Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2002), *16*, 363–384. Printed in the USA.
Copyright © 2002 Cambridge University Press 0890-0604/02 $12.50
DOI: 10.1017/S0890060402165036

Causal reasoning using geometric analysis

LEVENT BURAK KARA AND THOMAS F. STAHOVICH
Mechanical Engineering Department, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, USA

(RECEIVED November 7, 2001; ACCEPTED September 23, 2002)

Abstract

We describe an approach that uses causal 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 the behaviors caused and prevented by the feature and thus provide useful cues about the feature's purpose. A clear understanding of the feature's purpose, however, requires a detailed analysis of the causal connections between the caused and prevented behaviors. This presents a significant challenge because one has to understand how a behavior that normally takes place affects (or is affected by) another behavior that is normally absent. This article describes techniques for identifying such elusive relationships. These techniques employ a set of rules that can determine if one behavior enables or disables another that is spatially and temporally far away. They do so by geometrically examining the traces of the causal processes in the device's configuration space. Using the results of this analysis, our program can automatically generate text output describing how the feature performs its function.

Keywords: Causal Reasoning; Computing Purpose; Configuration Space; Design Rationale Construction; Geometric Reasoning; Simulation; Spatial Reasoning

1. INTRODUCTION

This work is motivated by the desire to decrease the cost of documenting a design. Good documentation is essential for performing a variety of common tasks during the product life cycle; however, creating good documentation places a significant burden on the designer. Furthermore, it is usually not the designer but is instead others downstream in the product life cycle who benefit from this effort.

Our goal is to create methodologies for automatically computing particular types of documentation. There are a variety of different kinds of information commonly included in design documentation. For example, it can contain a history of the decision making process, a list of the alternatives considered, and a description of the intended purpose of each part of the design. Our work is concerned with the latter type of information, which is necessary for modifying a design without introducing unintended side effects. This kind of information is essential for resolving conflicts in distributed and collaborative design, modifying a design to make it more easily manufacturable, redesigning a product to add new (marketing) features, and adapting an existing design to a new application.

Toward 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. We have focused on features because our informal analysis has revealed that this is what people typically do. It is common to find documentation of the form "the notch on part X is intended to . . ." As further justification, the work by Knuffer and Ullman (1990) indicates that questions about the construction, purpose, and operation of features are among those most frequently asked by professional engineers during a redesign exercise. In the design speaking-aloud protocol studies they conducted, over 25% of the questions concerned features. The importance of features in understanding the operation of a device is not surprising when one considers that if the designer bothered to create a feature, it most likely has some intended purpose.

Our approach identifies purpose by comparing a dynamic simulation of the original device (nominal simulation) to a simulation of the device with the feature removed

Reprint requests to: Thomas F. Stahovich, Mechanical Engineering Department, Scaife Hall 415, 5000 Forbes Ave., Carnegie Mellon University, Pittsburgh, PA 15213, USA. E-mail: stahov@andrew.cmu.edu

(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 a feature, we developed a 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 both. 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 be directly translated into human-readable explanations of purpose using text templates.

At this point, the description of a feature's purpose would be expressed 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 prevent 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 desired and prevented behaviors would clearly give us a better understanding of the feature's purpose. The challenge, however, is that 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. 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.

2. 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 Stahovich and Kara, 2001, for a 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 an 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.

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. Consider blocks A and B shown in Figure 1. Initially, F1 pushes A into B's path. Then as F2 pushes B downward, B collides with A and the two blocks start moving together. In this scenario, B is able to interact with A precisely because F1 has put A in a position that allows the collision and F2 pushed B toward A. Therefore, the history list of process [B pushing A] contains the earlier processes [F1 pushing A] and [F2 pushing B].

Similarly, our representation records all of the previous causal processes that caused a spring to store potential en-

[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 article is reasoning about causality between simulations.

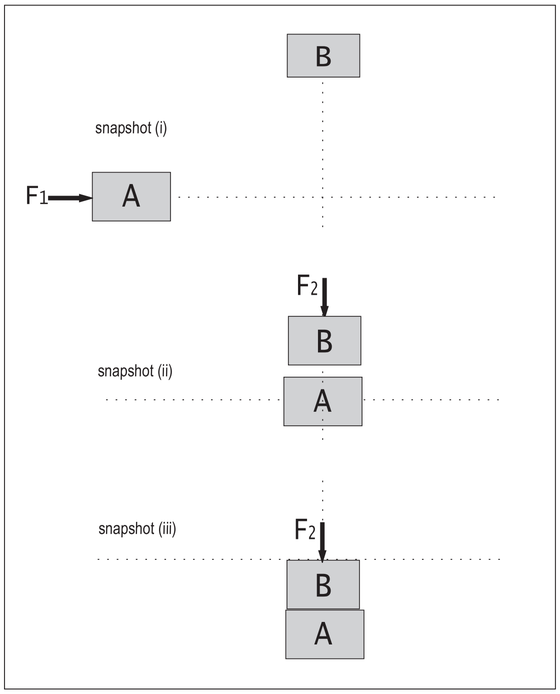

Fig. 1. Three snapshots of a two-block system illustrating the concept of history lists. The last process of [B pushing A] cannot occur unless the earlier processes [F1 pushing A] and [F2 pushing B] take place. The history list of [B pushing A] will therefore contain these two processes.

ergy. 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 the causal-process representation, consider the system of four blocks shown in Figure 2. Our goal is to compute the purpose of the triangular protrusion on block B. In this hypothetical device, each block is constrained to translate along a single fixed axis: block A moves horizontally and blocks B, C, and D move vertically. (Note that in Fig. 1, A could move both horizontally and vertically whereas here all blocks have only 1 degree of freedom.) The device's operation begins when external force F1 begins to push A to the right. As A advances, force F2 begins pushing B downward. Then, before A can reach C, B collides with C and pushes it out of A's path. After the spring returns B to its initial position, A's path is clear and A moves uninterrupted until coming to rest at A-stop. Later, when F3 is applied to block D, D strikes A and comes to rest.[2]

If the triangular feature is removed from B and the experiment is repeated, the behavior will be quite different. B will no longer displace C, and thus A will collide with C and stop. Furthermore, because A fails to move into D's path, D will now strike D-stop rather than block A. Figure 2(b,c) shows the final configurations of the blocks in the two experiments.

Figure 3 shows the collection of causal processes from the nominal and modified simulations (the history lists are not shown). To compare the two simulations, a process 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 a direct fashion: two dynamic processes are the same if they include the same nodes connected with the same arcs and the rigid bodies and springs 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 must have the same history lists.

Taking the history lists into account and comparing the two sets of causal processes, three processes unique to the nominal simulation and two unique to the modified simulation are identified (highlighted in Fig. 3). 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 (a 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. For simplicity, we will use M to refer to a missing process and E for an extra process.

Through this analysis, the program identifies five purposes of the feature. Missing process M1 indicates that the feature enables B to push C. Missing process M2 indicates that the feature enables A to be in equilibrium at A-stop. Missing process M3 indicates that the feature enables D to be in equilibrium against A. Extra process E1 indicates that the feature prevents A from being in equilibrium against C. Extra process E2 indicates that the feature prevents D from being in equilibrium at D-stop.

At this point in the analysis, the five purposes are *separate* from one another. However, we know that they are causally connected: the feature causes B to push C out of A's path, thus preventing A from striking C. (Missing process M1 prevents extra process E1.) Because A does not strike C, it strikes A-stop instead. (The absence of E1 enables missing process M2.) Because A strikes A-stop, D collides with A, rather than striking D-stop. (M2 causes missing process M3 and prevents extra process E2.) Thus, the ultimate purpose of the feature is to prevent D from striking D-stop. Note that the program is able to reach this conclusion even though the feature never *directly* interacts with D.

Identifying these kinds of connections requires elucidating the frequently complex causal relationships between

[2]This example is used because it illustrates nearly all of the geometric reasoning techniques we have developed. The device is a form of an interlock, because it prevents A and D from interacting unless B has been pressed and released. Such a device could be used as a switch that cannot be accidentally operated.

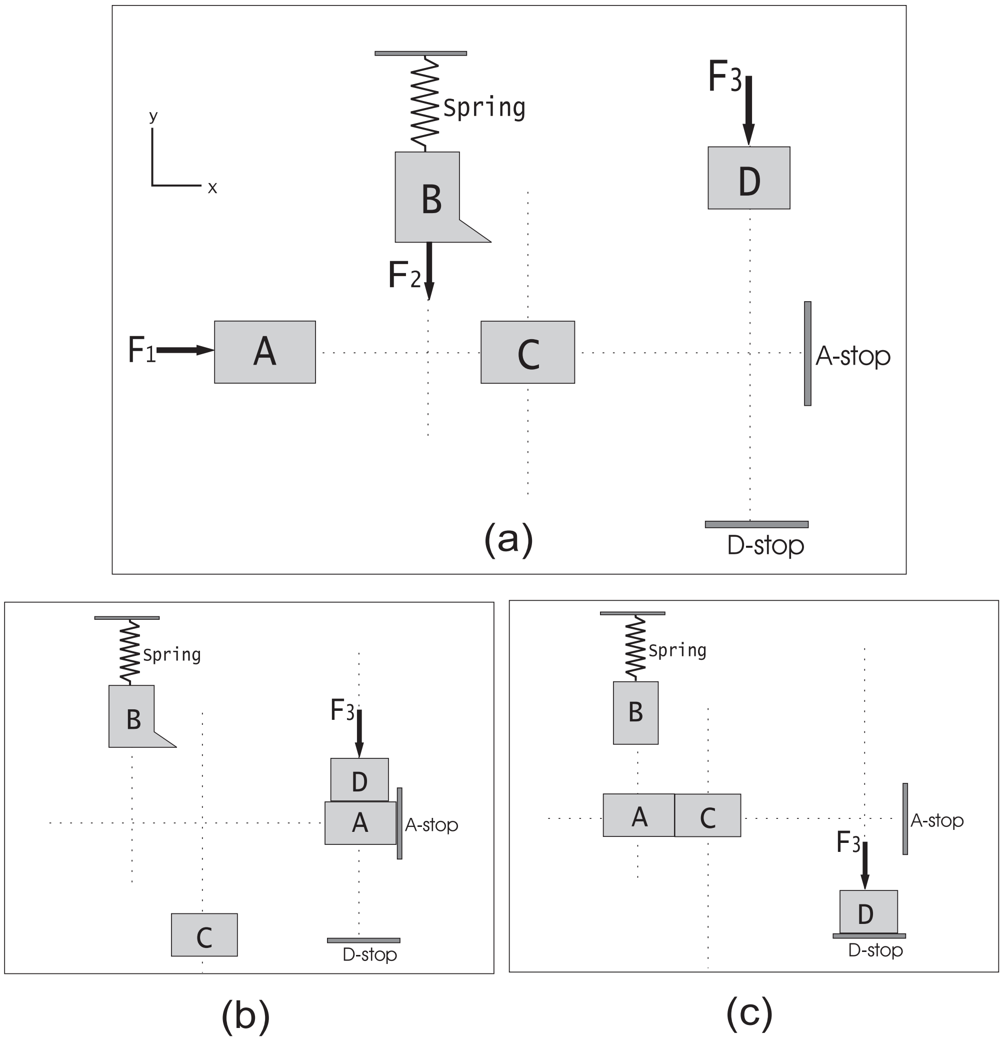

Fig. 2. A hypothetical device consisting of four blocks: (a) the initial configuration, (b) the final configuration of nominal simulation, and (c) the final configuration of modified simulation.

spatially and chronologically distant behaviors. As described in the next section, our program identifies such relationships by *geometrically* analyzing the causal process representation of behaviors.

