Computer-aided Creative Mechanism Design

Boi Faltings and Kun Sun
Artificial Intelligence Laboratory (LIA)
Swiss Federal Institute of Technology (EPFL)
IN - Ecublens, 1015 Lausanne, Switzerland
e-mail: faltings@lia.di.epfl.ch, FAX: +41-21-693-5225

Abstract

A popular saying claims that "innovation is 1% inspiration and 99% perspiration." In this paper, we present techniques for automating most of the perspiration involved in creative design. We assume that creative design consists of three steps: *discovery* of a new technique, *understanding* it, and *generalizing* it to useful applications. Our program uses first principles to automate the understanding and generalization phases which involve most of the perspiration, and extends its knowledge accordingly so that it can construct practical designs based on the new technique. The technique could be used to create new devices automatically, but in practice user interaction is necessary to control the search.

We present a system which implements the method in the domain of elementary mechanisms, also called *kinematic pairs*. Its main novelties are a formalism for modeling qualitative mechanical function, and a technique similar to explanation-based learning which generalizes a qualitative analysis of a novel device to extend domain knowledge. The results are generalizable to other domains with similar characteristics.

1 Introduction

A clock escapement (Figure 1) is the central mechanism of a mechanical clock. It is used to regulate the motion of a *scape wheel* to advance one tooth per oscillation of a pendulum. Since the pendulum has a fixed period of oscillation, this means the wheel moves at constant speed and can drive the hands of a clock. The regulating function of the escapement requires *blocking* the rotation of the scape wheel in certain positions of the pendulum, while letting it advance in others. In the device of Figure 1, this is accomplished by the shapes of the two sides of a piece attached to the pendulum, called the *pallet*.

A ratchet (Figure 2 a) is a device intended to block rotation of a wheel in one direction, but not in the other. Another property of the device in Figure 2), rather undesirable for the function of the ratchet, is that this blocking only occurs when the lever is in a certain range of positions, but not in others (Figure 2 b). This property can be used to create an escapement device such as the one in Figure 1.

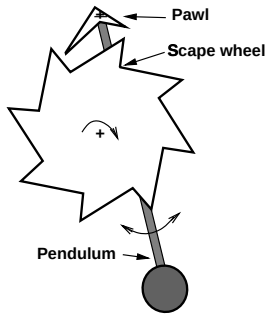


Figure 1: *An escapement design produced using our system. The wheel is driven clockwise, and the pawl is attached to a pendulum which creates an oscillation with constant period.*

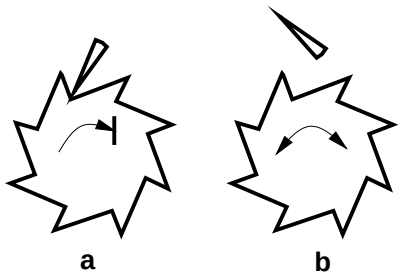


Figure 2: *A ratchet in a blocking (a) and non-blocking (b) state.*

Noticing this "spurious" property of a ratchet to create an escapement is an example of a creative design. It involves three processes:

- *discovery* of the fact that the ratchet device has a function which is required in an escapement, and
- *analysis* to define a shape feature which is responsible for the desired function, thus creating new design knowledge, and
- *adaptation* of the shape feature in a new design con-

text.

Discovery of interesting behaviors seems to require human intuition which is little understood and probably too complex to automate. The more tedious activity of analyzing a behavior and adaptating it into new design contexts, however, *can* be supported by knowledge-based systems, as we will show in this paper.

Creative design defines new functions, devices and mappings between them. A computer tool which supports it must provide:

- a language for expressing the functions of a device.
- a language for defining the new design features essential to the creative idea and their mapping to functions.

Current knowledge-based design systems are based on fixed vocabularies of functional and design features. For example, *prototypes* ([Gero, 1990]) or *intelligent objects* functions, and use these to define a library of standard solutions which the designer can use to compose designs automatically. Other researchers have attempted to generate designs from symbolic domain knowledge using model-based abductive reasoning (for example [Williams, 1990; Williams, 1990]). They allow function to be specified by a generative language, but generate *specifications* for artifacts without consideration of how they could be physically realized.

