AD-A037 953 MARYLAND UNIV COLLEGE PARK COMPUTER SCIENCE CENTER F/G 9/2
THE DECLARATIVE REPRESENTATION AND PROCEDURAL SIMULATION OF CAU--ETC(U)
MAR 77 C RIEGER, M GRINBERG N00014-76-C-0477
UNCLASSIFIED TR-513 NL

1 OF 1
ADA037953

END

DATE
FILMED
4-77

COMPUTER SCIENCE TECHNICAL REPORT SERIES

UNIVERSITY OF MARYLAND
COLLEGE PARK, MARYLAND
20742

(12)

TR-513
N00014-76C-0477

March 1977

THE DECLARATIVE REPRESENTATION AND PROCEDURAL SIMULATION OF CAUSALITY IN PHYSICAL MECHANISMS

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

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

This report is distinct in content from, but similar in topic to U.O.M. TR-495, entitled "The Causal Representation and Simulation of Physical Mechanisms", Nov. 1976.

→ to AD-A034 194.

(Cover + p. 1)

The Declarative Representation and Procedural Simulation of Causality in Physical Mechanisms

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

Abstract: A theory of cause-effect representation is used to describe man-made mechanisms and natural laws. The representation, consisting of 10 link types that interconnect events into large declarative patterns, is illustrated on a relatively sophisticated device, the home gas forced air furnace. Next, a procedure and framework for translating the declarative description of a mechanism into a population of associatively triggerable computation units is described. The associative, or procedural, form can then be used to perform a discrete cause-effect simulation of the device. The delcarative to procedural translation, including a simulation trace, is shown for the furnace. Topics of mechanism abstraction and mechanism invention are discussed, and the entire "Mechanisms Laboratory" is placed in the larger perspective of our research into human problem solving.

Keywords: cause-effect representation, declarative-procedural correspondence, simulation of physical systems, spontaneous computation, mechanisms invention

1. Introduction

Man-made physical mechanisms provide an interesting domain in which to study human problem solving and cause-effect knowledge representation. In this paper, we describe and illustrate a representation framework which permits us to express and simulate the commonsense internal cause-effect structure of a variety of man-made mechanisms (both mechanical and electronic), as well as a variety of natural mechanisms (laws of physics, physiological mechanisms, etc).

Since our strategy has been to use the mechanisms domain as a medium for investigating human problem solving and cause-effect knowledge representation, we have developed our "Mechanisms Lab" within the existing framework of our Commonsense Algorithm (CSA) Project, a broader investigation of cognitive mechanisms. Following this strategy has lead us to a mechanisms theory in which the "ground" representation of a mechanism is a declarative, cause-effect graph, but in which simulation is accomplished by transforming such a representation into procedural form, then executing it. Thus, in addition to the theory of representation and simulation, we feel the technique provides an interesting case study in declarative-procedural correspondence.

The discussion of our Mechanisms Laboratory is divided into five main parts: (1) background and theoretical framework of the mechanisms research, (2) cause-effect representation, (3) background and strategy for the simulation aspects of the project, (4) a simulation example, and (5) philosophy of the approach.

2. Motivation

Our motive in this mechanisms research is to understand better human problem solving and cause-effect knowledge representation. Man-made mechanisms provide an excellent medium for carrying out such research because mechanisms are in a sense final snapshots of the human problem solving process. By developing representations for the cause-effect notions inherent to all mechanisms as humans perceive them, we stand a good chance of also gaining insight into how humans encode the various principles and physical laws that are evoked during the mechanism's invention or design.

Besides this theoretical relevance of studying mechanisms, there are some interesting practical applications of a cognitive theory of mechanisms representation and simulation. For example, a mechanisms theory can provide the basis of a potentially powerful CAI facility in which a mechanisms-naive user could interactively explore the design principles and behavior of a device. Provided the system is conversant in the same terms as the user (i.e. higher-level, symbolic terms, rather than lower-level numerical or parametric terms), such interaction can communicate concepts rather than details.

Many other specific applications exist. In one, the theory could provide the basis for medical applications, through the modeling of, e.g., human physiological mechanisms. In a second, the theory could be applied to provide a self-model in systems with a self-maintenance capabilities (martian rovers, 2001 vacuum cleaners?). In a third, the theory could provide the basis of interactive design systems in which engineers specify high level goals to the system, which then does the design. Completely automatic "mechanisms invention" is within the realm of possibility and, we feel, would closely resemble the behavior of current-day problem solving and program synthesis theories. Finally, a future application might be to use the representation as a sort of Dewey Decimal System of cause-effect pattern classification.

3. Background and Related Work

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 more analytical in their approach. In analytical simulations, the mechanism is typically described by parameterizing it in a form suitable for a numerical simulator. The problem with this approach is that the representation of a mechanism is very different from the human cause and effect knowledge of the mechanism.

We are by no means the only ones addressing the issues of symbolic modeling or simulation of physical systems. Two other notable examples are the MYCIN project and an MIT electronic circuit analysis program (EL). 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 symbolically modeling cause-effect mechanisms. EL, Sussman's and Stallman's electronic circuit analysis program ([SS1] and [SS2]), 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 both 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.

4. Theoretical Framework

We have approached the representation and simulation of mechanisms from within the framework of our larger Commonsense Algorithms (CSA) Project. Since one main purpose of the project has been to understand more about how humans might represent and use all sorts of causal knowledge, we have explicitly chosen to develop the mechanisms theory in the context of the larger CSA theory, drawing upon existing CSA mechanisms which themselves seem to be broadly applicable in all aspects of cognitive modeling. Specifically, this

has resulted in (1) adopting as the mechanisms representation the same representation that is used in the other two phases of the project, namely, the plan synthesizer [R1] and the language comprehender [R1] and [R2], and (2) applying our notions of "spontaneous computation" [R3] to simulation. Although the running LISP Mechanisms Laboratory which has resulted is not efficiency-wise competitive with applications simulators, this is of little concern, since efficiency considerations are not relevant to our current goals.

In summary, then, we want to be able to express and simulate mechanisms (and hence, also the physical principles upon which they are founded) in ways that might approximate how humans do these tasks. To accomplish this, we have chosen to embed the simulator in the same framework as the plan synthesis and language comprehension components of our project.

4.1. CSA Cause-Effect Representation

The declarative representation of a mechanism is a cause-effect graph whose nodes are events and whose links are drawn from a set of the ten theoretical forms of inter-event causal interaction. Each event falls into one of four categories: action, tendency, state or statechange. With the exception of "tendency", these terms are intended to reflect their standard connotations. We use the term "tendency" to describe action-like events in which there is an absence of intention. Gravity, for example, is a tendency, since it is a force generator, but has no choice about, or reasons for, acting.

The links of our theory are intended to reflect what we believe to be necessary (and close to sufficient) underlying human concepts relating to causality in physical mechanisms. (These same links are also the basis of the plan synthesizer's representation.) We arrived at several of the ten by obvious intuitive reasoning, and at the remaining ones by considering many mechanisms and attempting to capture the recurring ideas. None of the links by itself is particularly novel to us, or individually provocative; yet, taken as a set, we believe these links are both theoretically significant, and habitable in practice.

