the part shapes and initial placements. The output is a partition of the mechanism CS into *uniform motion regions* in which part contacts are constant and parts move monotonically along fixed axes (for example clockwise rotation). The uniform motion regions describe the operating modes of the mechanism. The mechanism switches modes when its configuration crosses between regions. The program represents the partition with a *region diagram* whose nodes describe the uniform motion regions and whose links specify region adjacencies. Fig. 1 summarizes the three main steps of the program.

In this paper, we discuss the composition step. We present a composition algorithm for general subassemblies. The inputs are the subassembly CSs partitioned into uniform motion regions. The output is the mechanism region diagram. The algorithm optimizes CS construction by incrementally enumerating and testing only the regions reachable from the initial mechanism configuration. We implement the algorithm for subassemblies whose CSs are polyhedral (defined by lin-

*Authors listed in alphabetical order. Elisha Sacks is supported by the National Science Foundation under grant No. IRI–9008527 and by an IBM grant.

Input: Part shapes and initial part placements.

1. Identify motion axes and interacting pairs of parts.
2. Construct CSs for the interacting pairs.
3. Compose the pairwise CSs.

Output: Region diagram, a partition of the CS.

Figure 1: CS construction algorithm.

ear inequalities) or of dimension two or lower. The program constructs the exact CS when all the subassembly regions are polyhedral and an approximate CS otherwise. The approximate CS usually is qualitatively correct and in good quantitative agreement with the true CS. Like other qualitative reasoning techniques, it can contain unrealizable behaviors, but cannot miss true behaviors. We demonstrate the program on a two-speed transmission and assess its coverage by surveying 2500 mechanisms from a mechanical engineering encyclopedia. The program covers roughly 2/3 of the mechanisms, including most composed of linkages and fixed-axes subassemblies. We conclude by sketching a composition algorithm for general mechanisms.

Kinematic analysis of a transmission

We motivate the kinematic analysis of mechanisms with a realistic engineering example: a fixed-axes, two-speed transmission. Fig. 2 shows a side view of the transmission. The input shaft $S1$, the output shaft $S2$, and the gearshift P are mounted on the fixed frame F. Gears $G1$ and $G2$ are mounted on $S1$ and rotate freely around it. Engager E is mounted on a square section of $S1$ and translates along axis $o1$. Gears $G3$ and $G4$ are rigidly attached to $S2$. The engager E has six lateral teeth on each side that can engage with the six lateral teeth of $G1$ and $G2$. Input shaft $S1$ drives output shaft $S2$ via $G1$ and $G3$, drives $S2$ via $G2$ and $G4$, or does not drive $S2$ depending on whether the gearshift P is in the *low, neutral,* or *high* setting. The three settings of P define the three operating modes of the transmission. The gear ratios $G1/G3$ and $G2/G4$ define the transmission rates in the *low* and *high* modes.

Step 1 of the kinematic analysis program finds the motion axes and motion types of the parts and assigns a coordinate to each motion. For example, x_E and θ_E measure the translation and rotation of E along $o1$. The program finds the interacting kinematic pairs by intersecting motion envelopes. For example, $G1$ and $G3$ interact, but E and $S2$ do not.

Step 2 of the program constructs the CSs of the pairs and partitions them into uniform motion regions. The engager E and gear $G1$ have a 2D CS (Figs. 3 and 4). In the 2D region r_0, the engager and gear $G1$ turn independently and the engager is in *neutral* or in *high*. In the six 1D regions r_1–r_6 (one for each of the six lateral teeth in $G1$), the engager is in *low* and meshes with $G1$. Region transitions occur when the engager shifts

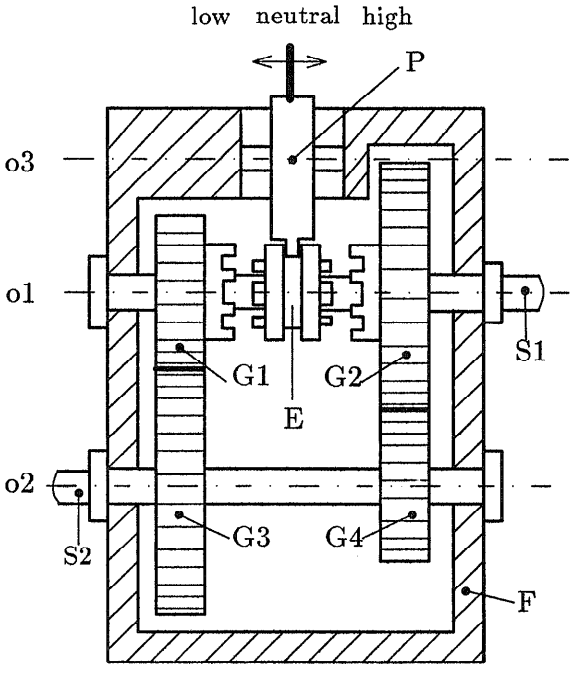

Figure 2: A side view of a transmission.

