THE COMMONSENSE ALGORITHM AS A BASIS FOR COMPUTER MODELS OF HUMAN MEMORY, INFERENCE, BELIEF AND CONTEXTUAL LANGUAGE COMPREHENSION

Chuck Rieger
Department of Computer Science
University of Maryland

ABSTRACT

The notion of a commonsense algorithm is presented as a basic data structure for modeling human cognition. This data structure unifies many current ideas about human memory and information processing. The structure is defined by specifying a set of proposed cognitive primitive links which, when used to build up large structures of actions, states, statechanges and tendencies, provide an adequate formalism for expressing human plans and activities, as well as general mechanisms and computer algorithms. The commonsense algorithm is a type of framework (as Minsky has defined the term) for representing algorithmic processes, hopefully the way humans do.

. INTRODUCTION AND MOTIVATION

It is becoming increasingly evident to uman intelligence model builders and heorists that, in order to characterize uman knowledge and belief as computer data tructures and processes, it is necessary to eal with very large, explicitly unified tructures rather than smaller, ununified ragments. The reason for this seems to be hat the original experiences which caused he structures and processes to exist in the irst place come in chunks themselves; nowledge is never gained outside of some ontext, and in gaining some piece of nowledge X in context C, X and C become nseparable. This suggests that it is eaningless to model "a piece of knowledge" ithout regard for the larger structures of hich it is a part. If our goal is to build robot which behaves and perceives in anners similar to a human, this means that he process by which the robot selects a iece of knowledge as being applicable to he planning, executory, inferential or nterpretive process at hand at the moment s a function not only of the specific roblem, but also of the larger context in hich that instance of planning, execution, nference or interpretation occurs. If, for xample, our robot sees his friend with a retched facial expression, the inference he akes about the reasons for his friend's isery will reflect the larger picture of hich he is aware at the time: his friend as just returned from a trip to purchase pera tickets vs. his friend has just eaten he cache of mushrooms collected yesterday s. .... The same pervasiveness of context xists in the realm of the robot's nterpretations of visual perceptions: the ery same object (visible at eye level) will be perceived out of the corner of his eye in one situation as the cylindrical top of his electric coffee grinder (he is at home in his kitchen), but as the flasher of a police car (he is speeding on the freeway) in another. This suggests that at every moment, some fairly large swatch of his knowledge about the world somehow has found its way to the foreground to exert its influence; as our robot moves about, swatches must fade in and out, sometimes coalescing, so that at any moment, just the right one is standing by to help guide acts of planning, inference and perception.

Marvin Minsky has captured this whole idea very neatly in his widely-circulated "Frames" paper [M1]. While this paper describes an overall approach to modeling human memory, inference and beliefs, we still lack any specific formulation of the ingredients which make up the large, explicitly-unified structures which seem to underlie many higher-level human cognitive functions. It is the purpose of this paper to define the notion "commonsense algorithm" (CSA) and to propose the CSA as the basic cognitive structure which underlies the human processes of planning, inference and contextual interpretation of meaning.

I do not have a complete theory yet: the intent of this paper is to record a memory dump of ideas accumulated over the past few months and to show how they can unify my past ideas on inference and memory, as well as the ideas of others.

II. THE SCOPE OF THE CSA'S APPLICABILITY

Most of human knowledge can be classified as either <u>static</u> or <u>dynamic</u>. For example, a person's static knowledge of an automobile tells him its general physical shape, size, position of steering wheel, wheels, engine, seats, etc.; these are the abstract aspects of a car which, although many differ in detail from car to car, are inherently unchanging. They are in essence the physical definition of a car. On the other hand, a person's dynamic knowledge of a car tells him the <u>functions</u> of the various components and how and why to coordinate them when the car is applied to some goal. The static knowledge tells the person where to expect the steering wheel to be when he gets in; the dynamic knowledge tells him how to get in in the first place, and what to do with the wheel (and why) once he is in. For a robot immersed in a highly kinematic world -- physically, psychologically and socially -- a very large part of his beliefs and knowledge must relate to dynamics: how he can effect changes in himself and his world, and how he perceives other robots effecting changes. It is the purpose of the CSA to capture the dynamics of the world in belief structures which are amenable to computer manipulation of plans, inference and contextual interpretation.

It should be stressed that the phrase "dynamics of the world" is intended in its broadest possible sense. As will be elaborated upon in a later section, the phrase is intended to encompass such seemingly diverse robot/human activities as:

1. communicating with another robot/human (e.g., how to transfer information, instill wants, convince, etc.)
2. getting about in the world
3. building things (both physical and mental) and understanding the operation of things already built by others
4. conceiving, designing and implementing computer programs and other commonsense algorithms (a special form of building)
5. interpreting sequences of perceptions (e.g., language utterances) in context
6. making contextually meaningful inferences from perceptions

I am convinced that all such dynamics of the world can and should be expressed in a uniform CSA formalism built around a relatively small number of cognitively primitive ingredients.

III. EVOLUTION OF THE CSA IDEA

The next section will define a CSA as a network-like structure consisting of events tied together by primitive links. Taken as a whole, the CSA specifies a process: how to get something done, how something works, etc. A computer scientist's first reaction to this type of structure is "Oh yes, that's an AND/OR problem-reduction graph" (see Nilsson [N1] for example). Figure 1 shows an AND/OR graph for how to achieve the goal state "a McDonald's hamburger is in P's stomach." Edges with an arc through them specify AND successors of a node (subgoals, all of which achieved imply the parent node has been achieved); edges with no arc through them specify OR successors (subgoals, any one of which being sufficient to achieve the parent goal).

AND/OR graphs have been demonstrated adequate in practice for guiding various aspects of problem-solving behavior in existing robots (see [S1] for example). However, they are intuitively not theoretically adequate structures for representing general knowledge of world dynamics: their principal deficiency is that they are ad-hoc constructions which express neither the implicit conceptual relationships among their components, nor the inherent types of their components. Because of this, there is no constraint on their organization, and this means that two AND/OR graphs which accomplish or model the same thing might bear very little resemblance to one-another when in fact they are conceptually very similar. This may be little more than a nuisance in practice, but it is undesirable in principle because it makes learning, reasoning by analogy, sharing of subgoals, etc. tedious if not impossible in a generalized problem solver.

A refinement of the notion of an AND/OR graph introduces the concepts of <u>causality</u> and <u>enablement</u>, and <u>actions</u> and <u>states</u> (statechanges); edges in the graph are distinguished as either causal or enabling, the nodes are distinguished as either actions or states, and the graph obeys the syntactic contraints:

(a) actions <u>cause</u> states
(b) states <u>enable</u> actions

Bob Abelson [A1] was among the first to employ these historically very old concepts in the framework of a computer model of human belief, and since then, numerous computer-oriented systems of knowledge representation (e.g., Schank's conceptual dependency[S2], Schmidt's models of personal causation [S4]), as well as systems of inference (Rieger [R1], Charniak [C1]) have found these four concepts to be vital to meaning representation and inference. In some sense, enablement, causality, states and actions seem to be cognitive primitives. Figure 2 is a refinement of Figure 1 which makes explicit the nature of each node and each connecting arc, and hence the underlying gross conceptual structure of the algorithm.

While the inclusion of these four concepts (and their resulting syntactic constraints) in the basic paradigm makes for a theoretically more coherent representation, the scheme is still too coarse to capture the kinds of detailed knowledge of algorithms people possess. The following section proposes an extended framework of event types and event connectors based on these four notions and some others. These event types and connectors will be regarded as model-primitives which hopefully are in correspondence with "psychological primitives" in humans.

IV. DEFINITION OF THE COMMONSENSE ALGORITHM

In the new formalism, a CSA consists of nodes of five types:

1. WANTS
2. ACTIONS
3. STATES
4. STATECHANGES
5. TENDENCIES

The first four types are not new (see [S3] for example), and will not be covered here beyond the following brief mention. A WANT is some goal state which is desired by a potential actor. An action is something an (animate) actor does or can do: it is enabled by certain states (certain conditions which must be true in order for the action to begin and/or proceed), and in turn causes other states (discrete) or statechanges (continuous) to occur. Actions are characterized by an actor, a model-primitive action, a time aspect, a location aspect, and a conceptual case framework which is specific to each model-primitive action. States are characterized by an object, an attribute, a ılue and a time aspect; statechanges are ıaracterized by an object, a continuous tate scale (temperature, degree of anger, istance, etc.), a time aspect and beginning ıd end points on the scale.