3. 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 limited use because there is often 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, often it is 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 and relative locations of the components play crucial roles in causing or preventing mechanical behaviors. For instance, in the nominal simulation of the four-block example, block D strikes block A precisely because block B pushed C out of A's path, thereby allowing A to advance to A-stop. However, when the triangular feature

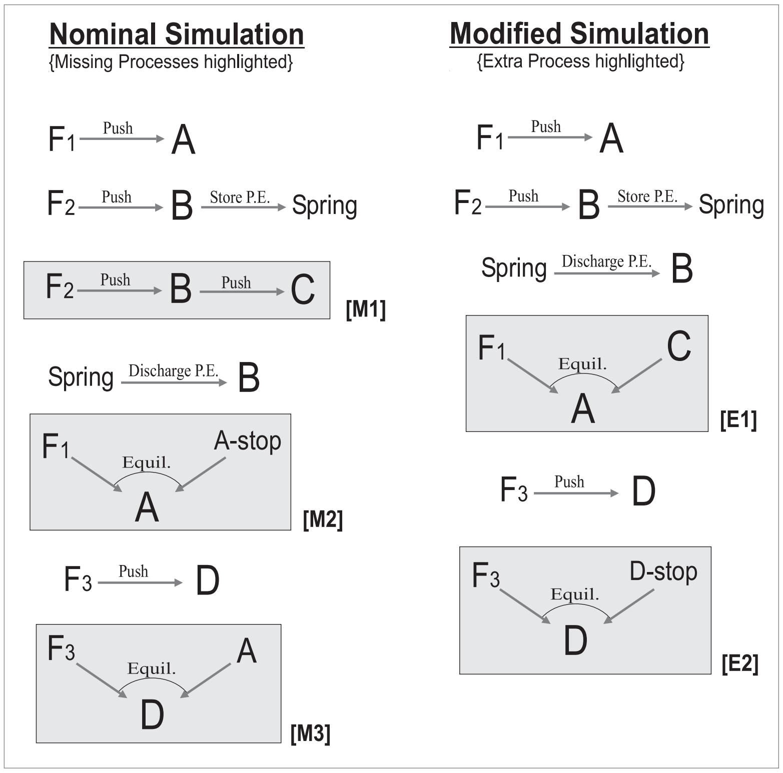

Fig. 3. The causal processes of the nominal and modified simulations. The processes highlighted with shaded boxes are the missing and extra processes.

is removed from B, D strikes D-stop instead, because now A fails to be at the "right location" to obstruct D'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 or enables the existence of another process.

3.1. C-Spaces

This section provides a brief overview of c-space. See Stahovich et al. (1998) for a more detailed discussion.

Configuration space is a representation that describes the allowed 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 multidimensional c-space as a set of 2-dimensional (2-D) projections, called configuration space planes (cs-planes).

Figure 4(a) shows the cs-planes from the nominal simulation of the four-block example from Figure 2. Each cs-plane represents the interaction between a pair of fixed-axis bodies. The shaded 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 regions behind the cs-curves indicates blocked space, configurations in which one body would penetrate another. The unshaded regions in front of the curves represent 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 B and C in the middle of Figure 4(a). The first causal process to occur is F2 pushing B and B compressing the spring. This process spans the small vertical segment and the diagonal segment.

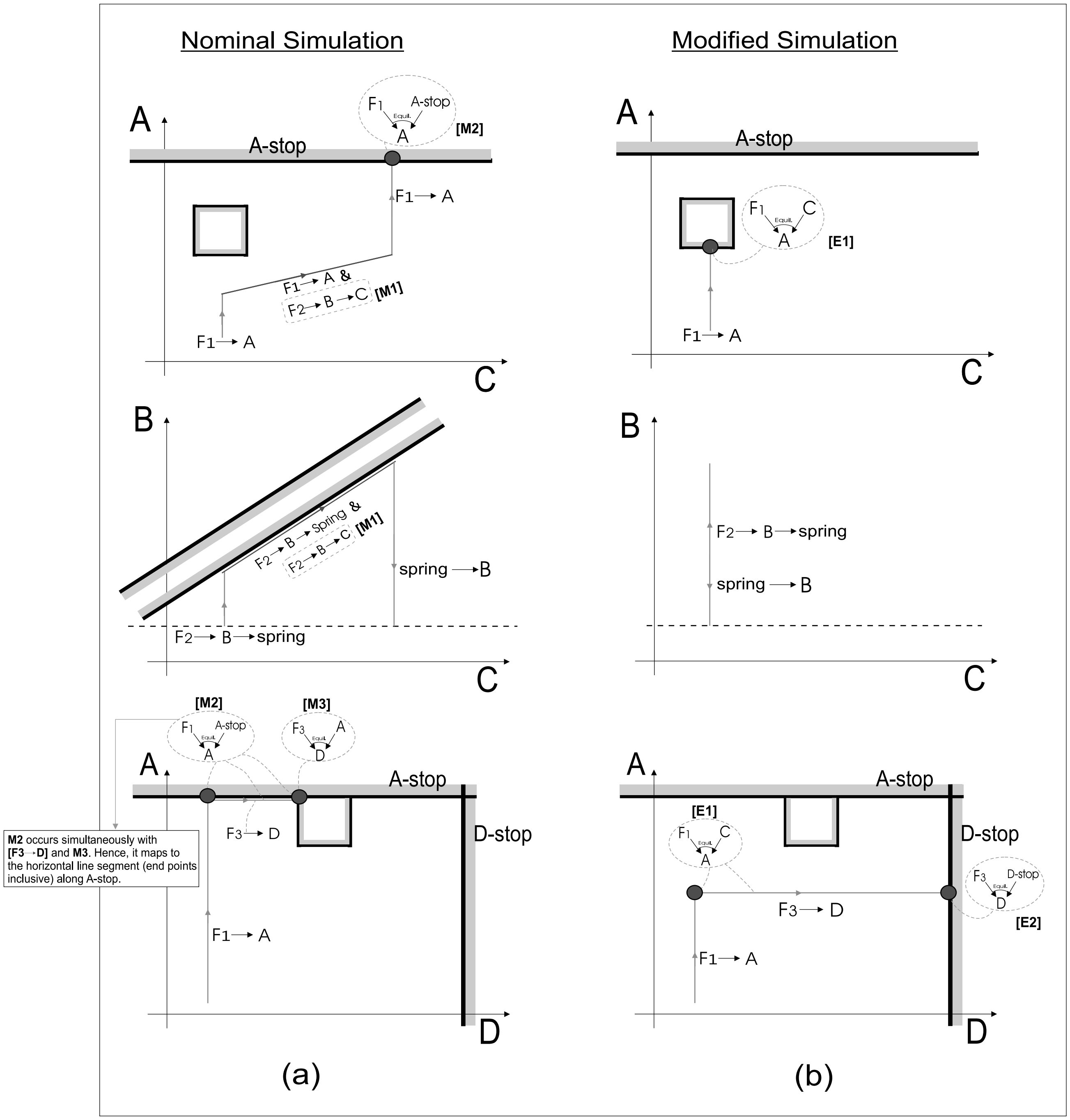

Fig. 4. The c-spaces of the (a) nominal and (b) modified simulations of the four-block example. The missing and extra processes are encircled with dashed lines. Blocks A and B, B and D, and C and D do not interact with one another in the simulations and thus their cs-planes are not shown.

At the beginning of the diagonal segment, B begins to push C. Thus, during the diagonal segment, there are two simultaneous processes: [F2 pushing B, compressing the spring] and [F2 pushing B, pushing C].

A static process describes a situation in which a body does not move. Hence, a static process will often map to a point on the cs-plane. However, if the static process occurs simultaneously with a dynamic process, the corresponding trace will be a line. The cs-plane of blocks A and D in the bottom of Figure 4(a) provides an example of this. The static process [M2: A in equilibrium at A-stop] and the dynamic process [F3 pushing D] occur simultaneously: block

A continues to be in equilibrium at A-stop while F3 pushes D. Therefore, both M2 and [F3 pushing D] map to the horizontal line segment along A-stop.

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 B precludes its interaction with block C. As shown in Figure 4(b), this results in the removal of the diagonal cs-curves from the cs-plane of blocks B and C.

3.2. Computing causal relationships

In this section, we describe our rules for identifying the causal relationships between missing (M) and extra (E) processes, where C is a process common to the two simulations. For the sake of clarity, the rules in the following paragraphs consider only the case in which a missing process M prevents an extra process E. Although not discussed further, our program considers the converse case, that is, E preventing M, by simply applying the converse of the rules.

Our rules work directly from 2-D projections of the causal processes on the cs-planes. By definition, a causal process will be located in one of two regions of a cs-plane: the free space, where bodies are not touching, or along a cs-curve, where bodies are in contact. Our studies have shown that the way in which M disables E is strongly influenced by whether M occurs in free space or along a cs-curve. We thus reduced the analysis to these two cases. When the M process occurs in free space, it disables the E process by actively shifting, deflecting, or terminating the C process that would otherwise produce E. The first three rules, which consider such cases, differ in the way in which M alters C. By contrast, when the M process is along a cs-curve, the cs-curve participates in disablement by intervening and distorting a trajectory that could potentially lead to the E process. For these cases, detecting disablement requires removing the cs-curve and *envisioning* probable new processes. Rule 4 explains our techniques for this.

The first four rules identify *direct* causal relationships between causal processes. By direct, we mean relationships that involve only one missing process and one extra process. Such relationships usually exist between temporally and spatially proximate processes. However, a causal process may *indirectly* influence a seemingly unrelated process via one or more intermediate processes. Rules 5 and 6 use the results of the first four rules to explore such circuitous causal relationships.

