Reinterpretation of Causal Order Graphs Towards Effective Explanation Generation Using Compositional Modeling

T.K. Satish Kumar
Knowledge Systems Laboratory
Stanford University
tksk@ksl.stanford.edu

Abstract

Compositional modeling provides a number of advantages over conventional simulation software in explanation generation mainly because of its causal interpretation of data. However, little work was done with regard to a supporting algorithm that can generate cogent explanations from the simulation values and causal graphs of model parameters. Earlier attempts did not solve the problem of irrelevant details introduced by using compositional modeling; as a result of which misleading references resulted in attempting explanation of device behavior. This was mainly because they were based merely on equation tracing and did not try to infer anything about the working phenomena from the causal order graph. We present a domain independent algorithm that interprets causal order graphs in terms of working template phenomena rather than in terms of quantities defined in the equation model. A byproduct of this is in capturing the user's psychology in terms of phenomena rather than in terms of mathematical equations defined by some other person. The explanation is in the form of natural language rather than graphs of numerical variables. We also describe a number of extensions of the algorithm to handle issues such as scalability and ranking by significance.

Introduction

We present an algorithm for generating explanations of device behavior characterized by systems of mathematical constraints over model parameters and quantities; without using domain-specific knowledge. Such models are widely used in engineering for dynamical systems, such as electromechanical and thermodynamic control systems. Conventional simulation software can predict and plot the values of these quantities over time given the mathematical equations describing the model. However, this mass of data would be difficult to interpret without any correlation to the structure of the modeled system or the working physical laws.

In engineering tasks requiring design and diagnosis, causal and functional interpretations of data, rather than graphs of numerical variables would prove very useful in anticipating and understanding system behavior. In this paper, we focus on causal interpretations and how we can produce explanations without using knowledge of the domain; interpreting causal order graphs in terms of active template physical phenomena.

The CML (Compositional Modeling Language) (Falkenhainer and Forbus 1991) provides formalisms for integrating models from modular pieces called model fragments, and the DME (Device Modeling Environment) (Iwasaki and Low 1993) provides the environment for their qualitative and numerical simulation along with explanation capabilities. A separate report introduces the intended explanation architecture and describes the text generation and human interface techniques (Gruber and Gautier 1993). Compositional modeling and causal ordering (Kumar 2000), are effective in providing the formalism for capturing much of the knowledge that typically resides in the mind of the designer and is otherwise never communicated to the user. However, the formalism is not complete in itself unless it has a supporting algorithm that extracts the encoded knowledge back to a human-consumable form.

The algorithm presented, works independently of domain knowledge and tries to infer template phenomena of mutation and evolution from causal order graphs which are themselves inferred at run time (Kumar 2000) during simulation. This approach is natural and fits perfectly with the psychology of the user who would be concerned only with the active phenomena and not with the actual equations and intermediate quantities defined in the equation model of the designer. The users can remain oblivious of the peculiarities and particulars in the lines of thought of the designer and the intermediate parameters used in the equation model. This would help them understand and predict the system behavior without any knowledge of the design issues involved in composing the model.

In the next section, we describe two running examples to which we will apply our algorithm and try to illustrate the usefulness of the explanations produced. In the following two sections, we analyze why it works, explaining how combining the techniques of compositional modeling and causality with the supporting algorithm makes it possible to achieve the desired requirements. The final two sections provide a summary and compare related work. The actual algorithm along with other motivating ideas and extensions are described progressively in intermediate sections.

