Proceedings of DETC'01
2001 ASME Design Engineering Technical Conferences
September 9-12, 2001, Pittsburgh, Pennsylvania

DETC01/DTM-21684

SPATIAL REASONING ABOUT MECHANICAL BEHAVIORS

Levent Burak Kara
Mechanical Engineering Dept.
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
Email: lkara@andrew.cmu.edu

Thomas F. Stahovich
Mechanical Engineering Dept.
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
Email: stahov@andrew.cmu.edu

ABSTRACT

We describe an approach that uses causal reasoning and geometric reasoning to construct explanations for the purposes of the geometric features on the parts of a mechanical device. To identify the purpose of a feature, the device is simulated with and without the feature. The simulations are then translated into a "causal-process" representation, which allows qualitatively important differences to be identified. These differences reveal behaviors that the feature causes to occur and those it prevents from occurring. The focus of this paper is geometric reasoning techniques that reveal causal relationships between the caused and prevented behaviors. For example, these techniques can determine if a particular caused behavior is responsible for preventing a particular prevented behavior.

INTRODUCTION

Our main objective is to create methodologies that allow a computer program to automatically compute particular types of design documentation. There are a variety of different kinds of information that are commonly included in design documentation such as a history of the decision making process, a list of alternatives considered, and a description of the intended purpose of the each of the parts of the design. Our focus is on the latter type of information, which is necessary for modifying a design without introducing unintended side effects.

Towards our goal, we have built a computer program called ExplainIT-II that automatically computes and documents the purposes of geometric features on the parts of a mechanical device. Our program computes purpose by comparing a dynamic simulation of the original device (nominal simulation), to a simulation of the device with the feature removed (modified simulation). The differences between the two simulations are indicative of the feature's purpose. We distinguish between two kinds of purposes: Behaviors that occur only when the feature is present are behaviors that the feature causes. Conversely, the new behaviors that occur only after the removal of the feature are those the feature prevents.

To identify the behaviors caused and prevented by the feature, we have developed a new causal representation that describes a rigid-body dynamic simulation as sets of "causal processes." A causal-process represents the interactions between the components of a device during a time interval in which both the components' behaviors and the causes of the behaviors remain qualitatively uniform. To compare the two simulations, our program first represents them as sequences of causal-processes and prunes out any processes common to the two simulations. The purposes of the feature are then extracted from the remaining causal-processes: Causal-processes unique to the nominal simulation constitute the set of behaviors that the feature causes, those unique to the modified simulation represent behaviors prevented by the feature. Because these differences are already expressed in the form of causal descriptions, they can then be directly translated into human-readable explanations of purpose using text templates.

At this point, the description of a feature's purpose would be described in terms of two separate sets of causal-processes representing the behaviors caused and prevented by the feature. The resulting explanation would consist of a list of isolated processes. To obtain a more complete explanation of purpose, it is

necessary to identify any causal relationships that exist between the individual processes. For example, a behavior caused by the feature (desired behavior) might be preventing an undesired behavior that would have occurred in the absence of the feature. Likewise, a desired behavior may occur precisely because a behavior prevented by the feature fails to occur. Identifying such causal connections between observed and prevented behaviors would clearly give us a better understanding of the feature's purpose. The challenge, however, is that the these kinds of causal relationships often exist between processes from *different* simulations.[1]

The essential task is to determine if the presence or absence of a process in one simulation affects the occurrence of a process in a different simulation. Our approach to this task is to examine the causal-processes in relation to the device's configuration space (c-space). A c-space describes the allowed motions or kinematics of a device. As described in the following sections, each causal-process can be mapped onto one or more segments of the simulation trace in the c-space. By analyzing these traces, we can determine if the presence of a causal-process in one simulation would geometrically prevent a particular process in the other simulation.

The next section provides a brief overview of our causal-process representation. We then explain the geometric reasoning techniques we use to identify the causal relationships between the processes of different simulations and show how the results of this analysis are used to generate explanations of purpose.

Background: CAUSAL-PROCESS REPRESENTATION

To facilitate the comparison of two simulations, we have developed a representation that allows us to determine when two mechanical behaviors are the same. (See [8] for a more complete discussion of the representation.) We have found that mechanical behaviors can be conveniently represented as "causal-processes". A causal-process represents an interaction in which one set of bodies causes another set to do something. Both the behavior and the causes of the behavior remain qualitatively uniform during a causal-process. In mechanical devices, force causes (or prevents) motion. Hence we identify the causes of behavior by examining the flow of forces in the system.

