AD-A034 194 MARYLAND UNIV COLLEGE PARK DEPT OF COMPUTER SCIENCE F/G 9/2
THE CAUSAL REPRESENTATION AND SIMULATION OF PHYSICAL MECHANISMS--ETC(U)
NOV 76 C RIEGER, M GRINBERG N00014-76-C-0477
UNCLASSIFIED TR-495 NL

1 OF 2
AD-A
034 194



U.S. DEPARTMENT OF COMMERCE
National Technical Information Service

AD-A034 194

THE CAUSAL REPRESENTATION AND SIMULATION OF PHYSICAL MECHANISMS

MARYLAND UNIVERSITY, COLLEGE PARK, MARYLAND

NOVEMBER 1976

012008

ADA034194

COMPUTER SCIENCE TECHNICAL REPORT SERIES

UNIVERSITY OF MARYLAND
COLLEGE PARK, MARYLAND
20742

REPRODUCED BY
NATIONAL TECHNICAL
INFORMATION SERVICE
U. S. DEPARTMENT OF COMMERCE
SPRINGFIELD, VA. 22161

DDC
RECEIVED
JAN 12 1977
A

TR-495
N00014-76C-0477

November 1976

THE CAUSAL REPRESENTATION AND SIMULATION OF PHYSICAL MECHANISMS

Chuck Rieger and Milt Grinberg
Department of Computer Science
University of Maryland
College Park, Maryland 20742

ABSTRACT: The end goal of the research described here is to understand human problem solving. In this paper we examine how a human might represent and use cause-effect knowledge about physical mechanisms and laws, since (we believe) physical mechanisms provide a tangible domain in which many of the interesting topics of problem solving can be studied. First, we develop and illustrate a system of representation for cause-effect knowledge of physical mechanisms. This representation seems to be applicable to, and possibly adequate for, a wide variety of mechanisms and physical laws. A home gas forced air furnace is illustrated in detail. Then, we describe a theory and computer implementation of a mechanisms simulator which can accept any mechanism described in our representation, automatically translate it into a population of associatively-triggerable computation units, then, upon suitable triggering, "execute" the mechanism, tracing its cause-effect behavior in time. Since the translation from declarative to procedural form is uniform with respect to our system of representation, the simulator will be extendable to simulations in the social and psychological domains. In this sense, we are developing a more general "plan simulator" which will interact with the existing CSA plan synthesizer as it builds complex plans. We also consider the interesting area of mechanisms invention. Both the theory and LISP implementation of our "Mechanisms Laboratory" are described, and appendices containing additional representation examples and simulator traces are included.

The research described in this report was funded by the Office of Naval Research under contract number N00014-76C-0477.

DDC
RECEIVED
JAN 12 1977
A

CONTENTS

1. Introduction 1

1.1. Background 2
1.2. Related Work 4

2. The CSA Cause-Effect Description Language 6

2.1. Resolution Level Flexibility 7
2.2. An Example 8
2.3. The Cause-Effect CSA Links 17
2.3.1. CAUSE 17
2.3.1.1. Gating 18
2.3.2. STATECOUPLE 18
2.3.3. EQUIVALENCE 19
2.3.4. ANTAGONISM 23
2.3.5. ENABLEMENT 24
2.3.6. THRESHOLD 26
2.3.7. RATE CONFLUENCE 27
2.4. Event Representation 28
2.4.1. States 29
2.4.2. Statechanges and Descriptors 30
2.4.3. Actions and Tendencies 33
2.5. Uniqueness 34

3. Spontaneous Computation and Mechanism Simulation 35

3.1. Simulation Strategy 35
3.2. CSA Implementation of Spontaneous Computation 37
3.2.1. Channels 39
3.2.2. Simulation 41

4. The Mechanism Simulator 43

4.1. Overview 43
4.2. Example Simulation 44
4.2.1. Input Syntax 48
4.2.2. Conversion from Declarative to Procedural Form 51
4.2.3. Simulation 53
4.3. Spontaneous Computations 55
4.4. Overview of Simulator 56
4.4.1. Initial World 56
4.4.2. Trigger Events 57
4.4.3. Context Level 57
4.4.4. Data Base Interaction 57
4.4.5. Mech-Agenda 58
4.4.6. CSA Number System and Arithmetic 58
4.5. Conversion of Declarative Description to Procedural 59
4.5.1. Continuous Scale Statechange 59
4.5.1.1. Instantaneous Descriptor Event (ID-EVENT) 63
4.5.1.2. Statechange Event 64
4.5.1.3. Rate Confluence Link 64
4.5.1.4. Rate Calculation 66
4.5.1.5. Example of RATE-CONFL 67
4.5.1.6. Threshold Events 69
4.5.1.7. Converting Thresh-Events into Triggering Conditions 70
4.5.1.8. Threshold Link 72
4.5.1.9. State Incrementor 74
4.5.1.10. Statechange Links Example 75
4.5.1.11. Time in the Simulator 76
4.5.2. Non-Statechange-Related Links 78
4.5.2.1. Hiding and Unhiding SC's 78
4.5.2.2. Continuous Causal Link 80
4.5.2.3. One Shot Causal Link 81
4.5.2.4. State Couple Link 82
4.5.2.5. State Equivalence Link 84
4.5.2.6. Enablement Links 87

4.5.2.7. Antagonism Link 88
4.6. User Notes 90
4.6.1. Normal Termination 90
4.6.2. User Interaction 92

5. Next Phases of the Mechanisms Lab 94

5.1. Interfacing with General World Knowledge 94
5.2. Mechanism Invention 95

6. Conclusions 96

7. References 97

Appendix A: Computer Flip-Flop 99

Appendix B: Incandescent Light Bulb 106

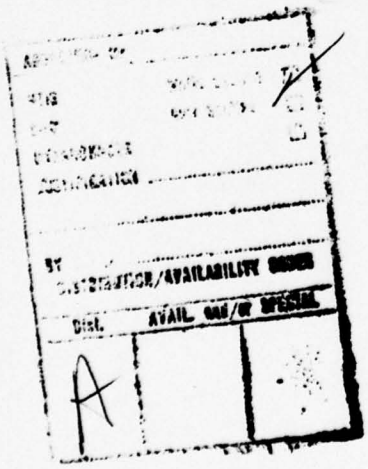
Appendix C: Drinking Duck 112

Appendix D: Train Wreck 116

Appendix E: Bicycle Horn 126

ACKNOWLEDGEMENTS

We thank Phil London, Mache Creeger and John Boose for their comments and suggestions. Additionally, we thank Boose for his Drinking Duck, and Creeger for his Train Wreck.



The Causal Representation and Simulation of Physical Mechanisms

Chuck Rieger and Milt Grinberg
Department of Computer Science
University of Maryland
College Park, Maryland 20742

1. Introduction

This paper describes a theoretical framework and a LISP implementation for describing and simulating the cause-effect behavior of mechanisms. For the purposes of this research we define a mechanism to be any physical device, complex or simple, which exhibits cause and effect relationships useful to humans. Ordinarily, this will mean any purposively constructed object, such as a vacuum cleaner, a pencil, a button, a lightbulb or a computer. However, we will also include in the definition any naturally-occurring physical devices and principles whose cause and effect relationships are of use to humans. Additionally, we will wish to consider information-manipulating "mechanisms" such as computer programs as though they were physical in nature.

We will first define and illustrate a representation language for expressing physical cause-effect knowledge, showing some examples of mechanisms represent in these terms. Then we will we describe a system called the "CSA Mechanisms Laboratory" which will accept any mechanism defined in this descriptive language, convert this declarative form into a population of demon-like procedures, then activate the population by allowing the population to "see" a stimulus to which the mechanism would react. In this manner the simulation will test or demonstrate the mechanism with respect to its internal and overt cause-effect behavior. The simulator will also provide the foundation for a "what-if" QA or design system wherein the user (or a Mechanism Inventor subsystem) can experiment with new designs, interface numerous submechanisms, etc.

1.1. Background

Our research into mechanisms has been motivated from within the context of a larger project to study human intelligence, called the Commonsense Algorithm Project. The focus of the project is on representations of knowledge and processes which are relevant to two aspects of human intelligence, (1) problem solving and (2) language comprehension. We are especially interested in defining what it is these two aspects of human intelligence have in common. So far we have a representation language, an operational plan synthesizer which conforms to our theory of representation, and some basic language comprehension tools which, e.g., allow the system to arrive at context-sensitive interpretations of sentences. We are presently beginning a larger scale project to apply the CSA theory to the comprehension of a full children's story. These parts of the CSA project have been described in [R1], [R2], [R3], and [R4].

Why study mechanisms within the framework of a model of human intelligence? There are several important reasons for doing so. The most compelling one is this: a mechanism which was invented by a human is a frozen exemplar of human problem solving. Most acts of problem solving are fleeting, in that no record of them is preserved; that clever strategy we employed to fix the TV yesterday must be carefully documented after the fact in order to be preserved. However, any mechanism is an inherently self-documenting example of human problem solving, at least insofar as it represents a final product. It is therefore an interesting place to begin in the study or reconstruction of the underlying processes of mechanism invention, and hence (it is believed) all human problem solving.

There are many ways to approach the description of a mechanism. Most in the past (e.g. [JW1], [LBR1], [RJS1], [SL1], and [WBL1]) have tended to be highly analytical in their approach. In analytical simulations, the mechanism is typically described by parameterizing it in a form suitable for, say, a numerical simulator. The problem with this type of approach, at least from our point of view, is that the representation of a mechanism must then be very different in form from the general theoretical representations we are proposing as the basis for human knowledge of cause and effect. In other words, since most simulation systems have had goals <u>other</u> than the integration

of mechanisms description into a larger framework of human intelligence, they employ representations that are probably quite different from the representations humans employ. Hence, although a more conventional simulation system would probably be of more practical value for any given real-world application, we want to represent and simulate mechanisms in a way which we feel is closer to the way humans might do these things.

It is this idea that makes mechanisms description and simulation of interest within the CSA framework. Beyond this, there are some other specific benefits to be derived from such research. Among them are:

(1) The representation will provide a theory of cause and effect which will allow large numbers of mechanisms to be described, catalogued and retrieved on the basis of utility (i.e. what they do when activated) or on the basis of the cause-effect principles upon which they are founded, rather than on their physical or mechanical properties. This could, e.g., make it possible for a design engineer to state subgoals of a design in terms of what effects he desires from each subsystem, and let the searching functions retrieve relevant mechanisms from the cause-effect pattern library. Since patterns in the library would specify not only the effects of each mechanism, but also the requirements (preconditions) for each, the designer who commits himself to a particular sub-mechanism could also be advised of the mechanism's preconditions. These could help direct him in other parts of the design.

(2) A dedicated background pattern matcher with access to a large database of mechanisms from all aspects of science could continuously seek commonly recurring subpatterns. This could help to identify common principles across the many disciplines of science and engineering, serving to fertilize areas. Science has apparently become so complex that any given principle is prone to reinvention many times within different disciplines before anyone recognizes it as a common idea. By describing cause and effect in a uniform and abstract enough way, interchanges of ideas and discoveries could be facilitated by such an automatic searching

system.

(3) By representing and simulating mechanisms in the terms presumed to be employed in general human problem solving, we set the stage for an automated mechanism inventor, debugger and tester. Two modes of use suggest themselves. In the first, a mechanically or electrically naive user would be able to confront the system with his need, causing the system either to search for an existing mechanism satisfying that need, or to invent a mechanism to fill the need, drawing upon its knowledge of existing mechanisms. In the second mode, the system would run free, simply trying new combinations of mechanisms in its "spare time". More often than not, we would expect either a dismal failure or a Rube Goldberg device; however, it could happen that such a system would occasionally stumble across some new insights.

(4) The simulation aspect of the Mechanisms Lab would allow a trainee or a mechanically naive person to experiment with a mechanism at a conveniently high level. Rather than having to alter often obscure numerical parameters between simulations, the user would be able to converse with the system in terms like: "Suppose WIRE23 breaks. What would happen?". Conversely, a diagnostician might confront the system with a condition and ask what circumstances might possibly have caused such a condition. This would presumably be of use to car mechanics and clinical pathologists alike.

Some of these (particularly 2!) are blue-sky ideas. Lest we induce an undue level of excitement in the reader, we will now describe the theory and computer system as they stand, suggesting where appropriate what lies ahead in the immediate future.

1.2. Related Work

We should preface the discussion by pointing out that we are by no means the only group doing research in this general area. The MYCIN medical diagnosis project at Stanford [D1] has been constructing models of the

techniques clinical pathologists presumably apply when attempting to make sense out of the raw data which describes some possible pathology or set of pathologies. In this sense they too are modeling cause-effect mechanisms. But their system is, first of all, more specialized than ours. Beyond this, whereas we are attempting to specify both the conceptual contents of the representation as well as its internal structure in a computer, the MYCIN project seems principally interested in the internal structure, and intends not to restrict the semantics (contents) of what is represented. Also, the MYCIN project has has no strong requirement for a general mechanism simulator.

Another related project (which predates ours) is Sussman's and Stallman's electronic circuit analysis program ([SS1] and [SS2]). It is both an application program and a theory of circuit design. The main concerns of the project are: how are laws of electronics expressed in a way that is of use to the analysis of a given circuit; e.g. the user specifies the starting states of all "active" components such as transistors, then asks the program to analyze the circuit's behavior, or the user requests the system itself to derive the various modes of operation of the active devices in the circuit and to the design of a circuit from descriptions of what it should accomplish.

One high level goal of this project seems to be identical to one of our high level goals: to shed light on the human problem solving process. One area of emphasis in Sussman's and Stallman's work is the use of "dependencies" as a model of hypothetical reasoning. However, since our overall focus is different from their effort, we encounter a different set of issues at the next level down from this highest goal of understanding human problem solving.

A third piece of research which we find very compatible with our present goals is Freiling's system of description for mechanical mechanisms [F1]. In this system Freiling has considered a wide variety of fairly abstract relationships which describe the physics of mechanical mechanisms (relations like rigid attachment, motion conversion, conduits, etc.) We regard his level of description as the level immediately "below" the cause-effect level at which we describe a mechanism. For our system to become complete, in the sense that the cause-effect patterns are finally tied to the detailed physics of the mechanism they describe, we will want to interface it with a Freiling-like description.

2. The CSA Cause-Effect Description Language

In the CSA representation, we have attempted to identify a small number of cause-effect relationships which are necessary and sufficient for connecting together events into larger patterns which then reflect the operation of a mechanism. There are 16 relationships (links) in the current CSA theory, but since some of them deal with intentionality and other concepts relating to conscious human activity, we require only about half of the relationships in the representation of mechanisms. Although this subset once was rather transient, after having examined a reasonably broad spectrum of mechanisms, it seems to have stabilized on the set we will present in a moment. We are not yet prepared to assert that this is the sought-after necessary and sufficient set of relations, nor are we prepared to defend this set against other possible sets. We merely point out that this seems to be a reasonable way to factor cause-effect relationships in mechanisms, and that we are growing increasingly confident that the set is at least necessary. Of course, one never knows whether he is looking through rose colored glasses after growing accustomed to a particular system of representation; perhaps we like the system because we have learned to ignore those parts of reality for which the system fails! We hope this is not the case, and feel it is not.

The CSA links connect events. An "event" in the terms of the Mechanism Lab is always exactly one of the following:

(1) an ACTION, the generation of some type of force by an animate actor

(2) a TENDENCY, an inanimate force-generator (i.e. a natural phenomenon)

(3) a STATE, a static condition

(4) a STATECHANGE, a changing condition with respect to some continuous attribute (e.g. temperature, distance, volume, voltage)

We can regard these four categories as syntactic types for events. As a result, it will be possible to define a syntax for each cause-effect link. Such a syntax will specify the types of events which any given link can structurally relate, regardless of the event's "meaning". Having such a

syntax helps standardize the way in which we approach the representation of a mechanism; it typically will prevent us from "cutting corners", or becoming sloppy in the representation, and it will frequently force us into making explicit important aspects of a mechanism that might otherwise have remained only implicit (e.g. since the syntax prescribes the existence of some intervening state, we are forced into verbalizing that state).

2.1. Resolution Level Flexibility

We have felt from the beginning that an important characteristic of a mechanism description language, and indeed of any language in which problem-solving knowledge is couched, is that it be able to accommodate any level of descriptive resolution desired. "Resolution" means the amount of detail included in the description. This seems essential for several reasons. First, and most obvious, the description will only be as good as the human's knowledge of the mechanism he is describing. He simply might not know much about the mechanism, or he might know more detail about some of its aspects than others. He ought to find it possible and comfortable to express precisely what he does know, without always being forced into too much detail. For example, most people can give a very defocused description of how a radio works, but few can describe the cause-effect relations more than a few levels deep, beginning with the power plug and the various front-panel controls. We want to be able to accommodate defocused descriptions as well as the high resolution details provided by the theoretician who invented the radio, or the design engineer who implemented it.

Another obvious reason for resolution level flexibility is that even an expert's knowledge stops at some point (or so it is rumored!). There will always be events which are known to occur but are not explainable in causal terms. It will clearly be imperative that the descriptive language allow the expert to bow out gracefully where his knowledge terminates.

Finally, there will frequently be applications where it might be irrelevant to describe some part of some mechanism in detail. If, for instance, we want to focus on the receiver section of a radio, we might want to provide only a sketchy coverage of the power supply circuit, describing

just enough to account for the receiver section's dependence upon it.

To achieve resolution level flexibility, we employ several links which can be used as level shifters. We will discuss these at the appropriate time in a later section.

2.2. An Example

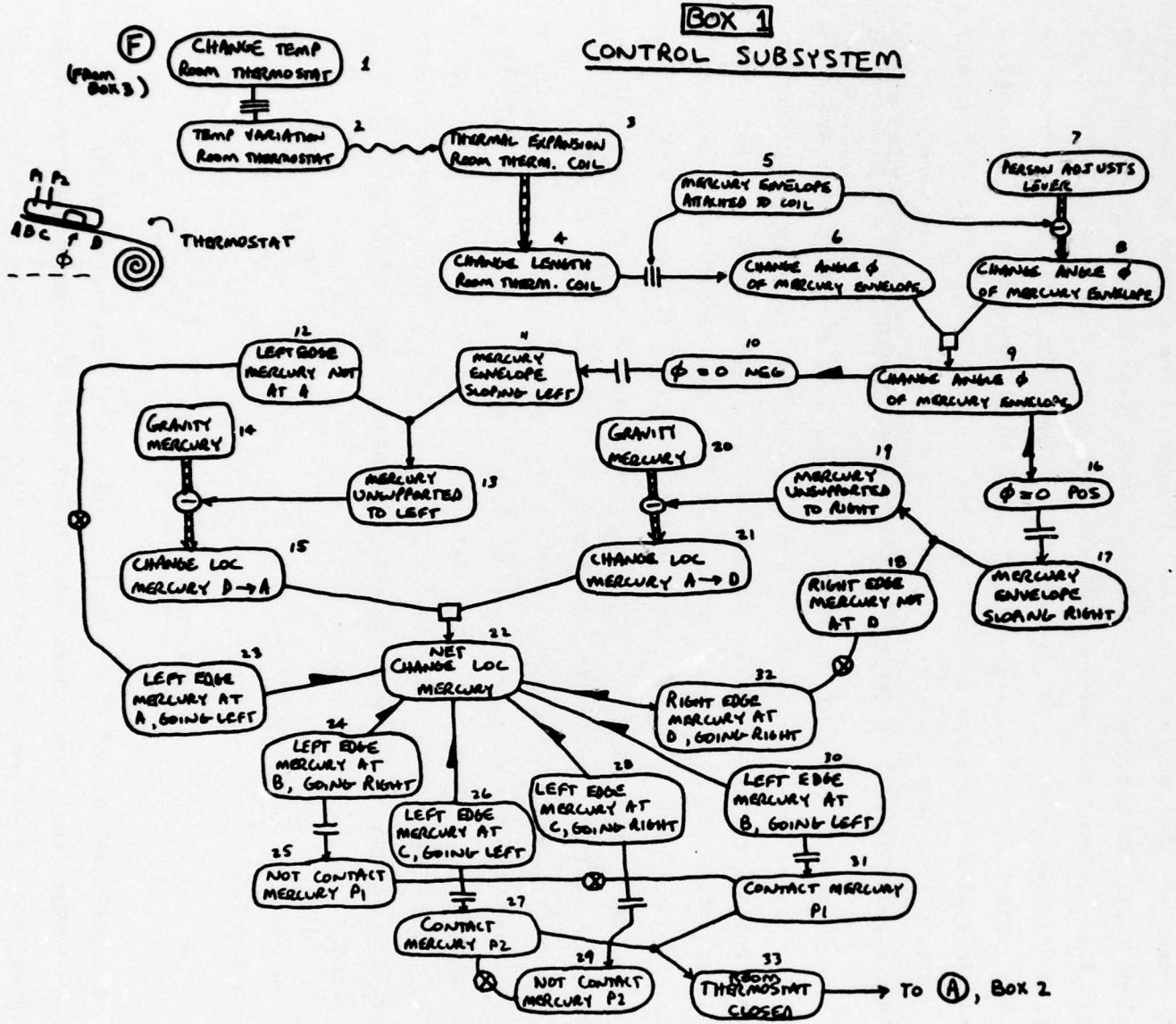
Before describing the representation link-by-link, we will trace through an example mechanism which has been represented in the CSA Mechanisms Lab form. Our example is the Gas Forced Air Home Heating System, and is of interest because it brings together quite a number of physical principles into one mechanism with several modular subsystems and some interesting feedback loops.

The CSA graph which represents the GFA Heating System is shown in Displays 1a, 1b and 1c. The furnace can be described in terms of three subsystems: the Control Subsystem (Box 1), the Heating Subsystem (Box 2), and the Delivery Subsystem (Box 3). Tie-points A through G (in circles) denote inter- or intra-box connections which would be too cumbersome to draw explicitly. To avoid confusion, we will omit references to numerical rates in this example. Also, rather than representing each event as formally as would be required by the simulator, we use descriptive English phrases.

BOX 1 (Control Subsystem)

We assume that the system heats a single room, and that this room contains a mercury-filled, glass-envelope style thermostat. A temperature change of the thermostat (1) is equivalent to a state of temperature fluctuation (2) (upper left hand corner). Such fluctuation continuously enables the tendency THERMAL EXPANSION (3) to produce a continuous change in the length of the (coiled) thermostat strip (4). Provided the glass envelope is attached to this coil (5), this length change is equivalent to a change in the angle (phi) of the glass envelope (6). The angle of the glass envelope can also be influenced by an external adjusting action (7,8). At any moment, the net change (9) of this angle is governed by these two sources.

As phi changes, there are two points of interest: one when phi thresholds


BOX 1
CONTROL SUBSYSTEM
F
(FROM BOX 3)
1 CHANGE TEMP ROOM THERMOSTAT
2 TEMP VARIATION ROOM THERMOSTAT
3 THERMAL EXPANSION ROOM THERM. COIL
4 CHANGE LENGTH ROOM THERM. COIL
5 MERCURY ENVELOPE ATTACHED TO COIL
6 CHANGE ANGLE φ OF MERCURY ENVELOPE
7 PERSON ADJUSTS LEVER
8 CHANGE ANGLE φ OF MERCURY ENVELOPE
9 CHANGE ANGLE φ OF MERCURY ENVELOPE
10 φ = 0 NEG
11 MERCURY ENVELOPE SLOPING LEFT
12 LEFT EDGE MERCURY NOT AT A
13 MERCURY UNSUPPORTED TO LEFT
14 GRAVITY MERCURY
15 CHANGE LOC MERCURY B→A
16 φ = 0 POS
17 MERCURY ENVELOPE SLOPING RIGHT
18 RIGHT EDGE MERCURY NOT AT D
19 MERCURY UNSUPPORTED TO RIGHT
20 GRAVITY MERCURY
21 CHANGE LOC MERCURY A→D
22 NET CHANGE LOC MERCURY
23 LEFT EDGE MERCURY AT A, GOING LEFT
24 LEFT EDGE MERCURY AT B, GOING RIGHT
25 NOT CONTACT MERCURY P1
26 LEFT EDGE MERCURY AT C, GOING LEFT
27 CONTACT MERCURY P2
28 LEFT EDGE MERCURY AT C, GOING RIGHT
29 NOT CONTACT MERCURY P2
30 LEFT EDGE MERCURY AT B, GOING LEFT
31 CONTACT MERCURY P1
32 RIGHT EDGE MERCURY AT D, GOING RIGHT
33 ROOM THERMOSTAT CLOSED
TO (A), BOX 2
THERMOSTAT
P1 P2
A B C D
φ

Display 1a.

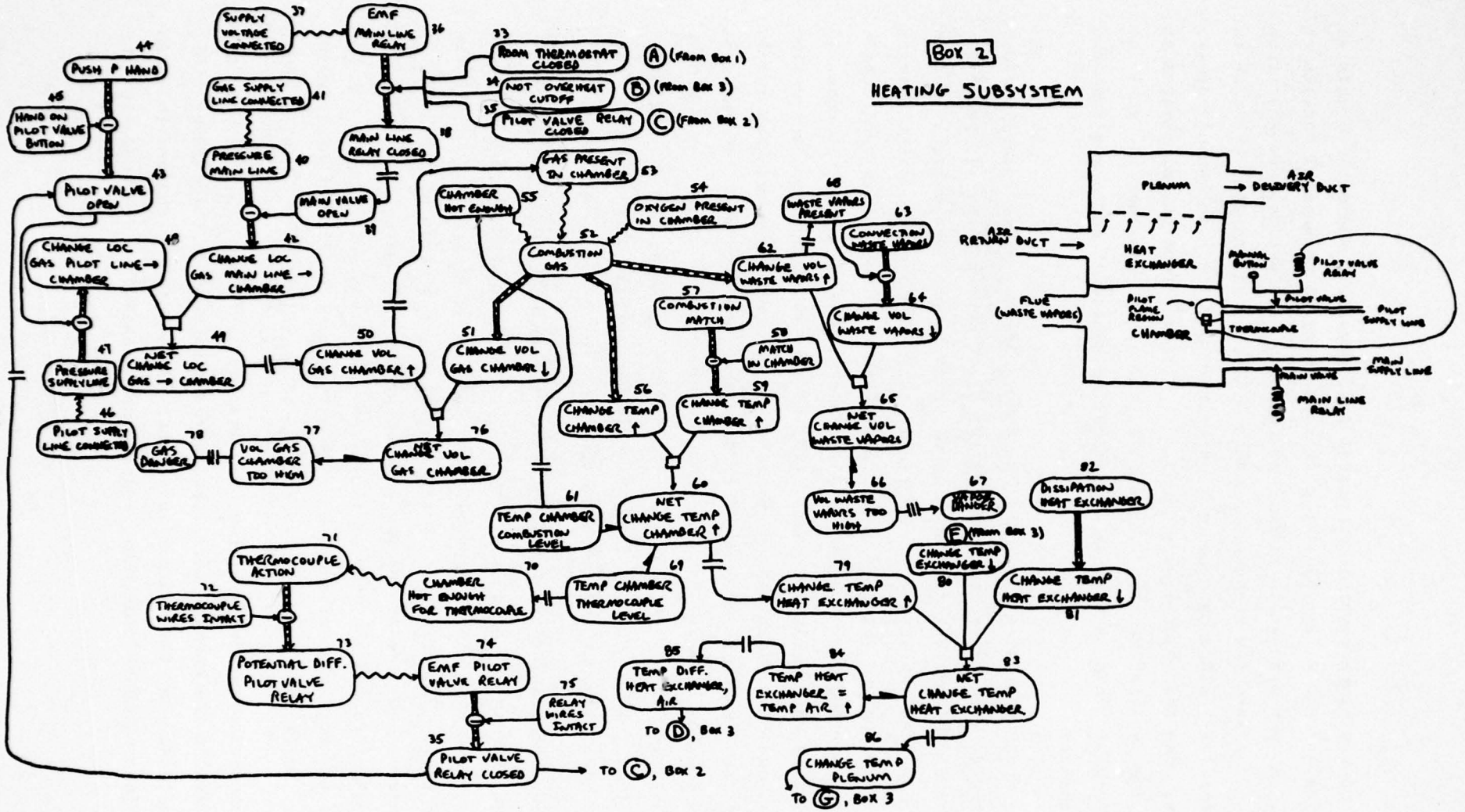
at zero going negative (10), i.e. the envelope begins tilting to the left (11), and one when it thresholds at zero going positive (16), i.e. begins tilting to the right (17). When the envelope is tilting left (11), and the left edge of the mercury is not already at the extreme left end of the envelope (12), the mercury is in a condition of being unsupported to the left (13). This allows the tendency GRAVITY (14) to manifest itself and to begin continuously changing the mercury's location toward the left (15). Returning to the other threshold point for phi (when phi crosses zero, increasing), a symmetric system of unsupportedness pertains (16,17,18,19,20,21), influencing the mercury to move toward the right. At any moment, the net change in the mercury's location (22) is governed by these two systems.

There are six points of interest (23,24,26,28,30,32) along the mercury's path of travel (lower left part of Box 1). When the left edge of the mercury reaches point A moving left (23), the mercury will cease being unsupported, and GRAVITY's influence will be severed (i.e. the mercury will stop moving left). When the mercury reaches point B moving left (30), such a condition will be synonymous with the physical contact of the mercury and electrical contact pin P1 (31). A similar structure exists at point C for pin P2 (26,27). Conversely, contacts between the mercury and P1 and P2 cease at these points when the mercury is moving from left to right (24,25,28,29). When the mercury is at the far left, it will be contacting both P1 and P2 (27,31), and these conditions, taken together, amount to the thermostat having closed the furnace's control circuit (33). This condition feeds into Box 2 via tie point A.

The Control Subsystem participates in a large feedback loop via tie point F from Box 3.

BOX 2 (Heating Subsystem)

Thermostatic control of the Heating Subsystem enters at point A (top center). When the room thermostat is closed (33), providing there is no overheat cutoff (34) and the pilot valve relay is on (35), the tendency EMF (36) will, assuming it is enabled by a connection to the supply voltage (37), continuously cause the main gas line relay to be closed (38). (Actually, what we are calling "relays" are more properly termed solenoids, since their basic


BOX 2
HEATING SUBSYSTEM
SUPPLY VOLTAGE CONNECTED
EMF MAIN LINE RELAY
ROOM THERMOSTAT CLOSED
(A) (From Box 1)
NOT OVERHEAT CUTOFF
(B) (From Box 3)
PILOT VALVE RELAY CLOSED
(C) (From Box 2)
PUSH P HAND
HAND ON PILOT VALVE BUTTON
GAS SUPPLY LINE CONNECTED
PRESSURE MAIN LINE
MAIN LINE RELAY CLOSED
MAIN VALVE OPEN
PILOT VALVE OPEN
GAS PRESENT IN CHAMBER
CHAMBER HOT ENOUGH
OXYGEN PRESENT IN CHAMBER
COMBUSTION GAS
WASTE VAPORS PRESENT
CONVECTION WASTE VAPORS
CHANGE VOL WASTE VAPORS ↑
CHANGE VOL WASTE VAPORS ↓
CHANGE LOC GAS PILOT LINE → CHAMBER
CHANGE LOC GAS MAIN LINE → CHAMBER
NET CHANGE LOC GAS → CHAMBER
PRESSURE SUPPLY LINE
PILOT SUPPLY LINE CONNECTED
GAS DANGER
VOL GAS CHAMBER TOO HIGH
NET CHANGE VOL GAS CHAMBER
CHANGE VOL GAS CHAMBER ↑
CHANGE VOL GAS CHAMBER ↓
COMBUSTION MATCH
MATCH IN CHAMBER
CHANGE TEMP CHAMBER ↑
CHANGE TEMP CHAMBER ↑
NET CHANGE VOL WASTE VAPORS
VOL WASTE VAPORS TOO HIGH
VAPOR DANGER
(E) (From Box 3)
TEMP CHAMBER COMBUSTION LEVEL
NET CHANGE TEMP CHAMBER ↑
DISSIPATION HEAT EXCHANGER
CHANGE TEMP EXCHANGER ↓
CHANGE TEMP HEAT EXCHANGER ↓
THERMOCOUPLE ACTION
CHAMBER HOT ENOUGH FOR THERMOCOUPLE
TEMP CHAMBER THERMOCOUPLE LEVEL
CHANGE TEMP HEAT EXCHANGER ↑
THERMOCOUPLE WIRES INTACT
POTENTIAL DIFF. PILOT VALVE RELAY
EMF PILOT VALVE RELAY
RELAY WIRES INTACT
TEMP DIFF. HEAT EXCHANGER, AIR
TEMP HEAT EXCHANGER = TEMP AIR ↑
NET CHANGE TEMP HEAT EXCHANGER
PILOT VALVE RELAY CLOSED
TO (C), Box 2
TO (D), Box 3
CHANGE TEMP PLENUM
TO (G), Box 3
PLENUM
AIR DELIVERY DUCT
AIR RETURN DUCT
HEAT EXCHANGER
MANUAL BUTTON
PILOT VALVE RELAY
PILOT VALVE
FLUE (WASTE VAPORS)
PILOT FLAME REGION
CHAMBER
THERMOCOUPLE
PILOT SUPPLY LINE
MAIN VALVE
MAIN SUPPLY LINE
MAIN LINE RELAY

Display 1b.

function is to translate electrical current into physical motion.) A closed relay implies an open valve (39) (we omit references to mechanical details at this point). An open main valve allows the tendency PRESSURE to move gas from the supply line into the combustion chamber (42), providing that PRESSURE is enabled in the first place by a connection to the outside gas supply (41). (Enablements, denoted by the wavy lines, are distinct from gates, denoted as arrows into the "valves" of the various links. This distinction will be discussed later.) This subsystem comprises one source of gas entry into the combustion chamber.

Another source of gas entry into the chamber is the pilot supply line. Gas will enter via this route whenever the pilot supply valve is open (43). One way to cause this valve to be open is manually to push the valve's button (44,45) (i.e. when the furnace is initially lit). The other causal mechanism for keeping the pilot valve open derives from a self-sustaining cycle involving a thermocouple system (71-75). This will be described in turn. An open pilot supply valve allows PRESSURE (47) to move gas into the combustion chamber (48), again provided that there is a connection to the main gas supply (46). These two sources of gas entry into the chamber, the pilot and main supply lines (42,48), combine into a net movement of gas into the chamber (49), and it is this net movement that will govern the rate of heat production. (The rate of heat production would be implied by the various rates of flow in a simulation of this mechanism.)

A movement of gas into the chamber can be rephrased as an increase in the volume of gas present in the chamber (50). There will be an opposing influence (COMBUSTION) which will simultaneously cause the volume of gas in the chamber to decrease (51). This is caused by the continuous combustion of the gas (52) which requires as enablements the presence of gas in the chamber (53) (implied by any buildup of gas in the chamber (50)), the presence of oxygen in the chamber (54) (a nominal condition), and the presence of sufficient heat in the chamber (55). Sufficient heat will eventually be sustained by a feedback loop.

The combustion produces three simultaneous effects (51,56,57): the decrease of gas in the chamber (51), the increase of heat in the chamber (56), and the increase in the volume of waste vapors in the chamber (62). An opposing influence, CONVECTION (63), continuously causes a decrease in the

vapors (64), provided there are any present in the first place (68). The net accumulation of vapors is influenced by these two sources (62,64), and there is some threshold (66) at which the presence of waste vapors becomes unacceptably high (67).

In addition to the production of waste material, the combustion (52) increases the temperature of the chamber. Another causal source of chamber temperature increase can be a match (57), provided it is burning and in the chamber (58). (This, in conjunction with depressing the pilot valve, indicate how to "initialize" the mechanism.) The net rate of temperature increase (60) causes some threshold eventually to be reached (61) wherein the combustion cycle becomes self-sustaining (52-56-60-61-55-52). (This will amount to the pilot flame starting to exist.)

At some other threshold, the chamber's temperature will be sufficient (70) to begin enabling the tendency THERMOCOUPLE ACTION (71). (There is a thermocouple in the pilot flame's region which "senses" whether or not the pilot flame is present.) Provided the thermocouple wires are intact, this tendency will continuously cause a potential difference across the pilot valve relay (73), enabling EMF (74) to maintain the relay in an "on" state (35) (providing the relay wires are intact (75)). The pilot relay being on (35) implies that the pilot gas valve is open (43). Thus, there is a large, self-sustaining cycle which, after the thermocouple is hot enough, keeps the pilot valve open (43, 48, 49, 53, 52, 56, 60, 69, 70, 71, 73, 74, 76, 43). (This is to preclude the leakage of pilot gas when there is no combustion to consume it.) Also, because of the connection of (35) as a gating condition on the closing of the main gas relay (38), unless the pilot light is on, there is no possibility of the main line being opened (causing an even more rapid buildup of unburnt gas).