4.2. Representation Example: Home Gas Forced-Air Furnace

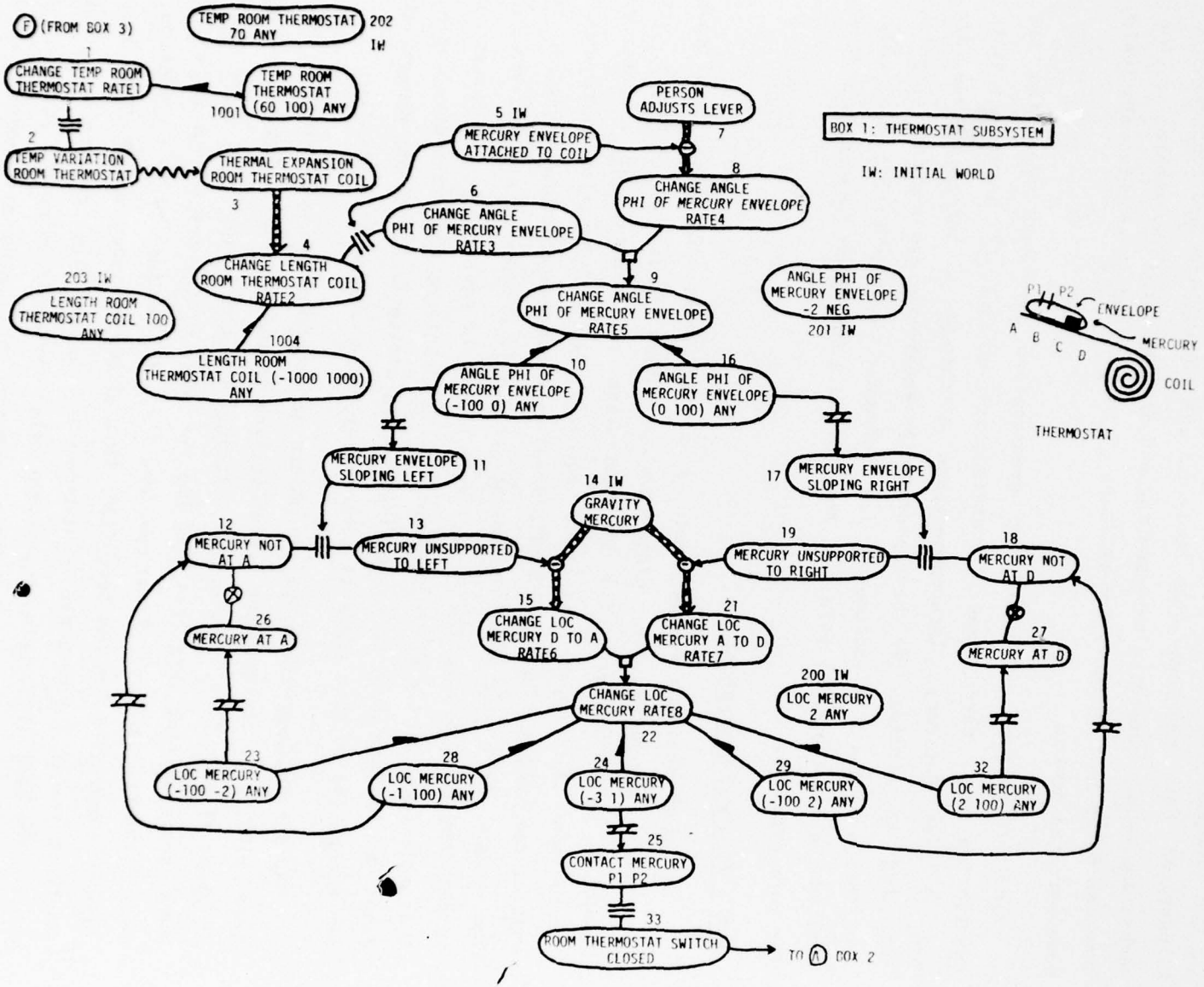
We introduce the CSA mechanisms representation first via an example that illustrates both the individual links and the general size and complexity of mechanisms we are currently representing and simulating. The example is the Home Gas Forced-Air Furnace, and reflects our understanding of this relatively sophisticated device.

For the purpose of convenient presentation, we have segmented the description into three boxes: (1) thermostatic control, (2) heat generation, and (3) heat delivery. Interconnections A through G between boxes are indicated in circles. We have provided an event-wise cross-indexed English description of Box 1 to accompany its schematic representation, but have omitted the English descriptions of Boxes 2 and 3 for space reasons (see [RG1] for more details).

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 causal sources.

As phi changes, there are two points of interest: one when phi is below zero (10), i.e. the envelope begins tilting to the left (11), and one when it is above at zero (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 is above zero), a symmetric system of unsupportedness pertains