It is the notion of a <u>tendency</u> which is ɜw and which serves to unify a class of ·oblems which have been continually ⲭperienced in representing processes. ısically, a tendency is an actorless ɔtion. Tendencies are characterized by ɔecifying a set of enabling conditions and set of result states and/or statechanges. ıenever the enabling conditions are ıtisfied, the tendency, by some unspecified ɜans, causes the states and statechanges ɔecified as the tendency's results. Hence, tendency may be regarded as a special type f non-purposive action which <u>must</u> occur ıenever all its enabling conditions are ıtisfied. Contrasting the notion of a ɜndency with the notion of an action yields rather compact definition of what makes a volitional" action volitional: a volitional ɔtion is an action which need not occur ven though all its physical enabling ɔnditions are met. The reason it may not ɔcur is, of course, that the actor does not ɜsire it to occur; tendencies have no such ɜsires.

The abstract notion of a tendency is ɜant to be general-purpose, to characterize wide variety of phenomena which are not ɔtions, but action-like. Examples of ɜndencies are:

1. GRAVITY, PRESSURE, MAGNETISM, ATOMIC-FISSION, HEAT-FLOW, and the host of other physical principles. Commonsense GRAVITY might be captured as follows:**

```
((TYPE . TENDENCY)
 (REFERENCE-NAME . GRAVITY)
 (ENABLEMENTS . (UNSUPPORTED OBJ)
          (LESSP (DISTANCE OBJ EARTH)
                 (ORDERMILES) )
 (RESULTS . (STATECHANGE OBJ VELOCITY X
                 X+d (LOC OBJ)
                 (LOC EARTH)))
```

2. human biological functions: a tendency to GROW-HUNGRY, GROW-SLEEPY, GROW-OLDER (sole enabling condition is the passage of time!), GROW-LARGER, etc. For example:

```
((TYPE . TENDENCY)
 (REFERENCE-NAME . GROW-HUNGRY)
 (ENABLEMENTS . ((NOT (LOC NUTRIENTS
                          STOMACH))
            (DURATION * ORDERHOURS)))
 (RESULTS . (WANT P (INGEST P NUTRIENTS
                 MOUTH STOMACH))))
```

3. human psychological functions: the tendency to GROW-LONELY, the tendency to FORGET, etc. For example:

```
((TYPE . TENDENCY)
 (REFERENCE-NAME . GROW-LONELY)
 (ENABLEMENTS . ((ALONE P)
            (DURATION * ORDERDAYS))
 (RESULTS . (WANT P (COMMUNICATE P
                           X))))
((TYPE . TENDENCY)
 (REFERENCE-NAME . FORGET)
 (ENABLEMENTS . (INHEAD ITEM P)
         ((UNREFERENCED ITEM P)
          (DURATION * ORDER??)))
 (RESULTS . (STATECHANGE ITEM
         REFERENCE-DIFFICULTY X X+d)))
```

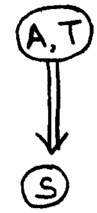
Tendencies, thus characterized, will play an important role in modeling algorithmic processes via CSA's. In fact, adopting the notion of a tendency as a model primitive points out a rather ubiquitous principle: humans spend a large amount of time in planning either how to overcome tendencies which stand in the way of their goals, or how to harness them at the proper times in place of an action (e.g., dropping the large rock on the coconut). Although a tendency's primary use is at the edge of the world model, where things happen simply because "that's the way things are", it will probably be desirable to have the ability to regard as tendencies things which in fact can be explained. Characterizing something as a tendency even though it may be reduceable to further algorithms is probably one tactic a human employs when confronted with the analysis of very complex, olny partially understood processes. Even though something <u>could</u> be further explained, the system of representation should allow that something to be treated as though it were a tendency.

Tendencies have numerous aspects which will require explicit characterization in a computer model. Two such aspects relate to (1) the inherent rapidity with which a tendency exerts itself and (2) the tendency's periodicity, if any. That is, how quickly does a person become hungry (slope of curve), how long does it take to forget something, how rapidly does an object accelerate, how fast does the water flow through the nozzle, etc.? If the tendency is periodic, what are the parameters describing its periodicity? The primitive CSA links described in the next section will serve in part to capture such aspects, but they are not yet adequate.

<u>The CSA primitive Links</u> Using these five event-types as building blocks (WANTS, ACTIONS, STATES, STATECHANGES, TENDENCIES), the goal is to be able to express the dynamics of just about anything, be it a physical device, a psychological tactic employed by one person on another, how a person purchases a McDonald's hamburger, or how a computer program functions or was constructed. There are 25 primitive links in the current formulation. They will only be defined here, leaving justificaion and details of their use for the examples which will follow, and for subsequent papers on the subject. In the following definitions, W, A, S, SC and T will stand for WANT, ACTION, STATE, STATECHANGE and TENDENCY, respectively.

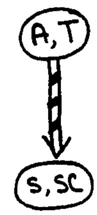
**The LISP notation reflects some concurrent thinking on how a commonsense algorithm system might actually be engineered. A forthcoming report will describe progress toward implementing the ideas in this paper.

TYPE 1: ONE-SHOT CAUSALITY



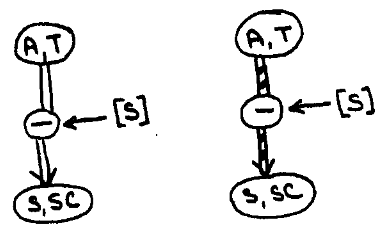
Action A or tendency T causes state S.
The action or tendency need occur only once; thereafter S will persist until altered by another action or tendency. For any given S, there will ordinarily be numerous alternative A´s or T´s in the algorithmic base which would provide the one-shot causality.

TYPE 2: CONTINUOUS CAUSALITY



Action or tendency A,T´s continuing existence continually causes state or statechange S,SC.
Whether one-shot or continuous causality is required to maintain S or SC is both a function of S or SC and its particular environment in a particular algorithm (i.e., what other tendencies and actions are influencing it). Again, there will ordinarily be numerous actions or tendencies in the algorithmic base which could provide continuous causality for any given state or statechange.

TYPES 3,4: GATED ONE-SHOT AND CONTINUOUS CAUSALITY

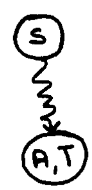


A,T causes S,SC either one-shot or continuously, providing that all states in [S] are satisfied.
The <u>flow</u> of causality cannot occur unless states specified by [S] exist. That is, even though A,T is occuring and there is a potential causal relationship between A,T and S,SC, the relationship will not be realized until the gating states become true.

TYPE 5: ONE-SHOT ENABLEMENT

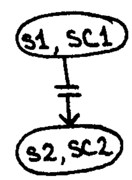
State S´s one-time existence allows action A or tendency T to proceed.
Thereafter, A,T´s continuation is no longer a function of S. A,T will ordinarily have numerous one-shot enablements, in which case, <u>all</u> must be satisfied in order for A or T to proceed.

TYPE 6: CONTINUOUS ENABLEMENT



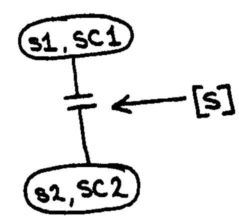
State S´s continued presence is requisite to action A´s or tendency T´s continuance.
S´s removal causes A or T to halt. Any given A or T will ordinarily have numerous continuous enablements, in which case <u>all</u> must reamin true in order for A or T to proceed.

TYPE 7: CAUSAL STATE COUPLING



States S1, S2 or statechangges SC1, SC2 are causally coupled; because of this coupling, changes in S1 or SC1 are synonomous with changes in S2 or SC2. This link provides a way of capturing the relatedness of various aspects of the same situation.

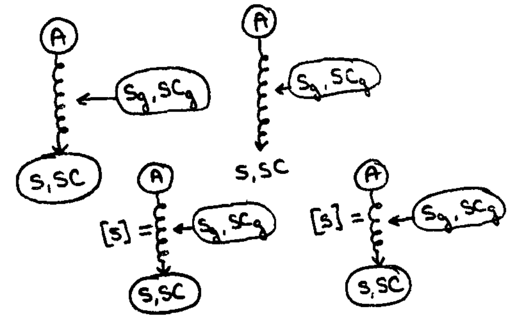
TYPE 8: GATED CAUSAL STATE COUPLING



State S2 or statechange SC2 is synonymous with (causally coupled to) S1 or SC1, provided that all states in [S] are true.

This link is similar to ungated state coupling, except for the existence of factors which could disrupt the coupling. To illustrate, the flow of a fluid into a container (a statechange in location of the water) is synonymous with an increase in the amount of water in the container (another statechange), but only providing that there is no source of exit from the container's bottom.