If combustion is occurring, there will be a net temperature increase in the combustion chamber (60) at a rate governed by the net rate of gas flow into the chamber. Because of the layout of the furnace, this implies an increase in temperature of the heat exchanger (79). There are two other influences (80,81) which will cause a decrease in the exchanger's temperature: DISSIPATION (81,82) (a general leakage), and HEAT EXCHANGE with the cooler air (Box 3) which will be passing through the heat exchanger (80). These three

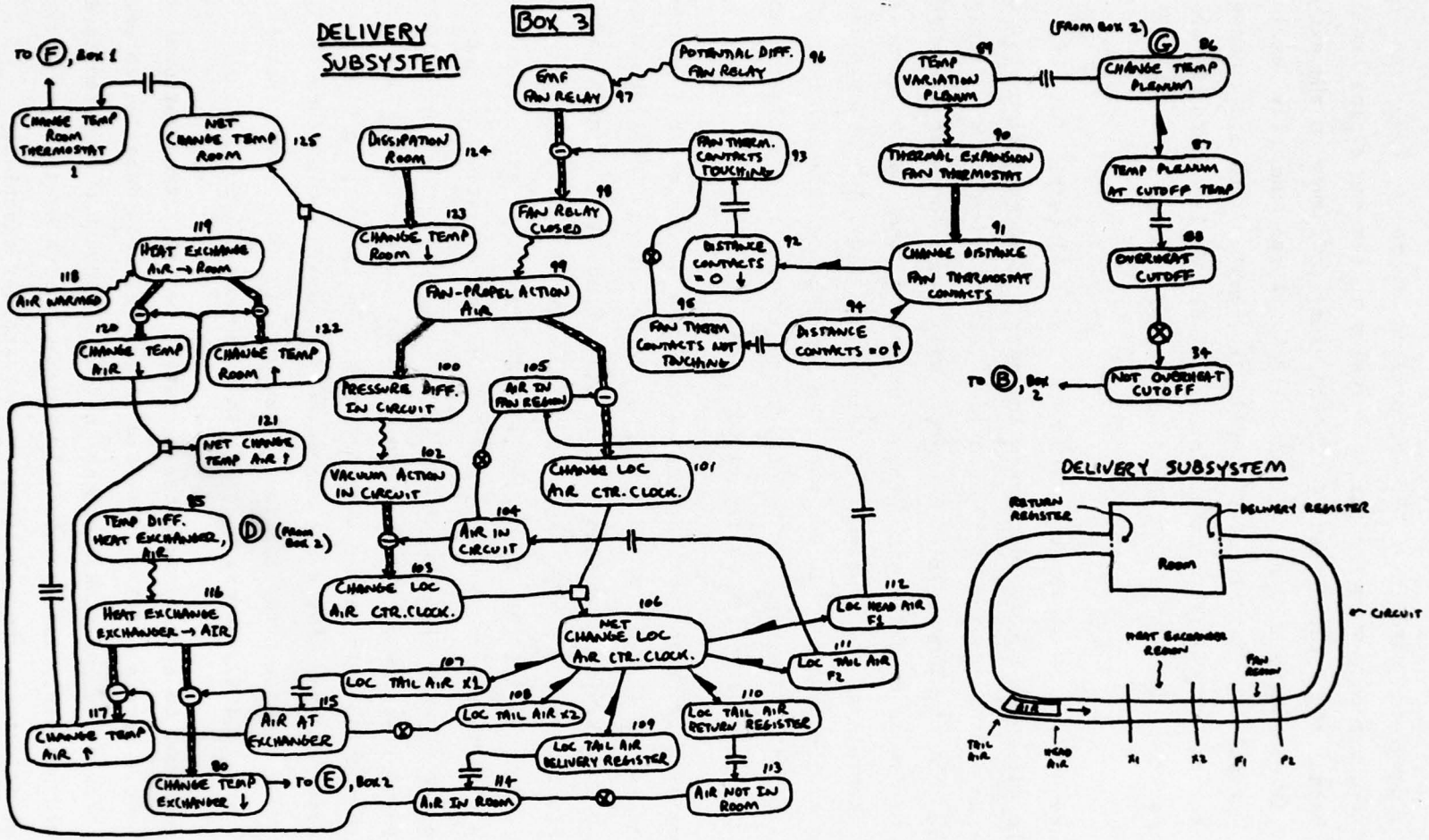
influences combine into a net temperature change in the exchanger (83). At some (early) point in the exchanger's heatup, its temperature will exceed the ambient temperature of the air (84), giving rise to a state of temperature difference between the air and exchanger (85). This will allow the tendency HEAT TRANSFER in Box 3 to warm the air as it passes through the exchanger region. The exchanger's heat buildup will also be reflected in a heat increase of the plenum (86) (the large bonnet over the furnace) where the fan and overheat sensors are located. This sets the stage for the activities of the Delivery Subsystem in Box 3.

BOX 3 (Delivery Subsystem)

The Delivery Subsystem is triggered by a rising temperature in the plenum (86). If this temperature ever reaches some abnormally high value (87), this will amount to a condition of overheat cutoff (88) (via a thermostat which we have not described in the graph). A condition of overheat cutoff will interfere with the operation of the main gas supply line valve (34) causing the valve to shut and the system to cool down.

As long as this overheat cutoff does not occur, the temperature variation (either a rise or fall) in the plenum (89) continuously enables the tendency THERMAL EXPANSION (90) to convert temperature variation into distance variation between the fan thermostat contacts (91). (Here we slip into a more detailed description of the fan cut-on mechanism.) When the distance between these thermostat contacts reaches zero on the decrease (92), this amounts to the contacts touching (93), and this allows EMF (97) to cause the fan relay to close (98). Alternatively, when the contacts are at distance zero on the increase, this amounts to a termination of the contacts' touching condition, removing the gating condition on the EMF causality and causing the fan relay to open (94,95,93).

The fan relay's state of being closed will enable the tendency FAN PROPEL ACTION (99). This tendency causes two conditions: A pressure difference in the ductwork circuit (100), and a propelling of air in the counterclockwise direction around the circuit (101). (We view the air as a chunk which moves around as a unit in the ductwork, rather than as a continuous flow. The understanding of the mechanism is facilitated by focussing on one unit of air


DELIVERY SUBSYSTEM
BOX 3
TO (F), Box 1
CHANGE TEMP ROOM THERMOSTAT
1
NET CHANGE TEMP ROOM
125
DISSIPATION ROOM
124
CHANGE TEMP ROOM
123
EMF FAN RELAY
97
POTENTIAL DIFF. FAN RELAY
96
FAN RELAY CLOSED
98
FAN THERM. CONTACTS TOUCHING
93
DISTANCE CONTACTS = 0
92
FAN THERM CONTACTS NOT TOUCHING
95
DISTANCE CONTACTS = 0
94
CHANGE DISTANCE FAN THERMOSTAT CONTACTS
91
THERMAL EXPANSION FAN THERMOSTAT
90
TEMP VARIATION PLENUM
89
(From Box 2) G
CHANGE TEMP PLENUM
86
TEMP PLENUM AT CUTOFF TEMP
87
OVERHEAT CUTOFF
88
NOT OVERHEAT CUTOFF
84
TO (B), Box 2
HEAT EXCHANGE AIR → ROOM
119
AIR WARMED
118
CHANGE TEMP AIR ↓
120
CHANGE TEMP ROOM ↑
122
FAN-PROPEL ACTION AIR
99
PRESSURE DIFF. IN CIRCUIT
100
AIR IN FAN REGION
105
NET CHANGE TEMP AIR ?
121
VACUUM ACTION IN CIRCUIT
102
CHANGE LOC AIR CTR. CLOCK.
101
TEMP DIFF. HEAT EXCHANGER, AIR
85
(D) (From Box 2)
AIR IN CIRCUIT
104
CHANGE LOC AIR CTR. CLOCK.
103
HEAT EXCHANGE EXCHANGER → AIR
116
NET CHANGE LOC AIR CTR. CLOCK.
106
LOC HEAD AIR F1
112
LOC TAIL AIR X1
107
LOC TAIL AIR X2
108
LOC TAIL AIR F2
111
LOC TAIL AIR RETURN REGISTER
110
LOC TAIL AIR DELIVERY REGISTER
109
AIR AT EXCHANGER
115
CHANGE TEMP AIR ↑
117
CHANGE TEMP EXCHANGER ↓
80
TO (E), Box 2
AIR IN ROOM
114
AIR NOT IN ROOM
113
DELIVERY SUBSYSTEM
RETURN REGISTER
DELIVERY REGISTER
Room
CIRCUIT
HEAT EXCHANGER REGION
FAN REGION
AIR
TAIL AIR
HEAD AIR
X1
X2
F1
F2

Display 1c.

as it makes a complete circuit.) The propelling tendency (99) can exert influence on the air only when the air is actually in the fan region (105). On the other hand, the vacuum action (102) can exert influence on the air only when the air is elsewhere in the circuit (104). These mutually exclusive sources of propulsion for the air (103,101) combine into a net change of location of the air around the ductwork (106). (Our whole approach to the Delivery Subsystem represents a discrete approximation to a continuous process.)

There are six points of interest around the duct circuit. Two of these are when the air chunk enters and leaves the heat exchanger region (107,108). When the air is in this region (115), provided the exchanger is sufficiently heated (85), the tendency HEAT EXCHANGE will cause the heat exchanger to impart some of its heat to the air, thus increasing the air's temperature (117), and decreasing the exchanger's temperature (80). Two other points of interest around the duct circuit are when the air chunk enters and leaves the fan region (111,112). At these two points, there is a switchover in the source of the air's propulsion system as described above (101,102).

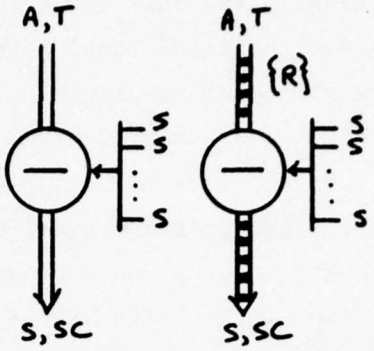
The remaining two points of interest around the duct circuit are when the air chunk enters and leaves the room which is to be warmed (109,110). When the air is in the room region (governed by 113, 114), provided the air has been warmed (118), the tendency HEAT EXCHANGE will cause some of the air's heat to be imparted to the room (119), resulting in a drop in the air's temperature (120), and a rise in the room's temperature (122). (There will be a net change in the air's temperature (121) which is of no direct consequence.) As the room is being heated (122), there is an opposing force of DISSIPATION (124) to the outside which continuously causes a decrease in room temperature (123). These two influences combine for a net change in the room's temperature (125). A change in room temperature implies a change in the room thermostat's temperature, and, VOILA!, we are back to the beginning (1, Box 1).

If we were now to specify particular rates in all the various state changes, we would have a model which would be sufficiently detailed to perform an actual simulation of the furnace in time. Although our system could do this, it would be fairly incomprehensible because of its sheer size. Later in the paper, we illustrate the simulator on some smaller examples.

2.3. The Cause-Effect CSA Links

We now present and discuss each link.

2.3.1. CAUSE



This is the primary cause-effect CSA link. It is employed wherever we wish to assert a conceptually direct cause-effect relationship between two events in the mechanism. The caused event is either a state or a statechange; the causing event is either an action or a tendency. These requirements define the CAUSE link's syntax. There are three versions of this link, each reflecting a different type of causality with respect to time. The ONESHOT version (denoted by the symbol OS-CAUSE in the implementation) predicates that the one-time existence of the causing action or tendency (we do not currently define how long it must exist) causes some state or statechange to come into, and remain in, existence until something else causes it to cease to exist. Thus for example, flipping a light switch is a one-time action which causes the switch to enter and remain in the state to which it was flipped; pushing a mass in a gravityless environment causes the mass to enter a condition of continuously changing location until otherwise acted upon.

The second version of the CAUSE link is CONTINUOUS CAUSE (denoted by the symbol C-CAUSE in the implementation). This version also allows either a state or statechange as the caused event, and predicates that the action or tendency is continuously required to maintain the existence of the state or statechange. Removing the action or tendency causes the state or statechange to cease. Examples are: holding an object continuously causes the object to be supported (i.e. the holding is continuously responsible for the force which counteracts gravity); running a pump continuously causes a statechange in the location of some fluid.

The third version of the CAUSE link is the REPETITIVE CAUSE (denoted R-CAUSE in the implementation). It accepts either a state or statechange as the caused event, asserting that the causing action or tendency, being repeatedly performed (again, we have not to date dealt with frequency specifications), causes the state or statechange to remain in existence. Examples are: hitting a stake with a hammer R-CAUSEs the stake's depth in the ground to increase; repeatedly retriggering a retriggerable monostable flip-flop preserves the state "flip-flop is on", and so forth.

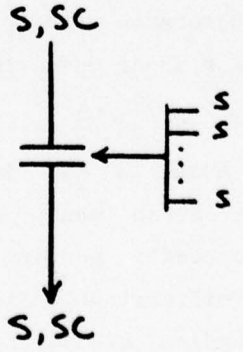
2.3.1.1. Gating

Associated with all forms of the CAUSE link (and with some of the other links) is the important notion of gating. The concept of causality is that some condition results from some event "when all other conditions in the environment are right". It is this last phrase which identifies the concept of a gate. Associated with each CAUSE link, in addition to the causing and caused events, is a set of so-called GATING states, attached graphically to the left or right of the "valve" in the causal link. The semantics of this set is that each of the conditions (states) it specifies must be true in order for "causality to flow" from causing event to caused event. Notice that this set has nothing to do with the existence of the causing event itself; that event may quite well be proceeding, yet not producing any results because of some missing gating condition (e.g. the car's engine is running, but the car is not moving because the gears are not engaged; or the robot is grasping, yet not succeeding in picking up some object because his fingers were never around the desired object). Thus, gates help us segregate the conditions which are requisite to the causing action or tendency's existence from the conditions which govern the influence such an action has on its environment.

Except for a few details, we feel these three versions of CAUSE link, together with the concept of gating, provide a reasonably complete basis for describing "direct" cause and effect. We use sneer quotes here because direct is obviously a relative term: what one person believes or chooses to regard as direct may very well not be what another regards as direct.

2.3.2. STATECOUPLE

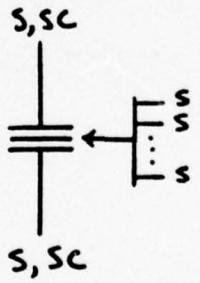
There are times when we know there is some cause-effect relationship between two events, and we know that it is not "direct", but we do know know the intervening CAUSE structure. Alternatively, there are times when we wish to be sketchy in part of our representation and leave large hunks of it out, even though we know better. For both of these purposes, we define



the STATECOUPLE link (denoted in the implementation by S-COUPLE). Its syntax is always: STATE or STATECHANGE couples to (implies) STATE or STATECHANGE. Since using this link will frequently amount to a kind of "buffering" between alternate descriptions of the same underlying condition (using different points of view which are causally related), we use a symbol which resembles a capacitor symbol. Semantically, the link tells us that the "causing state's" (states cannot actually cause anything) existence implies the caused state's existence, but the caused state need not be exclusively dependent upon the "causing state". Therefore, removing the "causing state" does not imply that the caused state will cease; some other causality will have to occur in order to disrupt the condition.

Abstractly, the S-COUPLE and CAUSE links express the same underlying concept. In theory, we might therefore wish to have three versions of S-COUPLE also. However, we have had no need to do so; perhaps this is because when S-COUPLE is used, it is describing a situation which is defocussed enough (at a coarse enough level of resolution) that a discrimination of the time aspect (OS, C, R) would be irrelevant. Alternatively, maybe we simply have not considered the right mechanisms yet, and in fact there should be three S-COUPLE versions. In either case, for the time being, we define only one variety. We do, however, require the notion of gating in state coupling.

2.3.3. EQUIVALENCE

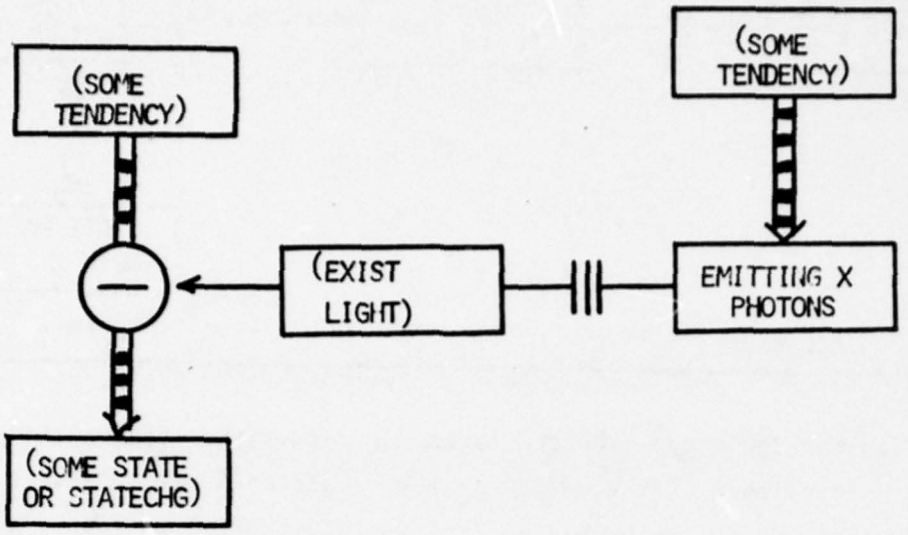


It will frequently happen that, even in the most consistent style of event representation, there are two legitimate yet symbol-wise distinct ways to represent the same piece of information. For example, if we connect PIN-1 of integrated circuit X to PIN-2 of integrated circuit Y, certain electrical conditions at PIN-1 (e.g. current and voltage) become synonymous with the conditions at PIN-2. Or if we connect PART-A to PART-B with a rigid rod, certain mechanical motions (e.g. lateral displacement) of PART-A and PART-B become synonymous (providing the linkage remains intact). In either case, the notion is less of causality than it is a definition of the physical layout of the mechanism.

Providing the ability to describe a situation from two aspects which are equivalent is one of the three conceptual uses of the EQUIVALENCE link (denoted S-EQUIV in the implementation), written graphically with three parallel bars.

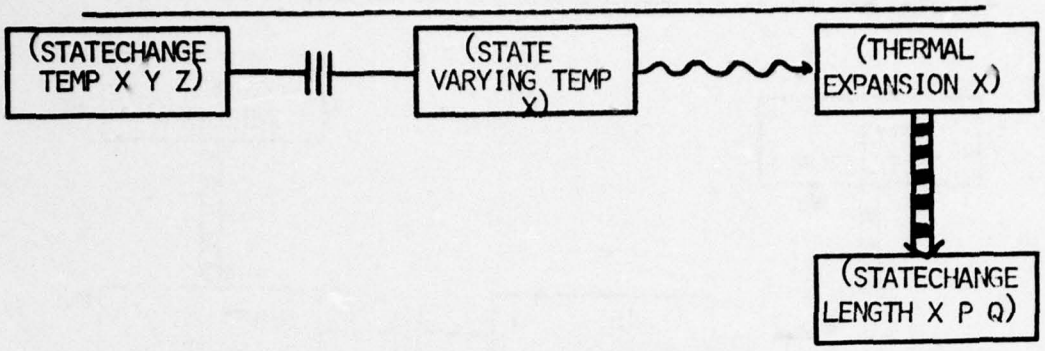
The second use of the equivalence link, and simply another way of looking at the same underlying notion from a different point of view, is as a "domain shifter". For example, for one purpose we might wish to talk about the emission of photons from a body, yet for another purpose we might wish to describe an event in terms of the existence of light. Clearly, existence of photons (at the right energy level) is simply another way of saying that light exists. Yet, thinking in terms of a symbolic pattern matcher given the two patterns (LUMINOUS X) and (EMITTING X PHOTONS), it will be essential that the definitional equivalence of these two patterns which express the same idea be made explicit (either directly in the description, or in the abstract database of eqivalence knowledge). Without such explicitness, it would be impossible to match, and hence we would not be able to play this kind of convenient "symbol switching" game.

Symbol switching will be essential when we get into the business of mechanism invention and become interested in coupling in the outputs of one mechanism as the inputs of another. For, suppose we have a mechanism that requires light to shine on some component (say as a gating condition), and suppose we have a mechanism whose final product is (EMITTING X PHOTONS). In order to stitch the two together symbolically we need to massage the output of one into the shape required as input to the other, as shown in Display 2.



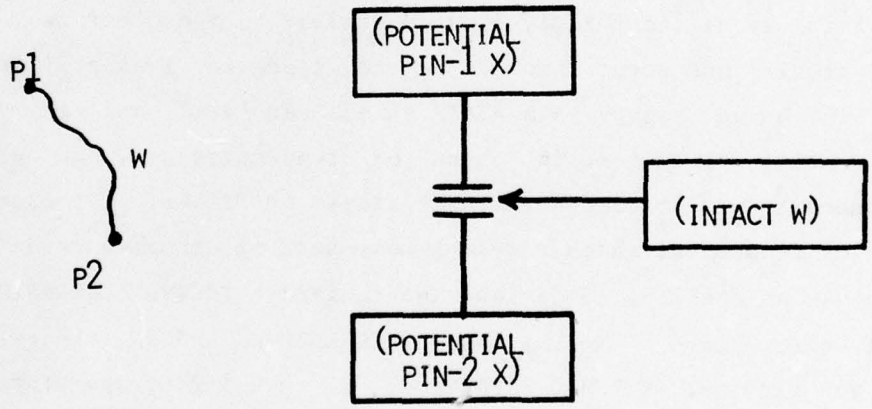
Display 2. Pattern massaging via the EQUIV link.

The third use of the S-EQUIV link is similar to the other two, but allows us to characterize one event which we regard at one level as a *STATECHANGE* as an event which we regard as a STATE at another level, and vice-versa. This idea is essential for a certain class of tendencies if, as we believe, enabling conditions for tendencies must always be STATES. For example, there is a class of tendencies which requires some sort of changing condition in the environment as an enabling condition. The tendency THERMAL-EXPANSION is one member of this class. In the case of THERMAL-EXPANSION, the pattern reads like: "If you give me, THREMAL-EXPANSION, a changing temperature, I will produce for you a corresponding change in the dimensions of an object." Of course, we could always omit the reference to this tendency and write the pattern "changing temperature implies changing dimensions" with the STATECOUPLE link, but this would bypass reference to THERMAL-EXPANSION, something we might wish to identify explicitly. This use of the S-EQUIV link will occur in contexts illustrated in Display 3.



Display 3. Statechange-state resolution level shifting.

As with the CAUSE and S-COUPLE links, we permit the attachment of gates to the equivalence link. The gating conditions will always reflect definitional states in the mechanism, things like "WIRE-1 is intact", "SHAFT-1 connected", etc. as shown Display 4.

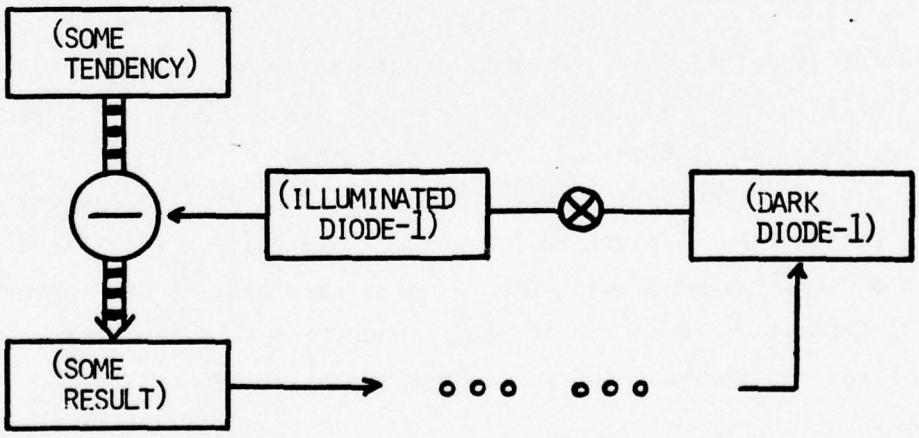


Display 4. Gated EQUIV states

2.3.4. ANTAGONISM



Any logic must incorporate some notion of negation, or non-existence of a condition. In the Mechanisms Lab, negation is modeled by a link called the antagonism link (ANTAG in the implementation). This link is used to denote that two states are mutually exclusive - that the existence of one implies the non-existence of the other, and vice-versa. It is therefore similar to the S-EQUIV link just described. As with that link, the ANTAG link's primary application is "pattern massaging", which relates one formulation of a state, X, with another formulation which is the negation of that state, not necessarily expressed as (NOT X). This permits us, for example, to describe an absence of light condition as (DARK <location>) in one part of a mechanism, and as (ILLUMINATED <location>) in another part of the mechanism, then relate the two as mutually antagonistic conditions. Typically, the ANTAG link will be the focal point of feedback loops underlying oscillation patterns, and in this role will connect to enabling or gating states as illustrated below.

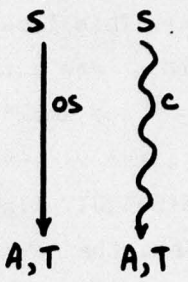


Display 5. The ANTAG link in feedback loops

Although it would be natural to include a negative counterpart of the S-COUPLE link (state X implies the non-existence of state Y, but state Y does not necessarily imply the non-existence of state X), we have as yet found no

use for such a link.

2.3.5. ENABLEMENT



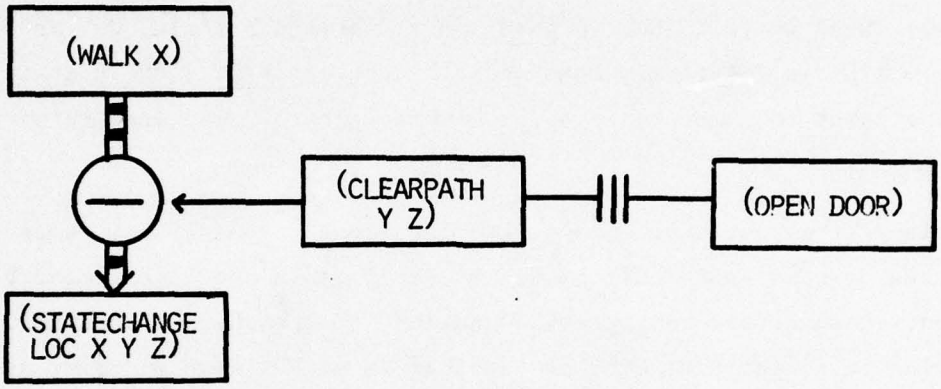
CAUSE links have gates which specify when an ongoing action or tendency will exert an influence. However, there are other conditions which govern whether or not the action or tendency can begin or proceed in the first place. These conditions will always be states and are designated as the action or tendency's enabling conditions. We relate each enabling condition of an action or tendency to it via the ENABLEMENT link (denoted by the symbol C-ENABLE in the implementation), whose syntax prescribes a STATE sustaining (enabling) an action or tendency. Semantically, an action cannot proceed until all its enabling states exist simultaneously.

Examples of the C-ENABLE concept are: "the power supply must be connected to NAND-GATE-1 in order for it to serve as an active element in the circuit"; "there must be a reservoir of water in the water tank in order for the tendency PRESSURE to exist".

In general (and this is a comment about all the links), we include explicitly only those enabling events which are peculiar or of particular interest to the description of a given mechanism. Any other generic enablements for an action or tendency are always implied. Although the current simulator takes into account only events explicitly specified in the description of a mechanism it has been asked to simulate, we are beginning to build an interface between the simulator and the CSA system's more abstract knowledge of various actions' and tendencies' enabling conditions.

The semantics of the C-ENABLE link are that the enabling state be continuously present. Whenever an enablement is severed (i.e. the enabling state ceases to exist), the action or tendency will also cease. For this reason, we think of the C-ENABLE link as a continuous requirement.**

** General CSA utilizes another version of enablement, OS-ENABLE, standing for "one-shot enablement". This link predicates that the existence of the OS enabling condition is required only momentarily to trigger the action or tendency. This link seems always to occur in situations where the action or tendency, once triggered, inherently alters its environment in a way which makes its continuation no longer dependent upon the initial enabling condition. For example, a one-time enablement for actor A to perform the "action" LEAVE-A-ROOM is that the door to the outside be open; however, once A has passed the door, even though A is performing the same "action", the state of the door is no longer of consequence. The "one-shottedness" of this relations results from factoring a context-dependent element into the definition of the action. It seems always to be possible to rephrase this type of OS enablement in more general terms using continuous enablement. For example, we could instead talk in terms of the more generic action WALK, whose continuous enablement is that the path be clear. Then, the details of any instance of an enabling condition for a WALK action could be tied to the generic description via S-EQUIV links:



Therefore, although OS-ENABLE is a useful descriptive tool, it, like one-shot causality, is retained more for descriptive convenience than for theoretical reasons. The Mechanisms Lab uses only the continuous version.

The most common use of the C-ENABLE concept, at least in the Mechanisms Lab, is in indicating the preconditions for tendencies. We have defined a tendency to be an actor-like event that produces a force, but not a true action because there is no element of choice to act or not to act involved. Tendencies will always model commonsense laws of nature such as electromotive force, incandescent radiation, thermal conductivity, and pressure. Often, it will happen that what we denote as a tendency could in reality be elaborated into an entire mechanism-like pattern itself. This is not a violation of our principles; in fact, to the contrary, it is quite in keeping with our goal to keep the representation resolution-wise flexible.

2.3.6. THRESHOLD

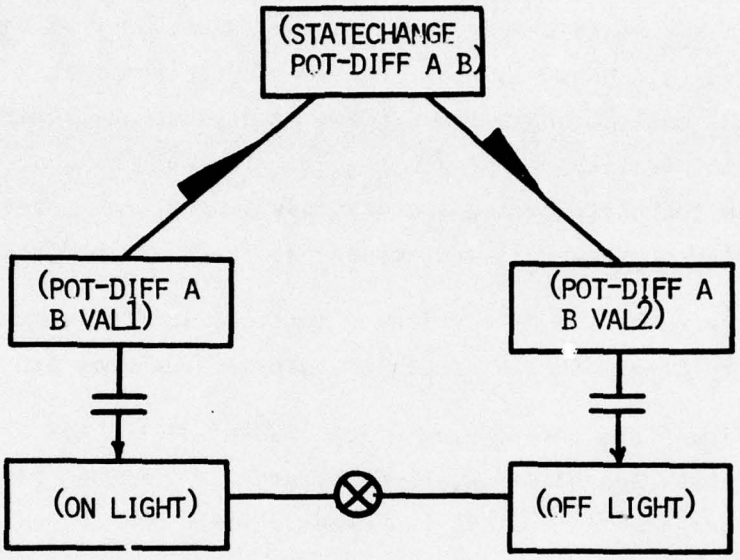
CSA defines a statechange to be a change in value of some property of some object along some continuous scale. Examples of continuous scales are temperature, distance, light intensity, and voltage level. Components of mechanisms can frequently be characterized as existing at points on various continuous scales: (TEMPERATURE FILAMENT-1 300-DEG-F), or as undergoing changes on these scales: SC: (TEMPERATURE FILAMENT-1 <from> <to> <rate>). When we talk about a point on a continuous scale, we are referring to a STATE in CSA terms; when we talk about a shift along a scale from some starting point to some ending point with some rate, we are referring to a STATECHANGE.



Since it is quite often a particular level (point) of some mechanism component on some scale that is of interest, and since levels must be attained by continuous statechanges, the notion of reaching a threshold from some direction (e.g. increasing or decreasing) is a vital one wherever statechanges are involved. For this purpose, we define the THRESHOLD link (denoted THRESH in the implementation) whose syntax requires a statechange on its tail and a state (level) on its head. We graphically distinguish two versions: one for a threshold reached by an increase, and one for a threshold reached by a decrease. (In the simulator, this difference is represented in the description of the threshold state by including a POS or NEG marking in the state description itself).

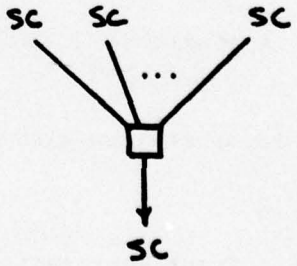
Typically, a state which has been reached as a threshold will begin to enable some tendency or contribute to some CAUSE, S-COUPLE or S-EQUIV gate. The semantics of THRESH are that, providing the threshold state has been reached from the proper direction, whatever conditions become enabled or gated as the result of reaching the threshold remain enabled or gated until some other part of the mechanism explicitly shuts them off. This means, for instance, that simply recrossing the threshold from the other direction does not imply that the activities set in motion by the original threshold will cease. In cases where they ought to cease, there must be another explicit

THRESH link of opposite polarity to specify the shutoff threshold. A simple example of this would be a neon lamp which lights when the voltage crosses some threshold, VAL1, on the increase, but which shuts down at a different (and lower) threshold, VAL2, when the voltage is on the decrease, as illustrated in Display 6.



Display 6. Thresholds and hysteresis.

2.3.7. RATE CONFLUENCE



The net statechange of a mechanism component with respect to some scale will often be a product of numerous influences of different causal origins. If the scale is one-dimensional (as are most of the scales we have employed in the Mechanisms Lab so far), the net statechange will be some composition of all positive and negative influences. Since, conceptually, the causal sources of each contributing statechange might be quite distinct, we felt it desirable to keep the description of the net statechange modular. The result is the so-called

RATE CONFLUENCE link (denoted by RATE-CONFL in the implementation). Its syntax prescribes a set of contributing statechanges, all referencing the same scale and object, which result in a net statechange of the object on that scale. The rate of the net statechange is derived by an arbitrary computation involving other simulation parameters, typically the rates of the contributing statechange events.

Semantically, wherever there is a rate confluence, any thresholds must be attached to the net statechange event, rather than to any of the contributing statechange events. This is for the obvious reason that a rate confluence represents not a collection of independent events but an itemized description of what is in reality a single event. It would make no sense to derive thresholds from the contributing statechanges, since no one of them will adequately reflect the overall statechange at any given moment.

Statechange rates are of particular interest in the simulator; we will describe in more detail in the simulation section how they are handled.

This concludes the description of the subset of the CSA links used within the Mechanisms Lab. Appendices A, B, C, D, and E contain more examples of mechanisms represented in the CSA language as follows:

Appendix A: Computer Flip Flop (2 versions)

Appendix B: Incandescent Lightbulb

Appendix C: Drinking Duck

Appendix D: Bernoulli's Train Wreck

Appendix E: Bicycle Horn

These appendices also contain computer trace output of the Mechanisms Simulator.

2.4. Event Representation

We have described the CSA event connecting syntax without considering how the events themselves are represented. Since an event is in general an arbitrary condition in the world, to develop a complete specification for

event representation would be tantamount to solving the "AI representation problem". Although we have made no strong theoretical commitments in this area, there are some principles by which we try to abide, and the simulator does require syntactically well-formed events in certain cases. So we devote a brief section to some of these points.

Since the simulator relies on a totally symbolic pattern matcher at its lowest level, it has no preconceptions about what are legal or illegal event-describing predicates. In practice, we take advantage of this lack of intelligence by representing states by English-like phrases which are easy for humans to interpret. However, in principle, and as the pattern matcher becomes more sophisticated, we will shift to a more formal event description syntax such as is about to be described. We emphasize, however, that (except for statechange forms) there is presently no need to adhere to a formal event representation syntax.

2.4.1. States

States are described by a 3-, 4- or 5-tuple of the form:

(STATE <attribute> <object> { <value> { <direction> } })

Display 7.

---

where <direction> applies only to states which describe points on continuous scales as discussed earlier. For this type of state, the direction specifies the direction from which the point was reached. Examples of states are thus:

(STATE TEMPERATURE FILAMENT-1 300-DEG-F POS)
(STATE DISTANCE (PIN-1 SHAFT-1) 3-CM NEG)

Display 8.

2.4.2. Statechanges and Descriptors

Statechange events are represented by 6-tuples of the form:

(CHANGE <scale> <obj> <start> <stop> <rate>)

Display 9.

Such a form denotes that object <obj> undergoes a change with respect to <scale> from beginning point <start> to ending point <stop> with rate <rate>. The start point, stop point and rate are ultimately numeric. However, it will more often than not happen that absolute numbers are either unknown at description time, or irrelevant. In either case, it is essential that symbolic references to points on scales and rates be tolerated both in the description syntax and by the simulator. Hence, the description syntax incorporates the notion of a descriptor.

A descriptor is a symbolic reference to some entity whose actual identity (e.g. numeric value) is not known, or is contextually dependent upon other factors not yet known. A descriptor has the form:

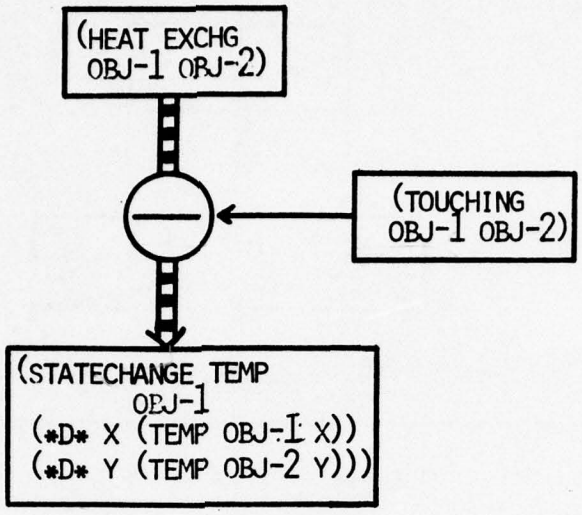
(*D* <var> <pattern-containing-var> { . <specific-alt-list> })

Display 10.

---

with the semantics being that the entire descriptor describes some entity by defining a pattern which the entity must satisfy. (The <specific-alt-list> is a possibly empty list of specific suggestions as to the identity of the descriptor. Although this is of considerable importance to the language comprehension reference mechanism and to parts of the CSA problem solver, we have not used it in the Mechanisms Lab.)

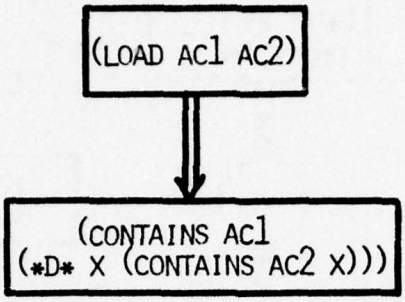
To illustrate the use of descriptors, suppose we wish to indicate that after a certain amount of time in contact, OBJECT-1 acquires the same temperature as OBJECT-2. Then we would write a pattern of the form:



Display 11. Using descriptors

meaning "Provided OBJECT-1 and OBJECT-2 are in contact, the tendency HEAT-TRANSFER will cause a statechange in OBJECT-1's temperature from starting point X, such that X is OBJECT-1's current temperature, to Y, which is OBJECT-2's temperature, with some rate which we have not specified in the example. (Of course, as a physical phenomenon, this pattern is not correct, since, actually, both objects' temperatures would be affected; we use this pattern only to illustrate descriptors.)

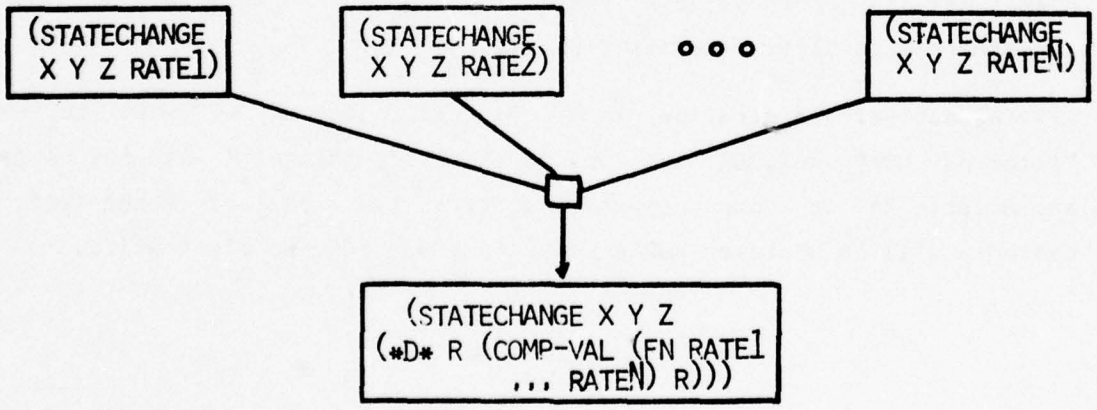
As another illustration of descriptors, suppose we want to express: "After a LOAD AC1,AC2 instruction (load accumulator 1 with the contents of accumulator 2) in some computer program (an abstract mechanism), AC1's contents will be whatever AC2's contents are." Then we might write:



Display 12.

i.e., the action LOAD causes AC1's contents to become X, such that X is the current contents of AC2.

Statechange rate computations can be handled similarly by descriptors. Suppose that in a rate confluence, the net rate needs to be expressed as the algebraic sum of the rates of the contributing statechanges (which may in turn have rates expressed symbolically in terms of other factors in the mechanism). Then, conceptually, we will write:



Display 13. Rate confluence descriptors.

We say "conceptually" because the details of how the current simulator implements rate descriptors varies sightly in form from this account. (Grinberg will describe this later.)