Running Examples

We choose to illustrate the algorithm with the help of two physical settings which are both simple. We choose them in

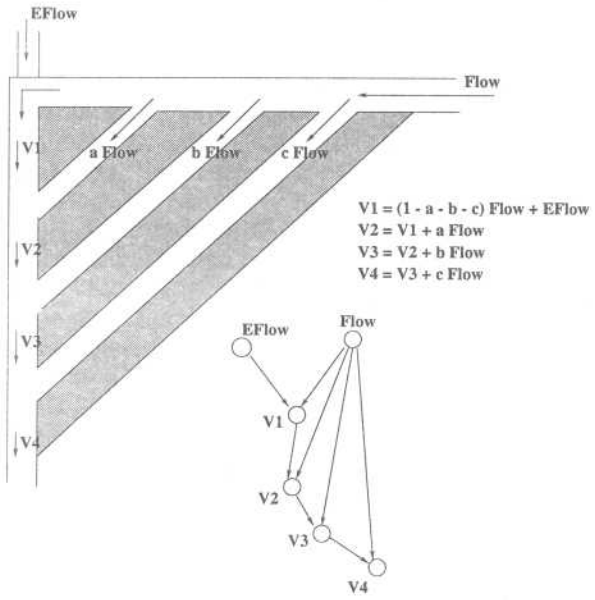


FIG 1: Interconnected Pipelines

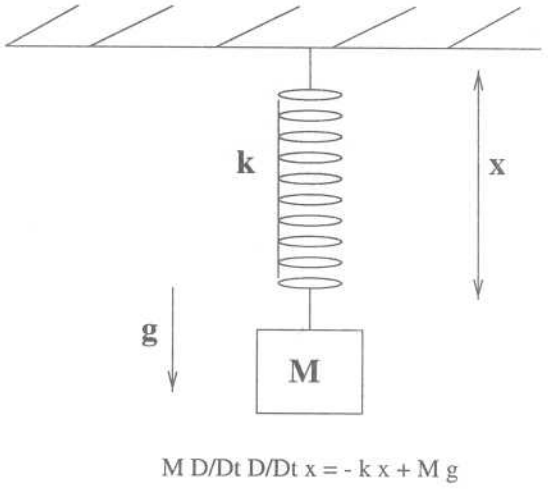


FIG 2: Mass and Spring

particular because they are examples where earlier attempts fail but the new algorithm succeeds. Readers can also convince themselves why the algorithm would apply to any arbitrary scenario.

Interconnected Pipelines

In this scenario [Fig 1], pipes are interconnected in a way that flow at a point is related to flow at other points through equations of conservation. We may want to be able to query why the flow at a particular point changed.

Mass and Spring

In this scenario [Fig 2], the mass is connected to a spring hanging from a horizontal rigid structure. The dynamics of the system would be defined in terms of the gravitational constant, force constant of the spring etc. We will consider the case in which the gravitational constant depends on the distance from the surface of the earth (i.e. related to x).

Compositional Modeling and Explanation

Engineering models used to describe and predict the behavior of systems are typically mathematical models specified by constraints over continuous variables. The world outside the system's abstraction is captured in terms of exogenous quantities and approximations. Model formulation is the task of constructing a model from available primitives to answer some query. Typically, engineering models are constructed by hand by an expert in that domain from background knowledge of physics and engineering.

In the compositional modeling approach (Falkenhainer and Forbus 1991) to model formulation, engineering models are constructed from modular pieces, called model fragments. It is an attempt to relegate the expertise and intelligence required on the side of the designer to a system which can compose sophisticated models from model fragments. A model fragment is an abstraction of some physical domain, mechanism, structure, or other constituent of a model that contributes constraints and partial descriptions to the overall behavior description of a model. Model fragments can represent idealized components such as resistors, transistors, logical gates, electrical junctions, and pipes, and physical processes such as flows.

Each model fragment has a set of activation conditions that specify when an applicable model holds in a given simulation. (e.g. the model of flow of current in an electrical circuit can only hold when there is a voltage source and a closed loop of conductors). Each model fragment has consequences specified in the form of algebraic and logical constraints on the values of simulation variables. The compositional modeling approach assembles a mathematical model from a library of model fragments. Model formulation and simulation are interleaved. During a simulation, the activation conditions of model fragments are monitored; at each state, the system combines the equations of active model fragments into a set called the equation model which it uses to derive a numerical simulation. An equation model characterizes a *qualitative state* which is a period during which the equation model remains unchanged. Within a qualitative state, the numeric values of quantities can change within a certain boundary. When a quantity crosses a boundary, the system triggers the proposal of a new equation model under a new qualitative state. Boundary values are determined by a particular set of model fragments which get activated in that region because of their activation conditions getting satisfied. An equation model changes when new model fragments not originally present in the working model, get activated or some of the already present model fragments get deactivated by virtue of their activation conditions getting satisfied or not respectively. Changes in equation models mark the transitions in qualitative states. These transitions are kept track of by DME and used towards explanation generation.