Our representation of mechanical behavior consists of two types of causal-processes: those that keep an object in motion (dynamic processes) and those that keep an object in equilibrium (static processes). Both types of causal-processes are represented as acyclic, directed graphs in which the nodes represent bodies, springs, external forces or fixed surfaces in the system, and the arcs represent the causal relationships between the nodes. For a dynamic process, an arc represents either one object causing another to move, or an object causing a spring to store potential energy. For a static process, arcs represent the interactions between the body in equilibrium and all of the objects that keep it in equilibrium.

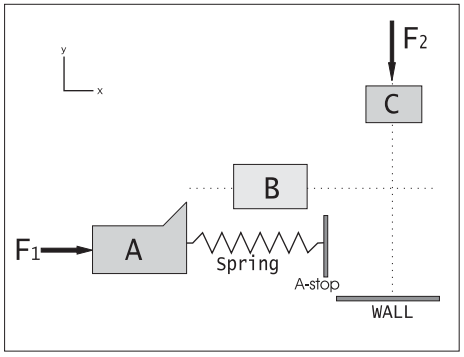

Figure 1. A 3-block system. F1 pushes block-A to the right. As A advances, it hits block-B and pushes it to the right. After F1 is turned off and A returns to its initial position, F2 is applied to block-C. Because B has been moved into C's path, C collides with B and stops.

In addition to describing the instantaneous properties of behavior, causal-processes also capture the causal history leading to the current processes by recording the list of all previous causal-processes that carried a given part to its current location. For a rigid-body to be involved in a causal-process, it must be in the right location at the right time. The history leading up to the current position of a body is thus one of the factors that enable the current process. Similarly, our representation records all the previous causal-processes that caused a spring to store potential energy. This stored energy allows the spring to cause new processes to occur. Having explicit records of past causal history allows our program to resolve ambiguities when comparing seemingly similar behaviors: for two causal-processes to be the same, the causal histories of the parts must be the same.

To illustrate our causal process representation, consider the system of 3 rigid blocks shown in Figure 1. Our goal is to compute the purpose of the triangular protrusion on block-A. In this hypothetical device, A is pushed to the right by external force F1. As A advances, it collides with B, causing B to move to the right. When F1 is turned off, A returns to its original position, leaving B at rest. Later, when F2 is applied to block-C, C strikes B and comes to rest. If, however, the triangular feature is removed and the experiment is repeated, C is no longer stopped by B, but instead collides with the wall. Hence, (as we shall see below) the purpose of the feature is to make A push B into C's path so that C strikes B instead of the wall.

Figure 2 shows the collection of causal-processes from the nominal and modified simulations (the history lists are not

[1] Sometimes there are causal relationships between processes from the same simulation. These can usually be handled in a more direct fashion through the facilities provided by our causal-process representation (specifically history lists). The primary focus of this paper is reasoning about causality between simulations.

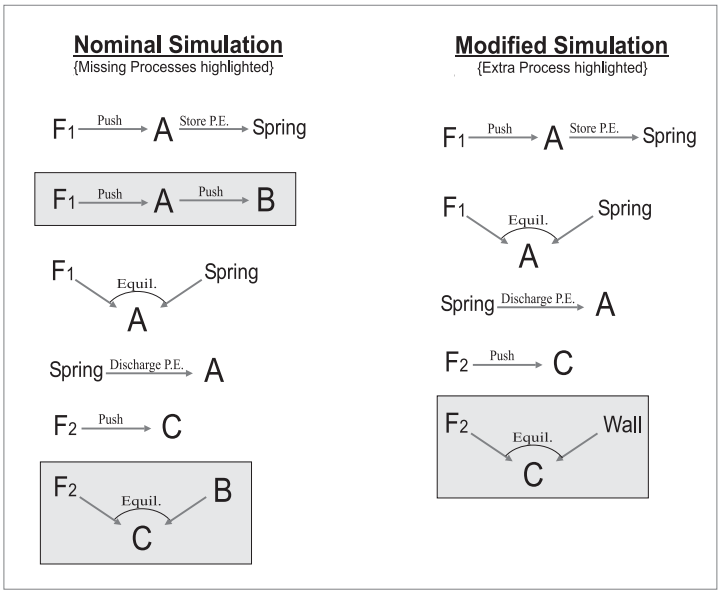

Figure 2. Causal-processes of the nominal and modified simulations. Processes highlighted with shaded boxes are the missing and extra processes.

shown). To compare the two simulations, a processes of elimination is used to find those processes that are unique to one or the other of the two simulations. Any processes that exist in both simulations are pruned; the rest are the unique ones. Determining if a causal-process from one simulation is the same as one from the other simulation is accomplished in the obvious way: Two dynamic processes are the same if they embody the same nodes connected with the same arcs, and the rigid-bodies and springs on the causal path have the same history lists. Similarly, two static processes match if the part in equilibrium and all the forces keeping the part in equilibrium are the same. Again, the rigid-bodies and springs that contribute to the equilibrium must have the same history lists.