From the simulator's point of view, descriptors will look like Skolem functions which must be "evaluated" (i.e. converted from symbolic references to actual values) during the simulation. Descriptors will be detected and evaluated at specific times during the simulation as described in a later section.

2.4.3. Actions and Tendencies

Actions and tendencies are represented internally by the respective forms:

(ACTION <action> <actor> <arg1> ... <argn>)

and

(TENDENCY <name> <with-respect-to>)

Display 14.

The ACTION form specifies the action, the actor, and the conceptual cases required by that action. Each action requires its own individualized case framework.

Tendencies must always occur with respect to some object: GRAVITY can be regarded abstractly as an actor, but any specific occurrence of gravity playing the role of actor will be with respect to some object. Thus, the form of tendencies includes the object being influenced, e.g.:

(TENDENCY ELECTRICAL-RESISTANCE FILAMENT-1)
(TENDENCY GRAVITY LEVER-1)
(TENDENCY HEAT-TRANSFER FACEPLATE-1)

Display 15. Some tendencies.

2.5. Uniqueness

A desirable characteristic of any representation is that it help focus the way its users view the world (focus, not legislate!). The limiting case is where the representation naturally expresses conceptually similar mechanisms in similar internal structures, perhaps to the point where two real-world mechanisms are "causally isomorphic" if and only if their representations are isomorphic. While such a representation would have its obvious merits (e.g. for recognizing common principles across mechanisms from all walks of science), such a virtue seems to be incompatible with the other important characteristic of flexibility (e.g. each of us has his own view of cause and effect).

In what sense can it be said that the CSA representation focuses similar mechanisms onto similar descriptions? It can be said, with near certainty, that any two people will represent any given mechanism by CSA patterns which are not structurally isomorphic; they will almost certainly, at a minimum, group gating or enabling conditions differently, for example. Since we have been concentrating on the flexibility issue, we are neither surprised nor alarmed by this. In fact, we feel it is impossible in principle for any representation in a cause-effect domain to enforce uniqueness.

However, all is not lost. After a preliminary survey of the classes of disagreement between alternate representations of some of the mechanisms we have considered, it seems possible to begin defining a set of syntactic transformations which preserve meaning. If such is the case, then it will be possible to develop equivalence classes of mechanism patterns, where an equivalence class consists of all alternate formulations of a given abstract mechanism pattern.

Part of our future effort will be devoted to defining such transformations. Discovering such a set will be akin to writing primitive classes of inference on mechanism structures, and it might lead to some canonical forms which could then serve as a unique (or nearly so) mechanism representation devoid of human stylistic variations. However, we do not presently have any basis for predicting the success of this approach.

3. Spontaneous Computation and Mechanism Simulation

We have described the representation part of our theory. We now turn to the topic of simulation, and begin with a description of a model of associative memory.

3.1. Simulation Strategy

Since the Mechanisms Lab is embedded in the larger framework of a problem solving and language comprehension system, we have been intent on deriving the theory of the simulator from parts of the existing larger theory. The motive of course is to develop a coherent model of intelligence wherein one process, regardless of its end product, draws upon a pool of general theoretical tools as the means to its end. For the problem solver, the goal is to generate a context-sensitive plan to achieve some goal; for the language comprehension parts of the model, the goal is to arrive at some interpretation of a thought in context; for the mechanisms simulator, the goal is to "run through a mental simulation" of a plan-like entity which describes a physical or abstract mechanism. We want them all to interface and to draw upon the same basic theory, because they all will eventually interact in a full model of intelligence.

Given that we wanted to integrate the Mechanisms Simulator into an existing framework, the obvious part of the existing CSA theory for the simulator was the spontaneous computation subsystem. This is the component of the system which is associative, in that it causes computations to be invoked or aborted not because some other part of the model has requested so, but because a computation's "trigger pattern" has matched some condition in the system. The SC subsystem therefore is a very loosely organized population of associatively-triggerable specialists who perform computations whenever the circumstances for which they are applicable appear.

Because of the inherent modularity of the CSA SC system (e.g. computations can be masked, unmasked, deleted, created and split. independently since they are not linked together in any preconceived calling hierarchy), an SC environment is a natural one for the simulation of a mechanism. A mechanism consists of a number of modular parts and principles

which have been brought together to interact in some globally meaningful way. Alternatively, the components of a mechanism can be individually disabled, enabled, altered, combined, masked, to produce new mechanisms, or to gain insights into how the global picture would be altered if one of the components were altered or deleted; so too can SC's.

Because of this obvious similarity, we adopted the following overall strategy for simulation:

(1) Accept as input a declarative (CSA graph) representation of the mechanism, such as we have been illustrating.

(2) Convert each event (node) into a spontaneous computation which will run whenever the preconditions for the event in the graph it represents become true. (Preconditions will mean enabling or gating states, flow of causality, and so forth.) At that point, the mechanism will have a procedural representation which is the dual of the original declarative one.

(3) Assert any nominal starting *conditions* which have been declared in the mechanism's description at input time (i.e. assert such conditions in the central database which will reflect the overall status of the mechanism at any point during the simulation).

(4) Finally, set the simulation into motion by asserting the existence of those conditions which set the mechanism in motion (e.g. apply a voltage across two wires, push a switch up, etc.) The population of independent SC's will then exhibit an overall coordinated activity whose products are seen as newly-created or altered conditions which represent the mechanism's overall influence on its environment.

Our strategy is therefore a form of automatic program generation, starting from a declarative representation of a "problem". But because it is not a particularly difficult conceptual leap to convert a CSA-like graph into a population of SC's, we do not claim to be doing automatic programming to any great degree. However, since this strategy will make it quite simple to hook mechanisms together in the Mechanisms Invention phase of our project, it sets

the stage for some of the tougher problems of automatic program synthesis.

The full spontaneous computation component of the CSA system is described in detail in [R3]. We will give here only a summary of those features which are significant to the simulator.

3.2. CSA Implementation of Spontaneous Computation

Th CSA SC component is a refinement and generalization of some ideas originally implemented in MICROPLANNER [SWC1] and CONNIVER [MS1]. There may be arbitrarily many spontaneous computations (SC's). Each SC consists of a trigger part and a body. The body may be any LISP computation, and it will be queued and eventually run whenever the its associated trigger condition is satisfied. The trigger condition may be a complex specification of conditions (e.g. mechanism events) composed out of the logical connectives AND, OR and ANY.

Since there will be many SC's, each having a potentially complex trigger pattern, some organization is required to make possible the efficient and exhaustive retrieval of all SC's which have the potential for reacting to a given stimulus (condition). For this purpose, we have developed a structure called a trigger tree. A trigger tree serves as a receptacle for complex trigger patterns, such that each path from the root of the trigger tree to a terminal node of the tree corresponds to one part of one complex trigger. The exact structure of the tree is unimportant; however, it gives rise to the metaphor of "planting a trigger pattern in a trigger tree".

To define a spontaneous computation, we "plant" the components of its trigger pattern in some tree via the function $PLANT:

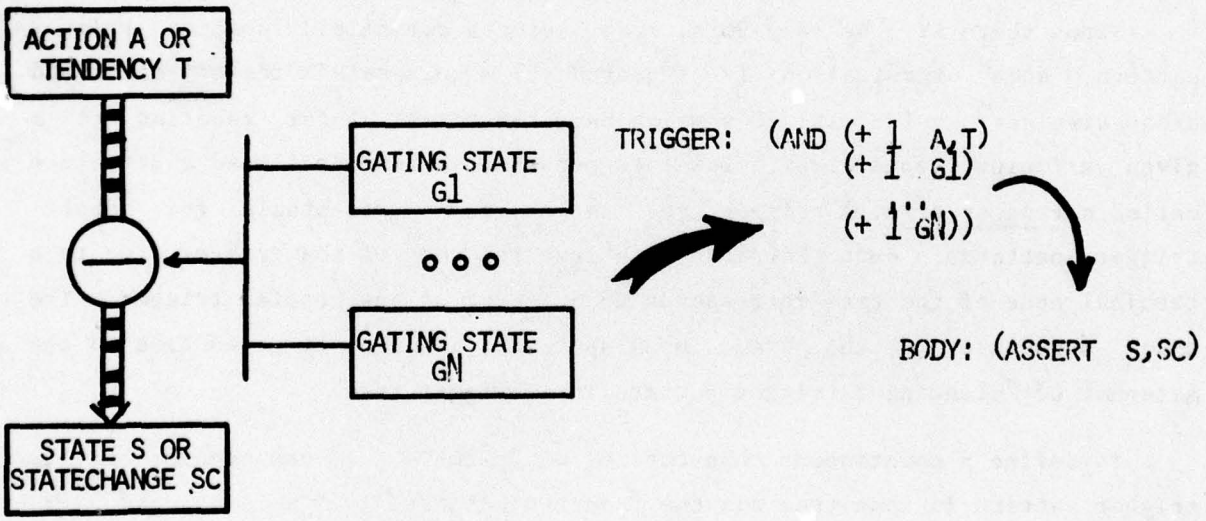
($PLANT <complex-trigger-pattern> <body> <trigger-tree>)

Display 16.

This will give rise to a new SC in the system (a symbol such as G47 as it will appear in the simulator examples) with which the specified body is associated,

and which itself is referenced from terminal nodes of the trigger tree in which it has been planted. Each terminal node will correspond uniquely to a path in the tree, and the path will correspond to one of the associative components of the complex trigger pattern. It thus will be possible to arouse the SC from numerous terminal nodes in the tree, each corresponding to a stimulus which is applicable to the triggering of the SC.

The display below illustrates one simple type of correspondence between a CSA declarative construction and a spontaneous computation. In this case the tendency (or action) plus its gating conditions comprise the trigger pattern of an SC which will assert the existence of the caused state whenever the tendency (or action) becomes active (i.e. its enabling conditions feeding in from above but not pictured all become true) and all gating conditions become true. This is a rather simple case of declarative-procedural correspondence; Grinberg will describe later all the various nuances involved in this process.



Display 17. Declarative-procedural correspondence.

Tree activation, i.e. associative lookup, is accomplished by the function $ACTIVATE:

($ACTIVATE <stimulus> <trigger-tree>)

Display 18.

which is given as stimulus some event description, such as (TENDENCY VOLTAGE FILAMENT-1) and some trigger tree, and which then causes any relevant SC's to be associatively accessed. In the mechanisms simulator this associative reaction to an event stimulus will alert any SC's which derive influence from such an event, possibly causing them to execute.

Since an SC's trigger pattern will generally consist of numerous parts, several or all of which must be satisfied before the SC can run, initial associative triggering will in general initiate a process of polling wherein the remaining components of the triggered SC's trigger pattern will be tested in order to determine whether or not the SC is actually ready to run (e.g. all the enabling conditions for some tendency exist). Since the current state of the total mechanism is modeled by a collection of event descriptions asserted in a central database, polling in the case of the Mechanisms Simulator will amount to a single database lookup. In the general CSA model, however, polling may initiate general deductive activities attempting to determine the truth or falsity of some polled condition.

3.2.1. Channels

SC's can be created, deleted, masked (temporary deletion), unmasked and activated via functions summarized in Display 19. (The mechanisms simulator will employ all these operations.) Additionally, entire trigger trees can be organized into higher constructions, called channels, which will make possible the orderly control of associative activation. A channel is thought of as a broadcast medium which carries event descriptions (regarded as signals) being broadcast to named "servers" and trees attached to the channel. An example of a channel is depicted in Display 20, accompanied by the calls to the SC and channel building and invoking functions, $PLANT, $CONNECT and $INJECT which would construct and then use such a channel.

Create an SC: ($PLANT <trigger-pattern> <body> <trigger-tree>)

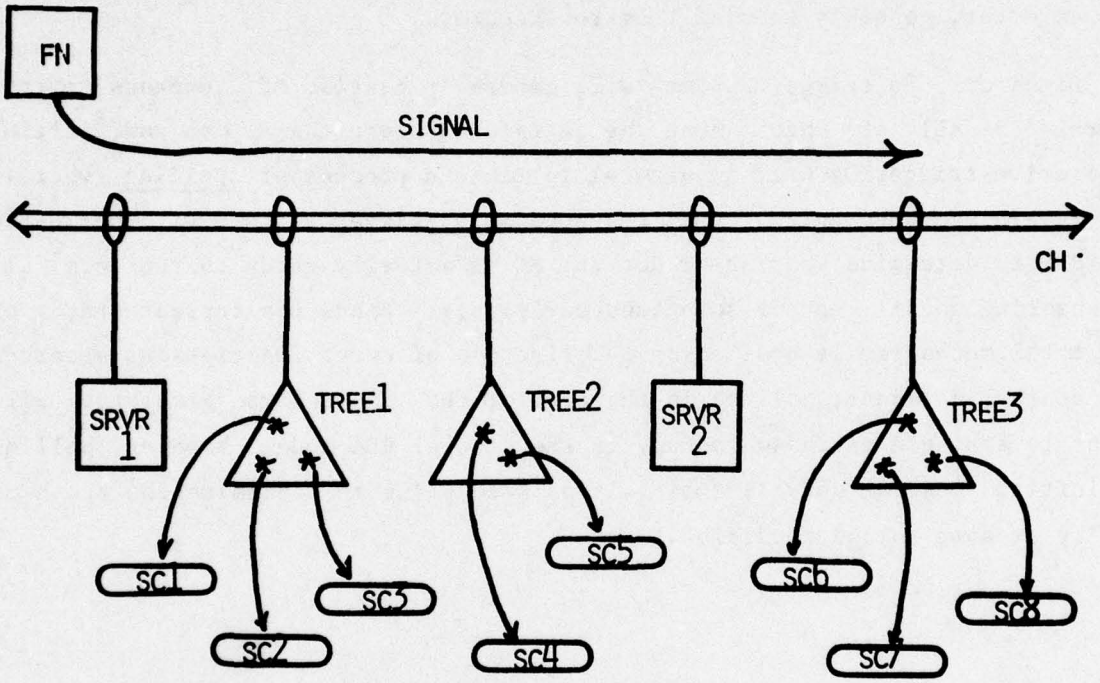
Remove an SC: ($KILL <sc>)

Mask an SC: ($HIDE <sc>)

Unmask an SC: ($UNHIDE <sc>)

Activate a trigger tree: ($ACTIVATE <stimulus> <trigger-tree>)

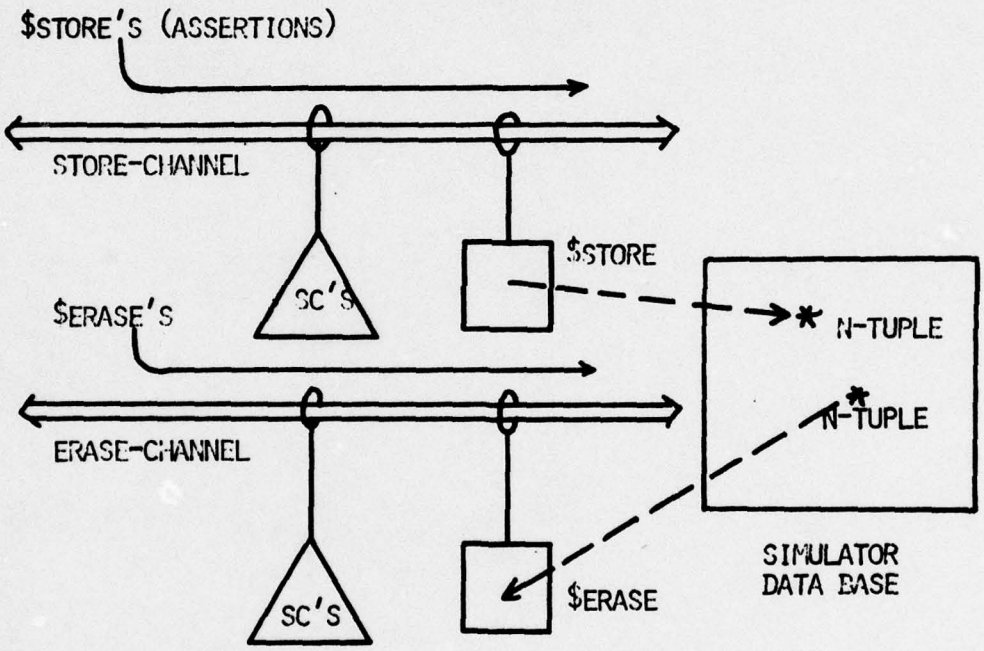
Display 19. Spontaneous computation defining and activating functions.



($PLANT SC1 TREE1) ($PLANT SC2 TREE1) ($PLANT SC3 TREE1)
($PLANT SC4 TREE2) ($PLANT SC5 TREE2) ($PLANT
($PLANT SC6 TREE3) ($PLANT SC7 TREE3) ($PLANT SC8 TREE3)
($CONNECT SRVR1 CH TRANSPARENT SERVER AT LEFTEND)
($CONNECT SRVR2 CH TRANSPARENT SERVER AFTER SRVR1)
($CONNECT TREE1 CH TRANSPARENT WATCHER AFTER SRVR1)
($CONNECT TREE2 CH MODIFYING WATCHER BEFORE SRVR2)
($CONNECT TREE3 CH TRANSPARENT RESPONSE-WATCHER AT RIGHTEND)
($INJECT SIGNAL SRVR2 CH AT LEFTEND RIGHT)

Display 20. Constructing and using a channel.

Since the Mechanisms Simulator will be concerned with two generic types of event, the creation of a condition and the destruction of a condition, it will employ only two distinct channels. To one will be attached a population (tree) of SC's designed to react to newly-arisen conditions; to the other will be attached a population of SC's designed to react to conditions whose existence is terminating. New conditions are stored in the database via the function ($STORE <event>); existing conditions are terminated via the function ($ERASE <event>). The arrangement of the two simulator channels is depicted in Display 21.



Display 21. The two Mechanisms Simulator channels.

3.2.2. Simulation

After having created the two populations of SC's and their associated channels from the mechanism description and having asserted the existence of any nominal starting conditions (specified at the time the CSA mechanism description was given to the system), the simulator asserts the mechanism's

trigger conditions. This gives rise to a sequence of SC activity: activation, deactivation, masking and unmasking, controlled by a synchronous, discrete timer. The intermediate products of the simulation are visible as a trace of SC activity; the final products are indicated by the collection of conditions which remain in the database at the end of the simulation.

We now describe and illustrate the operation of the Mechanisms Simulator in detail. After this description, we discuss the next phases of the project, one of which is already in progress.

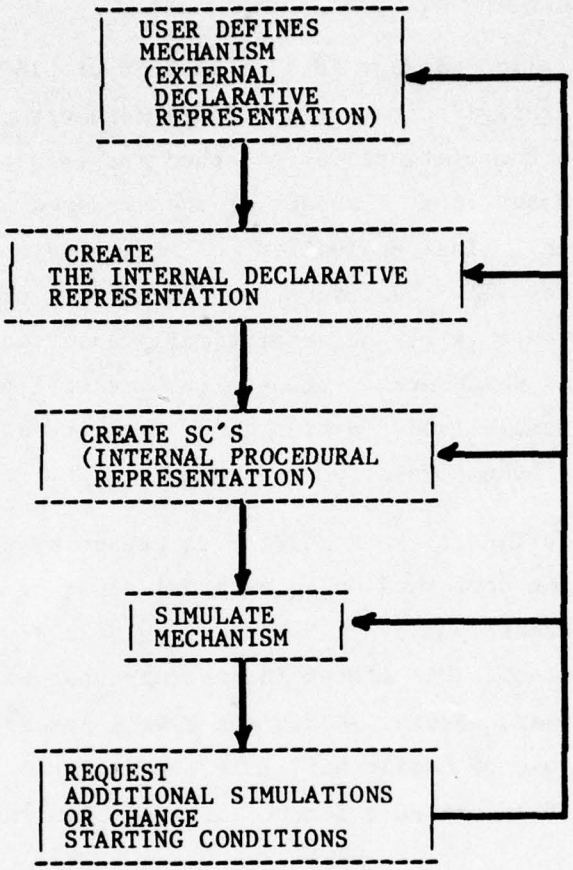
4. The Mechanism Simulator

4.1. Overview

The simulation package is a collection of LISP functions which take the internal declarative representation of a mechanism, convert it to a population of Spontaneous Computations (SC's), then awaken a subset of the population via a triggering assertion. A subset of the awakened SC's will request that their bodies be run. The evaluation of the bodies will cause other SC's to awakened. This will constitute an execution of the mechanism. In the execution, events will be automatically asserted and deasserted in the data base and states which are changing with time will be updated. The status of an event is established in accordance with its causal relationships to other events as they become asserted and deasserted.

The initiation of the simulator is caused by the assertion of some event: either an action performed by an external actor or caused by a natural force, or by the assertion of a state, e.g., as derived from the simulation of another mechanism. This latter initiating event will provide for mechanism interaction where several mechanisms have a common event and the assertion of that event in one mechanism will initiate execution of the other mechanism. (This assumes that the SC's for the other mechanisms have been created.)

Each link in the internal declarative representation of a mechanism has a set of SC's automatically created for it. An SC is activated when its preconditions have been satisfied. The running of an SC will generally cause a data base modification or a masking or unmasking of another SC.



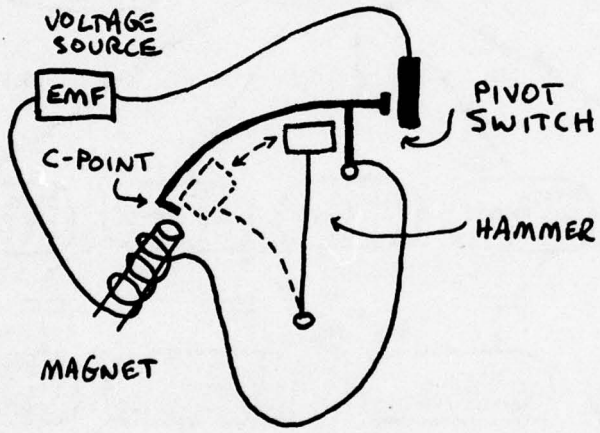
Display 22. Flow Of Control

4.2. Example Simulation

Before discussing the theory behind the design and implementation of the simulator, an example of a simplified mechanism will be presented and the mechanism will be run through the current Mechanisms Laboratory. We make no claims that the example mechanism is complete, that the events are all at the same level of description, or that the rates or discrete values used are

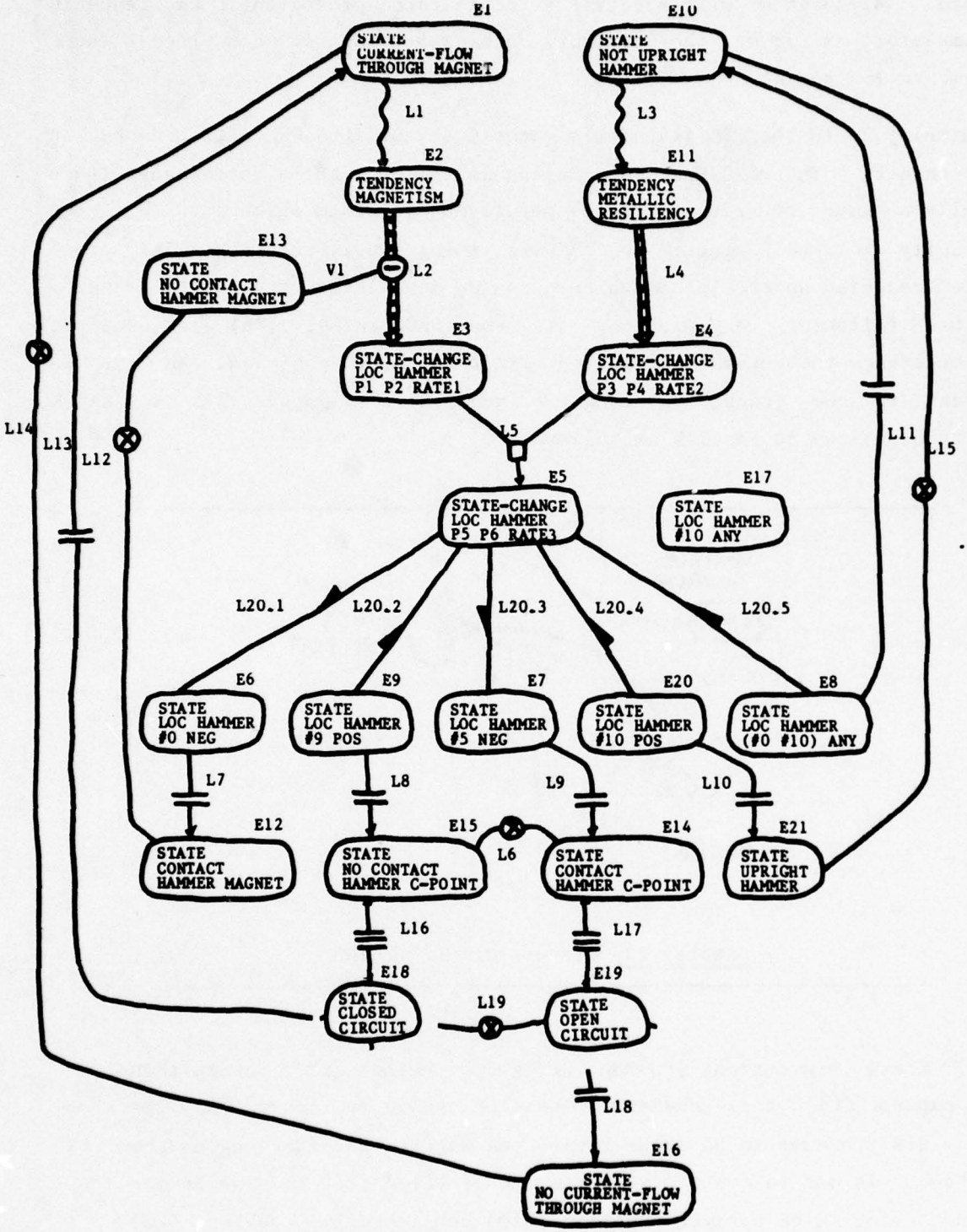
accurate. Although we will eventually incorporate more semantic knowledge in the simulator, as far as the current Mechanisms Lab is concerned, these factors are immaterial.

Display 24 is the CSA graph of a mechanical OSCILLATOR operated by an electromagnet. The OSCILLATOR functions as follows: the electromagnet (when on) pulls a hammer towards it; the hammer reaches a plate which cuts off the electricity to the magnet; the hammer then slowly returns to its normal position reaching another plate which turns on the electricity to the magnet. In the following explanation of the OSCILLATOR, {Xn} is used to cross-reference the verbal explanation with the graphic position. An {En} is an event in the graph, {Ln} is a link and {Vn} is a gate. This OSCILLATOR could be explained in English as follows:



Display 23. Picture Of OSCILLATOR

If there is a current flowing through the magnet {E1}, then this enables {L1} the tendency MAGNETISM {E2} to cause {L2} the metallic hammer to be drawn toward the magnet {E3} (as long as the hammer is not in contact with the magnet {E13} {V1}). The hammer not being in an upright position {E10} continuously enables {L3} the tendency of METALLIC RESILIENCE {E11} continuously to cause {L4} the hammer to move back to an upright position {E4}.



Display 24. CSA graph of OSCILLATOR.

The change in location of the hammer to an upright position {E4} contacting the magnet {E3} combines {L5.1 L5.2} into a net rate of change in the location of the hammer {E5}.

If the hammer position is at 0 {E6}, this implies {L7} that the hammer is in contact with the magnet {E12} which contradicts {L12} the state: no contact between the hammer and the magnet {E13}.

If the hammer's location is at 10 {E20}, then {L10} the hammer is in an upright position {E21} which contradicts {L15} the state: hammer not being upright {E10}.

If the hammer's location is less than 10 {E8}, then {L11} the hammer is not upright {E10} (which enables {L3} the tendency of METALLIC RESILIENCE {E11}).

The hammer passing through 5 in the negative direction {E7} (moving toward 0) implies {L9} that the hammer and contact-point are in contact {E14}. This is equivalent {L17} to an open circuit {E19} which implies {L18} that no current is flowing through the magnet {E16}. This contradicts {L14} the state: current flowing through the magnet {E1}.

The hammer location being above 9 {E9} implies {L8} that there is no contact between the hammer and the contact-point {E15}. This is equivalent {L16} to a closed circuit {E18} which implies {L13} that current is flowing through the magnet {E1}.

4.2.1. Input Syntax

The Mechanisms Lab requires that the graph be coded into a machine readable form. This form is the external declarative CSA-representation described earlier and is used as input to a mechanism defining function called $MECHANISM. This function generates the internal declarative representation of the mechanism from the external declarative representation. The external declarative form of this OSCILLATOR is shown in Display 25.

```
($MECHANISM '(
(NAME OSCILLATOR)
(EVENTS (1 S (CURRENT FLOW THROUGH MAGNET))
        (2 T (MAGNETISM))
        (3 SC (STATE-CHANGE LOC HAMMER P1 P2 RATE1))
        (4 SC (STATE-CHANGE LOC HAMMER P3 P4 RATE2))
        (5 SC (STATE-CHANGE LOC HAMMER P5 P6 RATE3))
        (6 S (LOC HAMMER #0 NEG))
        (7 S (LOC HAMMER #5 NEG))
        (8 S (LOC HAMMER (#0 #10) ANY))
        (9 S (LOC HAMMER #9 POS))
        (10 S (HAMMER NOT UPRIGHT))
        (11 T (METALLIC RESILIENCY))
        (12 S (CONTACT HAMMER MAGNET))
        (13 S (NO CONTACT HAMMER MAGNET))
        (14 S (CONTACT HAMMER C-POINT))
        (15 S (NO CONTACT HAMMER C-POINT))
        (16 S (NO CURRENT FLOW THROUGH MAGNET))
        (17 S (LOC HAMMER #10 ANY))
        (18 S (CLOSED CIRCUIT))
        (19 S (OPEN CIRCUIT))
        (20 S (LOC HAMMER #10 POS))
        (21 S (HAMMER UPRIGHT))     )
(LINKS  (C-ENABLE (1 2))
        (C-CAUSE (2 3) (13))
        (C-ENABLE (10 11))
        (C-CAUSE (11 4))
        (RATE-CONFL (3 4) 5))
        (THRESH (5 (6 7 8 9 20)))
        (S-COUPLE (6 12))
        (S-COUPLE (9 15))
        (S-COUPLE (7 14))
        (S-COUPLE (8 10))
        (ANTAG (16 1))
        (ANTAG (15 14))
        (ANTAG (13 12))
        (S-EQUIV (15 18))
        (S-EQUIV (14 19))
        (S-COUPLE (20 21))
        (ANTAG (21 10))
        (S-COUPLE (18 1))
        (S-COUPLE (19 16))
        (ANTAG (18 19))     )
(RATES  (RATE1 #-4)
        (RATE2 #1)
        (RATE3 (#PLUS RATE1 RATE2))   )
(INITIAL-WORLD  17 13)
(TRIGGER 1)
(PURPOSE 18 19)                    ))
```

Display 25. $MECHANISM External Declarative Representation

The list (EVENTS e1 ... en) is the list of events (i.e. nodes in the graph) for the mechanism. They are formatted as follows:

(<number> <type-event> <n-tuple>)

Display 26. $MECHANISM Event Form

<number> is a unique reference to this event and is used in the place of the event in all other sections of the external representation for cross-referencing purposes. <type-event> specifies the type of event, the options being 'A' - action; 'T' - tendency; 'S' - state; 'SC' - state-change. <n-tuple> is the actual n-tuple representing the event.

The list (LINKS l1 ... lm) is the list of links (i.e. cause-effect relationships in the graph connecting nodes) for the mechanism. The format is:

(<type-link> (<event1> <event2>) (<gate1> ... <gaten>))

Display 27. $MECHANISM Link Form

<type-link> is the type of arc connecting the events. The alternatives are the CSA links described earlier, namely, C-ENABLE, OS-ENABLE, C-CAUSE, OS-CAUSE, S-COUPLE, S-EQUIV, RATE-CONFL, THRESH, and ANTAG. (<event1> <event2>) is a list of two elements. The first is the causing event (or list of causing events) and the second is the caused event (or list of caused events). (<gate1> ... <gaten>) is an optional list which specifies the events which are gates on the link (for those links which can accept gates).

Besides the links and events in the graph, there are several other components of the external representation.

(1) (NAME X), the atom X is to be used as the CSA name of this mechanism. If no name is given then the system creates and reports

an arbitrary symbol to be used as the name. The internal representation is stored on the property list of the mechanism's name.

(2) (INITIAL-WORLD n1 n2 ...), a list of conditions which are assumed to be true when the mechanism simulation is begun (i.e. nominal starting conditions for the mechanism). Each INITIAL WORLD event is referenced by its event number. These events are asserted at the beginning of the simulation.

(3) (PURPOSE n1 n2 ...), a list of event numbers which describe the purpose of the mechanism. Although it is currently unused by the mechanism lab, it plays an important role in the CSA plan synthesizer (described in [R1]).

(4) (TRIGGER n1 n2 ...), a list of events which when asserted simultaneously will activate the automatic operation of the mechanism, i.e. events which cause the mechanism to become active. They are also referenced by event number.

(5) (RATES (rate1 function1) (rate2 function2) ...), a list of rate name and value pairs. The values may either be a CSA-number (to be defined) or an arbitrary function which evaluates to a CSA-number.

The internal CSA declarative representation for the OSCILLATOR created by ($MECHANISM '((NAME OSCILLATOR) ...)) is

```
**OBJECT OSCILLATOR (ABS-MECH)**
   NAME:  (OSCILLATOR)
   PARTS:  NIL
   EVENTS:  (G4 G5 G6 G7 G8 G9 G10 G11 G12 G13 G14
             G15 G16 G17 G18 G19 G20 G21 G22 G23 G24)
   PURPOSE:  (G21 G22)
   TRIGGER:  (G4)
   INITIAL-WORLD:  (G20 G16)
   RATES: (G1 G2 G3)
```

Display 28. Internal Representation of OSCILLATOR

The following three displays are samples of the internal representation of an

event, a link, and a rate.

```
**OBJECT G13 (ABS-MECH-EVENT)**
   PART-OF:  OSCILLATOR
   CLASS:  S
   NTUPLE:  (NOT UPRIGHT HAMMER)
   LINKS:  (G41 G34)
```

Display 29. Internal Representation of Event Node

```
**OBJECT G34 (ABS-MECH-LINK)**
   TYPE:  S-COUPLE
   INSTANCE-OF:  NIL
   EVENTS:  (G11 G13)
   GATES:  NIL
   SC-LIST: (G85 G86)
```

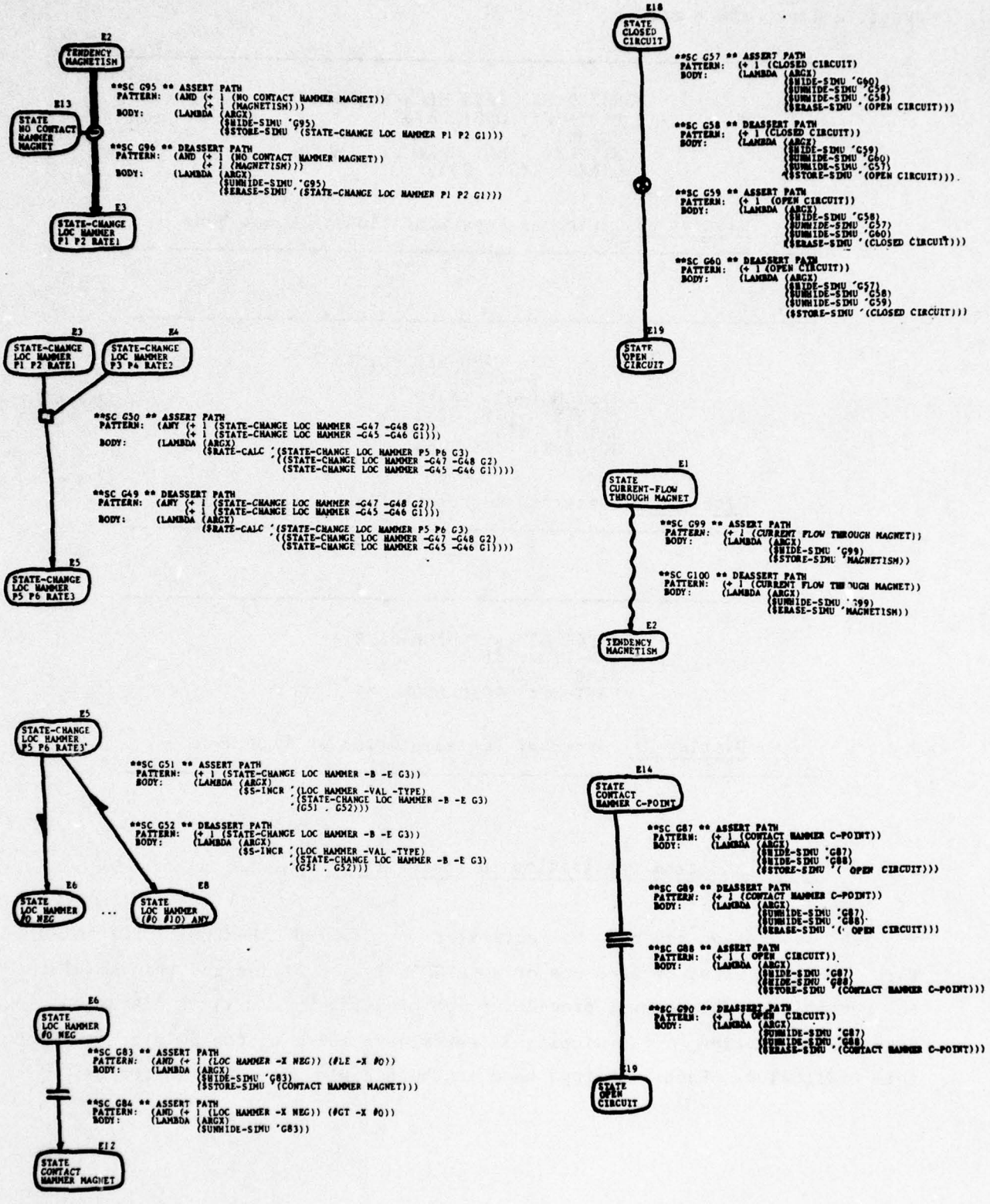
Display 30. Internal Representation of Link Node

```
**OBJECT G2 (ABS-MECH-RATE)**
   FUNCTION: #1
   NAME: RATE2
   PART-OF: OSCILLATOR
```

Display 31. Internal Representation of Rate Node

4.2.2. Conversion from Declarative to Procedural Form

The simulation can now be activated by ($SIMULATE-MECH 'OSCILLATOR). This function first creates one or more SC's for each link and then simulates the mechanism. The general procedures for creating SC's will be discussed in detail in Section 4.5. Display 32 contains samples of the SC's created for this OSCILLATOR. Each link type used in the example will be illustrated.