Interpreting the data produced in simulation requires an understanding of the knowledge used in formulating the model, such as physical mechanisms and component structures underlying the equations. If the engineer looking at the output is not the person who built the model, or if the model is complex and contains hidden assumptions, then it can be difficult for the engineer to make sense of the simulation output. DME's explanation services are intended to address this problem by relating predicted data to the underlying modeling choices and qualitative state transitions which are associated with the formulation of new equation models.

CML and DME play an essential role for explanation by providing the derivation of the equations from the model fragments. This derivation information is exploited in several ways by the algorithm presented in this paper for causal explanation of events occurring in the system.

Causal Ordering and Explanation Generation

DME infers the causal ordering at run time. We use an adaptation of the procedure (developed by the Author) (Kumar 2000) which performs a topological sort on the *PreCondition graphs* over active model fragments. Here, a partial order is created over *active* model fragments based on the relation of subsumption of *preconditions* and *defining quantities* of model fragments. We make use of a number of principles of meta-physics like *the principle of modularity* in inferring the *causal edges* from the topology of the graph. This algorithm is unlike the procedure developed by Simon and Iwasaki (Iwasaki and Simon 1986) which makes simplistic assumptions and therefore does not scale well to complex scenarios. The major defeating assumption that it makes is that the causal order graphs are *acyclic* and that relationships among model parameters always occur in a way such that we never need to solve simultaneous equations. A complete description of the algorithm developed by the Author appears in a separate research paper (Kumar 2000).

Interpretation of the Causal Order Graph - from Quantities to Phenomena

In the causal order graph that results, each arc represents an influence. We use this graph for generating explanations for the behavior of a physical system. The following thoughts and observations inspire us to come up with the algorithm presented later:

- If there are cycles in the graph, then the system *evolves* along those cycles; each quantity influencing the others *continuously*.
- If a considered path is not a cycle, then it represents a flow of influence from an exogenous quantity controlled by a *discrete* event to the other quantities in that path.
- The observation that (1) exhibits a *continuous* flow of influence and (2) reflects a *discrete* flow of influence is similar to the reproduction and mutation stages in evolution with simulation states corresponding to generations.
- As in the case of biological theories which try to answer the queries related to a generation of individuals by references to their ancestors in previous generations and possible mutations in that generation, we try to explain observations about quantities in a simulation state by references of the continuous phenomena to the previous simulation states and the discrete phenomena to the present simulation state. Thus, the algorithm reasons across different simulation states.

Apart from these inspirations, the user psychology is captured in a way that translates into ideas that fit well with this. This part is introduced at a later stage after the explanation of the notion of *Causal Colorings*.

The Algorithm

The algorithm proceeds in phases each of which tries to capture some crucial notions towards producing cogent explanations.

Causal Colorings

Our first task in explanation generation is to characterize how a quantity is influenced. There are two ways possible for anything to affect a quantity.

- The cycles with which it is involved.
- Other exogenous quantities that have reachability to this quantity along a directed path of causal influences. A quantity q is *reachable* from a quantity p if there exists a directed path from p to q.

In what follows, the terms *quantity* and *node* are used interchangeably. We imagine a *bag* at each node in the causal order graph that is eventually going to contain information about how that quantity is influenced. At this point, let us define a *color* to be a characteristic of an influence of a continuous or a discrete happening. There are colors associated with each cycle in the graph as well as with each exogenous quantity. We assume that each color is distinct.