The essence of creative design is to create new features and adapt them to new problems. Knowledge-based design systems rely on primitive features to provide support for *generating* designs. However, *analysis* does not need to be based on a fixed feature system: for example, finite element analysis can be applied to any representable structure. In this paper, we introduce a technique similar to explanation-based learning where an analysis of a new device is generalized to make it applicable to other contexts. This makes it possible to support creative design by extending domain knowledge as required.

An ideal domain for exploring this idea is shape design in mechanisms: designers routinely create shapes which involve new design features. At the same time, the kinematic behavior of any new device can be determined and explained using first principles. More specifically, we have developed our technique for the design of *kinematic pairs*, also called *elementary mechanisms*. A kinematic pair consists of two parts with one degree of freedom each, and achieves its function through the kinematic interactions of the parts. An example of a kinematic pair is a clock escapement, as shown in Figure 1. The qualitative *behavior* of a kinematic pair can be computed and represented using the theory of qualitative kinematics ([Faltings, 1990; Forbus *et al.*, 1991]). Furthermore, it is possible to *invert* the computation to determine the limits up to which it is valid ([Faltings, 1992]) and thus define the new design knowledge required to represent the new idea. Our techniques concentrate on the *conceptual* design stage where most creativity takes place. Subsequent *detail design* can further optimize the dimensions to accomodate non-qualitative specifications.

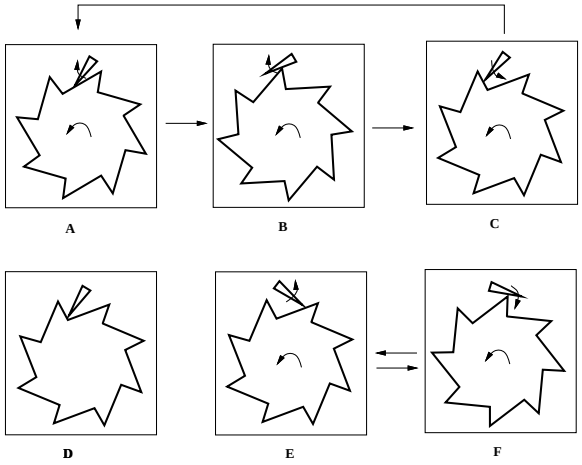


Figure 3: *Examples of kinematic states and transitions in a ratchet.*

Synthesis of kinematic pairs has been studied by Joskowicz and Addanki [Joskowicz and Addanki, 1988]. The techniques we present improve on their work in the following ways: (i) we use a realistic qualitative language for specifying function, and (ii) our techniques can deal with contact subsumptions and intermittent motion, required for devices of practical interest.

We first discuss the formal representation of kinematic behavior and show how languages for expressing qualitative *function* can be defined on its basis. We then show how the analysis of a particular device can be generalized to define the shape features which are responsible for the its function, and how the new features can be used to derive new designs. Finally, we give an example of a design in which a ratchet is transformed into an escapement.

2 A language for qualitative function

Function is a mapping from external influences to the behavior they cause. We define a set of *behavior predicates* and a formalism for expressing external influences on mechanisms. Functions are then defined by logical expressions connecting external influences and behavior predicates.

Representing qualitative kinematic behavior Textbooks on the subject explain kinematic behavior qualitatively by sequences of *kinematic states*. Examples of kinematic states of a ratchet device are shown in Figure 3. In qualitative physics terminology, a graph of kinematic states and transitions is called an *envisionment*. It can be computed based on a *place vocabulary* [Faltings, 1990], a graph where each node represents a different combination of contact relationships, and each arc represents a potential transition between them. The envisionment is obtained by combining each node of the place vocabulary with assumed motions and keeping only the states and transitions consistent with external forces and motions. We have developed and implemented complete algorithms to compute place vocabularies for arbitrary two-dimensional higher kinematic pairs in fixed-

axis mechanisms [Faltings, 1990]. These have been used to compute envisionments for a number of mechanisms, such as a mechanical clock [Forbus *et al.*, 1991].

We represent place vocabularies using a set of *behavior predicates* which characterize places, their features and their connectivity. For a kinematic pair, the place vocabulary defines a graph containing three types of kinematic states, corresponding to two, one and no contacts, and identified by the following behavior predicates:

- **point-place**(x): the contacts in x hold only in a single configuration.
- **edge-place**(x): the contacts in x hold in a one-dimensional set of configurations.
- **face-place**(x): x is a place without any contacts and two-degrees of freedom.