Taking the history lists into account and comparing the two sets of causal processes, two processes unique to the nominal simulation and one unique to the modified simulation are identified (highlighted in Figure 2). To facilitate discussion, we separate the set of unique processes thus obtained into two categories. Causal-processes that occur only in the nominal simulation are called “missing processes” because when the geometry of the device is modified (feature is removed), they no longer occur. Likewise, the processes unique to the modified simulation are called “extra processes” because they do not ordinarily occur unless the device is modified.

After identifying the missing and extra processes, our program concludes that the triangular feature has three, *separate* purposes: the feature allows A to push B, it allows C to strike B, and it prevents C from striking the wall. We, however, know that these three processes are causally connected. The first process (A pushing B) enables the second (C and B in equilibrium), and prevents the third (C and the wall in equilibrium). As described in the next section, by detecting these kinds of causal connections between processes, our program can generate a more complete explanation of purpose, such as: “the feature allows A to push B into C’s path, which prevents C from striking the wall.”

REASONING WITH C-SPACES

Our task at this point is to determine the causal relationships between the missing and extra processes. The key question is, does the presence of a particular missing process prevent the occurrence of a given extra process? Or, alternatively does the presence of a particular extra process prevent the occurrence of a given missing process?

To answer this sort of question, it is necessary to relate processes from *different* simulations, because by definition, missing processes and extra processes do not coexist in the same simulation. This presents a significant challenge. For example, chronology, which is often useful for examining causality, is of no use because there is no way to reliably relate time in the two simulations – when the feature is removed, motions may be either faster or slower and thus the time at which a particular behavior occurs may change. Similarly, because the two processes may involve different bodies, it is often not possible to examine the history lists to determine if the two processes have any history in common.

Our examination of the nature of causal-processes however, revealed that their spatial characteristics frequently provide important information about how the individual behaviors are achieved. For the class of devices we consider, the geometry of the components and their relative locations in the device play crucial roles in causing or preventing mechanical behaviors. For instance, in the nominal simulation of the 3-block example (Figure 1), block-C strikes block-B precisely because block-A was able to push B into C’s path. However, when the triangular feature is removed from A, i.e. its geometry is modified, C strikes the wall instead, because now B fails to be at the “right location” to obstruct C’s path.

In order to examine the link between geometry and behavior, we examine the traces of the causal-processes through the device’s configuration space (c-space). By doing so, we can determine whether achieving a particular causal-process geometrically prevents the existence of another process.

C-spaces

This section provides a brief overview of c-space. See [7] for a more detailed discussion.