3.2.1. M in free space

When M is in free space, it can disable E by either shifting, deflecting, or terminating C. Figure 5 shows an example. Both simulations begin when F1 starts pushing block A

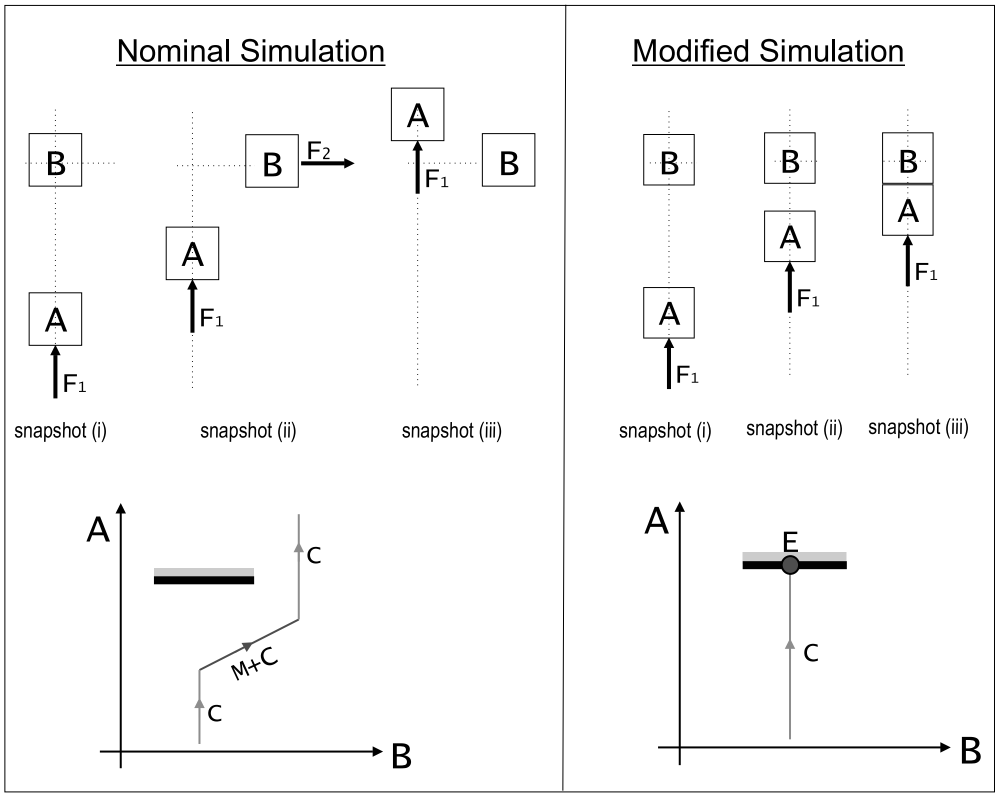

Fig. 5. An example of M disabling E by deflecting C. Snapshots of three sequential instances are shown in each simulation. Process C is the common process in which F1 pushes A toward B. Process M is F2 pushing B to the right. Process E is A striking B.

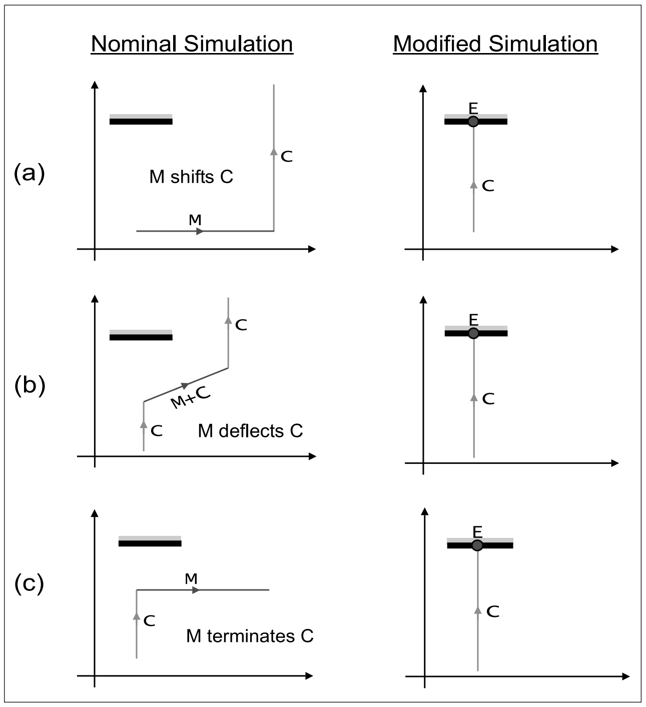

Fig. 6. Three ways M can alter C: (a) M shifts C by taking place before C, (b) M deflects C by taking place during C, and (c) M terminates C by occurring after C. In all three cases, E is disabled due to M's intervention in C.

toward block B (common process C). In the nominal simulation, as A is approaching B, F2 displaces B to the right (process M), thus clearing A's path. As a result, block A continues its motion without colliding with B. However, in the modified simulation, because B remains in A's path, A strikes B and stops (process E). In this case, it is clear that the M process disables the E process by *deflecting* the C process.

The above example illustrates how M, occurring simultaneously with C, can disable E. Similarly, M may disable E by taking place immediately before or immediately after C. Figure 6 illustrates all three cases: if M occurs before C it can shift C to a new location, if M occurs during C it can deflect C, and if M occurs after C it may terminate C. Note that in all three of these cases, E occurs strictly after C. In general, however, both M and E can each occur either before, during, or after C. This gives rise to nine distinct scenarios as shown in Table 1. The table summarizes the types of analyses we use to determine if M disables E. (The rules listed in the table are described below.)

Chronology does impose certain restrictions on causality. Our approach exploits these restrictions to rule out causal relationships that are physically impossible. These cases are indicated by × in Table 1. The principal idea here is that the future cannot influence the past. If process M occurs temporally after E, then it is not possible for M to influence E, and hence M cannot disable E. To determine the temporal ordering between M and E, we observe the processes'

Table 1. *Types of analyses used to determine if M disables E*

	E Before C	E During C	E After C
M before C	ND	Rule 1	Rule 1
M during C	×	Rule 2	Rule 2
M after C	×	×	Rule 3

Different analyses are needed, depending on the processes' positions relative to C. Rule numbers indicate cases that require geometric analysis. ×, entries need not be analyzed because temporal ordering of M and E rules out disablement; ND, the case where causal relationships cannot be determined.

ordering relative to C. The key here is that the common process corresponds to a time interval during which the nominal and modified simulations are in similar states. Hence, if M occurs after C in the nominal simulation and E occurs before C in the modified simulation, then M must be a process that would normally take place after E. From this, we conclude that M cannot influence E.

For cases in which chronology does not rule rule out causal relationships, we use geometric analysis to identify any causal relationships. The rules listed in Table 1 indicate the various geometric analyses that are used in each situation. There is only one special case. If M and E both occur before C, then we have no means of identifying causal relationships between them. This case is indicated by ND in Table 1. To determine causal relationships, our analysis typically examines how the presence or absence of a process alters the subsequent processes. The difficulty in this scenario is that the same process (C) occurs regardless of whether M occurs and E occurs. In this case, there are no immediate downstream consequences of the presence or absence of M and E. In such cases, there may be no casual relationships between M and E or the relationship may be subtle.

In the following paragraphs, we present the rules listed in Table 1. Note that all three rules apply to cases in which M is in free space. (Rule 4, which is described later, is for situations in which M is located along a cs-curve.) Before explaining our rules formally, we shall give the idea behind them and highlight their differences with the following example. Consider a launcher that fires a missile at a target. Somehow it is known that the missile misses its target. We will assume that the missile has abundant fuel and is not prone to malfunctioning. The first three of our rules consider the various possible reasons for the miss. Rule 1 would consider the case in which the launcher was transported to a new position before the missile was fired. The corresponding c-space in this case would look similar to that of Figure 6(a). Rule 2 would consider the case in which the missile was initially fired on-target but some unpredicted disturbance, such as strong wind, deflected the missile from its target. Figure 6(b) shows a typical c-space for this case. Finally, Rule 3 would consider the case in which the missile was destroyed in the air by another missile. Figure 6(c) shows a typical c-space.

We now formally present the first three rules.

Rule 1. Given processes M, E, and C, if

1. M is in free space,
2. M and C are dynamic processes,
3. M occurs immediately before C,
4. E occurs during or immediately after C, and
5. C in the modified simulation begins at the point at which M begins in the nominal simulation,

then M disables E.

This rule considers the case in which M occurs immediately before C and thus shifts C to a new location so that E does not occur. There are several characteristics that distinguish this situation. First, both M and C must be dynamic processes. Otherwise, M cannot displace C and C will not lead to new causal processes. Second, M occurs immediately before C in order to displace it to a new starting position. Third, in the modified simulation, E occurs during or immediately after, but not before, C. Fourth, in the modified simulation, C 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 7. 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, we conclude that M1 disables E. Note that although Figure 7 depicts the case in which E occurs *after* C, the same principles would apply if E occurred *during* C.

Rule 2. Given processes M, E, and C, if

1. M is in free space,
2. M and C are dynamic processes,
3. M occurs during C,
4. E happens during or immediately after C,
5. C in the modified simulation passes undistorted through the point where M begins in the nominal simulation, and
6. E occurs downstream of that point,

then M disables E.

In the previous rule, M shifts C to a new location so that C can no longer produce E. This rule, on the other hand, considers the situation in which the C process is heading in the right direction for E to occur, but M intervenes and deflects it so that E is prevented. The distinguishing characteristics of this case are as follows. First, M and C must be dynamic processes, otherwise M cannot deflect C and there would be no motion leading to new causal processes. Second, M occurs simultaneously with C in the nominal simulation and diverts it from its original path. Third, in the modified simulation E occurs during or after C, but in the nominal simulation E does not occur. Fourth, in the nominal simulation C passes through the point at which M begins in the nominal simulation. Fifth, E occurs downstream of the point where M begins in the nominal simulation. This ensures that E occurs spatially after M. If all of the characteristics are present, then there is good evidence that M disables E.

To illustrate this rule, consider the cs-planes shown in Figure 8. Two M processes (M1 and M2) occur in the nominal simulation, whereas one E process 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 uninterrupted and ultimately leads to E. Furthermore, in the modified simulation C passes straight through the point where M1 begins in the nominal simulation. This suggests that 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 3. Given processes M, E, and C, if

1. M is in free space,
2. C is a dynamic process,
3. M occurs immediately after C,

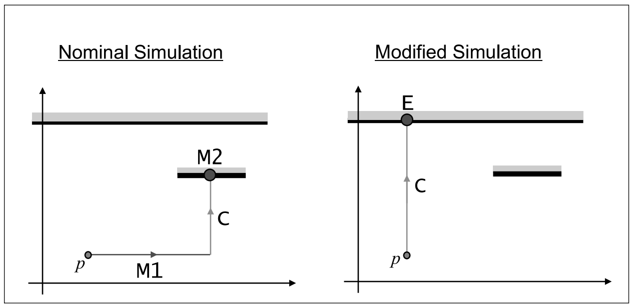

Fig. 7. The cs-planes used in the discussion of rule 1.

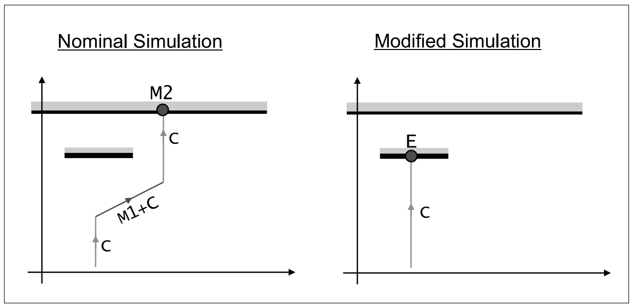

Fig. 8. The cs-planes used in the discussion of rule 2.