YPE 9,10,11,12: ByPRODUCT (ONE-SHOT/CONTINUOUS, GATED/NON-GATED)



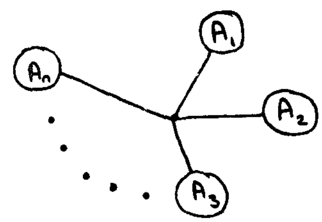
State S or statechange SC is a causal byproduct of action A, relative to goal state Sg or SCg.

That is, the actor of A, wishing to achieve state Sg or statechange SCg also produces state S or statechange SC. The byproduct link is truly a causal link; what is and is not a by product must obviously relate to the motive of the actor in performing the action. Where gated, all states in [S] must be satisfied in order for the byproduct to occur.

YPE 13: ORIGINAL INTENT

Want W is the original desire (goal state) of an actor. W is external to the CSA in that its origin is not explicable within the CSA itself; it is the outside directive which motivated the invocation of some acton. Within an algorithm for achieving some goal, motivations are explicable: every subaction is, by its nature, designed to produce subgoal states which, taken together, meet the original intent.

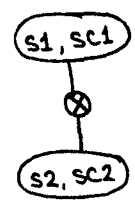
YPE 14: ACTION CONCURRENCY



Actions A1,...,An must be concurrently executed.

This link will arise in the dynamics of an actual plan, rather than be stored originally in the algorithmic base explicitly. As plans evolve and the actor learns concurrency by rote, the link will begin to appear in the algorithmic base as well. Action concurrency is nearly always caused by multiple enabling states for some other action, all of which must be continually present, or one-time synchronized as a collection of one-shot enablements.

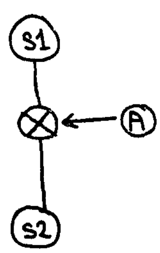
YPE 15: DYNAMIC ANTAGONISM



State S1 or statechange SC1 is antagonistic to state S2 or statechange SC2 along some dimension.

This link relates two states or statechanges which are opposites in some sense; typically the antagonism link will make explicit the final link in some sort of feedback cycle in an algorithm. The link is hard to describe outside the context of an example; examples will appear in the next section.

YPE 16: MOTIVATING DYNAMIC ANTAGONISM

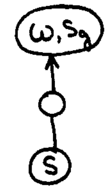


As with ordinary dynamic antagonism, S1, S2 are antagonistic states. Typically, S2 is required as an enabling state (continuous) for some action, but that action, or some other action, produces S1 as a byproduct; this gives rise to the need for another corrective action A which can suppress the byproduct, therby preserving the original required enablement. This link is intended to capture the execution dynamics of a situation in which antagonistic states are expected to arise. That is, it will provide a representation wherein antagonisms can be anticipated in advance of the SCA's actual execution. An example of motivating dynamic antagonism is included in the next section.

TYPE 17: GOAL-REALIZATION COUPLING

State S is an alternative way of expressing original goal W or subgoal Sg.

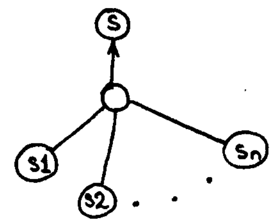
This link supplies a way of specifying termination criteria for CSA's involving repretition. Its use is illustrated in one of the examples.



TYPE 18: COMPOUND GOAL STATE DEFINITION

State S is a shorthand for expresing the set of goal states S1,...,Sn.

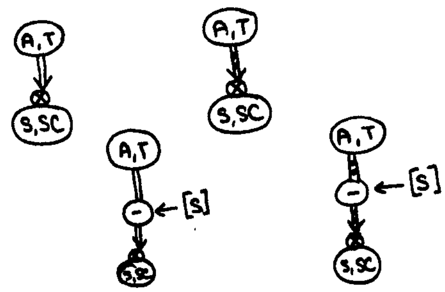
This link allows a "situation" to be characterized as a collection of goal states. When all goal states are satisfied, the situation is satisfied. An example of a compound goal state would be: "get the kids ready for the car trip", where this means a set of things rather than one thing.



TYPES 19,20,21,22: DISENABLEMENT (ONE-SHOT/CONTINUOUS, GATED/NON-GATED)

Action A or tendency T one-shot/continually causes state S or statechange SC not to exist.

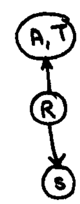
These four forms are shorthands for causality in conjunction with antagonism. They will be principally useful for representing acts of disenabling unwanted tendencies.



TYPE 23: REPETITION UNTIL THRESHOLD

Action A or tendency T occurs repeatedly until state S becomes true.

This link provides for the repeated application of an action or tendency. Normally, the action or tendency will, directly or indirectly, causally produce a statechange along some scale; this statechange will eventually threshold at state S.



TYPE 24: INDUCEMENT

State S's or statechange SC's existence induces want W in a potential actor.

Origins of wants can be explicitly represented via this link. Typically, W will be a state which is antagonistic to S or SC. For example, if the temperature is too high in the room, the want is that the temperature become lower; if the tendency, PRESSURE, has been enabled, allowing blood to flow out of P's body, the induced want is that this tendency be disenabled, and hence that the antagonism of one of PRESSURE's enabling states start to exist.

TYPE 25: OPTIMIZATION MARKER

State S is an enabling condition for action A, and this relationship makes possible an optimization during the execution of A in a particular environment.

When several actions arise in a plan, they may share enabling states. This means that when the plans are considered together, some of the states needed for one action may coincide with those needed for another. The optimization marker allows this phenomenon to be recorded. Its interpretation is: when state S becomes true, consider performing acton A, because action A also has S as an enabling state. ¢ denotes a savings.

These are the commonsense algorithm primitive links. It is felt that they are conceptually independent enough of one-another so that unique algorithms will be forced into unique, or at least similar, representations under this formalism. Although it is the eventual intent of the theory to be able to capture all the nuances of intentional human problem-solving behavior, there is no real feeling yet for the completeness of this set of links in this regard; all that can be said now is that they do seem to suggest a reasonable approach to representing large classes of purposive human behavior. The adequacy of these primitives for representing devices and mechanisms, on the other hand, is easier to see, at least intuitively; the links seem to be adequate for some fairly complex "purposive" mechanisms. Accordingly, the first example of their use will be to characterize a mechanism very dear to most of us.

V. EXAMPLES OF COMMONSENSE ALGORITHMS

EXAMPLE 1. Operation of a reverse-trap toilet [Figure 3]

As a first test of the theory, the reverse-trap toilet is a relatively demanding mechanism. It is a complex feedback mechanism which is the product of some rather sophisticated human problem-solving. It is therefore interesting both in its own right and as a tangible manifestation of human-concocted causality and enablement. By one simple action, a complex sequence of tendencies is unleasehed; the sequence not only stops itself, but restores the system to its initial state, and does something useful in the process.

The English description of the schematic of Figure 4 is as follows: The trip handle is pushed down, one-shot causing the flush-ball to be raised; this one-shot enables the tendency to float, in turn continually causing the float ball to remain raised. The float ball's being raised is synonomous with the flush valve being open, and this openness continuously enables the tendency of gravity to move water from the tank to the bowl beneath (as long as water remains in the tank, of course.) This movement of water is synonomous with two other state changes: a decrease of water height in the tank, and an increase of water height in the bowl. The increase of bowl water height thresholds when the water reaches waste channel lip level, at which time it begins providing continuous enablement for gravity to move the water into the waste channel; this movement thresholds when the channel fills, providing the beginning of continuous enablement of the tendency capillary action. This tendency, in turn, sustains the flow of water from the bowl to the waste channel, continually moving waste water into the sewer. This action ceases when the bowl becomes empty. Meanwhile, the tendency gravity is continually moving water from the tank to the bowl. This is synonomous with a decrease in tank water height, and this decrease thresholds at point X, synonomous with the fresh water supply valve opening. This opening enables the tendency pressure to move water from the fresh water line into the tank; this is synonomous with an increase in tank water height, but only providing that the flush valve is closed (this will have to wait for the movement of waste from tank to bowl to cease). When the tank water height finally begins its increase, this increase will threshold at point X again, this time being synchronous with the ball cock supply valve's being closed, stopping the fresh water and hence the tank water height increase. At this point, the system has become quiescent again. (Note: in the actual simulation which will be performed, flow rates, or more generally, rates of statechanges, will be incorporated.)