In the first step of this phase of the algorithm, all the exogenous quantities add their colors to the bags of all the nodes that are reachable along directed edges from that quantity. In the second step, whenever a cycle is detected, each member of that cycle gets a color characterizing that cycle in its bag. Cycles can be detected easily as a side-effect of finding reachability graphs. At the end of it all, each quantity ends up having a bag of colors which characterize what phenomena affect its changes.

Causal Frontiers and Incremental Explanation - Capturing the Psychology of the User

This is the part of the algorithm that tries to capture the psychology of the user in the sense that when a user queries about what caused the changes in a quantity at a given simulation state, he does not expect mere equation tracing. For example, previous attempts (Gruber and Gautier 1993) tried to report each of its causal parents and tried to skip a parent if it had just one other parent. This is too naive an idea because it fails in very simple settings such as the one described in the interconnected pipelines scenario [Fig 1]. If nothing else has changed except for *Flow* and the user queries about the cause for the change in v_4, he is made to go through all the

potentially many intermediate flow points v_3, v_2 .. etc. This can be very misleading and confusing to the user. This is precisely the drawback of working with *quantities* that are used in equations, rather than trying to infer something about the *physical phenomena* that the designer is trying to encode through those equations. The following idea based on the assignment of colors to each bag obtained after the *Causal Coloring* phase, captures these expectations of the user.

- set S = {Parents of query node}
- **Until** there is no change to S **do**
  for t **in** S such that all parents of t are either in S or are of the same colour bag as that of t, set S = S - {t} U Parents(t)
- Maintain a list so that once a node is considered and deleted, we rule out the possibility of adding it again. This is just to avoid infinite loops.

The above procedure defines a *causal frontier* for each query; which is the set S which results from the above algorithm. Finally, the explanation is attributed to some simulation state (which will be decided later) of all the nodes in this *causal frontier*.

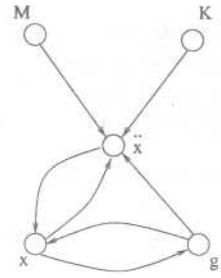
There is a psychological reason behind why the user will expect explanations attributed to the *causal frontier* that gets defined as above - the reason is that the user thinks in terms of what *phenomena* may have caused the change; not in terms of what *equations* might have affected the quantity. Also, if the user expects *incremental explanations* in a step by step fashion, then he would expect an explanation or a reference to the set of phenomena that got added or deleted from the set which he presently imagines to be in the working model. The *colors* rightly capture the different *phenomena* that cause *influence flows* and the *causal frontier* just described, rightly captures the psychology of *incremental explanations*. The property that we are ensuring during deletion of a node from the set S is that the set of phenomena remains the same in the working model; but at the same time, we are maximizing the depth of the reference. It is the trade-off between the dual goals of *incremental explanation* and *removing irrelevance*.

Causal Stages - Deciding Simulation States

As was explained earlier, as to why we would like to associate continuous changes with previous simulation states and discrete changes with the current simulation state, we now find for each quantity in the causal frontier, the corresponding cycles and paths involving itself and the query quantity; and generate the appropriate references (text strings) for each such cycle or path. Any straightforward text generation procedure to produce an interface similar to the one shown in the examples may be used. The assumption is that the equations associated with edges in the graph are kept track of, so that we could retrieve that information during actual explanation text output.

Behavior on Running Examples

In the interconnected pipelines scenario, earlier attempts to query for *what changed flow at* v_4 *?* caused explanations attributed to v_3 v_2..etc and then this whole chain of potentially many such points had to be traversed; even though the actual cause might have been just the change in *Flow*. It is easy to verify that this algorithm identifies that easily. The causal order graph for this scenario is shown in [Fig 1]. Initially, the parents of v_4 are v_3 and *Flow*. However, v_3 can be removed from S and replaced by v_2 because of v_2 having the same color bag as that of v_1 ($\{Flow, EFlow\}$) and the other parent of v_3, namely *Flow*, already being in S. Similarly, v_2 can be eliminated and replaced by v_1. However, v_1 cannot be eliminated because its parent $EFlow$ is neither already in S nor has the same color bag as v_1 (color bag of $EFlow = \{EFlow\}$). Therefore, the *causal frontier* for v_4 would consist of $Flow$ and v_1. An intermediate explanation point directly at v_1 is produced for further querying because it is only here that there is a possibility of *EFlow* having caused the change too. An extension of the algorithm described under a later section, introduces the idea of *Ranking and Significant Changes* with which we can directly attribute the change in value of v_4 to *Flow* if we know that *EFlow* has not changed over the concerned time interval.