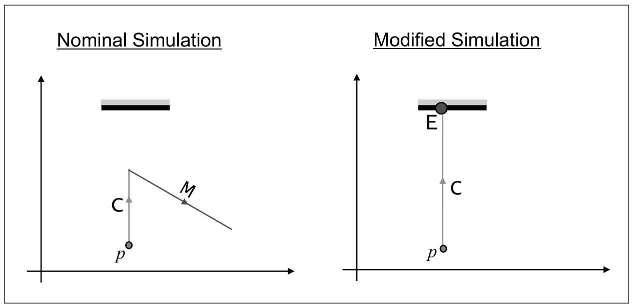

Fig. 9. The cs-planes used in the discussion of rule 3.

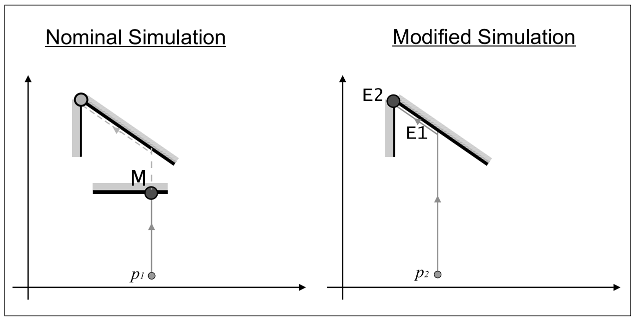

Fig. 10. The cs-planes used in the discussion of rule 4.

4. E occurs immediately after C,
5. C in the modified simulation passes undistorted through the point where M begins in the nominal simulation, and
6. E occurs downstream of that point,

then M disables E.

This rule considers the case where M and E both happen after C. In such cases, M disables E by terminating C, which would otherwise produce E. To determine this, we once again examine the spatial ordering between M and E. The requirement is still the same: for M to disable E, it has to take place prior to E. Although rules 2 and 3 use the same reasoning to examine the link between M and E, they differ in the way M accomplishes its task. In rule 2, M disables E by temporarily deflecting C, whereas in rule 3, M disables E by terminating C. Note that in this case M does not need to be a dynamic process.

Figure 9 shows an example of M disabling E by terminating C. In the nominal simulation, M suddenly brings C to a stop and produces a new trajectory that does not lead to E. In the modified simulation, because C is uninterrupted, it results in E. Because M stops the trajectory before E can happen, we conclude that M disables E.

The analyses so far have considered configurations in which the M process occurs in free space. The next rule considers cases in which M occurs along a cs-curve.

3.2.2. M Along a cs-curve

Rule 4. Given two processes M and E, if removing process M from the nominal simulation and removing 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 E process from occurring. This can happen if the cs-curve either stops[3] or redirects a trajectory that would otherwise have led 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 10. 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 might 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 involving 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

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

block the trajectory, such as the upside down V in Figure 10. 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 problems we have examined, this rule did produce the correct results.

3.2.3. *Computing indirect causal relationships*

The above rules consider direct causal relationships between pairs of missing and extra processes. Sometimes there are additional indirect relationships between pairs of causal processes. 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 1–4 to detect direct disablement. We then examine the identified direct causal relationships to find any indirect relationships.

Rule 5. Given processes M1, M2, and E, if

1. M1 and M2 both disable E and
2. M1 occurs in the history list of M2,

then M1 indirectly disables E by producing M2.

This rule examines the situation in which missing process M1 indirectly disables an E process through the assistance of missing process M2. The issue here is that rules 1–4 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 to the occurrence of a particular causal process.[4] 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 1–4 would indicate 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 7. Rule 1 determines that M1 disables E, as explained previously. Applying rule 4 on the same cs-plane, we determine 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 rule 5, we can conclude that M1 indirectly disables E by causing M2. An informal examination of the trajectories in Figure 7 reveals 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 occurred. 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.

[4]See Stahovich and Kara (2001) for a complete discussion of history lists.

Rule 6. Given processes M1, M2, and E, if

1. M1 disables E and
2. E disables M2,

then M1 indirectly enables M2 by disabling E.

Our final rule examines indirect enablement. If missing process M1 disables an E process 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 8 provide an example of this case. Using rule 2, we previously determined that M1 disables E. By applying rule 4 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 particular case, M1 is also a direct cause of M2 because M1 is in the history list of M2.)

Returning to our four-block example, we can now determine how the previously identified missing and extra processes are related to one another. Figure 11 shows the final result. (Fig. 4 shows how the missing and extra processes map to c-space.) This result is obtained as follows.

From rule 2 we determine that missing process M1 disables extra process E1. The rule is applied in the cs-planes of blocks A and C shown at the top of Figure 4. Applying rule 4 to the same cs-planes, we also determine that process E1 disables M2: When the short, horizontal cs-curve is removed in the modified simulation, the envisionment shows that the trajectory will reach the A-stop cs-curve.[5] Because M1 disables E1 and E1 disables M2, it follows from rule 6 that M1 *indirectly enables* M2.

Now, considering the cs-planes of A and D, we determine from rule 4 that M3 disables E2: When the short, vertical cs-curve is removed, the envisionment shows that the tra-

[5]During the envisionment process of rule 4, we allow the extended trajectory to penetrate through a cs-curve if the trajectory approaches the curve from the blocked space side. In such cases, the curve does not dictate the trajectory's behavior. Therefore, when the trajectory preceding E1 is extended, it penetrates through both short, horizontal cs-curves and reaches A-stop.

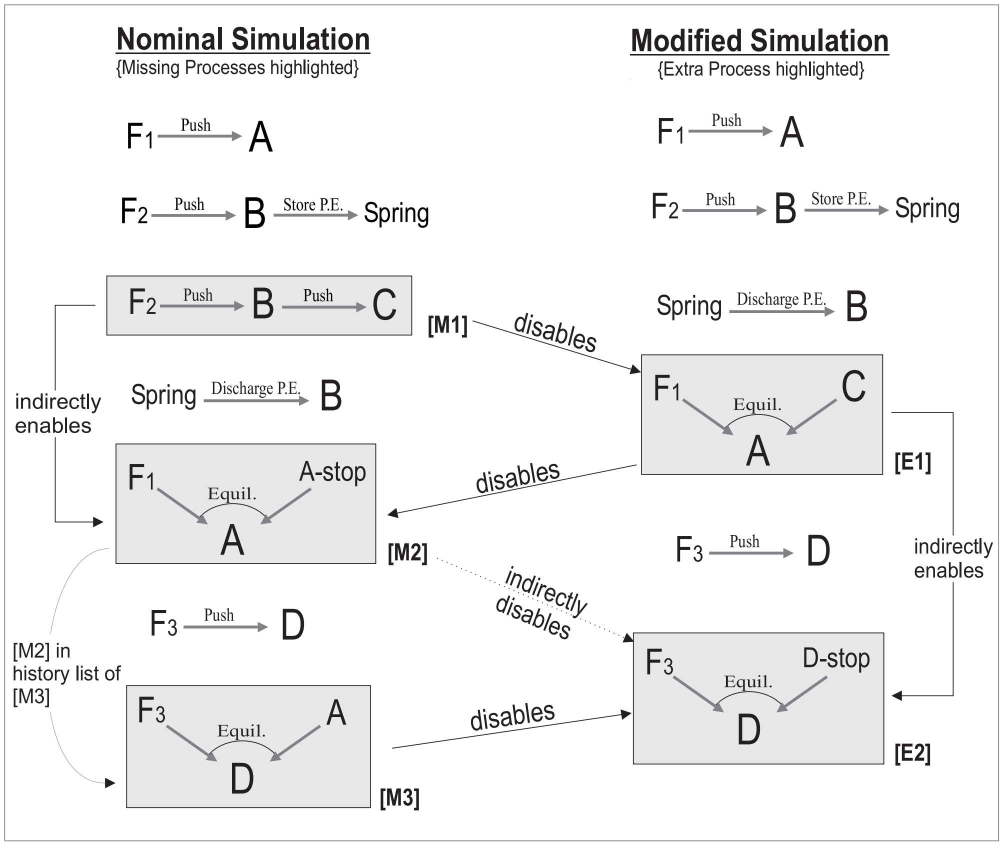

Fig. 11. The causal relationships between the missing and extra processes of the four-block example.

jectory will reach the D-stop cs-curve. With a second application of rule 4, we similarly conclude that M2 disables E2. Note that this second application is really the same as the first because M2 and M3 occur simultaneously. In both cases, the short, vertical cs-curve is removed and the envisionment leads to D-stop.

Next, note that both M2 and M3 disable the same extra process, E2. Additionally, the history list of M3 contains M2. (This can be observed by noting that M2 is on the trajectory leading to M3.) Therefore, from rule 5 we deduce that M2 *indirectly disables* E2 by causing M3. Finally, having determined that E1 disables M2 and that M2 (indirectly) disables E2, we conclude from rule 6 that E1 *indirectly enables* E2 by preventing M2.

3.3. Program implementation and computational complexity

We implemented a computer program that performs the causal analysis approach described above. The program begins by assembling the cs-planes of the nominal and modified simulations using information supplied in a file. This file encodes the names of the bodies considered in the cs-planes and the locations of the cs-curves, the list of missing, extra, and common causal processes, the motion data consisting of a sequence of monotonic trajectories, and the list of causal processes that reside on each trajectory. For a device composed of *b* bodies, trajectories are represented as *b*-dimensional displacement vectors. The program trivially computes the 2-D projections of the trajectories onto the cs-planes. For the examples we considered, we generated the simulations and data files manually. Note that we previously developed a software system capable of identifying the missing, extra, and common processes. That system is described in Stahovich and Kara (2001). We are currently working on integrating that software with the software described here and a commercial dynamic simulation tool.

The cs-planes we consider describe the configurations of only two bodies at a time. Although this simplifies the subsequent geometric analysis, the number of cs-planes to consider in this way can be prohibitive in the worst case. For a device composed of *b* bodies, the number of 2-D cs-planes is given by the pairwise combinations of the bodies, $C(b, 2)$, which results in $O(b^2)$ complexity.[6] However, due to the nature of problems encountered in the domain, this worst case scenario is very unlikely. Such cases arise only if each

[6] $C(b, 2) = \dfrac{b(b-1)}{2} = O(b^2)$.

body interacts with all the other bodies in the system. For real world devices, however, the parts that make up the system are almost never fully connected (except when the number of components is small). This is because dense connectivity inhibits mechanisms from performing useful tasks and additionally hinders manufacturability and reliability. To get a sense of the complexity experienced in practice, we examined 50 purely mechanical devices from a catalogue (Chironis, 1991). For this sample set, the mechanisms contained an average of 7.3 components (standard deviation = 2.45), and each component was connected to only 2.3 other parts (standard deviation = 0.55). This kind of sparse connectedness in mechanical systems greatly reduces the number of cs-planes to be considered and the analysis that needs to be performed.

After projecting the simulation data onto the cs-planes, our program applies the causal analysis rules. The program considers all possible pairs consisting of one missing and one extra process. For each such pair, the program applies rules 1 through 4 to determine if the missing process disables the extra process. However, before the full geometric analysis is performed, each process pair is first tested to see if the temporal precedence principle rules out any causal relationships (i.e., the cases marked $\times$ in Table 1). As explained previously, rules 1–4 work from the projections of the causal-processes onto the cs-planes. Because the projections often exhibit different geometric characteristics in different cs-planes, it is necessary to analyze each missing/extra process pair in all of the individual cs-planes. Rules 1–4 are therefore applied to each process pair multiple times, once in each cs-plane. If none of the rules detects a causal relationship for a particular pair, our program concludes that the missing process does not disable the extra process. The entire process is then repeated using the converse of the rules to determine if any extra processes disable any missing processes.