Configuration space is a representation that describes the al-

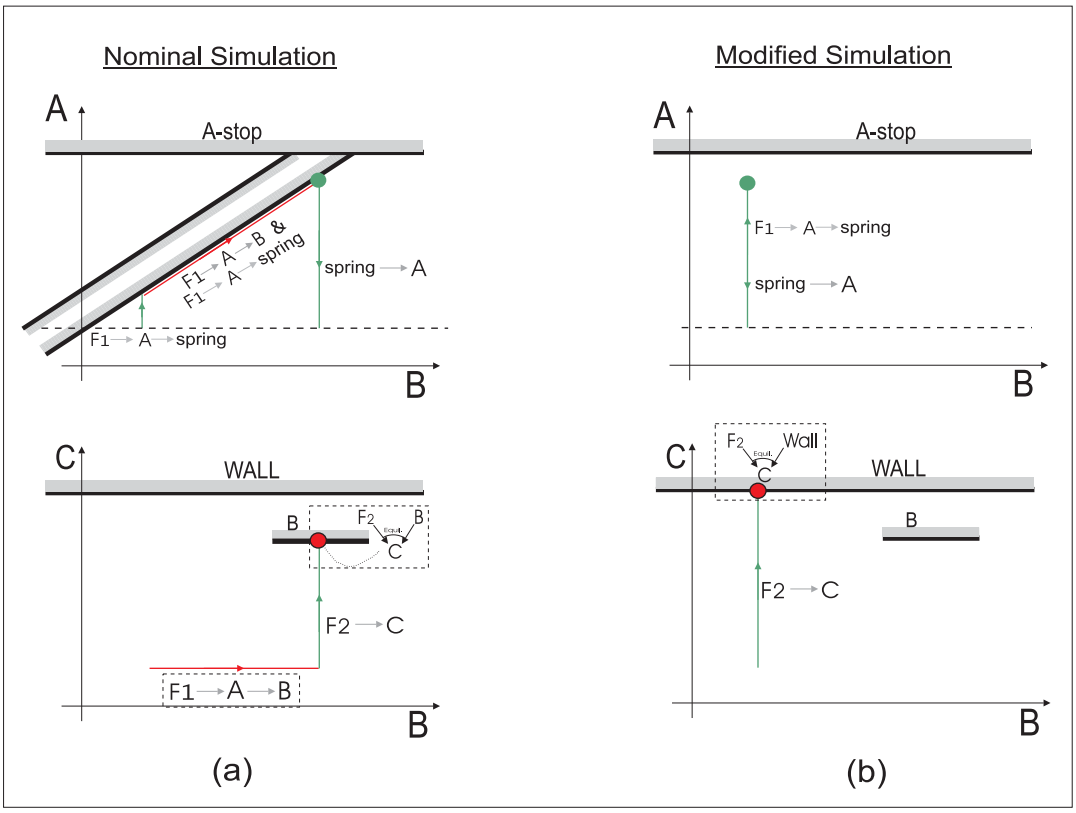

Figure 3. C-spaces of the nominal and modified simulations of the 3-block example.

lowed motions (kinematics) of a device. The axes of a c-space correspond to the position parameters of the bodies. The dimension of a c-space is thus equal to the number of degrees of freedom of the device. Because we restrict our attention to devices with fixed-axis parts (each part rotates about a fixed axis or translates along a fixed axis), we can represent a multi-dimensional c-space as a set of 2D projections, called configuration space planes (cs-planes).

Figure 3a shows the cs-planes from the nominal simulation of the 3-block example from Figure 1. Each cs-plane represents the interaction between a pair of fixed-axis bodies. The curves (lines) shown in the cs-planes are called configuration space curves (cs-curves). They represent the set of configurations in which the surfaces of one body touch those of another. The shaded region behind the cs-curves indicates blocked space, configurations in which one body would penetrate another. The unshaded region in front of the curves represents free space, configurations in which the faces do not touch.

The motions of the bodies can be represented as sequences of directed traces or "*trajectories*" through c-space. These can be projected into the individual cs-planes. To facilitate the causal analysis, we decompose the projections into monotonic segments. Each segment represents a state of the device during which the behaviors are uniform. However, by definition, an individual causal-process also represents a behavior in which the motion and its causes remain uniform. As a result, causal-processes can be easily mapped onto the segments.

Frequently, multiple causal-processes occur simultaneously. Thus, a given segment of the trajectory can correspond to multiple causal-processes. Furthermore, a given causal-process can span more than a single segment. Consider for example the cs-plane of blocks A and B in Figure 3a (top). The first causal process to occur is F1 pushing A, and A compressing the spring. This process spans the small vertical segment and the diagonal segment. At the beginning of the diagonal segment, A begins to push B. Thus, during the diagonal segment, there are two simultaneous processes: (1) F1 pushing A, compressing the spring, and (2) F1 pushing A, pushing B.

For parts in equilibrium, the corresponding trace on the cs-plane is a point. Hence, a static causal-process maps to a point on the cs-plane.

Modifying the geometry of a component results in an alteration of the cs-curves involving that component. For example, removing the triangular feature from block-A precludes its inter-

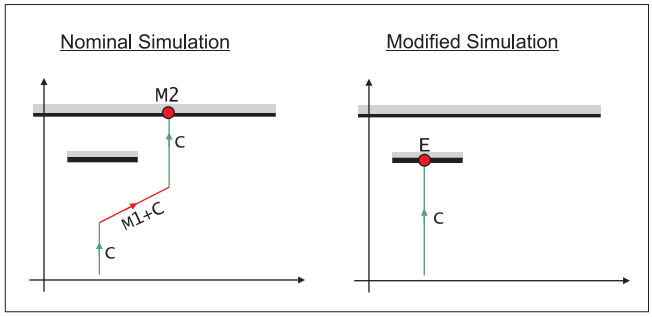

Figure 4. Cs-planes used in discussion of rule 1.