Display 32. Sample SC's for the OSCILLATOR

4.2.3. Simulation

An SC is associated with either the assertion or deassertion of its triggering events. The assertion or deassertion of an event can be considered as being performed on one of two separate paths into the data base. After each SC has been created and enabled to watch its respective data path, the simulation begins. How this occurs is described in more detail later. Display 33 is a listing of the actual simulation of the OSCILLATOR. The left side of the display is the computer output. The right side is an annotation of what was occuring at that moment.

```
EVAL: ($SIMULATE-MECH 'OSCILLATOR)
 MAX TICK COUNT (T = INFINITY)  12

 ** INITIAL WORLD EVENTS **
(LOC HAMMER #10 ANY) STORED G104 CAUSED BY IW                          initial
(NO CONTACT HAMMER MAGNET) STORED G108 CAUSED BY IW                    world
 ** END INITIAL WORLD **                                               established
   G72 HIDDEN CAUSED BY G69

                                                                       trigger
(CURRENT FLOW THROUGH MAGNET) STORED G109 CAUSED BY TRIGGER            asserted

TICK 0 REAL TIME 440
   G99 HIDDEN CAUSED BY G99                                            magnet
(MAGNETISM) STORED G110 CAUSED BY G99                                  starts to
   G95 HIDDEN CAUSED BY G95                                            affect location
(STATE-CHANGE LOC HAMMER P1 P2 G1) STORED G111 CAUSED BY G95           of hammer
(STATE-CHANGE LOC HAMMER P5 P6 G3) STORED G112 CAUSED BY G49
   G62 HIDDEN CAUSED BY G63

TICK 1 REAL TIME 997
   G51 HIDDEN CAUSED BY G51
   G52 HIDDEN CAUSED BY G51
(LOC HAMMER #6 NEG) CHANGED G104 CAUSED BY G51
   G85 HIDDEN CAUSED BY G85
(HAMMER NOT UPRIGHT) STORED G113 CAUSED BY G85                         metallic
   G101 HIDDEN CAUSED BY G101                                          resilience
(METALLIC RESILIENCY) STORED G114 CAUSED BY G101                       starts to
   G97 HIDDEN CAUSED BY G97                                            affect location
(STATE-CHANGE LOC HAMMER P3 P4 G2) STORED G103 CAUSED BY G97           of hammer
   G54 HIDDEN CAUSED BY G55

TICK 2 REAL TIME 1949
(LOC HAMMER #3 NEG) CHANGED G104 CAUSED BY $S-INCR                     hammer reaches
   G81 HIDDEN CAUSED BY G81                                            cutoff position
(CONTACT HAMMER C-POINT) STORED G107 CAUSED BY G81                     and shuts off
   G87 HIDDEN CAUSED BY G87                                            current to
   G88 HIDDEN CAUSED BY G87                                            magnet
(OPEN CIRCUIT) STORED G106 CAUSED BY G87
   G75 HIDDEN CAUSED BY G75
(NO CURRENT FLOW THROUGH MAGNET) STORED G116 CAUSED BY G75
   G64 HIDDEN CAUSED BY G61
   G62 UNHIDDEN CAUSED BY G61
(CURRENT FLOW THROUGH MAGNET) ERASED G109 CAUSED BY G61
   G99 UNHIDDEN CAUSED BY G100
(MAGNETISM) ERASED G110 CAUSED BY G100
   G95 UNHIDDEN CAUSED BY G96
(STATE-CHANGE LOC HAMMER P1 P2 G1) ERASED G111 CAUSED BY G96
   G58 HIDDEN CAUSED BY G59
   G66 HIDDEN CAUSED BY G67
```

```
TICK 3 REAL TIME 4226                                          | hammer starts
(LOC HAMMER #4 POS) CHANGED G104 CAUSED BY $S-INCR             | to move back to
                                                               | start position
TICK 4 REAL TIME 4641
(LOC HAMMER #5 POS) CHANGED G104 CAUSED BY $S-INCR

TICK 5 REAL TIME 5029
(LOC HAMMER #6 POS) CHANGED G104 CAUSED BY $S-INCR
   G81 UNHIDDEN CAUSED BY G82

TICK 6 REAL TIME 5406
(LOC HAMMER #7 POS) CHANGED G104 CAUSED BY $S-INCR

TICK 7 REAL TIME 5786
(LOC HAMMER #8 POS) CHANGED G104 CAUSED BY $S-INCR

TICK 8 REAL TIME 6165
(LOC HAMMER #9 POS) CHANGED G104 CAUSED BY $S-INCR             | hammer reaches
   G79 HIDDEN CAUSED BY G79                                    | switch
(NO CONTACT HAMMER C-POINT) STORED G111 CAUSED BY G79          | and current
   G91 HIDDEN CAUSED BY G91                                    | starts to flow
   G92 HIDDEN CAUSED BY G91                                    | into magnet
(CLOSED CIRCUIT) STORED G110 CAUSED BY G91
   G77 HIDDEN CAUSED BY G77
(CURRENT FLOW THROUGH MAGNET) STORED G109 CAUSED BY G77
   G99 HIDDEN CAUSED BY G99
(MAGNETISM) STORED G115 CAUSED BY G99
   G95 HIDDEN CAUSED BY G95
(STATE-CHANGE LOC HAMMER P1 P2 G1) STORED G105 CAUSED BY G95   | hammer starts
   G62 HIDDEN CAUSED BY G63                                    | to move toward
   G64 UNHIDDEN CAUSED BY G63                                  | magnet
(NO CURRENT FLOW THROUGH MAGNET) ERASED G116 CAUSED BY G63
   G60 HIDDEN CAUSED BY G57
   G58 UNHIDDEN CAUSED BY G57
(OPEN CIRCUIT)    ERASED G106 CAUSED BY G57
   G87 UNHIDDEN CAUSED BY G90
   G88 UNHIDDEN CAUSED BY G90
(CONTACT HAMMER C-POINT) ERASED G107 CAUSED BY G90
   G65 HIDDEN CAUSED BY G68
   G66 UNHIDDEN CAUSED BY G68
   G75 UNHIDDEN CAUSED BY G76
   G68 HIDDEN CAUSED BY G65
   G65 UNHIDDEN CAUSED BY G65

TICK 9 REAL TIME 7688
(LOC HAMMER #6 NEG) CHANGED G104 CAUSED BY $S-INCR
   G79 UNHIDDEN CAUSED BY G80

TICK 10 REAL TIME 9082
(LOC HAMMER #3 NEG) CHANGED G104 CAUSED BY $S-INCR             | hammer reaches
   G81 HIDDEN CAUSED BY G81                                    | cutoff position
(CONTACT HAMMER C-POINT) STORED G107 CAUSED BY G81             | and magnet is
   G87 HIDDEN CAUSED BY G87                                    | shutoff
   G88 HIDDEN CAUSED BY G87
(OPEN CIRCUIT)    STORED G106 CAUSED BY G87
   G75 HIDDEN CAUSED BY G75
(NO CURRENT FLOW THROUGH MAGNET) STORED G116 CAUSED BY G75
   G64 HIDDEN CAUSED BY G61
   G62 UNHIDDEN CAUSED BY G61
(CURRENT FLOW THROUGH MAGNET) ERASED G109 CAUSED BY G61
   G99 UNHIDDEN CAUSED BY G100
(MAGNETISM) ERASED G115 CAUSED BY G100
   G95 UNHIDDEN CAUSED BY G96
(STATE-CHANGE LOC HAMMER P1 P2 G1) ERASED G105 CAUSED BY G96
   G58 HIDDEN CAUSED BY G59
   G60 UNHIDDEN CAUSED BY G59
(CLOSED CIRCUIT) ERASED G110 CAUSED BY G59
   G91 UNHIDDEN CAUSED BY G94
   G92 UNHIDDEN CAUSED BY G94
(NO CONTACT HAMMER C-POINT) ERASED G111 CAUSED BY G94
```

```
   G67 HIDDEN CAUSED BY G66
   G68 UNHIDDEN CAUSED BY G66
   G77 UNHIDDEN CAUSED BY G78
   G66 HIDDEN CAUSED BY G67
   G67 UNHIDDEN CAUSED BY G67

TICK 11 REAL TIME 10565                                    | hammer starts
(LOC HAMMER #4 POS) CHANGED G104 CAUSED BY $S-INCR         | moving toward
                                                           | upright position
TICK 12 REAL TIME 10944

LIST ADDITIONAL ACTIVITIES:   nil                          | simulation is
                                                           | terminated
** FINAL WORLD **                                         |
(LOC HAMMER #4 POS)                                        | final state
(NO CURRENT FLOW THROUGH MAGNET)                           | of the
(OPEN CIRCUIT)                                             | oscillator
(CONTACT HAMMER C-POINT)
(STATE-CHANGE LOC HAMMER P3 P4 G2)
(METALLIC RESILIENCY)
(HAMMER NOT UPRIGHT)
(STATE-CHANGE LOC HAMMER P5 P6 G3)
(NO CONTACT HAMMER MAGNET)
OSCILLATOR RESIMULATED 10991
```

Display 33. Simulation of the OSCILLATOR

4.3. Spontaneous Computations

The Spontaneous Computation (SC) routines are an essential part of the simulation package. Once the trigger events in the external representation are asserted, almost all of the remaining actions of the simulator are caused by SC's responding to the assertion, deassertion, or change of an event. To understand the simulator it is necessary to have a basic knowledge of the SC's. (For a more complete report on SC's and their implementation in CSA see [R3]).

An SC in the Mechanisms Lab simulator watches a particular data path to or from the data base. An SC can be associated with one of two data paths; the ASSERT data path and the DEASSERT data path. (These are essentially the same as the THASSERT and THERASE of MICROPLANNER.) When an event is asserted (stored) in the data base it becomes true. For our purposes, any condition not stored in the data base is considered not to exist. The paths into the data base for the assertion and deassertion of an event are different. Each SC will be connected to (i.e. watching) either the ASSERT or the DEASSERT path.

An SC is composed of two parts: a trigger and a body. The trigger can be a complex grouping of events or a single event. When one of these events passes along the particular data path the SC is watching, the SC is awakened. The SC polls the remaining events in the trigger (if any). If all the polled events are asserted (true), the SC invokes its body (i.e. requests that its body be run). For example, if an SC is watching the assertion path with a trigger of (AND STATE1 STATE2), then asserting STATE2 will cause the SC mechanism to ask if STATE1 is in the data base. If it is, the SC body is placed on a list of processes which want to run. If STATE2 is not in the data base then the polling fails and the SC does nothing. The deassertion of STATE1 or STATE2 will not effect this SC since it is watching the ASSERT path into the data base.

In the Mechanisms Lab, each SC associated with the DEASSERT path will invoke its body when any of its trigger pattern events are deasserted. A typical SC watching the DEASSERT path will have a trigger pattern (OR STATE1 STATE2 ... STATEn) which is the negation of some SC on the ASSERT path. The deassertion of either STATE1 or STATE2 or ... or STATEn will cause the SC to invoke its body.

The SC's are automatically created by the simulator. Before each link is presented and the SC's created for it are given, an overview of some of the important details of the system will be discussed.

4.4. Overview of Simulator

4.4.1. Initial World

The initial world conditions are those events which are assumed to be in effect at the time simulation is to begin. They are listed in the external representation in the list headed by INITIAL-WORLD. The simulator must assert these facts in the data base. Typically, an initial world event will be a gating condition on a link or the ID-EVENT of a state-change (see Section 4.5.1.1). The SC's invoked by the assertions of the INITIAL WORLD events are processed before the mechanism triggering events are asserted.

4.4.2. Trigger Events

The trigger events are the events which initiate the execution of the actual simulation. The assertion of these events awakens some SC's, who place their bodies on a list of procedures whose running is pending. The simulator then executes the mechanism to completion.

4.4.3. Context Level

The simulator uses the CSA chronological context facility. The simulator normally begins at a context level of 0. The SC's are planted at context level 1 and the simulation's data base accesses (stores, erases, changes, and SC hiding and unhiding) are performed at context level 2. This enables the user to expunge the simulator's entire factual data base by simply returning to context level 1. The mechanism could then be resimulated without requiring that the SC's for the links (which still exist at context level 1) be recreated. The SC's themselves can be expunged by returning to context level 0. This would be necessary, e.g., if an error is detected in the external representation and a new internal representation must be created. The incrementation of the context level is automatically performed by the simulator; however, the decrementation of the context level must be performed by the user via calls of the form ($POP-CONTEXT) to decrement the context level by 1. CONTEXT-STACK is the list of data base datum created at each context level. CONTEXT-LEVEL is the current context level.

4.4.4. Data Base Interaction

The CSA data base access functions can be run in an interactive mode wherein failures to locate a fact in the data base will cause a query to the user to be posted. Thus if an event is not found in the data base, the user can have the opportunity to indicate its truth or falsity. This interaction is not necessary in the simulator since all information relevant to the simulator is assumed to be contained in the internal representation. Hence the simulator disables the interactive facility and assumes that any event not

found in the data base is false.

4.4.5. Mech-Agenda

A run queue of pending procedures is maintained by the system. An SC may be added to the list as the result of having been triggered by the execution of another SC or by a "self-regenerating" technique. To "activate" an SC is to place its body on the run queue for subsequent EVALing. Some procedures as part of their evaluation may place themselves back onto the run queue (self-regeneration). $TIMER and $S-INCR (discussed in detail later) are two examples of "self-regenerating" procedures. Hence the run queue maintains a list of activities which must be EVALed. When the run queue becomes empty the simulation enters termination mode. In the simulator, the variable MECH-AGENDA is the pointer to the run queue. If the simulator should fail due to a LISP error, MECH-AGENDA will point to the list of pending activities.



MECH-AGENDA QUEUE

Display 35. Abstract View of MECH-AGENDA

4.4.6. CSA Number System and Arithmetic

The Mechanisms Lab cannot use the normal number system available in LISP. This is due to the particular method of storing events in the data base. Wisconsin LISP does not allow numbers to possess property lists. However, the data base mechanics requires that all elements of an event's n-tuple accept property lists. To alleviate this problem, all the numbers used in the n-tuple of an event or in the symbolic arithmetic expression of a rate are

prefixed by an '#'. For example, the number 5 will be represented as #5 and the number -4 will be represented as #-4. These atoms which represent numbers will be called CSA-numbers.

In order to manipulate the CSA-numbers, a package of arithmetic routines was written. These routines correspond to the LISP arithmetic package with the exception that their arguments must be CSA-numbers. The arithmetic predicates return either 'T' or NIL, while the functions return a CSA-number. The procedures in this package are named after their corresponding procedure with the exception that their names are prefixed by a '#'. For example PLUS in the CSA-arithmetic package is named #PLUS.

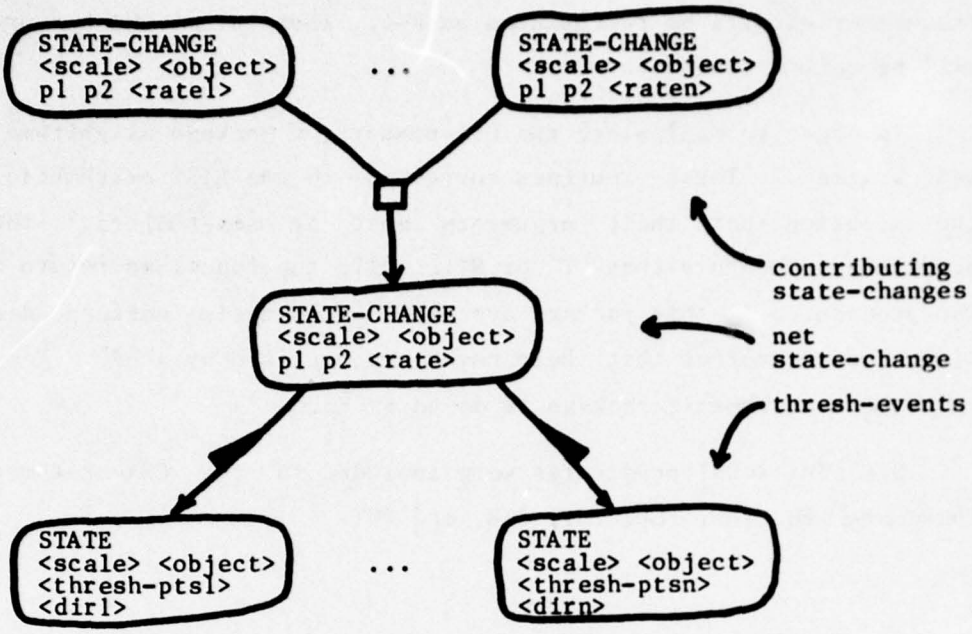
Six additional predicates were included in the CSA-arithmetic package, these are #EQ, #NE, #LE, #LT, #GE, and #GT.

4.5. Conversion of Declarative Description to Procedural

We now describe the theory of the declarative-procedural conversion process which transforms a CSA mechanism description into a population of SC's. The discussion is organized by CSA links and specific subgraph configurations of interest.

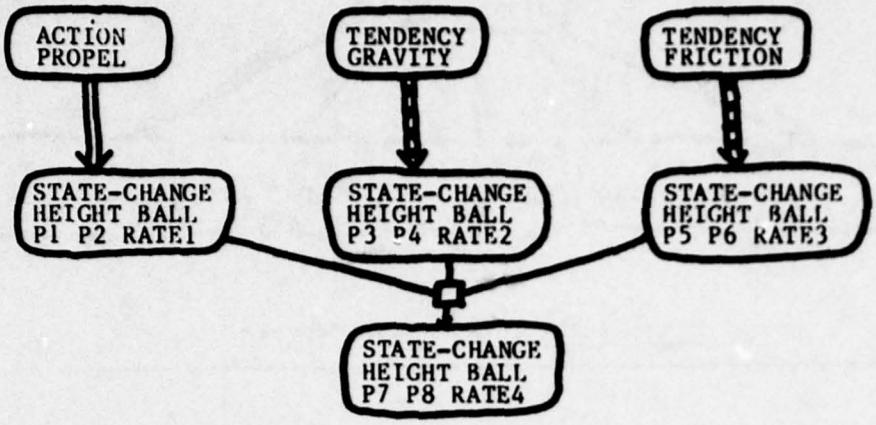
4.5.1. Continuous Scale Statechange

The continuous scale state-change construction is generally composed of two links. The first link is the RATE-CONFLuence link (RATE-CONFL). This link combines many different state-changing influences into a single state-change event. The second link is the THRESHold link (THRESH). This link joins a state-change event with many THRESH-EVENTs, points of interest along some continuous scale.



Display 36. STATE-CHANGE Link

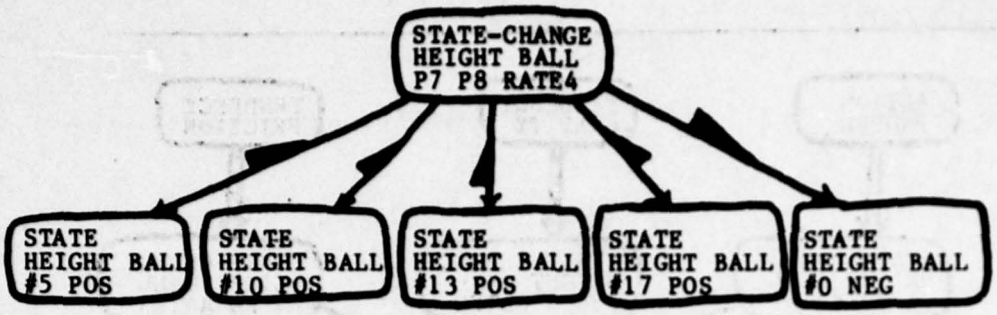
For example, consider the test-of-strength device often found at carnivals. In this device a ball is propelled upward through a tube which has a gong at the top and smaller bells located at specific heights along the tube. As the ball moves up the tube the bells ring as they are passed by the ball going up. If the ball reaches the top the gong rings. The ball always returns to the starting position. Three significant forces affect the movement of the ball. The first force is the acceleration (a positive force) given the ball when the hammer hits the lever upon which the ball rests. The second force is gravity (a negative force). The third force is friction of the ball along the side of the tube (a negative force). These three forces affect the height of the ball in the tube.



[(RATE1 #5), (RATE2 #-3), (RATE3 #-1)
and (RATE4 (#PLUS RATE1 RATE2 RATE3))]

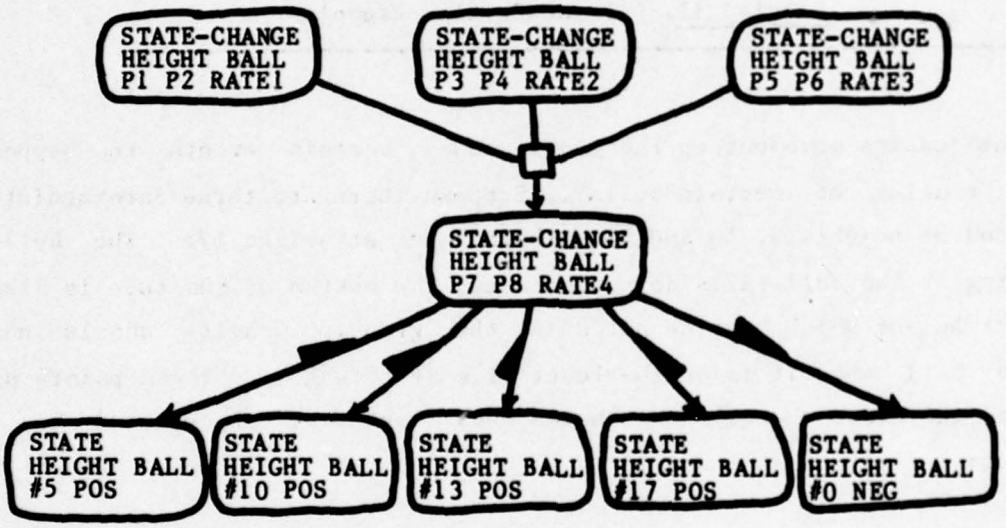
Display 37. RATE-CONFL Link Example

The ball on its movement up the tube causes certain events to happen (i.e. the ringing of certain bells). Suppose there are three intermediate bells located at heights 5, 10 and 13 with the gong at height 17. The bells do not ring as the ball falls down the tube. The bottom of the tube is also an important height which must be noted in the graph. Gravity should not affect the ball when it is on the lever (i.e at height 0). These points of interest are the state-change's thresholds and would be represented in a CSA-graph as:



Display 38. THRESH Link Example

Combining these two links produces the following CSA-graph.



[(RATE1 #5), (RATE2 #-3), (RATE3 #-1)
and (RATE4 (#PLUS RATE1 RATE2 RATE3))]

Display 39. STATE-CHANGE Links Example

Before these two important links, Rate-Confluence and Threshold, are discussed in detail, several other important elements of the state-change structure will be explained. These elements are the ID-EVENT, SC-EVENT, and

THRESH-EVENT. Following the link discussion will be a discussion of two procedures deeply embedded in the simulator's state-change related control structure, $S-INCR and $TIMER.

4.5.1.1. Instantaneous Descriptor Event (ID-EVENT)

An Instantaneous Descriptor event (ID-EVENT) is a state (stored in the data base) representing the current value of an object on a scale. It is a 4-tuple of the form:

(<scale> <object> <value> <direction>)

Display 40. ID-EVENT Form

The value of the object will be typically influenced by a state-change. For example, the ID-EVENT (TEMP AIR #90 POS) represents an object AIR having value 90 along a scale of TEMP. The <direction> field is used to indicate whether the <value> field is increasing or decreasing; here, the last value for the TEMP of AIR was less than 90. An ID-EVENT (HEIGHT WATER #10 ANY) represents WATER at a HEIGHT of 10 where the last value of the HEIGHT is unknown.

Since every state-change involved in a THRESH link must have one and only one value associated with it, there must be one and only one ID-EVENT for each <scale> <object> pair involved in a state-change event. The ID-EVENT is not directly involved in any link. However since an initial value is required, an ID-EVENT will appear in the EVENT section of the external representation of a mechanism. It will also be included in the INITIAL-WORLD section which comprises the starting set of conditions in the data base. In the example in Section 4.2, event {E17} is an ID-EVENT. The <value> and <direction> slots of this ID-EVENT is changed throughout the simulation. Notice that it is not included in any links but is included as an INITIAL-WORLD event.

4.5.1.2. Statechange Event

The State-Change (SC) event is a 6-tuple of the form:

(STATE-CHANGE <scale> <object> <start-pos> <end-pos> <rate>)

Display 41. STATE-CHANGE Event Form

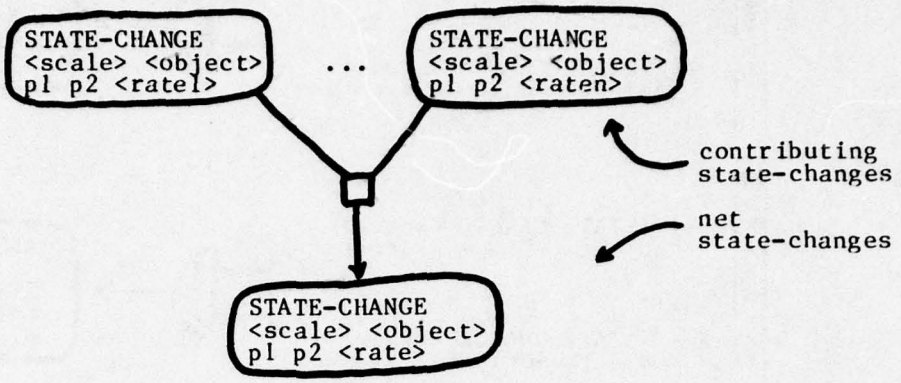
When stored as a fact in the data base, it represents the rate at which the value of the <object> moves along the <scale>. It also specifies starting and ending values for the state-change. Each <scale> <object> pair may have more than one state-change event associated with it.

The <start-pos> <end-pos> fields are not currently implemented and dummy constants are used in these positions. For example, (STATE-CHANGE TEMP AIR P1 P2 RATE) represents a change in the TEMP of AIR by "RATE" units for each time period. (The time mechanics are discussed in Section 4.5.1.11). As long as the state-change event is asserted the change in TEMP of the AIR by "RATE" units will occur for each time unit.

4.5.1.3. Rate Confluence Link

The rate confluence (RATE-CONFL) link is used to combine a group of contributing state-change events into a single net state-change event. The form of the contributing events and the net event are the same. That is, the <scale> <object> pairs are the same for each state-change event involved in the link. The link specifies those state-change events which influence the overall rate at which the <scale> <object> pair is changing. All contributing state-change events which are asserted (i.e. which exist as assertions in the data base) contribute their rate to the combined rate of the net state-change event. Those which are not asserted will be assumed to have a rate of 0. The net state-change will remain asserted and cause the incrementation of the ID-EVENT's value as long as at least one of its contributing state-changes remains asserted. The deassertion of any of the contributing state-changes

will remove its influence on the combined rate by causing the SC which is modeling the rate-confluence to become active, recompute a new rate, then become quiescent again.



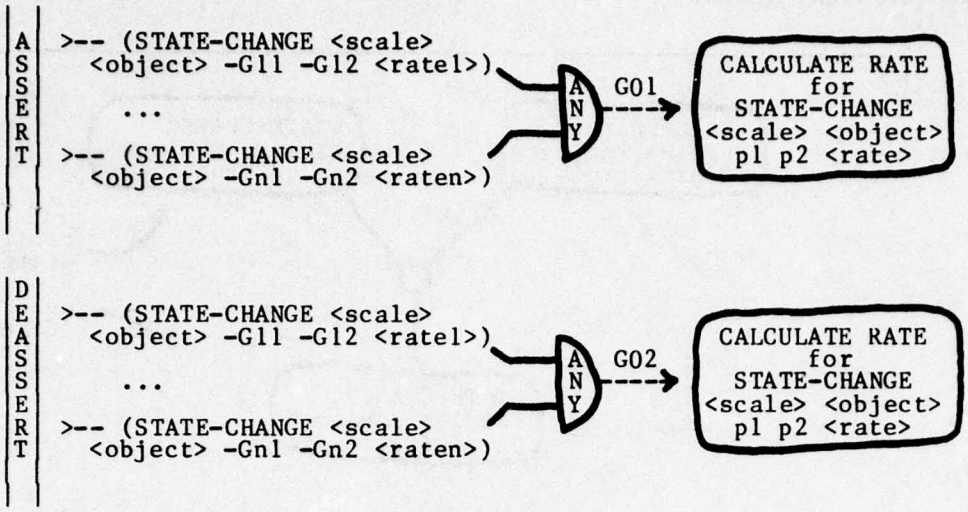
Display 42. RATE-CONFL Link

(In the following sections, the SC's created for each link will be illustrated in a display form. In our display conventions, each SC representing some potential event in the mechanism will be "eyeballing" a path to the data base. The path type (ASSERT or DEASSERT) will be located between the double bars on the left side of the display. The logical connective of the triggering events is to the right of the triggering events and the arrows out of the triggering events flow into this connector. On the far right side of the display is the body of the SC. These are the activities which will be performed when the SC is run. A reference name for the SC is located on the arrow into the body of the SC.

In some instances computables may be attached to some events in the triggering pattern. These computables are not "eyeballing" the data paths. They will be evaluated only when their accompanying event sees its pattern pass along the data path.

Two SC's are created for the RATE-CONFL link. They are identical in form (i.e. the trigger patterns and bodies are identical). However one is attached to the ASSERT path while the other is attached to the DEASSERT path. Their bodies are used to calculate the net rate and if necessary either ASSERT or

DEASSERT the net state-change event.



Display 43. RATE-CONFL Corresponding SC

4.5.1.4. Rate Calculation

Each rate node has a symbolic arithmetic computation associated with it. This association was established in the external representation. In the internal representation of the rate node, this symbolic arithmetic computation is stored as the property FUNCTION on the rate node.

The rate calculation is performed in a procedure named $RATE-CALC. It is activated by the assertion or deassertion of any contributing state-change. It is responsible for determining which contributing state-changes are asserted (if any) in a RATE-CONFL link. It creates a LISP ASSOC list of internal rate names and either the rate name's FUNCTION property or #0 depending on whether the corresponding contributing state-change is asserted or not. It then retrieves the FUNCTION property from the net state-change's rate node and then substitutes for the symbolic rate node the corresponding value from the ASSOC list. This newly created symbolic computation is then stored as property VALUE on the net state-change's rate node. $RATE-CALC then

determines if the net state-change is currently asserted. If it is not, then it is asserted. If none of the contributing state-changes are asserted ($RATE-CALC was activated by the deassertion of the last contributing state-change) then the net state-change is deasserted. The arguments to $RATE-CALC are: (1) The net state-change n-tuple and (2) a list of contributing state-change n-tuples. For example, the $RATE-CALC calling sequence for the example in Display 44 would be.

```
($RATE-CALC '(STATE-CHANGE HEIGHT BALL P7 P8 RATE4)
            '((STATE-CHANGE HEIGHT BALL -G1 -G2 RATE1)
              (STATE-CHANGE HEIGHT BALL -G3 -G4 RATE2)
              (STATE-CHANGE HEIGHT BALL -G5 -G6 RATE3)))
```

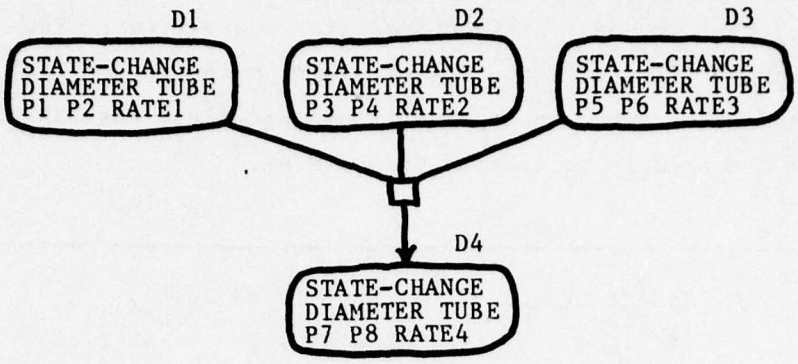
Display 44. $RATE-CALC Calling Sequence Example

A rate node has several properties associated with it. A property called NAME is the external represention name of the node, used to reference the rate in the CSA-graph. A property called FUNCTION is the symbolic arithmetic expression which was used to define the rate in the external representation. This property will be used by the rate calculator in place of the internal rate name for any asserted contributing state-change when the net rate is calculated. The net state-change's rate node has two additional properties. The property VALUE is the FUNCTION property on the net state-change with the contributing rate's FUNCTION property substituted for the symbolic name of the contributing rates. #0 is substituted for non-asserted contributing state-change rates. This property is an unevaluated symbolic arithmetic computation. The property NUM-VALUE has the previous evaluated VALUE for that rate node. When the state incrementor routine is called, the VALUE property of the net state-change rate is retrieved, evaluated, and then stored as the property NUM-VALUE. This is the rate used by the simulator in incrementing the state value.

4.5.1.5. Example of RATE-CONFL

Suppose there are three state-change events influencing a state-change.

The graph representation would be



[(RATE1 #3), (RATE2 #-3), (RATE3 #2)
and (RATE4 (#PLUS RATE1 RATE2 RATE3)]

Display 45. Another RATE-CONFL Example

The FUNCTION property associated with the four rates are: (RATE1 #3), (RATE2 #-3), (RATE3 #2) and (RATE4 (#PLUS RATE1 RATE2 RATE3)). Now assume that (STATE-CHANGE DIAMETER TUBE P1 P2 RATE1) is asserted. This will activate any SC's watching for the assertion of this pattern, in this case, D4. The body of the D4 SC consists of a call to a system function $RATE-CALC. $RATE-CALC executes by retrieving the FUNCTION property of RATE4, substituting for RATE1 its FUNCTION property (#3) and substituting #0 for RATE2 and RATE3 (their state-changes have not been asserted). This new symbolic expression (#PLUS #3 #0 #0) is then placed on the VALUE property of RATE4. D4 is then asserted by $RATE-CALC. Assume now that D2 becomes asserted. $RATE-CALC will then calculate a new VALUE property from the FUNCTION property and the asserted contributing state-change rates. The new VALUE property will be (#PLUS #3 #-3 #0). Also assume that D3 becomes asserted. The VALUE property will now be changed to (#PLUS #3 #-3 #2). Now assume That D1 is deasserted. The new VALUE property of RATE4 is (#PLUS #0 #-3 #2). Later, both D2 and D3 are deasserted. $RATE-CALC determined the new VALUE property, which is (#PLUS #0 #0 #0), and since none of the contributing state-changes are asserted, D4 is deasserted.

4.5.1.6. Threshold Events

A THRESHold event (THRESH-EVENT) is an event very similar to the ID-EVENT. It is incorporated into the CSA graph in only one place, as the state reached as threshold in a THRESH link (discussed in Section 4.5.1.8). A Threshold event is a 4-tuple of the form:

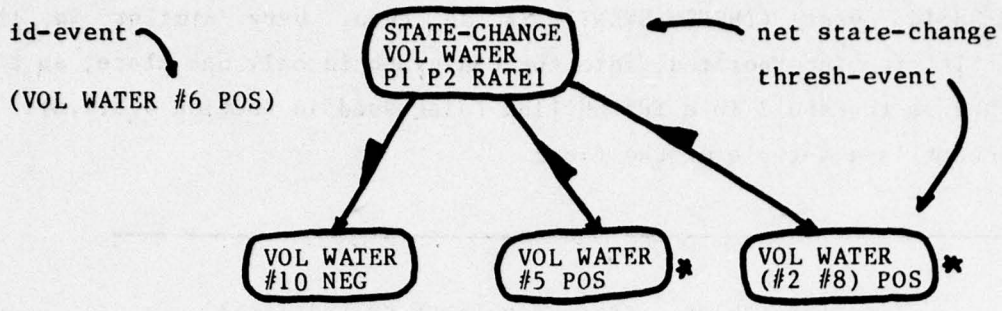
(<scale> <object> <thresh-points> <direction>)

Display 46. THRESHOLD Event Form

The <thresh-points> can either be a single value or a list of two values which represent the starting and ending values over some interval. The <direction> represents the type of movement the value must make across the <thresh-points> required by the event.

A THRESH-EVENT is never asserted or deasserted in the factual data base. A THRESH-EVENT becomes true (or false) when an appropriate value of the associated ID-EVENT is asserted. If the <thresh-points> is a single value then the THRESH-EVENT is true when the value of the ID-EVENT crosses over the <thresh-points> moving in the direction of <direction>. The THRESH-EVENT is false when the value of the ID-EVENT is on the opposite side of the <thresh-points>. For example, the THRESH-EVENT (VOL WATER #10 NEG) (the volume of water has reached the value 10 going in the negative direction) would be true if the ID-EVENT were (VOL WATER #6 NEG). It would be false for ID-EVENTs (VOL WATER #11 NEG), (VOL WATER #12 POS) and (VOL WATER #7 POS).

If the <thresh-points> is a list of two values then the THRESH-EVENT is true when the value of the ID-EVENT is anywhere between the two numbers moving in the direction of <direction>. It is false outside this range or when the direction is incorrect. For example, the THRESH-EVENT (POSITION INDICATOR (#10 #20) POS) will be true for the ID-EVENT (POSITION INDICATOR #15 POS). It will be false for ID-EVENTs (POSITION INDICATOR #9 POS), (POSITION INDICATOR #21 POS), and (POSITION INDICATOR #15 NEG).



* these thresh-events are true.

Display 47. Truth of THRESH-EVENTS

4.5.1.7. Converting Thresh-Events into Triggering Conditions

The THRESH-EVENT must be handled differently from other events when used in the trigger of an SC. Since the THRESH-EVENTs are not asserted or deasserted in the data base, their truth can only be ascertained by the current status of the ID-EVENT. In the trigger of any SC, the THRESH-EVENT must be true whenever the value of the corresponding ID-EVENT is within the range of the <thresh-points> moving in the correct direction. It must be false at all other times. The SC's watching for a THRESH-EVENT to become false are attached to the ASSERT path, not to the DEASSERT path as other links are. This is necessary because THRESH-EVENTs are not asserted or deasserted. They become true or false depending on the ASSERTION of an ID-EVENT. Hence they must always watch the ASSERT path. To accomplish this interaction, the SC creation routines use the strategies for building the trigger patterns for THRESH-EVENTS shown in the displays below.