EXAMPLE 2. Sawing a board in half to decrease its length (Figure 5)

Figure 5 is a bare-bones representation of a purposive human process: sawing a board in two using a handsaw. This CSA illustrates the concepts of motivating dynamic antagonsim, original intent and byproduct with respect to a goal. The schematic of Figure 5 is only a fragment of the larger algorithm; many enabling states and byproducts, as well as their compensatory actions have been omitted. In this CSA, the act of sawing for the purpose of decreasing the board's length produces, among others, the byproduct of the board's moving. Since a stationary board is a gate condition on the flow of causality from the sawing action to the statechange in cut depth, the two states joined by the motivating dynamic antagonsim link form an antagonistic pair, indicating in advance of actual execution that it will be necessary to perform a compensatory action: hold the wood down. If we were to illustrate more of this algorithm, it might be found that holding the wood down would require more hands than were available. This would provide another dynamic antagonsim which would motivate the engagement of another compensatory action, such as "call for help," "go to a vise," etc.

It should again be pointed out that points of antagonism could alternatively be detected at the execution time of the CSA and compensatory solutions dynamically fabricated. This would likely occur via some sort of interrupt mechanism. But the antagonsim link allows for planning ahead (e.g. when two arbitrary algorithms are selected to accomplish a task, their coexistence will probably not always be without antagonism -- this allows the planning mechanism to anticipate and solve such antagonisms before execution). Also, after a successful plan involving antagonisms has actually been executed, this link provides a means of recording once and for all the compensating actions which were performed.

EXAMPLE 3. Vicious cycles (Figure 6)

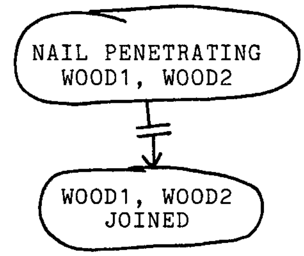
Consider tendencies such as fire and forgetfulness. Both roughly follow the paradigm: a tendency has state S as a continuous enablement, and produces the same state as continuous causality. Once started, such a system is self-sustaining. In the case of fire, a one-shot causing action causes a statechange in temperature which thresholds at the point of the material´s combustion temperature; this enables the tendency to burn, which in turn produces as a continual byproduct heat, causing a vicious cycle. In forgetting, the tendency to forget X is enabled by not referencing X for periods of time; but as X grows more forgotten, it becomes less referenceable. Here, dynamic antagonism lies at the root of the vicious cycle.

EXAMPLE 4. Description (synthesis) of a computer algorithm (Figure 7)

Suppose the goal is to compute the average of a table of numbers, TABLE(1),...,TABLE(n). Figure 7 shows both how to conceive of the algorithm and how the algorithm will actually run. As a computer algorithm, this is not as fully explicit as might be desired: it lacks explicit iteration and explicit termination criterion testing. These will have to be worked out before the theory adequately handles repetition.

Causal gating seems to play a central role in this sort of computer algorithm. Intuitively, this is the case because, though a computer instruction typically has no physical enabling conditions (it could be issued at any time), desired effects can be achieved only by tying the syntax of instruction causality to the semantics of logical causality. For example, the flow of causality from the action "fetch location SUM to AC1" to the logical semantic state "partial sum in AC1" can take place only if location SUM logically contains the actual partial sum at that point! Otherwise, garbage is fetched.

The relationships of certain types of causal gating and state coupling (e.g. the valve closing because the float has risen in the toilet tank) are not completely apparent yet. Perhaps state coupling is a shorthand for an implicit sequence of gated causalities between two statechanges. On the other hand, state coupling between two states, as opposed to statechanges, seems to be a concept which is independent of gated causality. To illustrate; "a nail through two pieces of wood" (state 1) has to be regarded as state-coupled to "the pieces of wood are joined" (state 2, a description of the same situation, but at a different level):

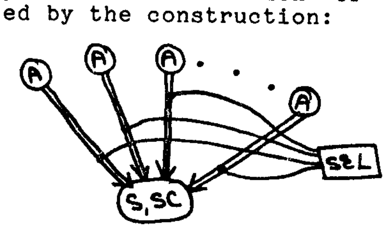


In this type of situation, the state coupling concept is required at this level to stop the representation of some sort of inexplicable "micro-causality" when it transcends the model´s knowledge of the world.

VI. ALTERNATIVE ACTION SELECTION

In looking at devices and simple processes such as sawing a board in half, there have been few choices; the causality and enablement are in a sense already built in or strongly prescribed. In a real planning environment on the other hand, there will ordinarily be numerous alternative actions which could causally produce some desired goal state, providing all gating conditions could be met. For example, if the goal of a planner is to produce a statechange in his location to some specified point, the various subplans of walking, driving a car, hitching a ride, bicycling, taking a plane, etc. all suggest themselves as potentially relevant, some more than others. The one the planner actually selects will be a function of more than just the relative costs of each alternative; the selection will also relate to the inherent applicability, or reasonableness of the plan, based on the specifics of where his destination is relative to his current location, weather conditions, etc. Of course, all the relevant factors could eventually be discovered by simulating each alternative plan before choosing, watching out for undesirable or suboptimal events. For example, in simulating the walking, hitchhiking or bicycling plans, the planner finds himself outside for potentially long durations. Hence, if it is raining, the cost is judged high. If the distance is less than a mile, or is indoors, simulation of the airplane plan leads to some absurdly high costs and perhaps some unsolvable antagonisms. Certainly, a degree of such forward simulation must occur in planning; however, it seems that the process of selecting among alternative actions is, intuitively, more unified than just a collection of forward simulations.

For this reason, the model of CSA´s incorporates the notion of a selector, denoted by the construction:



SEL is a place where heuristics, as well as forward simulation control can reside. The heuristics test relevant dimensions (e.g. distance, weather conditions, etc.) of the context in which the state or statechange is being sought (either for execution of some larger plan, or for interpreting what another might do in some context). Based on the outcomes of such tests, the SELector chooses one alternative action as most reasonable. Currently, the selector function is imagined to exist "outside" the CAS formalism as an unrestricted program which runs and decides. Eventually, since it is one goal of the CSA formalism to be able to represent arbitrary decision processes (these are, after all, just other algorithms), the SELector function should simply reference other CSA's which carry out the heuristic testing. In other words, defer the "intelligence" in selecting an alternative at this level to unintelligent CSA's at the next level, and so on.

VII. LEVELS OF RESOLUTIONS IN CSA'S

The algorithmic content of a CSA can be described at many different levels of resolution. For example, the "action" "take a plane to San Francisco" is quite a bit higher in level and more abstract than the action "grasp a saw". In the former, the act of taking a plane somewhere is not really an action at all, but rather a description of an entire set of actions, themselves related in a CSA; "take a plane to San Francisco" is a high level surrogate for a low level collection of true actions in the sense of actually performing physical movements, etc. in the real world (things like grasping a saw, reaching into pants pocket for some money, and so on).

Another example of resolution level differences relates to how enabling states for actions are characterized. For example, in (A2) Abelson employs the primitive (OKFOR object application), as in (OKFOR AUTO TRAVEL). The question here is, what is the relationship between this high level description of OKness and the <u>specifics</u> of what OKFOR means for any given object? That is, for a car, OKFOR means "gas in tank", "tires inflated", "battery charged",..., whereas (OKFOR TOILET FLUSHING) means quite a different set of things. The basic issue is: should the memory plan and interpret in the abstract realm of OKFORedness, then instantiate with details later, or must the details serve as the primary planning basis, with the abstract ideas being reserved for other higher level processes such as reasoning by analogy, generalization and so forth? There is probably no cut-and-dried answer; however, the tendency in a CSA system would be to favor the details over the abstract. But the CSA representation is intended to be flexible enough to accomodate both the abstract and the concrete. The idea of state coupling is an illustration of this.

VIII. THE THEORY HAS ONLY JUST BEGUN

A later version of this paper will contain more examples of the CSA, including its use in language context problems. The theory is by no means complete; to illustrate:

(1) Is there such a thing as <u>gated</u> enablement? The answer seems to be "yes", since it seems reasonable to regard enablement as a flow which can be cut off in much the same way as causality. Perhaps an example of gated enablement is when the horses begin their race at the racetrack: the start gate's being open is a one-shot enablement for the horse to run, but only if the horse is in the box to start with! If he's not in the box, the gate's position isn't relevant as an enablement to run; its flow is severed.

(2) What kinds of time and sequencing information need to be incorporated in the formalism? For example, causality can be either abrupt or gradual: taking medicine for an ulcer provides a conceptually gradual statechange in the stomach's condition, whereas surgery provides a conceptually abrupt cure! This suggests the need for classifying statechanges on some discrete conceptual scale. Another inadequacy of the present model is its inability to specify time sequencing; adoption of some traditional flowchart concepts will probably prove adequate for this.

(3) There is no convenient way to model decision-making processes on the part of the planner of a CSA. This will have to be developed.