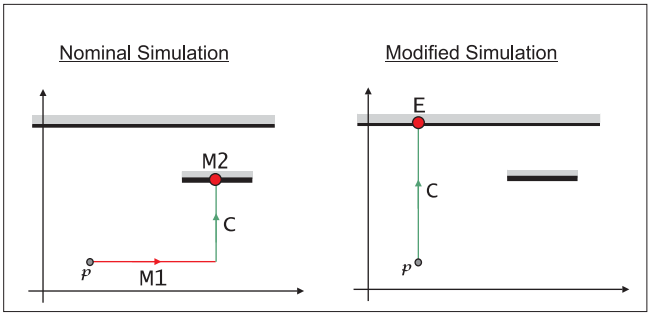

Figure 5. Cs-planes used in discussion of rule 2.

action with block-B. As shown in Figure 3b, this results in the removal of the diagonal cs-curve from the cs-plane of blocks A and B.

Computing Causal Relationships

In this section, we describe our rules for identifying the causal relationships between a missing and an extra process. As for notation, we use "M" to refer to a missing process, "E" for an extra process, and "C" for a process common to the two simulations. For sake of simplicity, we consider only the case in which a missing process prevents an extra process. However, the rules can be reversed in the obvious way to explore the converse case.

In our analysis, we distinguish between two types of missing processes: those that occur in free space and those that occur along a cs-curve. In the former case, to detect disablement, we examine the missing process to see if it deflects the common process such that the extra process is geometrically precluded. In the latter case, we determine if the cs-curve on which the missing process occurs, geometrically precludes the extra process. After examining a number of problems, we reduced these two kinds of analyses to a set of rules. The first two rules examine cases in which M occurs in free space. The third examines cases in which M occurs along a cs-curve.

Rule 1: *"Given processes M, E and C: If 1) C is a dynamic process, 2) E happens immediately after C in the modified simulation, 3) M in the nominal simulation occurs during C, and 4) C in the modified simulation passes through the point where M begins in the nominal simulation, then M disables E. "*

The first rule considers the situation in which the common process C was heading in the right direction for E to occur, but M intervened and deflected the trajectory. There are several characteristics that distinguish this situation. First, C must be a dynamic process, otherwise there would be no motion leading to new causal processes. Second, in the modified simulation, C ultimately leads to E, but in the nominal simulation E does not occur. Third, C is diverted by M in the nominal simulation. Fourth, C in the nominal simulation passes through the point at which M begins in the nominal simulation. Thus, if M had occurred in the modified simulation, it would have been able to deflect C. If all of the characteristics are present, then there is good evidence that M disables E.

To illustrate this rule, consider the hypothetical cs-planes shown in Figure 4. Two missing processes, M1 and M2 occur in the nominal simulation, whereas one extra process E occurs in the modified simulation. As shown, M1 happening simultaneously with C temporarily diverts C's upward trend. In the modified simulation, however, C extends uninterruptedly and ultimately leads to E. Furthermore, C in the modified simulation passes straight through the point where M1 would have begun in the nominal simulation. In other words, if M1 had occurred in the modified simulation, it would have diverted the trajectory and prevented E. Hence, we conclude that M1 disables E.

Rule 2: *"Given processes M, E and C: If 1) M is a dynamic process, 2) M occurs immediately before C, 3) E occurs immediately after C, 4) C in the modified simulation begins at the point at which M begins in the nominal simulation, then M disables E."*

In the previous rule, M intervenes in the middle of C and deflects it so that E is prevented. This rule, however, considers the case in which M occurs immediately before C, but still deflects C so that E does not occur. In this case, M is shifting C to a new location so that E is prevented. The distinguishing characteristics of this case are as follows. First, C must be a dynamic process, otherwise there would be no motion leading to new causal processes. Second, in the modified simulation, C leads to E, but in the nominal simulation E does not occur. Third, C in the nominal simulation begins at the end of M. Fourth, C in the modified simulation begins at the location at which M begins in the nominal simulation. If all of the characteristics are present, there is good evidence that M disables E.