THRESH-EVENT	TRIGGER PATTERN
(<scale> <object> #N POS)	(AND (<scale> <object> -X POS) (#GE -X #N))
(<scale> <object> #N NEG)	(AND (<scale> <object> -X NEG) (#LE -X #N))
(<scale> <object> #N ANY)	(AND (<scale> <object> -X -Y) (#GE -X #N))
(<scale> <object> (#N1 #N2) POS)	(AND (<scale> <object> -X POS) (#GE -X #N1) (#LT -X #N2))
(<scale> <object> (#N1 #N2) NEG)	(AND (<scale> <object> -X NEG) (#LE -X #N1) (#GT -X #N2))
(<scale> <object> (#N1 #N2) ANY)	(AND (<scale> <object> -X -Y) (#GE -X #N1) (#LT -X #N2))

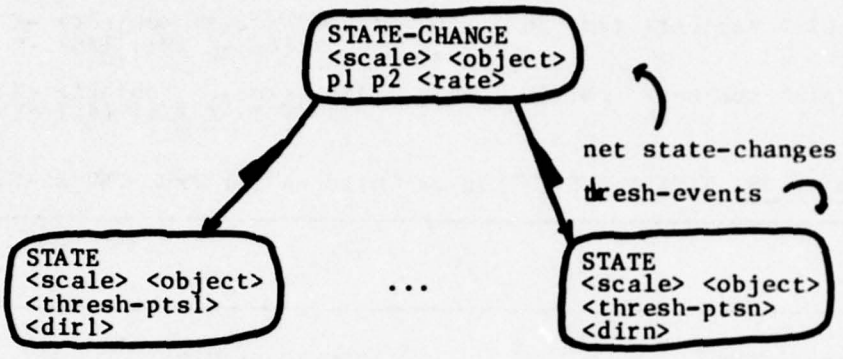
Display 48. THRESH-EVENT Trigger Patterns for True THRESH-EVENT

THRESH-EVENT	TRIGGER PATTERN
(<scale> <object> #N POS)	(AND (<scale> <object> -X -Y) (#LT -X #N))
(<scale> <object> #N NEG)	(AND (<scale> <object> -X -Y) (#GT -X #N))
(<scale> <object> #N ANY)	(AND (<scale> <object> -X -Y) (#LT -X #N))
(<scale> <object> (#N1 #N2) POS)	(OR (AND (<scale> <object> -X -Y) (#LT -X #N1)) (AND (<scale> <object> -X -Y) (#GE -X #N2)))
(<scale> <object> (#N1 #N2) NEG)	(OR (AND (<scale> <object> -X -Y) (#GT -X #N1)) (AND (<scale> <object> -X -Y) (#LE -X #N2)))
(<scale> <object> (#N1 #N2) ANY)	(OR (AND (<scale> <object> -X -Y) (#LT -X #N1)) (AND (<scale> <object> -X -Y) (#GE -X #N2)))

Display 49. THRESH-EVENT Trigger Patterns for False THRESH-EVENT

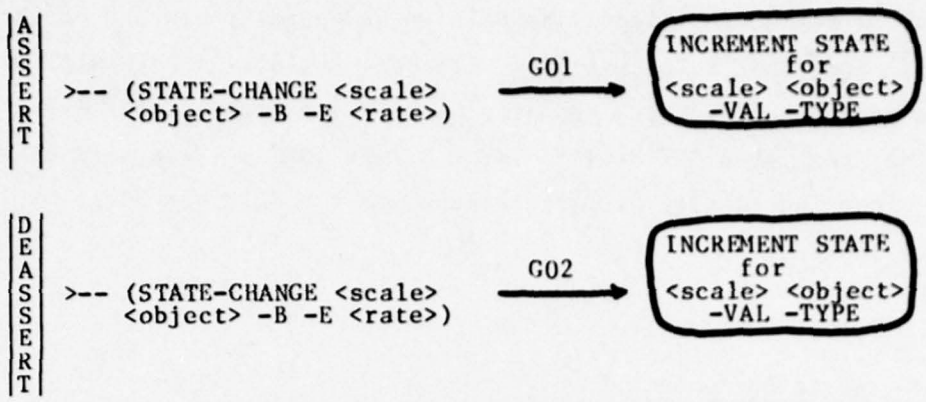
4.5.1.8. Threshold Link

The THRESHold (THRESH) link is used to connect a net state-change event with its corresponding THRESH-EVENTs. When the state-change is asserted, the SC created in correspondence to the link will alter the value of the ID-EVENT. The THRESH-EVENTs become true when the proper ID-EVENT is asserted. This allows the SC's watching the links out of the THRESH-EVENTs to awaken.



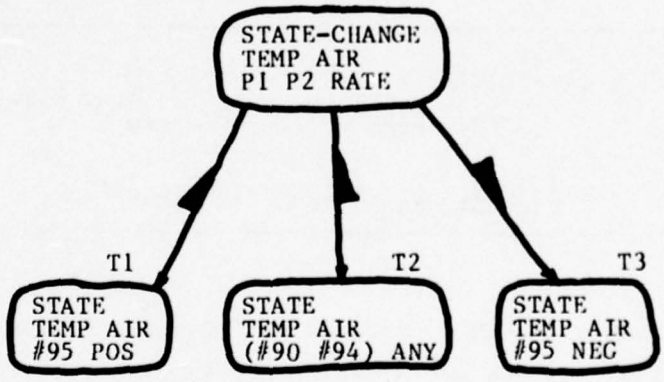
Display 50. THRESH Link

Display 51 shows the two SC's created for the THRESH link. They are identical in their trigger patterns and bodies. However, each is required to watch a separate path into the data base. The trigger pattern is the net state-change event and the body is a call on the procedure used to increment the ID-EVENT associated with the state-change. The procedure used to perform this incrementing is named $S-INCR and is discussed in detail in the next Section 4.5.1.9.



Display 51. THRESH Corresponding SC

For example, suppose the state-change event (STATE-CHANGE TEMP AIR P1 P2 RATE1) is linked to the following THRESH-EVENTs: T1 is (TEMP AIR #95 POS), T2 is (TEMP AIR (#90 #94) ANY), and T3 is (TEMP AIR #95 NEG).



Display 52. THRESH Link Example

Suppose that the current VALUE property of RATE1 is #4 and suppose the starting ID-EVENT is (TEMP AIR #89 POS). On the first incremented change to the ID-EVENT, the ID-EVENT becomes (TEMP AIR #93 POS). T2 is now true and all SC's watching for the truth of T2 are awakened. On the next change to the

ID-event the value becomes (TEMP AIR #97 POS). This will cause T1 to become true and T2 to become false. The SC's watching for the truth of T1 will be awakened and the SC's watching for the falsity of T2 will be awakened. Suppose next that the VALUE property of RATE1 is altered to #-5. The next value of the ID-EVENT will be (TEMP AIR #92 NEG). This will cause the SC's watching for the falsity of T1 to awaken and the SC's watching for the truth of T3 to awaken.

4.5.1.9. State Incrementor

The body of the SC created for the THRESH link consists of a call on a procedure called $S-INCR. $S-INCR is a function which computes the actual changes to the ID-EVENT's value. $S-INCR has three arguments: (1) the ID-EVENT pattern which is to be altered (variables replace the rate and direction); (2) the trigger pattern (variables replace the <st-pos> <end-pos> and the <rate>); and (3) the SC names used to place this $S-INCR function on the queue of SC's wanting to run. For example, the function call for the example of Display 52 above would be.

```
($S-INCR '(TEMP AIR -VAL -TEMP)
         '(STATE-CHANGE TEMP AIR -B -E RATE)
         '(<SC1-name> ... <SCn-name>) )
```

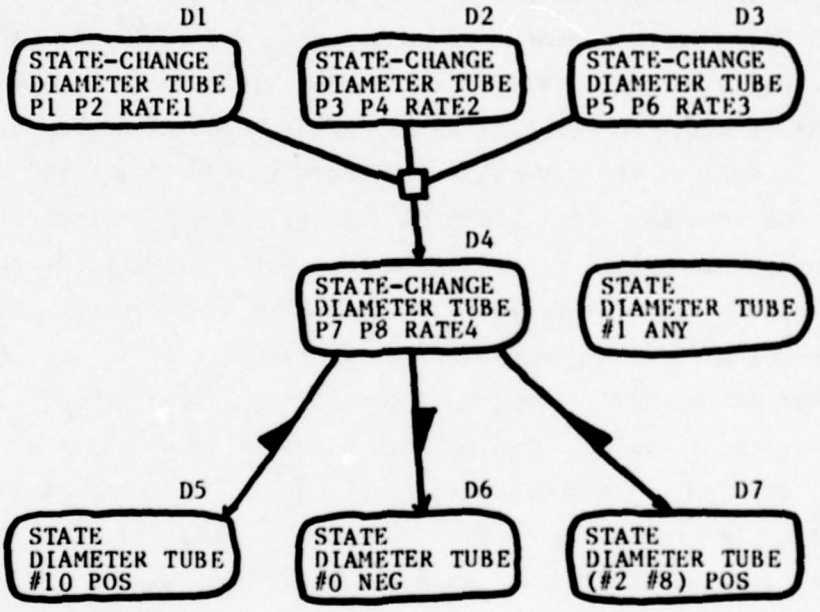
Display 53. $S-INCR Procedure Call

The processing of $S-INCR proceeds as follows. If the rate is non-zero, $S-INCR determines if the net state-change is asserted. If it is, the VALUE property of the state-change rate node is retrieved and evaluated to a CSA-number. This CSA-number is then stored as the NUM-VALUE property on the rate node. The ID-EVENT pattern is used to fetch the current value from the data base. The NUM-VALUE property of the rate is added to the value of the ID-EVENT and the sign of NUM-VALUE is used to determine the direction of the change. The old ID-EVENT is erased. This new ID-EVENT value is then asserted. This is recorded as a change to the ID-EVENT. The $S-INCR function

is then queued onto the list of pending procedures. If however the net state-change was not asserted in the factual data base then $S-INCR does nothing. $S-INCR will not queue itself on the run queue and hence no further alterations of the ID-EVENT will be done until the net state-change is reasserted.

4.5.1.10. Statechange Links Example

Suppose there are three contributing state-change events influencing a net state-change. Also suppose that there are three THRESH-EVENTS monitoring the net state-change. The graph representation would be



[(RATE1 #3), (RATE2 #-3), (RATE3 #2)
and (RATE4 (#PLUS RATE1 RATE2 RATE3))]

Display 54. STATE-CHANGE Link Example for Diameter of Tube

The property FUNCTION associated with the four rates are: (RATE1 #3), (RATE2 #-3), (RATE3 #2) and (RATE4 (#PLUS RATE1 RATE2 RATE3)). The ID-EVENT's initial value is #2 (i.e. DIAMETER TUBE #2 ANY)). Now assume that D1 is asserted. This will activate the SC watching for the assertion of this

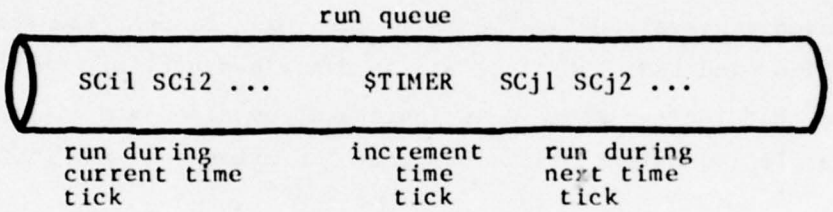
pattern. The body of this SC is $RATE-CALC. $RATE-CALC retrieves the FUNCTION property of RATE4, substituting for RATE1 its FUNCTION property (#3) and substituting #0 for RATE2 and RATE3 (their state-changes have not been asserted). This new symbolic expression (#PLUS #3 #0 #0) is then placed on the VALUE property of RATE4. D4 is then asserted by $RATE-CALC. This will trigger the SC watching for the assertion of this net state-change. $S-INCR is then run. $S-INCR will verify that D4 is asserted; it will then retrieve the VALUE property of RATE4 and evaluate it. This value #3 is then stored on the NUM-VALUE property of RATE4. The ID-EVENT is then changed to (DIAMETER TUBE #5 POS). This reflects the increase of the value of the previous ID-EVENT by the NUM-VALUE. $S-INCR then places itself back on the run queue. Assume now that D2 becomes asserted; $RATE-CALC will then calculate a new VALUE property from the FUNCTION property and the asserted contributing state-change rates. The new VALUE property will be (#PLUS #3 #-3 #0). Also assume that D3 becomes asserted. The VALUE property will now be changed to (#PLUS #3 #-3 #2). When $S-INCR is next run, the VALUE property of RATE4 will be (#PLUS #3 #-3 #2). This will be evaluated by $S-INCR and stored on the NUM-VALUE property of RATE4. #2 will then be used to increment the ID-EVENT to (DIAMETER TUBE #7 POS). $S-INCR then places itself back on the run queue. Now assume that D1 is deasserted, the new VALUE property of RATE4 is (#PLUS #0 #-3 #2). When $S-INCR runs this new VALUE property is obtained and the new NUM-VALUE property is #-1. The ID-EVENT is again incremented by #-1 changing it to (DIAMETER TUBE #6 NEG). It now has started to decrease in value. $S-INCR again places itself on the run queue. Later, both D2 and D3 are deasserted. $RATE-CALC determined the new VALUE property which is (#PLUS #0 #0 #0) and since none of the contributing state-changes are asserted, D4 is deasserted. Now when $S-INCR is run, it notices that D4 is no longer asserted. It will do no processing of the VALUE property list. It will also not place itself on the run queue. Hence until one of the contributing state-change events is again reasserted, the ID-EVENT will remain at (DIAMETER TUBE #6 NEG).

4.5.1.11. Time in the Simulator

Time can be thought of as the beating of a clock with some time units.

The rates used in the state-change events are with respect to this "arbitrary" time unit. A function $TIMER is used to increment the time units. On each processing of $TIMER, time is incremented, the time is printed, and it then queues itself on the list of pending procedures.

All assertions, deassertions, and changes to the events within a time unit will be considered as happening concurrently. SC's when invoked will place their bodies on a list of pending procedures. They may either queue their bodies or stack them. If the bodies are stacked then the action takes place during the current time period. If the bodies are queued, the action will occur in the next time period.



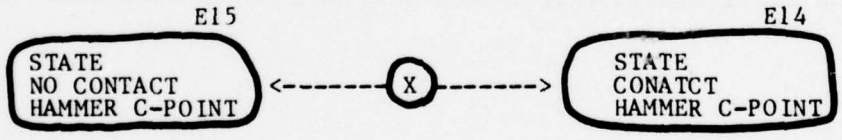
Display 55. Pictorial Representation of $TIMER on RUN QUEUE

4.5.2. Non-Statechange-Related Links

4.5.2.1. Hiding and Unhiding SC's

Hiding an SC involves the temporary "removal" of the SC from its data path. Unhiding is eliminating the temporary "removal" of the SC from its data path; i.e., the SC is returned to original status. Normally when an SC is unhidden and some data base activity coincides with one of its trigger pattern events, the SC polls its remaining trigger pattern events to see whether it should activate itself. However if an SC is hidden, then a data base activity on one of its trigger pattern events will not cause the SC to poll the remaining trigger pattern events.

A simple example using the ANTAG link will demonstrate the importance of the HIDING and UNHIDING of SC's. As described earlier, this link is used to indicate that the two state descriptions it couples are mutually exclusive. For example, consider link [L6] from the example of the OSCILLATOR [Section 4.2]



Display 56. ANTAG Link <L6>

```
**OBJECT G65 (SC)**
   SC-PATTERN:   (+ 1 (NO CONTACT HAMMER C-POINT))
   SC-BODY:      (LAMBDA (ARGX)
                         ($HIDE-SIMU 'G68)
                         ($UNHIDE-SIMU 'G67)
                         ($UNHIDE-SIMU 'G66)
                         ($ERASE-SIMU '(CONTACT HAMMER C-POINT)))

**OBJECT G66 (SC)**
   SC-PATTERN:   (+ 1 (NO CONTACT HAMMER C-POINT))
   SC-BODY:      (LAMBDA (ARGX)
                         ($HIDE-SIMU 'G67)
                         ($UNHIDE-SIMU 'G68)
                         ($UNHIDE-SIMU 'G65)
                         ($STORE-SIMU '(CONTACT HAMMER C-POINT)))

**OBJECT G67 (SC)**
   SC-PATTERN:   (+ 1 (CONTACT HAMMER C-POINT))
   SC-BODY:      (LAMBDA (ARGX)
                         ($HIDE-SIMU 'G66)
                         ($UNHIDE-SIMU 'G65)
                         ($UNHIDE-SIMU 'G68)
                         ($ERASE-SIMU '(NO CONTACT HAMMER C-POINT)))

**OBJECT G68 (SC)**
   SC-PATTERN:   (+ 1 (CONTACT HAMMER C-POINT))
   SC-BODY:      (LAMBDA (ARGX)
                         ($HIDE-SIMU 'G65)
                         ($UNHIDE-SIMU 'G66)
                         ($UNHIDE-SIMU 'G67)
                         ($STORE-SIMU '(NO CONTACT HAMMER C-POINT)))
```

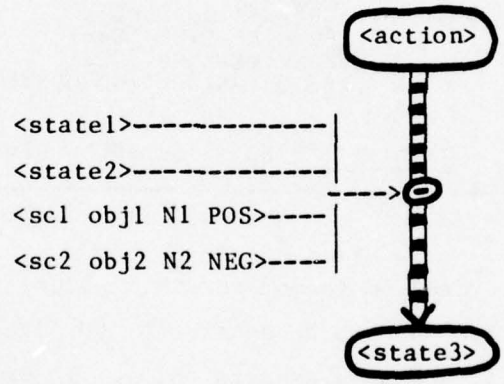
Display 57. SC's for ANTAG Link <L6>

Four SC's were created to monitor this link. SC-G65 watches for the assertion of {E15} and then deasserts {E14}. SC-G68 watches for the deassertion of {E14} and then asserts {E15}. It is easy to see the type of circular situation which may evolve. {E15} is asserted causing SC-G65 to deassert {E14} causing SC-G68 to assert {E15} causing SC-G65 to deassert {E14} causing ... The hiding of SC-G65 and SC-G68 by SC-G65's first execution will eliminate this circular situation. Now when {E15} is asserted causing SC-G65 to hide SC-G65 and SC-G68 and deassert {E14}, no further activity will occur since SC-G68 couldn't react to the deassertion of {E14}. Somehow, SC-G65 and SC-G68 must be unhidden at some later time or else this section of the mechanism will no longer be involved in the simulation. SC-G67 and SC-G66 have the responsibility of unhiding SC-G65 and SC-G68. SC-G67 watches for the deassertion of {E15} and then asserts {E14}, while SC-G66 watches for the assertion of {E14} and then deasserts {E15}. Now when {E14} is asserted this

activates SC-G66 which causes {E15} to be deasserted. But it also hides SC-G66 and SC-G67 as well as unhiding SC-G68 and SC-G65. Now when {E15} is asserted the correct SC's can be awakened.

4.5.2.2. Continuous Causal Link

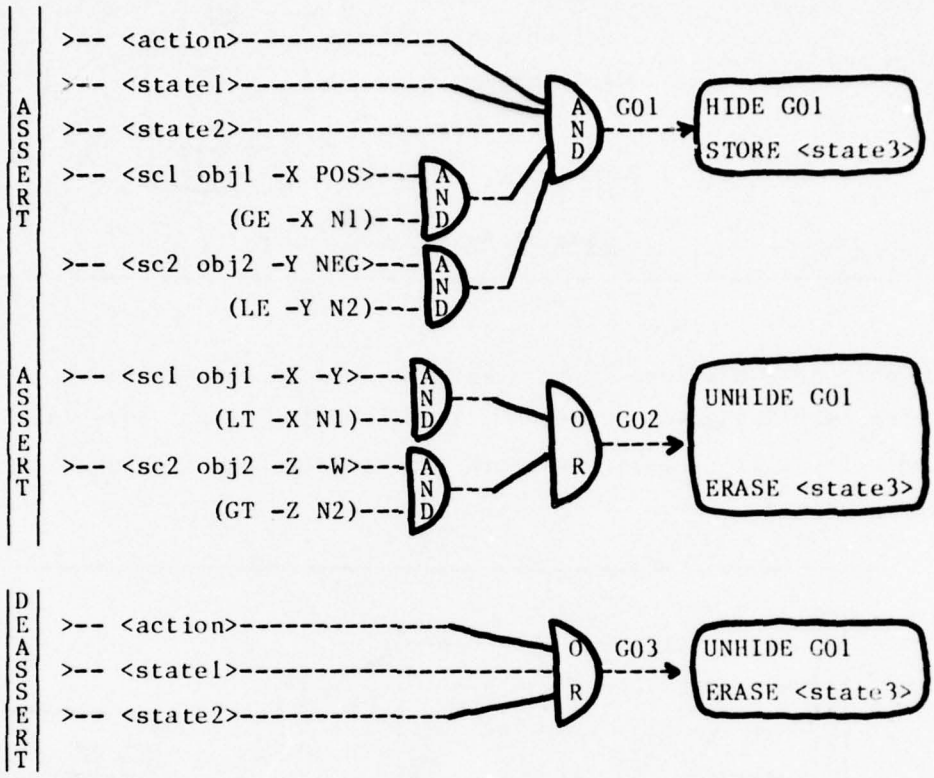
The continuous causal (C-CAUSE) link connects an action event and a state or state-change event. The link represents a continuous causal relationship between the action and the state or state change. Gating events may also be included in the link. The truth of all the gating events and the occurrence of the action are necessary for the caused state to remain asserted. If any of the gates or the action itself is deasserted in the data base, the caused state must also be deasserted.



Display 58. C-CAUSE Link

There are three possible SC's created for this link. G01 which is always created is used to assert the caused state. G03 is also always created and is responsible for the deassertion of the caused state when either the causing action (or tendency) or one of the normal gating conditions is deasserted. G02 is a specialized SC which is created only when one or more of the gating conditions is a THRESH-EVENT. In this case the SC watches for the ID-EVENT to pass along the ASSERT path and then determines if the computable associated

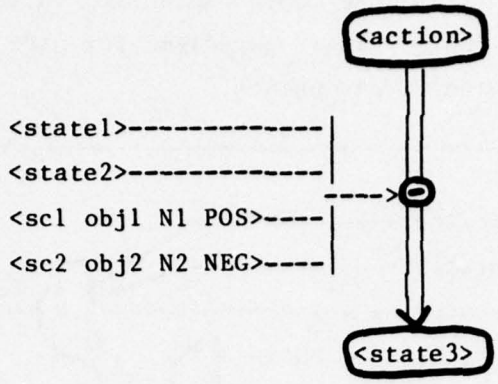
with the THRESH-EVENT is true or false. If it is true, the caused state is deasserted. Notice that the computable used in G02 is the negative of the computable used in G01. This paradigm for the THRESH-EVENTS is used throughout the following SC discussion.



Display 59. C-CAUSE Corresponding SC

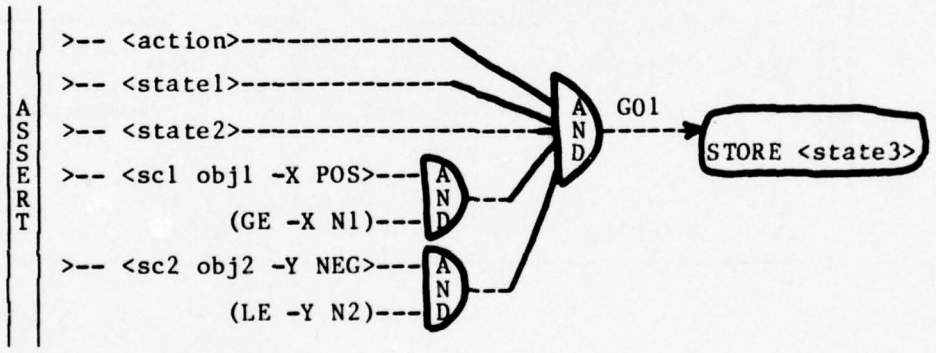
4.5.2.3. One Shot Causal Link

The one-shot causal (OS-CAUSE) link differs from the C-CAUSE link in that once the caused state has been asserted, the deassertion of any of the trigger pattern events has no effect on the caused state. No hiding of the SC is necessary in this link.



Display 60. OS-CAUSE Link

In this case only one SC is created. When the action and all the gating conditions are asserted in the factual data base, the caused state is asserted. It will remain asserted regardless of the condition of its triggering events.

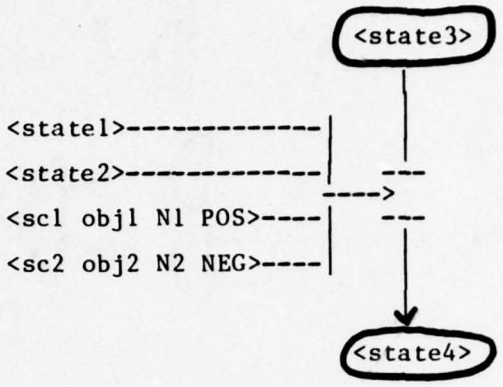


Display 61. OS-CAUSE Corresponding SC

4.5.2.4. State Couple Link

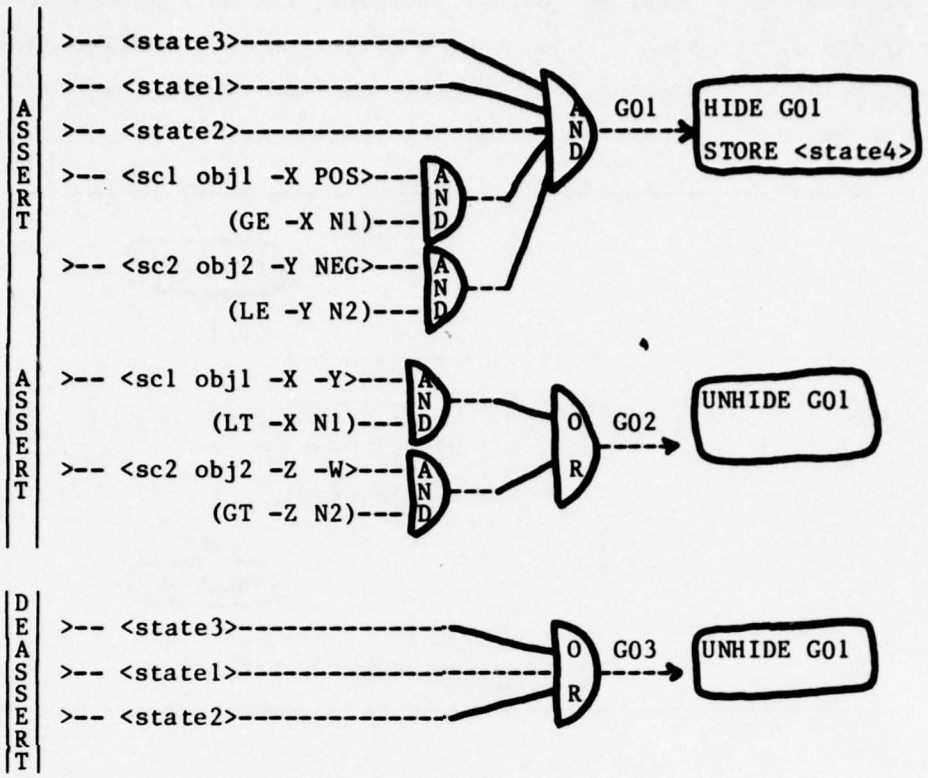
The state coupling (S-COUPLE) link represents an "unknown" causal relationship between a implying event and a implied event. This link is a

unidirectional link. The truth of the implying event and the gates implies the truth of the implied event. However, the deassertion of the implying event or any of the link's gates has no effect on the implied event. Also, the assertion or deassertion of the implied event has no effect on the assertion or deassertion of the implying event.



Display 62. S-COUPLE Link

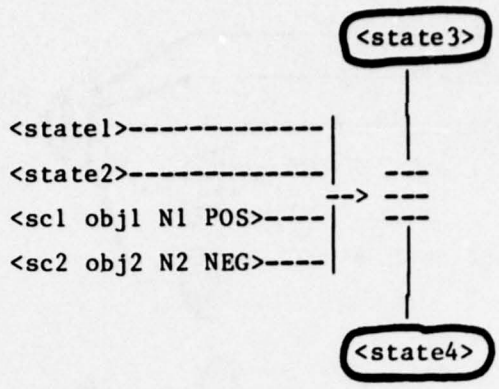
Here again three SC's are created for this link. G01 is responsible for the assertion of the implied state. This SC is hidden and remains hidden as long as there is no alteration to the triggering events. G02 and G03 watch for the deassertion of a triggering event or the falsity of a THRESH-EVENT and then unhide G01.



Display 63. S-COUPLE Corresponding SC

4.5.2.5. State Equivalence Link

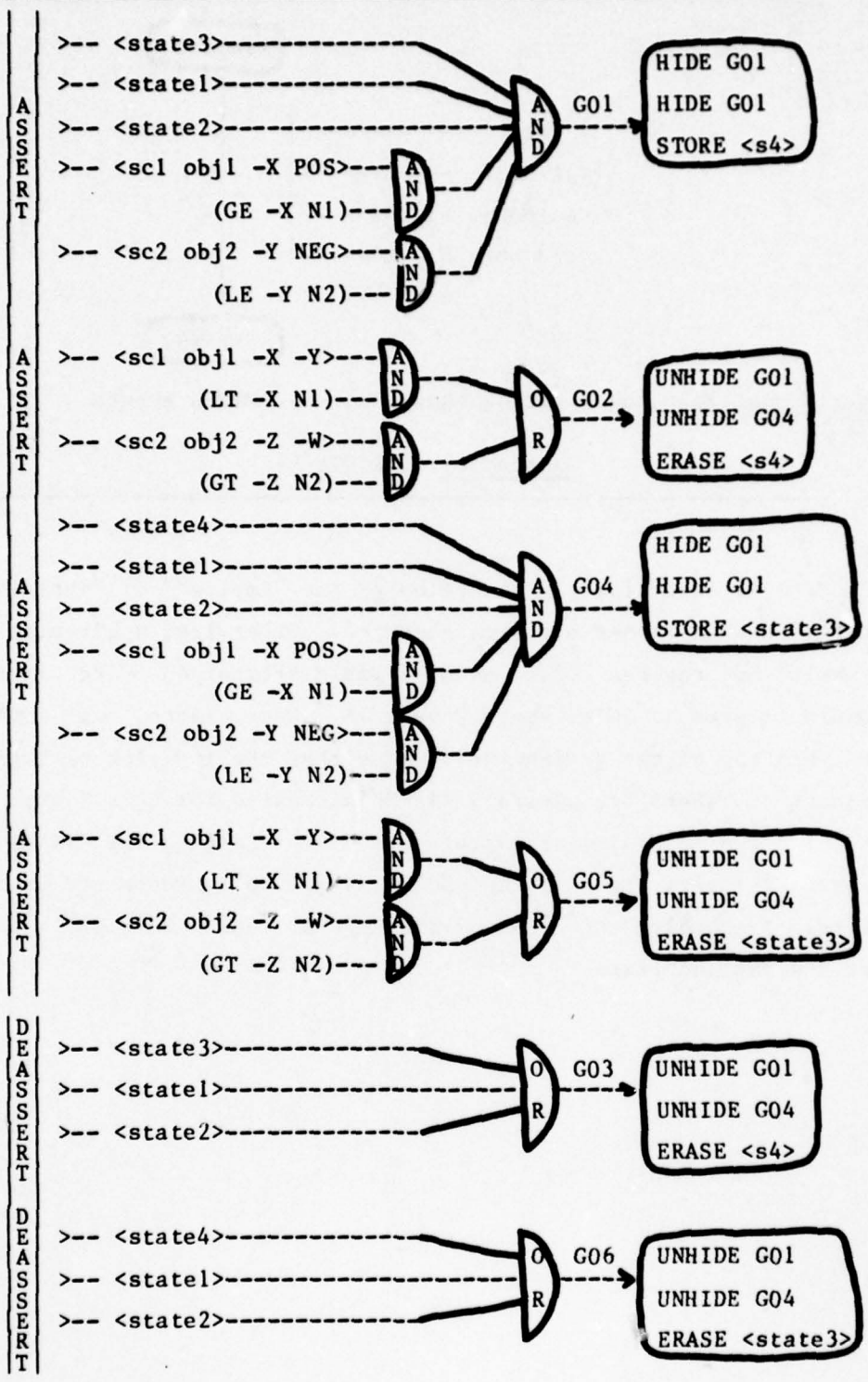
The state equivalence (S-EQUIV) link connects two events which are equivalent. This link is used to change focus or level of descriptive resolution. The link is bi-directional. The events connected by this link can each be the implying event or the implied event. The assertion of either event and all the gating events implies the assertion of the other event. The deassertion of either event or any of the gating events implies the deassertion of the other event.



<state3> and <state4> should not be THRESH events.

Display 64. S-EQUIV Link

The SC's watching for the assertion of the "implied" or "implying" event must be hidden when either event is asserted. Otherwise, a circular assertion chain would be entered (i.e. event 1 would trigger an SC to assert event 2 which would trigger an SC to assert event 1). Once hidden, an EQUIV SC is unhidden when any of the gating events or either the implying or implied event is deasserted. There are possibly six SC's created for this link. There are two sets of three SC's similar to those in the STATE-COUPLE link with two exceptions. Firstly, the asserting SC's must be both hidden and unhidden from both sets. Secondly, the deasserting SC's (G02, G03, G05 and G06) must deassert the implied state.



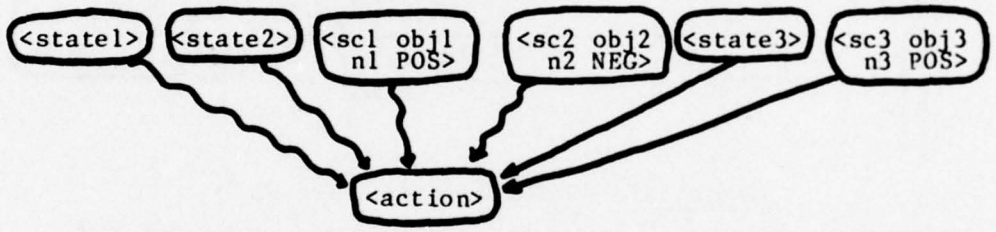
Display 65. S-EQUIV Corresponding SC

4.5.2.6. Enablement Links

The one shot enablement (OS-ENABLE) link connects a state to an action or tendency. Its sematics are that the state must be asserted before the action or tendency can be asserted. Once the action has been asserted, the OS-ENABLing state's deassertion has no effect on the enabled action.

The continuous enablement (C-ENABLE) link is similar to the OS-ENABLE link except that the enabling state's deassertion will cause the deassertion of the enabled action.

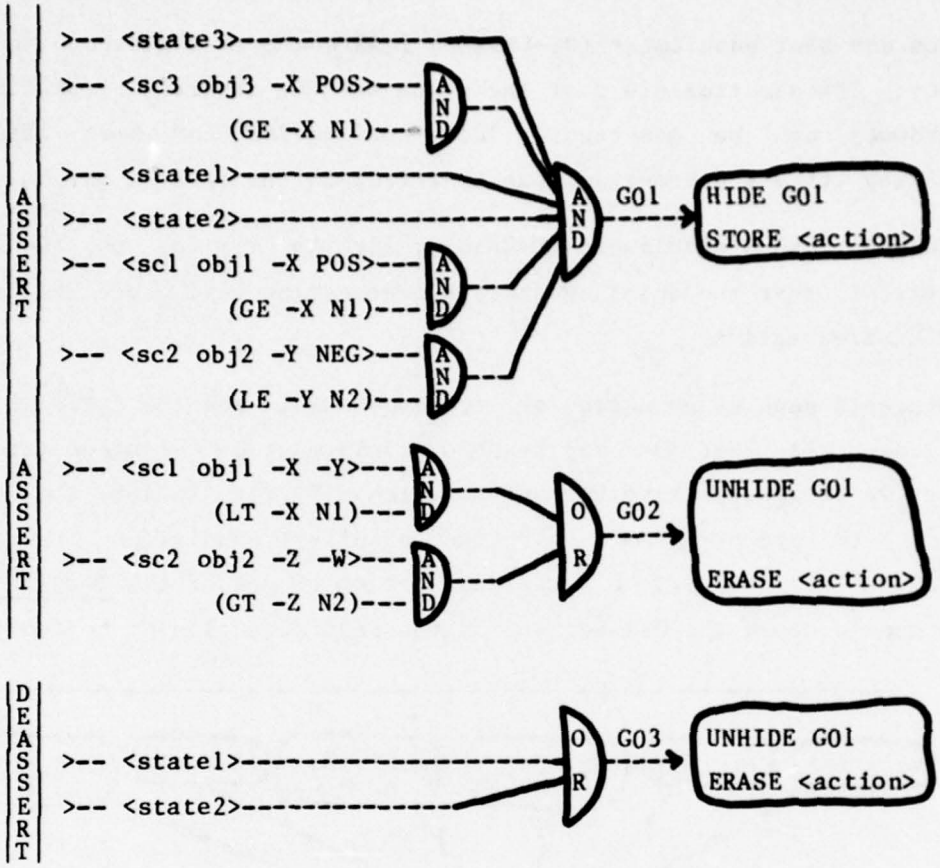
Since in general an action or tendency will require several enabling conditions, all OS-ENABLE and C-ENABLE links must be collected into a single conjunctive trigger pattern for one SC which will fire (allow the action or tendency to proceed) when all the enabling conditions are in effect simultaneously. However, only the deassertion of one of the C-ENABLE enabling events should cause the deassertion of the enabled action or tendency.



Let <state1>, <state2>, <sc1 obj1 n1 POS>, and <sc2 obj2 n2 NEG> be C-ENABLE enabling events.
Let <state3> and <sc3 obj3 n3 POS> be OS-ENABLE enabling events.

Display 66. ENABLEMENT Link

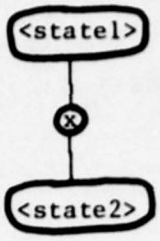
The enabling link also has three possible SC's created for it. G01 watches for the truth of all the enabling conditions and then asserts the action (or tendency). G02 and G03 have as their trigger patterns only the C-ENABLEing states. They will deassert the action (or tendency) when either one of the C-ENABLEing THRESH-EVENTS becomes false or one of the C-ENABLEing states is deasserted.



Display 67. ENABLE Corresponding SC

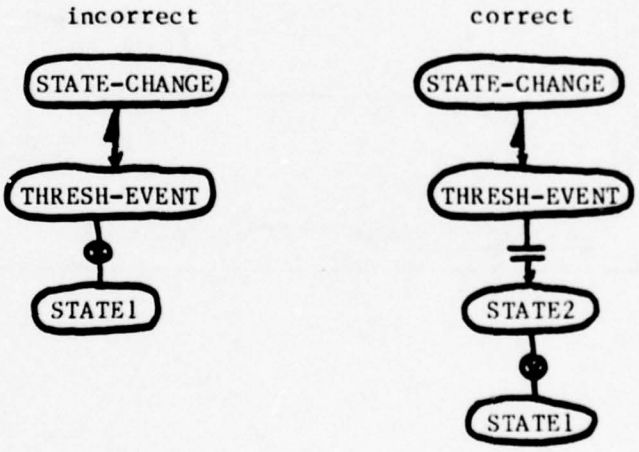
4.5.2.7. Antagonism Link

The antagonism (ANTAG) link connects two events which are mutually exclusive. The truth of one implies the falsity of the other and conversely.