IX. APPLICATIONS OF THE CSA

On the brighter side, the CSA provides a unified basis for problem-solving-related cognitive models. Specifically, I believe it shores up, under one basic data structure, the ideas presented in my own past research in conceptual memory and inference (R1,R2) and in conceptual overlays (R3) which suggests a meaning context mechanism for language comprehension based around CSA's. I want to conclude by listing anticipated applicaions of CSA's. The applications have been divided into two categories: general (those which are central to some major theoretical issues in language understanding and problem-solving), and specific (those which provide some local insights into memory organization).

<u>General Applications</u>

1. <u>As the basis for active problem-solving</u>

The CSA supplies an algorithmic format wherein plans can be conceived, synthesized and executed. One immediate goal of research should be to construct a commonsense algorithm interpreter which could "execute" the contents of portions of its own CSA memory in order to effect actions of moving about, communicating, and so forth.

2. As the basis for conceptual inference

In (R1), which describes a theory of conceptual memory and inference, sixteen classes of conceptual inference were identified as the logical foundation of a language-based meaning comprehension system. Interestingly enough (but not surprising), nine of those inference classes correspond directly to traversals of CAS primitive links. In the theory of (R1), every language stimulus, represented in conceptual form via Schank's conceptual dependency notation (S2), was subjected to a spontaneous expansion in "inference space" along the sixteen dimensions corresponding to the sixteen inference classes. Making an inference in that model corresponds to identifying each perception as a step in one or more CSA's, then expanding outward from those points along the CSA links breadth-first. Although there is certainly a class of more goal-directed conceptual inference, this kind of spontaneous expansion seems necessary to general comprehension, and the CSA is a natural formalism to use. The nine classes of inference which relate directly to CSA links are:

1. causative
2. resultative
3. motivational
4. enablement
5. function
6. enablement-prediction
7. missing enablement
8. intervention
9. action-prediction

3. As the basis for the conceptual representation of language.

A very large percentage of what people communicate deals with algorithms, the how and why of their activities in the world. Schank's conceptual dependency framework does a good job at representing rather complex utterances which reference underlying actions, states and statechanges. This theory of CSA's extends this framework to accomodate larger chunks of experience and language to begin dealing with paragraphs and stories instead of isolated sentences.

4. As the basis for modeling mechanisms

Every man-made mechanism, as well as every naturally-occurring biological system, is rich in algorithmic content. As illustrated in a previous example, CSA's can do a respectable job at characterizing complex servo- and feedback mechanisms. It is not hard to envision the CSA as a basis for physiological models in such an application as medical diagnosis. Since all biological systems are purposively constructed mechanisms in the evolutionary sense, representing such mechanisms in terms of causality, enablement, byproducts, thresholds, etc. is quite meaningful.

5. As a basis for modeling dynamic meaning context in language comprehension and general perception

(R3) describes an expectancy-based system called "conceptual overlays" which can impose high-level, contextual interpretations on sentences by consulting its algorithmic base. In that paradigm, some stimuli (i.e. meaning graphs resulting from a conceptual parser which receives language utterances as input) activate action overlays, while other stimuli fit into previously activated action overlays. Since an overlay is a collection of pointers to CSA's in the algorithmic base which have been predicted as likely to occur next, to "fit into" is to identify subsequent input as steps in the various algorithms actors have been predicted to engage. For example, knowing what the sentence "John asked Mary for the keys" means contextually is quite a bit more simply understanding the "picture" this utterance elicits (its conceptual dependency representation). If we know that John was hungry:

> John hadn't eaten in days.
> John asked Mary for the car keys.

we activate an overlay which expects that John will engage CSA's which will alleviate his inferred hunger; needing car keys fits nicely as a continuous enablement in several of these algorithms. Of course, the virtue of such a system is that it allows the high-level interpretation of a sentence to change as a function of its contextual environment:

> John had some hamburger stuck
> in his teeth.
> John asked Mary for the car keys.

Change the expectancies, and the interpretation changes!

6. As the basis for characterizing computer algorithm synthesis and operation

Since a computer algorithm is a relatively direct reflection of a programmer's internal model of an algorithmic process, it seems reasonable that both the processes of synthesis and final implementation be represented in the same terms as his internal model. The present theory only suggests an approach; it is not yet adequate for general computer algorithms. But it seems that the idea of a CSA might be very relevant to recent research in the area of automatic programming, at least as a basis of representation.

7. As a basis of a self-model

If a CSA interpreter can indeed be defined, and if indeed the CSA can eventually capture any computer algorithm, then creating a self-model amounts to specifying the CSA interpreter in terms of CSA´s. For example, an act of communication amounts to the communication of enough referential information (features of objects, times, etc.) to enable the comprehender to identify, in his own model, the concepts being communicated. The how-to-communicate algorithm which the CSA interpreter employs could itself be a CSA.

8. As a basis for investigation of algorithm learning

If we posit the existence of a small set of primitive CSA links and make the assumption that these are either part of the brain´s hardware, or learned implicitly as soon as the intellect begins perceiving, we have a basis from which to study how a child learns world dynamics. For example, how, and at what point, does the toddler know that he must grasp the cup in an act of continuous enablement before he can lift it to his mouth, and how does he know it must be at his mouth before he can successfully drink? Perhaps algorithmic knowledge develops from random experimentation within the syntactic constraints imposed by the set of CSA primitive links.

Specific CSA Applications

1. For representing the functions of objects

As with mechanisms, any man-made object is made for a purpose. Translated to CSA´s, this means that part of every purposively-constructed object´s definition is a set of pointers into the algorithm base to CSA´s in which the object occurs. This is true for all objects from pencils, to furnaces, to window shades, to a bauble which provided its constructor amusement, to newspapers. An object in memory can be completely characterized (in the abstract) by a set of intrinsic features (shape, size, color, etc) and this set of pointers to CSA´s.

2. For representing people´s professions

To say (ISA JOHN1 PLUMBER) skirts what it means to be a plumber. Rather, to be a plumber means to engage plumbing algorithms as a principal source of income. Thus, a profession can be defined by a set of pointers to the CSA´s which are characteristic of that profession. This makes it possible to observe someone at work and identify his profession, to compare professions, etc.; these would not be possible if CSA´s were not the basis of representation.

3. For detecting and explaining anomalous situations and potentially antagonistic states

A person notices a license plate yearly sticker on upside down; a person notices two fire engines approaching an intersection, rushing to a fire; at the intersection, one turns left, the other turns right; a person notices that the rain that morning will interfere with the picnic plans that afternoon. How do such situations get judged "anomalous", and how does the perceiver try to explain or cope with them? The answer undoubtedly relates to expectancies and a knowledge of algorithms for putting things on one-another, getting somewhere in a hurry and antagonistic states when eating outdoors. By playing experience against CSA´s we discover things which would not otherwise be discovered.

4. For filling in missing information

If a person is perceiving in a noisy or incomplete environment, having CSA´s available to guide his interpretations of perceptions provides enough momentum to fill in missing details, scarcely noticing their absence. If John is hammering a nail into the wall with his hand on the backswing, but the object in his hand is occluded, it requires very little effort to surmise that it is a hammer. If we believe that Mary is going to McDonald´s to buy a hamburger, but she comes back into the house saying "It won´t start", we have a pretty good idea "it" refers to the car. This application of CSA´s corresponds to the notion of a specification inference in (R1).

X. CONCLUSIONS

Instead of a conclusion, I will simply state the order in which research along CSA lines should, and hopefully will at the University of Maryland, progress:

1. Reimplementation of the conceptual overlays prototype system described in (R3) to reflect the new CSA ideas and replace the ad-hoc AND/OR graph approach described in that report.

2. Implementation of a mechanism simulator which could accept, in CSA terms, the definition of a complex mechanism (electronic circuit or toilet), simulate it, respond to artificially-induced malfunctions, and answer questions about the mechanism´s cause and effect structure.

3. Engineering of a new total conceptual memory, along the lines of the original one of (R1), but incorporatng CSA´s and the new idea of a tendency. This would involve reimplementing the inference mechanism and various searchers.

4. Development of a CSA interpreter which could not only use CSA´s as data structures in the various cognitive processes, but also could execute them to drive itself.

5. Applying CSA´s to medical diagnosis and automatic programming.

6. Investigating the problem of story comprehension via conceptual overlays and CSA´s. Perhaps also investigating generation of stories (e.g. the story of the trip to McDonald´s) or the generation of a description of a complex electronic circuit, encoded as a CSA, in layman´s terms.