Consider the cs-planes in Figure 5. The trajectories of both simulations have reached the same point *p*. In the modified simulation, process C begins at point *p* and ultimately produces E. However, M1 in the nominal simulation shifts C away from point *p* to a new position, and as a result C no longer leads to E. Hence,

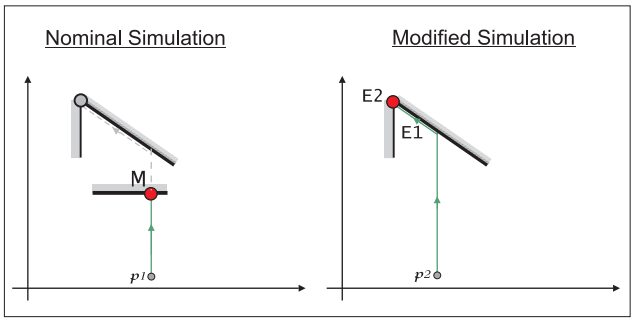

Figure 6. Cs-planes used in discussion of rule 3.

we conclude that M1 disables E.

Rule 3: *"Given two processes M and E, if removing process M from the nominal simulation, together with the geometric constraints that cause M, produces E, then M disables E."*

This rule considers the case in which M occurs along a cs-curve. Frequently, the presence of a cs-curve in the nominal simulation may prevent an extra process E from occurring. This can happen if the cs-curve causes M by either stopping [2] or redirecting a trajectory that would otherwise have lead to E. To determine if M disables E, we remove M together with the cs-curve on which it occurs, and then extend the trajectory farther through the cs-plane to "envision" the new processes that might subsequently happen. The envisionment involves computing a set of processes that are generated by the interaction of the extended trajectory with the other cs-curves in the plane. If E is found among the envisioned processes, then we have suggestive evidence that M disables E.

Consider processes M, E1 and E2 in Figure 6. In the nominal simulation, the vertical trajectory collides perpendicularly with the horizontal cs-curve producing static process M. In the modified simulation, however, the horizontal cs-curve is absent and E1 and E2 occur. To determine if M was preventing these extra processes from occurring, we hypothetically remove M and the horizontal cs-curve. We then predict the possible new processes that might happen. In this imagined scenario, the path of the trajectory would be cleared from obstacles and thus could extend without obstruction until it collides with the diagonal cs-curve. Upon collision, the trajectory would start following the diagonal cs-curve until reaching the vertical cs-curve, at which point the trajectory's further advance would be precluded. The new processes created during this "envisioned" traversal resemble the two extra processes: The segment of the trajectory following the diagonal cs-curve is similar to dynamic process E1 because they both occur along the same cs-curve with the same direction of traversal. Likewise, the equilibrium process created at the end of the diagonal cs-curve is similar to E2 because they are both static processes and involve the same cs-curves. As shown, E1 and E2 *could* happen in the nominal simulation if process M and the horizontal cs-curve were removed. From this, we have suggestive evidence that M disables both E1 and E2.

This rule involves extending a trajectory and envisioning the new processes that might subsequently occur. In extending the trajectory we assume it continues to tend to move in the direction it was moving just prior to when M would have occurred. If the extended trajectory strikes a new cs-curve, we assume it is deflected by it. However, if the trajectory is perpendicular to the new cs-curve, we assume the trajectory will stop. Similarly, we assume the trajectory will stop if it reaches the intersection of two cs-curves that could block the trajectory, such as the upside down "v" in Figure 6. This set of assumptions produces a set of new processes that might plausibly occur. If the extra process E is found among these envisioned processes, then we have evidence that M could disable E. However, because the envisionment was based on a number of simplifying assumptions, even if E is not contained in the set of envisioned processes, it is still possible that M disables E. Conversely, finding E among the envisioned processes is not proof of disablement. Nevertheless, in the sample problems we have examined, this rule did produce the correct results.

The above rules consider direct causal relationships between pairs of missing and extra processes. Sometimes there are additional indirect relationships between pairs of pairs. For example, sometimes a missing process may be disabling an extra process by causing another missing process that in turn disables the extra process. In such situations, we deduce that the first missing process *indirectly disables* the extra process. Similarly, there may be indirect causal relationships between two missing processes (or extra processes). For instance, a missing process may *indirectly enable* a future missing process by eliminating an extra process that would have disabled it. To identify these indirect relationships, we first apply rules 1 – 3 to detect direct disablement. We then examine the identified direct causal relationships to identify any indirect relationships.