Display 68. ANTAG Link

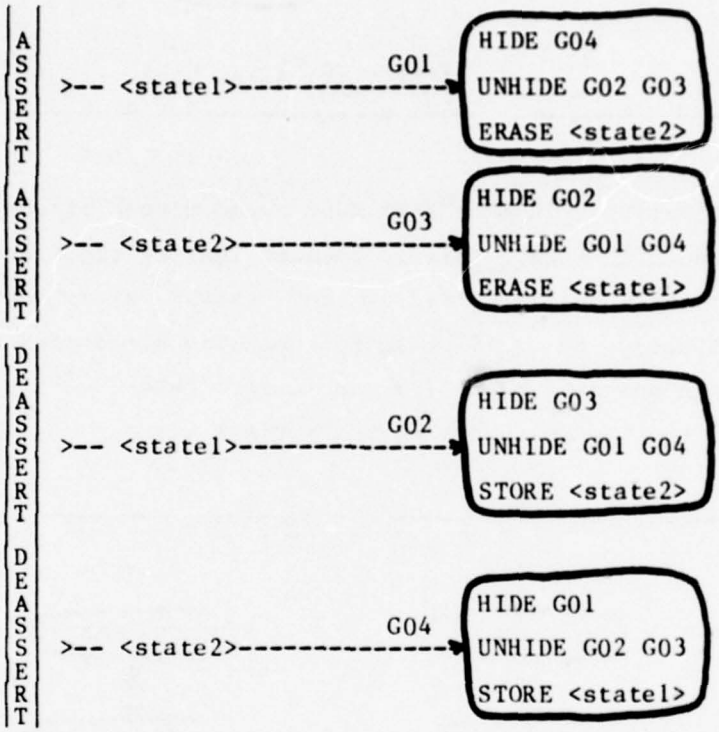
Syntactically, a THRESH-EVENT must never appear directly as one of the states of an ANTAG link. This is because THRESH-EVENTs can neither be erased from the data base nor falsified by any method except the change of the ID-event (Display 69). If it is conceptually neccessary that a THRESH-EVENT be involved in an antagonism link, an intermediate S-COUPLE link should be used between the THRESH-EVENT and the ANTAG link.



Display 69. Syntactic CSA Graph form with THRESH-EVENT and ANTAG-LINK

Four SC's come to be associated with each ANTAG link. Two SC's (one for each event) are required to watch the ASSERT path and deassert the other event when triggered. The other two SC's (one for each event) are required to watch

the DEASSERT path and assert the other event when triggered. Hiding and unhiding of the SC's is again required to prevent a flurry of circular activity, for if the correct SC were not hidden then the assertion of statel (Display 68) would cause the deassertion of state2 which would cause the assertion of statel, and so on.



Display 70. ANTAG Corresponding SC

4.6. User Notes

4.6.1. Normal Termination

The simulator has two methods of entering termination mode. The first is when the mechanism has conceptually completed. The second is when a

predetermined maximum time, specified by the user at the beginning of the simulation, has been surpassed.

When the simulator enters termination mode, the user is given the option of making alterations to the data base and global LISP variables or to allow the simulator to terminate. The program will request with

LIST ADDITIONAL ACTIVITIES:

the user to input a list of LISP S-EXPRESSIONs which the simulator will evaluate. If the list is not null, the list is appended to MECH-AGENDA, $TIMER is queued on MECH-AGENDA, and the normal processing is continued. If the list is null then the simulator terminates by printing the factual data base to indicate the final world after the simulation. The assertion of an event in the S-EXPRESSION would be done with ($STORE-SIMU '<event>); the deassertion by ($ERASE-SIMU '<event>). Normal LISP S-EXPRs can also be included in this list. This can be used to alter the value of a LISP atom, for example.

The first method of entering the termination mode is upon the conceptual completion of the simulation of the mechanism. This is accomplished when no further assertions, deassertions, or changes to the data base are pending. Thus the SC's created for the mechanism have no further data base activity to which to react. This situation is determined by $TIMER. When $TIMER is being processed and MECH-AGENDA is null then the simulator enters termination mode.

The second method of termination is used when a mechanism's simulation may take a long time before entering termination mode by the first method (if it enters at all). For instance, the OSCILLATOR given in the initial example will never enter termination mode by the first method. The user is queried upon initiating the simulation with the following message:

MAX TICK COUNT (T = INFINITY)

The user should now type in a number or a 'T'. If a number is typed, then when the simulator's clock reaches that number the simulator will enter termination mode. If 'T' is typed, the simulator will assume it is to run for an infinite time and only enter termination mode by the first method.

The user can include in the list of additional activities a reference to

AD-A034 194 MARYLAND UNIV COLLEGE PARK DEPT OF COMPUTER SCIENCE F/G 9/2
THE CAUSAL REPRESENTATION AND SIMULATION OF PHYSICAL MECHANISMS--ETC(U)
NOV 76 C RIEGER, M GRINBERG N00014-76-C-0477
UNCLASSIFIED TR-495 NL

2 OF 2
AD-A
034 194



END
DATE
FILMED
2-16-77
NTIS

the atom ::QUITTICK. This will alter the maximum time already given, and thus the mechanism's simulation may continue for additional ticks. If the value of the atom ::QUITTICK is not increased, but a list of other activities is given, then the simulator will process one additional tick and then again enter termination mode.

4.6.2. User Interaction

The simulator is activated by

($SIMULATE-MECH '<name>)

where <name> is the internal CSA-atom representing a mechanism. The simulator now starts executing automatically, asking the user only for a maximum time and if additional processing is to be done at the end.

The simulator first pushes the context-level. Each link involved in the mechanism has its set of SC's appended to a global atom PLANT-LIST. When all the links have been processed, the SC's on PLANT-LIST are actually attached to their appropriate data paths. The context-level is again pushed (it is now at level 2). The simulator requests

MAX TICK COUNT (T = INFINITY)

The user now inputs the maximum number of clock units the simulator will run before entering termination mode. The user may input a number representing the maximum count or 'T' which specifies that the simulation is to continue until the mechanism has conceptually been completed.

The simulator then takes the list of INITIAL-WORLD events from the internal representation of the mechanism and asserts these events in the data base. The assertion of some of these events may awaken SC's. If any SC is invoked its body is placed on MECH-AGENDA. The MECH-AGENDA queue is then processed to get the mechanism into the actual starting configuration. The TRIGGER events from the internal representation are asserted in the data base. Additional SC's should be awakened and those invoked also place their bodies on MECH-AGENDA. $TIMER is then placed on MECH-AGENDA. The MECH-AGENDA list is then evaluated, one item at a time. If the evaluation of the item invokes

additional SC's their bodies are then placed on MECH-AGENDA. The simulator continues in this mode of evaluating the first item on MECH-AGENDA and adding SC bodies to MECH-AGENDA until either the maximum time has been reached or MECH-AGENDA becomes empty. At this point the simulator requests

LIST ADDITIONAL ACTIVITIES:

The user can now give it a list of S-EXPR which are stacked on MECH-AGENDA. The simulator then checks to see if the list the user gave is null. If it is then the simulation terminates. Otherwise, the simulator continues in the evaluation of MECH-AGENDA. When the simulator terminates, the CSA-EVENTS which are asserted in the data base are printed under the heading of FINAL WORLD.

5. Next Phases of the Mechanisms Lab

5.1. Interfacing with General World Knowledge

Although the simulator has been designed to deal with cause-effect mechanisms in the comparatively simple physical domain, it represents a general approach to plan simulation in other more complex (e.g. social) domains. By extending the simulator's repertoire of cause-effect links to include those CSA links which represent human actions (intention, motivation, etc.) we will eventually have, not simply a mechanisms lab, but a Plan Simulation Lab capable of testing the outputs of the plan synthesizer for social and psychological as well as physical plans.

Of course, when we progress to the more complex domains, the simulator will inevitably be called upon to incorporate knowledge which has not been given to it explicitly. Clearly, for instance, we will want a general population of inference-makers to observe the simulation as it progresses, interjecting new insights into the factual database from which the simulator derives its raw data. In principle, this is also requisite to a good physical mechanisms simulator which has been asked to make a careful diagnosis of a proposed mechanism; since the designer of the mechanism may not have considered all contingencies (say, the magnetic field of one component jams another because of some obscure layout or timing problem), it might in general be necessary to call on knowledge (e.g. tendencies involving magnetism) not explicitly referenced in the design. This is analogous to making inferences in the social plan synthesizer.

A mechanisms lab with access to the general database of cause-effect schemata which describe the myriad physical principles found in mechanisms would also make it possible to perform semantic checks on a mechanism as its description is entered into the system. (This would be very attractive since we seem as yet to have no effective means for guiding a human's description along lines deemed theoretically correct by the system. A system with preconceptions about, e.g., what each physical tendency required as enablements and gates in order to function, would be able to cooperate interactively with the human as a description is entered.

Therefore, one of our immediate goals is to give the simulator access to the database of CSA cause-effect schemata which describe the general principles upon which all mechanisms are based. (The structure and organization of this database are described in [R1]). Doing this will enable the simulator to do more than it was requested to do"; such an ability will also be crucial to the next immediate goal of the research: a Mechanisms Inventor.

5.2. Mechanism Invention

If, as we have argued, a mechanism is a frozen exemplar of general human problem solving, then it ought to be possible to apply the CSA problem solver to the task of designing a mechanism to accomplish some task - to invent a particular mechanism, drawing on general knowledge about cause and effect.

Centralmost to the existing CSA plan synthesizer (described in [R1], and in more detail in [L1]) is the notion of appropriateness of a strategy in a given context. Clearly it would be possible to propose Rube Goldberg mechanisms ad-infinitum. However, if, e.g., a subgoal is to cause a quick, short rise of an inflammable object's temperature, a heater coil is probably a better approach than a ruby laser!

Since we already have a running plan synthesizer which will provide the nucleus of the Mechanism Inventor, our main focus is now on amassing a large enough database of physical principles, expressing them as CSA graphs in the terms required by the plan synthesizer. This will enable us to begin experimenting with mechanism "invention" of items at the level of complexity of, say, a gas forced air heating system or a telephone. The repertoire will include the cause-effect specifications of physical laws, expressed as metaphorical actors and tendencies. The numerous strategies will address generic problems of heat production, mechanical to electrical conversion, electrical to mechanical conversion, amplification (electrical and physical motion) strategies, fluid control strategies (valves, overflow controls), and so forth. Given the input description of a task, the synthesizer will do its best to construct a functional and appropriate mechanism which accomplishes the task. It will then pass it along to the simulator for verification, and,

hopefully, insights which can only be determined via a dynamic simulation. Alternatively, the synthesizer may need to interact incrementally with the simulator in order to compute the potential effects of solving a subtask in some particular way.

6. Conclusions

The Mechanism Lab is an attempt to gain insights into how humans represent and apply cause-effect knowledge. Our theory attempts to define important concepts related to causality, and then to support them by demonstrating their utility in a system which puts them to good use. In this case, "good use" has meant the "mental simulation" of mechanisms in terms of these concepts. Whether or not such a simulation bears any relationship to psychological reality can never (we feel) be conclusively determined. However, since it is the intent of the CSA representation to allow humans to describe cause-effect knowledge in the most natural terms possible, and since, empirically, the traces of the simulator output correspond to verbal explanations we ourselves offer when asked to describe how a mechanism operates, we feel we are at least paralleling some interesting mechanisms of human intelligence.

For the immediate future of the Mechanisms Lab, our goals are threefold: (1) to extend the simulator so that it can access the existing CSA database of cause-effect schemata, (2) investigate the syntactic transformations which preserve meaning in the CSA representation, and (3) to apply and extend the existing CSA plan synthesizer to mechanism invention.

7. References

[BB1] Brown, J., and Burton, R., Multiple Representations of Knowledge for Tutorial Reasoning, in Representation and Understanding, D. Bobrow and A. Collins (eds.), Academic Press, 1975

[D1] Davis, R., Applications of Meta Level Knowledge to the Construction, Maintenance and Use of Large Knowledge Bases, Doctoral Dissertation, Stanford A.I. Lab. AIM 283, 1976

[F1] Freiling, M., Understanding Mechanical Systems, M.I.T. unpublished, 1975

[JW1] Johnston, R., and White, M., Simulation of an Artificial Heart System, 1971 Summer Computer Simulation Conference, 1971

[L1] London, P., Abstraction Mapping as a Self-Organizing Scheme for Problem Solving Systems, Doctorial Dissertation Proposal, University of Maryland, 1976

[LBR1] Luetscher, J., Boyer, D., and Resneck, J., Control of the Renal Circulation and Kidney Function, 1972 Summer Computer Simulation Conference, 1972

[MS1] McDermott, D. and Sussman, G., The CONNIVER Reference Manual, M.I.T. AI Memo 259a, 1974

[R1] Rieger, C., An Organization of Knowledge for Problem Solving and Language Comprehension, Artificial Intelligence, vol. 7, no. 2, 1976

[R2] Rieger, C., The Representation and Selection of Commonsense Knowledge for Natural Language Comprehension, Proc. Georgetown University Linguistics Roundtable, 1976

[R3] Rieger, C., Spontaneous Computation in Cognitive Models, University of Maryland TR 459, 1976

[R4] Rieger, C., The Commonsense Algorithm as a Basis..., Proceedings TINLAP Workshop, M.I.T., 1975

[RJS1] Raines, J., Jaffrin, M., and Shapiro, A., A Computer Simulation of the Human Arterial System, 1971 Summer Computer Simulation Conference, 1971

[SL1] Sahinkaya, Y.,and Lee, Y., A digital Computer Simulation Model for a SCR DC to DC Voltage Converter, 1972 Summer Computer Simulation Conference, 1972

[SS1] Sussman, G., and Stallman, R. Heuristic Techniques in Computer Aided Circuit Analysis, M.I.T. AI Memo 328, 1975

[SS2] Sussman, G., and Stallman, R. Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis, M.I.T. AI Memo 380, 1976

[SWC1] Sussman, G., Winograd, T., and Charniak, E., MICRO-PLANNER Reference Manual, M.I.T. AI Memo 203a, 1971

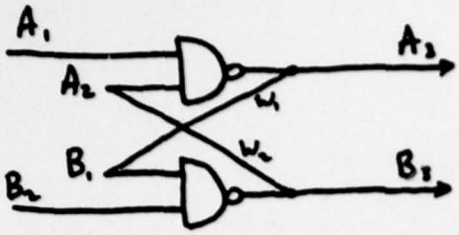
[WBL1] Wright, E., Blokland, G., and Lawrence, B., Simulation of a Natural Circulation Boiler Using a Hybrid Computer, 1972 Summer Computer Simulation Conference, 1972

Appendix A: FLIP-FLOP

There are two CSA versions of a computer FLIP FLOP. The first FLIP FLOP's CSA description doesnot take into account the detailed timing sequences of the NAND gate components (i.e. propogation delays). It is only involved with the I/O specifications of the FLIP FLOP. The second CSA description of the FLIP FLOP takes into account the timing problems by viewing the NAND gates as causing state-changes to the output lines. This allows switching delays to be incorporated into the model.

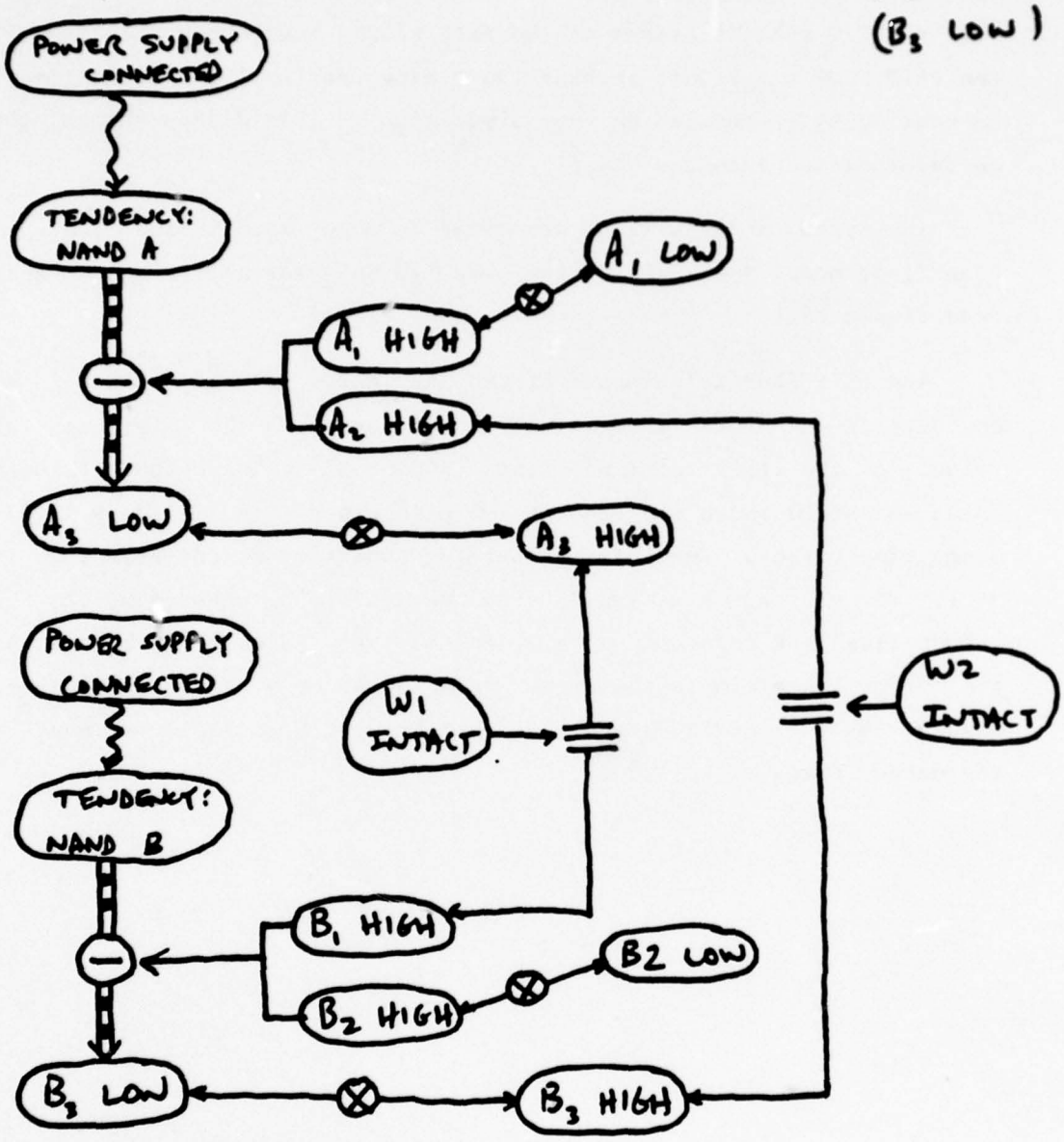
The second model will be described in more detail and then simulated. (The first model incorporates the same I/O behavior as the second and has also been simulated.)

The FLIP FLOP is composed of two NAND gates with the output lines of each one used as one of the input lines of the other. The user can alter the values of the other two input lines (A1 and B2). The output of the FLIP FLOP is at A3 and B3 which are opposite of each other (i.e. if A3 is 1, B3 must be 0 and vice verse). The initial starting condition of the FLIP FLOP is with A1 = 1, B2 =1, A3 = 1 and B3 = 0. When B2 is brought to 0, the FLIP FLOP's output lines are reversed (A3 = 0, and B3 = 1). When B2 is brought back to 1, the output lines remain the same. Bringing A1 to 0 causes the output lines to reverse (A3 = 1 and B3 =0). A1 is then brought high again without affecting the output lines.



FLIP FLOP

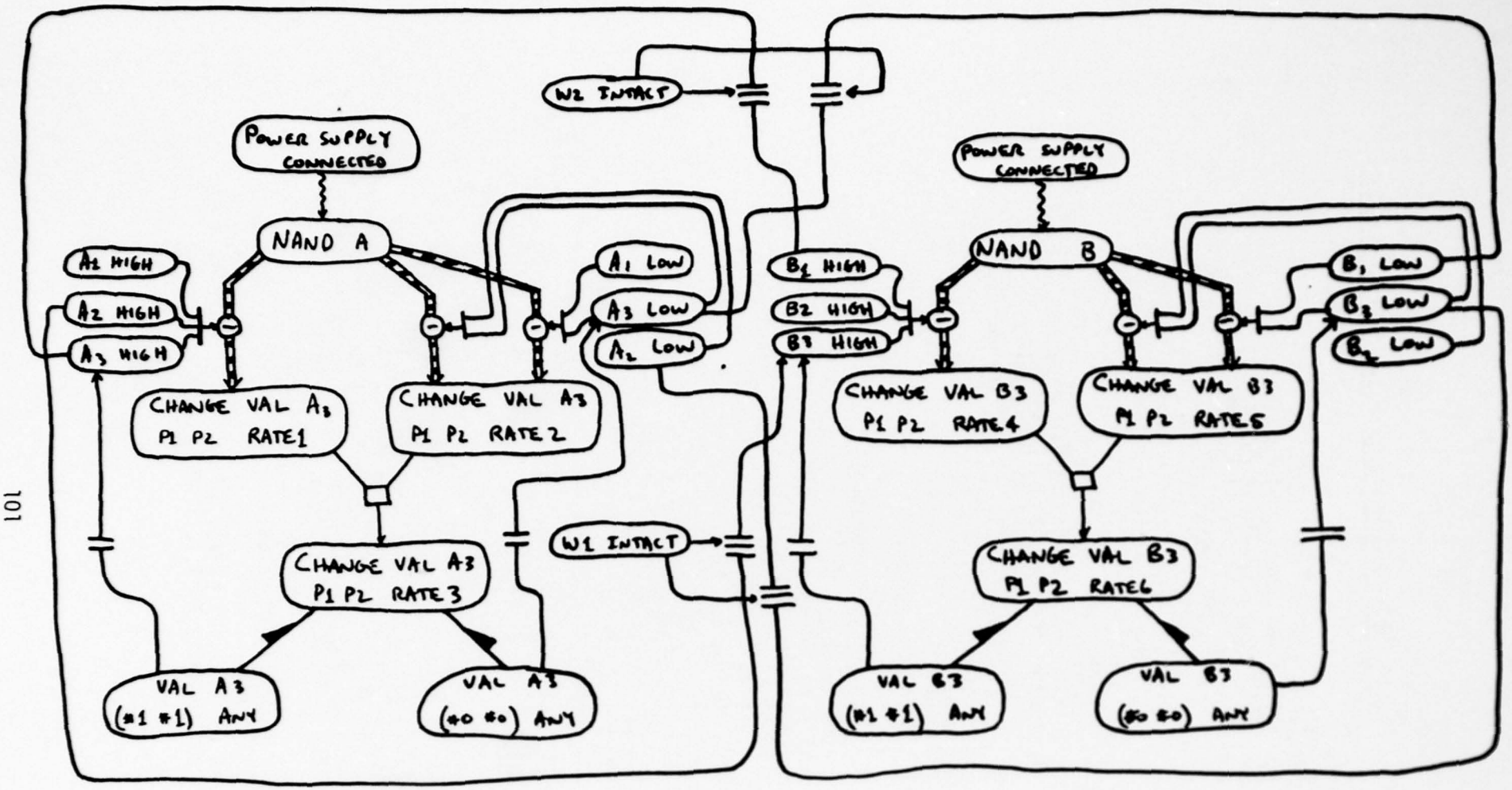
INITIAL WORLD: (A_3 HIGH)
(A_1 HIGH)
(B_2 HIGH)
(B_3 LOW)



TRIGGERS: (B2 LOW)

OR

(A1 LOW)

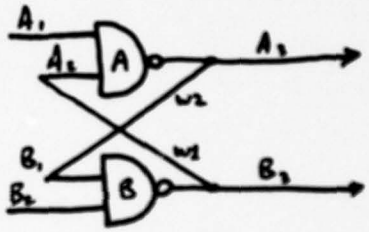


VAL A3 #1 POS I

VAL B3 #0 NEG I

COMPUTER FLIP FLOP

(WITH DETAILED SWITCHING DELAY MODEL)



FLIP-FLOP External Declarative Representation

```
($MECHANISM '(
(NAME FLIP-FLOP)
(EVENTS (1 T (NAND A))
        (2 SC (STATE-CHANGE VAL A3 P1 P2 RATE1))
        (3 SC (STATE-CHANGE VAL A3 P1 P2 RATE2))
        (4 SC (STATE-CHANGE VAL A3 P1 P2 RATE3))
        (5 S (VAL A3 (#1 #2) ANY))
        (6 S (VAL A3 (#0 #1) ANY))
        (7 S (A1 HIGH))
        (8 S (A2 HIGH))
        (9 S (A3 HIGH))
        (10 S (A1 LOW))
        (11 S (A3 LOW))
        (12 S (A2 LOW))
        (13 T (NAND B))
        (14 SC (STATE-CHANGE VAL B3 P1 P2 RATE4))
        (15 SC (STATE-CHANGE VAL B3 P1 P2 RATE5))
        (16 SC (STATE-CHANGE VAL B3 P1 P2 RATE6))
        (17 S (VAL B3 (#1 #2) ANY))
        (18 S (VAL B3 (#0 #1) ANY))
        (19 S (B1 HIGH))
        (20 S (B2 HIGH))
        (21 S (B3 HIGH))
        (22 S (B1 LOW))
        (23 S (B3 LOW))
        (24 S (B2 LOW))
        (25 S (POWER SUPPLY CONNECTED))
        (26 S (W1 INTACT))
        (27 S (W2 INTACT))
        (28 S (VAL A3 #1 POS))
        (29 S (VAL B3 #0 NEG))      )
(LINKS  (C-CAUSE (1 2) (7 8 9))
        (C-CAUSE (1 3) (10 11))
        (C-CAUSE (1 3) (11 12))
        (C-CAUSE (13 14) (19 20 21))
        (C-CAUSE (13 15) (22 23))
        (C-CAUSE (13 15) (23 24))
        (RATE-CONFL ((2 3) 4))
        (RATE-CONFL ((14 15) 16))
        (THRESH (4 (5 6)))
        (THRESH (16 (17 18)))
        (S-COUPLE (5 9))
        (S-COUPLE (6 11))
        (S-COUPLE (17 21))
        (S-COUPLE (18 23))
        (S-EQUIV (19 9) (27))
        (S-EQUIV (21 8) (26))
        (S-EQUIV (22 11) (27))
        (S-EQUIV (12 23) (26))
        (ANTAG (7 10))
        (ANTAG (8 12))
        (ANTAG (9 11))
        (ANTAG (19 22))
        (ANTAG (20 24))
        (ANTAG (21 23))
        (C-ENABLE (25 13))
        (C-ENABLE (25 1))      )
(RATES   (RATE1 #-1)
         (RATE2 #1)
         (RATE3 (#PLUS RATE1 RATE2))
         (RATE4 #-1)
         (RATE5 #1)
         (RATE6 (#PLUS RATE4 RATE5))    )
(INITIAL-WORLD 28 29 26 27 25 9 7 23 20 12 19)
(TRIGGER 24)
))
```

FLIP-FLOP Simulation

```
EVAL: ($SIMULATE-MECH 'FLIP-FLOP)
 MAX TICK COUNT (T = INFINITY) T

 ** INITIAL WORLD EVENTS **
(VAL A3 #1 POS) STORED G144 CAUSED BY IW
(VAL B3 #0 NEG) STORED G145 CAUSED BY IW
(W1 INTACT) STORED G146 CAUSED BY IW
(W2 INTACT) STORED G153 CAUSED BY IW
(POWER SUPPLY CONNECTED) STORED G154 CAUSED BY IW
(A3 HIGH) STORED G160 CAUSED BY IW
(A1 HIGH) STORED G142 CAUSED BY IW
(B3 LOW) STORED G148 CAUSED BY IW
(B2 HIGH) STORED G152 CAUSED BY IW
(A2 LOW) STORED G143 CAUSED BY IW
(B1 HIGH) STORED G157 CAUSED BY IW
 ** END INITIAL WORLD **
   G110 HIDDEN CAUSED BY G110
   G111 HIDDEN CAUSED BY G110
   G97 HIDDEN CAUSED BY G94
   G89 HIDDEN CAUSED BY G86
   G122 HIDDEN CAUSED BY G122
   G123 HIDDEN CAUSED BY G122
   G83 HIDDEN CAUSED BY G84
   G91 HIDDEN CAUSED BY G92
   G81 HIDDEN CAUSED BY G78
   G101 HIDDEN CAUSED BY G98
   G140 HIDDEN CAUSED BY G140
(NAND B) STORED G158 CAUSED BY G140
   G138 HIDDEN CAUSED BY G138
(NAND A) STORED G159 CAUSED BY G138
   G102 HIDDEN CAUSED BY G102
   G108 HIDDEN CAUSED BY G108

(B2 LOW) STORED G155 CAUSED BY TRIGGER

TICK 0 REAL TIME 6869
   G126 HIDDEN CAUSED BY G126
(STATE-CHANGE VAL B3 P1 P2 G5) STORED G147 CAUSED BY G126
(STATE-CHANGE VAL B3 P1 P2 G6) STORED G150 CAUSED BY G66
   G79 HIDDEN CAUSED BY G80
   G81 UNHIDDEN CAUSED BY G80
(B2 HIGH) ERASED G152 CAUSED BY G80

TICK 1 REAL TIME 8699
   G74 HIDDEN CAUSED BY G74
   G75 HIDDEN CAUSED BY G74
(VAL B3 #1 POS) CHANGED G145 CAUSED BY G74
   G104 HIDDEN CAUSED BY G104
(B3 HIGH) STORED G152 CAUSED BY G104
   G118 HIDDEN CAUSED BY G118
   G119 HIDDEN CAUSED BY G118
(A2 HIGH) STORED G156 CAUSED BY G118
   G136 HIDDEN CAUSED BY G136
(STATE-CHANGE VAL A3 P1 P2 G1) STORED G149 CAUSED BY G136
(STATE-CHANGE VAL A3 P1 P2 G3) STORED G151 CAUSED BY G72
   G93 HIDDEN CAUSED BY G90
   G91 UNHIDDEN CAUSED BY G90
(A2 LOW) ERASED G143 CAUSED BY G90
   G122 UNHIDDEN CAUSED BY G124
   G123 UNHIDDEN CAUSED BY G124
(B3 LOW) ERASED G148 CAUSED BY G124
(STATE-CHANGE VAL B3 P1 P2 G5) ERASED G147 CAUSED BY G129
(STATE-CHANGE VAL B3 P1 P2 G6) ERASED G150 CAUSED BY G67
   G126 UNHIDDEN CAUSED BY G127
   G82 HIDDEN CAUSED BY G85
   G83 UNHIDDEN CAUSED BY G85
   G85 HIDDEN CAUSED BY G82
   G82 UNHIDDEN CAUSED BY G82
```

```
   G102 UNHIDDEN CAUSED BY G103

TICK 2 REAL TIME 15722
(STATE-CHANGE VAL B3 -B -E G6) NO LONGER CHANGING
   G74 UNHIDDEN CAUSED BY $S-INCR
   G75 UNHIDDEN CAUSED BY $S-INCR
   G76 HIDDEN CAUSED BY G76
   G77 HIDDEN CAUSED BY G76
(VAL A3 #0 NEG) CHANGED G144 CAUSED BY G76
   G108 UNHIDDEN CAUSED BY G109
   G106 HIDDEN CAUSED BY G106
(A3 LOW) STORED G150 CAUSED BY G106
   G95 HIDDEN CAUSED BY G96
   G97 UNHIDDEN CAUSED BY G96
(A3 HIGH) ERASED G160 CAUSED BY G96
   G136 UNHIDDEN CAUSED BY G137
(STATE-CHANGE VAL A3 P1 P2 G1) ERASED G149 CAUSED BY G137
(STATE-CHANGE VAL A3 P1 P2 G3) ERASED G151 CAUSED BY G73
   G110 UNHIDDEN CAUSED BY G113
   G111 UNHIDDEN CAUSED BY G113
(B1 HIGH) ERASED G157 CAUSED BY G113
   G88 HIDDEN CAUSED BY G87
   G89 UNHIDDEN CAUSED BY G87
(B1 LOW) STORED G157 CAUSED BY G87
   G114 HIDDEN CAUSED BY G114
   G115 HIDDEN CAUSED BY G114

TICK 3 REAL TIME 20757
(STATE-CHANGE VAL A3 -B -E G3) NO LONGER CHANGING
   G76 UNHIDDEN CAUSED BY $S-INCR
   G77 UNHIDDEN CAUSED BY $S-INCR

TICK 4 REAL TIME 20851

LIST ADDITIONAL ACTIVITIES:  (($store-simu '(b2 high)))

(B2 HIGH) STORED G151 CAUSED BY $TIMER
   G81 HIDDEN CAUSED BY G78
   G79 UNHIDDEN CAUSED BY G78
(B2 LOW) ERASED G155 CAUSED BY G78

TICK 5 REAL TIME 21485

LIST ADDITIONAL ACTIVITIES:  (($store-simu '(a1 low)))

(A1 LOW) STORED G155 CAUSED BY $TIMER
   G134 HIDDEN CAUSED BY G134
(STATE-CHANGE VAL A3 P1 P2 G2) STORED G149 CAUSED BY G134
(STATE-CHANGE VAL A3 P1 P2 G3) STORED G160 CAUSED BY G72
   G99 HIDDEN CAUSED BY G100
   G101 UNHIDDEN CAUSED BY G100
(A1 HIGH) ERASED G142 CAUSED BY G100

TICK 6 REAL TIME 23605
   G76 HIDDEN CAUSED BY G76
   G77 HIDDEN CAUSED BY G76
(VAL A3 #1 POS) CHANGED G144 CAUSED BY G76
   G108 HIDDEN CAUSED BY G108
(A3 HIGH) STORED G142 CAUSED BY G108
   G97 HIDDEN CAUSED BY G94
   G95 UNHIDDEN CAUSED BY G94
(A3 LOW) ERASED G150 CAUSED BY G94
   G134 UNHIDDEN CAUSED BY G135
(STATE-CHANGE VAL A3 P1 P2 G2) ERASED G149 CAUSED BY G135
(STATE-CHANGE VAL A3 P1 P2 G3) ERASED G160 CAUSED BY G73
   G114 UNHIDDEN CAUSED BY G117
   G115 UNHIDDEN CAUSED BY G117
(B1 LOW) ERASED G157 CAUSED BY G117
   G86 HIDDEN CAUSED BY G89
   G88 UNHIDDEN CAUSED BY G89
(B1 HIGH) STORED G157 CAUSED BY G89
```

```
   G130 HIDDEN CAUSED BY G130
(STATE-CHANGE VAL B3 P1 P2 G4) STORED G160 CAUSED BY G130
(STATE-CHANGE VAL B3 P1 P2 G6) STORED G149 CAUSED BY G66
   G110 HIDDEN CAUSED BY G110
   G111 HIDDEN CAUSED BY G110
   G106 UNHIDDEN CAUSED BY G107

TICK 7 REAL TIME 29978
(STATE-CHANGE VAL A3 -B -E G3) NO LONGER CHANGING
   G76 UNHIDDEN CAUSED BY $S-INCR
   G77 UNHIDDEN CAUSED BY $S-INCR
   G74 HIDDEN CAUSED BY G74
   G75 HIDDEN CAUSED BY G74
(VAL B3 #0 NEG) CHANGED G145 CAUSED BY G74
   G104 UNHIDDEN CAUSED BY G105
   G102 HIDDEN CAUSED BY G102
(B3 LOW) STORED G150 CAUSED BY G102
   G83 HIDDEN CAUSED BY G84
   G85 UNHIDDEN CAUSED BY G84
(B3 HIGH) ERASED G152 CAUSED BY G84
   G130 UNHIDDEN CAUSED BY G131
(STATE-CHANGE VAL B3 P1 P2 G4) ERASED G160 CAUSED BY G131
(STATE-CHANGE VAL B3 P1 P2 G6) ERASED G149 CAUSED BY G67
   G118 UNHIDDEN CAUSED BY G120
   G119 UNHIDDEN CAUSED BY G120
(A2 HIGH) ERASED G156 CAUSED BY G120
   G92 HIDDEN CAUSED BY G91
   G93 UNHIDDEN CAUSED BY G91
(A2 LOW) STORED G156 CAUSED BY G91
   G122 HIDDEN CAUSED BY G122
   G123 HIDDEN CAUSED BY G122

TICK 8 REAL TIME 34327
(STATE-CHANGE VAL B3 -B -E G6) NO LONGER CHANGING
   G74 UNHIDDEN CAUSED BY $S-INCR
   G75 UNHIDDEN CAUSED BY $S-INCR

TICK 9 REAL TIME 34426

LIST ADDITIONAL ACTIVITIES:  (($store-simu '(a1 high)))

(A1 HIGH) STORED G149 CAUSED BY $TIMER
   G101 HIDDEN CAUSED BY G98
   G99 UNHIDDEN CAUSED BY G98
(A1 LOW) ERASED G155 CAUSED BY G98

TICK 10 REAL TIME 35070

LIST ADDITIONAL ACTIVITIES:  nil

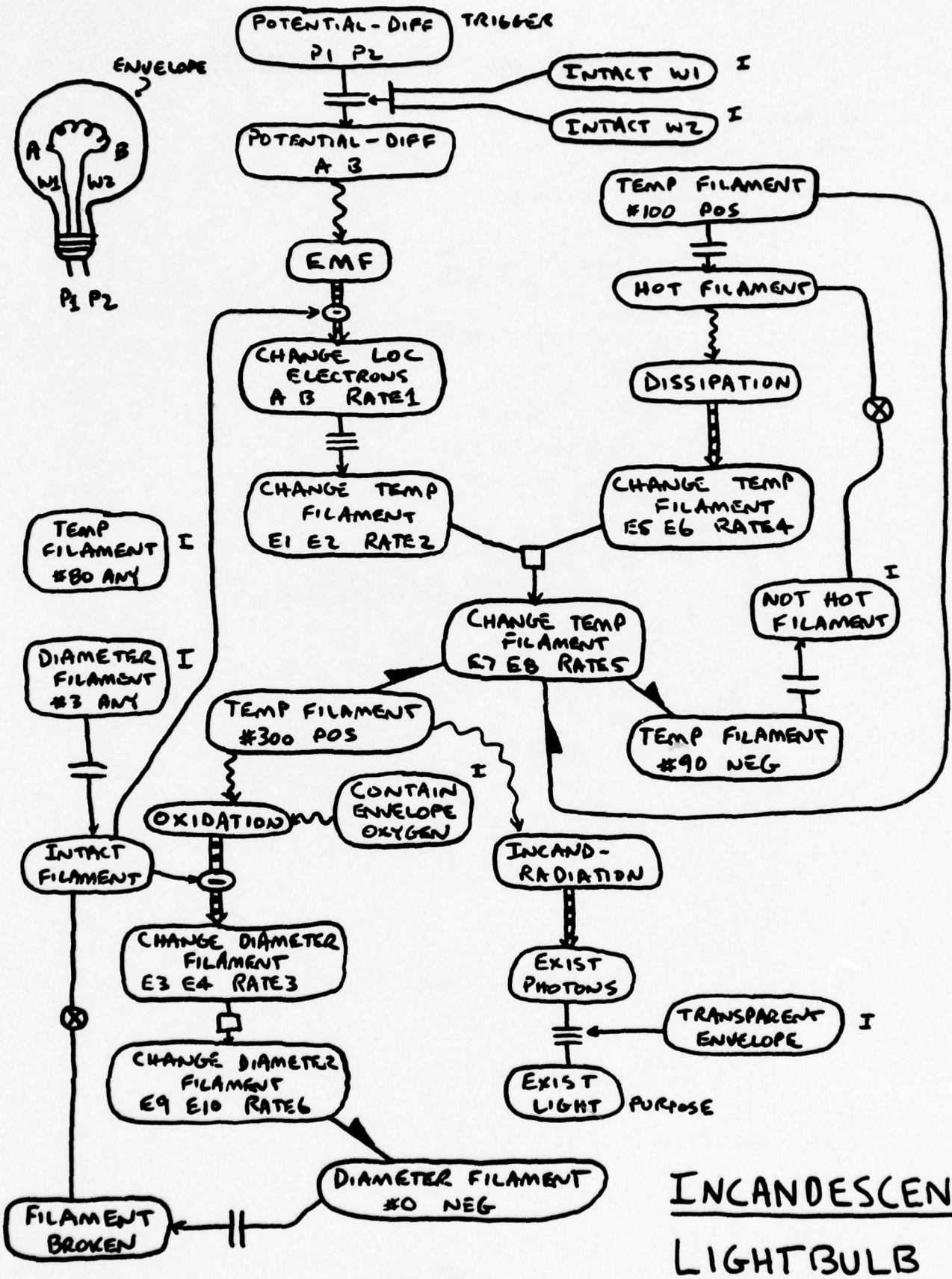
** FINAL WORLD **
(A1 HIGH)
(A2 LOW)
(B3 LOW)
(VAL B3 #0 NEG)
(B1 HIGH)
(A3 HIGH)
(VAL A3 #1 POS)
(B2 HIGH)
(NAND A)
(NAND B)
(POWER SUPPLY CONNECTED)
(W2 INTACT)
(W1 INTACT)
FLIP-FLOP SIMULATED 35217
```

Appendix B: LIGHTBULB

The LIGHTBULB is a standard incandescent lightbulb. It is composed of a transparent envelope, a filament, wires connecting the filament to two external points on the metal base of the bulb. These points would then be connected to the external AC power supply (not described).