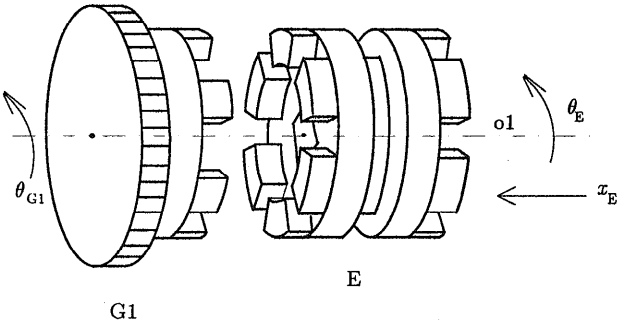

Figure 3: The engager E and gear $G1$ pair

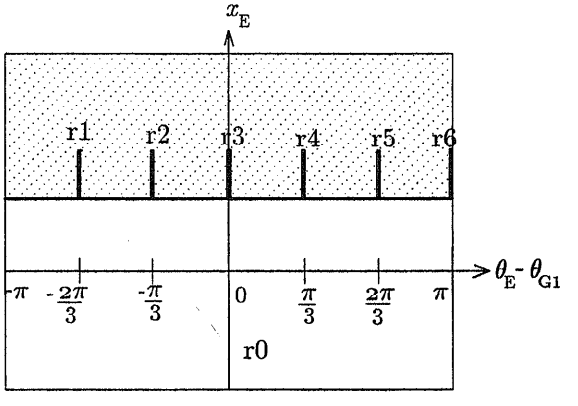

Figure 4: The CS of E and $G1$. The angle between E and $G1$ is $\theta_E - \theta_{G1}$. Shading indicates part overlap.

between *neutral* and *low*. The engager and $G2$ have a dual CS in which $-x_E$ replaces x_E. The other pairwise CSs are 1D. The $G1/G3$ and $G2/G4$ CSs each consists of a single region, a line with negative slope, since the pairs mesh. The engager/gearshift CS reduces to a point, since the parts translate in unison. The other CSs describe the interactions between the frame and the moving parts and between the shafts and the parts mounted on them. Each CS consists of one region.

Step 3 of the program composes the pairwise CSs into the mechanism CS, which describes the mechanism kinematics. The program determines that engager E moves freely in *neutral*, engages $G1$ in *low*, engages $G2$ in *high*, and never engages $G1$ and $G2$ at once. It computes the transmission ratio between $S1$ and $S2$ for the three gearshift settings. It describes the behavior of the transmission with the region diagram shown in Fig. 5. The lefthand, middle, and righthand regions represent the operating modes *low*, *neutral*, and *high*. The six low_i and six $high_i$ regions represent the six different angular offsets in which $G1$ and $G2$ can mesh with the engager. The regions specify the motion axes and motion types of the parts and the algebraic relations among the part coordinates (Fig. 6).

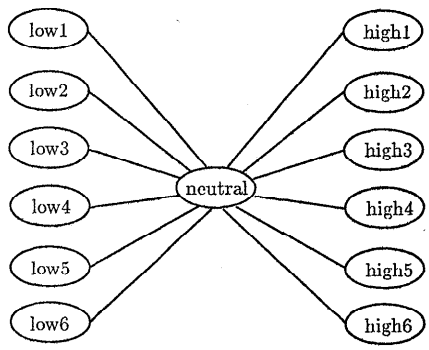

Figure 5: Region diagram of the transmission.