Bag(x) = {K, M, <y,x>, <x,y,g>, <x,g>}
Bag(y) = {K, M, <y,x>, <x,y,g>}
Bag(g) = {<x,y,g>, <x,g>}
Bag(K) = {K}
Bag(M) = {M}
y = D/Dt D/Dt x

FIG 3: Causal Order Graph and Bags for Mass and Spring Scenario

The corresponding bag assignments for the spring and mass scenario is shown in [Fig 3]. Notice here that we have ignored the dependence of g on R_e (radius of the earth) and g_0. It is trivial and inconsequential to add them in. The readers can verify the relevance of the explanations produced upon querying a simulation value of a quantity. For example, the references combined with a few straightforward NLP techniques might produce something like:

Q: What changed the value of the length of the spring ??
A: In the previous state it had a value [] and its second derivative had a value [] which affected it in this state because of the fact that it is the second derivative.
[In The Previous State]
Q: What changed the value of the second derivative ??
A: In the state before, x had a value [] which affected it through the equations [] along the evolutionary cycle []. Also, in the state before, g had a value [] which affected x through the equations [] along the evolutionary cycle []. In this state however, it was also affected by the new values of K = [] mutating it through the equations [] and M = []

mutating it through the equations [].

It is possible for the system to use more domain specific strings called *pretty strings* (Gruber and Gautier 1993), instead of the words *evolutionary* and *mutating* when these are provided by the user. This information can be incorporated at different levels of modularity during model composition.

Extensions

Ranking and Significant Changes An important extension of the algorithm is that of ranking the explanations produced in terms of their importance. For example, although many quantities might have a causal influence on a given quantity, it is not wise to treat these causal influences statically. We might be interested in giving references to the quantities that have varied significantly over the period of interest as our primary explanation and giving the other static references as an aside to the user. For example, in Q2, where K and M usually do not vary, the references to them are produced last or perhaps only upon user request. This approach requires the definitions for specifying quantitatively, how much a quantity has varied in its recent history. Also, analysis of how much a small variation in one quantity affects the other, should be made. A perturbation analysis can be done over the quantities; at least approximately, from the equations provided.

Incorporation of this feature in the algorithm would be reflected as an ordering on the nodes of the *causal frontier* based on a measure of the variation of quantities depending upon their simulation values across different simulation states. The metric of variation may be something similar to the adjustment of process priorities in operating systems (solaris) where we give more importance to variations in the near past. The system can be given parameters for defining thresholds towards ignoring quantities that rarely change. For example, we may not want any references to K or M if we know that they are not going to change throughout the simulation. Typically, such quantities fall under the *throughout* conditions specified in the definition of *scenarios* using CML. However, explicit requests for actual equation tracing can also be made. Sometimes, it may be difficult to get a total ordering on the quantities in the *causal frontier*. In practice however, a *partial order* is good enough for significantly improving the quality of explanations produced.

Incremental Version Since causal order graphs are dynamic; in the sense that new quantities and dependencies may get established, depending upon new model fragments getting activated, we feel the necessity of an incremental algorithm. An incremental step upon the addition of extra nodes or edges in the graph can be taken towards developing an incremental version of the algorithm that copes with the requirements. If the new node is an exogenous quantity, we just construct its reachability graph and add its color to the bag of colors at all those nodes. If it is not an exogenous quantity, then we first cater to its incoming edges by letting its bag of colors to be the union of all the colors of its in-neighbors and then we construct its reachability graph and add to all reachable nodes, all the colors in its bag. If the addition of the node creates cycles, which we can detect in computation of its reachability graph, we add a cycle characteristic color to each member of the cycle. Creation of multiple cycles is also detected in the same way. Incremental addition of edges have a very similar treatment except that no new colors by virtue of exogenous nodes enter the system. Similarly, deletion of edges and nodes can be done by imagining the notion of *anti-colors* which cancel out the presence of a particular color when added to a bag that contains the color; and which have no effect on a bag that does not contain the color.