XI. ACKNOWLEDGEMENTS

My thanks to the members of the Commonsense Algorithm Study Group at the University of Maryland: Bob Eberlein, Milt Grinberg, Bob Kirby, Phil London and Tom Skillman. They have provided considerable intellectual stimulation. We hope to continue as a group and eventually issue a working paper and computer system.

REFERENCES

(A1) Abelson, R., "The Structure of Belief Systems," in Schank and Colby (eds.), Computer Models of Thought and Language, W.H. Freeman, 1973

(A2) Abelson, R., "Frames for Understanding Social Actions," Paper for Carbonell Conference, Pajarro Dunes CA, May 1974.

(C1) Charniak, E., "Toward a Model of Children's Story Comprehension," Doctoral dissertation, M.I.T., AI TR-266, 1972.

(M1) Minsky, M., "A Framework for Representing Knowledge," M.I.T. AI TR-306, 1974.

(N1) Nilsson, N., Probelm Solving Methods in Artificial Intelligence, McGraw Hill, 1971.

(R1) Rieger, C., "Conceptual Memory: A Theory and Computer Program for Processing the Meaning Content of Natural Language Utterances," Doctoral dissertation, Stanford Univ. AI Memo 233, 1974.

(R2) Rieger, C., "Understanding by Conceptual Inference," American Journal of Computational Linguistics (in press), 1975. (Also available as Univ. of Maryland Technical Report #353)

(R3) Rieger, C., "Conceptual Overlays: A Mechanism for the Interpretation of Sentence Meaning in Context," to appear in Proceedings 4IJCAI 1975. (Also available as Univ. of Maryland Technical Report #354)

(S1) Sacerdoti, E., "Planning in a Hierarchy of Abstraction Spaces," in Proceedings 3IJCAI 1973.

(S2) Schank, R., "Identifications of Conceptualizations Underlying Natural Language", in Schank and Colby, Computer Models of Thought and Language, W.H. Freeman, 1973.

(S3) Schank, R., Goldman, N., Rieger, C., and Riesbeck, C., "Primitive Concepts Underlying Verbs of Thought," Stanford Univ. AI Memo 162, 1972.

(S4) Schmidt, C., and D'Addamio, J., "A Model of the Common-Sense Theory of Intention and Personal Causation," Proceedings 3IJCAI, 1973.

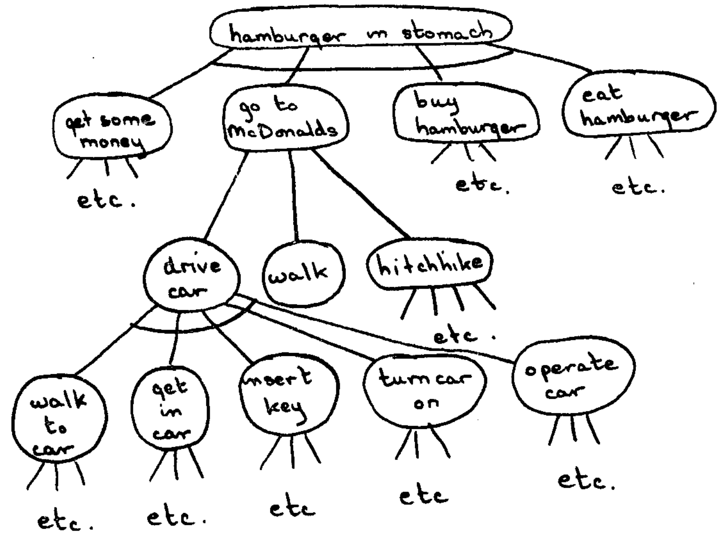


Figure 1

Unrestricted AND/OR graph for getting a McDonald's hamburger into stomach.

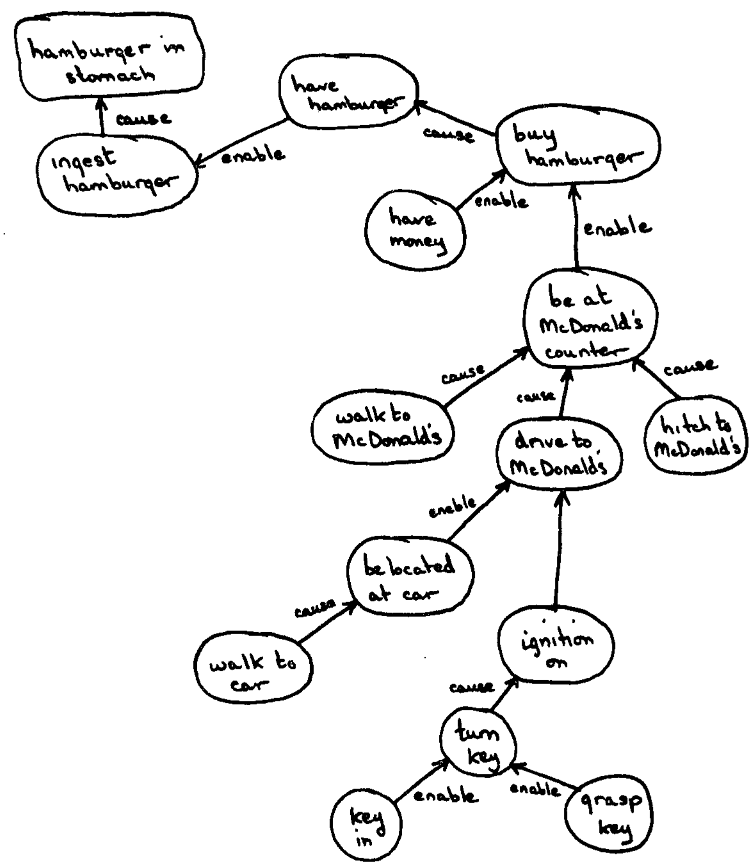


Figure 2

Hamburger algorithm, with actions, states, causality and enablement explicit.

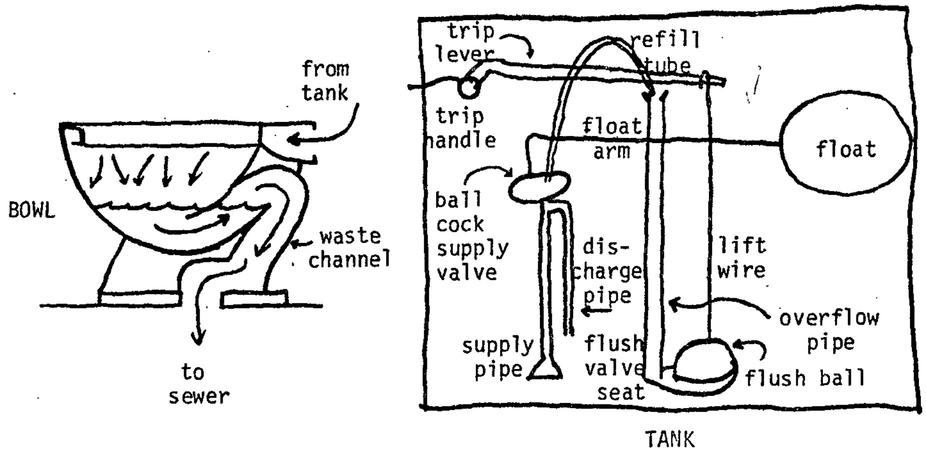


FIGURE 3

A reverse-trap toilet.