Motion types	Motion relations
$translation(P, o3, x_P)$	$1 \leq x_P \leq 2$
$translation(E, o1, x_E)$	$x_E = x_P$
$rotation(E, o1, \theta_E)$	
$rotation(S1, o1, \theta_{S1})$	$\theta_E = \theta_{S1}$
$rotation(G1, o1, \theta_{G1})$	$\theta_{G1} = -\theta_{G3}/2$
$rotation(G2, o1, \theta_{G2})$	$\theta_{G2} = -2\theta_{G4}$
$rotation(S2, o2, \theta_{S2})$	
$rotation(G3, o2, \theta_{G3})$	$\theta_{S2} = \theta_{G3}$
$rotation(G4, o2, \theta_{G4})$	$\theta_{S2} = \theta_{G4}$

Figure 6: Descriptor of the *neutral* region.

Incremental CS construction

We now describe the algorithm for composing the subassembly region diagrams into a mechanism region diagram. A mechanism configuration is realizable if no two parts of any subassembly overlap, that is if every subassembly configuration is realizable. Hence, the mechanism CS equals the intersection of the subassembly CSs. We obtain a partition of the mechanism CS into uniform motion regions by intersecting all combinations of subassembly uniform motion regions, called *component sets*, and discarding the empty intersections. We guarantee that each component set yields at most one region by splitting the subassembly motion regions into convex regions. Two regions are adjacent if every pair of corresponding components is identical or adjacent in its subassembly CS.

Enumerating and intersecting all the component sets is impractical for most mechanisms. The computation time equals the number of component sets times the intersection time, both of which are exponential in the number of parts. For example, a mechanism with ten parts has 45 kinematic pairs, which yield $2^{45} = 3.5 \times 10^{13}$ component sets when each pairwise CS has two regions. We develop an incremental algorithm that examines only the component sets reachable from the initial mechanism configuration. The algorithm performs well because of the design of mechanisms, although it cannot avoid the exponential worst-case time complexity of CS construction [Canny, 1988]. The tight coupling among parts makes most component sets unreachable from the initial configuration. The restrictions on the shapes and motions of parts simplify component set intersection.

Component set enumeration

We implement component set enumeration for general mechanisms. The program initializes a search queue with the component set that contains the initial mechanism configuration. The components are the regions of the subassembly CSs that contain the initial configuration. At each step, the program removes and intersects the first component set in the queue. If the intersection is nonempty, it records the new region, enumerates the reachable component sets, and adds the new ones to the queue. The reachable component sets are the adjacent component sets whose members are connected to the current region. Two component sets are adjacent if every pair of corresponding components is identical or adjacent. Two sets are connected if one contains a closure point of the other. For example, $(0, 1)$ connects to $[1, 2)$ but not to $(1, 2)$. The program can ignore unreachable component sets because the mechanism cannot enter the corresponding regions.

We illustrate the program on a 3-puzzle consisting of a fixed frame f containing square tiles a_1, a_2, and a_3 (Fig. 7). Tile a_i translates in the xy plane with coordinates (x_i, y_i) relative to a reference point at its bottom left corner. (We assume for simplicity that the tiles cannot rotate). The initial placements of a_1, a_2, and a_3 are $(0, 1)$, $(1, 1)$, and $(0, 0)$. Fig. 8 shows the

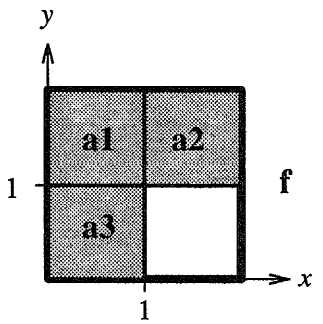

Figure 7: The 3-puzzle.

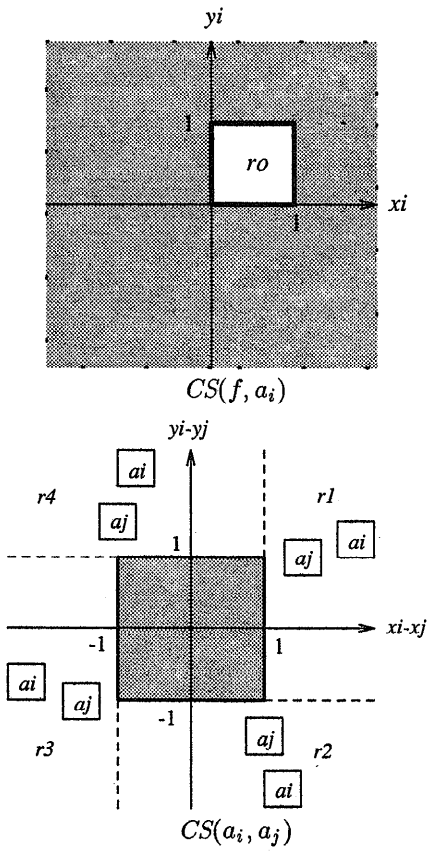

Figure 8: Pairwise CSs. Dashed lines delimit uniform motion regions, labeled with typical tile configurations. Shading indicates part overlap.