Summary and Analysis

The use of compositional modeling and causal ordering techniques is responsible for several desired properties of the explanation approach we have presented.

First, it is possible to generate causal interpretations of models that are designed for engineering analysis and simulation, rather than being crafted specially for explanation. Because explanation is integrated with compositional modeling, explanations of the causes of changes in qualitative state can be determined by an analysis of the logical preconditions of model fragments that are activated and deactivated. The set of conditions to report need not be anticipated in advance, and it can change as the model fragment library evolves.

Second, the explanations can be presented in a suitable format for human consumption. Ultimately, this is the feature that is the most required and crucial. This is ensured by the algorithm described in this paper which makes use of the formalisms that CML provides in inferring the active phenomena that the designer is attempting to model. The approach taken in doing this independently of the domain, is to characterize phenomena as falling under templates of evolution and mutation and inferring them from the causal order graph (which is itself inferred at run time from the *Pre-Condition graph* over active model fragments; using general principles of meta-physics (Kumar 2000)). This provides great advantages in generating explanations of device behavior which are cogent enough to the user who knows nothing about the design issues or the internal parameters and quantities the designer used in building the equation model.

The explanation module can reason across different simulation states and relate present behavior of the system to its past, allowing the user to query and know about the complete history of the system and understand why it followed a particular course of behavior. Since explanation is in the form of natural language and not graphs of numeric variables, the user can be provided with a causal interpretation of the events occurring in the system. None of the explanation code knows anything about flows, or junctions. It knows only about the structure of model fragments-activation conditions, behavior constraints, quantity variables, and some algorithms for extracting knowledge captured in these model fragments towards text generation for explanation of system behavior. Furthermore, the model builder may add textual annotations incrementally which the system would compose to more informative labels. This capability is possible because of the modularity and compositionality enabled by the compositional modeling.

Related Work

Much of the work in explanation has concentrated on the generation of high-quality presentations in natural language based on discourse planning and user modeling ((Feiner and McKeown 1990), (Forbus 1984), (Suthers, Woolf and Cornell 1992)). There have been various efforts in the past towards the task of giving a causal interpretation to device behavior ((de Kleer and Brown 1986), (Iwasaki and Simon 1986), (Top and Akkermans 1991)). Some of the approaches include context sensitive causality (Lee, Compton and Janson 1992), bond graphs (Top and Akkermans 1991) etc. These methods differ in the information they require and the class of models they accept. The algorithm described in this paper can be coupled with any of these techniques; it is not tailored specifically for the causal order graphs produced in DME alone.

In QUALEX (Douglas and Liu 1989), a causal graph is computed from a set of confluences (de Kleer and Brown 1984), and the graph is interpreted by the propagation of qualitative perturbations. However, the confluence equations can only predict the sign of the first derivative and do not scale well with dynamically changing models. Although in general, Qualitative models do not scale well, they have been used to generate explanations in tutoring and training systems (White and Frederiksen 1990). DME uses QSIM (Kuipers 1986) for simulation of Qualitative models.

SIMGEN ((Falkenhainer and Forbus 1992), (Falkenhainer and Forbus 1990)) employs compositional modeling approach to build parallel qualitative and quantitative model libraries, analyze the possible qualitative state transitions for a given scenario description, and compile out an efficient numeric simulator. While DME determines model fragment activation and assembles equation models at run time, SIMGEN precomputes and stores the information relating the quantitative model and the qualitative model. SIMGEN reasons only across single-step influences and does not summarize chains of influences.