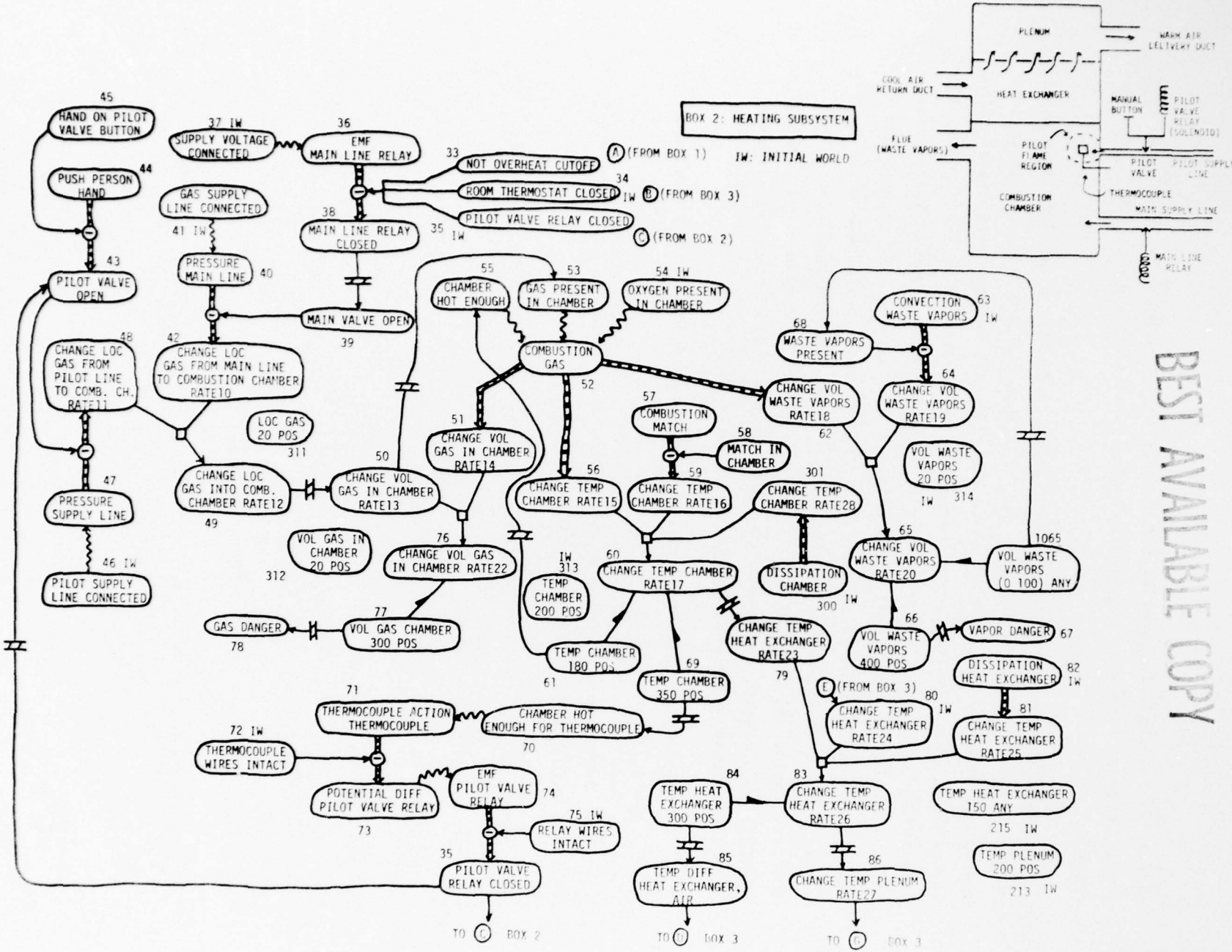

(F) (FROM BOX 3)
TEMP ROOM THERMOSTAT 70 ANY
202 IW
1
CHANGE TEMP ROOM THERMOSTAT RATE1
TEMP ROOM THERMOSTAT (60 100) ANY
1001
2
TEMP VARIATION ROOM THERMOSTAT
THERMAL EXPANSION ROOM THERMOSTAT COIL
3
5 IW
MERCURY ENVELOPE ATTACHED TO COIL
PERSON ADJUSTS LEVER
7
8
CHANGE ANGLE PHI OF MERCURY ENVELOPE RATE4
6
CHANGE ANGLE PHI OF MERCURY ENVELOPE RATE3
4
CHANGE LENGTH ROOM THERMOSTAT COIL RATE2
203 IW
LENGTH ROOM THERMOSTAT COIL 100 ANY
1004
LENGTH ROOM THERMOSTAT COIL (-1000 1000) ANY
9
CHANGE ANGLE PHI OF MERCURY ENVELOPE RATE5
ANGLE PHI OF MERCURY ENVELOPE -2 NEG
201 IW
10
ANGLE PHI OF MERCURY ENVELOPE (-100 0) ANY
16
ANGLE PHI OF MERCURY ENVELOPE (0 100) ANY
MERCURY ENVELOPE SLOPING LEFT
11
17
MERCURY ENVELOPE SLOPING RIGHT
14 IW
GRAVITY MERCURY
12
MERCURY NOT AT A
13
MERCURY UNSUPPORTED TO LEFT
19
MERCURY UNSUPPORTED TO RIGHT
18
MERCURY NOT AT D
26
MERCURY AT A
15
CHANGE LOC MERCURY D TO A RATE6
21
CHANGE LOC MERCURY A TO D RATE7
27
MERCURY AT D
200 IW
LOC MERCURY 2 ANY
CHANGE LOC MERCURY RATE8
22
23
LOC MERCURY (-100 -2) ANY
28
LOC MERCURY (-1 100) ANY
24
LOC MERCURY (-3 1) ANY
29
LOC MERCURY (-100 2) ANY
32
LOC MERCURY (2 100) ANY
25
CONTACT MERCURY P1 P2
33
ROOM THERMOSTAT SWITCH CLOSED
TO (A) BOX 2
BOX 1: THERMOSTAT SUBSYSTEM
IW: INITIAL WORLD
P1
P2
ENVELOPE
A
B
C
D
MERCURY
COIL
THERMOSTAT

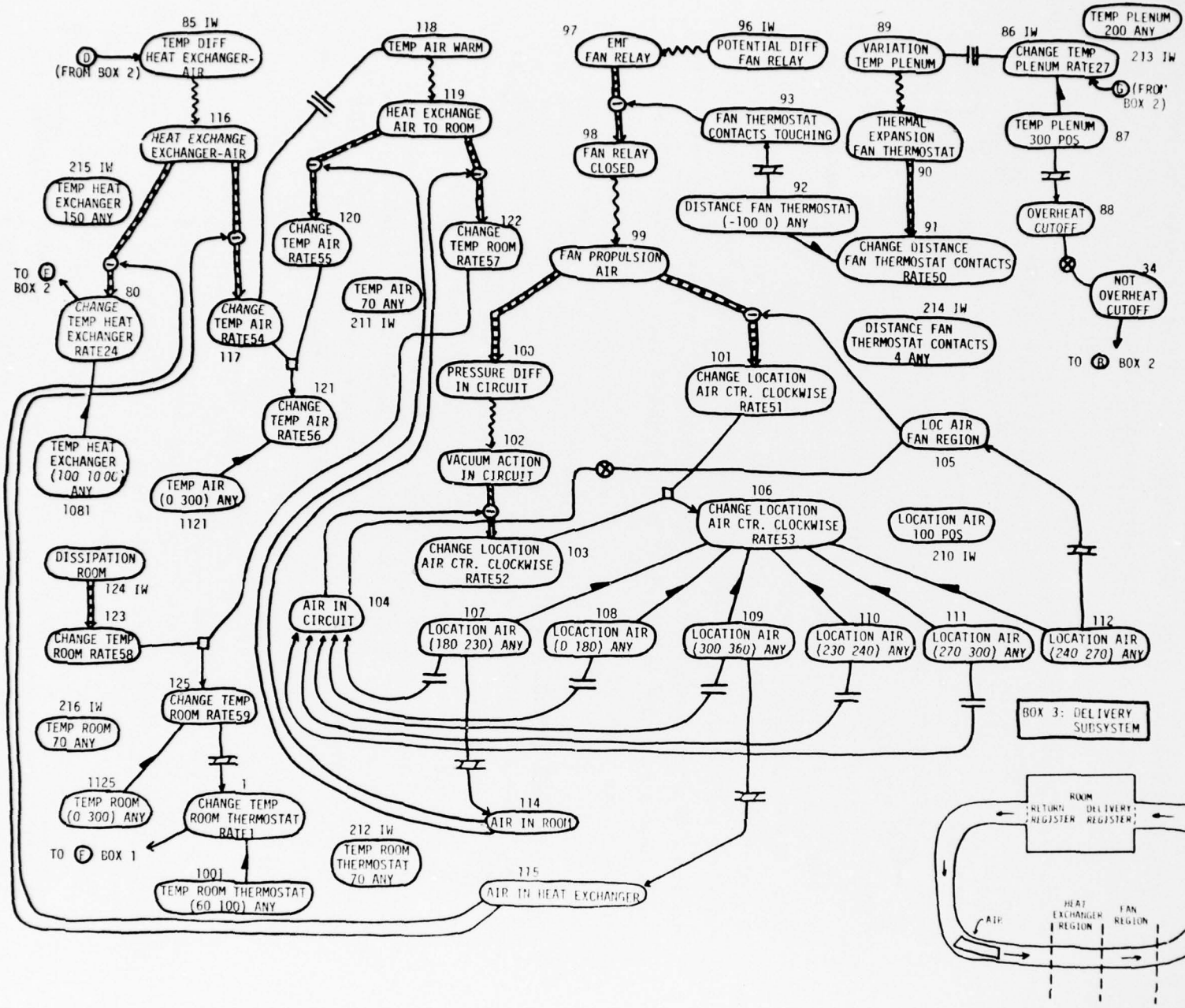
(16,17,18,19,14,21), influencing the mercury to move toward the right. At any moment, the net change in the mercury's location (22) is governed by these two systems.

There are five points of interest (23,24,28,29,32) along the mercury's path of travel (lower part of Box 1). When the left edge of the mercury reaches point A (less than or equal to -2) (23), the mercury will cease being unsupported, and GRAVITY's influence will be severed (i.e. the mercury will stop moving left). While the mercury is between points B and C (i.e. between -3 and 1) (24) there will be physical contact of the mercury and the electrical contact pins P1 and P2 (25). This amounts to the thermostat having closed the furnace's control circuit (33). This condition feeds into Box 2 via tie point A, serving as one precondition for the main supply gas valve opening. Conversely, contact between the mercury and P1 and P2 ceases at points less than -3 or greater than 1.

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

The descriptions of Boxes 2 and 3 are similar, and Box 1 introduces all the link concepts in the representation.

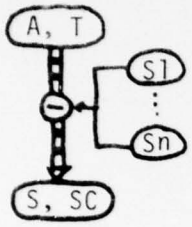




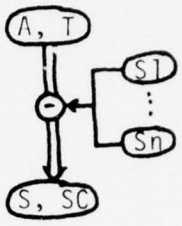
4.3. Mechanisms Cause-Effect Links

With this example in mind, we now give a very brief description of the ten links. See [RG1] and [R2] for more detailed discussions.

Continuous and One-Shot Causal



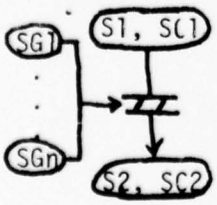
Action A or tendency T causes state S or statechange SC to exist, providing gating conditions S1,...,Sn are in effect. For the continuous form, the action's continued presence is required to sustain the state or statechange (i.e. there is some other unspecified force which would annihilate S or SC if A or T were removed). For the one-shot form, A or T is required only momentarily. The gates govern the effects the action or tendency will have on its environment, in that their continued and simultaneous presence during the action is required for the indicated causality "to flow" from the action or tendency to the state or statechange. These gates are distinct from A or T's enablements; enablements govern the execution of the action itself, independently from any effects it may produce.



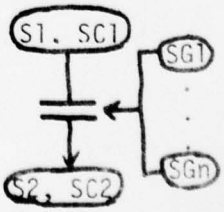
Continuous and One-shot Enablement

State S enables action A or tendency T. For the continuous form, S's continued presence is required in order to begin and sustain A or T. For the one-shot form, S's presence is required only momentarily to allow the action to begin. (Semantically, one-shot enablement occurs when the performance of A or T modifies its environment in a way which liberates the action from the influence of the original enabling condition.)