CS pair	region	constraints	
$CS(f, a_1)$	r_0	$0 \le x_1 \le 1$	$0 \le y_1 \le 1$
$CS(f, a_2)$	r_0	$0 \le x_2 \le 1$	$0 \le y_2 \le 1$
$CS(f, a_3)$	r_0	$0 \le x_3 \le 1$	$0 \le y_3 \le 1$
$CS(a_1, a_2)$	r_3	$x_1 - x_2 \le -1$	$y_1 - y_2 < 1$
$CS(a_1, a_3)$	r_4	$x_3 - x_1 \le -1$	$y_3 - y_1 < 1$
$CS(a_2, a_3)$	r_1	$x_2 - x_3 \ge 1$	$y_2 - y_3 > -1$

Table 1: Components of the initial puzzle position.

pairwise CSs describing the interactions between the frame and a tile and between two tiles. The first CS shows that the tiles stay inside the frame. The second CS shows that each tile in a pair can move around the other. (We explain the reduction to the relative 2D coordinates $(x_i - x_j, y_i - y_j)$ in the next section.)

The program finds the initial component set (Table 1). It intersects the components $(r_0, r_0, r_0, r_3, r_4, r_1)$ and records the resulting region G_0, a 1D submanifold of the 6D CS, defined by the constraints:

$$x_1 = 0, y_1 = 1, x_2 = 1, 0 < y_2 \le 1, x_3 = 0, y_3 = 0. \quad (1)$$

The first three components (interactions of the tiles with the frame) have no neighbors. The fourth has neighbors r_2 and r_4, but r_2 is unreachable because f and a_3 prevent a_2 from being above a_1. The fifth has no reachable neighbors because f and a_2 block a_1. The sixth has neighbors r_2 and r_4, but r_2 is unreachable because f and a_1 prevent a_3 from being above a_2. All told, G_0 has three neighboring component sets: $(r_0, r_0, r_0, r_4, r_4, r_1)$, $(r_0, r_0, r_0, r_3, r_4, r_4)$, and $(r_0, r_0, r_0, r_4, r_4, r_4)$. The program places them on the queue and searches them in turn. The first two yield the neighbors of G_0 and the third is empty. The full region diagram appears in Fig. 9.

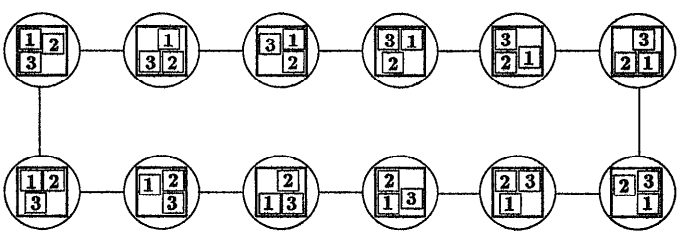

Figure 9: Region diagram of the 3-puzzle CS. Regions are represented by typical tile configurations. The top left region is G_0.

The 3-puzzle demonstrates the effectiveness of the reachability criterion. The puzzle has 64 configurations because each tile has four possible placements, but only 24 are reachable from the initial configuration. The 32 configurations where the tiles appear in counterclockwise order are not adjacent to any region. Another 8 configurations are not connected to any region because parts cannot leave the frame.

Component set intersection

We implement component set intersection for mechanisms in which the uniform motion regions of the subassemblies are polyhedral or are of dimension two or lower. A subassembly with a 2D CS normally has two degrees of freedom, but we can construct 2D CSs for some kinematic pairs with three or four degrees of freedom by switching to relative coordinates. If two interacting parts translate along parallel axes or rotate around the same axis, we can specify the relative

coordinate of one with respect to the other ($x_i - x_j$), instead of both coordinates (x_i and x_j), thus reducing the dimensionality by one. This reduction occurs in the engager/$G1$ CS (Fig. 4) and in the tile/tile CS of the 3-puzzle (Fig. 8).

The program first tests whether the component set defines a region, that is whether the the algebraic inequalities that define the components have a common solution. The worst-case time complexity of the test is exponential in the number of variables, making it impractical for most mechanisms [Canny, 1988]. Instead, the program approximates the nonlinear inequalities with a larger set of linear inequalities, which it tests in expected polynomial time with the BOUNDER inequality prover [Sacks, 1987].