Rule 4: *"Given processes M1, M2 and E: If M1 and M2 both disable E, and M1 occurs in the history list of M2, then M1 indirectly disables E by producing M2."*

This rule examines the situation in which a missing process (M1) indirectly disables an extra process (E) through the assistance of another missing process (M2). The issue here, is that rules 1 – 3 cannot distinguish between direct and indirect disablement. Thus, if those rules determine that M1 and M2 each individually disable E, it is still possible that one of them is indirectly disabling E by causing the other. This can be determined by examining the history lists of the two missing processes. Recall that a history list is the list of all causal-processes that lead

[2] The trajectory will stop if it collides with a cs-curve perpendicularly, or if it is entrapped between two cs-curves.

to the occurrence of a particular causal-process.[3] For example, if the history list of M2 contains M1, then M1 is one of the causes of M2. Thus, if M1 does not occur, neither will M2. However, if M2 does not occur, the previous analysis with rules 1 – 3 indicates that the extra process, E, will occur. Thus, M1 indirectly disables E with the help of M2.

To illustrate this rule, consider once again the cs-planes in Figure 5. Rule 2 determines that M1 disables E as explained previously. Similarly, rule 3 determines that M2 disables E. Additionally, the history list of M2 contains M1. This can be observed by noting that M1 is part of the trajectory leading to M2. Thus, according to the new rule, we can conclude that M1 indirectly disables E by causing M2. An informal examination of the trajectories in Figure 5 gives the intuition behind this rule. We see that if M2 did not occur (i.e., the short cs-curve were removed) then E would still occur, even if M1 occurs. E would simply occur at a different location on the same cs-curve. Thus, M1 alone does not disable E. It does so indirectly by causing M2.

Rule 5: *"Given processes M1, M2 and E: If M1 disables E, and E disables M2, then M1 indirectly enables M2 by disabling E."*

Our final rule examines indirect enablement. If a missing process M1 disables an extra process E, and E disables another missing process M2, then we can conclude that M1 indirectly enables M2 by disabling E. Note that this kind of indirect enablement cannot be identified by examining M2's history list: Because M1 does not directly cause M2, it will not appear in the list.

The cs-planes shown in Figure 4 provide an example of this case. Using rule 1, we previously determined that M1 disables E. By applying rule 3 in the reverse direction (i.e., considering the case in which an extra process disables a missing one), we can determine that E disables M2. From this, we conclude that M1 indirectly enables M2 by preventing E from happening. In this case, however, M1 is also a direct cause of M2 because M1 is in the history list of M2 (i.e., M1 is on the trajectory leading to M2).

Returning to our 3-block example, we can now determine how the previously identified missing and extra processes are related to each other. The first missing process was [F1 pushing A pushing B] and the second was [C in equilibrium at B]. The only extra process was [C in equilibrium at the wall]. These processes are encircled with dashed lines in Figure 3. From the second rule, we determine that the process [F1 pushing A pushing B] disables the process [C in equilibrium at the wall]. Similarly, from the third rule, the process [C in equilibrium at B] disables [C in equilibrium at the wall]. Note that both missing processes individually disable the extra process. Additionally, the history list of the second missing process contains the first missing process. This can be observed by noting that the first missing process is on the trajectory leading to the second. Therefore, from the fourth rule we can conclude that [F1 pushing A pushing B] *indirectly disables* [C in equilibrium at the wall] by causing [C in equilibrium at B]. With appropriate text generation facilities, this could be translated into: "the feature prevents C from coming to rest on the wall by enabling A to push B so that C comes to rest on B."

RELATED WORK

ExplainIT-II builds upon a previous system called ExplainIT [9]. Although the idea of comparing simulations and identifying the differences is common to both systems, there are a number of significant differences between them. The previous approach directly compared the two simulations to identify the first "significant" difference. In order to determine purpose, the system then performed a causal analysis of the differing behaviors. In ExplainIT-II, however, we perform causal analysis prior to comparing the two simulations. One of the primary advantages of this new approach is that it allows us to accurately identify when behaviors are the same, even when they occur in different orders in the two simulations. This new capability is essential for analyzing more complicated devices including those with cyclic behavior.

Design rationales are descriptions of why a design was designed the way it was. The descriptions of purpose that ExplainIT-II computes are one form of design rationale. There is a large and growing body of work in design rationale capture and construction. [1] and [4] offer good overviews to this work. However, much of that work is focused on tools for managing documentation that is human generated whereas our work aims to automatically compute documentation.