Because the above step considers each process pair on every cs-plane, the computational complexity is a function of both the number of causal processes and the number of cs-planes. Denoting the number of missing and extra processes by m and e, respectively, this step therefore results in $O(m \cdot e \cdot b^2)$ complexity. [The number of cs-planes is $O(b^2)$ as explained previously.] This computation can be expensive if m and e grow too large, which would happen if the device is so complex that a modification produces a long chain of new behaviors. However, this difficulty can be partially remedied by noting that the computed relationships are causal and causality is *directional*: if A causes B, then B cannot cause A.[7] Using this fact, some unnecessary tests can be avoided. Consider once again Figure 11. From rule 2 it was determined that M1 disabled E1. Once this relationship is identified, the possibility of the converse, that is, E1 disabling M1, need not be considered. In addition, if M1 had been produced by another missing process, say M0, that happened earlier, then E1 cannot be disabling M0 because M0 would have been the process that caused M1 in the first place. Our current implementation is naive, however, in that it does not take advantage of such previously computed causal relationships to guide its exploration. We implemented our system this way mainly to monitor if it produced erroneous results, such as determining cyclic causal relationships when there are none.

After rules 1–4 and their converses have been considered, rules 5 and 6 are applied to identify any *indirect* enablements and disablements. As described earlier, these two rules work from the causal relationships identified by the first four rules and are therefore applied only after those rules have been considered. If the first four rules do not identify any causal relationships, then neither will rules 5 and 6.

These last two rules do not involve any cs-plane computation. Hence, their complexity is small compared to the previous two steps. Despite their simplicity, our studies have shown that these two rules provide a more natural explanation of purpose as they interlace short, discrete causal links into single, coherent causal chain. A user study reported in Oltmans (2000) indicates that when describing causal flows in mechanical devices, humans tend to compose moderately long sentences by combining shorter causal relationships. Rules 5 and 6 are set to accomplish a similar task by shifting the focus from short, constituent causal paths to higher level causal influences. An additional feature of these rules is that instead of merely highlighting longer causal paths within the same simulation, they may elucidate how one process produces another by circumventing one that was never observed! For example, in the four-block example (Fig. 2), our program is able to conclude that B pushing C allows A to reach A-stop by preventing a collision that normally does not happen.

The following is the unedited output produced by our program for the four-block example:

```
   From Rule-2: The missing process M1 disables the extra process E1. Reason: In the nominal simulation, M1 happening simultaneously with C1[8] prevents C1 from leading to E1. This disablement was identified from the cs-plane of CA.
   From Rule-4: The extra process E1 disables the missing process M2. Reason: In the modified simulation, missing process M2 could have happened if E1 did not happen. This disablement was envisioned from the cs-plane of CA.
   From Rule-4: The missing process M3 disables the extra process E2. Reason: In the nominal simulation, extra process E2 could
```

[7] We assume that components do not exhibit cyclic causality.

[8] C1 is the dynamic process of F1 pushing A.

```
have happened if M3 did not happen. This
disablement was envisioned from the cs-
plane of DA.
  From Rule-4: The missing process M2 dis-
ables the extra process E2. Reason: In the
nominal simulation, extra process E2 could
have happened if M2 did not happen. This
disablement was envisioned from the cs-
plane of DA.
  From Rule-5: The missing process M2 in-
directly disables the extra process E2.
Reason: In the nominal simulation, missing
process M2 indirectly disables extra pro-
cess E2 by producing M3, which prevents E2
from occurring.
  From Rule-6: The missing process M1 in-
directly enables M2. Reason: In the nomi-
nal simulation, M1 enables M2 by preventing
the extra process E1 from happening. If E1
happens M2 cannot happen.
  From Rule-6: The extra process E1 indi-
rectly enables E2. Reason: In the modified
simulation, E1 enables E2 by preventing
the missing process M3 from happening. If M3
happens E2 cannot happen.
```

With postprocessing and appropriate text templates, these facts could be condensed into "The feature, enables A to collide with A-stop by preventing A from colliding with C. Moreover, because A reaches A-stop, D collides with A rather than D-stop." Clearly, the analysis has captured the essence of the purpose.

Compared to the first four rules, the output produced by rules 5 and 6 provides better insight into how the feature accomplishes its purpose. Note that each application of rule 5 or 6 introduces a new process to the path of indirect causal relationships. Therefore, multiple applications of these rules would generate progressively longer paths until the entire causal path is identified. Although this approach is computationally feasible, our program applies rules 5 and 6 only once, which results in paths that are at most three processes long. We chose to keep the path length to this size because as the causal paths become longer, the output produced by our program becomes too complicated. To generate explanations that are natural to humans in these cases, we would need additional ways to condense and summarize the raw output. Currently, our text generation procedures do not incorporate such techniques.

3.4. Completeness and scope

The geometric rules described above attempt to identify the causal connections between the missing and extra processes by investigating their traces on the cs-planes. Our studies have shown that the way in which M disables E markedly differs, depending on whether M occurs in free space or along a cs-curve. We therefore handle these two categories separately. The first three rules consider the case where M is in free space; individual rules differ according to whether M occurs before, during, or after C. The fourth rule considers the remaining cases where M occurs along a cs-curve.

Currently, our rule set has a number of limitations. In a given cs-plane, processes M and E can occur in a wide variety of different relative configurations while still being causally related. In this context, completeness can be guaranteed only if the analysis encompasses all such configurations regardless of how far apart the processes occur in the cs-planes. The first three rules do not meet this requirement because they are limited to configurations in which M and E take place only immediately before, during, or immediately after C. They are therefore applicable to processes that are strictly proximate. Within this restricted scope, we consider our rules to be *nearly* complete because they cover all nine cases shown in Table 1, except when both M and E occur before C (the case indicated by ND).

Further analysis has shown that our rule set can be extended to cover a broader class of configurations, including those in which M and E are separated by multiple intermediate processes. Consider the cs-planes shown in Figure 12. In the nominal simulation, the missing process M1 is followed by the common processes C1 and C2, which finally produce M2. In the modified simulation, C1 and C2 lead to the E process because of the absence of M1. A comparison of the two cs-planes suggests that M1 disables E by shifting the common processes in the nominal simulation. This closely resembles the cases considered in rule 1, except now M1 and E are separated by two common processes instead of one. In its current form, rule 1 will miss this disablement because it would require E to occur immediately after C1. However, the idea of M shifting the trajectories to disable E remains the same. A natural extension of rule 1 would therefore be to group consecutive common processes and treat them as one. This would allow our rules to be applicable to cases where the missing and extra processes are separated by an arbitrary number of common processes.

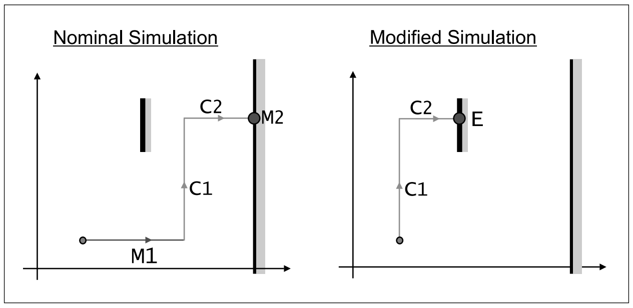

Fig. 12. Missing process M1 disables E by shifting the two common processes. Treating C1 and C2 as a single common process would allow rule 1 to detect the disablement.

A similar situation is shown in Figure 13. In the nominal simulation, M diverts the trajectory that produces E1 and E2 in the modified simulation. There is therefore a clear indication that M disables both E1 and E2. Rule 2 would determine that M disables E1. However, M disabling E2 would go undetected because E2 is not in the immediate proximity of C. Such difficulties can be resolved by examining the processes' history lists, which usually contain important cues for bridging the gap between distant processes. In this approach, if an M process is known to disable an E process, any subsequent extra process that contains E in its history list would also be considered as a process that M disables. The reason is that processes in the history lists constitute the root causes; and if the root cause is disabled, then so are all of its consequences. In the above example, because E1 resides on the path leading to E2, E1 is a root cause of E2 and therefore will appear in E2's history list. Moreover, M is known to disable E1. Using these facts, we can conclude that M also disables E2. Because the main idea is to exploit history lists to detect disablement, the same principles are also applicable to rules 1 and 3. Unlike the current approach, however, this approach would provide broader coverage by allowing the rules to identify disablement between processes that are far apart.

Rules 1, 2, and 3 assume that a C process exists between the two simulations. Although this may seem too restrictive, it is particularly reasonable in our case. The geometric modifications we consider are highly localized and therefore several device properties, such as the general layout of the components and the externally applied forces, remain mostly unchanged. As a result, the device is likely to exhibit a number of similarities in the two simulations, giving rise to common processes. Nevertheless, Stahovich and Raghavan (2000) showed that if the modification produces a set of entirely new behaviors, then the purpose of the feature is most likely to be found in the "first difference" between the simulations; the subsequent differences are simply a consequence of this root difference.

Our causal-process representation incorporates a number of simplifying assumptions about the device mechanics. First,

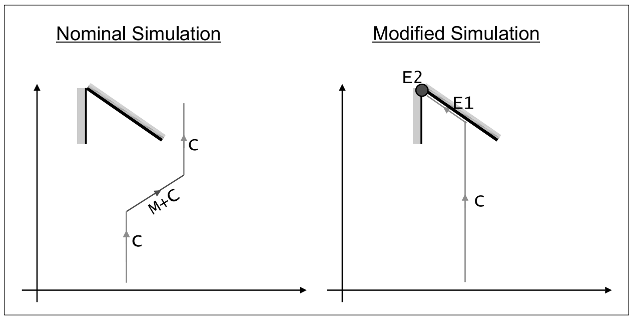

Fig. 13. Missing process M disables both E1 and E2 by deflecting C. In its current form, rule 2 would miss the fact that M disables E2. The history list of E2 helps resolve this difficulty. Because M disables E1 and E1 is in E2's history list, we can conclude that M also disables E2.

inertial effects are assumed to be negligible. Work by Sacks and Joskowicz (1993) indicates that this assumption is valid for a wide range of mechanisms. This fact simplifies the task of determining the causes of a body's motion. First, if inertia is negligible, a body's motion is caused by those forces having a component in the direction of motion. Therefore, only external forces, contact forces, and elastic (spring) forces have to be considered as possible sources of motion. Second, it is assumed that springs are overdamped and thus do not exhibit vibratory behavior after completing one cycle of oscillation. Third, collisions between bodies are considered to be inelastic. In the presence of elastic collisions, the mappings of causal processes along the cs-curves would consist of a series of rippled curve segments as opposed to straight lines (Stahovich et al., 2000). Although not implemented, it may be possible to smooth out such ripples by preprocessing the simulation data before performing the geometric analyses presented here. This might be accomplished by detecting and combining repetitive line segments that are spatially short compared to other trajectories in the cs-plane. Joskowicz (1990) describes a number of simplification and abstraction operators that might be particularly useful for this task. A similar approach could also be taken to account for the vibratory behaviors of springs, provided that such behaviors are not critical in the operation of the device. This would condense the set of causal processes for a spring's compression and relaxation cycles into one representative process that abstracts away oscillations.