For each place, the place vocabulary defines the allowed qualitative directions of motion:

- **qualitative-motion**(d): d is a qualitative vector (d_0, d_1) whose components indicate the direction of motion of each object: $d_i \in \{-, 0, +\}$.
- **allowed-motion**(x,d): motion d is possible everywhere in place x.

For each link between states, the place vocabulary defines the directions which can cause a transition:

- **transition**(x,y,d): motion d can cause a transition from place x to y.

The kinematic states of a mechanism are obtained by combining each place x with its maximal set of possible qualitative motions:

$$\mathcal{M}_{all}(x) = \{m \mid \textbf{allowed-motion}(x, m)\}.$$

In an actual behavior, only those motions $\mathcal{M}(x)$ which in fact *caused* by an external influence actually occur. The set of transitions between states is the set:
$\mathcal{T}_m = \{\ (x, y) \mid (\exists d \in \mathcal{M}(x))\ \textbf{transition}(x, y, d)\ \}$.
More details on envisioning mechanisms using place vocabularies can be found in [Nielsen, 1988].

Representing external influence In kinematic pairs, external influences can be either *forces*, represented by a set $\mathcal{F}_{ass}$ of qualitative vectors ([Faltings, 1990]), or *motions*, represented by a set $\mathcal{M}_{ass}$ also consisting of qualitative vectors. Since a qualitative force vector causes a qualitative motion in the direction of the same vector, the set of possible motions $\mathcal{M}_f$ caused by external forces is then given as:

$$\mathcal{M}_f(x) = \{v \mid v \in \mathcal{F}_{ass}\ \}$$

The actual set of motions $\mathcal{M}(x)$ to be considered in state x is then:

$$\mathcal{M}(x) = \mathcal{M}_{ass}(x) \cap \mathcal{M}_f \cap \mathcal{M}_{all}(x)$$

Formulating functions Functions are properties of behavior under certain assumed forces and motions. Therefore, qualitative functions can be defined as *conditions* on place vocabularies without first constructing the qualitative behavior. Using the behavior predicates which represent the place vocabulary, a set of functions can be defined as required for the application, and extended whenever required to express a new specification. For example, some functions our current prototype system uses are:

- **transmitting-place**(x, dir_1, dir_2):
  $(\forall d = (d_1, d_2))\{d_1 = dir_1 \Rightarrow d_2 = dir_2\} \wedge$
  $\{d_2 = -dir_2 \Rightarrow d_1 = -dir_1\}$
- **blocking-place**(x):
  $\neg(\exists d \in \mathcal{M}(x))$**allowed-motion**$(x, d)$
  (a place blocks motions if it does not allow any of the assumed motions).
- **partial-blocking-place**$(x, dirs)$:
  $\neg(\exists d \in dirs)$**allowed-motion**(x, d)
  (a partial blocking place blocks the specified motions)
- **possible-path**(x_0, x_n):
  $(x_0 = x_n) \vee \exists\ \mathcal{S} = (x_0, x_1, x_2, \ldots, x_n)$
  $(\forall i < n)(\exists d \in \mathcal{M}(x_i))$**transition**$(x_i, x_{i+1}, d)$
  (There is a path from place x_0 to place x_n whenever there is a sequence of places with transitions between them under at least one assumed motion)
- **cycle**$(x_0, \mathcal{C})$:
  $\mathcal{C} = (x_0, x_1, x_2, \ldots, x_n, x_0)\ (\forall x_i \in \mathcal{C})(\exists d \in \mathcal{M}(x_i))$
  transition$(x_i, x_{mod(i+1,n+1)}, d)$
  (there is a cycle of states $\mathcal{C}$ such that transitions between subsequent states are consistent with assumed and allowed directions of motion.)

A place vocabulary can only fulfill the required functions if the number of states and their connectedness is sufficient. Reasoning about such *topological* features is difficult in the place vocabulary itself, since it is based only on individual *boundaries* of shapes which cannot be modified individually. We use an explicit representation of the *kinematic topology* ([Faltings *et al.*, 1989]) of the mechanism to detect cases where the topology of particular object shapes would not permit the specified function. An example of a function defined on the basis of kinematic topology is:

- **cycle-topology**(c, d_1, d_2): if the first or second object have rotational freedom, the cycle involves d_1 rotations of the first or d_2 rotations of the second object. This predicate is defined directly on the kinematic topology of the mechanism.

which can be defined formally using similar behavior predicates as those which define place vocabularies.