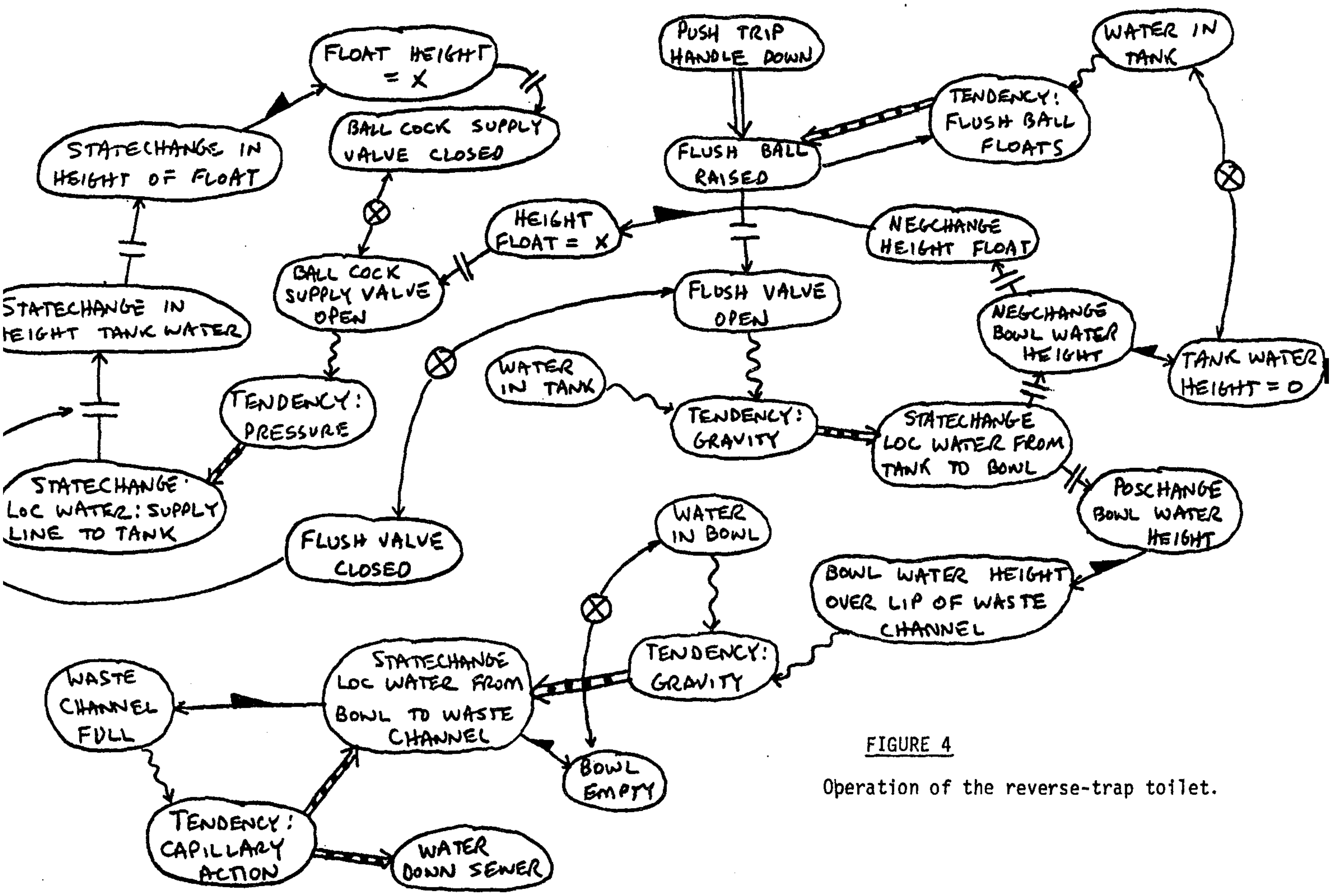


FIGURE 4

Operation of the reverse-trap toilet.

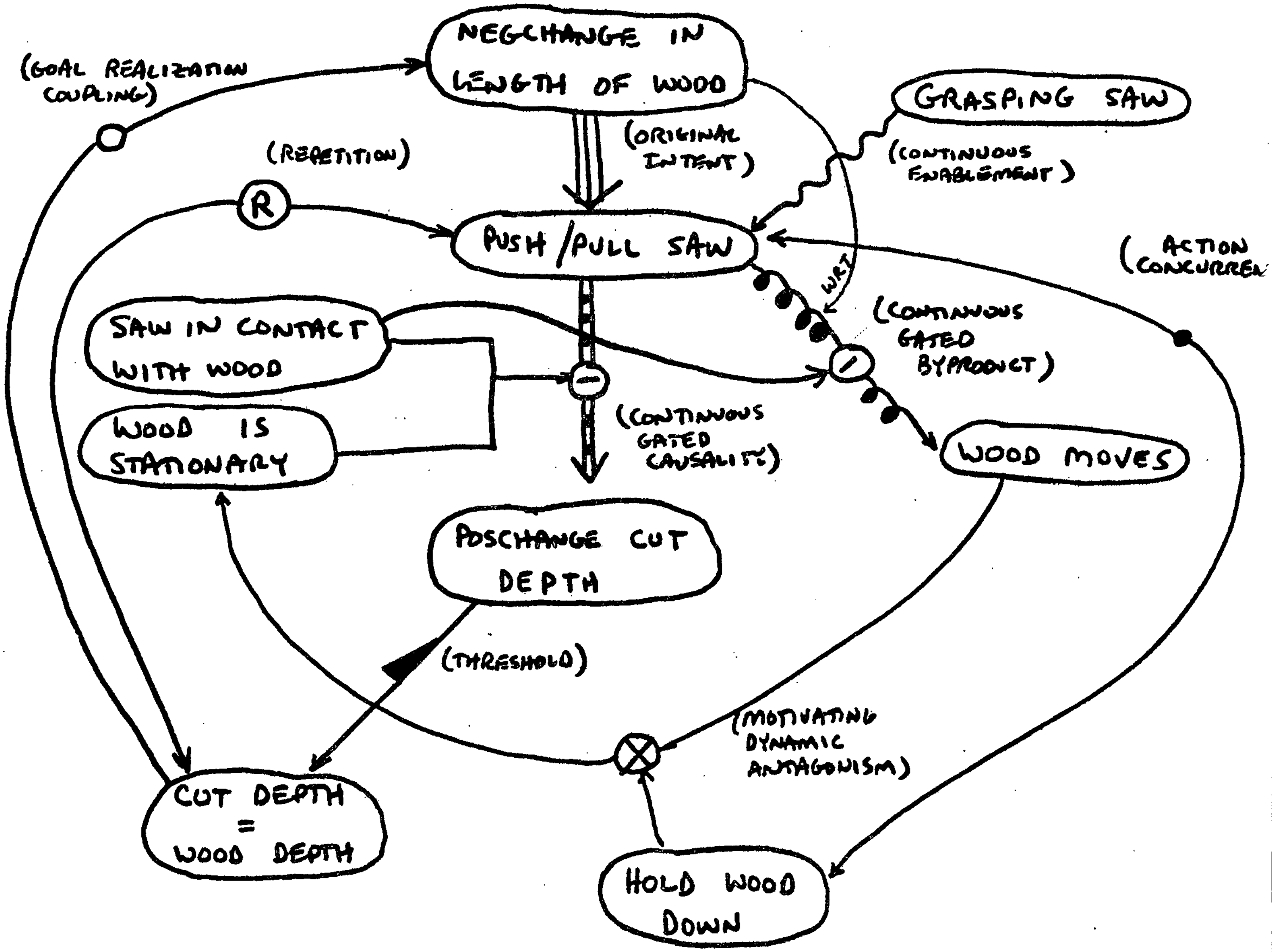
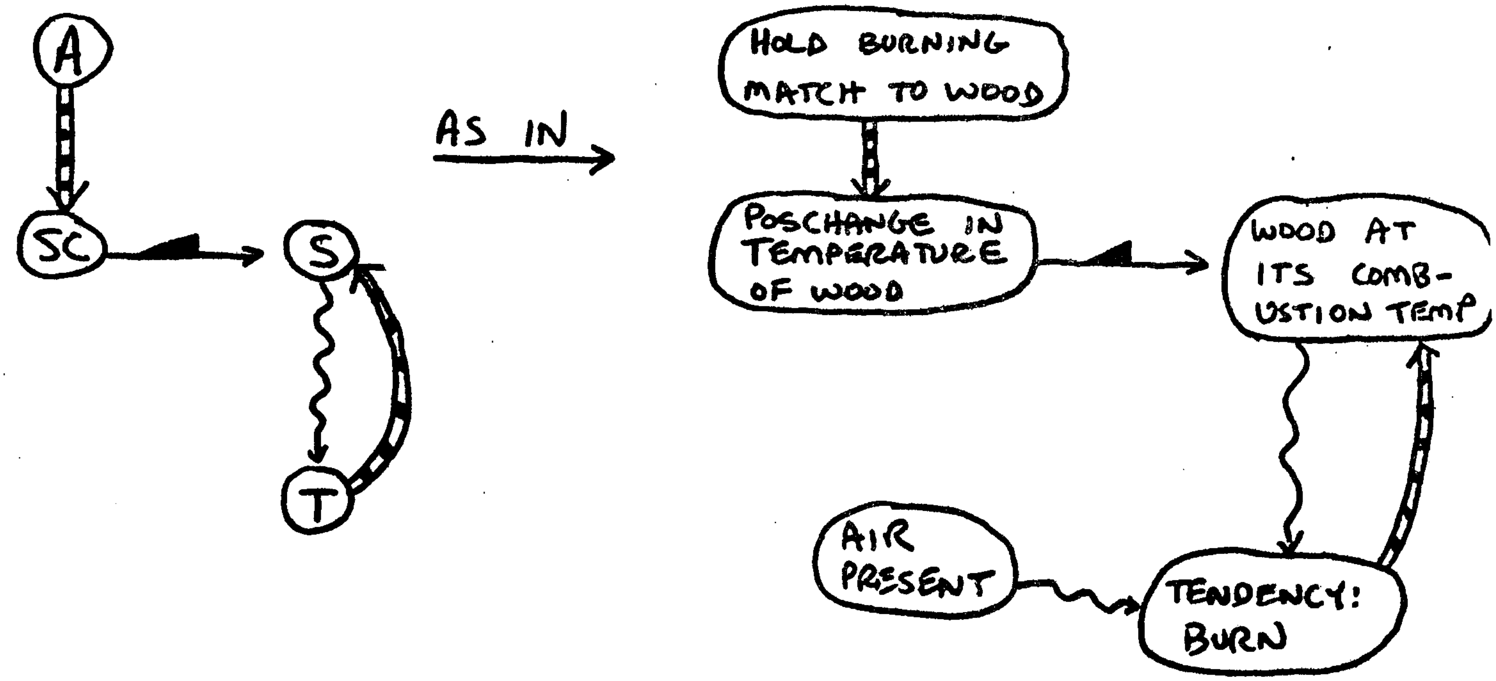
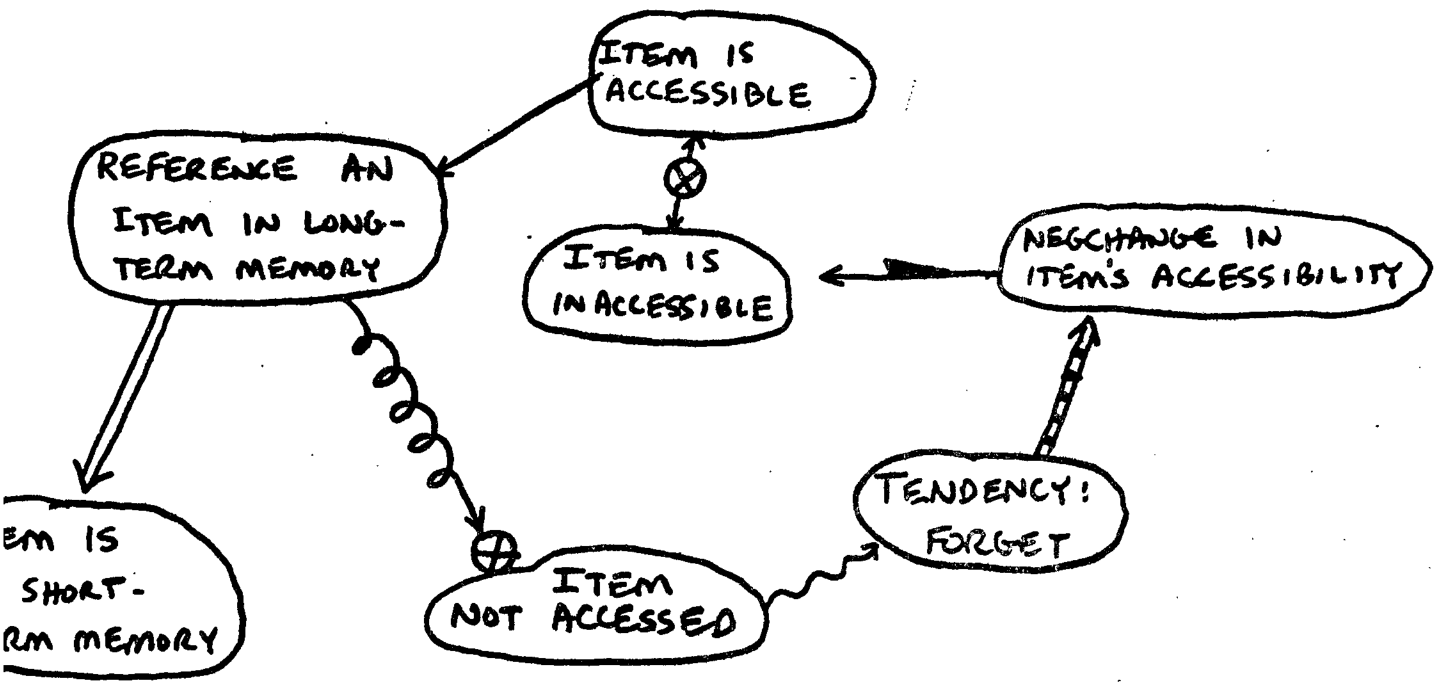