The algorithm described in this paper, has been implemented (by the Author) at the *Knowledge Systems Laboratory, Stanford University* and applied in a number of scenarios involving physical systems and devices (much more complicated than those illustrated in this paper), and successful explanations for system behavior produced.

Acknowledgements

The DME (Iwasaki and Low 1993) system is the work of multiple researchers, including Richard Fikes, Yumi Iwasaki, James Rice, Adam Farquhar and Tom Gruber; who are (were) the key members of the *How Things Work* Project at the *Knowledge Systems Laboratory, Stanford University*. We also thank Richard Fikes and Sheila McIlraith for their guiding comments on the paper.

References

[1] Crawford, J.; Farquhar, A.; and Kuipers, B. 1990. QPC: A Compiler from Physical Models into Qualitative Differential Equations. AAAI-91, pp. 365-371.

[2] de Kleer, J.; and Brown, J. S. 1984. A Qualitative Physics Based on Confluences. Artificial Intelligence, 24:7-83.

[3] de Kleer, J.; and Brown, J. S. 1986. Theories of Causal Ordering. Artificial Intelligence, 29(1):33-62.

[4] Douglas, S. A.; and Liu, Y. 1989. Generating Causal Explanation from a Cardio-vascular Simulation. IJCAI-89, pp. 489-494.

[5] Falkenhainer, B.; and Forbus, K. 1992. Self-explanatory Simulations: Scaling up to Large Models. AAAI-92, pp. 685-690.

[6] Falkenhainer, B.; and Forbus, K. 1991. Compositional Modeling: Finding the Right Model for the Job. Artificial Intelligence, 51:95-143.

[7] Feiner, S. K.; and McKeown, K. R. 1990. Coordinating Text and Graphics in Explanation Generation. AAAI-90, pp. 442-449.

[8] Forbus, K. 1984. Qualitative Process Theory. Artificial Intelligence, 24:85-168.

[9] Falkenhainer, B.; and Forbus, K. 1990. Self-explanatory Simulations: An Integration of Qualitative and Quantitative Knowledge. AAAI-90, pp. 380-387.

[10] Gruber, T. R.; and Gautier, P. O. 1993. Machine-generated Explanations of Engineering Models: A Compositional Modeling Approach. IJCAI-93.

[11] Iwasaki, Y.; and Low, C. M. 1993. Model Generation and Simulation of Device Behavior with Continuous and Discrete Changes. Intelligent Systems Engineering, 1(2).

[12] Iwasaki, Y.; and Simon, H. 1986. Causality in Device Behavior. Artificial Intelligence, 29:3-32.

[13] Keuneke, A. M.; and Tanner, M. C. 1991. Explanations in Knowledge Systems: The Roles of the Task Structure and Domain Functional Models. IEEE Expert, 6(3):50-56.

[14] Kuipers, B. 1986. Qualitative Simulation. Artificial Intelligence, 29:289-388.

[15] Lee, M.; Compton, P.; and Jansen, B. 1992. Modeling with Context-Dependent Causality. In R. Mizoguchi, Ed., Proceedings of the Second Japan Knowledge Acquisition for Knowledge-Based Systems Workshop, pp. 357-370.

[16] Paris, C. 1987. The Use of Explicit User Models in Text Generation: Tailoring to a User's Level of Expertise. PhD Thesis, Columbia University.

[17] Suthers, D.; Woolf, B.; and Cornell, M. 1992. Steps from Explanation Planning to Model Construction Dialogues. AAAI-92, pp. 24-30.

[18] Top, J.; and Akkermans, H. 1991. Computational and Physical Causality. IJCAI-91, pp. 1171-1176.

[19] White, B.; and Frederiksen, J. 1990. Causal Model Progressions as a Foundation for Intelligent Learning. Artificial Intelligence, 42(1):99-1550.

[20] Kumar, T. K. S. 2000. A Compositional Approach to Causality. Proceedings of the Symposium on Abstraction, Reformulation and Abstraction, 2000. To appear in Lecture Notes in Artificial Intelligence.