In addition to the simplified treatment of mechanics, we make use of certain simplifications about the mappings of causal processes onto the cs-planes. Whereas real behaviors can give rise to trajectories of arbitrary shape, the ones we consider are composed exclusively of straight lines. Although the straight-line approximation has been adequate for our purposes, it may be too restrictive for devices that exhibit substantial accelerations. In the presence of accelerations, the simulation traces will give rise to curved trajectories. In such cases, the processes predicted by the envisionment of rule 4 become questionable due to the imprecision associated with extending and manipulating curved segments. One way to perform envisionment in such cases would be to fit a low order curve to the actual trajectory and assume that the trajectory would continue its motion in the direction predicted by the curve. Because we ignore inertial effects, however, we were not concerned with curved trajectories and restricted our analysis to straight lines only.

So far, we have tested our techniques using fixed-axis mechanisms only. A fixed-axis mechanism is composed of parts that can either translate or rotate about axes that are fixed in space. We have focused on fixed-axis mechanisms because the resulting c-spaces are easily decomposable into 2-D cs-planes. For more general mechanisms, the c-space of a pair of bodies may contain complicated surfaces separating free and blocked spaces. In such cases, because there are no means to decompose the curves into orthogonal 2-D

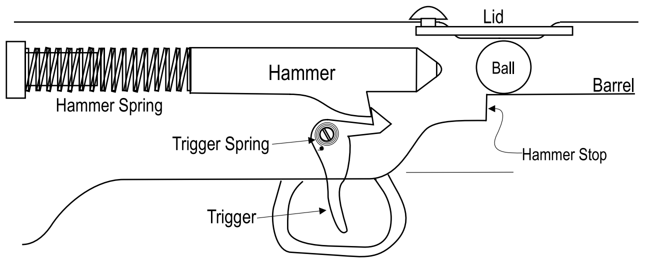

Fig. 14. A schematic of a firearm mechanism. In the configuration depicted, the device is loaded and ready to fire. Pressing the trigger releases the hammer, which then strikes and launches the rubber ball. Similar mechanisms are used in a wide variety of devices ranging from paintball guns (where the hammer is replaced by pressurized air) to stamping–forging machines (where the projectile is replaced by the work piece).

projections, the geometric reasoning about the causal processes would have to be carried out in the multidimensional c-space instead of the simpler 2-D cs-planes. Currently our techniques are restricted to 2-D trajectories and are therefore suitable for fixed-axis mechanisms only.

Mechanisms that involve part play and frequent contact changes often give rise to complex configuration spaces, even when the number of components is small. For instance, Joskowicz et al. (1998) describe a c-space that models the effect of manufacturing tolerances on the functioning of a Geneva mechanism. In such devices, slight variations in the geometry can cause substantial differences in the resulting behaviors. The devices we consider are less sensitive to such subtle geometric variations. This naturally allows our techniques to be more tolerant of the inaccuracies in the configuration space models. Conversely, in the presence of subtle geometric effects, our program will likely fail to identify the causal links between the feature and the observed behaviors. However, in certain situations, this can be used to the advantage of the designer: if our program cannot identify any causal relationship, perhaps there are more subtle effects. Our program would have performed a useful task by drawing the designer's attention to the need to document these subtleties.

4. EXAMPLE: FIREARM MECHANISM

In this section we demonstrate our approach on a practical example. Consider the firearm mechanism shown in Figure 14. In this device, pressing the trigger allows the cocked hammer to strike and fire rubber balls. The feature of interest is the triangular protrusion on the top right corner of the trigger.

Figure 15(a) shows the normal operation of the device. Initially, the weapon is unloaded and all components are at rest. The operation begins when the user starts compressing the hammer by applying force F1. As the hammer is being cocked, it collides and slightly deflects the trigger. After the hammer is sufficiently cocked (as evidenced by the clicking sound of the trigger), the user releases the hammer and the hammer starts moving to the right. On return, the trigger catches and locks the hammer in place. Next, the user inserts the rubber ball into the weapon through an opening on the top of the barrel (F2, pushing the ball downward). After the ball is properly loaded, the user covers the opening on the barrel by sliding the lid over the ball (F3, pushing the lid to the right). In this configuration, the mechanism is completely loaded and ready to fire. Finally, pressing the trigger releases the hammer, which then strikes and fires the ball.

Figure 15(b) shows the operation of the device with the triangular piece on the trigger removed. When the hammer is compressed and released, the trigger can no longer restrain the hammer. As a result, the hammer returns to its initial position, where it blocks the ball's path. Next, when the user attempts to insert the ball, it runs into the hammer instead of reaching the bottom of the barrel. This in turn causes the lid to jam because it cannot be closed completely. Finally, because the weapon is not charged, pressing the trigger does not fire it.

As shown, the absence of the feature has a number of serious effects on the rest of the system and in the end prevents the device from performing its main task of firing the ball. From the differences between the two simulations, we can conclude that the feature enables the trigger to restrain the hammer so the ball can be properly loaded and the lid can be fully closed. This ultimately enables the hammer to fire the ball after the user depresses the trigger.

Figure 16 shows the causal processes from the nominal and modified operations of the device. Once again the missing and extra processes are highlighted. There are a total of six missing processes and three extra processes. For easier visualization, the processes are graphically positioned so that similar processes in the two simulations are listed side by side.

To simplify the analysis, the process of the hammer striking and firing the ball in the nominal simulation is reduced to an equilibrium process (the missing process M6). In reality, this interaction should be represented as a dynamic process in which the hammer is pushing the ball to the right. However, this would require allocating a new horizontal degree of freedom for the ball in addition to its original vertical degree of freedom. The new degree of freedom would increase the number of cs-planes to consider and further complicate the analysis without providing any significant benefit. (The hammer pushing the ball would not lead to any future process.) We therefore introduced the simplification, although the more comprehensive analysis is still technically feasible.

Figure 17 shows the traces of the nominal and modified simulations mapped onto the cs-planes. The displacements are taken to be positive toward the right for the hammer and the lid, upward for the ball, and clockwise for the trigger. The effect of the triangular feature can be observed in the cs-planes of the trigger and the hammer [Fig. 17(a), top]. When the feature is present, the two components interact

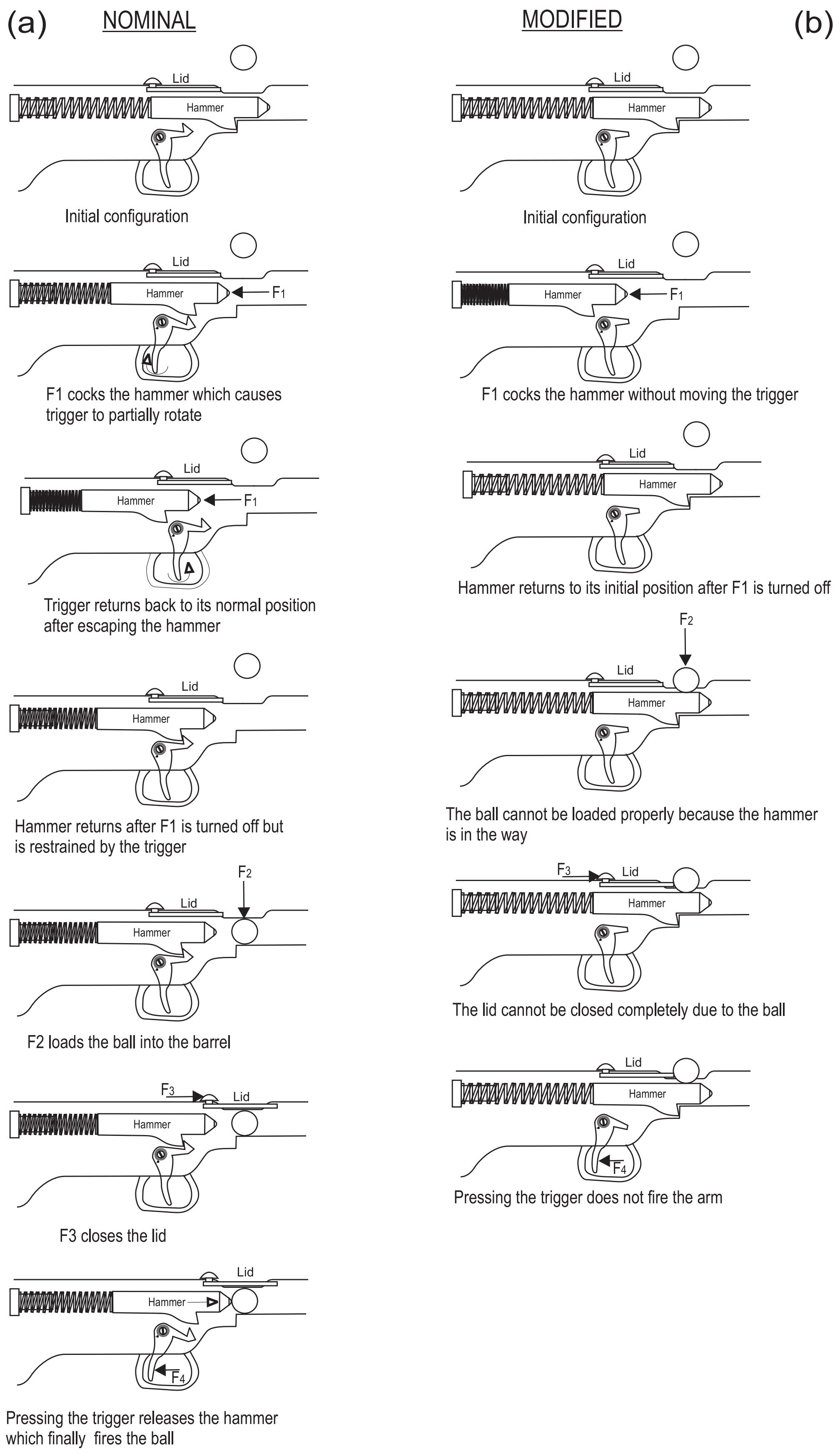
(a)
NOMINAL
Lid
Hammer
Initial configuration
F1
F1 cocks the hammer which causes trigger to partially rotate
Trigger returns back to its normal position after escaping the hammer
Hammer returns after F1 is turned off but is restrained by the trigger
F2
F2 loads the ball into the barrel
F3
F3 closes the lid
F4
Pressing the trigger releases the hammer which finally fires the ball
MODIFIED
(b)
Initial configuration
F1 cocks the hammer without moving the trigger
Hammer returns to its initial position after F1 is turned off
The ball cannot be loaded properly because the hammer is in the way
The lid cannot be closed completely due to the ball
Pressing the trigger does not fire the arm

Fig. 15. The nominal and modified operations of the firearm mechanism.