David Franke [3] has devised a language called Ted for representing teleological descriptions. Purposes of components are expressed in terms of behaviors prevented, guaranteed or introduced by particular components. This work is similar to ours in that it identifies the purpose of a component by examining the behaviors it adds to or removes from a system. However, Ted requires the user to enumerate both the desirable and undesirable behaviors of the device. Our program, on the other hand, identifies purpose by comparing two simulations of the device. In our study, we also address the question of "how" the component performs its identified purpose by generating text descriptions of the causal processes that are computed as differences.

There has been some previous work in trying to "understand" the behaviors of mechanisms. Forbus *et. al.* [2] describe a system that produces descriptions of the motions of the parts of a device. They decompose the device's configuration space into regions of uniform contact called "places," producing a "place vocabulary" for the device. They generate a description of the device's behavior by enumerating the sequence of places that are visited when the external inputs are applied to the device. Sacks

[3] See [8] for a complete discussion of history lists.

and Joskowicz [5] describe a similar system that partitions configuration space into a region diagram rather than a place vocabulary. These systems produce descriptions of what happens but do not derive causal relationships. Thus they do not provide explanations for why things happen.

Stahovich *et. al.* [6] describe a system for computing qualitative rigid-body dynamic simulations. That system uses a qualitative version of Newton's laws that are similar to the techniques used here for tracking the flow of causality through a device.

DISCUSSION AND CONCLUSION

Our work concerns the task of "purpose recognition." To compute the purpose of a geometric feature on a part, we simulate the behavior of the device with and without that feature. We then translate the two simulations into a "causal-process" representation and identify processes that exist in one simulation but not the other. Any such processes are indicative of the feature's purpose. This analysis results in a list of isolated behaviors that the feature either causes or prevents.

The focus of the current work is on identifying causal connections between the behaviors the feature causes and those it prevents. The sort of question we are trying to answer is: "does the feature cause one behavior to occur by preventing another from happening or vice versa?" Identifying these sorts of causal relationships allows us to construct more complete explanations of purpose. The difficulty is that identifying these relationships requires reasoning about why things *do not* happen. Most causal reasoning techniques address the converse problem of why things do happen. Our approach to the problem is to augment causal reasoning with geometric reasoning. To determine if one process is preventing another from occurring, we use geometric analysis to determine if the former geometrically precludes the latter.

At the present, we consider the results of our analysis to be suggestive evidence of disablement (and enablement) rather than a rigorous proof. More experimentation will be necessary to determine the accuracy of our rules. Additionally, our rules are fairly general in that they cover the two categories of missing processes: those that occur in free space and those that occur along a cs-curve. As we examine more complicated devices, we will likely encounter the need for additional rules. However, based on our current experience, we expect that we will need only a handful of additional rules.

We have fully implemented all of our rules in software, which we tested on a small set of example problems. The example presented here (Figure 1) was the simplest of the set; the others were moderately more complex. In all cases, our program successfully identified the correct causal relationships. Our work is clearly at an early stage. However, our current results suggest that geometric reasoning is a powerful tool for generating causal explanations of mechanical behavior.

Acknowledgements

This work has been supported by the National Science Foundation under Award Number 9813259.

REFERENCES

[1] Paul Chung and René Bañares-Alcántara Editors. Special issue: Representation and use of design rationale. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, 11(2), 1997.

[2] Ken D. Forbus, Paul Nielsen, and Boi Faltings. Qualitative spatial reasoning: The clock project. *Artificial Intelligence*, 51(9), 1991.

[3] David W. Franke. Deriving and using descriptions of purpose. *IEEE Expert*, pages 41–47, April 1991.

[4] Thomas Gruber, Catherine Baudin, John Boose, and Jay Weber. Design rationale capture as knowledge acquisition tradeoffs in the design of interactive tools. Technical Report KSL 91-47, Stanford University, Knowledge Systems Laboratory, 1991.

[5] Elisha Sacks and Leo Joskowicz. Automated modeling and kinematic simulation of mechanisms. *Computer-Aided Design*, 25(2):106–118, February 1993.

[6] Thomas F. Stahovich, Randall Davis, and Howard Shrobe. Qualitative rigid body mechanics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, 1997.

[7] Thomas F. Stahovich, Randall Davis, and Howard Shrobe. Generating multiple new designs from a sketch. *Artificial Intelligence*, 104(1–2):211–264, October 1998.

[8] Thomas F. Stahovich and Levent Burak Kara. A representation for comparing simulations and computing the purpose of geometric features. *AIEDAM*, 2001. in press.

[9] Thomas F. Stahovich and Anand Raghavan. Computing design rationales by interpreting simulations. *ASME Journal of Mechanical Design*, 122(1):77–82, 2000.