Figure 5

Sawing a board in half to decrease its length.



COMBUSTION



FORGETFULNESS

FIGURE 6
Vicious Cycles.

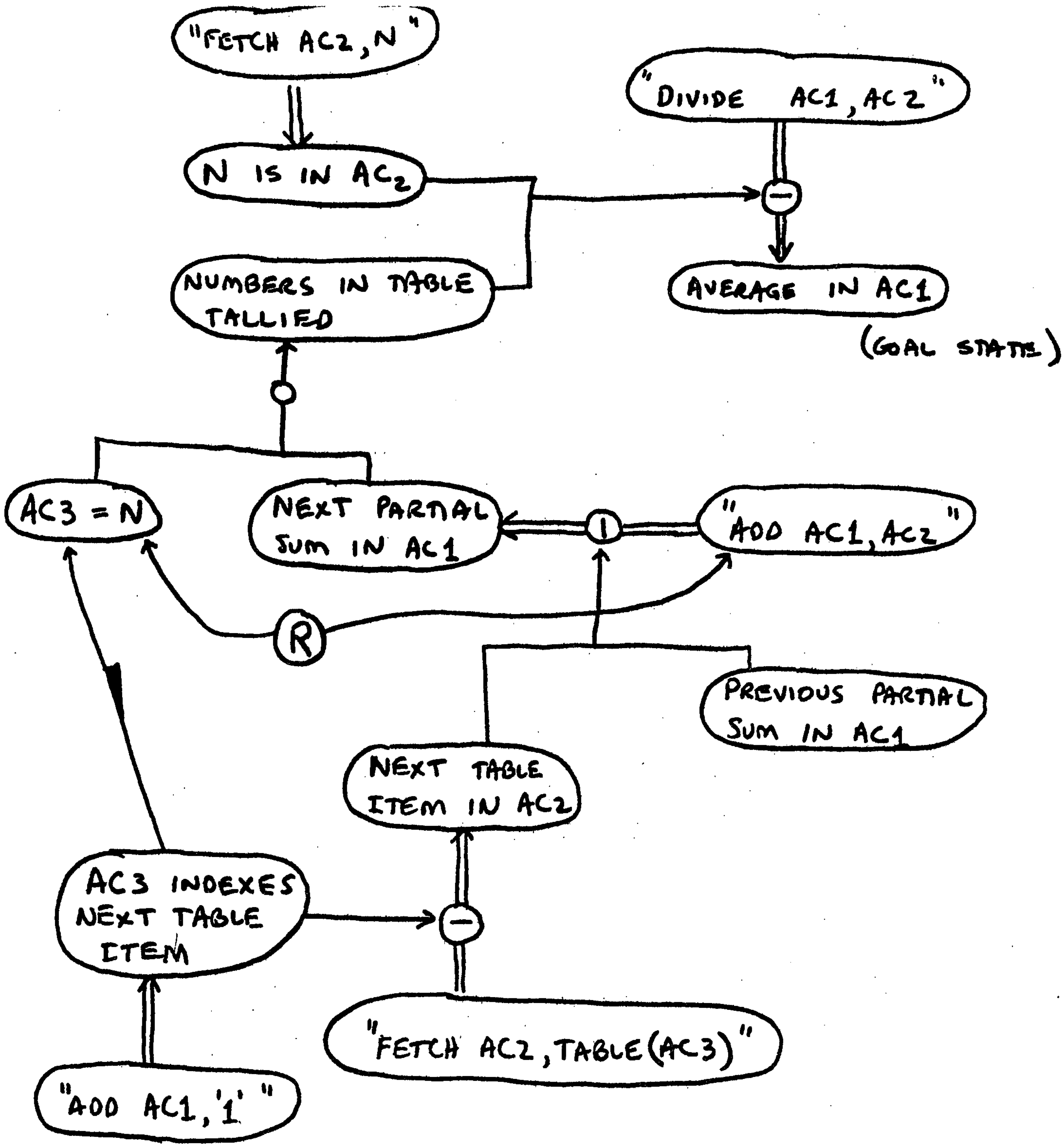


FIGURE 7

Computer algorithm to compute the average of TABLE(1),...,TABLE(N) expressed as a commonsense algorithm.

(NOTE: Initialization has not been shown. The assumptions are that AC3 begins with zero, that AC1 begins with zero, and that N and TABLE(1),...,TABLE(N) exist in core.)