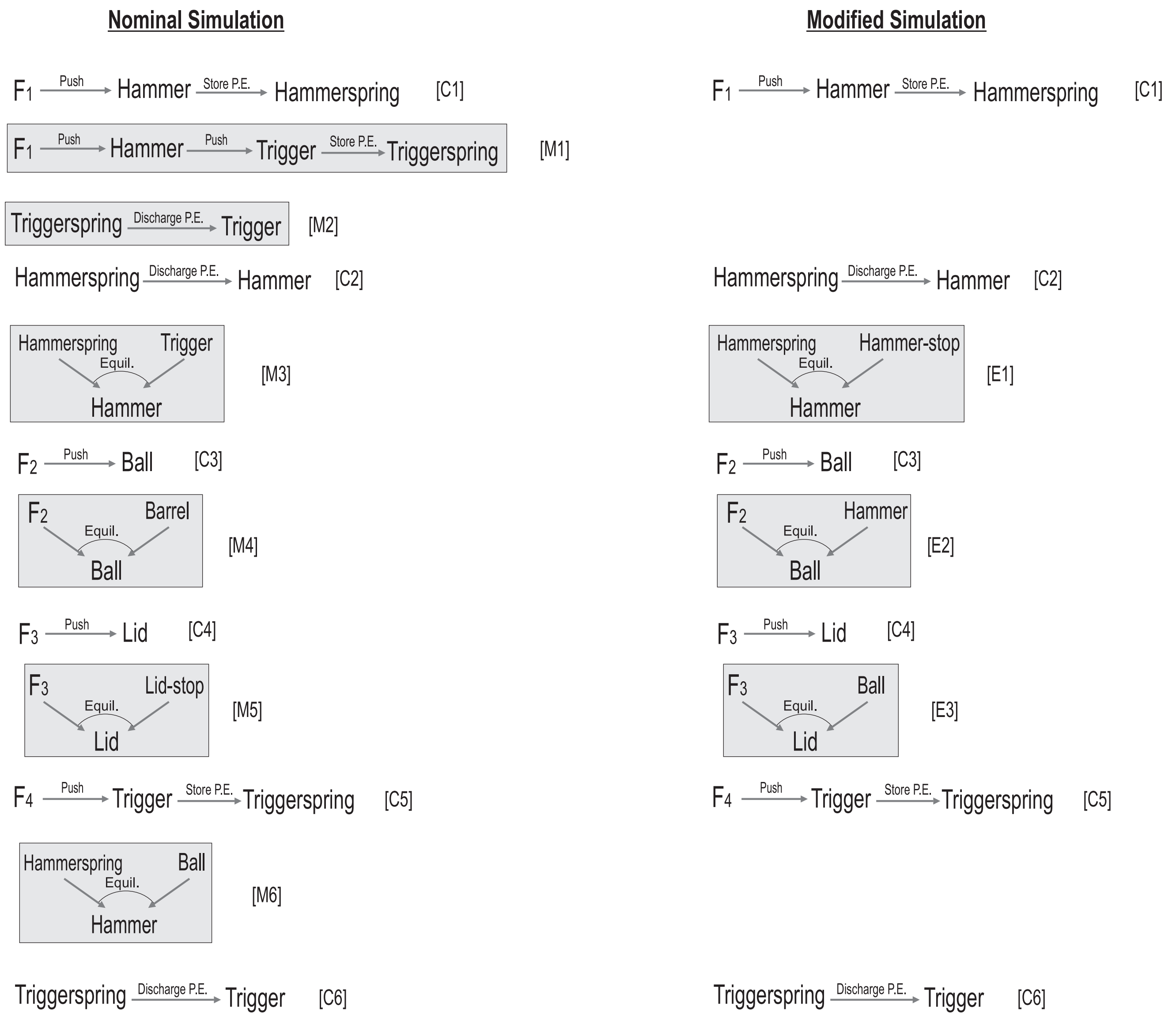
Nominal Simulation
Modified Simulation
F1 Push Hammer Store P.E. Hammerspring [C1]
F1 Push Hammer Push Trigger Store P.E. Triggerspring [M1]
Triggerspring Discharge P.E. Trigger [M2]
Hammerspring Discharge P.E. Hammer [C2]
Hammerspring Trigger Equil. Hammer [M3]
Hammerspring Hammer-stop Equil. Hammer [E1]
F2 Push Ball [C3]
F2 Barrel Equil. Ball [M4]
F2 Hammer Equil. Ball [E2]
F3 Push Lid [C4]
F3 Lid-stop Equil. Lid [M5]
F3 Ball Equil. Lid [E3]
F4 Push Trigger Store P.E. Triggerspring [C5]
Hammerspring Ball Equil. Hammer [M6]
Triggerspring Discharge P.E. Trigger [C6]

Fig. 16. The causal processes of the nominal and modified simulations of the firearm mechanism.

through the V-shaped cs-curves. On the other hand, removing the feature from the trigger precludes its interaction with the hammer. As shown in the top of Figure 17(b), this results in the removal of the two cs-curves from the cs-plane of the trigger and the hammer.

The simulation traces are numbered so as to make the sequence of trajectories easier to follow. For example, trajectories 1, 2, and 3 in the cs-planes of the trigger–hammer in the nominal simulation [Fig. 17(a), top] correspond to the respective processes of the hammer initially moving to the left, the hammer displacing the trigger while being compressed, and the trigger escaping the hammer and returning while the hammer is still advancing to the left. The dash-separated numbers indicate extended static processes in which two bodies are in equilibrium while other parts are in motion. For example, the static process denoted by "6-10" on the cs-plane of trigger–hammer indicates that both the trigger and the hammer are in equilibrium while processes 6–10 are being traced in the remaining cs-planes of ball–hammer and ball–lid. Note that this number scheme is only meant to help the reader follow the simulation sequence more easily and has no connection to the list of processes shown in Figure 16.

The following is the unedited output produced by our program. The causal processes of interest, namely the missing and extra processes, are depicted in the cs-planes of Figure 17.

```
  From Rule:4 The missing process [M3] dis-
ables the extra process [E1]. Reason: In the
nominal simulation, extra process [E1] could
have happened if [M3] did not happen. This
```

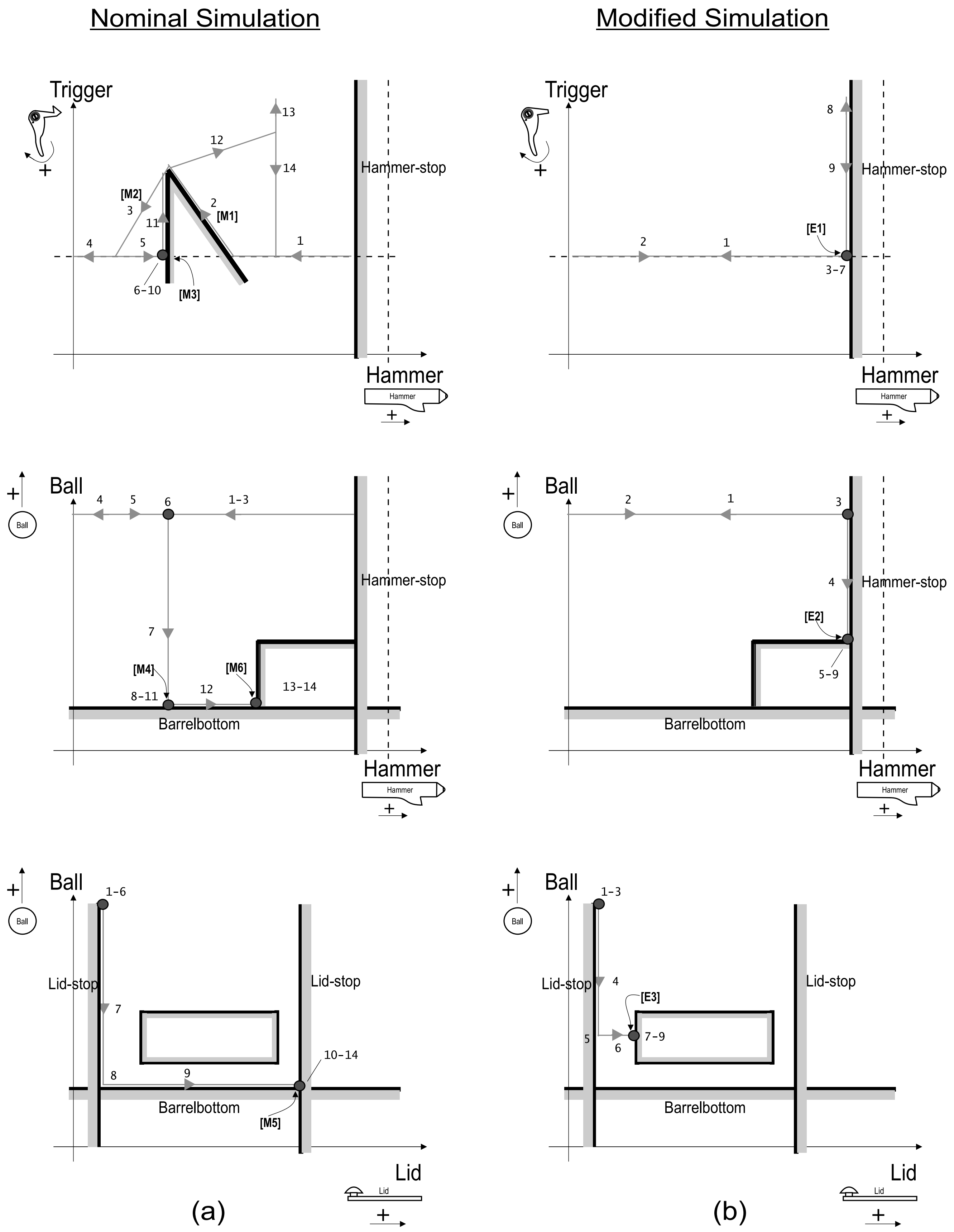

Fig. 17. The c-spaces of the (a) nominal and (b) modified simulations of the firearm mechanism.