The function of a ratchet can now be defined qualitatively as follows:
For all starting states a in which the orientation of the lever is in the interval $[0..\pi]$ (pointing to the left such that the moment gravity exerts on it is positive):

- for $\mathcal{M}_{ass}$=$\{(+, *)\} \wedge \mathcal{F}_{ass}$= $\{(*, +)\}$ (the '*' stands for either +, 0 or -):
  - **cycle**$(a, \mathcal{C}) \wedge$ **cycle-topology**$(c, 1, 0)$
  - $\neg(\exists x)$**blocking-place**$(x)\ \wedge$ **possible-path**(a, x)
  (assuming that the wheel turns counterclockwise and the lever is forced onto it, there is a cycle of states where the wheel can rotate, and no reachable blocking state from any starting state a.)
- for $\mathcal{M}(x)$=$\{(-, *)\} \wedge \mathcal{F}_{ass}$= $\{(*, +)\}$:
  $(\forall y)$**possible-path**$(a, y) \Rightarrow \{\neg$**cycle**$(y) \wedge$

$(\exists z)(\textbf{blocking-state}(z) \wedge \textbf{possible-path}(y, z))\}$
(assuming that the wheel turns clockwise, no reachable state leads to a cycle and all states can eventually lead to a blocking state).

Note that due to the ambiguities inherent in qualitative envisionments, the formalism always overgenerates behaviors. It is therefore only possible to define *necessary*, but never *sufficient* specifications of behavior and, consequently, function. For example, we can express the specification that clockwise motion leads to a blocking state only in an indirect manner: if there is no possibility to cycle, and there is at least one reachable blocking state, the device must eventually reach this state.

3 Extending the feature vocabulary and domain knowledge

Shapes are represented using a *metric diagram* ([Faltings, 1991; Faltings, 1992]). The metric diagram consists of a symbolic structure which defines vertices, edges and *metric parameters* for the positions of the vertices. In our current implementation, we restrict ourselves to polygons, but see [Faltings, 1990] for ways to extend it to include circular arcs.

While qualitative functions can be formulated in a symbolic language, nobody has so far found a similar language for defining features of *individual* shapes which is descriptive enough to infer kinematic behavior. Because of the strong dependence of shape features on the counterparts they interact with, it seems unlikely that such a language could be defined at all ([Faltings, 1991]). We now show how it is possible to define a language for features of *pairs* of shapes which predicts kinematic features of their interaction.

Using the metric diagram, a shape feature (which may involve several objects) is defined by:

- a set of vertices and edges,
- the metric parameters associated with them,
- a set of constraints which must hold simultaneously for the shape feature to be present.

For every behavior predicate of the place vocabulary for a novel device, our program constructs an *elementary shape feature* which implies the presence of that feature in the absence of subsumptions (see below). The computation of this definition is illustrated in Figure 4. For each behavior predicate in the place vocabulary, the trace of the computation shows a conjunction of tests on metric parameters which have been satisfied and led to asserting the predicate. Each test queries the sign of an algebraic expression in a set of metric parameters, and can be understood as a *constraint* on those parameters. These constraints, and the parameters they involve, define the new shape feature.

Subsumptions are interactions between unrelated parts of the shapes which make certain contacts physically impossible. In principle, every behavior predicate is valid only under the assumption that no subsumptions inhibit it. For efficiency reasons, our system generates tests for subsumptions only when they have been discovered to be important by leading to unexpected results.