The state of the LIGHTBULB for it to be in normal working condition is: (1) the two wires are intact between the external points and the filament and (2) the filament is intact. Also, there is some residual oxygen in the envelope.

When the AC power is applied to the external points, the filament increases in temperature. When it reaches 300 degrees, the bulb starts to glow. Since the filament is hot and the envelope contains oxygen, the filament starts to oxidize and thus to decrease in diameter. Eventually the diameter reaches zero which means that the filament is no longer intact. The bulb will then cease to glow. The heat of the filament when it was intact is dissipated into the environment.



INCANDESCENT LIGHTBULB

LIGHTBULB External Declarative Representation

```
($MECHANISM '(
(NAME LIGHTBULB)
(EVENTS (1 S (POTENTIAL-DIFF P1 P2) )
        (2 S (INTACT W1))
        (3 S (INTACT W2))
        (4 S (POTENTIAL-DIFF A B))
        (5 T (EMF))
        (6 S (INTACT FILAMENT))
        (7 SC (CHANGE LOC ELC A B RATE1))
        (8 SC (CHANGE TEMP FILAMENT E1 E2 RATE2))
        (9 S (TEMP FILAMENT #300 POS))
        (10 S (CONTAIN ENVELOPE OXYGEN))
        (11 T (OXIDATION))
        (12 SC (CHANGE DIAMETER FILAMENT E3 E4 RATE3))
        (13 S (DIAMETER FILAMENT #0 NEG))
        (14 S (BROKEN FILAMENT))
        (15 T (INCAND-RADIATION))
        (16 S (EXIST PHOTONS))
        (17 S (TRANSPARENT ENVELOPE))
        (18 S (EXIST LIGHT))
        (19 S (TEMP FILAMENT #100 POS))
        (20 T (DISSIPATION))
        (21 SC (CHANGE FILAMENT TEMP E5 E6 RATE4))
        (22 S (DIAMETER FILAMENT #3 ANY))
        (23 S (HOT FILAMENT))
        (24 S (TEMP FILAMENT #80 ANY))
        (25 S (TEMP FILAMENT #90 NEG))
        (26 S (NOT HOT FILAMENT))
        (27 SC (CHANGE TEMP FILAMENT E7 E8 RATE5))
        (28 SC (CHANGE DIAMETER FILAMENT E9 E10 RATE6))    )
(LINKS (C-ENABLE (23 20))
       (C-CAUSE (20 21))
       (S-COUPLE (19 23))
       (S-COUPLE (1 4) (2 3))
       (C-ENABLE (4 5))
       (C-CAUSE (5 7) (6))
       (S-EQUIV (7 8))
       (C-ENABLE (9 11))
       (C-CAUSE (11 12) (6))
       (C-ENABLE (10 11))
       (RATE-CONFL ((12) 28) )
       (THRESH (28 (13)))
       (S-COUPLE (13 14))
       (ANTAG (14 6))
       (C-ENABLE (9 15))
       (C-CAUSE (15 16))
       (S-EQUIV (16 18) (17))
       (S-COUPLE (22 6))
       (S-COUPLE (19 23))
       (ANTAG (26 23))
       (S-COUPLE (25 26))
       (RATE-CONFL ((8 21) 27))
       (THRESH (27 (9 19 25)))       )
(RATES (RATE1 #1000)
       (RATE2 #200)
       (RATE3 #-1)
       (RATE4 #-195)
       (RATE5 (#PLUS RATE2 RATE4))
       (RATE6 (#PLUS RATE3))    )
(TRIGGER 1)
(PURPOSE 18)
(INITIAL-WORLD 2 3 10 17 22 24 26 )
))
```

LIGHTBULB Simulation

```
EVAL: ($SIMULATE-MECH 'LIGHTBULB)
 MAX TICK COUNT (T = INFINITY) T

 ** INITIAL WORLD EVENTS **
(INTACT W1) STORED G115 CAUSED BY IW
(INTACT W2) STORED G116 CAUSED BY IW
(CONTAIN ENVELOPE OXYGEN) STORED G117 CAUSED BY IW
(TRANSPARENT ENVELOPE) STORED G118 CAUSED BY IW
(DIAMETER FILAMENT #3 ANY) STORED G119 CAUSED BY IW
(TEMP FILAMENT #80 ANY) STORED G120 CAUSED BY IW
(NOT HOT FILAMENT) STORED G121 CAUSED BY IW
 ** END INITIAL WORLD **
   G74 HIDDEN CAUSED BY G71
   G85 HIDDEN CAUSED BY G85
(INTACT FILAMENT) STORED G122 CAUSED BY G85
   G76 HIDDEN CAUSED BY G77

(POTENTIAL-DIFF P1 P2) STORED G123 CAUSED BY TRIGGER

TICK 0 REAL TIME 11911
   G89 HIDDEN CAUSED BY G89
(POTENTIAL-DIFF A B) STORED G124 CAUSED BY G89
   G107 HIDDEN CAUSED BY G107
(EMF) STORED G125 CAUSED BY G107
   G105 HIDDEN CAUSED BY G105
(CHANGE LOC ELC A B G1) STORED G126 CAUSED BY G105
   G95 HIDDEN CAUSED BY G95
   G96 HIDDEN CAUSED BY G95
(CHANGE TEMP FILAMENT E1 E2 G2) STORED G127 CAUSED BY G95
(CHANGE TEMP FILAMENT E7 E8 G5) STORED G128 CAUSED BY G65

TICK 1 REAL TIME 14862
   G69 HIDDEN CAUSED BY G69
   G70 HIDDEN CAUSED BY G69
(TEMP FILAMENT #280 POS) CHANGED G120 CAUSED BY G69
   G83 HIDDEN CAUSED BY G83
(HOT FILAMENT) STORED G129 CAUSED BY G83
   G113 HIDDEN CAUSED BY G113
(DISSIPATION) STORED G130 CAUSED BY G113
   G99 HIDDEN CAUSED BY G99
(CHANGE FILAMENT TEMP E5 E6 G4) STORED G131 CAUSED BY G99
   G72 HIDDEN CAUSED BY G73
   G74 UNHIDDEN CAUSED BY G73
(NOT HOT FILAMENT) ERASED G121 CAUSED BY G73
   G81 HIDDEN CAUSED BY G81

TICK 2 REAL TIME 17729
(TEMP FILAMENT #285 POS) CHANGED G120 CAUSED BY $S-INCR

TICK 3 REAL TIME 18756
(TEMP FILAMENT #290 POS) CHANGED G120 CAUSED BY $S-INCR

TICK 4 REAL TIME 20727
(TEMP FILAMENT #295 POS) CHANGED G120 CAUSED BY $S-INCR


TICK 5 REAL TIME 21740
(TEMP FILAMENT #300 POS) CHANGED G120 CAUSED BY $S-INCR
   G111 HIDDEN CAUSED BY G111
(INCAND-RADIATION) STORED G121 CAUSED BY G111
   G101 HIDDEN CAUSED BY G101
(EXIST PHOTONS) STORED G132 CAUSED BY G101
   G91 HIDDEN CAUSED BY G91
   G92 HIDDEN CAUSED BY G91
(EXIST LIGHT) STORED G133 CAUSED BY G91
   G109 HIDDEN CAUSED BY G109
(OXIDATION) STORED G134 CAUSED BY G109
   G103 HIDDEN CAUSED BY G103
(CHANGE DIAMETER FILAMENT E3 E4 G3) STORED G135 CAUSED BY G103
```

```
(CHANGE DIAMETER FILAMENT E9 E10 G6) STORED G136 CAUSED BY G59

TICK 6 REAL TIME 24522
(TEMP FILAMENT #305 POS) CHANGED G120 CAUSED BY $S-INCR
   G67 HIDDEN CAUSED BY G67
   G68 HIDDEN CAUSED BY G67
(DIAMETER FILAMENT #2 NEG) CHANGED G119 CAUSED BY G67
   G85 UNHIDDEN CAUSED BY G86

TICK 7 REAL TIME 26407
(TEMP FILAMENT #310 POS) CHANGED G120 CAUSED BY $S-INCR
(DIAMETER FILAMENT #1 NEG) CHANGED G119 CAUSED BY $S-INCR

TICK 8 REAL TIME 29204
(TEMP FILAMENT #315 POS) CHANGED G120 CAUSED BY $S-INCR
(DIAMETER FILAMENT #0 NEG) CHANGED G119 CAUSED BY $S-INCR
   G87 HIDDEN CAUSED BY G87
(BROKEN FILAMENT) STORED G137 CAUSED BY G87
   G78 HIDDEN CAUSED BY G75
   G76 UNHIDDEN CAUSED BY G75
(INTACT FILAMENT) ERASED G122 CAUSED BY G75
   G105 UNHIDDEN CAUSED BY G106
(CHANGE LOC ELC A B G1) ERASED G126 CAUSED BY G106
   G95 UNHIDDEN CAUSED BY G97
   G96 UNHIDDEN CAUSED BY G97
(CHANGE TEMP FILAMENT E1 E2 G2) ERASED G127 CAUSED BY G97
   G103 UNHIDDEN CAUSED BY G104
(CHANGE DIAMETER FILAMENT E3 E4 G3) ERASED G135 CAUSED BY G104
(CHANGE DIAMETER FILAMENT E9 E10 G6) ERASED G136 CAUSED BY G60

TICK 9 REAL TIME 34209
(TEMP FILAMENT #120 NEG) CHANGED G120 CAUSED BY $S-INCR
   G111 UNHIDDEN CAUSED BY G112
(INCAND-RADIATION) ERASED G121 CAUSED BY G112
   G101 UNHIDDEN CAUSED BY G102
(EXIST PHOTONS) ERASED G132 CAUSED BY G102
   G91 UNHIDDEN CAUSED BY G93
   G92 UNHIDDEN CAUSED BY G93
(EXIST LIGHT) ERASED G133 CAUSED BY G93
   G109 UNHIDDEN CAUSED BY G110
(OXIDATION) ERASED G134 CAUSED BY G110
(CHANGE DIAMETER FILAMENT -B -E G6) NO LONGER CHANGING
   G67 UNHIDDEN CAUSED BY $S-INCR
   G68 UNHIDDEN CAUSED BY $S-INCR

TICK 10 REAL TIME 37146
(TEMP FILAMENT #-75 NEG) CHANGED G120 CAUSED BY $S-INCR
   G79 HIDDEN CAUSED BY G79
(NOT HOT FILAMENT) STORED G134 CAUSED BY G79
   G74 HIDDEN CAUSED BY G71
   G72 UNHIDDEN CAUSED BY G71
(HOT FILAMENT) ERASED G129 CAUSED BY G71
   G113 UNHIDDEN CAUSED BY G114
(DISSIPATION) ERASED G130 CAUSED BY G114
   G99 UNHIDDEN CAUSED BY G100
(CHANGE FILAMENT TEMP E5 E6 G4) ERASED G131 CAUSED BY G100
(CHANGE TEMP FILAMENT E7 E8 G5) ERASED G128 CAUSED BY G66
   G83 UNHIDDEN CAUSED BY G84
   G81 UNHIDDEN CAUSED BY G82

TICK 11 REAL TIME 39857
(CHANGE TEMP FILAMENT -B -E G5) NO LONGER CHANGING
   G69 UNHIDDEN CAUSED BY $S-INCR
   G70 UNHIDDEN CAUSED BY $S-INCR

TICK 12 REAL TIME 39970

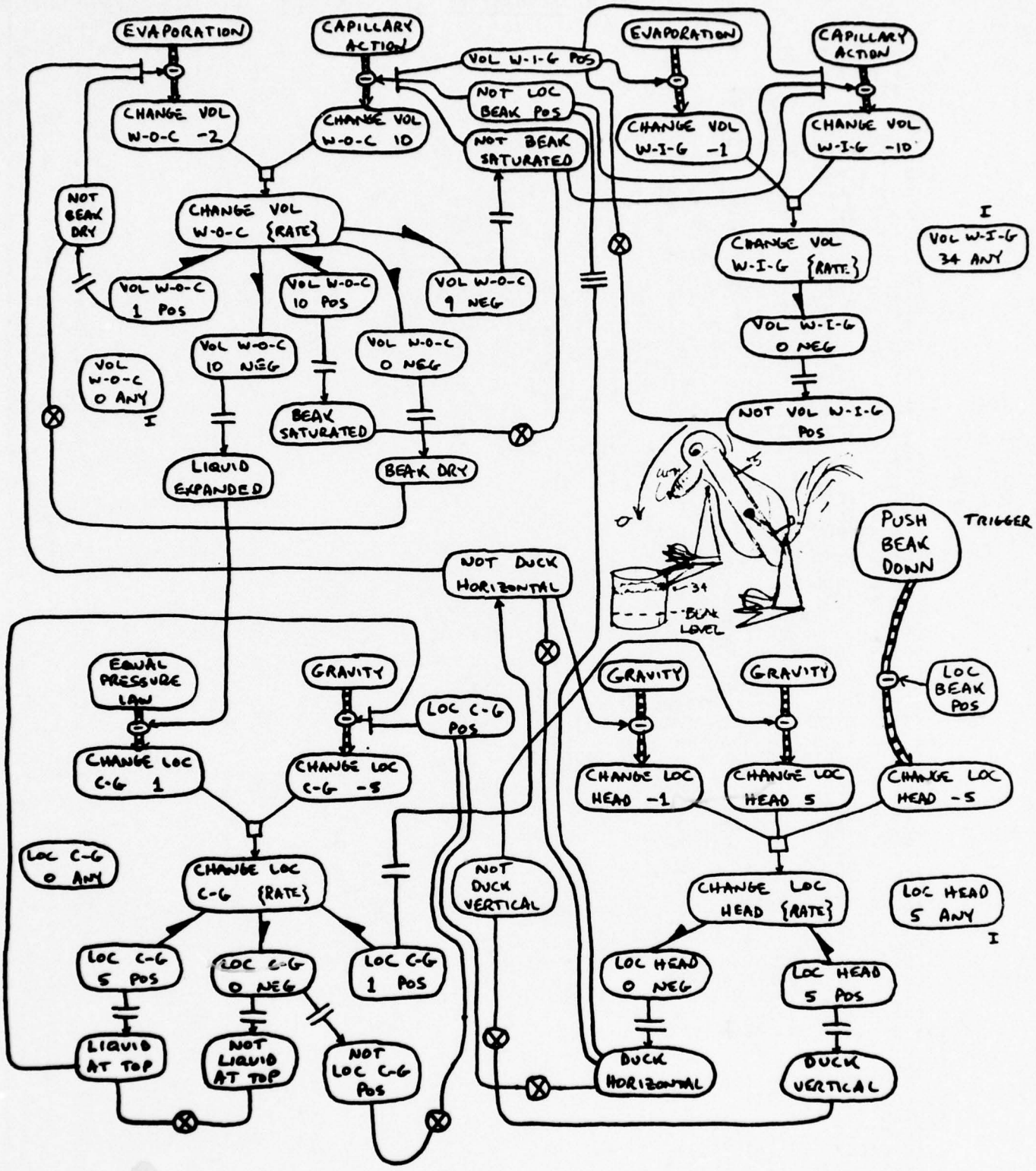
LIST ADDITIONAL ACTIVITIES: NIL

** FINAL WORLD **
(NOT HOT FILAMENT)
```

```
(TEMP FILAMENT #-75 NEG)
(BROKEN FILAMENT)
(DIAMETER FILAMENT #0 NEG)
(EMF)
(POTENTIAL-DIFF A B)
(POTENTIAL-DIFF P1 P2)
(TRANSPARENT ENVELOPE)
(CONTAIN ENVELOPE OXYGEN)
(INTACT W2)
(INTACT W1)
LIGHTBULB SIMULATED 40084
```

Appendix C: MARVELOUS DRINKING DUCK

Push the beak of the MARVELOUS DRINKING DUCK into the water glass. The beak becomes wet and the duck rights itself. As the water on the beak evaporates, the liquid in the duck's neck expands due to the temperature change. The rising center of gravity in the duck's neck makes it bend over until its beak is again wetted. At the same time that the duck is horizontal, the liquid in the neck spills out the top of its inner tube, running back down to the resevoir at the bottom. This causes the duck to right itself again. This process continues until all the water in the glass has evaporated.



MARVELOUS DRINKING DUCK

DUCK External Declarative Representation

```
($MECHANISM '(
(NAME DUCK)
(EVENTS
  (1 A (PUSH BEAK DOWN))
  (2 S (LOC1 BEAK POSITIVE))
  (3 SC (CHANGE11 LOC HEAD E1 E2 #-5))
  (4 S (LOC HEAD #5 ANY))
  (5 T (GRAVITY1))
  (6 SC (CHANGE12 LOC HEAD E3 E4 #5))
  (7 S (NOT DUCK VERTICAL))
  (8 SC (CHANGE LOC HEAD E5 E6 #0))
  (9 S (LOC HEAD #0 NEG))
  (10 S (DUCK HORIZONTAL))
  (11 S (LOC HEAD #5 POS))
  (12 S (DUCK VERTICAL))
  (13 T (GRAVITY2))
  (14 SC (CHANGE13 LOC HEAD E7 E8 #-1))
  (15 S (NOT DUCK HORIZONTAL))
  (16 S (LIQUID EXPANDED))
  (17 T (EQUAL-PRESSURE LAW))
  (18 SC (CHANGE21 LOC CG E9 E10 #1))
  (19 S (LOC CG #0 ANY))
  (20 T (GRAVITY3))
  (21 SC (CHANGE22 LOC CG E11 E12 #-5))
  (22 S (LOC2 CG POSITIVE))
  (23 SC (CHANGE LOC CG E13 E14 #0))
  (24 S (LOC CG #5 POS))
  (25 S (LIQUID-AT-TOP))
  (26 S (NOT LIQUID-AT-TOP))
  (27 S (LOC CG #0 NEG))
  (28 S (NOT LOC2 CG POSITIVE))
  (29 S (LOC CG #1 POS))
  (30 S (VOL W-O-C #0 ANY))
  (31 T (EVAPORATION1))
  (32 SC (CHANGE31 VOL W-O-C E15 E16 #-2))
  (33 S (NOT BEAK DRY))
  (34 T (CAPILLARY ACTION))
  (35 SC (CHANGE32 VOL W-O-C E17 E18 #10))
  (36 S (VOL1 W-I-G POSITIVE))
  (37 S (NOT LOC1 BEAK POSITIVE))
  (38 S (NOT BEAK SATURATED))
  (39 SC (CHANGE VOL W-O-C E19 E20 #0))
  (40 S (VOL W-O-C #1 POS))
  (41 S (VOL W-O-C #10 NEG))
  (42 S (VOL W-O-C #10 POS))
  (43 S (BEAK SATURATED))
  (44 S (VOL W-O-C #0 NEG))
  (45 S (BEAK DRY))
  (46 S (VOL W-O-C #9 NEG))
  (47 T (EVAPORATION2))
  (48 SC (CHANGE41 VOL W-I-G E21 E22 #-1))
  (49 T (CAPILLARY-ACTION2))
  (50 SC (CHANGE42 VOL W-I-G E23 E24 #-10))
  (51 SC (CHANGE VOL W-I-G E25 E26 #0))
  (52 S (VOL W-I-G #0 NEG))
  (53 S (NOT VOL1 W-I-G POSITIVE))
  (54 S (VOL W-I-G #34 ANY))
(55 S (NOT PUSH BEAK DOWN))
)
(LINKS
  (C-CAUSE (1 3) (2))
(S-COUPLE (10 55))
(ANTAG (55 1))
  (C-CAUSE (5 6) (7))
  (C-CAUSE (13 14) (15))
  (RATE-CONFL ((14 6 3) 8))
  (THRESH (8 (9 11)))
  (S-COUPLE (9 10))
```

```
   (S-COUPLE (11 12))
   (ANTAG (7 12))
   (ANTAG (10 15))
   (S-COUPLE (10 22))
   (S-EQUIV (10 37))
   (C-CAUSE (17 18) (16))
   (C-CAUSE (20 21) (22 25))
   (RATE-CONFL ((18 21) 23))
   (THRESH (23 (24 27 29)))
   (S-COUPLE (24 25))
   (S-COUPLE (27 26))
   (S-COUPLE (27 28))
   (S-COUPLE (27 7))
   (S-COUPLE (29 15))
   (ANTAG (25 26))
   (ANTAG (22 28))
   (C-CAUSE (31 32) (33))
   (C-CAUSE (34 35) (36 37 38))
   (RATE-CONFL ((32 35) 39))
   (THRESH (39 (40 41 42 44 46)))
   (S-COUPLE (40 33))
   (S-COUPLE (41 16) (33))
   (S-COUPLE (42 43))
   (S-COUPLE (44 45))
   (S-COUPLE (46 38))
   (ANTAG (43 38))
   (ANTAG (45 33))
   (C-CAUSE (47 48) (36))
   (C-CAUSE (49 50) (36 37 38))
   (RATE-CONFL ((48 50) 51))
   (THRESH (51 (52)))
   (S-COUPLE (52 53))
   (ANTAG (53 36))
)
(TRIGGER 1 36)
(PURPOSE 53)
(INITIAL-WORLD 2 4 19 30  38 45 54 5 13 17 20 31 34 47 49)
))
```

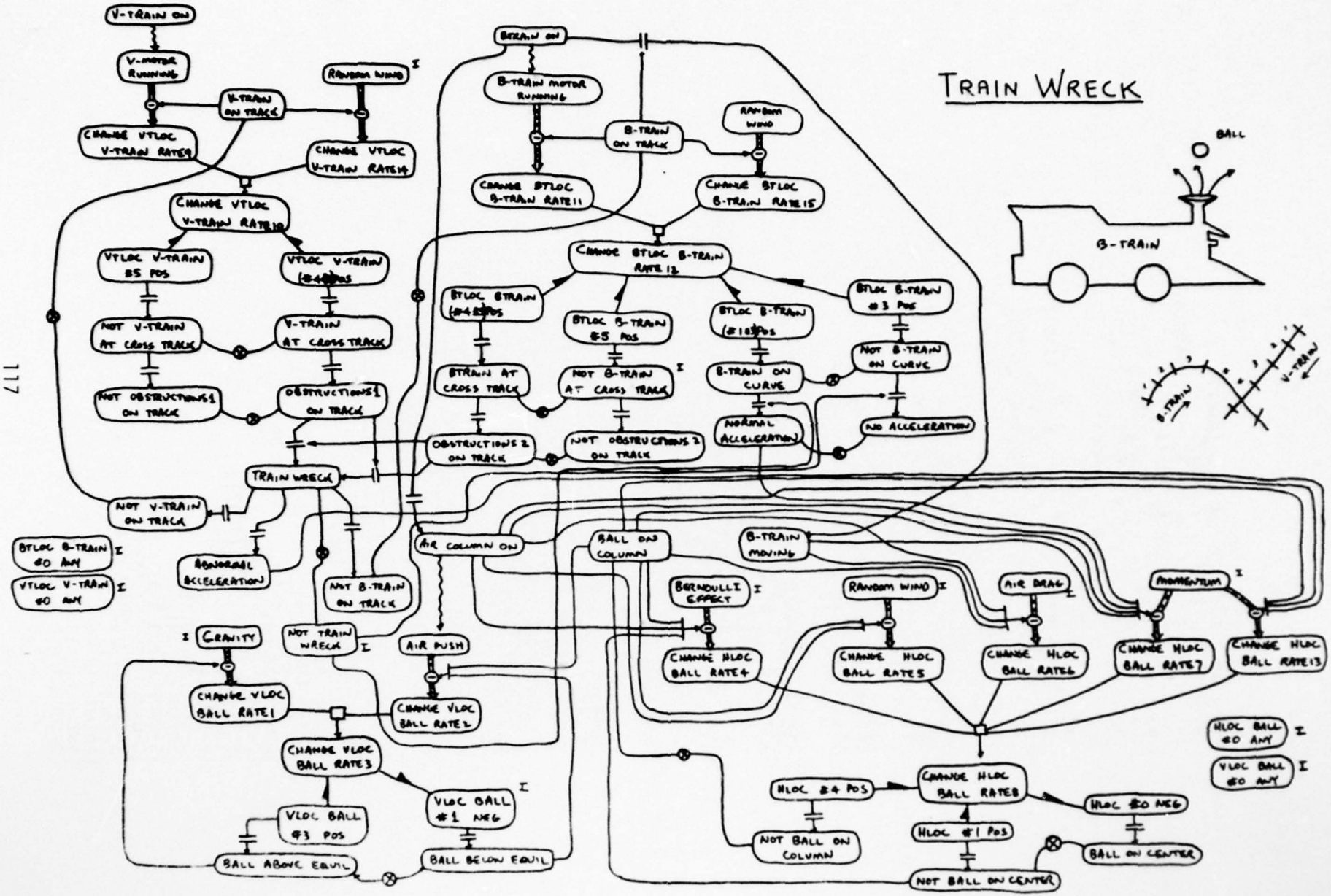
Appendix D: BERNOULLI'S TRAIN-WRECK

The scenario for the TRAIN-WRECK is as follows: There are two trains, The Ball Train (B-TRAIN) and the Villian Train (V-TRAIN), each proceeding down its own track to a common cross track. While the B-TRAIN is on, there is an air column blowing out of its smoke stack with a ball balanced on the air column. The CSA description incorporates models of the physical laws which control the ball's position (horizontal and vertical) with respect to a given world state. When the two trains meet at the cross track, each train views the other as an obstruction and they both perceive a train wreck. At this time both trains jump the track and are subject to an abrupt change in velocity (abnormal acceleration). This change in velocity causes the ball to be knocked off the air column.

TRAIN-WRECK External Declarative Representation

```
($MECHANISM '(
(NAME TRAIN-WRECK)
(EVENTS
        (1 S (B-TRAIN ON))
        (2 S (AIR COLUMN ON))
        (3 S (B-TRAIN ON TRACK))
        (4 S (B-TRAIN MOVING))
        (5 A (AIR PUSH))
        (6 T (GRAVITY))
        (7 SC (CHANGE1 VLOC BALL B E RATE1))
        (8 SC (CHANGE2 VLOC BALL B E RATE2))
        (9 SC (CHANGE3 VLOC BALL B E RATE3))
        (12 S (VLOC BALL #0 NEG))
        (13 S (VLOC BALL #1 POS))
        (15 S (BALL BELOW EQUILIBRIUM))
        (16 S (BALL ABOVE EQUILIBRIUM))
        (17 S (BALL ON COLUMN))
        (18 T (BERNOULLI EFFECT))
        (19 SC (CHANGE4 HLOC BALL B E RATE4))
        (20 SC (CHANGE5 HLOC BALL B E RATE5))
        (21 T (RANDOM WIND))
        (22 T (AIR DRAG))
        (23 SC (CHANGE6 HLOC BALL B E RATE6))
        (24 S (NORMAL B-TRAIN ACCELERATION))
        (25 T (MOMENTUM))
        (26 SC (CHANGE7 HLOC BALL B E RATE7))
        (27 SC (CHANGE8 HLOC BALL B E RATE8))
        (28 S (HLOC BALL #4 POS))
        (29 S (NOT BALL ON COLUMN))
        (30 S (HLOC BALL #1 POS))
        (31 S (NOT BALL ON CENTER))
        (32 S (HLOC BALL #0 NEG))
        (33 S (BALL ON CENTER))
        (35 A (V-TRAIN MOTOR RUNNING))
        (36 SC (CHANGE9 VTLOC V-TRAIN B E RATE9))
        (37 S (V-TRAIN ON))
```

TRAIN WRECK


V-TRAIN ON
V-MOTOR RUNNING
RANDOM WIND I
V-TRAIN ON TRACK
CHANGE VTLOC V-TRAIN RATE9
CHANGE VTLOC V-TRAIN RATE14
CHANGE VTLOC V-TRAIN RATE10
VTLOC V-TRAIN #5 POS
V-TRAIN AT CROSS TRACK
NOT V-TRAIN AT CROSS TRACK
OBSTRUCTIONS1 ON TRACK
NOT OBSTRUCTIONS1 ON TRACK
TRAIN WRECK
NOT V-TRAIN ON TRACK
BTLOC B-TRAIN #0 ANY I
VTLOC V-TRAIN #0 ANY I
ABNORMAL ACCLERATION
NOT B-TRAIN ON TRACK
I GRAVITY
NOT TRAIN WRECK I
AIR PUSH
CHANGE VLOC BALL RATE1
CHANGE VLOC BALL RATE2
CHANGE VLOC BALL RATE3
VLOC BALL #3 POS
VLOC BALL #1 NEG I
BALL ABOVE EQUIL
BALL BELOW EQUIL
BTRAIN ON
B-TRAIN MOTOR RUNNING
B-TRAIN ON TRACK
RANDOM WIND
CHANGE BTLOC B-TRAIN RATE11
CHANGE BTLOC B-TRAIN RATE15
CHANGE BTLOC B-TRAIN RATE12
BTLOC BTRAIN
BTLOC B-TRAIN #5 POS
BTLOC B-TRAIN
BTLOC B-TRAIN #3 POS
BTRAIN AT CROSS TRACK
NOT B-TRAIN AT CROSS TRACK I
B-TRAIN ON CURVE
NOT B-TRAIN ON CURVE
OBSTRUCTIONS2 ON TRACK
NOT OBSTRUCTIONS2 ON TRACK
NORMAL ACCLERATION
NO ACCLERATION
AIR COLUMN ON
BALL ON COLUMN
B-TRAIN MOVING
BERNOULLI EFFECT I
RANDOM WIND I
AIR DRAG I
MOMENTUM I
CHANGE HLOC BALL RATE4
CHANGE HLOC BALL RATE5
CHANGE HLOC BALL RATE6
CHANGE HLOC BALL RATE7
CHANGE HLOC BALL RATE13
HLOC BALL #0 ANY I
VLOC BALL #0 ANY I
HLOC #4 POS
CHANGE HLOC BALL RATE8
HLOC #0 NEG
HLOC #1 POS
NOT BALL ON COLUMN
BALL ON CENTER
NOT BALL ON CENTER
BALL
B-TRAIN
B-TRAIN
V-TRAIN

```
(38 S (V-TRAIN ON TRACK))
(39 S (NOT OBSTRUCTIONS2 ON TRACK))
(40 S (VTLOC V-TRAIN #0 ANY))
(41 SC (CHANGE10 VTLOC V-TRAIN B E RATE10))
(42 S (VTLOC V-TRAIN #5 POS))
(43 S (VTLOC V-TRAIN (#4 #5) POS))
(44 S (NOT V-TRAIN AT CROSS TRACK))
(45 S (V-TRAIN AT CROSS TRACK))
(46 S (OBSTRUCTIONS1 ON TRACK))
(47 S (TRAIN WRECK))
(48 A (B-MOTOR RUNNING))
(49 SC (CHANGE14 VTLOC V-TRAIN B E RATE14))
(50 S (NOT OBSRUCTIONS1 ON TRACK))
(51 SC (CHANGE11 BTLOC B-TRAIN B E RATE11))
(52 S (BTLOC B-TRAIN #0 ANY))
(53 SC (CHANGE12 BTLOC B-TRAIN B E RATE12))
(54 S (BTLOC B-TRAIN (#4 #5) POS))
(55 S (BTLOC B-TRAIN #5 POS))
(56 S (B-TRAIN AT CROSS TRACK))
(57 S (NOT B-TRAIN AT CROSS TRACK))
(58 S (OBSTRUCTIONS2 ON TRACK))
(59 S (NOT B-TRAIN ON TRACK))
(60 S (NOT V-TRAIN ON TRACK))
(61 SC (CHANGE13 HLOC BALL B E RATE13))
(62 S (ABNORMAL B-TRAIN ACCELERATION))

(63 SC (CHANGE15 BTLOC B-TRAIN B E RATE15))
(64 S (BTLOC B-TRAIN (#1 #3) POS))
(65 S (BTLOC B-TRAIN #3 POS))
(66 S (B-TRAIN ON CURVE))
(67 S (NOT B-TRAIN ON CURVE))
(68 S (NOT B-TRAIN ACCELERATING))
(69 S (NOT TRAIN WRECK))
(70 S (START SIMULATION))     )
(LINKS
(C-ENABLE (1 48))
(S-COUPLE (1 2))
(S-COUPLE (1 4) (3))
(C-ENABLE (2 5))
(ANTAG (3 59))
(C-CAUSE (5 8) (15 17))
(C-CAUSE (6 7) (16))
(RATE-CONFL ((7 8) 9))
(THRESH (9 (13 12)))
(S-COUPLE (12 15))
(S-COUPLE (13 16))
(ANTAG (15 16))
(ANTAG (17 29))
(C-CAUSE (18 19) (2 17 31))
(RATE-CONFL ((19 20 23 26 61) 27))
(C-CAUSE (21 20) (2 17))
(C-CAUSE (21 49) (38))
(C-CAUSE (21 63) (3))
(C-CAUSE (22 23) (2 4 17))
(S-COUPLE (66 24) (69))
(ANTAG (66 67))
(C-ENABLE (70 1))
(C-ENABLE (70 37))
(ANTAG (24 68))
(C-CAUSE (25 26) (2 4 17 24))
(C-CAUSE (25 61) (2 17 62))
(THRESH (27 (28 30 32)))
(S-COUPLE (28 29))
(S-COUPLE (30 31))
(ANTAG (31 33))
(S-COUPLE (32 33))
(C-CAUSE (35 36) (38))
(RATE-CONFL ((36 49) 41))
(C-ENABLE (37 35))
(ANTAG (38 60))
(ANTAG (39 58))
(S-COUPLE (57 39))
```

```
            (THRESH (41 (42 43)))
            (S-COUPLE (42 44))
            (S-COUPLE (43 45))
            (S-COUPLE (44 50))
            (ANTAG (44 45))
            (S-COUPLE (45 46))
            (S-COUPLE (46 47) (58))
            (ANTAG (46 50))
            (S-COUPLE (58 47) (46))
            (ANTAG (47 69))
            (S-COUPLE (47 60))
            (S-COUPLE (47 62))
            (S-COUPLE (47 59))
            (C-CAUSE (48 51) (3))
            (RATE-CONFL ((51 63) 53))
            (THRESH (53 (54 55 64 65)))
            (S-COUPLE (54 56))
            (S-COUPLE (55 57))
            (S-COUPLE (56 58))
            (ANTAG (56 57))
            (S-COUPLE (64 66))
            (S-COUPLE (65 67))
            (S-COUPLE (67 68) (69))     )
(RATES
            (RATE1 #-1)
            (RATE2 #1)
            (RATE3 (#PLUS RATE1 RATE2))
            (RATE4 #-4)
            (RATE5 #1)
            (RATE6 #1)
            (RATE7 #1)
            (RATE8 (#PLUS RATE4 RATE5 RATE6 RATE7 RATE13))
            (RATE9 #1)
            (RATE10 (#PLUS RATE9 RATE14))
            (RATE11 #1)
            (RATE12 (#PLUS RATE11 RATE15))
            (RATE13 #10)
            (RATE14 #0)
            (RATE15 #0)     )
(TRIGGER 70)
(INITIAL-WORLD 3 6 12 17 18 21 22 25 32 38 40 44 52 57 69)
(PURPOSE 5)
))
```