The program approximates every nonlinear component boundary from within and from without with piecewise linear segments (Fig. 10). It picks segments that preserve the topology of the local CS and the monotonicity of the curves and that differ from the original by a small tolerance [Joskowicz, 1989b]. The inner approximation specifies a subset of the true component and the outer approximation specifies a superset. If the intersection of the inner approximations is nonempty or the intersection of the outer approximations is empty, so is the region. Otherwise, the test is ambiguous. The program uses the outer result, which never misses regions, but can introduce spurious regions. The spurious regions correspond to part interactions beneath the granularity of the approximations. The program could resolve them by shrinking the granularity. Instead, we choose a reasonable tolerance based on standard machining assumptions.

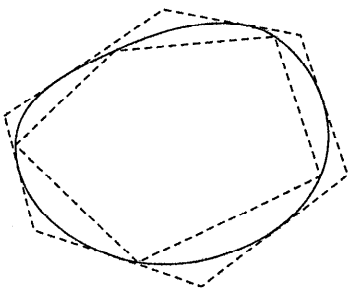

Figure 10: Inner and outer CS boundary approximations of a 2D region.

After intersecting a component set, the program determines the true degrees of freedom of the parts in the intersection. Global interactions often rule out potential degrees of freedom by blocking axes of motion. In the 3-puzzle configuration of Fig. 7, each tile has two potential degrees of freedom, translation along both axes, but tile a_1 has zero true degrees of freedom and the other tiles have one apiece, as shown in Equation (1). The program bounds the coordinates of the parts with BOUNDER. It eliminates the degrees of freedom for which the lower and upper bounds coincide. For example, it eliminates translation along the x axis for a_1 because x_1 equals 0. (Finding blocked degrees of freedom corresponds to finding implicit equalities in a set of linear inequalities [Huynh *et al.*, 1990].) The program uses the inner approximation for determining blocked degrees of freedom, so it ignores small motions caused by imperfectly fitting parts.

The algebraic relations that define a region succinctly characterize the precise kinematics of the mechanism. For fixed-axes mechanisms, the program also annotates the region with symbolic motion predicates that describe the kinematics qualitatively. Each CS coordinate is associated with its part, motion axis, and motion type: *fixed, translation,* or *rotation.* Table 2 contains the description of the initial 3-puzzle region.

Motion types	Motion relations
$\mathit{fixed}(a1, x, x_1)$	$x_1 = 0$
$\mathit{fixed}(a1, y, y_1)$	$y_1 = 1$
$\mathit{fixed}(a2, x, x_2)$	$x_2 = 1$
$\mathit{translation}(a2, y, y_2)$	$0 < y_2 \leq 1$
$\mathit{fixed}(a3, x, x_3)$	$x_3 = 0$
$\mathit{fixed}(a3, y, y_3)$	$y_3 = 0$

Table 2: Descriptor of the initial 3-puzzle region.

The transmission revisited

The program constructs a region diagram with 13 regions for the transmission (Fig. 5). It constructs the exact CS regions because all the constraints are linear. Gearshift P is in *neutral* and $S1$ and $S2$ turn independently in the 3D neutral region, which contains the initial mechanism position (Fig. 6). In the six 2D regions low_1–low_6, the gearshift is in *low* and $S1$ drives $S2$ via $G1$ and $G3$ with six different angular offsets. In the six 2D regions $high_1$–$high_6$, the gearshift is in *high* and $S1$ drives $S2$ via $G2$ and $G4$. The program intersects 13 component sets out of 49 because the others are unreachable. Without the filtering, the program would have to intersect prohibitively many component sets. The gear pairs $G1/G3$ and $G2/G4$ each can engage at m distinct angular offsets, where m is the number of meshing teeth per gear. This yields $49m^2$ component sets, or 19600 component sets for $m = 20$.

Conclusions

We have tested the incremental CS construction program on a dozen examples, including the 3-puzzle, a tilted 6-puzzle, a door lock, and the transmission. It produces an exact CS or a good approximation for every example. Running times range from two minutes for the 3-puzzle to five minutes for the transmission.