Continuous and One-Shot State Coupling

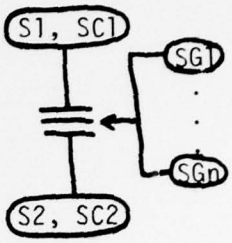


State S1 or statechange SC1 indirectly produces S2 or SC2. For the continuous form, S2's or SC2's continued existence is coupled to S1's or SC1's continued existence. For the one-shot form, S2 or SC2 is independent of S1 or SC1 after the initial coupling. These links provide a means of expressing implicit



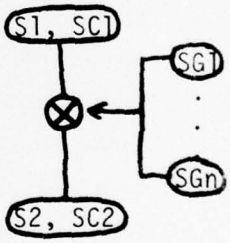
intervening causal relations which are either unknown or irrelevant to some description (i.e. states do not directly cause states). They accept gating states in the same manner as the causal links.

State Equivalence



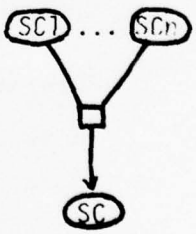
State S1 or statechange SC1 is an eqvivalent formulation of state S2 or statechange SC2, providing gating conditions SG1,...,SGn are present. "Equivalent formulation" means that the two states are paraphrases-syntactically different expressions of the same underlying event. Equivalences, we feel, will always be present in any representation, and hence ought to be dealt with explicitly. Most frequently, an equivalence plays the role of buffering two different points of view of the same event (e.g., the output of one mechanism is "photons exist", the input to another mechanism is "light present", and say, during mechanisms invention, we wish to join the two mechanisms). The existence or non-existence of either equivalenced event implies the existence or non-existence of the other.

State Antagonism



State S1 or statechange SC1 is antagonistic to state S2 or statechange SC2 (i.e. the two events are mutually exclusive), providing gating conditions SG1,...,SGn are in effect. This link is the companion of the state equivalence link.

Rate Confluence



Statechanges SC1,...,SCn represent the culminations of multiple causal sources for a net statechange of some entity with respect to some feature. The net statechange, SC, specifies the composite rate as a symbolic expression which can be dynamically evaluated during a simulation. Syntactically, all contributory statechanges, and the net statechange must reference the same entity, and the same varying feature of that entity (e.g. change in temperature of the heat exchanger).

Threshold

Net statechange, SC, reaches a threshold, S, of interest to the description of the mechanism. S is an instantaneous description of an entity with respect to the feature which is varying in the statechange, e.g. the temperature of the heat exchanger is 400 degrees. We distinguish positive and negative thresholds graphically and in the internal representation and simulation so as to provide for hysteresis.

This concludes the discussion of the CSA cause-effect links relating to mechanisms description. For interested readers, there are several other links relating to motivation and intentionality of human actors. These other links permit us to explore the areas of plan synthesis and language comprehension in the social as well as physical domain within the same representational framework.

We have represented a number of other physical, electrical and electronic mechanisms in these same terms, including (1) a computer flip-flop, (2) the "drinking duck" novelty toy, (3) an incandescent light bulb, (4) a reverse-trap flush toilet, (5) a mechanical oscillator, and (6) descriptions of composite events involving physical laws such as the Bernoulli effect, gravity and momentum. Additionally, we have used the representation to describe a computer algorithm for computing the average of a table of integers, and feel that the CSA links will provide a good language for programming concepts as well as physical laws. These and other examples appear in [RG1], [R1] and [R4].

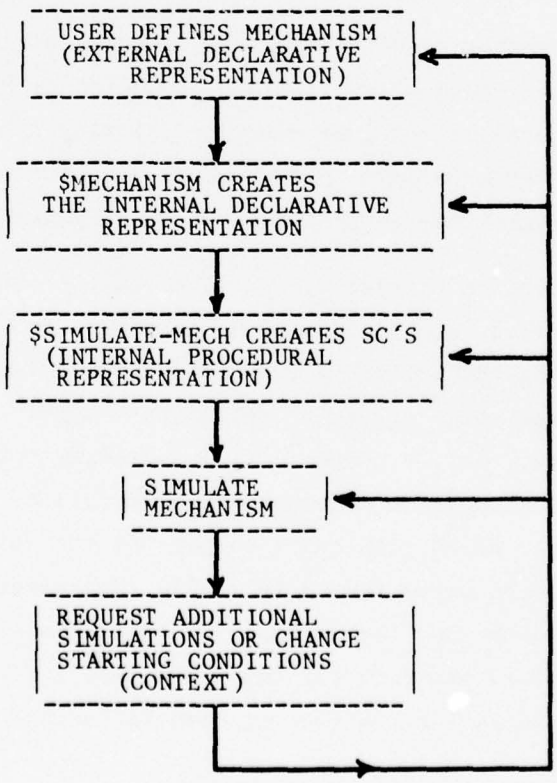
5. Simulation Strategy

As we pointed out earlier, our purpose has not been merely to build a mechanisms description and simulation laboratory, but to build one upon concepts which are generally applicable to a wide spectrum of other types of cognitive modeling as well. We feel that we have succeeded in the description aspect by embedding the mechanisms description language in the same framework as our problem solving and language comprehension languages, namely these CSA

links just covered.

For the simulation strategy, we have drawn upon another aspect of the CSA system that we call the "spontaneous computation" component. This is a generalized implementation of pattern-directed inference, wherein computations occur spontaneously, rather than on demand from another computation.

The simulation strategy (see flow diagram below) is this: convert the declarative (CSA) cause-effect representation of the mechanism to a population of autonomous computation units, each of which models one event in the declarative representation. If each unit in this population of spontaneous computations (SC's) contains a model of all other event schemata that can influence it, (both within, and external to, the particular mechanism in which the event participates) then the modeling SC can be caused to run when all influences are just right by including all those influences in its (potentially complex) invocation pattern. Thus, our strategy is to model each declarative event by an SC whose trigger pattern is sensitive to other prerequisite events in the mechanism.



A simulation will amount to embedding the reactive SC population which models the events in the mechanism in an environment in which static conditions are appropriate for triggering the mechanism. The environment, as well as all instantaneous states during the simulation, are modeled as a collection of database assertions. Triggering of the mechanism will lead to a conceptually parallel avalanche of activity wherein the running of one or more SC units can prompt the running of one or more other units, and so on. The simulation becomes quiescent when no remaining SC's perceive themselves to be relevant.

There are some important issues relating to <u>why</u> we have chosen this strategy for simulation rather than a more "straightforward" strategy that manipulates the declarative representation directly. We will return to the philosophy behind this approach to simulation later. We now describe the CSA system's spontaneous computation component which provides the substrate for the simulator.

5.1. <u>Spontaneous Computation</u>

Our implementation of spontaneous computation is a generalization of the pattern-directed invocation notions embodied in PLANNER [SWC1] and CONNIVER [MS1]. By "generalization", we mean specifically that we have provided for the specification and organization of more complex invocation patterns, and for more complex hierarchical organizations of SC populations.

The invocation (trigger) pattern of an SC in the CSA system is constructed from nested n-tuples composed in virtually any degree of complexity using the logical relations AND, OR and ANY. Each component of the trigger can be one of the following types: (1) associative, (2) non-associative, or (3) computable. Associative trigger components come to be organized into a reactive data structure we call a "trigger tree"; these are the components which can react to passing stimuli (to be defined) and cause the entire pattern to be tested (polled). Non-associative trigger components represent aspects of the trigger's environment that must be true (when explicitly polled) in order for the SC to fire fully, but which themselves are incapable of initiating the firing. Computables are any EVALable LISP forms

other than (1) or (2), and must evaluate non-NIL for full triggering of the pattern to occur.