Simulation of TRAIN-WRECK

```
($SIMULATE-MECH 'TRAIN-WRECK)
 MAX TICK COUNT (T = INFINITY)   7

       Summary of Initial World

       B-TRAIN and V-TRAIN are on track.
       Gravity, Bernoulli Effect, Random Wind, Air Drag
            and Momentum are asserted.
       Vertical and horizontal position of the ball is 0.
       Both trains are at track position 0.

** INITIAL WORLD EVENTS **
(B-TRAIN ON TRACK) STORED G308 CAUSED BY IW
(GRAVITY) STORED G309 CAUSED BY IW
(VLOC BALL #0 NEG) STORED G310 CAUSED BY IW
(BALL ON COLUMN) STORED G311 CAUSED BY IW
(BERNOULLI EFFECT) STORED G312 CAUSED BY IW
(RANDOM WIND) STORED G313 CAUSED BY IW
(AIR DRAG) STORED G314 CAUSED BY IW
(MOMENTUM) STORED G315 CAUSED BY IW
(HLOC BALL #0 NEG) STORED G316 CAUSED BY IW
(V-TRAIN ON TRACK) STORED G317 CAUSED BY IW
(VTLOC V-TRAIN #0 ANY) STORED G318 CAUSED BY IW
(NOT V-TRAIN AT CROSS TRACK) STORED G319 CAUSED BY IW
(BTLOC B-TRAIN #0 ANY) STORED G320 CAUSED BY IW
(NOT B-TRAIN AT CROSS TRACK) STORED G321 CAUSED BY IW
(NOT TRAIN WRECK) STORED G322 CAUSED BY IW
 ** END INITIAL WORLD **


   G181 HIDDEN CAUSED BY G182
   G248 HIDDEN CAUSED BY G248
(NOT OBSTRUCTIONS2 ON TRACK) STORED G323 CAUSED BY G248
   G203 HIDDEN CAUSED BY G200
   G205 HIDDEN CAUSED BY G206
   G254 HIDDEN CAUSED BY G254
(NOT OBSRUCTIONS1 ON TRACK) STORED G324 CAUSED BY G254
   G209 HIDDEN CAUSED BY G210
   G215 HIDDEN CAUSED BY G212
   G282 HIDDEN CAUSED BY G282
(CHANGE14 VTLOC V-TRAIN B E G14) STORED G325 CAUSED BY G282
(CHANGE10 VTLOC V-TRAIN B E G10) STORED G326 CAUSED BY G164
   G195 HIDDEN CAUSED BY G192
   G262 HIDDEN CAUSED BY G262
(BALL ON CENTER) STORED G327 CAUSED BY G262
   G217 HIDDEN CAUSED BY G218
   G276 HIDDEN CAUSED BY G276
(CHANGE15 BTLOC B-TRAIN B E G15) STORED G328 CAUSED BY G276
(CHANGE12 BTLOC B-TRAIN B E G12) STORED G329 CAUSED BY G146
   G223 HIDDEN CAUSED BY G220
   G270 HIDDEN CAUSED BY G270
(BALL BELOW EQUILIBRIUM) STORED G330 CAUSED BY G270
   G227 HIDDEN CAUSED BY G224
   G272 HIDDEN CAUSED BY G272
(B-TRAIN MOVING) STORED G331 CAUSED BY G272
   G199 HIDDEN CAUSED BY G196

(START SIMULATION) STORED G332 CAUSED BY TRIGGER

       Zero Tick of Simulation
       Summary of Events

       Both train motors start running.
       Air Column gets turned on.
       Air starts pushing the ball.
       Both trains advance to track position 1.
       B-TRAIN is on a curved track and is experiencing
            normal acceleration.
```

```
TICK 0 REAL TIME 29661
   G306 HIDDEN CAUSED BY G306
   G302 HIDDEN CAUSED BY G302
(B-MOTOR RUNNING) STORED G333 CAUSED BY G302
   G280 HIDDEN CAUSED BY G280
(CHANGE11 BTLOC B-TRAIN B E G11) STORED G334 CAUSED BY G280
   G274 HIDDEN CAUSED BY G274
(AIR COLUMN ON) STORED G335 CAUSED BY G274
   G298 HIDDEN CAUSED BY G298
(AIR PUSH) STORED G336 CAUSED BY G298
   G294 HIDDEN CAUSED BY G294
(CHANGE2 VLOC BALL B E G2) STORED G337 CAUSED BY G294
(CHANGE3 VLOC BALL B E G3) STORED G338 CAUSED BY G170
   G292 HIDDEN CAUSED BY G292
(CHANGE5 HLOC BALL B E G5) STORED G339 CAUSED BY G292
(CHANGE8 HLOC BALL B E G8) STORED G340 CAUSED BY G158
   G290 HIDDEN CAUSED BY G290
(CHANGE6 HLOC BALL B E G6) STORED G341 CAUSED BY G290
   G304 HIDDEN CAUSED BY G304
   G300 HIDDEN CAUSED BY G300
(V-TRAIN MOTOR RUNNING) STORED G342 CAUSED BY G300
   G284 HIDDEN CAUSED BY G284
(CHANGE9 VTLOC V-TRAIN B E G9) STORED G343 CAUSED BY G284
   G174 HIDDEN CAUSED BY G174
   G175 HIDDEN CAUSED BY G174
(VTLOC V-TRAIN #1 POS) CHANGED G318 CAUSED BY G174
   G172 HIDDEN CAUSED BY G172
   G173 HIDDEN CAUSED BY G172
(BTLOC B-TRAIN #1 POS) CHANGED G320 CAUSED BY G172
   G234 HIDDEN CAUSED BY G234
(B-TRAIN ON CURVE) STORED G344 CAUSED BY G234
   G228 HIDDEN CAUSED BY G228
(NORMAL B-TRAIN ACCELERATION) STORED G345 CAUSED BY G228
   G288 HIDDEN CAUSED BY G288
(CHANGE7 HLOC BALL B E G7) STORED G346 CAUSED BY G288
   G187 HIDDEN CAUSED BY G184
   G191 HIDDEN CAUSED BY G188

        First Tick of Simulation
        Summary of Events

        Air pushes the vertical location of the ball to a
             point above equilibrium.
        Random Wind, Air drag and Momentum move the
             horizontal location of the ball to a point
             off the center of the column.
        Both trains advance to position 2 of the track.

TICK 1 REAL TIME 75424
   G178 HIDDEN CAUSED BY G178
   G179 HIDDEN CAUSED BY G178
(VLOC BALL #1 POS) CHANGED G310 CAUSED BY G178
   G268 HIDDEN CAUSED BY G268
(BALL ABOVE EQUILIBRIUM) STORED G347 CAUSED BY G268
   G296 HIDDEN CAUSED BY G296
(CHANGE1 VLOC BALL B E G1) STORED G348 CAUSED BY G296
   G225 HIDDEN CAUSED BY G226
   G227 UNHIDDEN CAUSED BY G226
(BALL BELOW EQUILIBRIUM) ERASED G330 CAUSED BY G226
   G294 UNHIDDEN CAUSED BY G295
(CHANGE2 VLOC BALL B E G2) ERASED G337 CAUSED BY G295
   G270 UNHIDDEN CAUSED BY G271
   G176 HIDDEN CAUSED BY G176
   G177 HIDDEN CAUSED BY G176
(HLOC BALL #3 POS) CHANGED G316 CAUSED BY G176
   G264 HIDDEN CAUSED BY G264
(NOT BALL ON CENTER) STORED G337 CAUSED BY G264
   G286 HIDDEN CAUSED BY G286
(CHANGE4 HLOC BALL B E G4) STORED G330 CAUSED BY G286
   G219 HIDDEN CAUSED BY G216
   G217 UNHIDDEN CAUSED BY G216
```

```
(BALL ON CENTER) ERASED G327 CAUSED BY G216
   G262 UNHIDDEN CAUSED BY G263
(VTLOC V-TRAIN #2 POS) CHANGED G318 CAUSED BY $S-INCR
(BTLOC B-TRAIN #2 POS) CHANGED G320 CAUSED BY $S-INCR

       Second Tick of Simulation
       Summary of Events

       Gravity has forced the vertical location of the
            ball to a point below equilibrium.
       Since the ball was off the center of the column,
            the Bernoulli Effect has sucked it back on
            center.
       Both trains are at track location 3.
       B-TRAIN is now on straight track and normal
            acceleration has ceased.

TICK 2 REAL TIME 121038
(VLOC BALL #0 NEG) CHANGED G310 CAUSED BY $S-INCR
   G270 HIDDEN CAUSED BY G270
(BALL BELOW EQUILIBRIUM) STORED G327 CAUSED BY G270
   G294 HIDDEN CAUSED BY G294
(CHANGE2 VLOC BALL B E G2) STORED G349 CAUSED BY G294
   G227 HIDDEN CAUSED BY G224
   G225 UNHIDDEN CAUSED BY G224
(BALL ABOVE EQUILIBRIUM) ERASED G347 CAUSED BY G224
   G296 UNHIDDEN CAUSED BY G297
(CHANGE1 VLOC BALL B E G1) ERASED G348 CAUSED BY G297
   G268 UNHIDDEN CAUSED BY G269
(HLOC BALL #2 NEG) CHANGED G316 CAUSED BY $S-INCR
(VTLOC V-TRAIN #3 POS) CHANGED G318 CAUSED BY $S-INCR
(BTLOC B-TRAIN #3 POS) CHANGED G320 CAUSED BY $S-INCR
   G232 HIDDEN CAUSED BY G232
(NOT B-TRAIN ON CURVE) STORED G348 CAUSED BY G232
   G230 HIDDEN CAUSED BY G230
(NOT B-TRAIN ACCELERATING) STORED G347 CAUSED BY G230
   G185 HIDDEN CAUSED BY G186
   G187 UNHIDDEN CAUSED BY G186
(NORMAL B-TRAIN ACCELERATION) ERASED G345 CAUSED BY G186
   G288 UNHIDDEN CAUSED BY G289
(CHANGE7 HLOC BALL B E G7) ERASED G346 CAUSED BY G289
   G189 HIDDEN CAUSED BY G190
   G191 UNHIDDEN CAUSED BY G190
(B-TRAIN ON CURVE) ERASED G344 CAUSED BY G190
   G228 UNHIDDEN CAUSED BY G229
   G234 UNHIDDEN CAUSED BY G235

       Third Tick of Simulation
       Summary of Events

       Air has pushed the vertical location of the ball
            to a point above equilibrium.
       Random Wind, Air Drag and Momentum have moved the
            ball to a horizontal location off the center
            of the column.
       Both trains are at track position 4 which is the
            (cross track).  Each perceives the other as
            an obstruction on the track and each train
            has a train wreck.  The train wreck causes
            both trains to jump the track and to introduce
            abnormal acceleration into the system.

TICK 3 REAL TIME 171975
(VLOC BALL #1 POS) CHANGED G310 CAUSED BY $S-INCR
   G268 HIDDEN CAUSED BY G268
(BALL ABOVE EQUILIBRIUM) STORED G344 CAUSED BY G268
   G296 HIDDEN CAUSED BY G296
(CHANGE1 VLOC BALL B E G1) STORED G346 CAUSED BY G296
   G225 HIDDEN CAUSED BY G226
   G227 UNHIDDEN CAUSED BY G226
(BALL BELOW EQUILIBRIUM) ERASED G327 CAUSED BY G226
```

```
   G294 UNHIDDEN CAUSED BY G295
(CHANGE2 VLOC BALL B E G2) ERASED G349 CAUSED BY G295
   G270 UNHIDDEN CAUSED BY G271
(HLOC BALL #0 NEG) CHANGED G316 CAUSED BY $S-INCR
   G262 HIDDEN CAUSED BY G262
(BALL ON CENTER) STORED G349 CAUSED BY G262
   G217 HIDDEN CAUSED BY G218
   G219 UNHIDDEN CAUSED BY G218
(NOT BALL ON CENTER) ERASED G337 CAUSED BY G218
   G286 UNHIDDEN CAUSED BY G287
(CHANGE4 HLOC BALL B E G4) ERASED G330 CAUSED BY G287
   G264 UNHIDDEN CAUSED BY G265
(VTLOC V-TRAIN #4 POS) CHANGED G318 CAUSED BY $S-INCR
   G258 HIDDEN CAUSED BY G258
(V-TRAIN AT CROSS TRACK) STORED G330 CAUSED BY G258
   G256 HIDDEN CAUSED BY G256
(OBSTRUCTIONS1 ON TRACK) STORED G337 CAUSED BY G256
   G211 HIDDEN CAUSED BY G208
   G209 UNHIDDEN CAUSED BY G208
(NOT OBSRUCTIONS1 ON TRACK) ERASED G324 CAUSED BY G208
   G213 HIDDEN CAUSED BY G214
   G215 UNHIDDEN CAUSED BY G214
(NOT V-TRAIN AT CROSS TRACK) ERASED G319 CAUSED BY G214
   G254 UNHIDDEN CAUSED BY G255
(BTLOC B-TRAIN #4 POS) CHANGED G320 CAUSED BY $S-INCR
   G252 HIDDEN CAUSED BY G252
(B-TRAIN AT CROSS TRACK) STORED G319 CAUSED BY G252
   G246 HIDDEN CAUSED BY G246
(OBSTRUCTIONS2 ON TRACK) STORED G324 CAUSED BY G246
   G244 HIDDEN CAUSED BY G244
(TRAIN WRECK) STORED G327 CAUSED BY G244
   G240 HIDDEN CAUSED BY G240
(NOT B-TRAIN ON TRACK) STORED G345 CAUSED BY G240
   G197 HIDDEN CAUSED BY G198
   G199 UNHIDDEN CAUSED BY G198
(B-TRAIN ON TRACK) ERASED G308 CAUSED BY G198
   G280 UNHIDDEN CAUSED BY G281
(CHANGE11 BTLOC B-TRAIN B E G11) ERASED G334 CAUSED BY G281
   G276 UNHIDDEN CAUSED BY G277
(CHANGE15 BTLOC B-TRAIN B E G15) ERASED G328 CAUSED BY G277
(CHANGE12 BTLOC B-TRAIN B E G12) ERASED G329 CAUSED BY G147
   G272 UNHIDDEN CAUSED BY G273
   G238 HIDDEN CAUSED BY G238
(NOT V-TRAIN ON TRACK) STORED G329 CAUSED BY G238
   G193 HIDDEN CAUSED BY G194
   G195 UNHIDDEN CAUSED BY G194
(V-TRAIN ON TRACK) ERASED G317 CAUSED BY G194
   G284 UNHIDDEN CAUSED BY G285
(CHANGE9 VTLOC V-TRAIN B E G9) ERASED G343 CAUSED BY G285
   G282 UNHIDDEN CAUSED BY G283
(CHANGE14 VTLOC V-TRAIN B E G14) ERASED G325 CAUSED BY G283
(CHANGE10 VTLOC V-TRAIN B E G10) ERASED G326 CAUSED BY G165
   G236 HIDDEN CAUSED BY G236
(ABNORMAL B-TRAIN ACCELERATION) STORED G326 CAUSED BY G236
   G278 HIDDEN CAUSED BY G278
(CHANGE13 HLOC BALL B E G13) STORED G325 CAUSED BY G278
   G183 HIDDEN CAUSED BY G180
   G181 UNHIDDEN CAUSED BY G180
(NOT TRAIN WRECK) ERASED G322 CAUSED BY G180
   G230 UNHIDDEN CAUSED BY G231
   G242 HIDDEN CAUSED BY G242
   G201 HIDDEN CAUSED BY G202
   G203 UNHIDDEN CAUSED BY G202
(NOT OBSTRUCTIONS2 ON TRACK) ERASED G323 CAUSED BY G202
   G207 HIDDEN CAUSED BY G204
   G205 UNHIDDEN CAUSED BY G204
(NOT B-TRAIN AT CROSS TRACK) ERASED G321 CAUSED BY G204
   G248 UNHIDDEN CAUSED BY G249

    Fourth Tick of Simulation
    Summary of Events
```

```
        Due to the abnormal aceleration the ball gets
             knocked off the column and falls to the ground

TICK 4 REAL TIME 255348
(VLOC BALL #0 NEG) CHANGED G310 CAUSED BY $S-INCR
   G270 HIDDEN CAUSED BY G270
(BALL BELOW EQUILIBRIUM) STORED G321 CAUSED BY G270
   G294 HIDDEN CAUSED BY G294
(CHANGE2 VLOC BALL B E G2) STORED G323 CAUSED BY G294
   G227 HIDDEN CAUSED BY G224
   G225 UNHIDDEN CAUSED BY G224
(BALL ABOVE EQUILIBRIUM) ERASED G344 CAUSED BY G224
   G296 UNHIDDEN CAUSED BY G297
(CHANGE1 VLOC BALL B E G1) ERASED G346 CAUSED BY G297
   G268 UNHIDDEN CAUSED BY G269
(HLOC BALL #12 POS) CHANGED G316 CAUSED BY $S-INCR
   G266 HIDDEN CAUSED BY G266
(NOT BALL ON COLUMN) STORED G346 CAUSED BY G266
   G221 HIDDEN CAUSED BY G222
   G223 UNHIDDEN CAUSED BY G222
(BALL ON COLUMN) ERASED G311 CAUSED BY G222
   G294 UNHIDDEN CAUSED BY G295
(CHANGE2 VLOC BALL B E G2) ERASED G323 CAUSED BY G295
(CHANGE3 VLOC BALL B E G3) ERASED G338 CAUSED BY G171
   G292 UNHIDDEN CAUSED BY G293
(CHANGE5 HLOC BALL B E G5) ERASED G339 CAUSED BY G293
   G290 UNHIDDEN CAUSED BY G291
(CHANGE6 HLOC BALL B E G6) ERASED G341 CAUSED BY G291
   G278 UNHIDDEN CAUSED BY G279
(CHANGE13 HLOC BALL B E G13) ERASED G325 CAUSED BY G279
(CHANGE8 HLOC BALL B E G8) ERASED G340 CAUSED BY G159
   G264 HIDDEN CAUSED BY G264
(NOT BALL ON CENTER) STORED G340 CAUSED BY G264
   G219 HIDDEN CAUSED BY G216
   G217 UNHIDDEN CAUSED BY G216
(BALL ON CENTER) ERASED G349 CAUSED BY G216
   G262 UNHIDDEN CAUSED BY G263
(CHANGE10 VTLOC V-TRAIN -B -E G10) NO LONGER CHANGING
   G174 UNHIDDEN CAUSED BY $S-INCR
   G175 UNHIDDEN CAUSED BY $S-INCR
(CHANGE12 BTLOC B-TRAIN -B -E G12) NO LONGER CHANGING
   G172 UNHIDDEN CAUSED BY $S-INCR
   G173 UNHIDDEN CAUSED BY $S-INCR

        Fifth Tick of Simulation
        Summary of Events

        Nothing changing.

TICK 5 REAL TIME 289559
(CHANGE3 VLOC BALL -B -E G3) NO LONGER CHANGING
   G178 UNHIDDEN CAUSED BY $S-INCR
   G179 UNHIDDEN CAUSED BY $S-INCR
(CHANGE8 HLOC BALL -B -E G8) NO LONGER CHANGING
   G176 UNHIDDEN CAUSED BY $S-INCR
   G177 UNHIDDEN CAUSED BY $S-INCR

TICK 6 REAL TIME 289629

LIST ADDITIONAL ACTIVITIES: nil

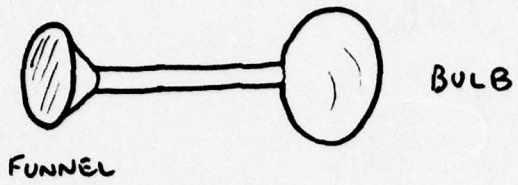
       Summary of Final World

       B-TRAIN and V-TRAIN are off the track.
       BAll is off the column, off center and below
            equilibrium.
       There is abnormal acceleration and a train wreck at
            track postion 4 (location of cross track).
       Gravity, Bernoulli Effect, Random Wind, Air Drag
            and Momentum are still asserted.
```

Both trains are still on.

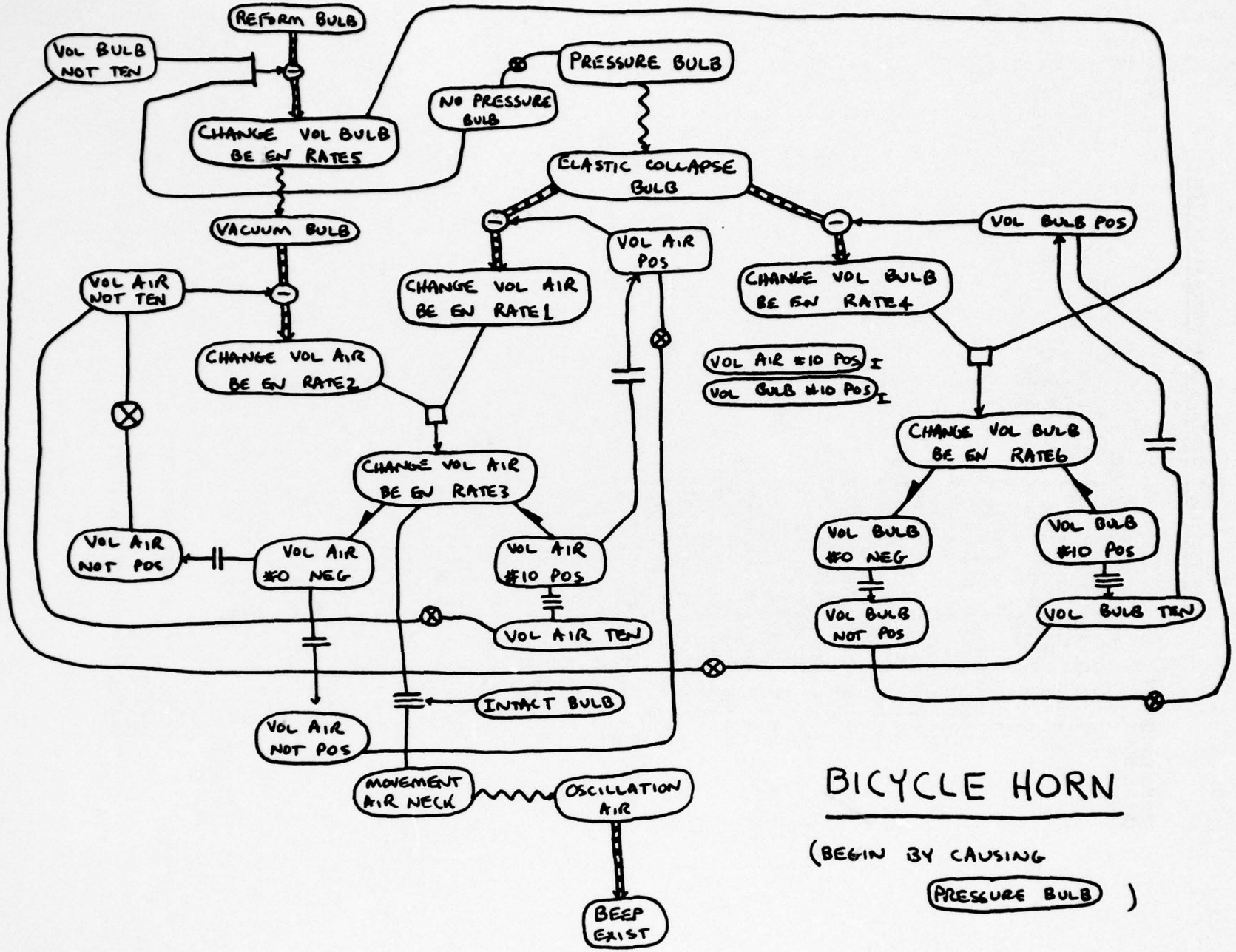
```
** FINAL WORLD **
(NOT BALL ON CENTER)
(NOT BALL ON COLUMN)
(HLOC BALL #12 POS)
(BALL BELOW EQUILIBRIUM)
(VLOC BALL #0 NEG)
(ABNORMAL B-TRAIN ACCELERATION)
(NOT V-TRAIN ON TRACK)
(NOT B-TRAIN ON TRACK)
(TRAIN WRECK)
(OBSTRUCTIONS2 ON TRACK)
(B-TRAIN AT CROSS TRACK)
(BTLOC B-TRAIN #4 POS)
(OBSTRUCTIONS1 ON TRACK)
(V-TRAIN AT CROSS TRACK)
(VTLOC V-TRAIN #4 POS)
(NOT B-TRAIN ON CURVE)
(V-TRAIN MOTOR RUNNING)
(AIR PUSH)
(AIR COLUMN ON)
(B-MOTOR RUNNING)
(START SIMULATION)
(MOMENTUM)
(AIR DRAG)
(RANDOM WIND)
(BERNOULLI EFFECT)
(GRAVITY)
TRAIN-WRECK SIMULATED 289719
```

Appendix E: BICYCLE HORN



This is a horn found on many children's bikes. It is composed of a long thin tube with a funnel shaped end. The other end of the tube has a soft rubber bulb fitting tightly over it.

The scenario of the horn in action is as follows: Air resides in the bulb. When the bulb is pressed the air is quickly pushed out of the bulb through the narrow neck and this causes the air to resonate and produce a sound. When the soft rubber bulb is released, it reforms to its original spherical shape. This will cause a vacuum to form in the bulb and air is slowly drawn in to the bulb through the tube.



BICYCLE HORN

(BEGIN BY CAUSING PRESSURE BULB)

HORN External Declarative Representation

```
($MECHANISM '(
(NAME HORN)
(EVENTS (1 A (ELASTIC COLLAPSE BULB))
        (2 SC (CHANGE VOL AIR P1 P2 RATE1))
        (3 SC (CHANGE VOL AIR P1 P2 RATE2))
        (4 SC (CHANGE VOL AIR P1 P2 RATE3))
        (5 SC (CHANGE VOL BULB P1 P2 RATE4))
        (6 SC (CHANGE VOL BULB P1 P2 RATE5))
        (7 SC (CHANGE VOL BULB P1 P2 RATE6))
        (8 S (VOL AIR #0 NEG))
        (9 S (VOL AIR #10 POS))
        (10 S (VOL BULB #0 NEG))
        (11 S (VOL BULB #10 POS))
        (12 S (NOT VOL AIR POSITIVE))
        (13 S (VOL AIR POSITIVE))
        (14 S (NOT VOL BULB POSITIVE))
        (15 S (VOL BULB POSITIVE))
        (16 S (INTACT BULB))
        (17 S (MOVEMENT AIR NECK))
        (18 T (OSCILLATION AIR))
        (19 S (BEEP EXIST))
        (20 T (REFORM BULB))
        (21 T (VACUUM BULB))
        (22 S (PRESSURE BULB))
        (23 S (NO PRESSURE BULB))
        (24 S (VOL BULB TEN))
        (25 S (NOT VOL BULB TEN))
        (26 S (VOL AIR TEN))
        (27 S (NOT VOL AIR TEN))
        (28 S (VOL AIR #10 POS))
        (29 S (VOL BULB #10 POS))    )
(LINKS  (C-CAUSE (1 2) (13))
        (C-CAUSE (1 5) (15))
        (C-CAUSE (20 6) (25 23))
        (C-CAUSE (21 3) (27))
        (C-CAUSE (18 19))
        (C-ENABLE (22 1))
        (C-ENABLE (17 18))
        (C-ENABLE (6 21))
        (RATE-CONFL ((2 3) 4))
        (RATE-CONFL ((5 6) 7))
        (THRESH (4 (8 9)))
        (THRESH (7 (10 11)))
        (S-COUPLE (8 12))
        (S-COUPLE (9 13))
        (S-COUPLE (10 14))
        (S-COUPLE (11 15))
        (S-EQUIV (4 17) (16))
        (S-EQUIV (11 24))
        (S-EQUIV (9 26))
        (ANTAG (12 13))
        (ANTAG (14 15))
        (ANTAG (22 23))
        (ANTAG (26 27))
        (ANTAG (24 25))    )
(INITIAL-WORLD 28 29 20)
(TRIGGER 22)
(RATES  (RATE1 #-2)
        (RATE2 #1)
        (RATE3 (#PLUS RATE1 RATE2))
        (RATE4 #-2)
        (RATE5 #1)
        (RATE6 (#PLUS RATE4 RATE5))   )
))
```

HORN Simulation

```
EVAL: ($SIMULATE-MECH 'HORN)
 MAX TICK COUNT (T = INFINITY) t

           summary of initial world
              air in bulb.
              bulb round.

 ** INITIAL WORLD EVENTS **
(VOL AIR #10 POS) STORED G134 CAUSED BY IW
(VOL BULB #10 POS) STORED G140 CAUSED BY IW
(REFORM BULB) STORED G132 CAUSED BY IW
 ** END INITIAL WORLD **
   G108 HIDDEN CAUSED BY G108
   G109 HIDDEN CAUSED BY G108
(VOL BULB TEN) STORED G138 CAUSED BY G108
   G83 HIDDEN CAUSED BY G80
   G96 HIDDEN CAUSED BY G96
(VOL BULB POSITIVE) STORED G136 CAUSED BY G96
   G89 HIDDEN CAUSED BY G90
   G104 HIDDEN CAUSED BY G104
   G105 HIDDEN CAUSED BY G104
(VOL AIR TEN) STORED G133 CAUSED BY G104
   G79 HIDDEN CAUSED BY G76
   G100 HIDDEN CAUSED BY G100
(VOL AIR POSITIVE) STORED G145 CAUSED BY G100
   G93 HIDDEN CAUSED BY G94

(PRESSURE BULB) STORED G141 CAUSED BY TRIGGER

           summary of tick 0
              bulb starts to collapse.
              forcing the air out.

TICK 0 REAL TIME 2459
   G130 HIDDEN CAUSED BY G130
(ELASTIC COLLAPSE BULB) STORED G142 CAUSED BY G130
   G124 HIDDEN CAUSED BY G124
(CHANGE VOL AIR P1 P2 G1) STORED G144 CAUSED BY G124
(CHANGE VOL AIR P1 P2 G3) STORED G139 CAUSED BY G70
   G122 HIDDEN CAUSED BY G122
(CHANGE VOL BULB P1 P2 G4) STORED G135 CAUSED BY G122
(CHANGE VOL BULB P1 P2 G6) STORED G143 CAUSED BY G64
   G87 HIDDEN CAUSED BY G84

           summary of tick 1
              air in bulb starts to decrease.

TICK 1 REAL TIME 6844
   G74 HIDDEN CAUSED BY G74
   G75 HIDDEN CAUSED BY G74
(VOL AIR #8 NEG) CHANGED G134 CAUSED BY G74
   G104 UNHIDDEN CAUSED BY G106
   G105 UNHIDDEN CAUSED BY G106
(VOL AIR TEN) ERASED G133 CAUSED BY G106
   G78 HIDDEN CAUSED BY G77
   G79 UNHIDDEN CAUSED BY G77
(NOT VOL AIR TEN) STORED G133 CAUSED BY G77
   G100 UNHIDDEN CAUSED BY G101
   G72 HIDDEN CAUSED BY G72
   G73 HIDDEN CAUSED BY G72
(VOL BULB #8 NEG) CHANGED G140 CAUSED BY G72
   G108 UNHIDDEN CAUSED BY G110
   G109 UNHIDDEN CAUSED BY G110
(VOL BULB TEN) ERASED G138 CAUSED BY G110
   G82 HIDDEN CAUSED BY G81
   G83 UNHIDDEN CAUSED BY G81
(NOT VOL BULB TEN) STORED G138 CAUSED BY G81
   G96 UNHIDDEN CAUSED BY G97
```

```
            summary of ticks 2-4
               air in bulb continues to decrease.

TICK 2 REAL TIME 10535
(VOL AIR #6 NEG) CHANGED G134 CAUSED BY $S-INCR
(VOL BULB #6 NEG) CHANGED G140 CAUSED BY $S-INCR

TICK 3 REAL TIME 13566
(VOL AIR #4 NEG) CHANGED G134 CAUSED BY $S-INCR
(VOL BULB #4 NEG) CHANGED G140 CAUSED BY $S-INCR

TICK 4 REAL TIME 15529
(VOL AIR #2 NEG) CHANGED G134 CAUSED BY $S-INCR
(VOL BULB #2 NEG) CHANGED G140 CAUSED BY $S-INCR

            summary of tick 5
               no air left in bulb.
               bulb is fully collapsed.

TICK 5 REAL TIME 17498
(VOL AIR #0 NEG) CHANGED G134 CAUSED BY $S-INCR
   G102 HIDDEN CAUSED BY G102
(NOT VOL AIR POSITIVE) STORED G137 CAUSED BY G102
   G95 HIDDEN CAUSED BY G92
   G93 UNHIDDEN CAUSED BY G92
(VOL AIR POSITIVE) ERASED G145 CAUSED BY G92
   G124 UNHIDDEN CAUSED BY G125
(CHANGE VOL AIR P1 P2 G1) ERASED G144 CAUSED BY G125
(CHANGE VOL AIR P1 P2 G3) ERASED G139 CAUSED BY G71
(VOL BULB #0 NEG) CHANGED G140 CAUSED BY $S-INCR
   G98 HIDDEN CAUSED BY G98
(NOT VOL BULB POSITIVE) STORED G139 CAUSED BY G98
   G91 HIDDEN CAUSED BY G88
   G89 UNHIDDEN CAUSED BY G88
(VOL BULB POSITIVE) ERASED G136 CAUSED BY G88
   G122 UNHIDDEN CAUSED BY G123
(CHANGE VOL BULB P1 P2 G4) ERASED G135 CAUSED BY G123
(CHANGE VOL BULB P1 P2 G6) ERASED G143 CAUSED BY G65

TICK 6 REAL TIME 24789
(CHANGE VOL AIR -B -E G3) NO LONGER CHANGING
   G74 UNHIDDEN CAUSED BY $S-INCR
   G75 UNHIDDEN CAUSED BY $S-INCR
(CHANGE VOL BULB -B -E G6) NO LONGER CHANGING
   G72 UNHIDDEN CAUSED BY $S-INCR
   G73 UNHIDDEN CAUSED BY $S-INCR

TICK 7 REAL TIME 25064

            summary of user  changes
               the bulb is released.
               this causes the bulb to start
               to reform.
               this causes a vacuum to form in the bulb.

LIST ADDITIONAL ACTIVITIES:   (($erase-simu '(pressure bulb)))

(PRESSURE BULB) ERASED G141 CAUSED BY $TIMER
   G130 UNHIDDEN CAUSED BY G131
(ELASTIC COLLAPSE BULB) ERASED G142 CAUSED BY G131
   G86 HIDDEN CAUSED BY G85
   G87 UNHIDDEN CAUSED BY G85
(NO PRESSURE BULB) STORED G142 CAUSED BY G85
   G118 HIDDEN CAUSED BY G118
(CHANGE VOL BULB P1 P2 G5) STORED G141 CAUSED BY G118
   G128 HIDDEN CAUSED BY G128
(VACUUM BULB) STORED G143 CAUSED BY G128
   G116 HIDDEN CAUSED BY G116
(CHANGE VOL AIR P1 P2 G2) STORED G135 CAUSED BY G116
(CHANGE VOL AIR P1 P2 G3) STORED G136 CAUSED BY G70
```

```
(CHANGE VOL BULB P1 P2 G6) STORED G144 CAUSED BY G64

          summary of tick 8
             as the bulb reforms
             air is drawn into the bulb.

TICK 8 REAL TIME 30719
   G74 HIDDEN CAUSED BY G74
   G75 HIDDEN CAUSED BY G74
(VOL AIR #1 POS) CHANGED G134 CAUSED BY G74
   G102 UNHIDDEN CAUSED BY G103
   G72 HIDDEN CAUSED BY G72
   G73 HIDDEN CAUSED BY G72
(VOL BULB #1 POS) CHANGED G140 CAUSED BY G72
   G98 UNHIDDEN CAUSED BY G99

          summary of ticks 9-16
             bulb continues to reform
             drawing air into the bulb.

TICK 9 REAL TIME 32718
(VOL AIR #2 POS) CHANGED G134 CAUSED BY $S-INCR
(VOL BULB #2 POS) CHANGED G140 CAUSED BY $S-INCR

TICK 10 REAL TIME 35589
(VOL AIR #3 POS) CHANGED G134 CAUSED BY $S-INCR
(VOL BULB #3 POS) CHANGED G140 CAUSED BY $S-INCR

TICK 11 REAL TIME 37448
(VOL AIR #4 POS) CHANGED G134 CAUSED BY $S-INCR
(VOL BULB #4 POS) CHANGED G140 CAUSED BY $S-INCR

TICK 12 REAL TIME 39333
(VOL AIR #5 POS) CHANGED G134 CAUSED BY $S-INCR
(VOL BULB #5 POS) CHANGED G140 CAUSED BY $S-INCR

TICK 13 REAL TIME 41244
(VOL AIR #6 POS) CHANGED G134 CAUSED BY $S-INCR
(VOL BULB #6 POS) CHANGED G140 CAUSED BY $S-INCR

TICK 14 REAL TIME 44105
(VOL AIR #7 POS) CHANGED G134 CAUSED BY $S-INCR
(VOL BULB #7 POS) CHANGED G140 CAUSED BY $S-INCR

TICK 15 REAL TIME 45958
(VOL AIR #8 POS) CHANGED G134 CAUSED BY $S-INCR
(VOL BULB #8 POS) CHANGED G140 CAUSED BY $S-INCR

TICK 16 REAL TIME 47815
(VOL AIR #9 POS) CHANGED G134 CAUSED BY $S-INCR
(VOL BULB #9 POS) CHANGED G140 CAUSED BY $S-INCR

          summary of tick 17
             bulb is completely reformed.
             bulb fully contains air.

TICK 17 REAL TIME 49733
(VOL AIR #10 POS) CHANGED G134 CAUSED BY $S-INCR
   G104 HIDDEN CAUSED BY G104
   G105 HIDDEN CAUSED BY G104
(VOL AIR TEN) STORED G145 CAUSED BY G104
   G79 HIDDEN CAUSED BY G76
   G78 UNHIDDEN CAUSED BY G76
(NOT VOL AIR TEN) ERASED G133 CAUSED BY G76
   G116 UNHIDDEN CAUSED BY G117
(CHANGE VOL AIR P1 P2 G2) ERASED G135 CAUSED BY G117
(CHANGE VOL AIR P1 P2 G3) ERASED G136 CAUSED BY G71
   G100 HIDDEN CAUSED BY G100
(VOL AIR POSITIVE) STORED G136 CAUSED BY G100
   G93 HIDDEN CAUSED BY G94
   G95 UNHIDDEN CAUSED BY G94
```

```
(NOT VOL AIR POSITIVE) ERASED G137 CAUSED BY G94
(VOL BULB #10 POS) CHANGED G140 CAUSED BY $S-INCR
   G108 HIDDEN CAUSED BY G108
   G109 HIDDEN CAUSED BY G108
(VOL BULB TEN) STORED G137 CAUSED BY G108
   G83 HIDDEN CAUSED BY G80
   G82 UNHIDDEN CAUSED BY G80
(NOT VOL BULB TEN) ERASED G138 CAUSED BY G80
   G118 UNHIDDEN CAUSED BY G119
(CHANGE VOL BULB P1 P2 G5) ERASED G141 CAUSED BY G119
   G128 UNHIDDEN CAUSED BY G129
(VACUUM BULB) ERASED G143 CAUSED BY G129
(CHANGE VOL BULB P1 P2 G6) ERASED G144 CAUSED BY G65
   G96 HIDDEN CAUSED BY G96
(VOL BULB POSITIVE) STORED G144 CAUSED BY G96
   G89 HIDDEN CAUSED BY G90
   G91 UNHIDDEN CAUSED BY G90
(NOT VOL BULB POSITIVE) ERASED G139 CAUSED BY G90

TICK 18 REAL TIME 58543
(CHANGE VOL AIR -B -E G3) NO LONGER CHANGING
   G74 UNHIDDEN CAUSED BY $S-INCR
   G75 UNHIDDEN CAUSED BY $S-INCR
(CHANGE VOL BULB -B -E G6) NO LONGER CHANGING
   G72 UNHIDDEN CAUSED BY $S-INCR
   G73 UNHIDDEN CAUSED BY $S-INCR

TICK 19 REAL TIME 59742

LIST ADDITIONAL ACTIVITIES:   nil

** FINAL WORLD **
(VOL BULB POSITIVE)
(VOL BULB TEN)
(VOL BULB #10 POS)
(VOL AIR POSITIVE)
(VOL AIR TEN)
(VOL AIR #10 POS)
(NO PRESSURE BULB)
(REFORM BULB)
HORN SIMULATED 59851
```