In other work [Joskowicz and Sacks, 1990], we assess the coverage of the program by surveying 2500 mechanisms from a mechanical engineering encyclopedia

[Artobolevsky, 1979]. The mechanisms fall into four categories: linkages (35%), fixed-axes (22%), fixed-axes coupled by linkages (9%), and complex mechanisms (34%). Standard linkage packages [Haugh, 1984, Kramer, 1990] handle linkages directly. The program covers the remaining mechanisms whose subassemblies have polyhedral or 2D CSs: most fixed-axes mechanisms, most mechanisms composed of fixed-axes coupled by linkages (Fig. 11), and some mechanisms with complex subassemblies, such as planetary gears and helical cams. All told, the program covers about 2/3 of the mechanisms. This estimate assumes that piecewise linearization yields qualitatively correct CSs. The only potential exceptions are mechanisms that rely on precise nonlinear relations among parts, such as mechanisms with logarithmic cam slots and straight-line linkage couplers.

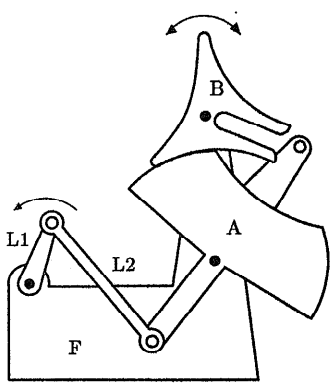

Figure 11: A dwell mechanism with fixed-axes pair A, B coupled to linkage A, L_1, L_2, F. The continuous rotation of crank L_1 causes A to rotate clockwise, rest, and rotate counterclockwise.

We can automate the complete analysis (Fig. 1) of general mechanisms by interfacing the composition program with subassembly analysis modules. In the longer paper, we implement the module and the interface for kinematic pairs. The user must provide the CSs of other subassemblies. We plan to interface the program with a linkage package and to develop a CS library for common complex subassemblies.

We obtain a composition program for general mechanisms by replacing the component set intersection module with Canny's decision procedure for polynomial inequalities, but at an exponential-time cost. We can also extend piecewise linearization to higher dimensions, but at an unclear computational cost. A better approach is for the current module to identify the hard cases (where the inner and outer approximations disagree) and pass them to a nonlinear inequality reasoner, such as BOUNDER, Kramer's program, or Canny's algorithm. Piecewise linearization makes the extra computational cost for the hard cases affordable by handling most cases quickly.

References

Artobolevsky, I. 1979. *Mechanisms in Modern Engineering Design*, volume 1–4. MIR Publishers, Moscow. English translation.

Brost, R. C. 1989. Computing metric and topological properties of configuration-space obstacles. In *Proceedings IEEE Conference on Robotics and Automation.*

Canny, J. 1988. *The Complexity of Robot Motion Planning.* MIT Press, Cambridge, MA.

Faltings, B. 1990. Qualitative kinematics in mechanisms. *Artificial Intelligence* 44(1–2)

Gelsey, A. 1989. Automated physical modeling. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence.*

Haugh, E.J., editor 1984. *Computer Aided Analysis and Optimization of Mechanical System Dynamics.* Springer-Verlag.

Huynh, T.; Joskowicz, L.; Lassez, C.; and Lassez, J. L. 1990. Reasoning about linear constraints using parametric queries. In *Proceedings 10th International Conference on Foundations of Software Technologies and Theoretical Computer Science*, Bangalore, India. Springer-Verlag.

Joskowicz, L. and Sacks, E. P. 1990. Computational kinematics. Technical Report CS-TR-300-90, Princeton University.

Joskowicz, L. 1989a. Reasoning about the kinematics of mechanical devices. *International Journal of Artificial Intelligence in Engineering* 4(1).

Joskowicz, L. 1989b. Simplification and abstraction of kinematic behaviors. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence.*

Kramer, G. A. 1990. Solving geometric constraint systems. In *Proceedings of the Tenth National Conference on Artificial Intelligence.* American Association for Artificial Intelligence.

Lozano-Pérez, T. 1983. Spatial planning: A configuration space approach. In *IEEE Transactions on Computers*, volume C-32. IEEE Press.

Nielsen, P. E. 1988. *A Qualitative Approach to Rigid Body Mechanics.* Ph.D. Dissertation, University of Illinois at Urbana-Champaign.

Reuleaux, F. 1963. *The Kinematics of Machinery: Outline of a Theory of Machines.* Dover Publications.

Sacks, E. P. 1987. Hierarchical reasoning about inequalities. In *Proceedings of the 7th National Conference on Artificial Intelligence.*