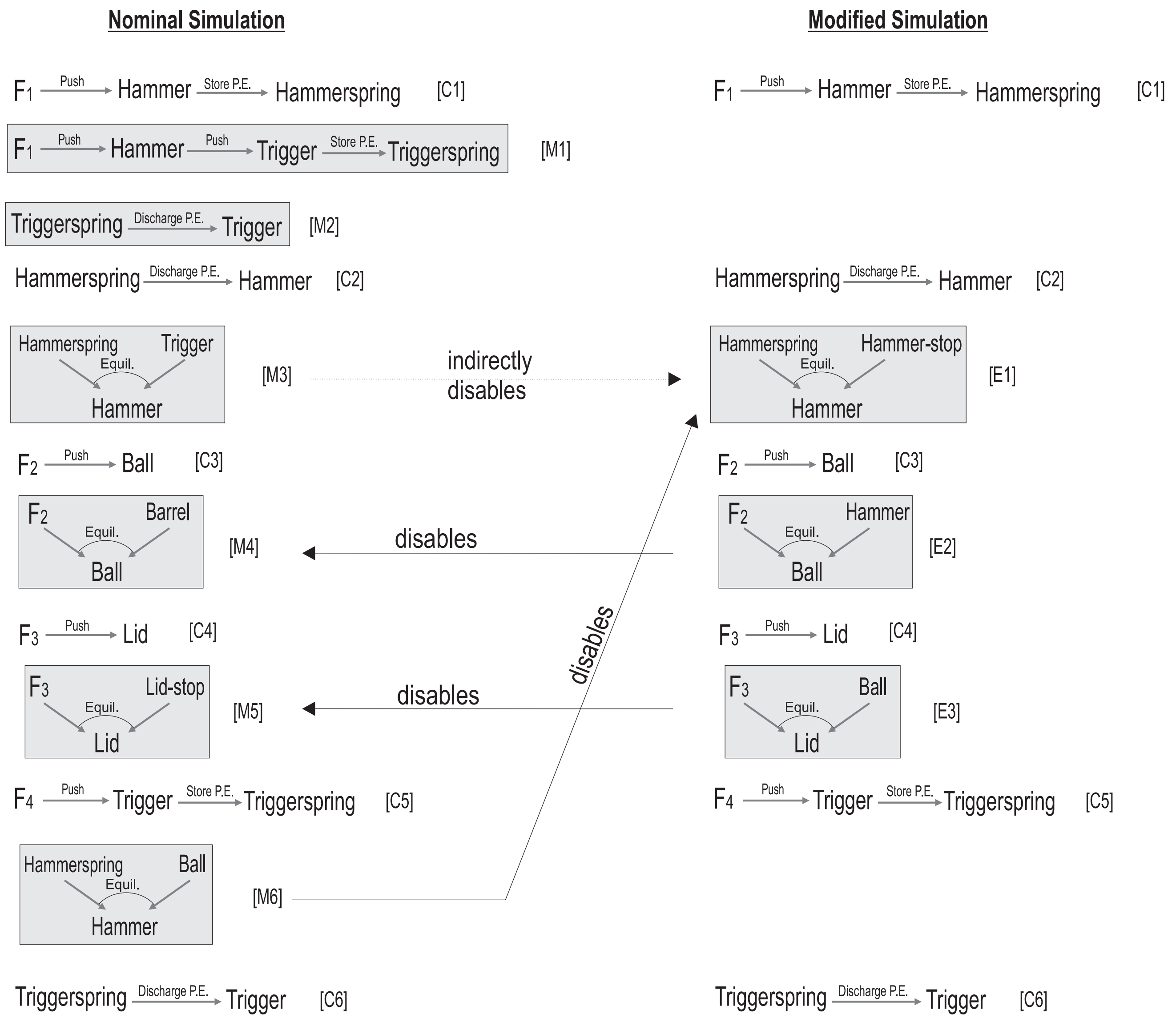
Nominal Simulation
Modified Simulation
F1 Push Hammer Store P.E. Hammerspring [C1]
F1 Push Hammer Push Trigger Store P.E. Triggerspring [M1]
Triggerspring Discharge P.E. Trigger [M2]
Hammerspring Discharge P.E. Hammer [C2]
Hammerspring Trigger Equil. Hammer [M3]
indirectly disables
Hammerspring Hammer-stop Equil. Hammer [E1]
F2 Push Ball [C3]
F2 Barrel Equil. Ball [M4]
disables
F2 Hammer Equil. Ball [E2]
F3 Push Lid [C4]
F3 Lid-stop Equil. Lid [M5]
disables
F3 Ball Equil. Lid [E3]
disables
F4 Push Trigger Store P.E. Triggerspring [C5]
Hammerspring Ball Equil. Hammer [M6]
Triggerspring Discharge P.E. Trigger [C6]

Fig. 18. The causal relationships between the missing and extra processes of the firearm mechanism.

```
disablement was envisioned from the cs-plane
of HT
```
[9]

```
  From Rule:4 The extra process [E2] dis-
ables the missing process [M4]. Reason: In
the modified simulation, missing process
[M4] could have happened if [E2] did not
happen. This disablement was envisioned from
the cs-plane of HB
  From Rule:4 The extra process [E3] dis-
ables the missing process [M5]. Reason:
In the modified simulation, missing pro-
cess [E5] could have happened if [E3] did
not happen. This disablement was envisioned
from the cs-plane of LB
```

[9]H, Hammer; T, trigger; B, ball; L, lid.

```
  From Rule:4 The missing process [M6] dis-
ables the extra process [E1]. Reason: In
the nominal simulation, extra process [E1]
could have happened if [M6] did not hap-
pen. This disablement was envisioned from
the cs-plane of HB
  From Rule:5 The missing process [M3] in-
directly disables the extra process [E1].
Reason: In the nominal simulation, missing
process [M3] indirectly disables extra pro-
cess [E1] by producing [M6], which prevents
[E1] from occurring.
```

These causal relationships are also shown in Figure 18. The results capture the following facts: the feature helps the trigger stop the hammer, which prevents the hammer from reaching the hammer-stop (M3 disabling E1). In the absence of the feature, the ball would be stopped by the hammer and that would prevent the ball from being inserted into the barrel (E2 disabling M4). Moreover, the lid would get stuck at the ball instead of reaching the lid-stop (E3 disabling M5). Finally, when the feature is present, the hammer would strike the ball instead of reaching the hammer-stop (M6 disabling E1). From the first and the last inference it can be concluded that the trigger restraining the hammer indirectly prevents the hammer from reaching its stop by allowing the hammer to strike the ball (M3 indirectly disabling E1 by causing M6).

The results indicate that the purpose of the feature is to allow the trigger to hold the hammer so that the ball can be properly loaded and the lid can be closed. These in turn enable the hammer to strike the ball instead of merely returning to its original position at the hammer-stop.

5. RELATED WORK

ExplainIT II builds upon a previous system called ExplainIT (Stahovich & Raghavan, 2000). 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 the 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. Chung and Editors (1997) and Gruber et al. (1991) offer good overviews of 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.

Franke (1991) 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.

There has been some previous work in trying to "understand" the behaviors of mechanisms. Forbus et al. (1991) 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 and Joskowicz (1993) 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.

Shrobe (1993) describes a system that understands the functioning of linkages. From the numerical simulation of the linkage, he extracts a "mechanism graph" that shows how motion propagates from link to link through the joints of the mechanism. The mechanism graph is parsed into more structured forms, from which the driving and driven modules of the system are identified. The curves traversed by the coupling points between these modules are analyzed to obtain the qualitatively important features. (These are features of the traces, not geometric features on the parts.) The function of the mechanism is then determined by deriving causal relationships between the features. The primary goal of that study is to compute the overall function of the mechanism, whereas ours is to compute the purposes of individual components. In addition, because Shrobe's study (1993) focuses on linkages, it is not concerned with devices with time-varying contacts as considered here.

Stahovich et al. (2000) describe a system for computing qualitative rigid-body dynamic simulations. That system uses a qualitative version of Newton's laws that is similar to the techniques used here for tracking the flow of causality through a device.

6. 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 whether the feature causes 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 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. In addition, 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. However, rules 1, 2, and 3 assume that there is a process common to both simulations. Hence, those two rules will not be applicable in cases where the device's modification produces behaviors that do not exhibit any similarity to the behaviors of the original device.

Our current techniques assume fixed-axis parts, inertia-free motion, and inelastic collisions. Although Sacks and Joskowicz (1993) demonstrate that these assumptions are valid for a wide range of mechanical devices, we still need to explore their limitations in our domain.

This study is focused on reasoning about qualitative behaviors of mechanical devices where the geometry plays a crucial role in the device's behaviors. Our current results suggest that geometric reasoning is a powerful tool for generating causal explanations of mechanical behavior. Our work is clearly at an early stage. As we examine more subtle and realistic devices, we will likely encounter the need for additional rules. Based on our current experience, however, we expect that we will need only a handful.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under Award 9813259.

REFERENCES

Chironis, N.P. (1991). *Mechanisms and Mechanical Devices Sourcebook*. New York: McGraw–Hill.

Chung, P., & Editors, R.B.-A. (1997). Special issue: Representation and use of design rationale. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 11(2)*.

Forbus, K.D., Nielsen, P., & Faltings, B. (1991). Qualitative spatial reasoning: The clock project. *Artificial Intelligence 51(9)*.

Franke, D.W. (1991). Deriving and using descriptions of purpose. *IEEE Expert 6*, 41–47.

Gruber, T., Baudin, C., Boose, J., & Weber, J. (1991). *Design Rationale Capture as Knowledge Acquisition Trade-offs in the Design of Interactive Tools*. Technical Report KSL 91-47. Stanford, CA: Stanford University, Knowledge Systems Laboratory.

Joskowicz, L. (1990). Mechanism comparison and classification for design. *Research in Engineering Design 1(2)*, 149–166.

Joskowicz, L., Sacks, E., & Kumar, V. (1998). Selecting an effective task-specific contact analysis algorithm. *IEEE Workshop on New Directions in Contact Analysis and Simulation*. New York: IEEE Press.

Knuffer, T., & Ullman, D. (1990). The information requests of mechanical design engineers. *Design Studies 11*.

Oltmans, M. (2000). *Understanding naturally conveyed explanations of device behavior*. Master's Thesis, Cambridge, MA: Massachusetts Institute of Technology.

Sacks, E., & Joskowicz, L. (1993). Automated modeling and kinematic simulation of mechanisms. *Computer-Aided Design 25(2)*, 106–118.

Shrobe, H. (1993). Understanding linkages. *Proc. AAAI-93*, pp. 620–625.

Stahovich, T.F., Davis, R., & Shrobe, H. (1998). Generating multiple new designs from a sketch. *Artificial Intelligence 104(1–2)*, 211–264.

Stahovich, T.F., Davis, R., & Shrobe, H. (2000). Qualitative rigid body mechanics. *Artificial Intelligence 119(1–2)*, 19–60.

Stahovich, T.F., & Kara, L.B. (2001). A representation for comparing simulations and computing the purpose of geometric features. *AIEDAM 15(2)*, 189–201.

Stahovich, T.F., & Raghavan, A. (2000). Computing design rationales by interpreting simulations. *ASME Journal of Mechanical Design 122(1)*, 77–82.

Levent B. Kara is a doctoral student in the Mechanical Engineering Department at Carnegie Mellon University. He earned his BS in mechanical engineering from the Middle East Technical University, Ankara, Turkey, in 1998 and his MS in mechanical engineering from Carnegie Mellon University in 2000. His research interests include causal and spatial reasoning about mechanical systems, automatic design rationale identification, and sketch interpretation techniques to enable natural user interfaces in design software.

Thomas F. Stahovich is an Associate Professor in the Mechanical Engineering Department at Carnegie Mellon University, where he is the Director of the Smart Tools Lab. He received a BS in mechanical engineering from the University of California at Berkeley in 1988 and SM and PhD degrees in mechanical engineering from the Massachusetts Institute of Technology in 1990 and 1995, respectively. His research interests focus on creating intelligent software tools for engineering design. Dr. Stahovich's current projects include sketch interpretation techniques to enable sketch-based design and analysis tools, techniques for capturing and reusing design knowledge, techniques for automatically documenting designs, and techniques for managing design modifications in large-scale engineered systems.