To illustrate, suppose we wish to create an SC which will fire (i.e. spontaneously execute) whenever "the temperature of the heat exchanger is greater than 400 degrees in coincidence with either valve A or valve B being open". We would express this condition by "planting" the following pattern in a reactive trigger tree:

```
($PLANT '(AND (+ 1 (TEMP XCHGR -X))
              (GREATERP -X 400)
              (OR (- 1 (OPEN VALVE-A))
                  (- 1 (OPEN VALVE-B))))
         <some body>
         <some tree>)
```

The "+" form denotes an associative component, the "-" form denotes a non-associative component, and the GREATERP is a computable denoting itself. The integer 1's indicate how much effort (in terms of database fetches by the deductive component) can be expended in the process of determining the presence or absence of all remaining parts of the pattern whenever the pattern is triggered via an associative component. Variables are prefixed by hyphen signs and are global to the entire trigger pattern, in that variables with the same name must be consistently bound across the entire pattern.

An SC body is an arbitrary EVALable LISP expression. Each SC is context sensitive, in that it can be masked and unmasked independently within its trigger tree. (The simulator relies on this feature to prevent looping in certain representation patterns.) Entire trigger trees are also context sensitive, as will be described.

The population of SC's represented by some trigger tree is caused to react to some stimulus (a fully-constant nested n-tuple) in either of the following ways:

```
($ACTIVATE <trigger tree> <stimulus>)
or
(<trigger tree> <stimulus>)
```

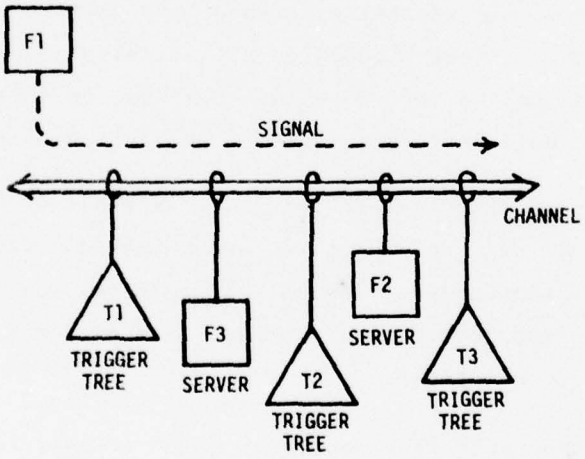
where, in the latter form, the tree is used semantically as a function. Application of a trigger tree to a stimulus yields a queue of SC's which are determined to have been fully triggered, i.e., initially triggered through one

component, then proven fully applicable by calls on database/deductive system for remaining components.

5.2. Channels

Beyond this organization of complex trigger patterns into trigger trees, the CSA SC component provides for the higher-level hierarchical organization of trigger trees around constructions called "channels". An SC channel is intended to be the analog of a hardware channel, and a generalization of the PLANNER/CONNIVER pattern-directed invocation scheme.

A typical channel is depicted below, accompanied by the calls on the channel-constructing function, $CONNECT, which would set it up. A channel has one-dimensional extent; signals can be injected at arbitrary points, to propagate in a specified direction, and these signals constitute the stimuli to which trigger trees can react.



Two types of entities can be attached to the channel at "tap points", of which there may be virtually any number. Attached entities are either trigger trees ("watchers") or arbitrary LISP functions ("servers").

The channel construct itself amounts to a fracturing of what would otherwise be a private inter-function calling sequence. If, wherever function F1 would ordinarily call function F2, we now arrange to have F1 inject a request to F2 over a channel (i.e. place the calling arguments as a signal on

the channel) other entities will then have a chance to inspect the signal as it passes, and (1) simply react benignly, (2) alter the signal as it passes, or (3) block the signal altogether. In this way, all function calls can be "fishbowled" by other servers and trees of spontaneous computations.

There may be any number of channels, and any given server or trigger tree can be attached to numerous channels, or to the same channel at numerous tap points. The SC's in one trigger tree attached to one channel may, when they run, inject signals on other channels, and so forth, providing for complex hierarchical organizations of spontaneous computations into conceptual populations. The connection of an entity to a channel is context sensitive. and entities are disconnected via the function ($DISCONNECT <entity> <channel>).

Signals are injected onto a channel via the function:

```
($INJECT <signal> <to-server> <on-channel>
         <in-relation> <to-tap-point> <moving-in-direction>)
```

i.e., inject a signal to some server on some channel at the injection point indicated by <in-relation> and <to-tap-point>, propagating in some direction. Thus, wherever F1 used to call F2 directly, when participating in the channel facility, it will now inject signals to F2 on some channel on which F2 exists as a server, e.g.:

```
($INJECT '(TEMP XCHGR 400) '$STORE 'CHANNEL1
         'AT 'LEFTEND 'RIGHT)
```

That is, store a fact by injecting it to $STORE of CHANNEL1, starting at the LEFTEND, propagating RIGHT. As the signal moves past watchers, or as the response signal back from $STORE moves past response-watchers, numerous SC's may be triggered and run.

The SC's that the mechanisms simulator creates as the procedural representation of a mechanism all exist in two trees attached to two channels. The passing signals on these channels will be the symbolic descriptions of changing events in the simulation, entering and exiting the database via these channels. When it runs, a fully-triggered SC will place a new event description on one of these channels, and possibly mask or unmask itself

and/or other SC's in the population (to be described).

We feel the trigger tree/channel paradigm is a powerful one. Currently, the mechanisms simulator employs only these two relatively simple channels in performing its tasks, (analogous to the store and erase channels of PLANNER and CONNIVER) but we expect it to utilize the full potentials of this system in future phases of the project, especially when we address the topic of mechanisms invention.

This concludes our brief sketch of the CSA SC component insofar as it relates to our purposes here. For a more complete discussion, including several other theoretical applications of the SC system in the areas of inference, problem solving and language comprehension, see [R3].

6. The Mechanisms Simulator

The simulation subsystem is a collection of LISP functions that take the internal declarative representation of a mechanism, convert it to a population of spontaneous computations, 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 database 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.

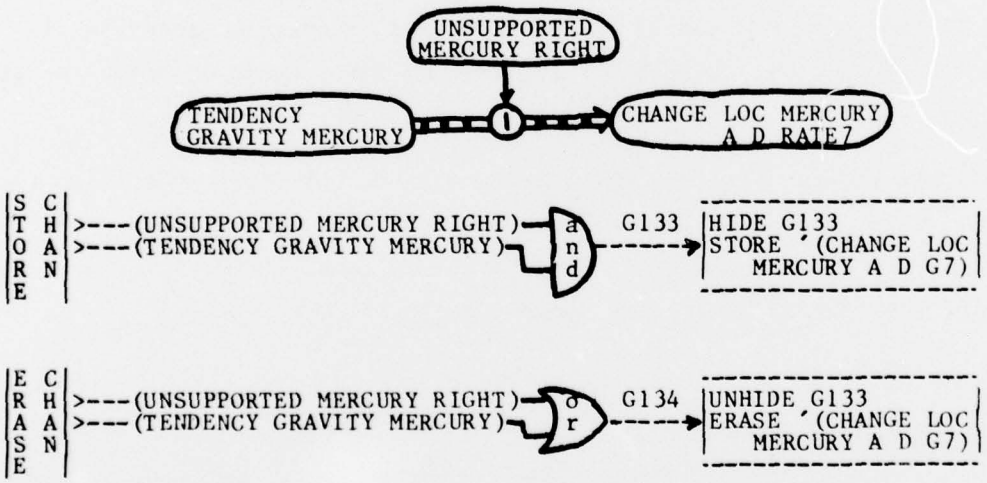
The initiation of the simulator is caused by the assertion of some event: semantically, either an action performed by an external actor, a tendency relating to a natural force, or a state, as derived, e.g., as an output from the simulation of another mechanism. This latter initiating event will provide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

Each link in the internal declarative representation of a mechanism has a set of SC's automatically created for it according to a set of rules for each

theoretical link type. An SC is activated when its preconditions have been satisfied. The running of an SC will generally cause a database modification or a masking or unmasking of another SC.