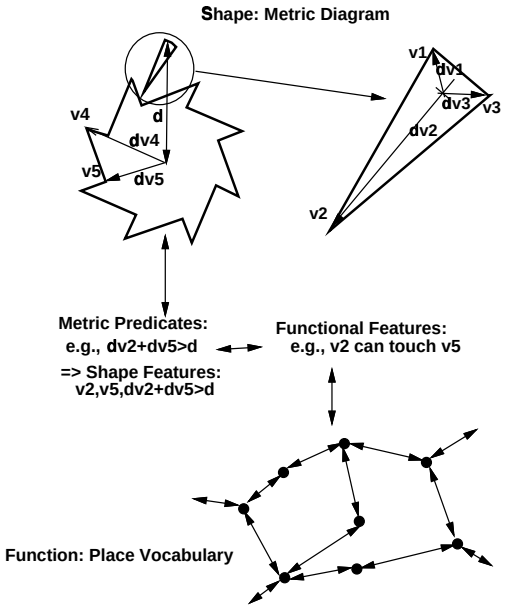


Figure 4: *Analysis of a device defines a place vocabulary as a set of features, for example the possibility of touch between the tip of the lever and the bottom of the wheel's teeth. Such features define* metric predicates *such as the one shown.*

More details on the mapping between functions and metric dimensions can be found in [Faltings, 1992].

A creative idea defines a new combination of shape features which satisfies a function. The combination is defined in the example which implements the idea and can be formulated as a rule:

$$\text{feature}_1 \wedge \text{feature}_2 \wedge ... \Rightarrow \text{function}$$

Note that it would in principle be possible to generate designs from first principles by using the translation from elementary functional predicates to elementary shape features to transform functional specifications into shape specifications. However, in pratice this is a bad idea, since the resulting dynamic constraint network is only a specification for a solution and there are no known general methods for solving such specifications.

Designing with new knowledge The formulation of metric predicates turns shape design into a constraint satisfaction problem: find a structure and associated dimensions which satisfy the constraints for the functions required by the specification. The constraint network is *dynamic*, since modifications of the shapes can change the constraint set, and it involves unbounded value sets. Another difficulty is that many constraints depend on a large number of variables, so that representations by binary constraint networks are not appropriate.

As a tractable, but incomplete, way of solving such constraint systems, we use incremental refinement of an initial solution using means-ends analysis. Each refinement is carried out by identifying a discrepancy between specifications and current function and applying a *modification operator* which is likely to eliminate this discrepancy. More efficient strategies are possible by analyzing

the dependencies between functions, similar to the techniques investigated by Williams [Williams, 1990]. Our system generates two types of modification operators:

- *dimensional* modifications, where the dimensions of parts are adjusted to fit the functional requirements, and
- *topological* modifications, where vertices are added to part shapes. A topological modification is always coupled with a dimensional modification to fix the dimensions of the new features.

Dimensional modification operators are indexed by their effect on the place vocabulary: *changing* the appearance of a state in the place vocabulary, or making a state *appear* or *disappear*. Based on matches between possibilities and active goals, the system computes a finite set of potentially applicable modification operators for proceeding with the design.

All modification operators are subject to the additional constraint that a polygon shape may not overlap itself, a configuration which would be physically impossible to achieve. We have developed an algorithm for incrementally generating and maintaining the constraints which express this condition.

Topological modifications are proposed when the constraints formulated by metric predicates are contradictory: adding an additional vertex gives two additional degrees of freedom to resolve the contradiction. More precisely, a contradiction occurs when there is no feasible region to place the vertex whose modification the system is considering[1]. The system then proposes two modification operators which add vertices in the centers of the two edges which meet at this vertex. Removal of vertices to simplify shapes is considered only when the final result is reached. A vertex can be removed if placing it on the straight line between its two neighbours is consistent with all metric constraints.

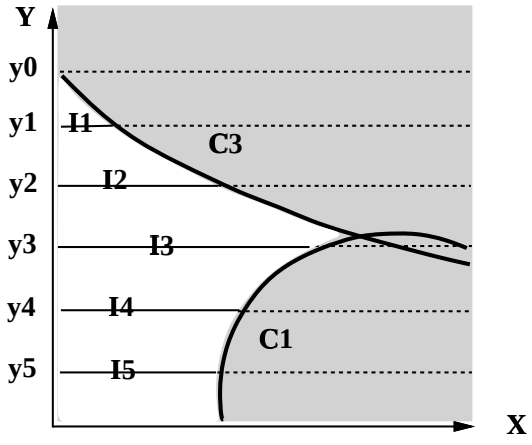
Dimensional modifications are based on the constraints defined by metric predicates. These divide up the space of shape parameters into regions in which the qualitative behavior does not change. Since any choice of value is equivalent, new dimensions are obtained by picking an arbitrary value combination within a region that exhibits the desired behavior [Faltings, 1992].