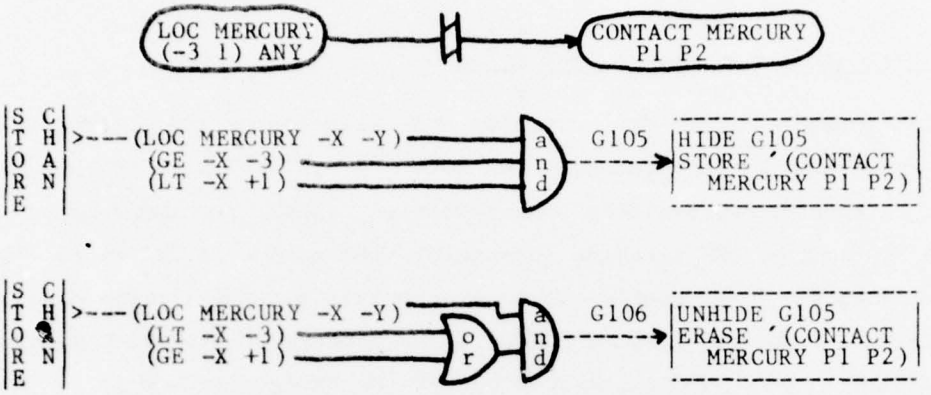
6.1. Declarative to Procedural Conversion

We now briefly give two examples to illustrate the process of declarative-procedural conversion. The SC's are diplayed with the following conventions. Each SC representing some potential event is imagined to be "eyeballing" a path to the database (channel). The path type (STORE or ERASE) is located between the double bars on the left side of the display. The logical connectives in the created SC's trigger pattern are to the right of the triggering events and the arrows out of the triggering events flow into these connectors. On the far right side of the figure 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.



To illustrate, in the example of the SC's created for the continuous causal (C-CAUSE) link governing the movement of the mercury in the furnace's thermostat, when the tendency GRAVITY or the state "mercury unsupported on the right" is asserted and the other event is currently in the database (i.e., also present), then the state-change of the location of the mercury is stored

(i.e. asserted) in the database. If either the tendency GRAVITY or "mercury unsupported on the right" is deasserted (i.e erased) from the database, then the state-change of the location of the mercury is erased from the database.



The second example is the SC's created for the continuous coupling (C-COUPLE) link which watches for the mercury to enter the region between P1 and P2. When the location of the mercury is between -3 and 1 there is contact between the mercury, *P1*, and *P2* (*i.e the contact event is asserted in the database*). When the location of the mercury moves above or below the given range, the contact event is erased from the database.

This illustrates the conversion process on two of the simpler link types. More complex processing is required for other links. For example, the rate confluence (R-CONFL) link requires symbolic rate computations to be performed. Details of the conversion process appear in [RG1].

6.2. Simulation Example

We have successfully simulated the entire furnace, illustrated earlier in its declarative form, although the simulation must be done in three pieces (the whole simulation will not fit into our limited memory). The furnace does indeed cut on, heat the room up, then cut off, providing a trace of all events in the sequence. To convey a picture of the simulator's operation, we include an excerpted and annotated sequence showing the simulation of the thermostatic control portion of the furnace.

Input Syntax for Box1

The Mechanisms Lab requires that each mechanism's CSA description be coded into a machine readable form. This form is the external declarative CSA representation and is used as input to a mechanisms defining function called $MECHANISM. This function generates the internal declarative representation of the mechanism from the external declarative representation. The call on $MECHANISM with the appropriate external declarative form for the thermostatic control section of the furnace is shown below.


```
($MECHANISM '(
(NAME BOX1)
(EVENTS
    (1 SC (CHANGE TEMP ROOM-THERMOSTAT P1 P2 RATE1))
    (2 S (VARIATION TEMP ROOM-THERMOSTAT))
    (3 T (TENDENCY THERMAL-EXPANSION R-T-COIL))
    (4 SC (CHANGE LENGTH-MM R-T-COIL P3  P4 RATE2))
    (5 S (ATTACHED MERCURY-ENVELOPE R-T-COIL))
    (6 SC (CHANGE ANGLE-PHI MERCURY-ENVELOPE P5 P6 RATE3))
    (7 A (ADJUST LEVER))
    (8 SC (CHANGE ANGLE-PHI MERCURY-ENVELOPE P7 P8 RATE4))
    (9 SC (CHANGE ANGLE-PHI MERCURY-ENVELOPE P9 P10 RATE5))
    (10 S (ANGLE-PHI MERCURY-ENVELOPE (#-100 #0) ANY))
    (11 S (SLOPING MERCURY-ENVELOPE LEFT))
    (12 S (NOT LOCA MERCURY A))
    (13 S (UNSUPPORTED MERCURY LEFT))
    (14 T (TENDENCY GRAVITY MERCURY))
    (15 SC (CHANGE LOC MERCURY D A RATE6))
    (16 S (ANGLE-PHI MERCURY-ENVELOPE (#0#1 #100) ANY))
    (17 S (SLOPING MERCURY-ENVELOPE RIGHT))
    (18 S (NOT LOCA MERCURY D))
    (19 S (UNSUPPORTED MERCURY RIGHT))
    (21 SC (CHANGE LOC MERCURY A D RATE7))
    (22 SC (CHANGE LOC MERCURY P11 P12 RATE8))
    (23 S (LOC MERCURY (#-100 #-2) ANY))
    (24 S (LOC MERCURY (#-3 #1) ANY))
    (25 S (CONTACT MERCURY P1 P2))
    (26 S (LOCA MERCURY A))
    (27 S (LOCA MERCURY D))
    (28 S (LOC MERCURY (#-1 #100) ANY))
    (29 S (LOC MERCURY (#-100 #2) ANY))
    (32 S (LOC MERCURY (#2 #100) ANY))
    (33 S (CLOSED ROOM-THERMOSTAT))
    (200 S (LOC MERCURY #2 ANY))
    (201 S (ANGLE-PHI MERCURY-ENVELOPE #-2 NEG))
    (202 S (TEMP ROOM-THERMOSTAT #70 ANY))
    (203 S (LENGTH-MM R-T-COIL #100 ANY))
    (1004 S (LENGTH-MM R-T-COIL (#-1000 #1000) ANY))
    (1001 S (TEMP ROOM-THERMOSTAT (#60 #100) ANY))   )
(LINKS
    (S-EQUIV (1 2))
    (C-ENABLE (2 3))
    (C-CAUSE (3 4))
    (S-EQUIV (4 6) (5))
    (C-CAUSE (7 8) (5))
    (RATE-CONFL ((6 8) 9) )
    (THRESH (9 (10 16)))
    (C-COUPLE (10 11))
    (S-EQUIV (12 13) (11))
    (C-CAUSE  (14 15) (13))
    (ANTAG (12 26))
    (C-COUPLE (23 26))
    (C-COUPLE (32 27))
    (RATE-CONFL ((15 21) 22))
    (THRESH (22 (23 24 32 28 29)))
    (C-COUPLE (28 12))
    (C-COUPLE (29 18))
    (C-COUPLE (24 25))
    (S-EQUIV (25 33))
    (ANTAG (27 18))
    (C-CAUSE (14 21) (19))
    (C-COUPLE (16 17))
    (S-EQUIV (18 19) (17))
    (THRESH (4 (1004)))
    (THRESH (1 (1001)))   )
(INITIAL-WORLD  14  5 200 201 202 203)
(TRIGGER 1)
(RATES
    (RATE1 RATE59)
    (RATE2 (#QUOTIENT RATE1 #2#0))
    (RATE3 RATE2)
    (RATE4 #1)
    (RATE5 (#PLUS RATE3 RATE4))
    (RATE6 #-4)
    (RATE7 #4)
    (RATE8 (#PLUS RATE7 RATE6))
    (RATE59 #1)   )
))
```