In practice, the computational complexity makes explicit computation of the region intractable or even incomputable, since most of the constraints are nonlinear and some are not even algebraic. A tractable, but necessarily incomplete solution is to select only a single *landmark parameter* for modification, and *project* the constraints onto *landmark values* of this parameter. The landmark values can be computed by numerical approximation, thus avoiding expensive computation with nonlinear curves.

As an example, consider the possibility of touch between $v3$ and $v4$ (Figure 4 as a function. The constraints required for proving the existence of the feature is:

$$d_{v3} + d_{v4} > d \text{ and } |d_{v3} - d_{v4}| < d$$

[1]Note that this does not imply that there is no solution at all, only that it cannot be found by changing a single vertex.



C1: $\sqrt{x_3^2 + y_3^2} - 4 > 0$

C2: $30 - |\sqrt{x_3^2 + y_3^2} - 26| > 0$ (satisfied everywhere)

C3: $4 - \frac{x_3 \times y_3 + y_3 \times (12 - x_3)}{y_3^2 + (x_3 - 12)} > 0$

Figure 5: *Metric predicates define the regions in the space of metric parameters where certain functions are satisfied.*

Assuming that d_{v3} has been chosen for modification, projection (using $d_{v4} = 26$ and $d = 30$) gives landmark values of $d_{v3} > 4$ $d_{v3} > -4$ and $d_{v3} < 56$. A modification operator which makes the touch *appear* would set d_{v3} to a value within the interval, for example at $d_{v3} = 6$. An operator which would make a touch *disappear* would set it to $d_{v3} = 2$, for example. This solution is incomplete, as it can only extend $v3$ straight outwards from the center of rotation, but not change its angle seen from that point.

For designing two-dimensional shapes, the most serious incompleteness problems can be avoided by considering the *pairs* of parameters which define the positions of individual shape features. Our system uses an approximation method to generate solutions for pairs of parameters x and y, illustrated in Figure 5. First, we arbitrarily pick one of the two parameters, say y, and define a number of equally spaced sample points (our implementation uses 40), shown as y_0 through y_5 in Figure 5. For each value y_i, we project the constraints to find the interval of x within the legal region: only intervals I_1 through I_3 (for y_1, y_2 and y_3) exist in the example. Finally, a new value combination is picked as the center of one of the intervals for x and the corresponding y_i for y.

Designing with modification operators The generation and application of modification operators must be controlled using domain knowledge to avoid excessive search. Since it is very difficult to formulate such domain knowledge, our current system asks the user to choose

- the discrepancy to modify based on suggestions presented by the system,

- the modification operator to apply from a list of possibilities (this implies selction of the variable to be changed),
- topological changes to create additional degrees of freedom, when required.

Automating the choice using explicit control knowledge is beyond the focus of this project, but we are investigating the use of case-based reasoning for this purpose.

4 Turning a ratchet into an escapement

As an example of using modification operators for design, we show how the escapement shown earlier in Figure 1 can be obtained from the ratchet of Figure 2.

Using the functions defined earlier, an escapement can be specified in the following way:

1. ($\exists$cycle$\mathcal{X} = \{x_0, x_1, x_2, \ldots, x_{n-1}\}$) of states: $\forall x_i$, **partial-blocking-place**$(x_i, (+, *))$ and **cycle-topology**($\mathcal{X}$,1,0)
2. ($\exists$cycle$\mathcal{Y} = \{y_0, y_1, y_2, \ldots, y_{n-1}\}$) of states: $\forall y_i$, **partial-blocking-place**$(y_i, (+, *))$ and **cycle-topology**($\mathcal{Y}$,1,0)
3. for $\mathcal{M}(x)$=$\{(*, 0), (*, -)\} \wedge \mathcal{F}_{ass} = \{(+, *)\}$: $(\forall x_i \in \mathcal{X})$**possible-path**$(x_i, y_i), y_i \in \mathcal{Y}$ (when the pendulum swings from right to left or is stationary, there exists a path from place x_i to y_i)
4. for $\mathcal{M}(x)$=$\{(*, 0), (*, +)\} \wedge \mathcal{F}_{ass} = \{(+, *)\}$: $(\forall y_i \in \mathcal{Y})$**possible-path**$(y_i, x_{mod(i+1,n)}), x_i \in \mathcal{X}$ (when the pendulum swings from left to right or is stationary, there exists a path from place y_i to the place following x_i in the cycle $\mathcal{X}$.)
5. for $\mathcal{M}(x)$=$\{(+, 0)\}$: $(\forall x)\neg(\exists \mathcal{C})$**cycle**$(x, \mathcal{C})$ (the wheel is prevented from rotating a full cycle whenever the pendulum does not move)

One possible trace of an incremental modification which achieves these specifications is illustrated in Figure 6.