Simulation Trace

The following illustrates the output of the simulator. There are 72 SC's created for this section of the furnace. For space reasons, only a small portion of the trace has been left intact to convey the flavor of the simulation. For clarity, the edited trace has all database activities removed except those which affect the location of the mercury and those events directly linked to them. The left-hand side of the trace is an English description of the simulation.

```
($SIMULATE-MECH 'BOX1)
MAX TICK COUNT (T = INFINITY) 7

 ** INITIAL WORLD EVENTS **

(TENDENCY GRAVITY MERCURY) STORED G143 BY                    GRAVITY  always in effect.
(ATTACHED MERCURY-ENVELOPE R-T-COIL) STORED G144 BY IW       envelope and coil attached.
(LOC MERCURY #2 ANY) STORED G145 BY IW                       initial location of mercury is 2.
(ANGLE-PHI MERCURY-ENVELOPE #-2 NEG) STORED G146 BY IW       initial angle is -2.
(TEMP ROOM-THERMOSTAT #70 ANY) STORED G147 BY IW             initial room temperature is 70.
(LENGTH-MM R-T-COIL #100 ANY) STORED G148 BY IW              initial length of coil is 100 mm.

 ** END INITIAL WORLD **

(SLOPING MERCURY-ENVELOPE LEFT) STORED G149 BY G111          sloping left since angle is -2.
(NOT LOCA MERCURY A) STORED G150 BY G103                     mercury not at A since at 2.

  **** unedited trace begins ****

(UNSUPPORTED MERCURY LEFT) STORED G151 BY G121               mercury unsupported at left since at 2.
   G135 HIDDEN BY G135                                       c-causal SC hidden.
(CHANGE LOC MERCURY D A G6) STORED G152 BY G135              mercury moving to A since left sloping and unsupp.
(CHANGE LOC MERCURY P11 P12 G8) STORED G153 BY G75           net state-change to mercury.
   G98 HIDDEN BY G95
   G99 HIDDEN BY G99
(LOCA MERCURY D) STORED G154 BY G99                          location of mercury is D since at 2.
   G94 HIDDEN BY G91

(CHANGE TEMP R-T P1 P2 G1) STORED G155 BY TRIGGER            temp change to thermostat is trigger for mechanism.

TICK 0 REAL TIME 26487

   G129 HIDDEN BY G129
   G130 HIDDEN BY G129
(VARIATION TEMP ROOM-THERMOSTAT) STORED G156 BY G129         temp variation caused by trigger.
   G141 HIDDEN BY G141
(TENDENCY THERMAL-EXPANSION R-T-COIL) STORED G157 BY G141    enabled by temp variation.
   G139 HIDDEN BY G139
(CHANGE LENGTH-MM R-T-COIL P3 P4 G2) STORED G158 BY G139     changing coil length caused by thermal-exp.

    **** end of unedited trace.  ****

(LOC MERCURY #-2 NEG) CHANGED G145 BY G87                    mercury moves to -2.
(CONTACT MERCURY P1 P2) STORED G161 BY G107                  contact between mercury-P1-P2 since location is -2.
(CLOSED ROOM-THERMOSTAT) STORED G162 BY G113                 thermo closed because of contact of mercury-P1-P2.
(NOT LOCA MERCURY A) ERASED G150 BY G104                     mercury now  at A so must erase this event.
(UNSUPPORTED MERCURY LEFT) ERASED G151 BY G123               mercury now supported on left.
(CHANGE LOC MERCURY D A G6) ERASED G152 BY G136              loc of mercury no longer changing due to left slope.
(CHANGE LOC MERCURY P11 P12 G8) ERASED G153 BY G76           net change to location of mercury no longer occuring.
(LOCA MERCURY A) STORED G153 BY G96                          mercury now at A.
(NOT LOCA MERCURY D) STORED G152 BY G101                     mercury not at D.
(LOCA MERCURY D) ERASED G154 BY G93

TICK 1 REAL TIME 35956

(CHANGE LOC MERCURY -B -E G8) NO LONGER CHANGING             mercury no longer moving.

TICK 5 REAL TIME 50528

(UNSUPPORTED MERCURY RIGHT) STORED G154 BY G117              mercury becomes unsupported on right.
(CHANGE LOC MERCURY A D G7) STORED G151 BY G133              mercury start moving toward D.
(CHANGE LOC MERCURY P11 P12 G8) STORED G150 BY G75           net location of mercury now changing.

TICK 6 REAL TIME 55813

(LOC MERCURY #2 POS) CHANGED G145 BY G87                     new mercury position is 2.
(CONTACT MERCURY P1 P2) ERASED G161 BY G108                  mercury no longer in contact with P1-P2 since at 2.
(CLOSED ROOM-THERMOSTAT) ERASED G162 BY G115                 thermostat no longer closed.
(LOCA MERCURY A) ERASED G153 BY G106                         mercury not at A since its at D.
(NOT LOCA MERCURY A) STORED G153 BY G98
(NOT LOCA MERCURY D) ERASED G152 BY G102                     mercury at D.
(UNSUPPORTED MERCURY RIGHT) ERASED G154 BY G119              mercury supported on right since at D.
(LOCA MERCURY D) STORED G150 BY G94

LIST ADDITIONAL ACTIVITIES: NIL

** FINAL WORLD **
(TEMP ROOM-THERMOSTAT #79 POS)                               current room temperature is 79.
(ANGLE-PHI MERCURY-ENVELOPE #2#0 POS)                        current angle is 2.
(LENGTH-MM R-T-COIL #1#04E2 POS)                             current coil length is 104 mm.
(LOCA MERCURY D)                                             current mercury location is D.
(NOT LOCA MERCURY A)                                         mercury not at A.
(LOC MERCURY #2 POS)                                         mercury located at 2.
(SLOPING MERCURY-ENVELOPE RIGHT)                             envelope sloping right.
(CHANGE ANGLE-PHI MERCURY-ENVELOPE P9 P10 G5)                envelope angle changing.
(CHANGE ANGLE-PHI MERCURY-ENVELOPE P5 P6 G3)
(CHANGE LENGTH-MM R-T-COIL P3 P4 G2)                         coil length changing.
(TENDENCY THERMAL-EXPANSION R-T-COIL)                        thermal expansion still occuring.
(VARIATION TEMP ROOM-THERMOSTAT)                             variation in temperature of thermostat.
(CHANGE TEMP ROOM-THERMOSTAT P1 P2 G1)                       temp of thermostat still changing.
(ATTACHED MERCURY-ENVELOPE R-T-COIL)                         envelope still attached to coil.
(TENDENCY GRAVITY MERCURY)                                   gravity still affecting the mercury.

BOX1 SIMULATED
```

7. Simulation Philosophy

Why do simulation this way? Namely, why do it symbolically, and why go through the contortions of converting the declarative form to spontaneous computation units? Clearly, it would be possible to simulate a mechanism by applying relatively simple graph algorithms directly to the declarative representation.

To understand why we have adopted this SC strategy, reflect on the nature of any physical system - a mechanism in particular. A mechanism is built from physical principles and components that act autonomously, in the sense that they are governed by physical laws that "run in parallel". The mechanism just happens to work in desired ways because the inventor has managed to identify, harness, and coordinate a population of autonomous agents. In this setting, very minor or very local alterations to one component or its environment can propagate to all parts of the mechanism in a falling-dominoes fashion. In particular, embedding the mechanism in alien or novel environments can significantly alter the micro and macro behavior of the mechanism in ways that will always relate to the individual components of the mechanism, viewed as autonomous agents, but never to the mechanism as a whole. Certainly, the net effect of embedding the mechanism in an environment will be to change the cumulative, overt behavior of the mechanism; but the overt behavior changes will be nothing more that the sum of many smaller, possibly unrelated influences.

By converting the mechanism's declarative form into SC form, we effectively crack the description open, exposing all the individual cause-effect relationships directly to the environment. Each is free to behave as its trigger pattern dictates. Thus, when we embed the eviscerated form in a new environment, each cause-effect relationship must fend and produce results on its own.

In particular, this makes possible the more detailed study or debugging of the mechanism when placed in an environment where there are other mechanisms, as occurs, say, when a new mechanism is invented out of existing mechanisms and physical laws. For example, suppose that two mechanisms, designed separately, are thrust into the same environment and made to coexist, as happens when the two become part of a larger mechanism. Without cracking

each decsription apart and casting the resulting spontaneous units into a large population in which the two mechanism's boundaries are lost, it would not be apparent how the two might interact. But with such a strategy, event-wise crosstalk is more readily discovered.** A case in point was the much publicized glitch in Polaroid's new SX-70 camera, where the sensitive electronics were interfered with by noise spikes produced by the concurrent operation of the picture ejecting motor. The interaction was a simple, but obscure one, and was solved by clever timing of events to remove the concurrency.