The ratchet (device A) already provides a cycle of blocking states which can be used to satisfy either the functional specification (1) or (2) of the escapement. However, it does not satisfy specification (5), and specifications (3) and (4) cannot be evaluated.

The system matches the cycle of blocking states of the ratchet to specification (1), and we assume that the user chooses to first satisfy specification (2). The first subgoal is then to create the cycle of places it requires, in a way that they satisfy the **partial-blocking-place** property. The user chooses to change the position of vertex $v3$ (see Figure 4) among the variables which the system determines suitable for modification operators. New contacts and thus new states will exist whenever vertex $v3$ is able to touch $v4$, which results in the condition $4 < d_{v3} < 42.43$ on the position of $v3$ for (potentially) removing discrepancy (2). Further constraints are added to ensure that the shape of the lever remains a legal polygon, and values are found by constructing the feasible regions in the space of x_{v3} and y_{v3}. Choosing $x_{v3} = -1.9, y_{v3} = 5.69$, the system simulates the new device and finds that specifications (3) and (4) are not satisfied: there is no transition between the blocking states

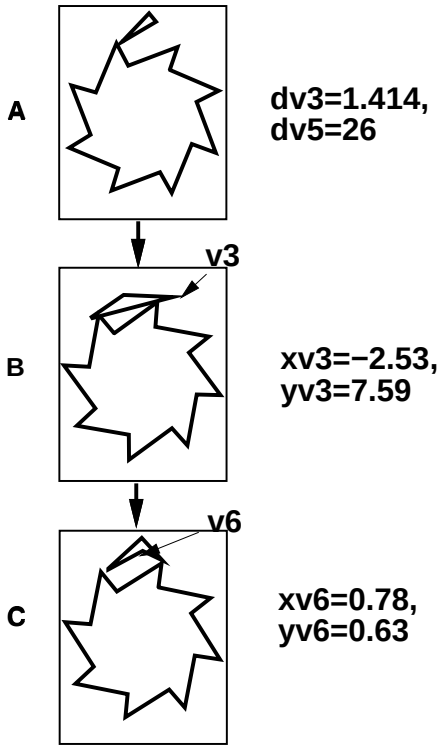


Figure 6: *Incremental design of an escapement using modification operators derived by causal inversion.*

where the pendulum does not move. The user chooses to correct this by modifying $v3$, thus creating another constraint and the new values $x_{v3} = -2.53, y_v3 = 7.59$, resulting in the device shown in Figure 6 B. The total number of possible modifications proposed by the system to address discrepancies existing in state A is 112, and computing and applying the modification operator took 1309 seconds on a Symbolics 3650 LISP machine. User interaction is therefore quite important to avoid excessive computation time.

Simulating version B shows that due to a subsumption, the required transitions between blocking states are still impossible. Again, the user chooses to satisfy this property by modifying $v3$, but this time the constraint for avoiding the subsumption is contradictory with the earlier ones and no modification can be found. The constraints on $v3$ are thus declared contradictory, and a new vertex $v6$ is introduced between $v2$ and $v3$ to create a new degree of freedom in version B. The user can now choose to modify $v6$ to solve the discrepancies; the system proposes $x_{v6} = 0.78$ and $y_{v6} = 0.63$, shown in Figure 6 C. Simulation shows that there are no subsumptions which produce unexpected behavior, and furthermore specification (5) also turns out to be satisfied. This time, there were 180 alternative ways of satisfying the specification, but computation of the actual modification took only 111 seconds. We note that user interaction is indispensible for controlling the modification process: had the system started by attempting to satisfy specification (5), for example, a long and not very fruitful search

would have resulted. The intuitions behind such choices appear very complex and we doubt that they could be formulated in a sufficiently concise way.

5 Conclusions and Future Work

We have shown how first-principles knowledge about kinematics can be used to support creative design of shapes in elementary mechanisms by providing:

- a language for formulating qualitative kinematic function, and
- a method for automatically extending the vocabulary of shape features and domain knowledge to accomodate creative design ideas.

An intelligent CAD system using the technique would allow the user to introduce creative ideas and have the system use them in automatically generated designs. By making the system search for novel ideas itself, one could also construct an autonomous creative system, but this would probably not be widely applicable due to the excessive computational complexity.

Our technique is similar to explanation-based learning. Its application to a continuous domain enables the explanation-based technique to actually acquire new knowledge, contradicting common belief that explanation-based techniques are not capable of extending the deductive closure of existing knowledge.

The system we have developed for adapting creative solutions with modification operators is computationally tractable and shows the feasibility of the concept. It incompleteness - solutions are only found by modifications of single vertices at a time - did not seem to be significant in practice. It is in fact surprisingly easy to create truly novel designs using the program. For example, starting from the observation that the pawl of an escapement can be used to drive the scape wheel, the program has invented the reversing mechanism shown in Figure 7, a much simpler and more reliable design than that found in a mechanism book which involves 4 parts and reliance on friction for function[2].

The incremental refinement method can only adapt a single solution, and requires guidance from the user to do so. However, by proposing itself the useful modifications, the system does automate the most difficult and tedious aspect of creative design, that of formulating the constraints and possible solutions. Our current research focusses on methods for *combining* two devices and their functions. This should make it possible, for example, to design an escapement by combining two ratchet devices. The combinatorial explosion of modification operators would then be significantly reduced and might make automatic adaptation tractable. In conjunction with a suitable mechanism library and indexing device such as [Williams, 1990], truly creative design systems might be possible.

[2]However, our program only takes kinematics into account. Constraints on materials or manufacturing may make another mechanism better.

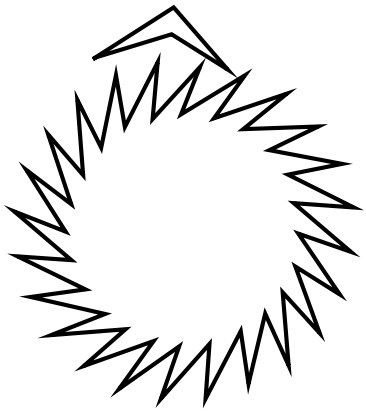


Figure 7: *A creative design for a reversing mechanism produced by our program.*

Acknowledgements

This research has been supported by the Swiss National Science Foundation (Fonds National de Recherche Scientifique).

References

[Faltings *et al.*, 1989] Boi Faltings, Emmanuel Baechler, and Jeff Primus. Reasoning about Kinematic Topology. *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Detroit, 1989.

[Faltings, 1990] Boi Faltings. Qualitative Kinematics in Mechanisms. *Artificial Intelligence*, 44(1), 1990.

[Faltings, 1991] Boi Faltings. Qualitative Models in Conceptual Design: A Case Study. *Artificial Intelligence in Design*, Butterworth-Heinemann, 1991.

[Faltings, 1992] Boi Faltings. A Symbolic Approach to Qualitative Kinematics. *Artificial Intelligence*,56(2), 1992.

[Forbus *et al.*, 1991] Ken Forbus, Paul Nielsen, Boi Faltings. Qualitative Spatial Reasoning: the CLOCK Project. *Artificial Intelligence*, 51(3), 1991.

[Gero, 1990] John Gero. Design Prototypes: A Knowledge Representation Schema for Design. *AI Magazine*, 11(4), 1990.

[Joskowicz and Addanki, 1988] Leo Joskowicz, S. Addanki. From Kinematics to Shape: An Approach to Innovative Design. *Proceedings of the National Conference of the AAAI*, St.Paul, August 1988.

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

[Williams, 1990] Katia Sycara, D. Navinchandra. Index Transformation Techniques for Facilitating Creative Use of Multiple Cases. *Proceedings of the 12th IJCAI*, Sydney, 1991.

[Williams, 1990] Brian Williams. Interaction-based Invention: Designing Novel Devices from First Principles. *Proceedings of the National Conference of the AAAI*, Boston, 1990.