** Forcing two independent mechanisms to coexist in one environment is the analogy of forcing brother subgoals to coexist in one environment in plan synthesis; in both domains, there is the possibility of unanticipated interactions. See [RL1].

7.1. A Related Issue: Mechanisms Abstraction

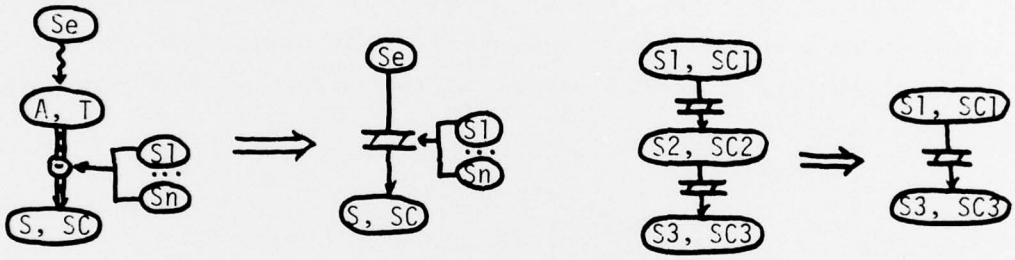
There is quite often a need to suppress much of the representation or simulation detail present in the system's full model of a given mechanism. There would be a need for suppresssion of detail, for example, when we wished to provide a very high-level overview of a mechanism to serve, say, as a black box in a larger invention effort, or as an introduction of the mechanism to a totally naive CAI user. Since the suppression of mechanism detail is analogous to the suppression of detail in text, we use the term "abstraction".

One of our current areas of interest is in automating the process of mechanisms abstraction. Note that we already have provisions in the representation for expressing abstracted forms: the state coupling link. This link allows us the freedom of direct state-state causality. The abstraction technique can therefore be one of syntactically replacing certain patterns in the detailed representation with state coupling links.

It appears that there will be only a small number of syntactic replacement rules. Two of the most obvious abstracting rules are depicted below. Such rules, applied transitively, could perform both minor and major abstractions. A major abstraction would amount to a simple coupling between

the input and output states of a mechanism, e.g. "A falling temperature causes the thermostat to close."

Such abstracting would be applied to the declarative representation before simulation, so that the converted computation units would directly reflect the abstraction. Abstraction could be applied uniformly over an entire mechanism, or locally to excise parts which are not of immediate interest. Non-uniform abstracting, in which all uninteresting sections have been defocussed, would deliver practical advantages by allowing the simulator to devote more time to the relevant aspects of the simulation. This could be important if, for example, we suspect the audio stages of a radio are at fault, but know the power supply is functional. Before simulation, the radio's power supply description could be defocussed before conversion to simulation representation (i.e. population of SC's).



Two Defocussing Rules

In addition to simple syntactic defocussing, we will also be exploring more semantic forms. Semantic mechanisms abstraction would be useful if, for example, we wished to know about the gas furnace with respect particular aspect of its operation, such as "heat production". We might scan through the description, assessing each event according to its semantic relevance to the concept of heat production. Then, using the highly ranked events as milestone events, the syntactic abstraction procedure could be applied to yield an abstraction with respect to the desired point of view, i.e. one which retained only the milestone states.

8. Conclusions

We have presented a theory of mechanism description and simulation which

is based on tenets and processes we hypothesize to be applicable to other aspects of human cognition. We feel the theory of cause-effect representation is sound and expressive for a wide variety of mechanisms, and that it bears significance as a theory of human cause-effect knowledge representation.

Our next specific goals are threefold: (1) to make the acquisition of new mechanism patterns interactive, having the model prompt the user, and verify that the user's use of the representation coincides with the model's notions of semantic well-formedness, (2) to study the processes of mechanisms abstraction, and (3) to apply the existing CSA plan synthesizer to the task of mechanisms invention, involving the simulator in a debugging loop.

Acknowledgments

We wish to thank the other members of the Maryland CSA group (Phil London, Mache Creeger, John Boose, Georgy Fekete and Shmuel Peleg) for their participation in numerous discussions and for their suggestions. We wish also to thank the Office of Naval Research for its support of this research under Grant N00014-76c-0477.

9. 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

[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 (to appear in Cognitive Science)

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

[RG1] Rieger, C., and Grinberg, M., The Causal Representation and Simulation of Physical Mechanisms, University of Maryland, TR 495, 1976

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

[RL1] Rieger, C., and London, P. Subgoal Protection and Unravelling during Plan Synthesis University of Maryland, TR 512, 1977

[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

Off of Naval Research
Branch Office, Boston
495 Summer St.
Boston, Mass. 02210

New York Area Office
715 Broadway-5th Floor
New York, N.Y. 10003

Mr. E. H. Gleissner
Naval Ship R+D Center
Computation and Math Department
Code 18
Bethesda, Maryland 20084

Capt. Grace M. Hopper
NAICOM/MIS Planning Branch
OP-916D
Off, Chf. of Naval Op.
Washington, D.C. 20350

Mr. Kin B. Thompson
Technical Director
Information Systems Div. OP-91T
Off., Chf. of Naval Op.
Washington, D.C. 20375

Naval Research Lab.
Technical Info. Division
Code 2627
Washington, D.C. 20375

Dr. A.L. Slafkosky
Scientific Advisor
Commandant, USMC
Code RD-1
Washington, D.C. 20380

National Security Agcy.
Attn: Dr. Maar
Fort Meade, Maryland 20755

Off. of Naval Research
Code 1021P
Arlington, Va. 22217

Asst. Chief for Tech.
ONR Dept. of Navy
Code 200
Arlington, Va. 22217

Off. of Naval Research
Information Sys. Program
Code 437
Arlington, Va. 22217

Off. of Naval Research
Code 455
Arlington, Va. 22217

Off. of Naval Research
Code 458
Arlington, Va. 22217

Defense Documentn. Cent.
Cameron Station
Alexandria, Va. 22314

Off. of Naval Research
Branch Office, Chicago
536 South Clark St.
Chicago, Ill. 60605

Off. of Naval Research
Branch Off., Pasadena
1030 East Green St.
Pasadena, Calif. 91106

Naval Electron. Lab. Ctr.
Adv. Software Tech. Div.
Code 5200
San Diego, Calif. 92152

unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER (6)	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): The Declarative Representation and Procedural Simulation of Causality in Physical Mechanisms.	5. TYPE OF REPORT & PERIOD COVERED (9) Technical rept.
6. PERFORMING ORG. REPORT NUMBER (14) TR-513	
7. AUTHOR(s) (10): Chuck Rieger and Milt Grinberg	8. CONTRACT OR GRANT NUMBER(s) (15): N00014-76C-0477
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Computer Science Ctr. University of Maryland College Park, Maryland 20742	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Information Systems Branch Office of Naval Research Washington, D.C. 20305 (11)	12. REPORT DATE: March 1977
	13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) (12) 31 p.	15. SECURITY CLASS. (of this report): unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Cause-effect representation
Declarative-procedural correspondence
Simulation of Physical systems
Spontaneous computations
Mechanism Invention

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Abstract: A theory of cause-effect representation is used to describe man-made mechanisms and natural laws. The representation, consisting of 10 link types that interconnect events into large declarative patterns, is illustrated on a relatively sophisticated device, the home gas forced air furnace. Next, a procedure and framework for translating the declarative description of a mechanism into a population of associatively triggerable computation units is described. The associative, or procedural, form can then be used to perform a discrete cause-effect simulation of the device. The delcarative to procedural translation, including a simulation trace, is shown for the furnace. Topics

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

403018

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ABSTRACT (continued)

of mechanism abstraction and mechanism invention are discussed, and the entire "Mechanisms Laboratory" is placed in the larger perspective of our research into human problem solving.

Keywords: cause-effect representation, declarative-procedural correspondence, simulation of physical systems, spontaneous computation, mechanisms invention